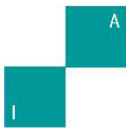
Inteligencia Artificial, 21(62) (2018), 1-11
doi: 10.4114/intartif.vol21iss62pp1-11



INTELIGENCIA ARTIFICIAL

http://journal.iberamia.org/

Sentiment polarity classification of tweets using an extended dictionary

Vladimir Vargas-Calderón vvargasc@unal.edu.co[1], Nelson A. Vargas Sánchez nelsona.vargass@konradlorenz.edu.co[2,3], Liliana Calderón-Benavides mcalderon@unab.edu.co[2], and Jorge E. Camargo jorgee.camargom@konradlorenz.edu.co[3]

[1]Universidad Nacional de Colombia
[2]Universidad Autónoma de Bucaramanga
[3]Fundación Universitaria Konrad Lorenz

Abstract With the purpose of classifying text based on its sentiment polarity (positive or negative), we proposed an extension of a 68,000 tweets corpus through the inclusion of word definitions from a dictionary of the Real Academia Española de la Lengua (RAE). A set of 28,000 combinations of 6 Word2Vec and support vector machine parameters were considered in order to evaluate how positively would affect the inclusion of a RAE's dictionary definitions classification performance. We found that such a corpus extension significantly improve the classification accuracy. Therefore, we conclude that the inclusion of a RAE's dictionary increases the semantic relations learned by Word2Vec allowing a better classification accuracy.
Resumen Con el propósito de clasificar texto basado en su polaridad de sentimiento (positivo o negativo), en este artículo se propone una extensión de un corpus de 68.000 tweets mediante la inclusión de palabras pertenecientes a definiciones de un diccionario de la Real Academia Española de la Lengua (RAE). Un cojunto de 28.000 combinaciones de 6 parámetros de Word2vec y Máquinas de Soporte Vectorial fueron considerados para evaluar qué tan positivamente se afecta el desempeño de la clasificación. Se encontró que al hacer la expansión propuesta se incrementan las relaciones aprendidas en Word2vec, permitiendo obtener una mayor precisión en la clasificación.

Keywords: Corpus extension, Word2Vec, Support vector machine, Sentiment analysis, Polarity, Semantics.
Palabras Clave: Extensión de corpus, Word2Vec, Máquinas de soporte vectorial, Análisis de sentimientos, Polaridad, Semántica.

1 Introduction

Sentiment analysis is one of the most challenging tasks in machine learning applied to text. One of its most active branches is text classification by sentiment polarity, which refers to the positivity or negativity of the sentiment represented by the text. There are prominent difficulties in classifying text by polarity when text is too short, as it is the case in Twitter in which text does not exceed 140 characters. To overcome these difficulties, this paper presents an extension to the TASS (sentiment analysis workshop of the Spanish society for natural language processing) General corpus [1] (hereafter just corpus). The TASS corpus consists of a set of tagged tweets by polarity, and the extension consists in including definitions from the Spanish Real Academy's dictionary (RAE) to the set of texts from which sentiment analysis algorithms learn. In this work, we compared a sentiment analysis algorithm trained with tweets with another trained with both tweets and RAE's dictionary definitions. The reason for extending the corpus with RAE's dictionary definitions is that they provide semantic relations between

ISSN: 1137-3601 (print), 1988-3064 (on-line)

words. The technique of extending a corpus composed of short texts is used to expand the context. Applying this technique allows to improve text classification by polarity, as shown by the studies in [2, 3]. Additionally, some text domains grow at such fast rates that it is practically impossible to maintain the training data updated. Therefore, using related domains with tagged text is a solution that has been lately used to overcome scarce tagged data. This is called cross-domain text classification, and has shown preliminary results. Nevertheless, some procedures have been developed and tested, and the results have improved, such as the work of Wang et al. [4], where Wikipedia is used to propagate labels between two different text domains which captures common words and semantic concepts contained in the documents; as well as the work of Mouriño et al. [5], where the concept of cross-language concept matching is proposed to perform cross-language text classification, which is based on the representation of concepts using Wikipedia correspondence of concepts in different languages.

The vast majority of research has been done using text in English. In spite of scarce research using text in Spanish, there has been recent and steady progress. However, it is evident that sentiment analysis research needs more attention on Spanish corpora since Spanish is a more inflected language than English [6]. Few studies, such as the one by Ortega et al. have deepen in Spanish text domain extension [7].

Some other relevant works using Spanish corpora for sentiment analysis have been based on the polarity analysis of lexicons. This is the case of the research by Pablos et al., in which Word2Vec is used to automatically create a lexicon dictionary with associated polarities [8]. Also, Cruz et al. rank lexicons by their impact on the task of text classification by polarity [9].

Other works have tackled some specific problems in the industry, e.g. [10, 11], while some research, such as [12], have enhanced results of binary classification in sentiment analysis, as well as in topic classification using the concept of maximum entropy. Concerning informal text, including the one found on the Internet, Mosquera et al. proposed preprocessing methods to regularize and normalize such texts for a better performance in sentiment analysis tasks.

In this paper, we propose a model based on Word2Vec that allows to represent words as vectors from an extended Twitter corpus with RAE's dictionary definitions. The model detects word's polarities given their occurrences in tweets tagged with positive or negative polarities. The main goal of this study was not to obtain high accuracy in text classification by polarity, but to measure the profit of classification accuracy when RAE's dictionary definitions are included in the training process of Word2Vec.

In section 2 we present the Word2Vec method, as well as review support vector machines. Section 3 shows the details of TASS corpus, the tool built to obtain RAE's dictionary definitions, and also shows a flow diagram of the proposed model for sentiment analysis. In section 4 we give and analyze the main results of our research. Finally, in section 5 we discuss the results and give some perspectives for future research, and we present the conclusions of this paper.

2 Theoretical Framework

In this section we present the tools used in our method of corpus extension. In order to do that we define some key concepts. A corpus is a collection or set of texts (also called documents) that share a topic or common origin. The vocabulary of this corpus is the set of all words contained in the corpus. In general, when texts are informal, there are sequences of characters that are not official words, but have a concrete meaning in a given cultural context. Thus, the concept of word is generalized by the concept of token, which is defined as any sequence of characters with a meaning. We also define the context of a word, referred as target word, as the set of words that are near to the target word. Specifically, the context has a size $2c$, which means that all the words that are at a maximum distance of c of the target word are context words.

Example 1 *Consider the sentence "en este día soleado me siento genial" (I feel great on this sunny day). Let "soleado" (sunny) be the target word. If $c = 2$, then the context of "soleado" is the set of words at a maximum distance of 2 from the target word. Thus the context is {"este", "día", "me", "siento"}.*

2.1 Word2Vec

One of the essential problems in automatic text analysis is to represent it numerically to establish quantitative relationships between words, phrases or longer texts. Mikolov et al. proposed the Word2Vec method, which supports the construction of embedded vector spaces whose dimensions harbor the semantic and syntactic meanings of words [13].

Word2Vec acts as a function $W2V : \mathcal{V} \to \mathbb{R}^N$, where $\mathcal{V} = \{w_i : i = 1, 2, \ldots, V\}$ is an ordered set of V words (the way in which the words are ordered does not matter). The words, or tokens w_i are initially represented by hot-vectors, that are the canonical basis of $\mathbb{R}^V$. For instance, if "sausage" is the second word w_2 of the vocabulary

$\mathcal{V}$, then its hot-vector is

$$\boldsymbol{w}_2 = (0, 1, 0, \ldots, 0), \tag{1}$$

and has V components. Hence, Word2Vec maps words from the vocabulary $\mathcal{V}$ to hot-vectors of V components, and later on, builds an embedded vector space of N dimensions in which each word w_i is paired with a vector $\boldsymbol{\omega}_i \in \mathbb{R}^N$. In general $N \ll V$, meaning that the embedded vector space has a much lower dimensionality than the number of words in the vocabulary (usually $N/V < 10^{-2}$).

The way in which the embedded vector space is built is via a neural network (NN). This NN learns the co-occurrences of words that share similar contexts. There exist two different possible configurations of the NN used by Word2Vec: CBOW and Skip-gram. CBOW means continuous bag of words, and predicts a target word given context words. This is, suppose that there are sentences in which a target word w_o is present. Beforehand, a context size c is set (to exemplify, suppose $c = 1$). This means that the target word w_o has contexts composed of two context words w_{before} and w_{after}, that are situated before and after w_o, respectively. Thus, the CBOW configuration is a NN whose input are the hot-vectors of w_{before} and w_{after}, and tries to predict the hot-vector of the target word w_o. On the other hand, the Skip-gram configuration takes as the input the hot-vector of the target word w_o and tries to predict its context words w_{before} and w_{after}.

For this work we selected the CBOW configuration that consisted of three neuron layers shown from bottom to top in Figure 1: the input, hidden and output layers. Again, if we suppose that the context has a size of $2c$, then the input layer contains $2c$ rows of V neurons in order to take the hot-vectors of the $2c$ context words. Each row of V neurons is fully connected to a hidden layer of N neurons. These connections are represented by $2c$ $N \times V$ matrices that relate each context word's hot-vector with the N-neurons hidden layer. Finally, the hidden layer is fully connected to the output layer, that contains V neurons. These connections are represented by a $V \times N$ matrix. Ideally, given a context, the NN should predict the target word's hot-vector. In the case of Figure 1, there are 4 context words and the correct target word is "soleado".

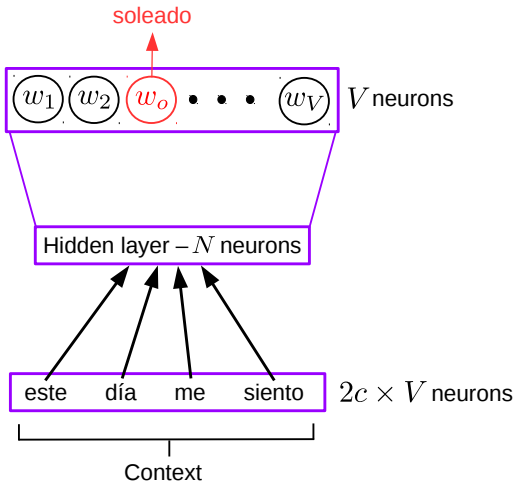


Figure 1: CBOW configuration of Word2Vec's NN using example 1.

The way in which the NN represents a word w_i as a vector in an N-dimensional vector space is by associating the i-th row of the connection matrix between the hidden and output layers with the i-th word. The reason why this works is that words that appear in similar contexts must have similar vector representations. This is achieved by defining the NN's error function as

$$E := -\log P(w_o|\mathcal{C}), \tag{2}$$

where $\mathcal{C}$ is a set of $2c$ words picked from the vocabulary, and $P(w_o|\mathcal{C})$ is the probability that w_o is the target word corresponding to the context $\mathcal{C}$. This probability is defined by a multinomial softmax distribution, which allows words from similar contexts to have similar vector representations in $\mathbb{R}^N$ [14], where the similarity is measured by the inner product.

2.2 Non linear support vector machine

A support vector machine (SVM) is a supervised learning method used for classification. This method consists in learning particular features of numerical objects that belong to different classes. The learning process starts with a training dataset

$$S := \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \ldots, (\boldsymbol{x}_m, y_m)\}, \tag{3}$$

where $\boldsymbol{x}_i \in \mathbb{X} \subset \mathbb{R}^p$ and $y_i \in \mathbb{Y} = \{-1, 1\}$ for all $i = 1, \ldots, m$. Note that there are only two tags because we performed binary classification (positive or negative polarity). Moreover, the observations $\boldsymbol{x}_i$ have p features.

A SVM looks for a pair of parameters $\boldsymbol{w} \in \mathbb{R}^p, b \in \mathbb{R}$ such that the hyperplane

$$\{\boldsymbol{x} \in \mathbb{R}^p : f(\boldsymbol{x}) := \langle \boldsymbol{w}, \boldsymbol{x} \rangle + b = 0\} \tag{4}$$

given by a discriminant function f, divides the space $\mathbb{R}^p$ in two semi-spaces: one that contains all the observations with tag -1, and the other one that contains the ones with tag 1. This means that the SVM builds two connected regions in $\mathbb{R}^p$, one for each class. Also, the distance from the hyperplane to the nearest observation's vector from each class must be maximized. These nearest vectors are called support vectors. The method so far presented is called linear SVM, and rarely is useful to classify real data.

To treat real data, Vapnik et al. proposed to map the set of observations $\mathbb{X}$ to a higher-dimensionality space, called the feature space $\mathbb{F} \subset \mathbb{R}^{\tilde{p}}$, where $\tilde{p} > p$, so that the observations are linearly separable in $\mathbb{F}$ [15]. This means that in $\mathbb{R}^{\tilde{p}}$ there exists a hyperplane that divides the space as desired. This map is built with a function $\Phi : \mathbb{R}^p \to \mathbb{R}^{\tilde{p}}$ that allows a re-definition of the discriminant function as

$$f(\boldsymbol{x}) = \langle \boldsymbol{w}, \Phi(\boldsymbol{x}) \rangle + b, \tag{5}$$

where $\boldsymbol{w} \in \mathbb{R}^{\tilde{p}}$. To make this learning process computationally efficient, a kernel $k(\boldsymbol{x}, \boldsymbol{x}')$ is defined as

$$k(\boldsymbol{x}, \boldsymbol{x}') := \langle \Phi(\boldsymbol{x}), \Phi(\boldsymbol{x}') \rangle. \tag{6}$$

If we write $\boldsymbol{w} = \sum_{i=1}^{m} \alpha_i \boldsymbol{x}_i$, where $\alpha_i \in \mathbb{R}, i = 1, 2, \ldots, m$, then

$$f(\boldsymbol{x}) = \sum_{i=1}^{m} \alpha_i k(\boldsymbol{x}, \boldsymbol{x}_i) + b. \tag{7}$$

The reason why this is computationally efficient is that it is not necessary to explicitly compute Φ in order to compute $k(\boldsymbol{x}, \boldsymbol{x}')$. This is the case of the Gaussian kernel, or radial basis functions kernel (RBF), defined by

$$k_{\text{RBF}}(\boldsymbol{x}, \boldsymbol{x}') = \exp(-\gamma ||\boldsymbol{x} - \boldsymbol{x}'||), \tag{8}$$

where $\gamma \in \mathbb{R}^+$ is a free parameter that represents the influence that a vector $\boldsymbol{x}$ has over a vector $\boldsymbol{x}'$. In other words, since k_{RBF} is a similarity measure between two vectors, then γ is the inverse radius of influence of the support vectors. If this radius is small, then the SVM-RBF allows non-connected regions in $\mathbb{R}^p$ for each class.

It can be shown that the function Φ_{RBF} corresponding to the kernel $k_{\text{RBF}}(\boldsymbol{x}, \boldsymbol{x}')$ maps the observations to a vector space of infinite dimension [16], i.e.:

$$\Phi_{\text{RBF}} : \mathbb{R}^p \to \mathbb{R}^{\infty}, \tag{9}$$

which eases, in theory, the task of finding the hyperplane (in $\mathbb{R}^{\infty}$) that divides the observations from different classes.

In our work, we selected the RBF kernel because it is able to separate non-linear data in high-dimensional spaces, which is not necessarily the case of linear kernels. However, in the SVM it is possible to change the kernel by other non-linear kernels.

3 Method and Materials

In this section we show the stages and different materials used in our proposed model of extending the TASS General corpus with RAE's dictionary definitions. The model, and the way in which we measure its advantages with respect to not including RAE's dictionary definitions, can be seen in Figure 2. In sections 3.1-3.4 we explain the sequential flow of our method.

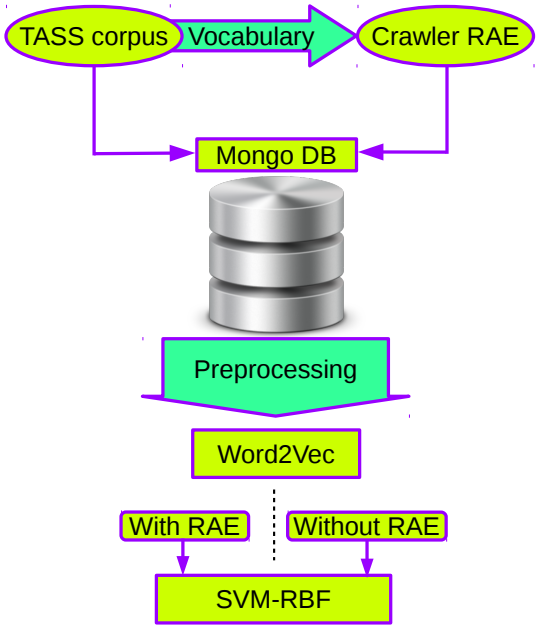


Figure 2: Flow diagram of the proposed model.

3.1 TASS corpus

TASS is a yearly-released workshop organized by the Spanish society for the natural language processing (SEPLN). TASS consists of activities related to sentiment analysis. Participants have access to tagged corpus that contains over 68,000 tweets in Spanish written by around 150 influent people (famous people or celebrities) from the world of politics, economy, media and culture, collected from November, 2011 to March, 2012 [1]. Figure 3 shows the tweet length in characters distribution. The figure exhibits the fact that this group of people tend to use the maximum number of characters to express their opinions.

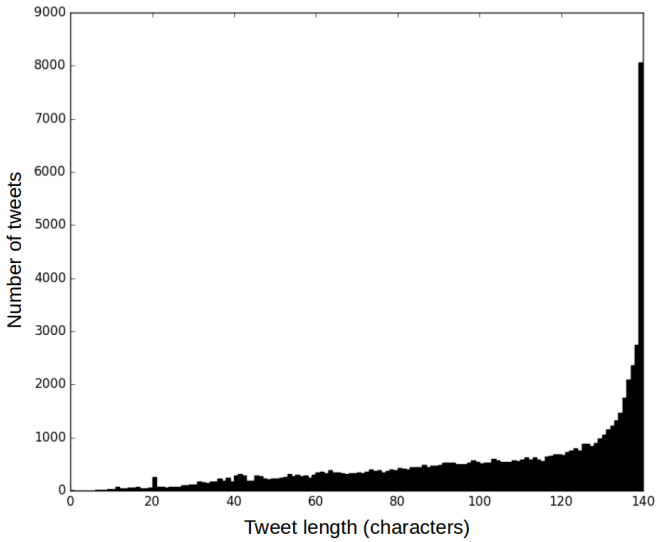


Figure 3: Histogram with tweets length distribution measured in characters.

Approximately 10% of the tweets are tagged with topic and polarity. For our study, the important tag is polarity and comes in 6 classes: strongly positive (P+), positive (P), neutral (NEU), negative (N), strongly negative (N+) and absence of polarity (NONE). In this work, we only used 4 polarity tags and grouped them

together in two main classes: the positive class (P+ y P), and the negative one (N+ y N). This means that the set of tagged tweets has around 5100 tweets, corresponding to 70% of the initial tagged tweets.

The corpus is originally written in XML format. To read it and manage its information efficiently, the Python's Beautiful Soup library was used [17]. Additionally, the corpus was stored in a Mongo database using Python's Pymongo library with the objective of reading and writing information rapidly [18].

3.2 Crawler

With the purpose of extending the TASS corpus with RAE's dictionary definitions, a crawler was built in order to obtain such definitions. The process consisted in searching each token from the TASS corpus' vocabulary and to obtain the first definition, which is commonly the most used one. These definitions were stored in a collection of the Mongo database.

In pursuance of building the crawler, Selenium was used [19]. This tool allows to automatize multiple processes in web browsers. In our case, we automatized the process of typing each token in the search bar of RAE's dictionary, and to obtain the definitions of those tokens found in the dictionary.

It is important to point out that most of the tokens are not words present in the RAE's dictionary, as shown in Figure 4. This figure is a heat map, or a 2D histogram, in which dark (light) zones correspond to many (few) tweets, and whose the distribution of number of tokens per tweet, as well as the portion of those tokens found in RAE's dictionary. If a tweet is on the black, green or cyan line means that every, half or a quarter of its tokens are found in RAE's dictionary, respectively. Therefore, Figure 4 shows that most tweets have from a quarter to half of their tokens present in RAE's dictionary.

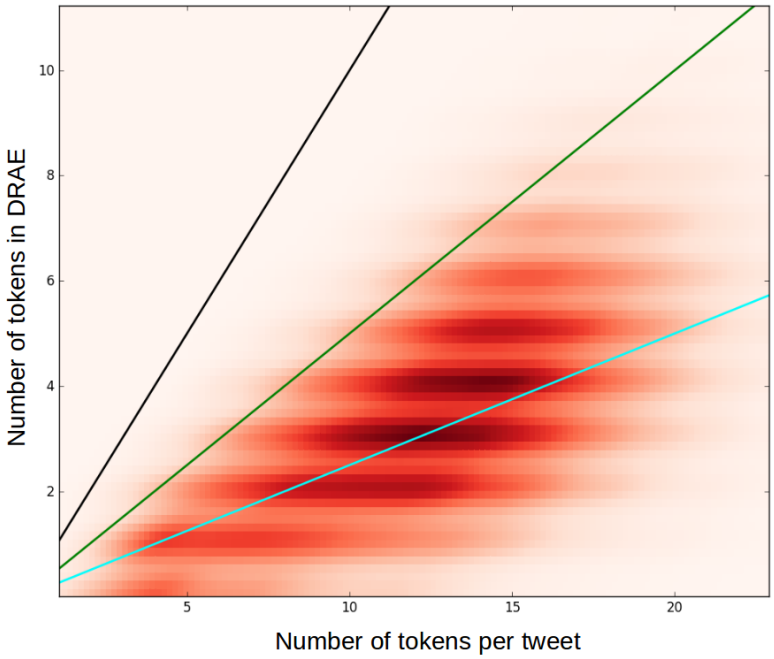


Figure 4: 2D histogram of number of tokens per tweet and the percentage of them that are found in RAE's dictionary (DRAE). Black, green and cyan lines correspond to 100%, 50% and 25%, respectively.

3.3 Preprocessing

The content of each tweet was modified following 5 preprocessing steps: cleaning, automatic correction, regularization, tokenization and lemmatization.

Cleaning

Usually, common words, also called stop-words, do not contribute to the semantic meaning of sentences. Therefore, they are not interesting for natural language processing since they do not supply new information and require

computing time for their processing. Normally, these words are articles, pronouns, prepositions, among others. To remove them from the tweets, the NLTK library was used [20].

Automatic correction

In order to avoid words written with mistakes, Language Tool's API for Python was used [21]. Language Tool verifies grammar, orthography and style in a text. This tool allowed us to automatically correct words and structures found in tweets that were badly written.

Regularization

Although uppercase might be useful for some natural language processing applications [22], a good practice to analyze text using Word2Vec is to lowercase all characters. Also, the tokenizer that was used (see section 3.3) in this work required to remove accents from characters because it interprets them as token separators, i.e., words like "canción" (song) were tokenized as ["canci", "on"].

Tokenization

Tokenization is the process of converting a string of characters to a list of tokens. The automation of this procedure is implemented by NLTK's tokenizer TweetTokenizer [20].

Lemmatization

The last step of preprocessing is lemmatization. Here, tokens that are actual words are replaced by their lemma. For example, the token "canciones" was replaced by "canción". This task was performed using the pattern library from the computational linguistics group CLiPS [23]. The objective of this step was to reduce the vocabulary so that sentences containing words with the same lemma would be related by their respective embedded vector.

3.4 Experiments

We designed a model for training Word2Vec and training the SVM-RBF classifier, and measured its performance with the classification accuracy metric. The model depends on some training parameters, denoted by a tuple $\boldsymbol{\theta}$. Therefore, the model is a function $M : \Theta \to [0, 1]$, where Θ is the set of all possible parameters of Word2Vec and SVM-RBF. Moreover, M takes on values from 0 to 1, where 0 corresponds to 0% of classification accuracy, and 1 to 100%.

The process of training the model with a tuple of parameters and obtaining its corresponding classification accuracy is called the execution of an experiment. Each experiment was carried out in two stages. The first one consists in training Word2Vec's NN, and the second one consists in training and testing the SVM-RBF classifier. Furthermore, Word2Vec's NN training was performed in two different ways: the first one used both tweets and RAE's dictionary definitions as training sentences, and the second one used only tweets (see dashed line zone in Figure 2). This allows to define a first parameter $\theta_1 = 0, 1$, where 0 represents the first way and 1 the second one. A second parameter $\theta_2 = 20, 40, 60, 80, 100$ was called the minimum number of occurrences, and implies that vector representations of tokens that appear less than θ_2 times in all the corpus were not built. $\theta_3 = 3, 5, 7, 10$ was the third parameter, and corresponds to the maximum distance from the target word that a token has to have in order to be in its context (i.e. $\theta_3 = c$). The fourth parameter, $\theta_4 = 30, 60, 90, 120, 150, 190, 220, 250$, was the embedded vector space dimension, this is, the number of neurons in the hidden layer of Word2Vec's NN.

Once we obtained the tokens' vector representations, we built the tweets' vector representations for those tweets that were tagged with polarity in the corpus. This was done by taking the average of the vector representation of the tokens that composed each tweet. However, these construction of tweets' vector representation was not done for every tagged tweet. We introduced a fifth parameter $\theta_5 = 4, 5, 6, 7, 8, 9, 10$ called the minimum number of tokens per tweet, and the vector representations were not built for tweets with less than θ_5 tokens. This means that θ_5 limited the number of tweets that served as observations for the training and testing of the SVM-RBF classifier. In particular, the number of tweets varied from 1,112 to 4,074 (because both θ_5 and θ_2 affect the number of tweets), and these were split in 80% for training and 20% for testing. In order to keep the classes with the same amount of tweets, tweets were randomly eliminated from the class containing the most tweets until both classes had the same number of tweets. It should be noted that from the first stage the tweets were split into these two sets.

With respect to Word2Vec's implementation, Python's gensim library was used [24]. Regarding the SVM-RBF classifier, Python's scikit-learn library was used [25]. The algorithm implemented in this library not only contains the γ parameter, which is the sixth parameter $\theta_6 = 10^{-3}, 10^{-2}, 10^{-1}, 10^{0}, 10^{1}$, but also has a parameter θ_7 that

takes the values $\theta_7 = 10^{-2}, 10^{-1}, 10^{0}, 10^{1}, 10^{2}$. The latter, trades off misclassification of training examples against simplicity of the discriminant surface. Low values of θ_7 generate smooth surfaces, while large values grants the algorithm the power to select more training vectors as support vectors, and build more complex discriminant surfaces.

In conclusion, the space of parameters Θ is the set of all possible tuples $\boldsymbol{\theta} = (\theta_1, \theta_2, \ldots, \theta_7)$ made with the mention values for each parameter. Therefore 56,000 experiments were executed in total. Thus, 28,000 experiments were executed without using RAE's dictionary definitions and 28,000 using it.

We define $\tilde{\Theta}$ as the set of all possible tuples $\tilde{\boldsymbol{\theta}} = (\theta_2, \theta_3, \ldots, \theta_7)$. The reason why we carried out this brute force work of many experiments was the construction and comparison of two classification accuracy distributions defined over $\tilde{\Theta}$: one corresponding to $\theta_1 = 0$ (without RAE) and the other one corresponding to $\theta_1 = 1$ (with RAE). These distributions helped us answer the question of how probable is that $M(\boldsymbol{\theta}|_{\theta_1=1}) > M(\boldsymbol{\theta}|_{\theta_1=0})$ when we choose $\tilde{\boldsymbol{\theta}}$ randomly, where $\boldsymbol{\theta} = \theta_1 \oplus \tilde{\boldsymbol{\theta}}$ ($\oplus$ is used as concatenation symbol). In other words, the distributions determined if it is the case that extending the corpus with RAE's dictionary definitions to the set of training sentences of Word2Vec's NN improve the classification accuracy of polarities.

4 Results and Analysis

The classification accuracy distributions corresponding to experiments that only used tweets, and experiments that used both tweets and RAE's dictionary definitions as training sentences for Word2Vec's NN are shown in Figure 5.

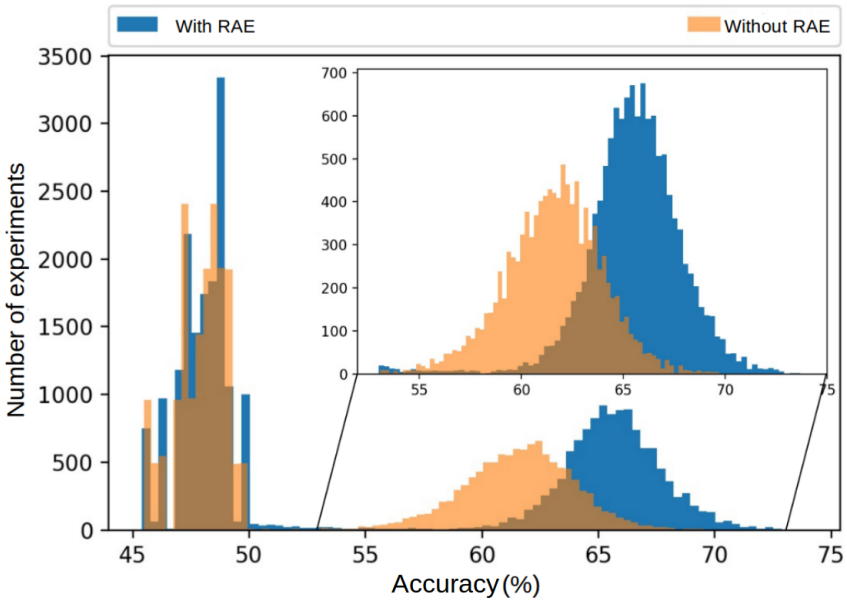


Figure 5: Classification accuracy distributions for experiments with and without RAE's dictionary definitions incorporated to the training sentences of Word2Vec's NN.

Figure 5 has two outstanding regions: the one corresponding to accuracy values greater or equal than 50% and the one with accuracy values lower than 50%. Both regions are interesting, but the first one offers overwhelming hints that using RAE's dictionary definitions within the set of training sentences of Word2Vec's NN improves classification accuracy significantly. In this region, both distributions are gaussian and contain 12,120 and 10,989 experiments that use and do not use RAE's dictionary definitions, respectively. Additionally, the gaussian distributions point out that, in average, using RAE's dictionary definitions as part of the training sentences of Word2Vec's NN improves the accuracy between 4%-5% with respect to the accuracy obtained when one does not use these definitions. Also, it is notable that the width of the distribution corresponding to experiments that incorporate RAE's dictionary definitions is lower than the width of the distribution of experiments that do not include them, which shows less dispersion and greater stability in the results.

The second region is interesting from the mathematical point of view. For two polarities, a random classifier would generate a gaussian distribution of classification accuracy centered at 50%. The data shown in Figure 5 below this percentage are clearly away from a gaussian distribution, and signal that the classifier's performance

is significantly worse than a random classifier. A possible explanation of this phenomenon is that for some values of γ and θ_7, the SVM-RBF classifier generates very simple hypersurfaces that prevent a correct discrimination of non connected regions corresponding to a single polarity. This is evident from Figure 6, which shows a histogram of experiments grouped by the value of γ for three intervals of accuracy. The larger γ, the more complex and versatile the hypersurfaces are, and the higher the classification accuracy is (green bars). Conversely, the lower γ, the lower the classification accuracy is (pink bars).

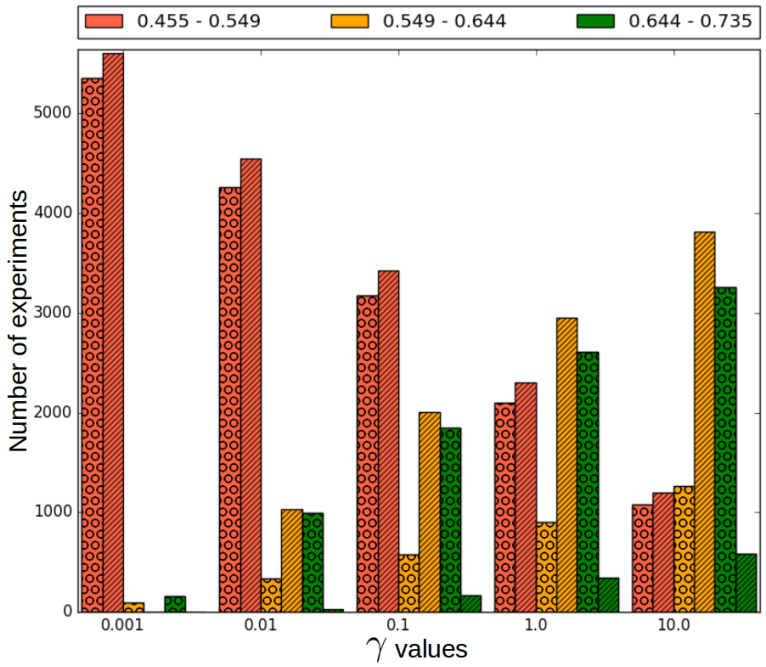


Figure 6: Histogram of number of experiments with different values of γ for three intervals of low (0.455-0.549), medium (0.549-0.644) y high (0.644-0.735) accuracies, shown with pink, yellow and green colors, respectively. Dotted (lined) bars correspond to experiments that use (do not use) RAE's dictionary definitions as part of Word2Vec's NN training sentences.

A similar behaviour was found with the parameter θ_7. Therefore, it is clear that the classifier is very sensible to the parameters $\gamma = \theta_6$ and θ_7. Since the best classification results were obtained with the highest values of θ_6 and θ_7, we built a comparison between experiments that used and did not use RAE's dictionary definitions as part of Word2Vec's NN training sentences, taking into account all the experiments with parameters $\theta_6 = 1, 10$ and $\theta_7 = 10, 100$. The comparison was made in the following way: A histogram of the following set was made:

$$\{M(\boldsymbol{\theta}|_{\theta_1=1}) - M(\boldsymbol{\theta}|_{\theta_1=0}) : \boldsymbol{\theta} \in \Theta \wedge \theta_6 = 1, 10 \wedge \theta_7 = 10, 100\}.$$

In other words, the comparison was made by producing a histogram of the improvement percentage in the classification accuracy when RAE's dictionary definitions were included as part of Word2Vec's NN training sentences. This is, given a tuple of parameters, the histogram shown in Figure 7 tells the probability that the accuracy obtained when using RAE's dictionary definitions is higher than when they are not used.

Figure 7 shows that the improvement percentage in classification accuracy is statistically significant (to 1σ). For the selected subset of experiments, this improvement is, in average, of 3%.

5 Conclusions

The main goal of this paper was to measure the impact on the classification accuracy provided by the inclusion of RAE's dictionary definitions in Word2Vec's NN training sentences. It was found that most of the tweets from the TASS corpus used near the maximum number of characters allowed by Twitter, and that about a quarter to a half of the tokens contained in each tweet were actual words found in RAE's dictionary. The arguments exposed in Section 4 assert that there is a significant improvement in the classification accuracy when RAE's dictionary definitions are included in the Word2Vec's NN training sentences, whenever γ and θ_7 are correctly

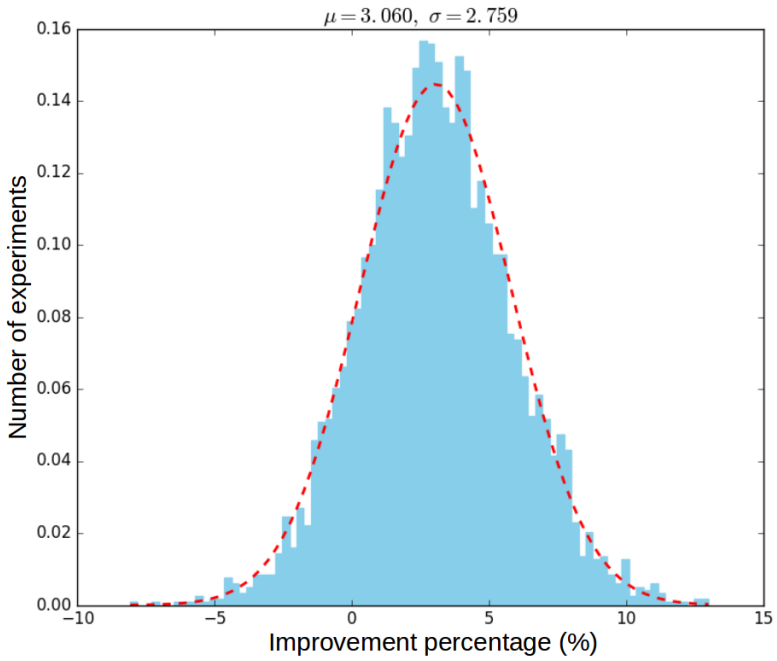


Figure 7: Normalized histogram of the number of experiments by the improvement percentage obtained when RAE's dictionary definitions were included as part of Word2Vec's NN training sentences. The red line shows a gaussian curve with mean 3.060% and standard deviation 2.759%.

selected. Furthermore, it is clear that the accuracy percentages of the binary polarity classification are low and can be improved including a lexicon polarity dictionary in order to build the vector representations of tweets more adequately. Finally, it is important to verify that this improvement does not only occur with the Word2Vec method, but also with other representation models (e.g. GloVe) of words as embedded vectors. Also relevant is to measure the impact of including RAE's dictionary definitions as part of Word2Vec's NN training sentences in different corpora; for instance corpora with longer and better written texts in order to ensure that a larger percentage of these texts' tokens are found in RAE's dictionary, which could be helpful for automatic news analysis like predicting stock prices with financial news articles [26].

References

[1] Eugenio Martínez Cámara, Miguel Á. García Cumbreras, Julio Villena Román, and Janine García Morera. Tass 2015 – the evolution of the spanish opinion mining systems. *Procesamiento del Lenguaje Natural*, 56:33–40, 2016.

[2] Yuan Man. Feature extension for short text categorization using frequent term sets. *Procedia Computer Science*, 31:663 – 670, 2014. 2nd International Conference on Information Technology and Quantitative Management, ITQM 2014. URL: `http://www.sciencedirect.com/science/article/pii/S1877050914004918`, `doi:http://dx.doi.org/10.1016/j.procs.2014.05.314`.

[3] S. Lee, J. Kim, and S. H. Myaeng. An extension of topic models for text classification: A term weighting approach. In *2015 International Conference on Big Data and Smart Computing (BIGCOMP)*, pages 217–224, 2015. `doi:10.1109/35021BIGCOMP.2015.7072834`.

[4] Pu Wang, Carlotta Domeniconi, and Jian Hu. Cross-domain text classification using wikipedia. *IEEE Intelligent Informatics Bulletin*, 9(1):5–17, 2008.

[5] Marcos Antonio Mouriño García, Roberto Pérez Rodríguez, and Luis Anido Rifón. Wikipedia-based cross-language text classification. *Information Sciences*, 406–407:12 – 28, 2017. URL: `http://www.sciencedirect.com/science/article/pii/S0020025517306680`, `doi:https://doi.org/10.1016/j.ins.2017.04.024`.

[6] Antonio Fernández Anta, Luis Núñez Chiroque, Philippe Morere, and Agustín Santos. Sentiment analysis and topic detection of Spanish tweets: A comparative study of NLP techniques. *Procesamiento del Lenguaje Natural*, 50:45–52, 2013.

[7] F. Javier Ortega, José A. Troyano, Fermín L. Cruz, and Fernando Enríquez. Enriching user reviews through an opinion extraction system. *Procesamiento del Lenguaje Natural*, 55:119–126, 2015.

[8] Aitor García-Pablos and Germán Rigau. Unsupervised word polarity tagging by exploiting continuous word representations. *Procesamiento del Lenguaje Natural*, 55:127–134, 2015.

[9] Fermín L. Cruz, José A. Troyano, Beatriz Pontes, and F. Javier Ortega. Ml-senticon: Un lexicón multilingüe de polaridades semánticas a nivel de lemas. *Procesamiento del Lenguaje Natural*, 53:113–120, 2014.

[10] Flor Miriam Plaza-del Arco, M. Teresa Martín-Valdivia, Salud María Jiménez-Zafra, M. Dolores Molina-González, and Eugenio Martínez-Cámara. Copos: Corpus of patient opinions in spanish. application of sentiment analysis techniques. *Procesamiento del Lenguaje Natural*, 57:83–90, 2016.

[11] L. Alfonso Ureña López, Rafael Muñoz Guillena, José A. Troyano Jiménez, and María-Teresa Martín Valdivia. Attos: Análisis de tendencias y temáticas a través de opiniones y sentimientos. *Procesamiento del Lenguaje Natural*, 53:151–154, 2014.

[12] Fernando Batista and Ricardo Ribeiro. Sentiment analysis and topic classification based on binary maximum entropy classifiers. *Procesamiento del Lenguaje Natural*, 50:77–84, 2013.

[13] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. *Advances in neural information processing systems*, pages 3111–3119, 2013.

[14] Yoav Goldberg and Omer Levy. word2vec explained: Deriving mikolov et al.'s negative-sampling word-embedding method, 2014.

[15] Bernhard E. Boser, Isabelle M. Guyon, and Vladimir N. Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, COLT '92, pages 144–152, New York, NY, USA, 1992. ACM. URL: `http://doi.acm.org/10.1145/130385.130401`, `doi:10.1145/130385.130401`.

[16] Nello Cristianini and John Shawe-Taylor. *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods*. Cambridge University Press, 1st edition, 2000.

[17] Leonard Richardson. Beautiful soup documentation, 2015. URL: `https://www.crummy.com/software/BeautifulSoup/bs4/doc/` [cited February 27, 2017].

[18] MongoDB Inc. Pymongo 3.4.0 documentation. URL: `https://api.mongodb.com/python/current/#about-this-documentation` [cited March 5, 2017].

[19] Sagar Shivaji Salunke. *Selenium Webdriver in Python: Learn with Examples*. CreateSpace Independent Publishing Platform, 1st edition, 2014.

[20] Steven Bird, Ewan Klein, and Edward Loper. *Natural language processing with Python: analyzing text with the natural language toolkit*. O'Reilly Media, Inc., 2009.

[21] Steven Myint. Language check. URL: `https://pypi.python.org/pypi/language-check` [cited March 10, 2017].

[22] Konstantin Buschmeier, Philipp Cimiano, and Roman Klinger. An impact analysis of features in a classification approach to irony detection in product reviews. *Proceedings of the 5th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, pages 42–49, 2014.

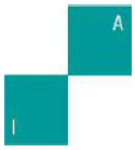
[23] Tom De Smedt and Walter Daelemans. Pattern for python. *Journal of Machine Learning Research*, 13:2063–2067, 2012.

[24] Radim Řehůřek and Petr Sojka. Software framework for topic modelling with large corpora. In *Proceedings of the LREC 2010 Workshop on NewChallenges for NLP Frameworks*, pages 45–50, Valletta, Malta, 2010. ELRA.

[25] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[26] Robert P. Schumaker and Hsinchun Chen. Textual analysis of stock market prediction using breaking financial news: The azfin text system. *ACM Trans. Inf. Syst.*, 27(2):12:1–12:19, 2009. URL: `http://doi.acm.org/10.1145/1462198.1462204`, `doi:10.1145/1462198.1462204`.

Inteligencia Artificial 21(62), (2018), 13-24
doi: 10.4114/intartif.vol21iss62pp13-24



INTELIGENCIA ARTIFICIAL

http://journal.iberamia.org/

Rules for predicting compliance with the quality of wastewater in a treatment plant applying data mining

Reglas para predecir el cumplimiento de la calidad del agua residual en una planta tratadora con minería de datos

Facundo Cortés-Martínez, Alejandro Treviño-Cansino, María-Aracelia Alcorta-García*, Arturo Tadeo Espinoza-Fraire, José Armando Sáenz-Esqueda, Julio Gerardo Lozoya Vélez
facundo_cm@yahoo.com.mx, atrevino@ujed.mx, maaracelia@gmail.com, tadeo1519@gmail.com, jase1588@gmail.com, gerardo_lovelez@ujed.mx

Abstract A problem faced by the water operators is the compliance with the regulations on the quality of the treated wastewater. The most important thing is to implement strategies that favor compliance with the regulations. Data mining is a tool that allows the prediction of the water quality in the effluent of water treatment systems. In this research job, a criterion for nominal variables and data preprocessing is proposed. Subsequently, the data mining system (classification) was applied to define the prediction of water quality. The classificatory methodologies applied were: OneR, Decision Table, J48, single level decision tree, PART y LMT. Results show that, the best algorithm was the *Decision Table* with 85.45% of the instances correctly classified. The algorithm determined two rules for the regulation's achievement. It is important to mention that currently there are data mining procedures to predict water quality in the effluent of a treatment system, although, these procedures use strictly numeric variables; while in the current research, nominal variables were considered. Finally, results are discussed and industrial processes that generate organic waste and other pollutants are indicated.

Resumen Un problema que enfrentan los organismos operadores de agua, es el cumplimiento de la normatividad en la calidad del agua residual tratada. Por lo que es recomendable implementar estrategias que favorezcan el cumplimiento de las regulaciones. La minería de datos es una herramienta que permite predecir la calidad del agua en el efluente de los sistemas de tratamiento. En el presente estudio se propone un criterio para el pre procesado de datos donde se consideraron variables nominales. Posteriormente se aplicó el sistema de minería de datos (clasificación) para definir la predicción de la calidad del agua. Se utilizaron los siguientes clasificadores: *OneR*; *DecisiónTable*, J48, árbol de decisión de un solo nivel, *PART* y *LMT*. Los resultados muestran que el mejor algoritmo fue el *DecisiónTable* con el 85.45 % de instancias correctamente clasificadas. El algoritmo determinó dos reglas para el cumplimiento de la normatividad. Es importante indicar que a la fecha existen procedimientos con minería de datos para predecir la calidad del efluente de un sistema de tratamiento, pero utilizan estrictamente variables numéricas; mientras que en el presente trabajo se utilizaron variables nominales, finalmente se discuten los resultados y se indican los procesos industriales que generan materia orgánica y otros contaminantes.

Keywords: Biochemical oxygen demand, decision table, nominal variables, classification, data mining.
Palabras clave: Demanda bioquímica de oxígeno, tabla de decisión, variables nominales, clasificación, minería de datos.

ISSN: ISSN 1137-3601 (print), 1988-3064 (on-line)

1. Introducción

Un reto para las autoridades tanto municipales, estatales y federales, es la observancia de la calidad del agua residual tratada. Por lo que es necesario desarrollar esquemas de análisis para el cumplimiento de los límites máximos permitidos en el efluente de las plantas de tratamiento. De acuerdo con [23], [24] y [26] la minería de datos es una herramienta que procesa grandes cantidades de información, y además, permite monitorear el estado del sistema de tratamiento, la clasificación de las aguas residuales, el control de los procesos así como la predicción de la calidad del agua tratada. En [35] se define la minería de datos como las actividades necesarias para extraer información sintetizada y desconocida de manera de generar nuevos conocimientos. Es decir se refiere a la extracción de datos no triviales que se encuentran implícitos en la información proporcionada. De acuerdo con [1] el proceso se describe en términos generales en cinco partes: a) selección de la información; b) transformación de los datos: se refiere a organizar la información como se necesite; c) minería de datos, es decir una vez modificada la información se aplica el sistema; d) interpretación de los resultados y e) la incorporación del nuevo conocimiento.

1.1 Revisión de la Literatura

Algunos estudios similares para la predicción de la calidad del agua: en [38] se inspeccionaron aguas residuales ácidas con minería de datos, luego en [48] por medio de modelos de simulación se reportó la predicción del caudal en el efluente del sistema de tratamiento. Tiempo después en [40] los autores llevaron a cabo estudios con el fin de identificar patrones ocultos con inducción de reglas, se utilizaron 18 variables; es decir, diferentes parámetros en el influente del sistema de tratamiento y las mediciones se aplicaron en la entrada y salida de cada proceso. Algunos parámetros considerados fueron la Demanda Bioquímica de Oxígeno (DBO); Demanda Química de Oxígeno (DQO); Gasto y Sólidos Suspendidos. Enseguida algunos estudios publicados acerca del uso de redes neuronales para predecir la calidad del agua residual, tanto en el influente como en el efluente de sistemas de tratamiento: [42], [46], [16], [37], [34], [36] y [2]. Un trabajo acerca de la identificación de patrones en los datos puede consultarse en [31] los autores publicaron un estudio para identificar patrones en el flujo: predicción de gasto en el influente del sistema de tratamiento. Luego [8] realizó estudios para definir el grado de contaminación en un río con minería de datos; enseguida [29] reportaron un estudio para predecir la calidad del agua en el influente del sistema de tratamiento, con el fin de ahorrar electricidad donde utilizaron series temporales para la predicción. Luego [44] realizaron una investigación para predecir los sólidos suspendidos totales con minería de datos y series temporales. Por ese tiempo [28] realizó un estudio de clasificación con muestras de agua contaminada, donde se incluyeron 30 industrias. Los resultados demostraron que las industrias presentaron un patrón de descarga similar. El análisis fue realizado con variables numéricas y los parámetros fueron: Demanda Bioquímica de Oxígeno; potencial de hidrógeno (pH); Demanda Química de Oxígeno, Sólidos en Suspensión, grasas y aceites. Enseguida [3] mencionaron que la técnica de agrupación de datos es muy útil en la aplicación de minería de datos y que esta técnica es muy usada en diferentes disciplinas: ingeniería, economía, geología, electrónica, estadística y psicología. Luego [9] monitoreó la eficiencia de depuración en una planta de tratamiento de lodos activados donde se aplicó un análisis estadístico multivalente y los parámetros analizados fueron: potencial de hidrógeno, conductividad, gasto, sólidos en suspensión DBO, DQO, nitrógeno y fósforo.

Todos los trabajos mencionados han contribuido satisfactoriamente con modelos de predicción. No obstante, en ningún estudio se han aplicado variables nominales para la predicción de la calidad del agua en sistemas de tratamiento. La aportación del presente estudio con respecto a los autores indicados es la manera en la que fueron pre procesados los datos con el uso de variables nominales. Se incluyó como base lo indicado en la normatividad y se agregó una variable nominal con dos posibles resultados: “NO_CUMPLE” y “SÍ_CUMPLE”.

La DBO se refiere al oxígeno disuelto necesario para que los microorganismos, presentes en el agua residual, puedan procesar la materia orgánica. Lo anterior en un lapso de tiempo de 5 días a 20 °C. [11], [12], [33]. El método de análisis en laboratorio de los Sólidos Sedimentables mide la cantidad aproximada de lodos que serán generados en un proceso de sedimentación primaria. La prueba se realiza con un *cono (Imhoff)* en 60 minutos [11]. Solidos Suspendidos: se refiere a la medición de partículas sólidas y suspendidas que contiene el agua residual. Este parámetro bloquea los rayos solares en los sistemas de agua y en ocasiones los sólidos suspendidos contienen metales [11].

Se utilizó la herramienta (*Waikato Environment for Knowledge Analysis WEKA* por sus siglas en inglés). Los algoritmos utilizados llevan a cabo el aprendizaje computacional, estadísticas, asociaciones, anomalías, eventos en los datos, visualización, inteligencia artificial y reconocimiento de patrones [1], [22]. Los procesos desarrollados por los sistemas de minería de datos tienen como propósito ajustar modelos o definir patrones.

El objetivo de presente artículo fue definir un criterio de pre procesado de datos incluyendo una variable nominal con dos posibles resultados, lo anterior considerando algunos parámetros tanto en el influente como en el efluente de un sistema de tratamiento de aguas residuales de lodos activados. Lo indicado con el propósito de predecir la calidad del agua residual en la salida de la planta aplicando la herramienta *WEKA.* Después de obtener los resultados, se realizarán recomendaciones para favorecer el control de la contaminación en los procesos industriales.

1.2 Procesos Industriales que Generan Materia Orgánica

Con el propósito de regular los límites de concentración en la entrada del sistema de tratamiento, y favorecer el cumplimiento de las condiciones que se indican en la normatividad, la CNA e IMTA en [11]; además en [19], [20], [21] y [43] sugieren se controle a las empresas que generan contaminantes. En la Tabla 1 se indican algunos de los procesos industriales que deben verificarse.

Tabla 1. Procesos que incluyen alto grado de contaminantes. Fuente: [11].

	Descripción	CE	pH	GyA	SS	SST	DBO	DQO	SAAM
3111	Industria de la carne			X		X	X	X	
3112	Fabricantes de lácteos			X		X	X	X	
3113	Procesamiento de alimentos enlatados	o		X	X		X	X	X
3114	Beneficio y molienda de cereales y otros productos agrícolas			X	X	X			
3115	Pastelerias			X	X			X	
3116	Molienda de maíz	o	o		X	X	X	X	
3117	Fabricantes de aceites y grasas			o				X	
3118	Industria de azucar	o	o			X	X	X	o
3119	Fabricación de cacao, chocolate y productos de confitería			X	X	X	X	X	
3121	Fabricación de otros productos alimenticios para el consumo humano			X	X	X	X	X	

El control de los contaminantes de las industrias y comercios tiene como propósito la protección del sistema de drenaje y alcantarillado sanitario, equipos de bombeo y la correcta operación de la planta de tratamiento municipal [11]. Algunos sistemas de tratamiento que pueden implementar las empresas previamente al vertido del agua residual de su proceso, son las siguientes: trampas de grasas, aceites y sólidos, balances de masa de procesos industriales, plantas de tratamiento y precipitación química entre otros.

El artículo está organizado de la siguiente manera: en el apartado 2 se indican los materiales y métodos, en 3 en forma detallada el criterio para el pre procesado de los datos; en 4 en filtrado de los datos, en 5 se muestran los resultados de la aplicación de los diferentes clasificadores; y por último en 6, las conclusiones.

2. Materiales y Métodos

La información de las mediciones del agua residual tanto en el influente como del efluente del sistema de tratamiento, fueron tomados de una Planta de Tratamiento de Aguas Residuales (PTAR) de lodos activados localizada en Manresa cerca de Barcelona. El sistema recibe un gasto en el influente de 30,000 m^3/día de una población de 75,000 habitantes como se establece en [30]. Esta base datos ha sido utilizada para otros estudios: [4], [6], [14], [17]. Debido al volumen de información no se incluye en el presente documento; sin embargo pueden verificarse en la cita ya señalada.

3. Pre procesado de Datos

En esta etapa se llevó a cabo un conjunto de operaciones con la finalidad de preparar los datos de análisis, con el objetivo de adaptarlos para aplicar la técnica de minería de datos que mejor se adapte al problema. Para alimentar el modelo se consideraron los siguientes parámetros: gasto en el efluente (Qe); ZNe, potencial de hidrógeno en el efluente (PHe): DBOe, DQOe, sólidos suspendidos en el efluente (SSe); sólidos suspendidos volátiles en el efluente (SSVe); sólidos sedimentables en el efluente (SSEDe) y conductividad.

Adicionalmente, como ya se indicó, se agregó un atributo nominal si cumple o no con la norma NOM-SEMARNAT-001-1996 [18]. Enseguida se muestra el pre procesado de datos realizado.

```
@relation 'DATOS_ULTIMO_NOM_001_Modificado-
weka.filters.unsupervised.attribute.StringToNominal-Rlast-
weka.filters.unsupervised.attribute.Reorder-
R1,2,3,4,5,6,7,8,9,10,11,12,13,14,16,15-
weka.filters.unsupervised.attribute.StringToNominal-Rlast-
weka.filters.unsupervised.attribute.Reorder-
R1,2,3,4,5,6,7,8,9,10,11,12,13,15,16,14-
weka.filters.unsupervised.attribute.StringToNominal-Rlast-
weka.filters.unsupervised.attribute.Reorder-
R1,2,3,4,5,6,7,8,9,10,11,12,14,15,16,13-
weka.filters.unsupervised.attribute.StringToNominal-Rlast-
weka.filters.unsupervised.attribute.Remove-R10-12,14-16'
@attribute Q-E numeric
@attribute ZN-E numeric
@attribute PH-E numeric
@attribute DBO-E numeric
@attribute DQO-E numeric
@attribute SS-E numeric
@attribute SSV-E numeric
@attribute SED-E numeric
@attribute COND-E numeric
@attribute CUMPLE_CON_LA_NORMA_NOM-SEMARNAT-001 {NO_CUMPLE,SI_CUMPLE}
```

4. Filtrado de Datos

Se cuenta con 526 instancias y 10 atributos, enseguida se realizó una limpieza de datos: a) existen valores perdidos en los atributos por lo que fueron sustituidos con la aplicación: (*WEKA-Filter: Unsupervised-Attribute-Replace Missing Value*, filtrar WEKA: atributo no supervisado-reemplazar valor perdido*)* La inclusión citada de los valores perdidos se realizó considerando la media para los valores continuos y la moda para los valores discretos; b) luego se identificaron los valores atípicos y extremos con (*WEKA-Filter*: *Unsupervised-Attribute-IntercuartileRange*, filtrar WEKA: atributo no supervisado-rango intercuartil) enseguida se eliminaron usando (*WEKA-Filter: Unsupervised-Instance-RemoveWithValues*, filtrar WEKA: instancia no supervisada-remover con valores). Finalmente se obtuvieron 493 instancias [15]. Para determinar el atributo nominal se consideró el diagrama mostrado en la Figura 1, el cual indica el límite máximo permitido por la norma para Sólidos Sedimentables, Sólidos Suspendidos Totales y DBO en el efluente.

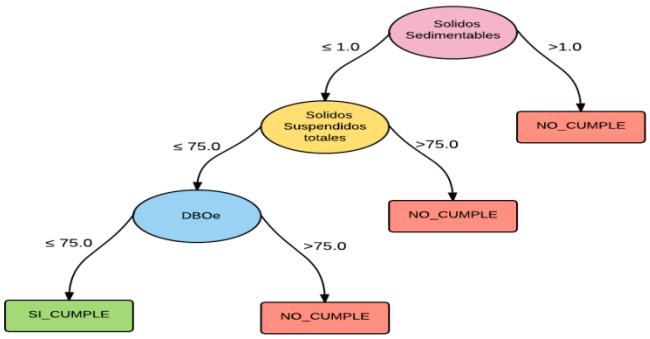


Figura 1. Pre procesado de datos para variable nominal “SÍ_CUMPLE” y “NO_CUMPLE”.

Al sistema *WEKA* se alimentaron los datos de la entrada de la planta y la variable nominal, la cual representa los datos en la salida, lo anterior para que el sistema encuentre la relación que existe entre estos datos y genere el posible resultado conforme a la norma; ya que no tiene caso alimentar el sistema con los datos de salida dado que la relación sería obvia para predecir la variable nominal: se obtendrían solamente los parámetros ya indicados por la norma y ninguna relación entre la entrada y la variable nominal.

De acuerdo con [27], [32], [33], [5] y [47] la Demanda Química de Oxígeno, la Demanda Bioquímica de Oxígeno, el potencial de Hidrógeno y Sólidos Suspendidos Totales, son indicadores muy usados para definir la calidad del agua residual. En la Tabla 2 se muestran los límites máximos permitidos para la descarga de agua residual tratada al cuerpo receptor, como se observa los valores son similares a los indicados en la Figura 1.

Tabla 2. Límites máximos permisibles para contaminantes básicos.

Parámetros	Aguas costeras Recreación (B). Promedio mensual
Sólidos sedimentables (mL/L)	1.0
Sólidos Suspendidos Totales (mg/L)	75.00
Demanda Bioquímica de Oxígeno (mg/L)	75.00

Fuente: [18].

4.1. **Algoritmos de Clasificación.**

Se ingresaron los datos pre procesados al *WEKA* y posteriormente se dividieron las instancias para entrenar el modelo (66%) y para medir su precisión (33%) [41].

Una vez que se delimitó el porcentaje para entrenar el modelo, se utilizaron seis clasificadores incluidos en el sistema *WEKA*: a) árboles de decisión de un nivel; b) clasificador IR (*OneR*); c) tablas de decisión; d) J48; e) *Projective Adaptiva Resonance Theory (PART)* teoría de la resonancia adaptativa proyectiva por sus siglas en inglés y f) *Logistic Model Trees (LMT)* modelo logístico de árboles por sus siglas en inglés. Finalmente se realizó un comparativo para determinar el mejor clasificador. Como ya se indicó el número de instancias, una vez aplicado el tratamiento de datos, fue de 493. Según [45] la instancia se refiere a la muestra que se realiza cada día en donde se miden todos los parámetros, por lo que las instancias corresponden al número de muestras que se tomaron del agua residual en un periodo de tiempo (por día). En la Tabla 3, se muestran las variables que alimentarán al modelo de pronóstico, de igual forma se observan los diferentes clasificadores que serán aplicados, para que posteriormente el sistema WEKA lleve a cabo la predicción.

Tabla 3. Clasificadores y parámetros para la predicción del modelo propuesto

Clasificadores y parámetros					
J48	DecisionStump	OneR	DecisionTable	LMT	PART
Qe	Qe	Qe	Qe	Qe	Qe
ZNe	ZNe	ZNe	ZNe	ZNe	ZNe
PHe	PHe	PHe	PHe	PHe	PHe
DBOe	DBOe	DBOe	DBOe	DBOe	DBOe
DQOe	DQOe	DQOe	DQOe	DQOe	DQOe
SSe	SSe	SSe	SSe	SSe	SSe
SSVe	SSVe	SSVe	SSVe	SSVe	SSVe
SSEDe	SSEDe	SSEDe	SSEDe	SSEDe	SSEDe
Conductividad	Conductividad	Conductividad	Conductividad	Conductividad	Conductividad
Variable nominal	Variable nominal	Variable nominal	Variable nominal	Variable nominal	Variable nominal
Sí cumple	Sí cumple	Sí cumple	Sí cumple	Sí cumple	Sí cumple
No cumple	No cumple	No cumple	No cumple	No cumple	No cumple

(*OneR*): es un algoritmo sencillo, no obstante desarrolla en parte el criterio utilizado por los árboles de decisión. Una característica de este algoritmo es que define el atributo que explica de mejor forma la clase de salida (ver [27]).

Tabla de decisión: se refiere a una matriz de renglones y columnas en donde se mencionan condiciones y acciones. Las reglas de decisión incluidas constituyen el procedimiento que debe seguirse cuando existe un número determinado de condiciones.

Árbol de decisión J48: es una versión afinada del modelo que incluye el *OneR*. El clasificador J48 utiliza el algoritmo C4.5. Éste es uno de los algoritmos más utilizados en los trabajos publicados en minería de datos.

Árbol de decisión de un nivel (*DecisiónStump*) algunas características de esta versión: a) es posible analizar datos categóricos; b) genera árboles de decisión binarios, es decir de un solo nivel ya sea con datos categóricos o numéricos y c) puede ser utilizado para mejorar los métodos [10].
Projective Adaptiva Resonance Theory (PART) teoría de la resonancia adaptativa proyectiva por sus siglas en inglés: no considera el trabajo de optimización global que se incluye en el algoritmo C4.5 y además, define una lista de decisión, pero sin tomar en cuenta restricciones (establece el procedimiento de divide y vencerás). Una característica del *PART* es que determina un árbol de decisión en forma parcial con el fin de determinar una regla. Otro punto importante es que para podar el árbol es necesario que se conozcan todas las consideraciones [10].
Logistic Model Trees (LMT) modelo logístico de árboles por sus siglas en inglés: genera un análisis detallado de los datos, el algoritmo utilizado, construye un árbol de decisión e incluye funciones de regresión [39].

4.2. Precisión por Clase de un Clasificador

De acuerdo con [7], se indican los parámetros de exactitud que incluye el *WEKA*:
Verdadero positivo (*TP* por sus siglas en inglés) se refiere a la magnitud de ejercicios o ejemplos que fueron clasificados como una clase x considerando todos los ejercicios que verdaderamente incluyen la clase x.
Falso positivo (*FP* por sus siglas en inglés) es una parte de ejemplos clasificados de la clase x pero que realmente son de otra clase.
La precisión: también es una parte de ejemplos que en verdad cuentan con la clase x considerando todas las clasificaciones de la clase x.
Medida: se refiere a una medición combinada entre la precisión y la cobertura.

5. Resultados y Discusión

5.1 Prueba con Diferentes Clasificadores

En la Tabla 4 se muestran los resultados obtenidos con los distintos clasificadores aplicados a las mediciones de la calidad del agua en el efluente de la planta de tratamiento.

Tabla 4. Resultados de la ejecución de clasificadores (33% instancias para la predicción)

Algoritmos	Instancias bien clasificadas (%)	Instancias mal clasificadas (%)
J48	83.6364	16.3636
DecisionStump	82.4242	17.5758
OneR	82.4242	17.5758
DecisionTable	**85.4545**	**14.5455**
LMT	83.0303	16.9697
PART	83.6364	16.3636

De acuerdo a la Tabla 4, el algoritmo que mejor clasificó fue la tabla de decisión: 85.45 % de probabilidad de aciertos. Se observa que los resultados de los algoritmos (*DecisionStump y OneR)* son similares; de igual forma el J48 y *PART*. Según [45] es motivo para desechar estos resultados. El algoritmo *LMT* resultó con el segundo lugar en la clasificación. Se analizarán solamente los resultados del algoritmo (*DecisionTable)*.

5.2 Precisión del Clasificador (*DecisionTable)*

La Tabla 5 muestra la precisión del clasificador (*DecisionTable)* por clase, se tiene entonces una media ponderada de 85.50 % para las instancias que resultaron correctamente clasificadas (verdaderos positivos) con

respecto a lo real; para los falsos positivos se obtuvo una media ponderada de 65.50 % de instancias incorrectamente catalogadas y una precisión del clasificador de 85.50 %.

Tabla 5. Precisión detallada por clase

	TP Rate	FPRate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area	Class
	0.993	0.793	0.854	0.993	0.918	0.377	0.763	0.911	NO_CUMPLE
	0.207	0.007	0.857	0.207	0.333	0.377	0.740	0.394	SÍ_CUMPLE
Weighted Avg.	0.855	0.655	0.855	0.855	0.816	0.377	0.759	0.821	

La matriz de confusión (Tabla 6) muestra las instancias clasificadas por clase, así como cuantas son clasificadas correcta e incorrectamente. Se han clasificado 158 instancias como “NO_CUMPLE”, de las cuales 135 fueron correctamente clasificadas y 23 incorrectas. Así mismo se muestra que fueron clasificadas 7 instancias como “SI_CUMPLE”, de las cuales solamente 6 instancias son correctas: la otra en realidad son “NO_CUMPLE”.

Tabla 6. Matriz de confusión con el clasificador Tabla de decisión

Predicción			
No cumple		Sí cumple	
Correcta	135	Incorrecta	1
Incorrecta	23	Correcta	6
Suma	158	Suma	7

5.3 Reglas del clasificador (*DecisionTable)*

La Tabla 7 indica las reglas definidas con el clasificador (*DecisionTable)*, la cual muestra el rango para cada contaminante en la entrada de la planta (ZN-E, DBO-E y SS-E) y el resultado obtenido para la variable nominal (Sí Cumple o No Cumple).

Tabla 7. Reglas para los parámetros de entrada en la planta de tratamiento de aguas residuales. Clasificador *DecisionTable*.

Regla	ZN-E		DBO-E		SS-E		Resultado
	Mínimo	Máximo	Mínimo	Máximo	Mínimo	Máximo	
1	-INF	0,475	153,5	INF	149	INF	NO_CUMPLE
2	0,475	1,715	153.5	INF	149	INF	NO_CUMPLE
3	1,715	INF	153.5	INF	149	INF	NO_CUMPLE
4	-INF	0,475	-INF	153.5	149	INF	SÍ_CUMPLE
5	0,475	1,715	-INF	153.5	149	INF	NO_CUMPLE
6	1,715	INF	-INF	153.5	149	INF	NO_CUMPLE
7	-INF	0,475	153.5	INF	-INF	149	NO_CUMPLE
8	0,475	1,715	153.5	INF	-INF	149	NO_CUMPLE
9	1,75	INF	153.5	INF	-INF	149	NO_CUMPLE
10	-INF	0,475	-INF	153.5	-INF	149	SÍ_CUMPLE
11	1,715	INF	-INF	153.5	-INF	149	NO_CUMPLE
12	0,475	1,715	-INF	153.5	-INF	149	SÍ_CUMPLE

Según *WEKA* el clasificador (*DecisionTable*) analiza el subconjunto de atributos con el algoritmo (*Best First)*, y selecciona los tres atributos mostrados en la Tabla 7. Se descarta el resto de los atributos que se alimentaron al inicio, de esta manera se depuran los datos en la entrada para simplificar las reglas resultantes. La Tabla 8 muestra las tres reglas con el resultado Sí Cumple. El resto de los casos resultaron No Cumple.

Tabla 8. Reglas con resultado Sí Cumple

Regla 4	Regla 10	Regla 12
Si:	Si:	Si:
ZN-E≤0,475	ZN-E≤0,475	0,475≤ZN-E≤1,715
DBO-E≤153,5	DBO-E≤153,5	DBO-E≥153,5
SS-E≥149	SS-E≤149	SS-E≤149
Entonces: Sí Cumple	Entonces: Sí Cumple	Entonces: Sí Cumple

En la Tabla 8 se observa que la regla 4 y 10 incluyen las mismas condiciones para ZN-E y DBO-E y además, la condición para SS-E es inversa entre ambas reglas, por lo tanto, se puede eliminar esta última condición. La Tabla 9 muestra las reglas resultantes.

Tabla 9. Reglas simplificadas con resultado Sí Cumple

Regla 4 y 10	Regla 12
Si:	Si:
ZN-E≤0,475	0,475≤ZN-E≤1,715
DBO-E≤153,5	DBO-E≥153,5
	SS-E≤149
Entonces: Sí Cumple	Entonces: Sí Cumple

Es importante aclarar que lo indicado en la Tabla 2 es el criterio para el pre procesado de los datos, y las reglas presentadas en la Tabla 9 tienen la finalidad de determinar si se cumple con los criterios indicados. Según los resultados de la Tabla 5 al aplicar la predicción se cumple con un 85.5% de precisión en el pronóstico de la variable nominal. Es importante indicar que las reglas mostradas en la Tabla 9 no corresponden a ecuaciones de predicción numérica, sino nominal, por lo que no es posible comparar la norma de la Tabla 2 con las reglas para la variable nominal.

De acuerdo con [24] cuando las reglas que conducen a una variable (sí o no) las condiciones no entran en conflicto, por lo que no existe ambigüedad en la interpretación de las reglas.

En la revisión de literatura realizada no se encontraron antecedentes de estudios similares al presente; es decir, donde fueran consideradas el uso de variables nominales. Por tal motivo en la discusión se consideraron trabajos con diferentes metodologías pero que también predicen la calidad del agua residual.

[29], [28] y [44] utilizaron para la predicción análisis estadísticos, series de tiempo y minería de datos; mientras que en el presente estudio se aplicó exclusivamente la clasificación de minería de datos. Una diferencia importante respecto a la investigación de [29], [34], [37] y [44] fue la determinación de reglas para favorecer el cumplimiento de los límites máximos permitidos, para la descarga de agua residual a los cuerpos receptores. Los criterios y resultados son diferentes, no obstante lo anterior, los estudios indicados son capaces de predecir la calidad del agua residual en el efluente de los sistemas de tratamiento. [25] publicó un estudio para predecir la demanda bioquímica de oxígeno en el efluente de una planta de tratamiento de aguas residuales, utilizando redes neuronales con 5 parámetros: DQO, SS, Conductividad Eléctrica (CE), Temperatura y Potencial de Hidrógeno. El estudio consistió en una comparación de resultados: redes neuronales y regresión lineal múltiple. Se obtuvieron mejores resultados con las redes neuronales; mientras que en el estudio propuesto sólo se aplicó la minería de datos considerando seis algoritmos, donde se obtuvieron tres importantes condiciones para la predicción de la calidad del agua residual. Es importante

destacar la simplicidad del modelo propuesto respecto a los indicados en el apartado de revisión de la literatura.

6. Conclusiones

Con el criterio del pre procesado de datos y el uso de variables nominales fue posible definir reglas para la predicción de la calidad en el efluente del sistema de tratamiento. Además, es posible utilizar el modelo para identificar los principales contaminantes que intervienen en la calidad final del agua, y de esta forma localizar a las empresas contaminantes con el fin de realizar un adecuado control de descargas. Es decir contar con elementos para la toma de decisiones: si es prudente activar o no el programa de control de descargas de aguas residuales de procesos industriales, comerciales y de servicios.

Los resultados del presente trabajo pueden tomarse como fundamento para la toma de decisiones. Por ejemplo, el citado programa de control ambiental necesita de grandes recursos económicos para la verificación del total de los procesos industriales que utilizan agua en su proceso. Las reglas obtenidas identifican únicamente los contaminantes críticos que deberán supervisarse, lo anterior restringe la utilización de recursos económicos en la verificación de las descargas de aguas residuales.

Es importante indicar que la aportación del presente trabajo fue el uso de variables nominales para el pronóstico de la calidad del agua residual con una precisión del 85.5%.

La minería de datos es una herramienta que favorece la generación de reglas para predecir la calidad del agua residual en la salida del sistema de tratamiento, por lo cual puede aplicarse en trabajos futuros a sistemas de lagunas de oxidación y como una herramienta auxiliar para el diseño de plantas de tratamiento y toma de decisiones.

Los resultados del proceso dependerán de los límites máximos permitidos indicados en la Tabla 2. La predicción que arroja el modelo es útil cuando se cuenta con los datos de entrada y se puede saber de antemano si se cumplirá o no con la norma.

El criterio de análisis propuesto en el presente estudio puede usarse en cualquier planta de tratamiento de aguas residuales, siempre y cuando se cuente con la base de datos del sistema.

Reconocimientos

Se agradece al Programa para el Fortalecimiento de la Calidad Educativa (PFCE) 2017 por los recursos asignados para la realización del presente estudio.

Referencias

[1] Arévalo1, J. L. y García, R. P. Estado del arte en la utilización de técnicas avanzadas para la búsqueda de información no trivial a partir de datos en los sistemas de abastecimiento de agua potable, 2008. www.lenhs.ct.ufpb.br/html/downloads/serea/trabalhos/A15_15.pdf. (accessed December, 29, 2016).

[2] Atasoy, A.D., Babar, B., Sahinkaya, E. Artificial neural network prediction of the performance of upflow and downflow fluidized bed reactors treating acidic mine drainage water. Mine water and the environment. New York, Springer, vol.1, 222–228, 2013.

[3] Ay, M., and Kisi, O. Modelling of chemical oxygen demand by using ANNs, ANFIS and k-means clustering techniques. Journal of Hydrology, 511, 279-289, 2014.

[4] Belanche, L., Sànchez, M., Cortés, U., and Serra, P. A knowledge-based system for the diagnosis of wastewater treatment plants. In Industrial and Engineering Applications of Artificial Intelligence and Expert Systems. Springer Berlin Heidelberg, 324-336, 1992.

[5] Bernard, O., Hadj-Sadok, Z., Dochain, D., Genovesi, A., and Steyer, J. P. Dynamical model development and parameter identification for an anaerobic wastewater treatment process. Biotechnology and Bioengineering, 75(4), 424–438, 2001.

[6] Béjar, J., Cortés, U., and Poch, M. Linneo, A classification methodology for illstructured domains. Facultat d'Informtica de Barcelona, Tech. Rep., 1993.

[7] Bouckaert, R. R., Frank, E., and Hal, M. WEKA manual for version 3-7-8, 2013.

[8] Cărbureanu, M. Pollution Level Analysis of a Wastewater Treatment Plant Emissary using Data Mining. Petroleum-Gas University of Ploieşti Bulletin Mathematics-Informatics-Physics Series, 62(1), 69-78, 2010.

[9] Carpe Cánovas, J. Monitorización de la eficiencia de depuración de una planta de tratamiento de agua residual urbana mediante el empleo de técnicas de análisis estadístico multivariante. Tesis of Master degree, 2015.

[10] Césari, M. Aprendizaje automático con WEKA, 2007.

[11] Comisión Nacional del Agua. Guía para el control de descargas a los sistemas de alcantarillado urbano o municipal. México2000a.

[12] Comisión Nacional del Agua. Manual de diseño de agua potable, alcantarillado y saneamiento. Manual de diseño de lagunas de estabilización. Instituto Mexicano de Tecnología del Agua, Jiutepec, México. 234, 2007b. http://www.conagua.gob.mx/conagua07/publicaciones/publicaciones/Libros/10DisenoDeLagunasDeEstabilizacion.pdf

[14] Cortés-Martínez, F., Treviño-Cansino, A., Sáenz-López, A., and Narayanasamy, R. Statistical analysis for the characterization of the wastewater in the influent of a treatment plant (Case of study). International Journal of Engineering and Technical Research, Vol. 4 (1). 103-110, 2016.

[15] Cortés-Martínez, F., Espinoza-Fraire, A.T., Sáenz-López. A. y Narayanasamy. Limpieza y descripción gráfica de datos con WEKA, *Memorias del Congreso Internacional de Investigación Academia Journals CICS*, Tuxpan 2017, Vol. 9, No. 4, 405-410, 2017.

[16] Chen, W.C., Chang, N.B., Shieh, W.K. Advanced hybrid fuzzy-neural controller for industrial wastewater treatment. Journal of Environmental Engineering 127 (11), 1048–1059, 2001.

[17] De Garcia Blanco, J. Avaluació de tècniques de classificació per a la gestió de bioprocessos: aplicació a un reactor de fangs activats, 1993.

[18] Diario Oficial de la Federación, (DOF). Norma Oficial Mexicana NOM-001-ECOL-1996, que establece los límites máximos permisibles de contaminantes en las descargas de aguas residuales a los sistemas en aguas y bienes nacionales, 1997. https://books.google.com.mx/books?id=N-5WAAAAMAAJ

[19] Enviromental Protection Agency. Guidance Manual for Preventing Interference at POTWs. USA, 1987.

[20] EPA 1991 Environmental Protection Agency Control de Descargas Irregulares Hacia las POTWs. USA.

[21] EPA 1994 Environmental Protection Agency, Guía, procedimientos y pautas recomendadas para establecer e implementar un programa de pretratamiento. USA.

[22] Grossman, R., Kasif, S., Moore, R., Rocke, D., Ullman, J., Data mining research: opportunities and challenges, A report of three NSF workshops on mining large, massive, and distributed data, September 18, 1998.

[23] Guclu, D. and Dursun, S. Artificial Neural network modelling of a large-scale wastewater treatment plant operation. Bioprocess Biosystems Engineering, vol 33, no. 9, 1051-1058, 2010.

[24] Hall, M., Witten, I., and Frank, E. , Data mining: Practical machine learning tools and techniques. Kaufmann, Burlington, 2011.

[25] Heddam, S., Lamda, H., and Filali, S. Predicting effluent biochemical oxygen demand in a wastewater treatment plant using generalized regression neural network based approach: a comparative study. Environmental Processes, 3(1), 153-165, 2016.

[26] Henze, M., Harremes, P., Jansen, J. L. C., and Arvin, E. Wastewater treatment: biological and chemical processes. New York: Springer, 2001.

[27] Jiménez, M. G., and Álvarez, A. Análisis de datos en WEKA–pruebas de selectividad, 2010. http://www.it.uc3m.es/jvillena/irc/practicas/06-07/28.pdf

[28] Kaur I. Cluster Analysis on various samples of water pollution. American International Journal of Research in Science, Technology, Engineering and Mathematics, 4(2), 149-153, 2013.

[29] Kusiak, A., Verma, A., and Wei, X. A data-mining approach to predict influent quality. Environmental monitoring and assessment, 185(3), 2197-2210, 2012.

[30] Lichman, M. UCI Machine Learning Repository. Irvine, CA: University of California, School of Information and Computer Science, 2013. https://archive.ics.uci.edu/ml/datasets/Water+Treatment+Plant

[31] Lindqvist, J., Wik, T., Lumley, D., Aijala, G. Influent Load Prediction Using low Order Adaptive Modeling. In: Proceedings of the 2nd IWA Conference on Instrumentation, Control and Automation, Busan, South Korea, 2005.

[32] Mamais, D., Jenkins, D., and Pitt, P. A rapid physical–chemical method for the determination of readily biodegradable soluble COD in municipal wastewater. Water Research, Biological Wastewater Treatment, IWA Publisher, 27(1), 195–197, 1993.

[33] Metcalf and Eddy, Inc.,Wastewater Engineering: Treatment, Disposal, Reuse. McGraw-Hill, New York, 1991.

[34] Mjalli, F. S., Al-Asheh, S., and Alfadala, H. E. Use of artificial neural network black-box modeling for the prediction of wastewater treatment plants performance. Journal of Environmental Management, 83(3), 329-338, 2007.

[35] Olaru, C.; Wehenkel, L. “Data Mining”. IEEE Computer Applications in Power, Volume 12, Number 3, 19-25, 1999.

[36] Pai, T. Y., Yang, P. Y., Wang, S. C., Lo, M. H., Chiang, C. F., Kuo, J. L., and Chang, Y. H. Predicting effluent from the wastewater treatment plant of industrial park based on fuzzy network and influent quality. Applied Mathematical Modelling, 35(8), 3674-3684, www.journals.elsevier.com/applied-mathematical-modelling. 2011.

[37] Patricia, K., Esquerrea, O., Seborgb, D.E., Moria, M., Bruns, R.E. Application of steady state and dynamic modeling for the prediction of the BOD of an aerated lagoon on a pulp and paper mill. Part II-nonlinear approaches. Chem. Eng. J. Nova Science Publishers, Inc, New York. 105 (5), 61–69, 2004.

[38] Qasim, S. R. Wastewater treatment plants: planning, design, and operation. New York, CRC Press, 1998.

[39] Romero Beltrán C.A. Interfaces de usuario “Explorer” y “Knowledge” Flow en Weka” Universidad Nacional de Colombia, p. 39, 2014.

[40] Sànchez-Marrè, M., Gibert, K., and Rodríguez-Roda, I. GESCONDA: A tool for knowledge discovery and data mining in environmental databases. Research on Computing Science. Centro de Investigación en Computación, Instituto Politécnico Nacional, México DF, México, 11, 348-364, 2004.

[41] Smith, T. C., and Frank, E. Introducing Machine Learning Concepts with WEKA. Statistical Genomics: Methods and Protocols, 353-378, 2016.

[42] Tay, J.H., Zhang, X. Neural fuzzy modeling of anaerobic biological wastewater treatment systems, J. Environ. Eng. ASCE, 125 (12) 1149–1159, 1999.

[43] Universidad Nacional Autónoma de México. Curso uso eficiente del agua y control de calidad de las descargas de aguas residuales en la industria. México, 2000.

[44] Verma, A., Wei, X., and Kusiak, A. Predicting the total suspended solids in wastewater: a data-mining approach. Engineering Applications of Artificial Intelligence, 26(4), 1366-1372, 2013.

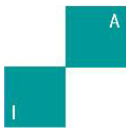
[45] Waikato Environment for Knowledge Analysis (WEKA) (undated). Minería de datos. Recovered from: http://www.it.uc3m.es/~jvillena/irc/practicas/03-04/18.mem.pdf

[46] Yu, H. Q., and Fang, H. P. Acidogenesis of gelatin-rich wastewater in an up flow anaerobic reactor: influence of pH and temperature. Water Research, 37(1), 55–66, 2003.

[47] Zhang, Q. and Stanley, S. Real-Time Water Treatment Process Control with Artificial Neural Networks. J. Environ. Eng., 153-160, 1999.

[48] Zhu, J., Zurcher, J., Rao, M., Meng, M.Q.H. An on-line wastewater quality prediction system, based on a time delay neural network. Eng. Appl. Artif. Intell. 11 (2), 747–758, 1998.

Inteligencia Artificial, 21(62) (2018), 25-39
doi: 10.4114/intartif.vol21iss62pp25-39



INTELIGENCIA ARTIFICIAL

http://journal.iberamia.org/

A Constraint-based Mission Planning Approach for Reconfigurable Multi-Robot Systems

Thomas M. Roehr
DFKI GmbH Robotics Innovation Center, Bremen, Germany
thomas.roehr@dfki.de

Abstract The application of reconfigurable multi-robot systems introduces additional degrees of freedom to design robotic missions compared to classical multi-robot systems. To allow for autonomous operation of such systems, planning approaches have to be investigated that cannot only cope with the combinatorial challenge arising from the increased flexibility of modular systems, but also exploit this flexibility to improve for example the safety of operation. While the problem originates from the domain of robotics it is of general nature and significantly intersects with operations research. This paper suggests a constraint-based mission planning approach, and presents a set of revised definitions for reconfigurable multi-robot systems including the representation of the planning problem using spatially and temporally qualified resource constraints. Planning is performed using a multi-stage approach, and a combined use of knowledge-based reasoning, constraint-based programming and integer linear programming. The paper concludes with the illustration of the solution of a planned example mission.

Keywords: Reconfigurable, Multi-Robot, Constraint-based, Vehicle Routing Problem

1 Introduction

Flexibility is the primary feature of reconfigurable multi-robot systems, since their modularity adds an additional degree of freedom to design robotic operations compared to the application of traditional multi-robot systems. For that reason Dignum et al. [11] discuss the so-called strategic flexibility, which offers an exploitation of proactive and reactive adjustment in the context of reconfigurable organizations. The strategic flexibility allows to tackle a set of unforeseen tasks with a robustly equipped system that allows recovery from malfunction thanks to increased redundancy. Exploiting strategic flexibility provides a strong motivation to combine increasingly capable autonomous robotic systems with a concept for modularity.

The main benefit of reconfigurable multi-robot systems lies in the fact that resources can easily, although not arbitrarily, be (re)used by any agent being a member of the reconfigurable system. Using this flexibility allows to balance resource usage and hence to adapt dynamically to operational demands. While modularity can lead to significant operational advantages it has drawbacks: if the level of modularity is chosen arbitrarily high this can lead to less capable systems. One can observe the effect in swarm-based systems, which come with a high degree of modularity: a swarm typically consists of cheap agents with a simple design, and thus , apart from emergent high-level behaviors, these agents come with a rather limited applicability by design. Although the mentioned emergent behaviors can be exploited, these behaviours remain harder to control or will be focused on a single task only. In general, reconfigurable multi-robot systems offer a feasible solution which consists of a mix of individually capable agents, including swarm-like units that can augment the overall robotic team. This augmentation is done either by acting as a fully autonomous agent or as an extension unit. Roehr et al. [17] implement this idea in the context of robotic space exploration missions in order to show the general feasibility and identify critical limitations: the

ISSN: 1137-3601 (print), 1988-3064 (on-line)

implemented approach validates the potential for increasing the flexibility in future robotic missions, but it also comes with increased operational demands. Thus, they suggest the introduction of a dedicated system model in order to automate operation of reconfigurable multi-robot systems and exploit the offered system capabilities to improve not only efficiency, but also safety of future robotic missions. Roehr and Kirchner [19] show how planning as essential element for automated operation for such a reconfigurable multi-robot system can be approached.

This paper details the problem definition and presents the results of the continued development of the planning approach. In Section 2 we briefly outline relevant background references for the state of the art. Section 3 introduces the planning problem, and in Section 4 we give details on the organization model and its extended use. Section 5 outlines the revised planning approach. We close with a conclusion and outlook in Section 6.

2 Background

The initial motivation for the planning problem is given by Sonsalla et al. [21], where a reconfigurable multi-robot system shall establish a logistics chain operation in order to support sample-return missions as part of extraterrestrial exploration. The robotic team consists of mobile and immobile agents, which can be physically connected via a set of electro-mechanical interfaces. By connecting one or more robots, they can form a new type of agent, comprising features none of the individual agents offers. The ability for reconfiguration offers novel ways of dealing with robotic missions, but Roehr and Kirchner [19] is to our best knowledge the only approach particularly dealing with reconfigurability. This mission planning problem can be understood as a logistic planning problem where mobile robots can transport other immobile and mobile robots. Hence, it is closely related to the Vehicle Routing Problem (VRP) [22]: a fleet of (most often homogeneous) mobile vehicles shall serve a set of customers, e.g., by delivering and/or picking up items, while minimizing a cost function – typically the overall travelled distance. The VRP applies to transportation and logistics scenarios and comes in many variants among which Capacitated VRP (CVRP), VRP with Time Windows (VRPTW) and VRP with Pick-up and Delivery (VRPPD) are the most popular ones. The pickup-and delivery problem can be further distinguished into a many-to-many (M-M), one-to-many-to-one (1-M-1), and one-to-one (1-1) problems, where the notation can be read as cardinalities for the origin, transition point, and target of a commodity, i.e. from-to or from-via-to. The M-M variant for example accounts for multiple commodity (good) origins and destinations, while the 1-M-1 variants assume a start and end of all vehicles at a single depot. The majority of these approaches are either focusing on a single-commodity case, homogeneous vehicle capacities or optimization of routing cost, were our approach has to deal with multi-commodities, heterogeneous vehicles, multi-depots, and fleet size optimization. Hence, a closer relation can be established to more specialized VRP approaches, e.g., such as the Heterogeneous or mixed Fleet VRP (HFVRP) [5] which accounts for a heterogeneous fleet and optionally with unlimited vehicle availability, or Dondo et al. [12] who approach Multi-depot heterogeneous fleet VRP with time windows (MDHFVRPTW). The variant VRP with Trailers and Transshipments (VRPTT) and more generalized VRP with multiple synchronization constraints (VRPMSs) [13] adds synchronization constraints between vehicles, which form a special instance of a reconfigurable multi-agent system containing agents or in this case vehicles of different categories: autonomous and non-autonomous, as well as support and task vehicles. Drexl [13] formulates a graph-based modelling approach to account for the interdependence of vehicles. He does not, however, provide an implementation of a solution approach.

While much of the research in VRP originates from the area of operational research, Coltin and Veloso [6, 7, 8] investigate a pick-up and delivery variant in the context of multi-robot systems and also apply their approach to a taxi problem with ridesharing. They implement optimal approaches as well as meta-heuristics, in particular simulated annealing, and Very Large Neighborhood Search (VLNS) [3] their application of VLNS results not only in a scalable approach, but also proves a general benefit of using transfers in a pickup and delivery scenario. In Section 3 we will outline the distinction between existing VRP and our approach, and provide additional constraints for our mission planning problem.

3 Mission Representation

The planning problem presented in the following aims to solve the problem of planning and scheduling a mission performed by a reconfigurable multi-robot system. While a mission can initially be seen as a task assignment for a multi-robot system, here it comes with an essential difference: agents are able to dynamically form physical coalitions referred to as composite agents. These composite agents are formed for three main reasons. Firstly,

to perform agent transport: one carrier agent attaches one or multiple (most likely, but not necessarily) immobile systems. Secondly, to provide functionality: some functionality is only available as so-called super-additive effect and requires two or more agents to join so that this functionality becomes available only for this composite agent, but not for the individual agents. Thirdly, to increase the functional redundancy: for agents that are assigned to fulfil requirements, we assume that adding relevant resources improves the redundancy and safety of operation, and effectively the likelihood of a successful performance of an agent.

While most VRP assume homogeneous agents, the team of agents in a reconfigurable multi-robot system is formed by heterogeneous agents; agents with individual capabilities and functionalities, as well as limitations to reconfigure and constraining attributes such as an overall transport capacity. We will look at a mission as a particular (minimal) partitioning problem of an agent team to achieve a requested agent- and function-distribution over space and time.

3.1 Definitions & Assumptions

In the following we introduce the basic notation, definitions and assumptions regarding reconfigurable multi-robot systems. The provided definitions describe a modular multi-agent system, which can form composite agents from a set of available agents:

Definition 3.1. *An **atomic agent** a represents a monolithic physical robotic system, where $A = \{a_1, \ldots, |A|\}$, is the set of all atomic agents, and $a \in A$ or equivalently $\{a\} \subseteq A$.*

Definition 3.2. *A mechanically coupled system of two or more atomic agents is denoted a **composite agent** CA, where $CA \subseteq A$, and $|CA| > 1$.*

Definition 3.3. *The type of an atomic agent a is denoted $\widehat{a}$ and equivalently for a composite agent CA the type is denoted $\widehat{CA}$. The set of all agent types is denoted $\theta(A) = \{1, \ldots, |\theta(A)|\}$, with the corresponding type-partitioned sets $A^1, \ldots, A^{|\theta(A)|}$, where $A = A^1 \cup \cdots \cup A^{|\theta(A)|}$.*

Definition 3.4. *A (general) agent is denoted GA, where $GA \subseteq A$, and $GA \neq \emptyset$. A (general) agent represents the wrapping concept for atomic and composite agents, with the corresponding type-partitioned sets $GA^1, \ldots, GA^{|\theta(A)|}$, where $GA = GA^1 \cup \cdots \cup GA^{|\theta(A)|}$*

Definition 3.5. *A (general) agent type $\widehat{GA}$ will be represented as a function $\gamma_{\widehat{GA}} : \theta(A) \to \mathbb{N}_0$, which maps an atomic agent type $\widehat{a}$ to the cardinality $c_{\widehat{a}}$ of the type partition, such that $c_{\widehat{a}} = |GA^{\widehat{a}}|$. The set of all constructible general agent types from a set of atomic agents A is denoted $\theta(\widehat{A})$; it represents the collection of all general agent types that are found in the powerset of all agents $\mathcal{P}^A$.*

Note, that a general agent type can equivalently be represented as tuple set of agent type and type cardinality: $\widehat{GA} = \{(\widehat{a}_1, c_1), \ldots, (\widehat{a}_n, c_n)\}$, where $a_i \in A$ and $c_i = |GA^i|$. $\widehat{GA} \supseteq \widehat{GA'} \iff \forall (a_i, c_i) \in \widehat{GA}, (\widehat{a}_i, c'_i) \in \widehat{GA'} : c_i \geq c'_i$, where $i = 1 \ldots |A|$.

Definition 3.6. *A set of atomic agents A is denoted an **agent pool** and it can be represented by a general agent type $\widehat{GA}$, such that $\forall a \in A : \gamma_{\widehat{GA}}(\widehat{a}) = |A^{\widehat{a}}|$.*

Definition 3.7. *An **atomic agent role** $r^{\widehat{a}}$ represents an anonymous agent instance of an atomic agent type $\widehat{a}$.*

Definition 3.8. *A coalition structure of an agent set A is denoted CS^A and is represented by a set of disjunct general agents $CS^A = \{GA_0, \ldots, GA_n\}$, where $GA_0 \cup \cdots \cup GA_n = A$, and $\forall i, j \wedge i \neq j : GA_i \cap GA_j = \emptyset$.*

3.2 Assumptions

Our design of the organization model and planning system for a reconfigurable multi-robot system, which both will be detailed in the following section, is based on a set of assumptions to simplify the modelling approach.

Assumption 3.1. *Each atomic and composite agent can be mapped to a single agent type only.*

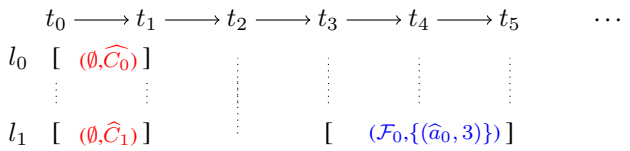


Figure 1: A mission specification example based on a space-time representation

A reconfigurable multi-robot system requires coupling interfaces, e.g., an electro-mechanical interface [10], to create physical linking between atomic agents to establish a composite system. Although multiple links could be considered between any two agents, interfaces cannot be arbitrarily coupled and the following assumption holds:

Assumption 3.2. *A mechanical coupling between two atomic agents can only be established through a single link and two and only two compatible physical coupling interfaces.*

3.3 Mission specification

The mission specification is a temporal database description, and it defines the initial, intermediate and goal state for a reconfigurable multi-robot system. A valid mission specification is described by the following two definitions:

Definition 3.9. *A spatio-temporal requirement is a spatio-temporally qualified expression (stqe) s which describes the functional requirements and agent instance requirements for a given time-interval and a particular location: $s = (\mathcal{F}, \widehat{GA}_r)@(l,[t_s,t_e])$, where $\mathcal{F}$ is a set of functionality constants, $\widehat{GA}_r$ is the general agent type representing the required atomic agent type cardinalities, $l \in L$ is a location variable, and $t_s, t_e \in T$ are temporal variables describing a temporal interval with the implicit constraint $t_s < t_e$. Variables associated with s will also be referred to using the following notation: $\mathcal{F}^s$,$\widehat{GA}_r^s$,l^s,t_s^s, and t_e^s.*

Definition 3.10. *The robotic mission is a tuple $\mathcal{M} = \langle \widehat{GA}, STR, \mathcal{X}, \mathcal{OM} \rangle$, where the agent pool $\widehat{GA}$ describes the available set of agents, STR is a set of spatio-temporally qualified expressions, $\mathcal{X}$ is a set of constraints, and $\mathcal{OM}$ represents the organization model.*

The initial state is defined by the earliest timepoint and binds available agents to their starting depot. The earliest timepoint is $t_0 \in T$ and $\forall t \in T, t \neq t_0 : t > t_0$. Figure 1 illustrates a mission specification, where

$$\begin{aligned}
\widehat{GA} &= \{(\widehat{a}_0,3),(\widehat{a}_1,2)\},\\
STR &= \{(\emptyset,\widehat{C_0})@(l_0,[t_0,t_1]),(\emptyset,\widehat{C_1})@(l_1,[t_0,t_1]),\\
&\quad (\mathcal{F}_0,\{(\widehat{a}_0,3)\})@(l_1,[t_3,t_5])\}\\
\mathcal{X} &= \{t_0 < t_1,\ldots,t_4 < t_5\}\\
\mathcal{OM} &= \{mobile(\widehat{a}_0),\neg mobile(\widehat{a}_1),\ldots\}
\end{aligned}$$

$\widehat{C}_0 = \{(\widehat{a}_0,2),(\widehat{a}_1,1)\}$, $\widehat{C}_1 = \{(\widehat{a}_0,1),(\widehat{a}_1,1)\}$, l_0, l_1 are location variables and $t_0,\ldots,t_5$ are timepoint variables. Two general agents $\widehat{C}_0$ and $\widehat{C}_1$ are assigned to location l_0 and l_1 respectively. Two stqes related to the interval $[t_0,t_1]$ define the initial agent assignments; no functional requirements are part of the initial state description. The goal state is defined over the interval $[t_3,t_5]$ and requires a functionality set $\mathcal{F}_0$ in combination of least 3 agents of type $\widehat{a}_0$ at location l_1.

3.4 Mission constraints

A mission can be detailed by constraints in the constraint set $\mathcal{X}$. The only initially required constraints are temporal ones to describe the starting state, e.g., in the presented example all stqes relating to a start at t_0, e.g., cardinality constraints allow to set upper and lower bounds on the usage of agents and functionalities to reduce the combinatorial challenge. Other optional constraints can be added to detail and constrain the evolution of a mission. The following list describes the available constraint types; minimum constraints come with a corresponding max constraint implementation:

temporal qualitative timepoints describe time intervals, where timepoint constraints are provided using point algebra ($<, >, =$) [9].

duration $minDuration(s,t)$, $s \in STR$: sets a lower bound of time t for the duration of the time interval associated with the stqe s.

min cardinality $minCard(s, \widehat{a}, c_{min})$, $s \in STR$: represents a minimum cardinality constraint so that $|GA^{\widehat{a}}| \geq c_{min}$

all distinct $allDistinct(S, \widehat{a})$ describes the constraint: $\forall s \in S : \bigcap A^{\widehat{a},s} = \emptyset$, where $S \subseteq STR$, and $A^{\widehat{a},s}$ represents the subset of agents of type $\widehat{a}$ which are associated with the stqe s.

min distinct $minDistinct(S, \widehat{a}, n)$ describes the constraint: $\forall s_i, s_j \in S, i \neq j : \left||A^{\widehat{a},s_i}| - |A^{\widehat{a},s_j}|\right| \geq n$, where $n \geq 0$, $S \subseteq STR$, and $A^{\widehat{a},s}$ represents the partition of A which contains only agents of type $\widehat{a}$ which are associated with the stqe s.

all equal $allEqual(S, A_e)$ describes the constraint: $\forall s \in S \exists A_e : A_e = A_r^s$, where $A_e \subseteq A$, $S \subseteq STR$.

min equal $minEqual(S, A_e)$ describes the constraint: $\forall s \in S \exists A_e : A_e \subset A_e^s$, where $A_e \subset A$, $S \subset STR$.

min-function $minFunc(s, f)$: requirement for a functionality f to be available at stqe s: $f \in \mathcal{F}^s$

min-property $minProp(s, f, p, n)$ constrains the property p_f of a functionality f to be $p_f \geq n$, where the constraint implies $minFunc(s, f)$

To handle service preferences within this representation, e.g., when a particular agent should visit two distinct locations, equality constraints are required. An equality constraints can define partial or full paths for the same instances of agents, e.g., to control that the same agent visiting location l_0 at timepoint t_0 will also visit location l_1 at t_1. Detailing functionality request with min and max property constraints are motivated by informed repair strategies, e.g., a property constraint can demand a mobile agent with a particular transport capacity. In Section 4 we will detail this reasoning further.

3.5 Distinction & Observation

Existing VRP based approaches most often only consider a subset of the presented constraints, while the mission planning problem formulation embeds the following VRP properties: time windows, capacity constraints, heterogeneous agents, fleet size minimization and vehicle synchronization. Furthermore, additional special features are introduced: (i) it is not only accounted for commodity demand, but rather a combination of commodities and vehicles that provide certain functional properties; (ii) the use of qualitative temporal constraints (in contrast to hard or soft quantitative time windows), which enables partially ordered requirements and increase the flexibility to synchronize agent activities; (iii) the mix-in of a multi-pickup multi-delivery problem in contrast to a single drop-off.

4 Organization Modeling

To reason upon a reconfigurable multi-robot system a special so-called organization model is introduced which describes all resources that can be part of a reconfigurable multi-robot team: atomic agents as well as their functionalities and properties thereof. As detailed in [19] the organization model builds upon an ontological description, which: (a) encodes information about resources that are associated with agent types, (b) associates interfaces with agent types, (c) defines compatibility between interfaces, (d) allows the identification of feasible, and (e) allows inferencing functionality of composite agents.

In combination of all features the organization model serves as main reasoner to identify composite agents and coalition structures, which are suitable to support a set of time and location bounded functional requirements.

The following sections will describe agent properties, and the details of identifying feasible composite agents, and subsequently suitable agent with respect to a given functionality.

4.1 Atomic agent type

Each agent type is associated with the following essential attributes:

mobility $mobile(\widehat{a})$ defines whether an agent of type $\widehat{a}$ is mobile or not.

transport capacity $tcap(\widehat{a})$ defines the maximum total capacity (measured in storage units) of an agent of type $\widehat{a}$ to transport others, and $tcap(\widehat{a_i}, \widehat{a_j})$ defines the maximum capacity of an agent type $\widehat{a_i}$ to transport an agent type $\widehat{a_j}$.

capacity consumption $tcon(\widehat{a})$ defines the number of storage units an agent of type $\widehat{a}$ consumes temporarily when being transported (currently this is set to 1 by default);

velocity $v_{nom}(\widehat{a})$ defines the nominal velocity of an agent type $\widehat{a}$, $v_{nom} \geq 0$ for mobile atomic agent types and $v_{nom} = 0$ for immobile

power $pw(\widehat{a})$ defines the nominal required power to operate an agent of type $\widehat{a}$

mass $mass(\widehat{a})$ defines the mass of an agent

energy $energy(\widehat{a})$ defines the available electrical energy that initially comes with an atomic agent

4.2 General and composite agent type

Some properties of composite agents can be inferred from their compositing atomic agents: Avella et al. [4] (though in the context of route constraints) label these as 'numerical totalisable', e.g., here mass and energy, which can be easily represented as sum of the property values of each atomic agent forming the composite agent. Inferring the capacity, in contrast, can be complex due to geometrical packaging constraints. The present model, however, currently ignores geometrical packing constraints and checks only connectivity based on interface compatibility.

4.3 Feasible agents

The main feature of a reconfigurable multi-robot system is the possibility for physical interconnection, but the not all composite agents are feasible. The compatibility and availability of connecting interfaces can restrict the design of a fully connected composite agent. Interfaces can come in different variants, e.g., for the reference system in [18] a male and female (also referred to as *EmiPassive* and *EmiActive*). But only one male and one female interface can be coupled. Atomic agents can comprise any number of interfaces, but based on Assumption 3.2 exactly one interface can be used for the connection to another agent's interface. For a successful connection, both interfaces need to be compatible.

Checking feasibility is a matching problem for graph $G = (V, E)$, with constraints for the existence of edges, where a vertex $v \in V$ represents a single interface. We denote I^A as the set of all interfaces of a set of agent A, so that $V = I^A$ with the corresponding partitioning $I^A = I^0 \cup I^1 \cup \ldots I^n$, where $n = |A| - 1$ and the set of interfaces of an agent a_0 is represented as $I^0 = \{i_{0,0}, i_{0,1}, \ldots, i_{0,|I^0|-1}\}$. The adjacency matrix is an $m \times m$ Matrix C, where $m = |I^A|$, and $\forall i, j \in I^A : c_{i,j} = 0, 1$ (rows and columns are annotated with the interface):

$$
\begin{array}{c}
 \\
i_{0,0} \\
i_{0,1} \\
\vdots \\
i_{n,|I^n|}
\end{array}
\begin{array}{c}
\begin{array}{cccc}
i_{0,0} & i_{0,1} & \cdots & i_{n,|I^n|}
\end{array} \\
\begin{pmatrix}
c_{i_{0,0},i_{0,0}} & c_{i_{0,0},i_{0,1}} & \cdots & c_{i_{0,0},i_{n,|I^n|}} \\
c_{i_{0,1},i_{0,0}} & c_{i_{0,1},i_{0,1}} & \cdots & c_{i_{0,1},i_{n,|I^n|}} \\
\vdots & \vdots & \ddots & \vdots \\
c_{i_{n,|I^n|},i_{0,0}} & c_{i_{n,|I^n|},i_{0,0}} & \cdots & c_{i_{n,|I^n|},i_{n,|I^n|}}
\end{pmatrix}
\end{array}
$$

Checking connectivity means search for a valid assignment for the adjacency matrix C, while the following constraints hold for this symmetric matrix, where $c_{p,q} = c_{q,p}$, $p, q \in I^A$:

$$\forall a_k \in A, p, q \in I^k : c_{p,q} = 0 \tag{1}$$

$$\forall a_k \in A, p \in I^k : \sum_{q \in I^A} c_{p,q} \leq 1 \tag{2}$$

$$\forall a_k, a_l \in A : \sum_{p \in I^k} \sum_{q \in I^l} c_{p,q} \leq 1 \tag{3}$$

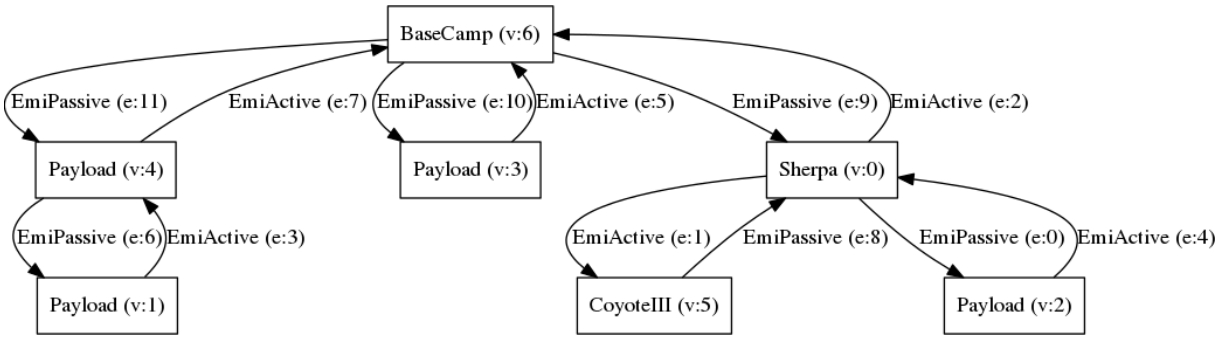


Figure 2: A feasible link structure for a composite agent after solving the assignment problem. Edges are annotated with the interface corresponding to the source vertex. Agent models and interfaces are related to the reference system described in [17].

Constraint 1 defines that no self links are allowed for an atomic agent, while Constraint 2 restricts each interface to be part of maximum one link only. Finally, Constraint 3 enforces Assumption 3.2, so that two atomic agents have to be connected by one link.

The assignment problem is solved using constraint-based programming and implemented using Generic constraint development environment (Gecode) [20], where the matrix entries represent the constraint-satisfaction problem (CSP) variables, each with the domain $D_c = \{0, 1\}$. Since a single agent might have multiple interfaces of the same type, the corresponding column assignments in the adjacency matrix are interchangeable, and create redundant solutions. We use symmetry breaking to reduce the number of redundant solutions, and to further speed the assignment process up, variable assignments are done in order of the least constrained agents, i.e.

$$a^* = \underset{a^k \in A}{\operatorname{argmin}} \frac{1}{|I^k|} \sum_{q \in I^A} \sum_{p \in I^k} c^*_{p,q} \tag{4}$$

, where

$$c^*_{p,q} = \begin{cases} 1 & \text{if } c_{p,q} \text{ is already assigned} \\ 0 & \text{otherwise} \end{cases}$$

In practice, we will also add a small fractional random bias which serves as tie breaker when between variables with equally constrained agents.

Figure 2 shows the result of a successful assignment procedure, for a set of seven agents, where the agent Sherpa comprises four male and two female interfaces, Payload one of each, CoyoteIII two male and BaseCamp five male.

4.4 Suitable agents

An atomic agent is associated with a set of resources, being either physical components or virtual ones such as capabilities and functionalities it can offer; the same holds for composite agents. Additionally, virtual resources can depend upon other resources, leading to a hierarchical dependency structure. In order to resolve the functional requirements of the mission specification to actual suitable agent type which support the requirements, the organization model provides a mapping function: $\mu : \mathcal{P}^{\mathcal{F}} \rightarrow \mathcal{P}^{\theta(\mathcal{A})}$, where $\mathcal{P}^{\mathcal{F}}$ represents the powerset of all functionalities, and $\mathcal{P}^{\theta(\widehat{A})}$ denotes the powerset of all general agent types. The function μ thus maps a set of functions to a set of general agent types which support this set of functions and forms feasible agents. The organization model encodes functionality based on resource availability, where resources can be physical devices and capabilities belonging to an agent. Thus, the organization model allows to infer functionality from a given agent type and its associated resource structure: $\mu^{-1} : \mathcal{P}^{\theta(\widehat{A})} \rightarrow \mathcal{P}^{\mathcal{F}}$. Thereby, the organization allows to map from agents to functionalities and back. An additional generalization can be achieved, when the mapping does not only account for a set of functionalities, but a set of arbitrary resource types which can be associated with a general agent. Currently, however, we have restricted the mapping to functionality.

Each agent type is associated with a maximum cardinality for a resource type, which reflects its initial and original state. Note, that setting the maximum cardinality still allows to lower the bound, in contrast to defining the exact cardinality. Therefore, the current modeling approach is prepared to consider resource failure or removal in future extensions.

Support is defined for an agent type and a single resource concept c as follows (cf. Roehr and Kirchner [19]):

$$support(\widehat{a}, c, f) \quad = \quad \frac{card_{max}(c, \widehat{a})}{card_{min}(c, f)} \tag{5}$$

, where $card_{min}$ and $card_{max}$ return the minimum and maximum required cardinality of resource instances. Accordingly, support of a function f with respect to a resource class c can be categorized as follows:

$$support(\widehat{a}, c, f) = \begin{cases} 0 & \text{no support} \\ \geq 1 & \text{full support} \\ > 0 \text{ and } < 1 & \text{partial support} \end{cases} \tag{6}$$

Since composite agents might comprise a high level of redundancy, the introduction of a saturation bound shall reduce the number of agents which have to be considered when a given set of functionalities is demanded. We define the *functional saturation bound* for an atomic agent type $\widehat{a}$ with respect to functionality f using the inverse of $support$:

$$FSB(\widehat{a}, f) = \max_{c \in \mathcal{C}} \frac{1}{support(\widehat{a}, c, f)}, \tag{7}$$

where $\mathcal{C}$ is a set of resource classes and $\forall c \in \mathcal{C} : card_{min}(c, f) \geq 1$ to account only for relevant resource classes. If there is no support for a $c \in \mathcal{C}$ such that $support(\widehat{a}, c, f)$ then $FSB(\widehat{a}, f) = \infty$. Similarly, the bound for a set of functions $\mathcal{F}$ is defined as:

$$FSB(\widehat{a}, \mathcal{F}) = \max_{f \in \mathcal{F}} FSB(\widehat{a}, f) \tag{8}$$

Identifying functionality support for a general agent type is equivalent to an atomic agent type, but to compute the maximum resource cardinalities the following holds:

$$card_{max}(c, \widehat{GA}) \quad = \quad \sum_{\widehat{a} \in \widehat{GA}} \gamma_{\widehat{GA}}(\widehat{a}) card_{max}(c, \widehat{a}) \tag{9}$$

, where $c \in \mathcal{C}$. Minimum resource cardinalities will be computed equivalently using $card_{min}(c, \widehat{a})$.

The number of general agent types that can support some functionality can be large, but it can be observed that for a supported set of functionalities a set of minimal general agent types $\mathcal{G}_{min}$ exists.

Definition 4.1. *A general agent type which supports a given set of functionalities and whose agent type cardinalities cannot be further reduced is denoted* ***minimal*** *with respect to the given set of functionalities.*

Hence, a minimal general agent type represents a lower bound to satisfy functionality requirements with a given combination of agent types.

5 Mission planning

The primary goal is to provide a valid assignment for the provided mission specification (cf. Section 3), while fleet size minimization and total cost minimization are secondary. The actual planning process is based on several stages in order to generate solutions:

(1) temporal ordering of all timepoints using a temporal constraint network
(2) upper and lower bounding of agent type cardinality for each spatio-temporal requirement
(3) generation of agent role timelines according to unification constraints and agent type cardinalities
(4) flow optimization to transfer immobile agents with mobile agents
(5) quantification of timepoints, based on transition times

Stage (1) creates a sequence of ordered timepoints, which is a necessary precondition for all next stages in order to identify concurrent resource usage and creating a commodity flow network. Stage (2) identifies the minimally required set of agent roles, and is for this reason a key element for minimization of the resources in use. Stage (3) takes all mission constraints into account in order to suggest a feasible agent role assignment. This assignment is the basis for local optimization in stage (4).

Constraint-based programming is involved in the reasoning of the organization model, and the planning stages (1),(2), and (3). Each of these stages involves the definition of appropriate branching strategies, and symmetry breaking conditions, and (3) uses of special implemented constraint propagator. If at any listed stage the search process fails, backtracking will be performed to the previous stage. The following sections will describe the details of the individual stages:

5.0.1 Temporal Ordering of Timepoints

To generate valid timelines and identify resource conflicts the approach requires a fully ordered set of timepoints. The generation of a fully constrained set of timepoints is based on qualitative temporal reasoning using point algebra with the set of relations $REL = \{>, <, =\}$ [16]. Consistency of the Temporal Constraint Network (TCN) is checked using a CSP which is defined by a set timepoint variables $T = \{t_1, t_2, \ldots, t_{|T|}\}$, a set $D = \{D_1, D_2, \ldots, D_{|T|}\}$ to represent the domain values for each timepoint $t \in T$, and a constraint set C with constraints of the form $C = \langle t_n, rel_i, t_m \rangle$, where $n, m = 1, \ldots, |T|$, and $rel_i \in REL$. A constraint is fulfilled if the relation described by $c \in C$ between two timepoint variables is fulfilled.

The final domain for each variable is restricted to a singleton: $|D_i| = 1$ and permitted values are $D_i \subseteq \{1, 2, \ldots, |T|\}$. If a full assignment of values can be found, the TCN is consistent, and the ordering of timepoints corresponds to the ordering of the assigned values. The qualitative temporal reasoning is sufficient to synchronize tasks, but only the quantification of time in the last stage of the planning approach will verify the temporal consistency of a solution.

5.0.2 Bounding agent type cardinality

To perform an upper and lower bounding of agent type cardinality a matrix based representation for spatio-temporal requirements and agent types is used, where $x_{i,j}$ represents the cardinality of agent type $\widehat{a}_j \in \widehat{A}$ and $s_i \in STR$. The following matrix representation with annotated rows and columns illustrates the meaning of each related CSP variable:

$$\begin{array}{c} \\ s_0 \\ s_1 \\ \vdots \\ s_m \end{array} \begin{array}{c} \begin{array}{cccc} \widehat{a}_0 & \widehat{a}_1 & \cdots & \widehat{a}_n \end{array} \\ \begin{pmatrix} x_{0,0} & x_{0,1} & \cdots & x_{0,n} \\ x_{1,0} & x_{1,1} & \cdots & x_{1,n} \\ \vdots & \vdots & \ddots & \vdots \\ x_{m,0} & x_{m,1} & \cdots & x_{m,n} \end{pmatrix} \end{array} \tag{10}$$

, where $n = |\widehat{A}| - 1$, and $m = |STR| - 1$. Each variable x has an initial domain of positive integers $D_x = \{0, 1, \ldots\}$.

Since resource availability is restricted, the general agent type $\widehat{GA}$ which is part of the mission specification defines an upper bound for all agent type cardinalities, which will be referred to as $\widehat{GA}_{UB}$ for readability.

Spatio-temporal requirements, however, can overlap, i.e. when they refer to the same location and their time intervals overlap. For a set of overlapping spatio-temporal requirements $\Omega = \{s_i, \ldots, s_j\}$, $s_i, s_j \in STR$ the upper bound is enforced as follows:

$$\forall \widehat{a}_j \in \widehat{A} : \qquad \sum_{s_i \in \Omega} x_{i,j} \leq \gamma_{\widehat{GA}_{UB}}(\widehat{a}_j) \tag{11}$$

The organization model is required to translate the requirements for functionalities into requirements for (suitable) general agent types, and apply the functional saturation bound. Lower bounds for each spatio-temporal requirement result from the combination of demanded functionalities and the given minimum agent type cardinalities. This lower bound represents a set of minimal general agents which is translated into the spatio-temporal

requirement's CSP variable domain. This domain is considered in the CSP by using extensional constraints for the assignment of model cardinalities, thus restricting model combination to minimal general agents. The extensional constraints enforce an exact assignment, but any full assignment of model cardinalities is only a lower bound for the subsequent agent role assignment stage. If no assignment can be found, too few resources are available to fulfil the mission requirements; the planning continues with another assignment of the temporal constraint network if possible or fails otherwise.

5.0.3 Agent roles

Subsequent to the CSP branching on bounded agent type cardinalities, a candidate assignment of agent roles to spatio-temporal constraints can be computed using a set of integer variables $y_{i,k,j}$, for $s_i \in STR$, $\widehat{a}_k \in \widehat{A}$, and $0 \leq j \leq \gamma_{\widehat{GA}_{UB}}(\widehat{a}_k)$, which have the domain $D = \{0, 1\}$:

$$\begin{array}{c} \\ s_0 \\ s_1 \\ \vdots \\ s_m \end{array} \begin{array}{c} \begin{array}{cccccc} & r_0^{\widehat{a}_0} & \cdots & r_0^{\widehat{a}_k} & \cdots & r_l^{\widehat{a}_n} \end{array} \\ \begin{pmatrix} y_{0,0,0} & \cdots & y_{0,k,0} & \cdots & y_{0,n,l} \\ y_{1,0,0} & \cdots & y_{1,k,0} & \cdots & y_{1,n,l} \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ y_{m,0,0} & \cdots & y_{m,k,0} & \cdots & y_{m,n,l} \end{pmatrix} \end{array} \tag{12}$$

, where $l = \gamma_{\widehat{GA}_{UB}}(\widehat{a_n}) - 1$, $m = |STR| - 1$, and $n = |\widehat{A}| - 1$.

Additional constraints are applied to guarantee unary agent role usage for time-overlapping constraints, and the general mission constraints described in Section 3 can directly be translated into low level CSP constraints, e.g., such as equality constraints minEqual, maxEqual as well as distinction constraints. Since agent roles of the same agent type are interchangeable symmetry breaking is applied to reduce the number of redundant solutions. While constraint propagation will reduce the corresponding domain and will lead to value assignment, full assignment of variables will only be performed for agent roles that (a) have an assignment apart from the single starting location, and (b) are mobile. To the first kind of agent roles we will also refer to as *active* agent roles. This partial assignment allows to extract full timelines for active mobile agents and partial timelines for active immobile agents. Both form the basis for a multi-commodity flow problem which is solved using integer linear programming.

5.0.4 Timeline Generation

Variable assignment for a single agent role variable assignment have to fulfill another important property: they have form a path in a temporal-expanded network. Ford and Fulkerson [14] have shown that networks can represent flow over time, and we similarly rely on what we call a temporal-expanded network to compute a flow-based representation for the mission planning problem. The temporal-expanded network has a bound on the number of edges by allowing only edges between vertices which are related to neighbouring timepoints and point forward in time:

Definition 5.1. *A time-expanded network for a set of timepoints T and a set of locations L is a graph $G = (V, E)$ with the following properties: Each vertex in V corresponds to a unique location timepoint tuple $v_{l,t} = (l, t)$, where $l \in L$, and $t \in T$. The set of edges is restricted: $e \in E \implies e = (v_{t_n,l_i}, v_{t_{n+1},l_j})$, where $n = 0, \ldots, |T| - 1$ and $i, j = 1, \ldots, |L|$. Without loss of generality $t_0 \leq t_1 \leq \cdots \leq t_{|T|-1}$.*

A custom (path) propagator has been implemented to exploit the structure of the network and enforce a constrained path in the network. This leads to a faster assignment process of agent role variables.

5.0.5 Multi-commodity flow

When the role assignment process is completed (fully for the mobile agents, and partially for the immobile ones), it is straightforward to translate the agent role timelines into a multi-commodity min-cost flow problem [2]: mobile agents represent transport providers, while immobile agents will be treated as individual commodities. Thus, edges in the network are either 'local' connections since they refer to the same location, or they are part of mobile agent routes. While we assume that local connections have infinite capacity, edges created as result of a mobile agent

transition have an upper capacity bound defined by the transport capacity of the corresponding mobile agent. All available mobile agents span a flow network over which commodities, or here immobile agents, can be routed to their target destinations. But agents are not restricted to a single target destination, so that requirements partially define a route for each agent. Therefore, the flow network represents all immobile agent requirements by minimum trans-flow requirements. Although bundling all agent types into one commodity would lead to a compact representation, route requirements for individual agents could not be set properly. Hence, each immobile agent role corresponds to a commodity, and role usage requirement are translated in to minimum transition requirements as already mentioned.

Mobile and immobile agent routes are transformed into a multi-commodity min-cost flow problem with unit commodity cost [2]:

$$\min \sum_{k,m} x_m^k$$

$$s.t. \sum_{e_m \in A_n} x_m^k - \sum_{e_m \in B_n} x_m^k = \begin{cases} S_k^+ & \text{if } n = s_k \\ -S_k^- & \text{if } n = t_k \\ 0 & \text{otherwise} \end{cases}, \forall\, n, k$$

$$x_m^k \geq l_m^k \wedge x_m^k \leq u_m^k$$

, where

$$\begin{aligned} G &= (V, E) \\ K &= \text{number of commodities}, k = \{1, \ldots, K\} \\ m &= \{1, \ldots, M\}, M = |E| \\ e_m &= \text{edge between node i and node j, i.e. (i,j)} \\ x_m^k &= \text{flow for commodity k in arc } e_m \\ c_m^k &= \text{unit cost for commodity k in arc } e_m \\ u_m^k, l_m^k &= \text{upper/lower bound for commodity k flow through edge m} \\ s_k, t_k &= \text{source/target of commodity k, } s_k \in V \\ S_k^+, S_k^- &= \text{supply/demand of } s_k \in V \\ B_n &= \text{set of incoming edges of node n} \\ A_n &= \text{set of outgoing edges of node n} \end{aligned}$$

To solve the network flow problem, the problem is first translated into a standard representation (CPLEX LP) so that different LP solvers can be used to solve the optimization problem (here: SCIP [1] and GLPK [15]). Any feasible and optimal solution of the network flow problem is also a feasible, but not necessarily an optimal solution for the mission assignment problem.

5.0.6 Quantification of time

A full solution still requires the quantification of temporal intervals: the qualified temporal network is therefore converted into a quantitative simple temporal network where the transitions between locations (and stqes) are based on the time required for the mobile systems to perform the location transitions and to form composite agents. Any min and max duration constraints will also apply at this planning stage.

5.1 Search & Solution repair

The previously described constraints lead to the generation of role timelines, and the CSP framework Gecode [20] has been used for the implementation. All role timelines are not only checked for feasibility via the multi-commodity min-cost flow optimization, but at the same time locally optimized. Still, finding a feasible solution for a highly restricted set of resources can be a significant challenge. Several strategies can be considering for search,

and our initial approach interprets lower agent type cardinality bounds as exact bounds - with the intention to keep the fleet size minimal and enlarge only when necessary. Hence, in the case when no optimal solution can be found, the infeasible (LP) solution is analysed to identify open flaws, i.e. unfulfilled minimum commodity trans-flow requirements. Upon identification of all flaws, a repair heuristic can be applied which injects additional transport provider requirements, thereby triggering either the change of existing mobile agent routes, or an increase of the lower agent type cardinality. The min-property constraint is used to augment the mission and restart the search after the local repair. For highly constrained missions, the repair process can reduce the number of flaws, but is slow at finding feasible solutions, hence showing that the heuristic is currently insufficient for complex setups.

An alternative is offered by the relaxation of cardinality bounds. In order to speed up finding an initial feasible solution, it is beneficial not to interpret the lower agent type cardinalities as exact bounds. Allowing an additional set of mobile systems (still within the number of the available ones) can reduce the time to find a feasible solution, but leads to higher redundancies and therefore less efficient solutions, since more agents will be required.

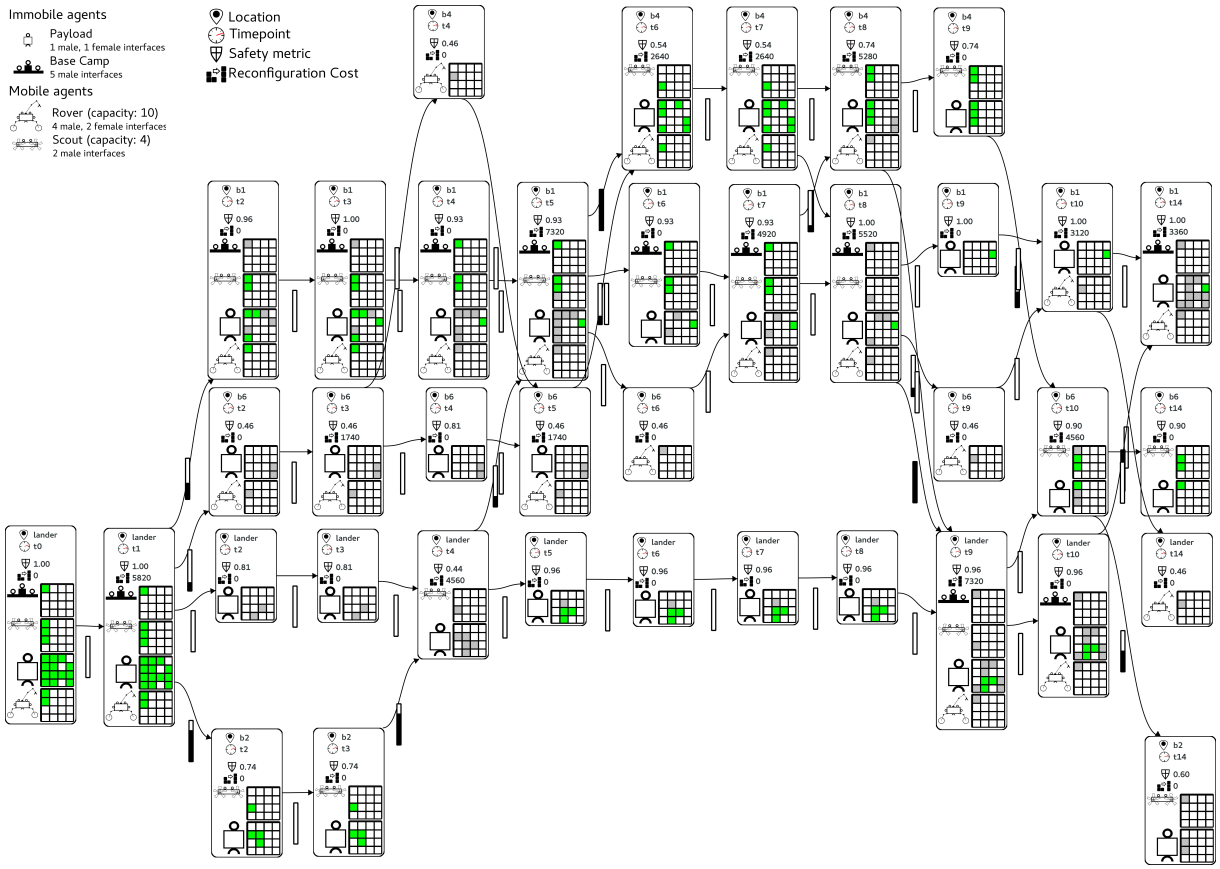


Figure 3: Example of a feasible solution for a full mission with locations = {lander,b1,b2,b4,b6}, timepoints = {t0,t1,...,t10,t14}. Fillbars indicate the consumed capacity of mobile agents. Color-coded boxes represent unique agent roles (only for better visualization limited to 16 per agent type): green = fulfilled requirement, gray = presence without a requirement.

Figure 3 shows a computed feasible solution. The general agents available for the mission are 3 Sherpa, 2 CREX, 3 Coyote II, 16 Payload, and 5 BaseCamp, where some agent interfaces are listed in the upper left corner of the figure. The assignment at the location lander shows, that only a subset of atomic agent is required for the solution. Fulfilled atomic agent requirements are highlighted as green squares, while the presence of systems without requirements is shown in green. These requirements, however, only represent one feasible set of atomic agent requirements which has been inferred from required functionalities. This solution has been computed within few seconds but only for a relaxed cardinality bound with two additional mobile agent roles (per mobile agent type).

5.2 Mission solution & cost function

The overall state of the agent organization, i.e. current connection state of atomic and composite agents is reflected by the coalition structure. In order to cost factor the dynamics in an agent organization two related concepts have to be used: policies and heuristics. Policies are required to define rules for selection and attribution. For example in the case of a transport multiple mobile robot may be available to perform this transport. To decide which one to take, a transport policy has been introduced, which chooses the agent with the largest transport capacity. For attribution energy consumption in a composite agent serves as main example. Since multiple power sources might exist is such system, a consumption policy has to distribute the consumption to all energy providers. Here, for the default policy each provider takes a share relative to its contribution to the overall energy capacity of the composite agent. Heuristics serve to interpolate a organizational state and estimate final mission costs: a duration heuristic for moving between locations relies on the information about the distance and the nominal speed of the transporting agent. Energy cost are depending upon the duration heuristic by relating duration to the power consumption of a composite system. Any reconfiguration changes this coalition structure, but requires a transition time, so that $\rho(CS_i^A, CS_j^A)$ defines the time to transition from one coalition structure CS_i^A to another CS_j^A. This cost heuristic assumes the same location of all agents in A.

The objectives of the planner is to find a solution that balances the overall energy consumed with the level of safety:

distance $d(a, \mathcal{M}_s)$ travelled distance of an agent a in mission $\mathcal{M}_s$

operation time $op(a, \mathcal{M}_s) = d(a, \mathcal{M}_s)/v_{nom}(a)$ duration of operation of an agent a

energy $E(a, \mathcal{M})$, where $E(a, \mathcal{M}_s) = op(a, \mathcal{M}_s) \cdot pw(\widehat{a})$ overall consumed energy by agent a to perform $\mathcal{M}_s$; $E(\mathcal{M}) = \sum_{a \in A}$ overall consumed energy per mission

safety $SAF(\mathcal{M}_s) = \min_{s \in STR} saf(s)$ represents the minimal safety level (here: redundancy) of the mission, where $saf(s)$ defines the safety metric associated with an stqe s based on the available (general) agent and with respect to the required set of resources; currently a redundancy based model is used to estimate the probability of survival based on an agent's set of component required to provide the functionalities in $\mathcal{F}$ (cf. [19] for details), such that $0 \leq saf(s) \leq 1$.

fulfillment $SAT(\mathcal{M}) = \frac{1}{|STR|} \sum_{s \in STR} sat(s)$ represents the ratio of fulfilled requirements, where

$$sat(s) = \begin{cases} 0 & \text{, } \textit{unfulfilled} \\ 1 & \text{, } \textit{fulfilled} \end{cases}$$

This following cost function reflects a balancing of three general mission aspects: efficiency through the energy cost function, efficacy through checking the level of fulfillment, and safety as redundancy dependant survival metric; for balancing the parameters α, β and $gamma$ can be used:

$$cost(\mathcal{M}_s) = \alpha E(\mathcal{M}_s) + \beta SAT(\mathcal{M}_s) + \gamma SAF(\mathcal{M}_s)$$

Figure 3 shows an example of a feasible solution. Each such solution can be translated into action plans for individual agent roles. Each vertex of the solution network serves as synchronization point and assumes reconfiguration operation to account for necessary coalition structure changes; the reconfiguration cost are annotated accordingly, along with the safety metric. Overall cost for the provided solution network are computed by constructing a simple temporal constraint network [9] where the bounds are defined by the transition times of the mobile agents.

6 Conclusion & Future Work

This paper presents the continued work for developing a planning system for a reconfigurable multi-robot system. The planner relies on constraint-based programming to specify and solve missions involving reconfigurable multi-robot systems, which is combined with multi-commodity flow optimization as local search. Furthermore, it suggests a multi-objective optimization target involving efficacy, efficiency and safety. The approach presented in this paper does not only result in a planning system for reconfigurable multi-robot system, but also in a tool which allows to analyse the effects of using reconfigurable multi-robot systems in robotics missions. Future work

will firstly focus on introducing better plan repair heuristics, and the extended use of meta-heuristic search strategies to improve the performance and scalability of the embedded local search approach. Secondly, a resource augmentation stage will be added in order to use previously unused resources to raise the level of safety.

Acknowledgments

This work was supported by the German Space Agency (DLR) under grant agreement 50RA1301 and 50RA1701, and the project Hi-Digit Pro 4.0.

References

[1] Tobias Achterberg et al. "Constraint Integer Programming: A New Approach to Integrate CP and MIP". In: *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems*. Vol. 5015 LNCS. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 6–20. ISBN: 354068154X. DOI: 10.1007/978-3-540-68155-7_4.

[2] Ravindra K. Ahuja, Thomas L. Magnanti, and James B. Orlin. *Network Flows: Theory, Algorithms, and Applications*. Michigan, US: Prentice Hall, 1993, p. 846.

[3] Ravindra K. Ahuja, James B. Orlin, and Dushyant Sharma. "Very large-scale neighborhood search". In: *International Transactions in Operational Research* 7.4-5 (Sept. 2000), pp. 301–317. ISSN: 0969-6016. DOI: 10.1111/j.1475-3995.2000.tb00201.x.

[4] Pasquale Avella, Maurizio Boccia, and Antonio Sforza. "Resource constrained shortest path problems in path planning for fleet management". In: *Journal of Mathematical Modelling and Algorithms* 3.1 (2004), pp. 1–17. ISSN: 1570-1166. DOI: 10.1023/B:JMMA.0000026675.50719.ce.

[5] Roberto Baldacci, Maria Battarra, and Daniele Vigo. "Routing a heterogeneous fleet of vehicles". In: *Operations Research/ Computer Science Interfaces Series*. Vol. 43. Springer, 2008, pp. 3–27. ISBN: 9780874216561. DOI: 10.1007/978-0-387-77778-8_1. arXiv: arXiv:1011.1669v3.

[6] Brian Coltin and Manuela Veloso. "Online pickup and delivery planning with transfers for mobile robots". In: *2014 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, May 2014, pp. 5786–5791. ISBN: 978-1-4799-3685-4. DOI: 10.1109/ICRA.2014.6907709.

[7] Brian Coltin and Manuela Veloso. "Ridesharing with passenger transfers". In: *Intelligent Robots and Systems (IROS 2014)*. 2014.

[8] Brian Coltin and Manuela Veloso. "Scheduling for Transfers in Pickup and Delivery Problems with Very Large Neighborhood Search". In: *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*. Québec City, Québec, Canada, 2014, pp. 2250–2256. ISBN: 9781577356790.

[9] Rina Dechter. "Temporal Constraint Networks". In: *Constraint Processing*. Ed. by Rina Dechter. The Morgan Kaufmann Series in Artificial Intelligence. San Francisco: Morgan Kaufmann, 2003. Chap. 12, pp. 333–362. ISBN: 978-1-55860-890-0. DOI: 10.1016/B978-155860890-0/50013-X.

[10] Alexander Dettmann et al. "Heterogeneous Modules with a Homogeneous Electromechanical Interface in Multi-Module Systems for Space Exploration". In: *Proceedings of the 2011 IEEE International Conference on Robotics and Automation (ICRA '11)*. Shanghai, China: ESA, May 2011, pp. 1964–1969.

[11] Virginia Dignum, ed. *Handbook of Research on Multi-Agent Systems: Semantics and Dynamics of Organizational Models*. IGI Global, 2009. ISBN: 9781605662572.

[12] Rodolfo Dondo and Jaime Cerdá. "A cluster-based optimization approach for the multi-depot heterogeneous fleet vehicle routing problem with time windows". In: *European Journal of Operational Research* 176.3 (2007), pp. 1478–1507. DOI: 10.1016/j.ejor.2004.07.077.

[13] Michael Drexl. "Applications of the vehicle routing problem with trailers and transshipments". In: *European Journal of Operational Research* 227.2 (2013), pp. 275–283. DOI: 10.1016/j.ejor.2012.12.015.

[14] Lester Randolph Ford and Delbert Ray Fulkerson. *Flows in networks*. Tech. rep. Santa Monica, California: The RAND Corporation, 1963, p. 152.

[15] Free Software Foundation. *GLPK (GNU Linear Programming Kit)*. Available at: https://www.gnu.org/software/glpk, (Accessed: 1 October 2015). 2015.

[16] Rina Detcher. *Constraint Processing*. Vol. 33. Morgan Kaufmann Publishers Inc., 2003, p. 480. ISBN: 978-1-55860-890-0.

[17] Thomas M. Roehr, Florian Cordes, and Frank Kirchner. "Reconfigurable Integrated Multirobot Exploration System (RIMRES): Heterogeneous Modular Reconfigurable Robots for Space Exploration". In: *Journal of Field Robotics* 31.1 (Jan. 2014), pp. 3–34. ISSN: 15564959. DOI: 10.1002/rob.21477.

[18] Thomas M. Roehr and Ronny Hartanto. "Towards safe autonomy in space exploration using reconfigurable multi-robot systems". In: *Proceedings of the International Symposium on Artificial Intelligence, Robotics and Automation in Space (i-SAIRAS 2014)*. ESA, 2014.

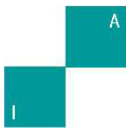
[19] Thomas M. Roehr and Frank Kirchner. "Spatio-Temporal Planning for a Reconfigurable Multi-Robot System". In: *Proceedings of the 4th Workshop on Planning and Robotics (PlanRob)*. Ed. by Alberto Finzi and Erez Karpas. London, 2016, pp. 135–146.

[20] Christian Schulte and Guido Tack. "View-based propagator derivation". In: *Constraints* 18.1 (2012), pp. 75–107. ISSN: 1383-7133. DOI: 10.1007/s10601-012-9133-z. arXiv: arXiv:0908.2050v1.

[21] Roland Sonsalla et al. "Towards a Heterogeneous Modular Robotic Team in a Logistic Chain for Extraterrestrial Exploration". In: *Proceedings of the International Symposium on Artificial Intelligence, Robotics and Automation in Space*. Montreal, Canada: ESA, 2014.

[22] Paolo Toth and Daniele Vigo, eds. *Vehicle Routing: Problems, Methods, and Applications, Second Edition*. 2nd ed. MOS-SIAM, 2014. ISBN: 1611973597. DOI: 10.1137/1.9781611973594.

Inteligencia Artificial, 21(62) (2018), 40-52
doi: 10.4114/intartif.vol21iss62pp40-52



INTELIGENCIA ARTIFICIAL

http://journal.iberamia.org/

Planning and Scheduling in Additive Manufacturing

Filip Dvorak, Maxwell Micali and Mathias Mathieu
Oqton, 832 Sansome Street
San Francisco, California 94111
{filip.dvorak,maxwell.micali,mathias.mathieu}@oqton.com

Abstract Recent advances in additive manufacturing (AM) and 3D printing technologies have led to significant growth in the use of additive manufacturing in industry, which allows for the physical realization of previously difficult to manufacture designs. However, in certain cases AM can also involve higher production costs and unique in-process physical complications, motivating the need to solve new optimization challenges. Optimization for additive manufacturing is relevant for and involves multiple fields including mechanical engineering, materials science, operations research, and production engineering, and interdisciplinary interactions must be accounted for in the optimization framework.

In this paper we investigate a problem in which a set of parts with unique configurations and deadlines must be printed by a set of machines while minimizing time and satisfying deadlines, bringing together bin packing, nesting (two-dimensional bin packing), job shop scheduling, and constraints satisfaction. We first describe the real-world industrial motivation for solving the problem. Subsequently, we encapsulate the problem within constraints and graph theory, create a formal model of the problem, discuss nesting as a subproblem, and describe the search algorithm. Finally, we present the datasets, the experimental approach, and the preliminary results.

Keywords: Planning, scheduling, csp, additive manufacturing, nesting, bin-packing, jobshop.

1 Introduction

As engineered components and aesthetic creations become more advanced and intricate, they push the boundaries of what is manufacturable via traditional manufacturing processes. These boundaries may be of a technical nature, in which certain geometries or certain materials simply cannot be produced with traditional equipment; or they may be economic in nature, such that the high mix and low volume of production orders cannot be manufactured in a cost-efficient manner due to high tooling and labor costs incurred with each associated setup. The onset of additive manufacturing (AM) technologies has presented new capabilities to engineers and designers, greatly expanding the domain of the manufacturable design space along several dimensions, allowing for design behaviors and concepts which previously may have only been conceivable and not producible: mass customization of components; intricate geometries with no restrictions on visibility, draft angles, or other design requirements; and exotic material behaviors such as negative stiffness and negative thermal expansion [9].

For the manufacturing and production engineers, AM technologies allow components to be printed "on demand", and for the supply chain to consist of digital files rather than physical components. Figure 1 shows an example of a complex geometry which can only be produced by additive manufacturing. While the unit production cost for a single additive component may be higher when compared to full-scale traditional production, additive manufacturing is drastically less expensive at low and medium volume production. The industry is beginning to embrace these new manufacturing methods in order to streamline aspects of production, such as the well-known example of GE printing fuel injector nozzles for jet engines in a single print [6]. The nozzle previously required producing of 20 complicated components, as well as the additional labor to weld and braze them into an assembly afterward. In addition to being more streamlined to produce, the additive version of the nozzle was also 25% lighter, which directly translates to additional fuel savings for the aircraft [3].

ISSN: 1137-3601 (print), 1988-3064 (on-line)

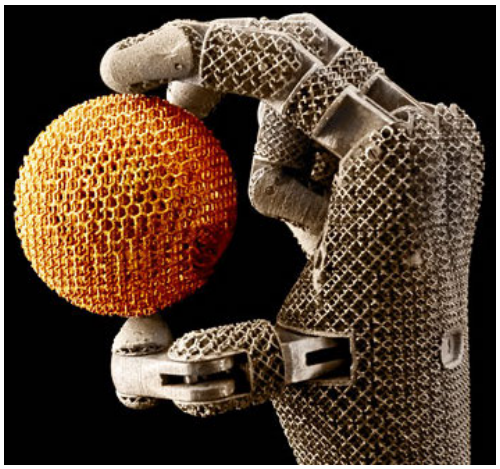


Figure 1: Example of a component which is impossible to manufacture by traditional manufacturing techniques, yet is able to be printed despite the intricate lattice structures. [13].

Additive manufacturing can result in drastic reductions in the overall production time of a component. This efficiency stems from reducing the number of procedures involved in the production process, even if a typical print takes longer than a typical process step in a traditional manufacturing flow. Whereas a single machining operation may take on the order of minutes or hours to execute, a single print is on the order of hours or days. However, traditional production of a part may require dozens of piecewise machining operations for completion, while a print is a single-operation process. Both additive and traditional manufacturing may require some standard finishing operations to accomplish things such as achieving the desired surface finish or removing structures used to fix the component to the machine during production, and these times are process-independent.

Given that additive manufacturing consists of a single, time-consuming, step and that there is an opportunity to fit multiple components onto the same build plate so they can be produced in parallel while incurring only marginal additional time costs, planning and scheduling for additive manufacturing at a factory scale brings a unique set of emerging opportunities and challenges while attempting to optimize the process.

In the rest of this section we describe some existing challenges in additive manufacturing and establish the problem of optimizing scheduling and planning for AM.

1.1 Additive Manufacturing

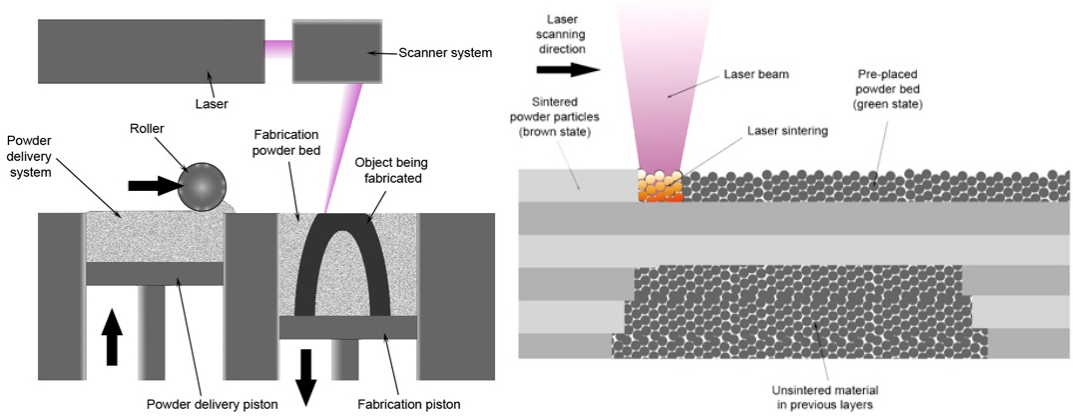


Figure 2: Schematic of selective laser melting (SLM), which is classified as a powder bed fusion process [19].

Additive manufacturing, also known as 3D printing and other names, has existed as a commercial technology since 1984 with the invention of stereolithography, although the overall vision of precisely replicating physical

objects preceded the invention of stereolithography by a century [18]. The roots of AM technology date back to the 1800s with developments in the fields of photo sculpture and topography, but the overall vision of printing three-dimensional objects was not possible until advancements in materials science, computer-aided design (CAD), computer numerical control (CNC), and laser technology were made, triggering the rapid development of a variety of AM techniques which can be grouped into seven major technology categories [1]:

- *Binder jetting*—an additive manufacturing process in which a liquid bonding agent is selectively deposited to join powder materials.
- *Directed energy deposition*—an additive manufacturing process in which focused thermal energy is used to fuse materials by melting as they are being deposited.
- *Material extrusion*—an additive manufacturing process in which material is selectively dispensed through a nozzle or orifice.
- *Material jetting*—an additive manufacturing process in which droplets of build material are selectively deposited.
- *Powder bed fusion*—an additive manufacturing process in which thermal energy selectively fuses regions of a powder bed.
- *Sheet lamination*—an additive manufacturing process in which sheets of material are bonded to form an object.
- *Vat photopolymerization*—an additive manufacturing process in which liquid photopolymer in a vat is selectively cured by light-activated polymerization.

Figure 2 shows a schematic of the selective laser melting (SLM) process, which is a type of powder bed fusion process. SLM is typically used for producing metallic parts, and due to its industrial value in many manufacturing verticals, SLM is the AM process focused on in this paper.

Many technological advancements have enabled the existence of AM processes, yet many challenges still remain as the field matures. One such challenge is how to incorporate additive manufacturing into a traditional manufacturing process flow, since AM has different planning and scheduling requirements and considerations. Other challenges involve understanding, modeling, and controlling complex physical behaviors and phenomena involved in the process, which are ultimately influenced by the planning and scheduling decisions made upstream of the process. Some examples of these decisions are how to optimally orient the geometries in space for printing, and how to optimally nest multiple parts within a single print without inducing any build failures. “Optimal” may be defined in terms of overall print cost, overall print quality, or some other metric of optimality. It is important to note that some process decisions may induce physical conditions within the print which yield total process failures, with the onset of cracking, layer delamination, or unacceptable degrees of component warping. For this paper, we will assume that planning and scheduling decisions are made in fully observable and deterministic world where all possibilities succeed.

1.2 Related Work

Additive manufacturing has been an emerging technology for several decades, and each advancement has provided new alternatives to traditional manufacturing methods for candidate geometries and applications. Mass production via additive manufacturing is now viable, yet the operations research techniques designed for traditional manufacturing are not directly transferable to AM. A number of cost models for AM have been recently developed in [10] and [15], a survey of cost models was performed in [4], and various AM products have been mapped in [3]. However, only a few models for production planning of AM have been proposed. Notably, [14] has proposed a constraint optimization model on top of CPLEX, which does not take into account deadlines and approximates nesting by the total surface area. In [2], the authors implemented a model in MATLAB which takes into account nesting and scheduling metrics of earliness and tardiness, however they only consider a single machine.

1.3 Challenge Statement

Production planning in AM starts when a customer sends design specifications (CAD file) of a part that needs to be manufactured, along with a production deadline and delivery location. The design specifications of the part include geometric dimensioning and tolerancing (GD&T) information, which restricts the decision space in the production planning process. For example, the GD&T information may specify tolerances in regions of the design which are not achievable by some types of printing processes. Since machines are capable of printing multiple parts simultaneously in a single print operation, it is important to consider optimizing the constitutive members of such sets and how they fit together. Each part has its printing orientation, and they must collectively fit into

the build volume of the machine – typically a regular hexahedron defined by height, width, and length. A single print operation is called a build, and we assume that the downward projections of different parts within a build cannot overlap. The duration of each build is functionally dependent on the speed that the laser beam in the machine is programmed to travel, the maximum height of the set of parts in the build, and the total path traveled by the laser beam during the build. Each machine behaves as a unary resource, and thus can execute only a single build at any moment in time.

The problem consists of a set of parts, N, including their possible orientations, configurations, and deadlines, as well as a set of machines, M. The solution to the problem is the set of builds, B, scheduled to machines, M, which maximizes the number of parts printed prior to their deadlines, while also minimizing the overall printing cost.

In this paper we are relaxing several requirements which are considered during production planning. In particular, the shipping deadlines to given delivery locations are relaxed by projecting the upper bound of shipping time into the scheduling deadline. In reality, the location of manufacturing assets around the globe would influence scheduling deadlines, but upper bounding of resource requirements for logistics allows for treating all machines as though they are in a single location. We consider 2D bin-packing (nesting) when combining different parts into a single build. While 3D bin-packing is conceptually possible, i.e. by using horizontal bridges, we do not consider it in this paper. Finally, builds may require additional preparation and post-processing steps when the machine is cleaned, the material is recycled, the machine configuration is adjusted, and small finishing operations are required on the parts before shipment. Those steps are performed by human operators that have their own shifts and schedules that vary across different time-zones (i.e. a high priority order may start to be printed in Hungary, because the operators are already up and can configure the machines, while a factory in Nevada is printing their night jobs). We are relaxing the human operators and upper bounding their work into the duration of printing a build, which also allows for relaxing the temporal reasoning to the sequence of builds on each machine.

The problem we are solving consists of multiple nested NP-hard problems (bin-packing, set cover, job-shop), and to describe the structure of our model we first encapsulate the configuration of parts. In the next section we formulate the assignment of parts into builds as a graph decomposition. Next we describe the challenges of nesting parts together, and finally we describe the complete model with experimental evaluation.

2 Configuration

A manufacturing configuration for a part P is a collection of rules that need to be followed to maximize the expectation of achieving the desired quality of part P. We model a manufacturing configuration O of part P as a constraint satisfaction problem $O = (V^p \cup V^m, C)$, where V^p are *persistent* variables that relate only to the part P, V^m are variables related to the manufacturing process of P that merge through composition, and constraints C then propagate across V^p variables through V^m variables. We say that the configuration O is *valid* iff there exists an assignment α of values to the variables $V^p \cup V^m$ such that all constraints in C are satisfied.

Having a set of all possible configurations Ω we define composition $\bigodot : \Omega \times \Omega \rightarrow \Omega$ as a function that composes two configurations into another configuration. Having two configurations $O_1 = (V_1^p \cup V^m, C_1)$ and $O_2 = (V_2^p \cup V^m, C_2)$ we construct a composition $O_1 \bigodot O_2 = (V_1^p \cup V_2^p \cup V^m, C_1 \cup C_2)$. As a practical example we can imagine having a part P, its configuration $O = (V^p \cup V^m, C)$, persistent variable *orientation* $\in \Re \times \Re \times \Re$, variable *height* $\in \Re$, and constraint *height* $>= h(O)$ (for simplicity we assume that function h computes the height of the part P given its configuration O that contains the *orientation* variable and volumetric data). We can note that for a composition of two parts, we will determine that the printing height has to be larger than the maximum height of either part. In similar fashion we can compose deadlines for different parts and all the configuration variables.

2.1 Compatibility Graph

Having a set of parts Π and their configurations Ω, we assume that all configurations are valid (invalid configurations trivially reflect a failure in the part preparation process), then we say that two configurations O_1 and O_2 are *compatible* iff $O_1 \bigodot O_2$ is valid. We can see that compatibility is a symmetrical, reflexive and non-transitive relation. Having a set of nodes $N = \Omega$ and a set of edges $E = \{(x, y) | x \text{ is compatible with } y\}$ we define the *compatibility graph* as $G = (V, E)$. We can observe that a build which combines together multiple parts has to be a complete subgraph (a clique) within the compatibility graph.

A compatibility graph allows us to represent relationships between individual parts, to provide a structure that can be quickly searched for build candidates, and to record structural information from previous searches, i.e. no-goods such as two parts that should not be in the same build because one of their deadlines will be violated.

Figure 3: Example of a compatibility graph for a problem with 20 machines and 50 parts. Nodes of the same color belong to the same clique. The clusters correspond to different materials.

2.2 Graph Decomposition

Choosing a single clique in the compatibility graph represents creating a job and scheduling it on a printer. Consequently, choosing a set of cliques that cover the whole graph without intersections on nodes represents a decomposition of the graph into complete subgraphs, and it is also equivalent to the exact set cover problem [5] with an additional restriction that each subset of the covering is a clique in the compatibility graph. Figure 3 illustrates a graph decomposition into cliques. Having an initial state I of the search space in which all cliques contain only a single node, we can look at the search space of the problem as a tree that merges two cliques at each branch if the parts within the two cliques can be nested together, until no two cliques can be merged. However, the size of the combinatorial space of choosing the cliques to merge one by one is $O(n^n)$, impractical for any exhaustive search algorithm.

While pairwise compatibility is a necessary condition for having a valid build, it is not a satisfactory condition, which is provided by running the nesting algorithm described in the next section.

3 Nesting

The nesting algorithm aims to efficiently arrange a given set of parts on a given build plate size by maximizing the total area covered with parts. Each part has an associated economic value and several possible orientations, which are considered under several (< 10) different rotations around the z-axis. The nesting algorithm maximizes the value of the build plate.

Nesting operates at three levels of fidelity with varying degrees of computational complexity - the highest fidelity nesting takes the longest to compute, while the lowest fidelity nesting is computed the fastest. Low-fidelity nesting uses 2D bounding boxes, medium-fidelity nesting uses silhouettes while packing the parts greedily, and high-fidelity nesting explores the search space using several search algorithms.

3.0.1 Low Fidelity

At the lowest fidelity of nesting we use the 2D bounding boxes of the given parts. First these rectangles are sorted in a descending sequence by the value of the part, then by their shortest side, and finally by their longest side. We place these rectangles using the Maximal Rectangles Best Short Side Fit algorithm [12]. This technique keeps track of the maximal free rectangles in the bin, these are the rectangles remaining after placing the bounding boxes. It then places each rectangle in one of these free rectangles, minimizing the shortest leftover side.

3.0.2 Medium Fidelity

This level of nesting uses a greedy algorithm to place the parts. First the silhouettes of the parts are constructed. The resulting polygons are sorted in descending sequence by the value of the object, and then by area. Then the polygons are placed one by one on the build plate. For each polygon we compare all possible rotations and

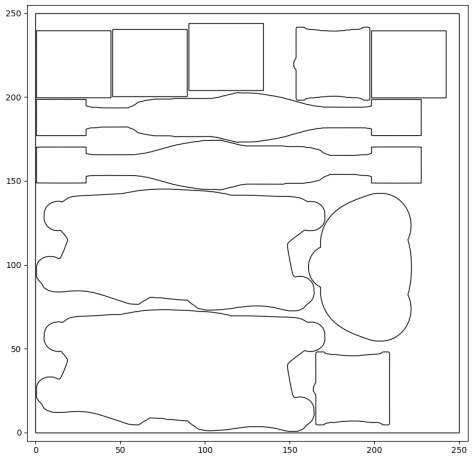


Figure 4: Example result from medium fidelity nesting.

positions, and choose the combination that results in the smallest bounding box. For this level of nesting only a single orientation per object is considered. Figure 4 shows an example of medium fidelity nesting.

3.0.3 High Fidelity

The highest fidelity nesting procedure explores the solution space using several search strategies - Genetic Algorithm, Simulated Annealing and Tabu Search. A solution to the problem consists of the order in which we place the parts, and the orientation and rotation of each part. The placement of parts is done with the Bottom-Left strategy [8], where we place each parts as close to the bottom-left of the build plate as possible. The nesting algorithm either maximizes the total area used area or the total value of the parts in the build.

4 Mathematical Model

We build up a representation of the problem as a meta constraint optimization problem [7] as a pair (V, C), where V is a set of variables and C is a set of constraints on top of V, where some constraints are hard and always need to be satisfied (a single machine cannot perform two activities at once), while others are soft (not every part can always be finished on time). Then the solution of the problem (V, C) is an assignment α that assigns exactly one value to each variable while all the hard constraints are satisfied. The quality of the solution is further evaluated based on the number of unsatisfied constraints, which is represented in the objective function.

We define the problem variables as follows:

- $P = \{p_1, .., p_n\}$ is a set of parts.
- $B = \{b_1, .., b_n\}$ is a set of builds to be printed.
- $M = \{m_1, .., m_m\}$ is a set of machines.
- d_{p_i} denotes the deadline for part p_i.
- d_{b_i} denotes the deadline for build b_i.
- $dur_{b_i}^{m_j}$ denotes the duration for the build b_i on machine m_j.
- c_{m_i} denotes the configuration of the machine m_i.

The decision variables of the problem are the following:

- c_{p_i} denotes the configuration assigned to the part p_i.
- $a_{p_i} = b_j$ denotes the build b_j is assigned to part p_i, we can also write $p_i \in b_j$.
- $a_{b_j} = m_k$ denotes the machine m_k assigned to build b_j, we can also write $b_j \in m_k$.

On top of the variables we are going to use multiple constraints and a strategy for ordering builds at the individual machines, described in the following sections.

4.1 Constraints

We use the following constraints to maintain the consistency of the solution and cut symmetrical parts from the search space. Note that some of the hard constraints are implicitly encoded in the representation, i.e. the constraint that a build can only be assigned to a single machine, and that a part can only be assigned to a single build. These constraints are written below for completeness:

$$\bigcup_{b \in B} b = P$$

$$\forall b_i, b_j \in B : b_i \cap b_j = \emptyset$$

$$\bigcup_{m \in M} m = B$$

$$\forall m_i, m_j \in M : m_i \cap m_j = \emptyset$$

4.1.1 Compatibility Constraints

enforce that parts within a single build are compatible with themselves, as well as the machine which is processing the build. We define the constraint as follows:

$$\forall b_i \in B, a_{b_i} = m_j : c_{m_j} \bigodot \prod_{x \in \{y | a_y = b_i\}} x \text{ is valid.}$$

The compatibility constraint is evaluated by solving (finding a witness solution) of the inner CSP problem, where the compatibility graph provides fast evaluation and filtering of impossible candidates. Preprocessing of the compatibility graph consists of computing the compatibility relation between every pair of possible configurations coming from two different parts, likewise every possible part configuration with every machine. The cost of preprocessing of the compatibility graph is negligible, $O(n^2)$, where n is the number of parts.

4.1.2 Nesting Constraints

are global constraints whose evaluation runs asynchronously in parallel with the main search algorithm. The main search algorithm uses the total surface area as an upper bound on how densely nested the parts can be in a build. Then those builds are sent to the nesting algorithm which provides a list of the parts that do fit into the build and the parts which are leftover from the build. We denote the set of all builds that were returned by the nester as Φ and also have $\emptyset \in \Phi$, representing that an empty build is trivially satisfied. We define the nesting constraint as follows:

$$\forall b_i \in B : \{p_j | a_{p_j} = b_j\} \in \Phi.$$

In other words, all builds need to be empty or confirmed by nester through the set Φ. We treat the nesting constraint as a soft constraint, where each build not contained in Φ represents a violated constraint. For the purpose of this paper, nesting within the nesting constraint is done at the medium fidelity level.

4.1.3 Deadline Constraints

guarantee that the parts are finished on time. We define the deadline for a build as $d_{b_i} = min_{p_j \in b_i} d_{p_j}$, in other words, a build's deadline is the earliest deadline among its parts. Then, assuming time at the beginning of the world is 0, we define the deadline constraints as follows:

$$\forall m_j \in M, \forall b_i \in m_j : \sum_{b \in m_j | d_b <= d_{b_i}} dur_b^{m_j} <= d_{b_i}$$

In other words, on a single machine the deadlines of all the builds must occur after the sum of durations of the builds whose deadlines are earlier. It may not be always possible to fulfill all the deadline constraints, and we may instead treat them as soft constraints which need to be minimized.

The computational problem of evaluating the deadline constraints is a standard scheduling problem that we focus on in the next section.

4.2 Tardy Builds

Once a build is assigned to a machine we can consider it to be independent of the builds assigned to the other machines. Such independence would disappear if we also considered that operators can service only a single machine at a single moment and whose availability is restricted. The advantage of the independence between machines is that instead of extending the state space with precedence constraints between builds on each machine we can choose a strategy that orders the builds. This approach follows the scheme of machine-based problem decomposition [16].

Having multiple builds assigned to a single machine, we are mostly interested in whether a build is either on time, or if it is late and by how much it has missed its deadline. There are several approaches to model tardiness. We are going to use the $\alpha|\beta|\gamma$ notation [11] for categorizing them as scheduling problems.

- Minimizing Total Tardiness of Builds. We minimize the sum of tardiness of all builds that missed the deadline. Categorized as $1||\sum T_j$, the problem is known to be NP-hard [16].
- Minimizing the number of Tardy Builds. The number of tardy builds represents the number of builds that have at least one late part. We can solve the problem $1||\sum U_j$ with an optimal algorithm in $O(nlog(n))$.
- Minimizing the number of weighted Tardy Builds. We can further extend the representation of tardy parts by adding a weight to each of them – $1||\sum w_j U_j$, which also makes the problem NP-hard – we can imagine to have all deadlines failing and we get the knapsack problem.

For the purpose of this paper we are going to minimize the number of tardy builds $1||\sum U_j$, which is efficiently computed using an adaptation of a standard scheduling algorithm [16]. The algorithm has two steps:

- Order the builds into an ascending sequence L based on the earliest deadline among the parts each build contains.
- Keep adding builds into a sequence S for as long as all builds in S are on time. If S has a build that misses the deadline, remove from S the build with the longest duration.

The algorithm is optimal in minimizing the number of builds that have at least one order miss its deadline [16], however it does not guarantee optimality in minimizing the total number of orders that are late. In the trivial case, when each build consists of a single order, the algorithm is optimal for minimizing the number of late parts as well. However, minimizing the number of late parts is NP-hard in general. We can show that when the deadlines for all parts are the same then the number of items in a build corresponds to a cost of an item in the knapsack problem. Consequently, we can translate a knapsack problem into minimizing the cost of parts that miss deadline.

4.3 Objective Function

The primary optimization criterion is to minimize the number of unsatisfied soft (deadline and nesting) constraints, and then minimize the makespan (maximum execution time) of the schedule. Given that the nester is run asynchronously, its output becomes integrated into the main search algorithm via the lazy evaluation of the Nesting Constraints.

5 Search Algorithm

The average problem size prevents the use of exhaustive search techniques such as branch and bound. Where we cannot expect solutions in reasonable time, we instead adopt local search methods aided by (meta)heuristics. We use a portfolio of local-search algorithms on top of the CSP representation of the problem combined with a range of steps that define the neighborhood for the local search. We use the following local search algorithms:

- Hill Climbing makes the move to the lowest cost state in the neighborhood.
- Tabu Search prevents Hill Climbing from being stuck in a cycle by remembering a list of recently visited states that should not be visited again.
- Simulated Annealing begins at one state, and at each step it can choose a state with the same minimal cost as HC, or it can choose a state that, with some probability, does not have the minimal cost. The probability of choosing suboptimal states reduces in time as the algorithm progresses, hence Simulate Annealing is more likely to escape from local optima early, while behaving like HC after certain amount of time.
- Step Counting uses the cost of the current state as a bound for several future states.
- Late Acceptance makes moves to states that are at least as good as the state several steps ago.

The neighborhood in the search space is defined through four atomic steps that generate successor states:

- *emptyMove* chooses builds b_i, b_j, part p and machine m_k, where $|b_i| > 1$, $p \in b_i$, and $b_j = \emptyset$. Then it removes p from b_i, adds it to b_j, and assigns b_j to m_k. This move represents a situation, where we remove a part from an existing build to start a new build, and then move that new build to a new machine. The main advantage of this step is symmetry breaking – it does not matter which empty build has been chosen, since we assign it to a new machine right away and we only assign it to a compatible machine.
- *moveBuild* chooses a build b_i, b_i and machine m_j and assigns b_i to m_j.
- *moveOrder* chooses a build b_i, $|b_i| > 0$ and part p_j and assigns p_j to b_i.
- *changeConfiguration* chooses a part p and a possible configuration $c \in \Omega$ and performs assignment $c_{p_i} = c$.

These four steps together give access to the whole search space of the problem, and the goal of the search algorithm is to efficiently explore it using heuristics.

5.1 Heuristics

We use several problem-specific heuristics that encapsulate some knowledge about the problem and a number of problem-independent heuristics. The first heuristic run is the *construction heuristic* that creates the initial state from which the search algorithm starts to explore the search space. The construction heuristic solely assigns each order into a build and then moves it to one of the compatible machines.

We use tie-breaking heuristics for variable and value selection. In particular, we choose configurations of parts whose height is the largest among the heights that still fit into the machine. When an order is moved between builds, the builds with close mean deadlines are tried first.

5.2 Solving Approach

Combining the previously defined structure and given a set of parts P and a set of machines M, we find the solution using following steps.

1. Pre-compute the compatibility graph for all configurations of all parts and all the machines. Then filter domains of the variables.
2. Create an initial state where each part is assigned to a different build and all the builds are assigned to some machines.
3. Perform a local search over the decision variables of the problem using the *moves* to change their values until the given time is spent.
4. Run nesting in parallel with the previous step, such that the builds confirmed by the nester are available in the nesting constraint and the nester is invoked whenever a new undiscovered and non-dominated build is considered by the nesting constraint.
5. Collect the assignments of parts to builds and schedule the builds to machines in time.

The simultaneous execution of steps 3 and 4 follows a *best effort* approach that allows two solvers for hard problems to exchange the information and reach higher quality solutions than if run in a sequence.

6 Experimental Evaluation

To gain insight into the difficulty of finding good solutions for the stated problem in additive manufacturing, we have created two problem generators, produced a dataset, and evaluated an implementation of our proposed model.

Note that at the time when we have run the experiments we have not yet integrated the nester implementing the nesting constraint, hence all the nesting constraints are considered violated across the search space and the nesting is only approximated as the total available surface.

6.1 Problem Generators

We use two types of problem generators. One generates random problems, and the other generates problem instances with known optimal values. The random problem generator *RG* for a given seed of randomness D produces a random problem using a uniform random generator integrated in Java with values in the following ranges:

- Machine is generated to support
  - Material $\in M, |M| = 5$.
  - Height $\in \{20, .., 70\}$ cm.
  - Width $\in \{50, .., 100\}$ cm.
  - Length $\in \{50, .., 100\}$ cm.
- Part is generated with the following features:
  - Material $\in Materials, |Materials| = 5$.
  - Configuration $\in C, |C| = 5$ and each configuration defines the following:
    - Volume $\in \{1, .., 100\}$ liters.
    - Height $\in \{10, .., 40\}$ cm.
    - Width $\in \{10, .., 50\}$ cm.
    - Length $\in \{10, .., 50\}$ cm, where $Volume = Length * Height * Width$.

In other words, we generate a selection of machines with various sizes and materials, then we generate parts which require certain material but can be oriented through the change of their configuration, which maintains the volume but modifies dimensions. Consequently, we create a new optimal problem generator *ROG* using the random generator *RG* as follows:

- Given seed D, use the Random Generator *RG* to produce a random problem.
- Run 20 seconds of hill-climbing to find a sub-optimal low-makespan schedule for the problem.
- Stretch all orders such that they perfectly occupy the space of the machines and that the builds perfectly occupy the time of the machines.
- Propagate the stretching of orders into its configurations such that the volumetric constraints are satisfied.
- Record the solution cost and randomize the perfect schedule.

Both *RG* and *ROG* generate a problem for each given seed and the numbers of machines and orders. *RG* generates problems that are normally distributed, and the performance of the planner upon those problems gives a realistic measure on how quickly the search algorithm can converge to some local optima, depending on the size of the problem, and where it is not helpful to invest more computational time. While the problems generated by *ROG* are not normally distributed, they give some estimation about the worst-case distance from the optimality when the planner converges to a local optima.

6.2 Implementation and Environment

We have implemented the model using the OptaPlanner [17] constraint solver on top of a Java representation. To run the experiments we have used a set of homogeneous AWS EC2 instances, each with Intel CPU E5-2676@2.40GHz and 1GB of RAM, running on Ubuntu 16.04 and OpenJDK 1.8.

6.3 Preliminary Results

Using *ROG* we have generated a dataset of 10 problem instances ranging from 150 to 1788 parts. We have run each search algorithm for 60 seconds and took out the best solution found. All of the algorithms found solutions which satisfy all of the deadline constraints, at which moment the optimization turns into makespan minimization. Figure 5 shows the makespan values in hours that were required to execute all the schedules, while Figure 6 shows the relative distance from the optimal solution for each of the solvers.

The results indicate that even with an enormous search space it is possible to find reasonable solutions in a matter of minutes using simple local search algorithms and underlying constraint representations. Hill Climbing in particular seems to provide consistent performance, although it is sometimes surpassed by its variants that provide better makespans. Hill Climbing tends to get stuck in local optima traps and the other algorithms try to escape such traps through different relaxations of greediness of neighborhood exploration, leading to occasional successes such as Late Acceptance finding optimal solutions for 336 and 672 orders. The results are not conclusive with regard to which local search algorithm to choose, since the behavior can quickly change based on heuristics and neighborhood definitions, but it shows that local search is able to quickly provide reasonable solutions.

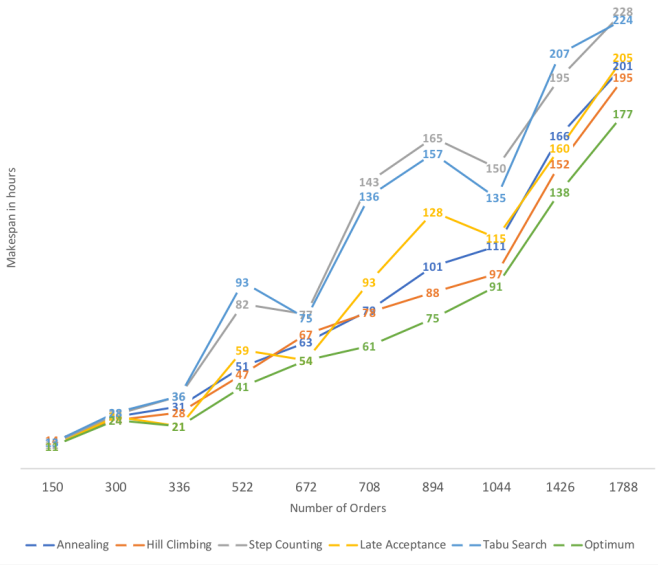


Figure 5: Shows the actual makespan in hours for how long it takes to execute the whole schedule. The graph includes the value for the precomputed optimal schedule and best schedule makespans found for other search algorithms run for 60 seconds.

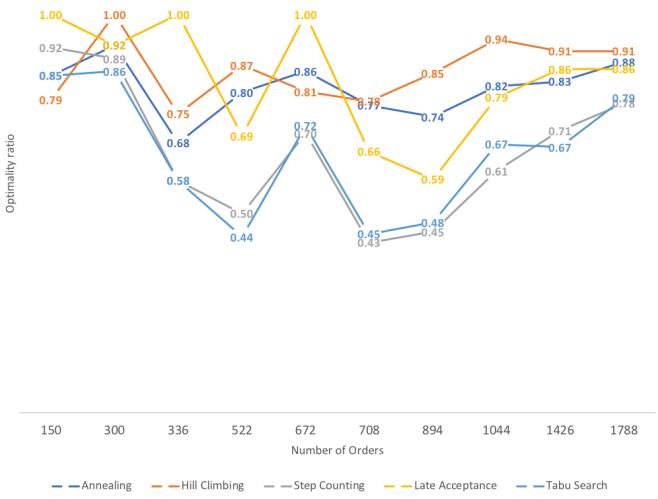


Figure 6: Shows the relative distance from the optimal makespan, computed as optimal/actual, found by different search algorithms running for 60 seconds on each problem instance. The optimum is a known value captured during the problem instance generation.

7 Conclusions

The focus of this paper has been to capture the fundamental difficulty involved in the newly emerging intersection between additive manufacturing, operations research, and artificial intelligence. We have introduced the field of additive manufacturing with its challenges; described and modeled the key optimization problem of nesting parts into builds and scheduling builds to machines while achieving deadlines and minimizing production time; and presented preliminary experimental results using an implementation on top of a constraint solver.

The main contribution of the paper is the new model which we hope to eventually extend into an end-to-end optimization model that captures all the discrete decision making challenges in additive manufacturing.

8 Future Development

This paper is one of the first attempts to describe the computational complexities faced in optimization for additive manufacturing. Multiple sub-problems have been relaxed and deserve further investigation. In particular, explicit temporal reasoning needs to be added to take into consideration operators that work with multiple machines and are required at different phases of the printing process. Then once the parts are printed, we need to take into consideration that they need to be packed and shipped to different destinations from factories at different locations around the world. Another parameter to explore is the fidelity of nesting, where we used only the medium fidelity in the experiments, but results may vary significantly for different fidelity levels and datasets. Finally, while the minimization of the number of tardy builds is a correct and satisfactory condition for achieving all deadlines, it may not accurately reflect the cost associated with the number of tardy parts when it is impossible to make all deadlines.

The experiments deserve further extension in direction of the time given to the planner per problem instance, various densities of constraints including over-constrained problems, and investigation of problems with lots of small parts, big parts, and various distributions among them.

References

[1] ASTM International. *Standard Terminology for Additive Manufacturing Technologies*, 2018. ISO/ASTM 52900.

[2] S. H. Chung, Felix T. S. Chan, and H. K. Chan. A modified genetic algorithm approach for scheduling of perfect maintenance in distributed production scheduling. *Eng. Appl. Artif. Intell.*, 22(7):1005–1014, October 2009.

[3] Brett P. Conner, Guha P. Manogharan, Ashley N. Martof, Lauren M. Rodomsky, Caitlyn M. Rodomsky, Dakesha C. Jordan, and James W. Limperos. Making sense of 3-D printing: Creating a map of additive manufacturing products and services. *Additive Manufacturing*, 1-4:64–76, 2014.

[4] G Costabile, Marcello Fera, Fabio Fruggiero, A Lambiase, and D Pham. Cost models of additive manufacturing: A literature review. 8:263–282, 04 2017.

[5] Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory*. Wiley-Interscience, New York, NY, USA, 1991.

[6] J Coykendall, M Cotteleer, J Holdowsky, and M. Mahto. 3D opportunity in aerospace and defense: additive manufacturing takes flight. *A Deloitte series on additive manufacturing*, 2015.

[7] Rina Dechter. *Constraint Processing*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2003.

[8] Kathryn A Dowsland, Subodh Vaid, and William B Dowsland. An algorithm for polygon placement using a bottom-left strategy. *European Journal of Operational Research*, 141(2):371–381, 2002.

[9] Eric B. Duoss, Todd Weisgraber, Keith Hearon, Cheng Zhu, Ward Small, Thomas R. Metz, John Vericella, Holly D. Barth, Joshua Kuntz, Robert Maxwell, Christopher M. Spadaccini, and Thomas Wilson. Three-dimensional printing of elastomeric, cellular architectures with negative stiffness. *Advanced Functional Materials*, 24(31):4905–4913, 2014.

[10] Marcello Fera, Fabio Fruggiero, Gianluca Costabile, A Lambiase, and D Pham. A new mixed production cost allocation model for additive manufacturing (miprocamam). pages 1–17, 05 2017.

[11] R. L. Graham, E. L. Lawler, J.K. Lenstra, and A.H.G. Rinnooy Kan. Optimization and approximation in deterministic sequencing and scheduling: a survey. *Proceedings of the Advanced Research Institute on Discrete Optimization and Systems Applications of the Systems Science Panel of NATO and of the Discrete Optimization Symposium.*, (5):287–â326, 1979.

[12] Jukka Jylänki. A thousand ways to pack the bin – a practical approach to two-dimensional rectangle bin packing. *retrieved from http://clb.demon.fi/files/RectangleBinPack.pdf*, 2010.

[13] Serope Kalpakjian and Steven R. Schmid. *Manufacturing Engineering and Technology*. Pearson, 7th edition, 2013.

[14] Qiang Li, Ibrahim Kucukkoc, and David Z. Zhang. Production planning in additive manufacturing and 3D printing. *Computers & Operations Research*, 83:157–172, 2017.

[15] Max Lutter-Günther, Stephan Wagner, Christian Seidel, and Gunther Reinhart. Economic and ecological evaluation of hybrid additive manufacturing technologies based on the combination of laser metal deposition and CNC machining. In *Energy Efficiency in Strategy of Sustainable Production*, volume 805 of *Applied Mechanics and Materials*, pages 213–222. Trans Tech Publications, 12 2015.

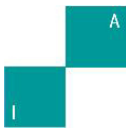
[16] Michael L. Pinedo. *Scheduling: Theory, Algorithms, and Systems*. Springer Publishing Company, Incorporated, 3rd edition, 2008.

[17] Geoffrey De Smet. *OptaPlanner User Guide*. Red Hat and the community, 2018. OptaPlanner is an open source constraint satisfaction solver in Java.

[18] Christopher L. Weber, Vanessa Pena, Maxwell K. Micali, Elmer Yglesias, Sally A. Rood, Justin A. Scott, and Bhavya Lal. The Role of the National Science Foundation in the Origin and Evolution of Additive Manufacturing in the United States. Technical report, IDA Science and Technology Policy Institute, 2013. IDA Paper P-5091.

[19] Wikimedia Commons. https://upload.wikimedia.org/ wikipedia/commons/3/33/Selective_laser_melting_system_schematic.jpg, 2018. User:Materialgeeza / CC-BY-SA-3.0.

Inteligencia Artificial, 21(62) (2018), 53-66
doi: 10.4114/intartif.vol21iss62pp53-66



INTELIGENCIA ARTIFICIAL

http://journal.iberamia.org/

Integrating Meeting and Individual Events Scheduling

Anastasios Alexiadis, Ioannis Refanidis, Iias Sakellariou

Department of Applied Informatics, University of Macedonia,
Egnatia 156, 54636, Thessaloniki, Greece.
talex@uom.edu.gr, yrefanid@uom.edu.gr, iliass@uom.edu.gr

Abstract Automated meeting scheduling is the task of reaching an agreement on a time slot to schedule a new meeting, taking into account the participants' preferences over various aspects of the problem. Such a negotiation is commonly performed in a non-automated manner, that is, the users decide whether they can reschedule existing individual activities and, in some cases, already scheduled meetings in order to accommodate the new meeting request in a particular time slot, by inspecting their schedules. In this work, we take advantage of SelfPlanner, an automated system that employs greedy stochastic optimization algorithms to schedule individual activities under a rich model of preferences and constraints, and we extend that work to accommodate meetings. For each new meeting request, participants decide whether they can accommodate the meeting in a particular time slot by employing SelfPlanner's underlying algorithms to automatically reschedule existing individual activities. Time slots are prioritized in terms of the number of users that need to reschedule existing activities. An agreement is reached as soon as all agents can schedule the meeting at a particular time slot, without anyone of them experiencing an overall utility loss, that is, taking into account also the utility gain from the meeting. This dynamic multi-agent meeting scheduling approach has been tested on a variety of test problems with very promising results.

Keywords: Scheduling, Optimization, constraint satisfaction, Meeting scheduling, Multi-Agents.

1 Introduction

Three friends, Alice, Bob and Charlie, want to meet for a coffee. Alice sends a request and proposes three alternative coffee shops, in different areas of the city, as well as a time frame. The duration of the coffee meeting is estimated to one hour.

Each friend has personal preferences for a variety of aspects of the proposed coffee meeting, such as the particular coffee shop, traveling arrangements needed to go there, as well as the alternative time slots. Their agendas are very tight, so rescheduling might be necessary in order to accommodate the new meeting. While rescheduling, they might consider travel time and expenses, as well as overall utility of the alternative schedules.

Meeting scheduling is a multiobjective optimization process, where multiple agents negotiate on a common time slot to hold the meeting, taking also into account their commitments and preferences. An agent may have already scheduled individual activities, such as tasks with deadlines undertaken, family duties, etc., as well as other meetings. Some of these activities may be flexible and can be rescheduled, while others may be not. Furthermore, some activities might be of lesser importance and the user may decide to abandon them, in order to be able to participate in the meeting. The agent may also have preferences concerning the location and time of the meeting, as well preferences towards scheduling its

ISSN: 1137-3601 (print), 1988-3064 (on-line)

individual activities. It might also have hard or soft constraints, e.g., ordering constraints, between his individual activities.

In this work we assume that the agents do not have preferences over how the meeting will be scheduled; they only care to schedule the meeting. Under this assumption, the simplest case is when the participating agents can find a common free time slot to schedule the meeting, without the need to resort to rescheduling. This time slot is considered optimal for the meeting. Unfortunately, this situation is rather rare. It is quite common that such a time slot does not exist. However, this does not mean that the meeting cannot be scheduled, since participating agents may negotiate to reach an agreement on a time slot that is not initially available to all agents. During the negotiation, time slots not available for all agents are considered, under the condition that they can become "available" if some agents manage to reschedule or even drop already scheduled activities.

The negotiation process can terminate without reaching an agreement, i.e., the agents may conclude that the meeting cannot be scheduled, since rescheduling or dropping other activities reduces the overall utility received by at least one of the agents (we assume that all the agents are veto participants of the meeting, that is, without anyone of them the meeting cannot hold). As a last resort, the agents might try to reschedule other meetings, either with agents not participating in the current negotiation, or among themselves. In that case, a recursive process may commence, where in order to schedule one meeting, other meetings need to be rescheduled, that may initiate rescheduling of other meetings with a different group of agents and so on.

In this work, we consider the problem of multi-agent meeting scheduling, where each agent has a set of already scheduled individual activities, that can be rescheduled in order to accommodate the new meeting. We adopt a rich model of individual activities, with a variety of unary and binary constraints and preferences, that has been proposed in [13]. Meetings are described in a simpler way, that is, duration, temporal domain and utility per agent. We consider the problem of meeting scheduling as a multi-agent constraint satisfaction problem, based on solving many single-agent constraint optimization problems, with each agent attempting to maximize his utility, with the latter being a function of all scheduled activities (individual activities and the meeting), as well as the way they have been scheduled (unary and binary preferences). We do not consider recursive meeting rescheduling, i.e., only individual agent activities can be rescheduled during the process.

The novelty lies in meeting scheduling empowered by rescheduling of individual tasks supporting such a rich model. To the best of our knowledge, there is no other work that combines these two aspects of personal time management.

A powerful scheduler based on greedy construction and stochastic optimization [2] is employed by each agent for constructing a schedule for his individual activities. To schedule a meeting among a set of agents, we have designed and implemented a meta-scheduler, that negotiates with each agent a time slot and a location that can be agreed upon. Each such proposal is evaluated by agents using the scheduler mentioned above. For each proposal, the agent employs distributively the individual activities scheduler to decide whether it can accept each proposed time slot or not. To accept a time slot, the overall utility for the agent when scheduling the new meeting in this time slot should be higher than keeping his original schedule and not scheduling the meeting at all.

In this work, we do not consider strategic behaviour on behalf of the agents. We have tested our approach on a variety of problem instances, with very promising results in terms of performance and scalability.

The rest of the article is structured as follows: Section 2 presents related work. Section 3 presents background information concerning scheduling individual activities. Section 4 defines the integrated scheduling problem and presents the proposed approach. Section 5 presents experimental results and, finally, Section 6 concludes the article and poses future directions.

2 Related Work

Not surprisingly, there exist numerous approaches to agent based meeting scheduling, since it has attracted research interest rather early. In fact, it was considered to be one of the main applications of intelligent personal assistants, a class of agent systems aiming to support the user in performing tedious everyday tasks. As expected, all approaches follow a similar pattern of interaction: there is a group

of agents, each one representing a participant, and possibly a host/meeting agent which coordinates a negotiation process with proposals and counter-proposals on meeting parameters. The agreement is, in most cases, a time slot that maximizes a collective preference value, computed by individual preference values of participating agents. There are quite a few issues in the above setting, such as the number of proposals in each round, the modelling of constraints and preferences, privacy issues, interoperability issues (semantic web), user modelling via learning, "bumping" strategies, etc. In this section we will present in more detail some of the approaches.

One of the earliest agent-based meeting scheduling applications is reported by Jennings and Jackson [9], according to which each participant is "represented" by a meeting scheduling agent (MSA), managing its user's calendar. The procedure followed, presents quite a few similarities with the well known Contract Net protocol: An MSA acting on behalf of its meeting host, announces its intention to arrange a meeting, along with respective time constraints and duration. Participants (their MSAs) respond with bids, that are possible time slots annotated with a preference value, which are used by the meeting host to discover the best common time slot with respect to its global preference value. This announce-bid cycle continues until either a suitable slot is found, or scheduling the meeting is determined to be non-possible, in which case, the initial announcement (meeting duration / time constraints) are changed and the process starts over again.

Sen and Durfee [14] address the problem in a similar setting, however agents report their availability (true/false) on a meeting announcement over n possible slots, along with m alternative slots. In the same work, authors report on rescheduling meeting strategy (bumping) when conflict occurs, maximizing a utility function.

The RETSINA (Reusable Environment for Task-Structured Intelligent Networked Agents) Calendar Agent (RCAL) [12], was aiming to bridge information available on the Semantic Web with the user's personal information managers, e.g., Outlook 2000. RCAL uses Contract Net to negotiate a meeting between participants, a process that involves receiving bids from involved parties to determine an appropriate meeting slot. The advantage of RCAL is that information regarding the user's schedule is obtained automatically through semantically annotated descriptions, and thus allows for a more "accurate" scheduling of meetings.

Chun et al. [7] treat the problem of meeting scheduling from the perspective of user privacy, that is, managing to optimally schedule a meeting without complete knowledge of individual participant preferences. The negotiation takes place between user Secretary Agents (SA), and a Meeting Agent (MA), that coordinates process. Proposals and counter proposals are annotated with a participant's preference value w.r.t. the meeting parameters. In each step the MA collects the set of proposals and generates a new set, sorted on global preference estimation values, that is, values computed from the annotated replies of the SA agents.

In DAS [15] lightweight agents called coordination agents (CA), are created by the participants for each proposed time slot of the meeting. Agents managing the same slot are combined to a single agent. The approach aims at reducing communication costs, by decoupling user agents from CA, and allowing interactions between the latter on the same computational host. Franzin et al. [8] provide a more rigorous constraint modelling of the problem considering hard and soft constraints, with the latter representing user preferences with respect to meeting parameters. The term "common assignment problem with preferences" is introduced to describe the collaboratively scheduling of a meeting, with agents having common variables to set, but possibly different "internal" constraints on these variables. Meeting scheduling proceeds with rounds of proposal-counter proposal, guided by the preference value, in order to reach optimal solutions. In a similar vein, MSRAC (meeting scheduling with reinforcement of arc consistency) [3], handles incremental scheduling, user preferences in the form of soft constraints, user availability by a set of hard constraints and common meeting times with other agents by equality constraints of the meeting time slot, but consider more than one meeting to be scheduled at each time point, i.e. a dynamic iterative value CSP problem. The solving process includes time slot proposals broadcasted by the host to participants, who return their available slots ranked by their preference. Bumping also considered, based on three different heuristic strategies.

CMRadar [11] offers a complete approach to calendar management, including multi-agent meeting scheduling capabilities. If a request for a meeting cannot be accommodated within the current calendar of the agent, then complex strategies involving altering other meetings of lower priority value are employed,

thus engaging the agent in a multi-negotiation process, referred to as the "bumping" problem by Modi and Veloso [10]. In that work, multi-agent meeting scheduling is modelled as a partial incremental Multi-agent Agreement problem (piMAP), where an iterative agreement protocol is employed. The main difference between earlier approaches is the use of bumping heuristic strategies, i.e. strategies to decide whether to modify an existing agent schedule to accommodate the meeting, that allow incremental scheduling. groupTime [6] is a Web based application that bases meeting scheduling on user preferences, in a semi-automatic fashion, i.e. users have the chance to object to an initial meeting time, set by the system. For each possible time slot the user can set its preference explicitly, or the system assigns a value based on the current user's schedule (events). Preference assigned by the system involves extracting schedule-agnostic features for each participant's schedule, based on a group-specific ontology and a set of weights that are defined by machine learning techniques using as training data the user's events and preferences. The set of preferences is then used to determine the meeting time.

The PTIME system [4, 5] is a calendaring assistant that offers meeting scheduling among a variety of time management functionalities. The aim in PTIME was the system to learn user preferences, using machine learning techniques, providing an adaptive personal assistant. The learner module interacts both with the user and the constraint reasoner in order for the system to provide meeting options on a constantly evolving user preferences model.

Chronos [16], is a multi-agent meeting scheduler, in which each user is represented by an Organiser Agent (OA), that learns user's preferences. Each OA represents its beliefs about both its user's preferences and other users he interacts with (acquaintances) using Bayesian Networks (BN). BN express the causal relationships between scheduling parameters, such as time and duration of a meeting and are used to reason about meetings, by computing the probability of an agent to accept a specific proposal. Probabilities are computed for both participants in a meeting and guide a negotiation process, involving proposals and counter proposals in a multi-agent negotiation setting.

Although many approaches consider bumping, i.e. rearranging another meeting in order to accommodate a new request, the issue of rearranging the participants private schedule according to his original constraints with a rich scheduling model, has not been investigated in depth until now.

3 Background: Individual activity scheduling

This section presents the model and the assumptions for scheduling individual activities, as it has been proposed by Refanidis and Yorke-Smith [13] and has been implemented in the SelfPlanner system. This model has been adopted entirely in the present work and has been extended to support also meetings. So, for the completeness of the article, it is purposeful to give a quick overview of it.

A person (equivalently an agent) may have a set T of N individual activities to accomplish. Each activity T_i $(1 \leq i \leq N)$ can be accomplished only during certain time periods, which constitute its temporal domain. This domain is defined as a set of temporal intervals $D_i = [a_{i1}, b_{i1}) \cup [a_{i2}, b_{i2}) \cup \ldots \cup [a_{i,F_i}, b_{i,F_i})$. The duration of an activity may be not fixed (e.g., visiting a museum), so for each activity T_i he can specify its minimum duration d_i^{min} and its maximum duration d_i^{max}, with more scheduled duration resulting in more utility for the person. Activities can be interruptible, that is, they can be scheduled into parts (e.g., reading a book or writing a paper). For each interruptible activity the person can specify its minimum part duration $smin_i$ and its maximum part duration $smax_i$ (e.g., the person wants to devote in book reading time periods not less that 1 hour and no more than 3 hours). The sum of the durations of an activity's parts (non-interruptible activities are considered to have one part) have to be at least d_i^{min} in order for the activity to be considered scheduled.

The set of locations associated with any of the person's activities is Loc, and let M being their number. A matrix $Dist$, not necessarily symmetric, defines the temporal distance between any pair of locations (in the simple model we assume a single means of transportation). We assume that time, temporal intervals and distances are discrete, particularly integers (as the unit of time it is used the minimum temporal interval of interest, e.g., 10, 15 or 30 minutes).

A set of locations $Loc_i \in Loc$ is associated with each activity T_i. A special location called $ANYWHERE$ denotes that the activity can be executed at any location ($ANYWHERE$ is considered to have a temporal distance of 0 from and to any other location). Traveling times between locations have to be taken into account when scheduling activities in time and space; that is, any two temporally adjacent

activities scheduled at different locations (not $ANYWHERE$) should have a large enough temporal gap between them, so as the person can travel between the two locations.

Activities with compatible locations may overlap in time, e.g., watching a lecture while reading emails. Each activity has a utilization value between 0% and 100%, denoting the percentage of the user's attention that the activity requires. The sum of the utilization values of activities scheduled at the same time cannot exceed 1.

The model supports three types of binary constraints between pairs of activities:

- Proximity constraints define a minimum or/and a maximum temporal distance between two activities. For example, the minimum temporal distance between two heavy load activities should be at least 8 hours. Or, the maximum temporal distance between reading a book and writing its synopsis should be at most two days.
- Ordering constraints define an order in time. For example, preparing the slides for the lecture should precede the lecture itself.
- Implication constraints define prerequisite activities. For example, in order to go to the theater, one should first buy tickets.

Note that, especially for the proximity constraints, they can be defined also over the different parts of an interruptible activity.

The overall single-agent problem of scheduling a set of individual activities is formulated as a constraint optimization problem, with the empty plan being the least preferred solution. The objective function is additive over the various sources of utility. These include:

- Each scheduled activity – they can have different utility values.
- Any scheduled duration above the minimum duration of the activity.
- The person's preference over the temporal intervals when the activity has been scheduled (e.g., morning vs evening).
- Proximity, ordering and implication preferences, that is, soft versions of the aforementioned constraints.

The constraint optimization problem is solved by finding values for the decision variables p_i, denoting the number of parts of activity T_i ($1 \leq i \leq N$), t_{ij} (start times), d_{ij} (durations), l_{ij} (locations), for each Activity part T_{ij} ($1 \leq j \leq p_i$), while trying to maximize the sum of the utility sources. The model, with a first solver based on the Squeaky Wheel Optimization framework has been initially presented by Refanidis and Yorke-Smith [13]. A more powerful solver, encompassing post-processing local search techniques (mainly simulated annealing) has been presented by Alexiadis and Refanidis [2].

A technique that allows to produce significantly different alternative plans has been presented by Alexiadis and Refanidis [1]. Producing such alternative plans and retaining them in some form of cache memory may be very useful in meeting scheduling, since they can be used to immediately check whether a meeting can be accommodated in a particular time slot and what is the cost for this accommodation. Particularly, if the time slot is free at any alternative plan kept in cache memory, then the meeting can be scheduled there and the cost is equal to the utility of the current plan minus the utility of the particular alternative one.

4 Collaborative Meeting Scheduling

4.1 Informal Problem Specification

Building on top of the individual activities problem specification, we assume a set of K agents, each agent k of them having its own set of individual activities, according to the model described in the previous section, as well as an already accepted schedule for them S_k, with instantiated decision variables. Furthermore, each agent may already have agreed to participate in meetings with other agents (not necessarily among the K agents of interest for the new meeting), with decided time and location.

One of the K agents invites the rest of them to participate in a new meeting. This agent will act as the coordinator for the particular meeting. The coordinating agent specifies a fixed duration for the meeting, alternative locations for the meeting (or $ANYWHERE$, if no physical presence is required, e.g., teleconferences), as well as a set of temporal intervals when the meeting could be scheduled. Participating in this meeting results in some utility gain for each invited agent (including the coordinator), which may be different for each agent. The meeting will be considered successful and all participants will get the corresponding utility, only in case all of them will be able to participate in it.

Agents may reschedule already scheduled individual activities (in this work we assume that they will not reschedule already scheduled meetings, in order to accommodate the new one). Rescheduling already scheduled activities may result in a utility loss wrt their current schedule S_k. An agent will accept to participate to the meeting at a particular time and location, only in case the utility loss from rescheduling already scheduled individual activities is no more than the utility gain by participating in the meeting. In this work we do not assume extra constraints and preferences, either unary (that is, concerning the meeting by its own) or binary (that is, concerning the new meeting and already scheduled activities).

The meeting scheduling problem may have one or more solutions or even none at all. In the case multiple solutions exist, various criteria could be adopted to select the best among them, thus treating the problem as an optimization problem. One criterion, in that case, is the *sum*, that is, maximizing the total utility over all agents. Another criterion is the *maxmin*, that is, maximizing the minimum utility gain, where the gain for each agent k is computed as u_k minus the utility loss due to rescheduling existing activities.

However, in this work we treat the problem as a constraint satisfaction problem, that is, we try to schedule the meeting with the extra constraint that no agent receives less utility from its new schedule (with the meeting), compared to its old schedule (without the meeting). However, from the point of view of each agent individually, it is a constraint optimization problem, that is, it tries to maximize its total utility received from scheduling its individual activities, plus the utility received from the meeting.

Meetings in this work are described in a simpler way than individual activities, that is, their attributes include just a temporal domain, which is a set of temporal intervals, a fixed duration and a utility value. No other constraints and or preferences are defined over meetings.

4.2 Algorithm

The *Joint Activity Scheduler* (JAS) presented in this Section works in two phases, that implement a message exchange mechanism between the coordinator and the invited agents. The first phase (Algorithm 1) attempts to trivially solve the problem without rescheduling already scheduled activities, by looking for a common empty interval in the current schedules of the agents (also shown in Figure 1). The second phase attempts to schedule the meeting by rescheduling already scheduled individual activities.

In the first phase, the coordinator iterates over all possible meeting time intervals, sending for each interval I a `query-if` message to ask participants if they are available at that time. If all agents reply affirmatively (`inform-t` message), the meeting is scheduled at I. If even a single agent replies negatively (`inform-f` message), then the process is repeated for the next interval in the meeting's temporal domain.

The meeting intervals I_i returned by $ja.all\_intervals()$ are subintervals of the meeting's temporal domain that have a duration equal to the meeting's length. Note that Algorithm 1 does not try to optimize any measure of utility. Since, in case of a successful result, the meeting will be scheduled without rescheduling any existing individual activity, there is no utility loss for any agent due to rescheduling. Furthermore, since the utility gain for any agent by scheduling the meeting is fixed and does not depend on when the meeting has been scheduled, the utility gain for each agent from successfully scheduling the meeting is fixed. So, from the point of view of Algorithm 1, scheduling the meeting is a constraint satisfaction problem (and not a constraint optimization one), so all solutions are equally desirable and the meeting is scheduled at the first commonly free temporal interval found.

On the other hand, if no common free interval is found, Algorithm 2 is employed, in a last attempt to find a common free interval between the agents, using rescheduling of existing individual activities. In that case it is desirable to reduce the overall need for rescheduling, since each call to the individual activity scheduler is a computationally expensive process.

In Algorithm 2 the agents receive four different types of requests from the coordinator. Specifically:

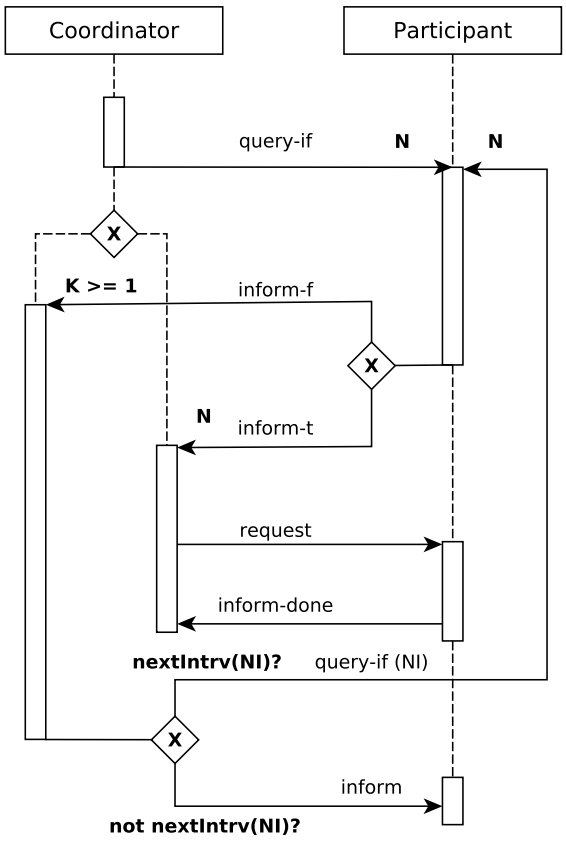


Figure 1: AUML Diagram of the agent interaction during the first phase of the Joint Activity Scheduling.

Algorithm 1 JAS-Phase-1

```
1: procedure JAS-PHASE-1-COORD(ja,users)
2:     for i ←1 to ja.all_intervals().length() do
3:         interval ← ja.all_intervals()[i]
4:         if ¬available(interval, current_plan) then
5:             continue
6:         available ← 0
7:         for k ← 1 to users.length() do
8:             send(agent_k, query-if(interval))
9:         for k ← 1 to users.length() do
10:            if users[k].msg = inform-t then
11:                available ← available + 1
12:            else
13:                break
14:        if available = users.length() then
15:            return interval
16:    return failure
17:
18: procedure JAS-PHASE-1-AGENT
19:    while true do
20:        interval ← get_msg(coordinator)
21:        if available(interval, current_plan) then
22:            send(coordinator, inform-t)
23:        else
24:            send(coordinator, inform-f)
```

- $l4$–$l5$:[1] Request to return to the coordinator their current busy schedule, without providing information about their scheduled activities.
- $l6$–$l12$: Request to attempt to render free a specific temporal interval, by rescheduling their activities; the reply is *success* of $failure$, accompanied with the utility gain in the first case. The utility gain is the improvement of the utility of the rescheduled plan to the utility of the old plan. Note that a $failure$ response occurs not only in cases where it was impossible to schedule the meeting in the particular time slot, but also in cases where scheduling the meeting is possible, however it results in overall utility loss (taking into account the meeting's utility). In case of a *success* reply, the agent must retain in its local memory the corresponding schedule, till the end of the meeting scheduling procedure.
- $l13$–$l15$: Request to schedule the meeting at a particular time interval, following a previously *success* reply for that interval. The agent adopts the corresponding schedule from its local memory and the meeting scheduling process terminates.
- $l16$–$l17$: Request to cancel the rescheduling process (if it is running); this request arises in case another agent has already replied a $failure$ for the time slot under consideration.

In order to minimize the average workload (by reducing the number of messages exchanged), the coordinator asks for the busy hours of all agents for the meeting's temporal interval and computes the overall workload for each possible time slot when the meeting could be scheduled ($l5$–$l7$).[2] The overall workload is stored in the *timeline* array (the length is the maximum number of time slots of the users' plans, whereas each t-th value is the number of events scheduled in the t-th time slot). Time slots with lower overall workload (i.e., the least busy) are given priority to try to schedule the meeting. Furthermore, if a more aggressive approach is adopted, not all possible time slots need to be checked.

[1] Procedure JAS-PHASE-2-AGENT.
[2] Procedure JAS-PHASE-2-COORD.

Algorithm 2 JAS-Phase-2

```
1: procedure JAS-PHASE-2-COORD(ja, users)
2:     timeline ← coord.timeline
3:     min_ints ← {−1 . . .}
4:     min_mins ← {∞ . . .}
5:     for k ← 1 to users.length() do
6:         users[k].timeline ← request_plan(k)
7:         timeline ← timeline + users[k].timeline
8:     for i ← 1 to ja.all_intervals().length() do
9:         int ← ja.all_intervals()[i].start
10:        temp ← timeline[int]
11:        for k ← int + 1 until int + ja.dur do
12:            temp ← temp + timeline[k]
13:        for k ← 1 to NUM_OF_TRIES do
14:            if min_mins[k] > temp then
15:                min_mins.insert(k, temp)
16:                min_ints.insert(k, int)
17:                break
18:    for i ← 1 to NUM_OF_TRIES do
19:        available ← 0
20:        for k ← 1 to users.length() do
21:            if users[k].timeline[min_ints[i]] then
22:                reschedule(k, min_ints[i], ja.dur)
23:        for k ← 1 to users.length() do
24:            if users[k].timeline[min_ints[i]] then
25:                if users[k].wait_for_repl() is yes then
26:                    available ← available + 1
27:                else
28:                    send_cancel_to_all()
29:                    break
30:            else
31:                available ← available + 1
32:        if available = users.length() then
33:            if coord.timeline[min_ints[i]] then
34:                plan ← schedule(problem + [min_ints[i], ja.dur, u_c])
35:                if [min_ints[i], ja.dur] ∈ plan then
36:                    current_plan ← plan
37:                    send_replace_to_all(min_ints[i])
38:                    return [min_ints[i], ja.dur]
39:            else
40:                send_replace_to_all(min_ints[i])
41:                return [min_ints[i], ja.dur]
42:    return −1
```

```
1: procedure JAS-PHASE-2-AGENT
2:     while true do
3:         msg ← get_msg(coordinator)
4:         if msg requests plan then
5:             send_agent(coordinator, current_plan)
6:         else if msg requests reschedule then
7:             plan[msg.start] ← schedule(problem + [msg.start, msg.dur, u_k])
8:             if [msg.start, msg.dur] ∈ plan then
9:                 u_gain = (u(plan[msg.start] − u_k) − u(plan))/u(plan)
10:                send_agent(coordinator, yes, u_gain)
11:            else
12:                send_agent(coordinator, no, 0)
13:        else if msg requests replacing then
14:            if plan[msg.start] then
15:                current_plan ← plan[msg.start]
16:        else if msg requests cancel then
17:            cancel_scheduling()
```

Algorithm 2 sorts the potential time slots to schedule the meeting in ascending order in terms of workload (that is, from the least busy to the busiest) ($l8$–$l17$). NUM_OF_TRIES denotes the number of temporal windows that Algorithm 2 will attempt to schedule there the meeting. Setting this constant to a very large number allows for attempting all possible time slots.

Then, for every attempted time slot the coordinator checks the busy schedule of every agent (as it was sent by the respective agent), and asks the agents that are busy to check whether rescheduling of existing activities is possible ($l18$–$l22$), thus being able to schedule the meeting at the time slot under consideration. If any of the agent replies that rescheduling failed ($l23$–$l31$), the coordinator sends a cancel request to the rest of the agents so as they terminate their rescheduling process, and proceeds to the next time slot in the list.

If all the agents reply positively to the time slot under consideration, including the coordinator ($l32$–$l34$). If the time slot is either available or the coordinator can reschedule it without a great loss to its current plan utility then the procedure succeeds and the coordinator informs the other agents that the meeting has been scheduled at the interval under consideration ($l35$–$l38$). Otherwise the next interval in line is examined until either NUM_OF_TRIES intervals have been evaluated or there are no more time slots to examine ($l39$–$l42$).

5 Experimental Evaluation

We implemented a non-distributed version of the above algorithms, i.e. the code is executed as a single process, in C++.[3]

In the current experimental evaluation we are more concerned with qualitative aspects of the algorithm, and not performance issues, like execution time and number of messages exchanged, thus we do not expect the results presented to change under a distributed, multi-agent version.

In order to evaluate JAS, we created 30 participant activity schedules using the SWO+SA scheduler [2]. These are the predefined fully specified activity plans of the agents participating in the process. The number of activities involved in each schedule, ranges from 6 to 31, in increments of 5. For each schedule size, we selected 5 instances, taken from the literature [1].

We considered four different meetings, all with a duration of four time units, but with a varying temporal domain, as shown in Table 1.

For each meeting, we created 20 random teams of the above agents (a total of 80 experiments), and attempt to schedule the meeting using our JAS implementation. For each meeting, we considered

[3]The full source code and the experiments' results are available online at: `https://drive.google.com/file/d/1vf_c728GK22hsz_tybo--uREZN_RN9-g/view`

Table 1: Meeting Definitions

Meeting id	Duration	Interval 1	Interval 2
1	4	[1..20]	-
2	4	[144..151]	-
3	4	[1..15]	[16..20]
4	4	[143..147]	[148..152]

participant populations of different size, from two agents to a max of five.

We set the meeting utility u_k for all users to 4, which is a value lower than the lowest activity utility in these problem instances. It should be noted that in the scheduling problem instances activity utilities were ranging from $5--12$. This choice was selected so as the scheduler tries to accommodate the meeting without "dropping" an activity previously scheduled, as explained later in the section. For all experiments we have set $NUM\_OF\_TRIES = 5$.

Table 2: Results of applying JAS to Meeting Scheduling Problems.

TS	*2 Participants*			*3 Participants*			*4 Participants*			*5 Participants*		
MID	Interval	Ph	Rs	Interval	Ph	Rs	Interval	Ph	Rs	Interval	Ph	Rs
	[7..11]	1	0	[16..20]	2	2	[1..5]	2	1	[9..13]	2	2
	[9..13]	1	0	[9..13]	1	0	[1..5]	2	3	[1..5]	2	4
1	[10..14]	2	1	[16..20]	2	2	[16..20]	2	1	[1..5]	2	3
	[6..10]	1	0	[7..11]	2	2	[1..5]	2	3	[16..20]	2	1
	[10..14]	1	0	[1..5]	2	3	[16..20]	2	3	[8..12]	2	2
	[147..151]	2	2	[145..149]	2	3	[146..150]	2	4	[147..151]	2	4
	[147..151]	2	1	[147..151]	2	2	[146..150]	2	4	[144..148]	2	4
2	[144..148]	2	2	[144..148]	2	3	[146..150]	2	3	[144..148]	2	5
	[144..148]	2	1	[147..151]	2	2	[144..148]	2	3	[144..148]	2	5
	[144..148]	2	2	[144..148]	2	2	[144..148]	2	4	[147..151]	2	3
	[16..20]	2	1	[8..12]	2	1	[16..20]	2	2	[1..5]	2	3
	[16..20]	2	1	[11..15]	2	2	[16..20]	2	2	[11..15]	2	5
3	[16..20]	2	2	[16..20]	2	1	[16..20]	2	2	[16..20]	2	2
	[16..20]	2	2	[8..12]	2	2	[16..20]	2	1	[16..20]	2	4
	[7..11]	1	0	[16..20]	2	3	[10..14]	2	1	[16..20]	2	2
	[143..147]	2	2	no sched.	2	0	[143..147]	2	4	no sched.	2	0
	no sched.	2	0	no sched.	2	0	no sched.	2	0	[143..147]	2	5
4	[143..147]	2	2	[143..147]	2	3	[143..147]	2	4	[143..147]	2	5
	[143..147]	2	2	[143..147]	2	3	no sched.	2	0	[143..147]	2	5
	[143..147]	2	2	[143..147]	2	3	[143..147]	2	4	no sched.	2	0

Table 2 presents an overview of the results of the experiments. The table depicts for all teams of agents (TS stands for Team Size), the scheduled interval found for each meeting (MID is the Meeting ID), the final JAS phase completed (column Ph) and the number of user schedules that required changed to accommodate the meeting (column Rs). In the Rs column we do not count every call to the scheduler but only the number of users whose plans were rescheduled at the end of the negotiation process. For the *schedule*() function of the second phase of JAS we called the SWO+SA scheduler[4] as an external process. The scheduler was called on a modified version of the problem instance, of the user being rescheduled, that included the meeting at the interval being tested.[5] As the SWO+SA scheduler is an optimizer, that is it tries to find the plan with the maximum utility, it would not omit an activity that gives a greater utility to include an activity that gives a lower one. By providing a low u_k value to JAS, this ensures that the SWO+SA scheduler would not remove one of the already scheduled activities to include the meeting.

[4]The SWO-SA scheduler is not an exact optimization algorithm.

[5]With a utilization value of 100%.

However, the SWO+SA scheduler could decrease the duration of a scheduled activity T_i though, if the condition $d_i^{min} < d_i^{max}$ was consistent and thus decrease the utility of the agent.

JAS managed to find a common interval in 73 out of the 80 runs of the algorithm. Since a small value was set to the NUM_OF_TRIES parameter, it is possible that a common interval could be found in the remaining 7 unsuccessful cases with more rescheduling tries. Although setting this parameter to higher values would increase the number of scheduler calls significantly, it allows testing every possible meeting interval. It should be noted that out of the 73 scheduled instances only 6 were scheduled in the first phase of JAS, i.e. solved trivially. In the rest, the meeting was successfully scheduled in the second phase, which attempted to reschedule the minimum number of users by using the *min_ints* list.

The utility change for the rescheduled users was minor in most of the cases. Table 3, shows the maximum, minimum and average percentage of utility change in the agent's plans without taking into account the added utility of the introduced meeting, in order to evaluate how rescheduling affected the previous activity plan of the agent. The worst drop was −15.62%. As the SWO+SA scheduler is stochastic there were many cases were there were even minor utility improvements in the rescheduled plans of the users. The best improvement was 2.46%. Obviously, for the cases that the meeting was not successfully scheduled, the agent utilities did not change and these cases are marked with a dash (-). From the total 175 rescheduled users, there were 95 utility drops in the rescheduled plans and 74 minor improvements. The rest had plans with the same utility.

Table 3: Utility Gains Change (%) when Scheduling a Meeting

TS	*2 Participants*			*3 Participants*			*4 Participants*			*5 Participants*		
MID	max	min	avg	max	min	avg	max	min	avg	max	min	avg
	0	0	0	0	-2.26	-0.92	0	-0.87	-0.22	0.35	-2.46	-0.42
	0	0	0	0	0	0	0.08	-2.69	-0.78	0.29	-2.63	-0.41
1	0	0	0	0.02	-1.85	-0.61	0.05	0	0.01	**2.46**	-6.61	-0.85
	0	0	0	0.31	0	0.1	0.2	-2.35	-0.73	0	-0.02	0
	0	0	0	0	-7.28	-3.25	0	**-15.62**	-3.91	0.11	-1.67	-0.31
	0.48	-0.41	0.04	0.43	-3.14	-0.82	0.05	-1.44	-0.73	0.53	-1.07	-0.23
	0	-0.04	-0.02	0.45	-0.1	0.12	0.37	-0.14	0.12	0.97	-3.94	-0.52
2	-0.17	-6.05	-3.11	0.32	-5.15	-1.71	1.19	-3.43	-0.3	0.68	-15.08	-3.39
	0.09	0	0.05	0	-1.09	-0.54	0.36	-15.14	-3.7	0.07	-5.75	-1.7
	0.26	-0.36	-0.05	0.97	-0.61	0.12	0.06	-0.81	-0.23	0.09	-0.3	-0.05
	0.03	0	0.02	0	-0.32	-0.11	0	-0.52	-0.26	0.19	-0.47	-0.09
	2.13	0	1.07	0.41	-0.21	0.07	1.19	0	0.31	-0.12	-0.93	-0.43
3	1.17	-0.19	0.49	0.08	0	0.03	0.85	-0.9	-0.01	0.16	0	0.06
	1.15	-0.99	0.08	0	-0.59	-0.25	0	-0.26	-0.07	0.15	-0.15	-0.03
	0	0	0	1.74	-0.98	0.35	0	-0.03	-0.01	0.2	-1.47	-0.25
	0.06	0.03	0.05	-	-	-	0.28	-0.59	0.02	-	-	-
	-	-	-	-	-	-	-	-	-	-0.07	-1.23	-0.6
4	0.16	0.02	0.09	1.87	0.07	0.76	0.78	-0.88	-0.06	0.2	-1.18	-0.3
	0.98	-0.43	0.28	0.27	-0.95	-0.21	-	-	-	0.14	-1.29	-0.38
	-0.04	-0.26	-0.15	1.23	-0.17	0.37	0.1	-1.45	-0.35	-	-	-

6 Conclusions

The work described in this article addresses the problem of automated meeting scheduling between a number of self-interested agents, under a rich model of individual activities and a typical model for the meeting. As commonly used in other work, the multi-agent negotiation process that takes place is driven by a coordinator agent responsible for generating proposals, to which agents reply based on utility values derived from employing a greedy construction and stochastic optimization scheduler to the new scheduling problem that includes the proposal. Initial experimental evaluation, demonstrates that the approach performs well, reaching an agreement on the meeting time slot in the majority of experiments.

There are several directions that the present work can be extended. Currently, we do not consider alternative meeting locations in proposals, a feature that would certainly increase the number of necessary negotiation rounds, and although such an option in most business cases might not be of great interest, it might be interesting in other cases. JAS can also be extended so that meetings could be converted to full joint-activities, that is both temporal preferences (i.e., schedule the meeting at morning/noon/evening) and binary preferences could be defined over them by each agent, as in individual activities.

More experiments should also be conducted to evaluate the proposed approach. JAS is deterministic and should always arrive to the same common scheduled interval (given a deterministic rescheduler), but SWO+SA is stochastic. When the interval being tested by SWO+SA is given a high enough utility, JAS should always arrive to the same common scheduled interval, while the rest of the users' rescheduled plans may be different though with different utilities between runs. By giving the interval being tested a low enough utility, it is possible that JAS could output different common scheduled intervals between runs. This needs to be tested. Moreover, different parameters for the algorithm, as well as different heuristics for the ordering of the intervals to be evaluated could also be tested. Future experiments should also measure the number of tested intervals and provide a metric for the computational efficiency of the proposed algorithms.

Rearranging agent meetings to accommodate the one under negotiation, i.e., "bumping", presents also a very interesting research direction that we aim to investigate, since there is a number of interesting questions that arise, as for example when should this "recursive" process terminate.

Finally, an experimental evaluation of the proposed algorithms in a real distributed environment would allow to measure the algorithms performance in terms of execution time and number of messages exchanged in order to fully evaluate its potential in a real-life setting. Obviously, integrating the proposed algorithms with modern calendar applications is the ultimate goal.

References

[1] Anastasios Alexiadis and Ioannis Refanidis. Alternative plan generation and online preference learning in scheduling individual activities. *International Journal on Artificial Intelligence Tools*, 25(3):1–28, 2016.

[2] Anastasios Alexiadis and Ioannis Refanidis. Optimizing individual activity personal plans through local search. *AI Communications*, 29(1):185–203, 2016.

[3] Ahlem BenHassine and Tu Bao Ho. An agent-based approach to solve dynamic meeting scheduling problems with preferences. *Engineering Applications of Artificial Intelligence*, 20(6):857–873, September 2007.

[4] Pauline Berry, Melinda Gervasio, Bart Peintner, and Neil Yorke-Smith. Balancing the needs of personalization and reasoning in a user-centric scheduling assistant. Technical report, SRI INTERNATIONAL MENLO PARK CA ARTIFICIAL IN[U+2121]LIGENCE CENTER, 2007.

[5] Pauline M. Berry, Thierry Donneau-Golencer, Khang Duong, Melinda T. Gervasio, Bart Peintner, and Neil Yorke-Smith. Evaluating User-Adaptive Systems: Lessons from Experiences with a Personalized Meeting Scheduling Assistant. In *IAAI*, volume 9, pages 40–46, 2009.

[6] Mike Brzozowski, Kendra Carattini, Scott R. Klemmer, Patrick Mihelich, Jiang Hu, and Andrew Y. Ng. groupTime: preference based group scheduling. In *Proceedings of the SIGCHI conference on Human Factors in computing systems*, pages 1047–1056. ACM, 2006.

[7] Andy Chun, Hon Wai, and Rebecca Y. M. Wong. Optimizing agent-based meeting scheduling through preference estimation. *Engineering Applications of Artificial Intelligence*, 16(7):727–743, October 2003.

[8] Maria Sole Franzin, E. C. Freuder, F. Rossi, and R. Wallace. Multi-agent meeting scheduling with preferences: efficiency, privacy loss, and solution quality. In *Proceedings of the AAAI Workshop on Preference in AI and CP*, 2002.

[9] Nick R. Jennings and A. J. Jackson. Agent-based meeting scheduling: A design and implementation. *Electronics letters*, 31(5):350–352, 1995.

[10] Pragnesh Jay Modi and Manuela Veloso. Bumping strategies for the multiagent agreement problem. In *Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems*, pages 390–396. ACM, 2005.

[11] Pragnesh Jay Modi, Manuela Veloso, Stephen F. Smith, and Jean Oh. Cmradar: A personal assistant agent for calendar management. In *Agent-Oriented Information Systems II*, pages 169–181. Springer, 2005.

[12] Terry R. Payne, Rahul Singh, and Katia Sycara. Rcal: A case study on semantic web agents. In *Proceedings of the first international joint conference on Autonomous agents and multiagent systems: part 2*, pages 802–803. ACM, 2002.

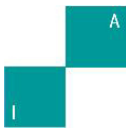
[13] Ioannis Refanidis and Neil Yorke-Smith. A constraint-based approach to scheduling an individual's activities. *ACM Trans. Intell. Syst. Technol.*, 1(2):12:1–12:32, December 2010.

[14] Sandip Sen and Edmund H. Durfee. A contracting model for flexible distributed scheduling. *Annals of Operations Research*, 65(1):195–222, 1996.

[15] F. Wang. Adaptive meeting scheduling for large-scale distributed groupware. *BT technology journal*, 21(4):138–145, 2003.

[16] Alejandro Zunino and Marcelo Campo. Chronos: A multi-agent system for distributed automatic meeting scheduling. *Expert Systems with Applications*, 36(3):7011–7018, April 2009.

Inteligencia Artificial, 21(62) (2018), 67-74
doi: 10.4114/intartif.vol21iss62pp67-74



INTELIGENCIA ARTIFICIAL

http://journal.iberamia.org/

Characterizing and Computing All Delete-Relaxed Dead-ends

Christian Muise
IBM Research
christian.muise@ibm.com

Abstract Dead-end detection is a key challenge in automated planning, and it is rapidly growing in popularity. Effective dead-end detection techniques can have a large impact on the strength of a planner, and so the effective computation of dead-ends is central to many planning approaches. One of the better understood techniques for detecting dead-ends is to focus on the delete relaxation of a planning problem, where dead-end detection is a polynomial-time operation. In this work, we provide a logical characterization for not just a single dead-end, but for *every* delete-relaxed dead-end in a planning problem. With a logical representation in hand, one could compile the representation into a form amenable to effective reasoning. We lay the ground-work for this larger vision and provide a preliminary evaluation to this end.

Keywords: deadends, dead-ends, knowledge compilation, d-DNNF, BDD, SDD

1 Introduction

Learning conflicts and reasoning about them has long been recognized as an important component in many fields of artificial intelligence and optimization. Recently, a type of conflict found in planning that represents a state where the goal cannot be achieved, referred to as a *dead-end*, has received increased attention in the field. This interest culminated in the first ever International Planning Contest on Unsolvability in 2016 [15], which required the planners to detect whether or not the initial state of a planning problem was a dead-end. Effective techniques for dead-end detection are useful for determining problem unsolvability, as evidenced by the winning planners in the contest, as well as solving classical planning problems [7, 13], non-deterministic planning problems [17], and probabilistic planning problems [12, 3].

Perhaps one of the most well understood dead-end detection techniques is to search for a solution in the delete relaxation of a problem (i.e., assuming that actions in our model never delete what is true). Because solvability of the delete relaxation is a polynomial time operation, we can determine if a state is a dead-end in the delete relaxation in polynomial time as well. If this is the case, we can further conclude that the original state is a dead-end; if no plan exists in the delete relaxation, then no plan exists in the original problem.

In this work, we focus on characterizing what it means for a state to be a delete-relaxed dead-end, and we do so by introducing a novel logical encoding of the problem. From the encoding we can compile the theory into an equivalent representation that compactly represents all delete-relaxed dead-ends of a particular minimal form, while at the same time allowing efficient queries or modifications over the theory. For example, one could quantify the likelihood that a partially observable state is in

ISSN: 1137-3601 (print), 1988-3064 (on-line)

a dead-end, or manipulate the representation to compute all states that could lead to a delete-relaxed dead-end through the application of a particular action by using standard logical operations. The logical representation provides interesting insights into the structure of the delete-relaxed dead-end detection problem in relation to other planning encodings, and we elaborate on these connections throughout the paper.

Our contributions are 3-fold: (1) we provide a simple logical encoding to represent the set of delete-relaxed dead-end states in a STRIPS planning problem; (2) we present an extension that compiles away some of the naïve encoding to give us an equivalent encoding that is easier to reason with; and (3) we present a preliminary evaluation on a range of domains and knowledge compilers.

2 Preliminaries

Delete-Relaxed STRIPS Planning

In this work, we assume a STRIPS model of planning [6]. A problem is defined as a tuple, $\langle \mathcal{F}, \mathcal{I}, \mathcal{A}, \mathcal{G} \rangle$, where $\mathcal{F}$ is the set of fluents in the problem, $\mathcal{I} \subseteq \mathcal{F}$ is the initial state (we assume all fluents not in $\mathcal{I}$ are false initially), $\mathcal{A}$ is the set of actions in the problem and $\mathcal{G} \subseteq \mathcal{F}$ is the goal that the planner must achieve. Every action $a \in \mathcal{A}$ is described by its precondition $\text{PRE}(a) \subseteq \mathcal{F}$ and add effects $\text{ADD}(a) \subseteq \mathcal{F}$. Traditionally, there is also a set of delete effects, $\text{DEL}(a) \subseteq \mathcal{F}$, but for this work we are only concerned with delete-relaxed planning: i.e., $\forall a \in \mathcal{A}, \text{DEL}(a) = \emptyset$. A complete state $s \subseteq \mathcal{F}$ describes what is true in the world, while every fluent in $\mathcal{F} \setminus s$ is presumed to be false. An action a is applicable in state s if and only if $\text{PRE}(a) \subseteq s$, and the result of applying a in s is defined to be $s \cup \text{ADD}(a)$. Note that because there are no delete effects, applying an action in a state can only add fluents to the state: i.e., make more of the fluents true. Further, because all of the preconditions are fluents as well, when an action is applicable it will remain applicable. We will use the following to refer to the actions that add a particular fluent:

$$adders(f) = \{a \mid a \in \mathcal{A} \text{ and } f \in \text{ADD}(a)\}$$

A planning problem, $\langle \mathcal{F}, \mathcal{I}, \mathcal{A}, \mathcal{G} \rangle$, is *solvable* if a sequence of action applications transforms the state $\mathcal{I}$ into one where all of the fluents in $\mathcal{G}$ are true, and it is *unsolvable* otherwise. The state s is a delete-relaxed dead-end for problem $\langle \mathcal{F}, \mathcal{I}, \mathcal{A}, \mathcal{G} \rangle$ if and only if $\langle \mathcal{F}, s, \mathcal{A}, \mathcal{G} \rangle$ is unsolvable.

Satisfiability

We use the standard notion of Boolean logic to characterize the delete-relaxed dead-ends in a planning problem. Here we describe the basic concepts and notation, but the reader is referred to [1] for further details. The task of Satisfiability (or simply SAT) is to find a satisfying assignment to a set of Boolean variables given a logical formula. The standard representation for a logical formula, and the one we adopt, is Conjunctive Normal Form (CNF). A CNF formula is a conjunction of clauses, and each clause is a disjunction of literals; either a Boolean variable or its negation. For example, the following CNF formula,

$$(x \vee \neg y) \wedge (\neg x \vee y)$$

has two satisfying assignments: $(x = \top, y = \top)$ and $(x = \bot, y = \bot)$. While CNF is a natural form to express many problems, reasoning with it is a difficult task. Many other forms have been proposed in the literature that are more amenable to reasoning, and *knowledge compilation* is the task of converting from one form (typically CNF) to another in order to make reasoning easier [5]. We forgo describing the alternative forms here, but provide some empirical results in Section 4 on the difficulty of knowledge compilation from our proposed CNF encoding to a variety of target forms.

The final notion we will use is *projection*. The projection of a satisfying assignment onto a subset of the variables is simply the portion of the assignment that involves variables in the subset. We use the notion of projection to focus on only one aspect of the model, and it may be the case that many full assignments will map to the same projected assignment.

3 Encoding All Delete-Relaxed Dead-ends

To reason about all of the delete-relaxed dead-ends for a planning problem, we must construct a CNF where satisfying assignments correspond to delete-relaxed dead-ends. However, we do not want just any representation of delete-relaxed dead-ends, but instead one that captures the core reason that a state cannot achieve the goal. This can be useful for interpretability of deadends, but additionally (as we see later) allows for a far more compact encoding than the standard Planning-as-SAT encodings. To that end, we will only model delete-relaxed dead-ends that are in a particular fixed-point of delete-relaxed reachability.

Definition 1 (Fixed-Point State) *We say that a state s in a delete-relaxed planning problem $\langle \mathcal{F}, \mathcal{I}, \mathcal{A}, \mathcal{G} \rangle$ is a fixed-point state iff $\forall a \in \mathcal{A}, \text{PRE}(a) \not\subseteq s$ or $s \cup \text{ADD}(a) = s$.*

Intuitively, a state is a fixed-point state whenever applying additional actions will have no effect on the state (either an action is not applicable, or adds no new fluents to the state). Every delete-relaxed dead-end will have a corresponding fixed-point state s where $\mathcal{G} \not\subseteq s$, and it is the set of delete-relaxed dead-end fixed-point states that we model with our SAT encoding.

Not only are fixed-point delete-relaxed dead-end states more easy to work with when modeling, but they also represent the relevant core of what is causing a state to be a dead-end. For any state s, its corresponding fixed-point state s' will include everything achievable from s (and thus $s \subseteq s'$). With our encoding, we capture what does *not* hold in the fixed-point state (i.e., $\mathcal{F} \setminus s'$), and this will always be more general (i.e., fewer fluents) than what does not hold in s.

First, we present a simple but naïve encoding that achieves our objective of modeling the fluents not achievable in a fixed-point delete-relaxed dead-end. Then we prove the correctness of the encoding, and show an alternative encoding that captures the same set of solutions using fewer variables. The alternative encoding provides a better representation for knowledge compilers to work with due to the reduction in variables. We conclude with a discussion of some of the interesting properties that our encodings have, and their relation to existing notions in the literature.

3.1 Naive Encoding

The key insight that we use for our encoding is to focus on what *cannot* be achieved, rather than what *can* be achieved. This is in stark contrast with the vast majority of existing SAT encodings for planning-related problems. We should note that our aim is to model the space of all states that are a delete-relaxed dead-end for a particular problem, and not just identify if the initial state is a delete-relaxed dead-end.

Because we are interested in the states from which the goal cannot be achieved, our encoding focuses on those aspects of the problem that are *unachievable*. A fluent f is unachievable from state s whenever the problem $\langle \mathcal{F}, s, \mathcal{A}, \{f\} \rangle$ is unsolvable. Similarly, an action a is unachievable from state s whenever the problem $\langle \mathcal{F}, s, \mathcal{A}, \text{PRE}(a) \rangle$ is unsolvable.

Viewing the task of representing all delete-relaxed dead-ends in terms of what cannot be achieved leads us to a simple CNF encoding where the variables are defined as:

- $x_{\bar{f}}$: For every fluent $f \in \mathcal{F}, x_{\bar{f}}$ represents the fact that f is unachievable.
- $x_{\bar{a}}$: For every action $a \in \mathcal{A}, x_{\bar{a}}$ represents the fact that a is unachievable.

The clauses that we use to characterize all fixed-point delete-relaxed dead-ends are as follows:

$$\bigvee_{f \in \mathcal{G}} x_{\bar{f}} \tag{1}$$

$$x_{\bar{a}} \rightarrow \bigvee_{f \in \text{PRE}(a)} x_{\bar{f}} \qquad \forall a \in \mathcal{A} \tag{2}$$

$$x_{\bar{f}} \rightarrow x_{\bar{a}} \qquad \forall f \in \mathcal{F}, \; \forall a \in adders(f) \tag{3}$$

The intuition behind each of the clause types is as follows: (1) some aspect of the goal must be unachievable; (2) if an action is unachievable, then some aspect of its precondition must be unachievable; and (3) if a fluent is unachievable, then every action that could add it must be unachievable.

The encoding is relatively small, and dominated by clauses of type 2 and 3. For type 2, each clause is only as large as the precondition of the action in question, and there are only $|\mathcal{A}|$ clauses of this type. For type 3, each clause is binary, and there will be $\sum_{f \in \mathcal{F}} |adders(f)|$ such clauses (a small number in typical planning problems). We will use $CNF(P)$ to refer to the encoding of planning problem P.

There are two special cases worth noting. First, if a fluent f has no action that can add it (i.e., $adders(f) = \emptyset$), then there will be no clauses of type 3: if f is false in the state of the world, it will remain unachievable. Second, if an action has no precondition then it is trivially *not* unachievable (i.e., it can always be executed, and its effects achieved).

A satisfying assignment will stipulate which actions and fluents are deemed unachievable. The *corresponding state* of a satisfying assignment is the state s defined from the fluent variable as $\{f \mid x_{\bar{f}} = \bot\}$. We can now establish the theoretical connection between the set of fixed-point delete-relaxed dead-ends for a problem P and $CNF(P)$:

Theorem 2 *The set of satisfying assignments for $CNF(P)$, projected to the fluent variables, corresponds one-to-one with the set of fixed-point delete-relaxed dead-ends of P.*

Proof Sketch. We first establish that the projection of any satisfying assignment onto the fluent variables corresponds to a fixed-point delete-relaxed dead-end. Formulae 2 and 3 together ensure that the assignment is a fixed-point state: if it was not, then some action a must be applicable in the corresponding state with $f \in \text{ADD}(a)$ and f marked as being unachievable: i.e., $x_{\bar{a}}$ and $x_{\bar{f}}$ hold in the assignment while a is applicable in the corresponding state. Note, however, that this leads to a contradiction: the contrapositive of formula 2 stipulates that if every precondition fluent of a is not unachievable, then a is not unachievable as well. Finally, the assignment must correspond to a delete-relaxed dead-end, as it is a fixed-point state and some aspect of the goal is unachievable (following formula 1).

For the other direction, consider a candidate delete-relaxed dead-end fixed-point state s. We construct a corresponding satisfying assignment by setting the following variables to true and all others false:

$$\{x_{\bar{f}} \mid f \notin s\} \cup \{x_{\bar{a}} \mid \text{PRE}(a) \not\subseteq s\}$$

Formulae 2 and 3 naturally follow from the properties of s being a fixed-point state, and 1 holds from the fact that s cannot contain the complete goal (as it is a dead-end). □

3.2 Fluent-based Encoding

While the above encoding is simple and intuitive, it unnecessarily contains information in the form of variables for action unachievement. Here, we derive an alternative encoding that removes the variables corresponding to the actions. We do so by using the transitive property of the implications in formulae 3 and 2 above; by combining them into a single formula, we obtain the following:

$$x_{\bar{f}} \rightarrow \bigvee_{f' \in \text{PRE}(a)} x_{\bar{f}'} \qquad \begin{array}{c} \forall f \in \mathcal{F}, \\ \forall a \in adders(f) \end{array} \tag{4}$$

Formulae 1 and 4 combined capture the precise set of delete-relaxed dead-ends for a problem. The number of variables is reduced to just the number of fluents in the problem, but the number of clauses is increased as a result: one clause of size $|\mathcal{G}|$ for the goal, and another $\sum_{f \in \mathcal{F}} |adders(f)|$ clauses of size $|\text{PRE}(a)|$. Just as $adders(f)$ is typically small, so is the size of $\text{PRE}(a)$. We found this increase to be negligible. Because the correctness of the fluent-based encoding follows directly from the logical combination of formulae 2 and 3 to produce formula 4, we forgo a formal proof here.

3.3 Discussion

A number of SAT encodings have been proposed for modeling planning problems, but to the best of our knowledge they all focus on modeling the existence of plans in some form. The idea of planning-as-SAT was pioneered in the early 1990's as a method for solving planning problems [9]. Recent advances

Domain	bddmin		c2d		cnf2bdd		DSHARP		minic2d		sharpSAT	
	act	flu	act	flu	act	flu	act	flu	act	flu	act	flu
airport (50)	0	2	7	**12**	2	6	3	4	7	10	4	5
floortile (20)	0	0	10	10	0	7	2	2	4	**14**	5	6
mystery (30)	0	4	7	8	2	6	6	6	6	6	7	**9**
parcprinter (20)	0	0	14	**16**	0	0	0	0	7	8	0	0
pegsol (20)	0	0	**20**	**20**	0	17	0	**20**	0	14	**20**	**20**
sokoban (20)	0	0	9	**10**	0	0	0	0	2	5	3	3
trucks (30)	0	1	10	**11**	0	6	1	3	5	5	2	4
woodworking (20)	0	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**
ALL (210)	0	8	78	**88**	5	43	13	36	32	63	42	48

Table 1: # of Problems Compiled. (Brackets): all encoded problems. **Bold**: greatest number of problems compiled.

have drastically improved the performance of planning-as-SAT [19], focused on optimal planning [20], computation of heuristics [2], planning with multi-valued variable representations [8], and computing maximally flexible plans [14]. All of these approaches, however, share a common thread: a satisfying assignment corresponds to a plan.

In contrast, our encodings capture (fixed-point) states of the delete-relaxed problem where no plan exists. Many planning-as-SAT encodings use a layered approach and must fix the number of actions in a plan without knowing in advance what depth would suffice. For those that do not use a layered approach (e.g., [20, 14]), a common bottleneck is the number of clauses required for preventing causal self support: e.g., action a_1 adds f_1, which enables a_2, which adds f_2, which enables a_1, etc. To rule out any such causal loops in a plan, a cubic number of clauses are required to model the transitive closure of "support". Our encodings do not need to pay this high cost in encoding complexity. Intuitively, this is because the fundamental property we are modeling (i.e., if a fluent is unachievable) is universally quantified: a fluent is unachievable if and only if *every* action that adds it is unachievable. This is in contrast with the fundamental *existential* property that non-layered planning-as-SAT encodings aim to capture: a fluent is achievable when *at least one* action (or the initial state) is able to achieve it. [1]

The final connection of interest is prime implicants and minimal delete-relaxed dead-ends. Prime implicants describe only those variable settings required for a Boolean assignment to be satisfying, and minimal dead-ends describe only those unachievable fluents required to be a dead-end, so there is reason to believe they would coincide. However, note that relaxing one Boolean variable in the encodings above amounts to saying that we have a fixed-point delete-relaxed dead-end *regardless* of whether or not a selected fluent or action is unachievable. Changing the setting to just one Boolean variable can easily cause the assignment to no longer be satisfying, and as a result the prime implicants do not correspond directly to the minimal delete-relaxed dead-ends. The final note of interest is with respect to minimal delete-relaxed dead-ends: i.e., a delete-relaxed dead-end with the fewest number of unachievable fluents specified. Every minimal delete-relaxed dead-end will correspond directly to some satisfying assignment of the proposed encoding. This follows from the fact that the set of satisfying assignments correspond to every fixed-point delete-relaxed dead-end, which necessarily includes the minimal delete-relaxed dead-ends, and it also underscores the subtle nature of the $x_{\bar{f}}$ variables: when set to false, many of the constraints become satisfied due to the structure of (3) or (4).

[1] The ability for the initial state to support a fluent is typically captured through an introduced "initial state action".

4 Evaluation

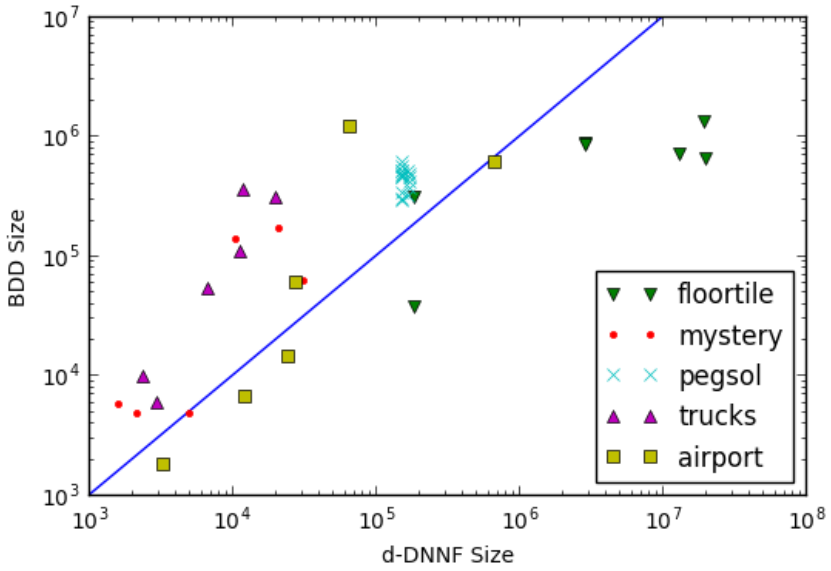


Figure 1: Size of the d-DNNF generated by c2d compared with the size of the BDD generated with cnf2bdd.

There are many possibilities for using the logical characterization of delete-relaxed dead-ends that we presented in Section 3. For our preliminary investigation, we evaluate the potential of compiling the theory into many of the popular target languages for knowledge compilation: (1) Deterministic Decomposable Negation Normal Form or d-DNNF [5]; (2) Sentential Decision Diagrams or SDD [18]; and (3) Binary Decision Diagrams or BDD [1]. We evaluate both encodings using the range of compilers that are available. For d-DNNF this includes Dsharp [16], c2d [4], and sharpSAT [23]. While sharpSAT is not a compiler, the Dsharp compiler is built on top of the sharpSAT code-base and so it provides a natural bound on performance for Dsharp. For SDD, we use the canonical compiler miniC2D [18]. Finally, for BDD, we use both BDD-MiniSAT [24] and the CUDD software package [22]; referred to here as cnf2bdd.

We implemented software to convert STRIPS planning problems (specified in PDDL) into CNF corresponding to the two encodings (the converter, benchmarks, and data will be released along with publication). All experiments were conducted on a 3.4Ghz Linux Desktop, and every trial was limited to 30min and 4Gb of time and memory. We used a selection of common benchmarks known to have dead-ends. In Table 1 we show the coverage for each of the tested compilers (in their best configuration) on each of the encodings. While the comparison across compilers is not direct, as they compile the encoding into different target languages, we can make some general observations.

The first aspect to note is that the fluent-based encoding strictly dominates the action-based one. Despite the increase in number of clauses, the search space becomes much more manageable. Investigating the data further, we found that roughly two thirds of all failures were due to memory violations, which indicates that storing the compiled form is the bottleneck in the action-based encoding.

The next key insight is that d-DNNF (via c2d) is the target language most easily compiled. This is a direct result of the more compact form the target language yields: Figure 1 shows the size comparison for the fluent-based problems both c2d and cnf2bdd could compile. Generally, the d-DNNF is far more compact with the notable exception of the floortile domain. While efficient reasoning can be done using d-DNNF, common symbolic planning operations such as progression or regression remain difficult. BDD's, on the other hand, are the target language of choice for many symbolic planning approaches, and so it is of particular interest to focus on cnf2bdd. We observed the biggest jump in coverage between encodings for a compiler occurred with cnf2bdd, indicating that there is far more structure in the fluent-based

encoding that the BDD can capture succinctly.

While only preliminary, the results paint a broad picture of the capabilities existing compilers have for dealing with the characterization of all delete-relaxed dead-ends in a domain. There remains plenty of room for improvement for the key compilers such as cnf2bdd in the form of planning-specific variable ordering [11] and approximate compilation [21].

5 Summary

Dead-end detection for classical planning is central to many state-of-the-art planners, and also a key component for approaches that solve more expressive planning formalisms. In this work, we take the first steps towards characterizing the space of all delete-relaxed dead-ends. We achieve this using a pair of Boolean logic encodings that are elegant in their simplicity, and remarkably, are not susceptible to the same level of complexity that similar related planning encodings face. We also performed a preliminary evaluation to test the range of knowledge compilation techniques that can process the Boolean encodings, and demonstrated the strengths and weaknesses that they have on a suite of benchmarks known to have dead-ends. Our work lays the foundation for incorporating a compact representation of delete-relaxed dead-ends in a larger system, and moving forward we hope to (1) leverage our encodings to improve planner performance; and (2) incorporate notions such as the Π-compilation for stronger dead-end detection by combining fluents [10].

References

[1] Armin Biere, Marijn Heule, Hans van Maaren, and Toby Walsh. *Handbook of satisfiability, frontiers in artificial intelligence and applications.* IOS Press, 2009.

[2] Blai Bonet and Hector Geffner. Heuristics for planning with penalties and rewards using compiled knowledge. In *Proceedings, Tenth International Conference on Principles of Knowledge Representation and Reasoning, Lake District of the United Kingdom, June 2-5, 2006*, pages 452–462, 2006.

[3] Alberto Camacho, Christian Muise, and Sheila A. McIlraith. From fond to robust probabilistic planning: Computing compact policies that bypass avoidable deadends. In *The 26th International Conference on Automated Planning and Scheduling*, 2016.

[4] A. Darwiche. New advances in compiling CNF to decomposable negation normal form. In *Proceedings of European Conference on Artificial Intelligence*, pages 328–332, 2004.

[5] A. Darwiche and P. Marquis. A knowledge compilation map. *Journal of Artificial Intelligence Research*, 17:229–264, 2002.

[6] Malik Ghallab, Dana S. Nau, and Paolo Traverso. *Automated planning - theory and practice.* Elsevier, 2004.

[7] Jörg Hoffmann, Peter Kissmann, and Álvaro Torralba. "Distance"? Who Cares? Tailoring Merge-and-Shrink Heuristics to Detect Unsolvability. In *21st European Conference on Artificial Intelligence (ECAI 2014)*, pages 441–446, 2014.

[8] R. Huang, Y. Chen, and W. Zhang. SAS+ planning as satisfiability. *Journal of Artificial Intelligence Research (JAIR)*, 43:293–328, 2012.

[9] Henry A. Kautz and Bart Selman. Planning as satisfiability. In *European Conference on Artificial Intelligence*, pages 359–363, 1992.

[10] Emil Ragip Keyder, Jörg Hoffmann, and Patrik Haslum. Semi-relaxed plan heuristics. In *Proceedings of the Twenty-Second International Conference on Automated Planning and Scheduling, ICAPS 2012, Atibaia, São Paulo, Brazil, June 25-19, 2012*, 2012.

[11] Peter Kissmann and Jörg Hoffmann. Bdd ordering heuristics for classical planning. *Journal of Artificial Intelligence Research*, 51:779–804, 2014.

[12] Andrey Kolobov, Mausam, and Daniel S. Weld. A theory of goal-oriented mdps with dead ends. In *Proceedings of the Twenty-Eighth Conference on Uncertainty in Artificial Intelligence*, pages 438–447, 2012.

[13] Nir Lipovetzky, Christian Muise, and Hector Geffner. Traps, invariants, and dead-ends. In *The 26th International Conference on Automated Planning and Scheduling*, 2016.

[14] Christian Muise, J. Christopher Beck, and Sheila A. McIlraith. Optimal partial-order plan relaxation via maxsat. *Journal of Artificial Intelligence Research*, 2016.

[15] Christian Muise and Nir Lipovetzky. Unsolvability international planning competition. `http://unsolve-ipc.eng.unimelb.edu.au`, 2016. Accessed: 2018-03-19.

[16] Christian Muise, Sheila A. Mcilraith, J. Christopher Beck, and Eric Hsu. DSHARP: Fast d-DNNF Compilation with sharpSAT. In *Canadian Conference on Artificial Intelligence*, 2012.

[17] Christian J Muise, Sheila A McIlraith, and J Christopher Beck. Improved non-deterministic planning by exploiting state relevance. In *Proceedings of the Twenty-Second International Conference on Automated Planning and Scheduling (ICAPS 2012)*, 2012.

[18] Umut Oztok and Adnan Darwiche. A top-down compiler for sentential decision diagrams. In *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence*, pages 3141–3148, 2015.

[19] Jussi Rintanen. Planning as satisfiability: Heuristics. *Artificial Intelligence*, 193:45–86, 2012.

[20] Nathan Robinson, Charles Gretton, Duc Nghia Pham, and Abdul Sattar. Partial weighted maxsat for optimal planning. In *11th Pacific Rim International Conference on Artificial Intelligence*, pages 231–243, 2010.

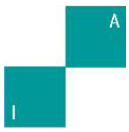
[21] Mathias Soeken, Daniel Gro, Arun Chandrasekharan, Rolf Drechsler, et al. Bdd minimization for approximate computing. In *2016 21st Asia and South Pacific Design Automation Conference (ASP-DAC)*, pages 474–479. IEEE, 2016.

[22] Fabio Somenzi. Cudd: Cu decision diagram package release. `https://github.com/ivmai/cudd`, 2015. Accessed: 2018-03-19.

[23] M. Thurley. sharpSAT — counting models with advanced component caching and implicit BCP. In *Ninth International Conference on Theory and Applications of Satisfiability*, pages 424–429, 2006.

[24] Takahisa Toda and Takehide Soh. Implementing efficient all solutions SAT solvers. *CoRR*, abs/1510.00523, 2015.

Inteligencia Artificial, 21(62) (2018), 75-90
doi: 10.4114/intartif.vol21iss62pp75-90



INTELIGENCIA ARTIFICIAL

http://journal.iberamia.org/

X and more Parallelism Integrating LTL-Next into SAT-based Planning with Trajectory Constraints While Allowing for Even More Parallelism

Gregor Behnke and Susanne Biundo

Institute of Artificial Intelligence, Ulm University, D-89069 Ulm, Germany
{gregor.behnke, susanne.biundo}@uni-ulm.de

Abstract Linear temporal logic (LTL) provides expressive means to specify temporally extended goals as well as preferences. Recent research has focussed on compilation techniques, i.e., methods to alter the domain ensuring that every solution adheres to the temporally extended goals. This requires either new actions or an construction that is exponential in the size of the formula. A translation into boolean satisfiability (SAT) on the other hand requires neither. So far only one such encoding exists, which is based on the parallel $\exists$-step encoding for classical planning. We show a connection between it and recently developed compilation techniques for LTL, which may be exploited in the future. The major drawback of the encoding is that it is limited to LTL without the X operator. We show how to integrate X and describe two new encodings, which allow for more parallelism than the original encoding. An empirical evaluation shows that the new encodings outperform the current state-of-the-art encoding.

Keywords: Temporally Extended Goals, Planning as SAT, Linear Temporal Logic.

1 Introduction

Linear temporal logic (LTL [18]) is a generic and expressive way to describe (state-)trajectory constraints. It is often used to specify temporal constraints and preferences in planning, e.g., to describe safety constraints, to state necessary intermediate goals, or to specify the ways in which a goal might be achieved. Most notably, the semantics of such constraints in PDDL 3.0 [11] is given in terms of LTL formulae, which is the de-facto standard for specifying planning problems.

Traditionally, LTL constraints are handled by first compiling them into an equivalent Büchi Automaton, and then translating the automaton into additional preconditions and effects for actions (see e.g. Edelkamp [8]). This compilation might be too expensive as the Büchi Automaton for a formula ϕ can have up to $2^{|\phi|}$ states. Recent work proposed another compilation using Alternating Automata [20]. These automata have only $\mathcal{O}(|\phi|)$ states allowing for a guaranteed linear compilation. There are also planners that do not compile the model, but evaluate the formula during forward search, e.g., TALplanner [7], TLplan [1], or the work by Hsu et al. [12]. However, heuristics have to be specifically tailored to incorporate the formula, or else the search becomes blind. TALplanner and TLplan even use the temporally extended goals for additional search guidance.

ISSN: 1137-3601 (print), 1988-3064 (on-line)

Another option is to integrate LTL into planning via propositional logic. Planning problems can be translated into (a sequence of) boolean formulae. A temporally extended goal can then be enforced by adding additional clauses to this formula. So far only one such encoding has been developed by Mattmüller and Rintanen [17]. It uses an LTL to SAT translation from the model checking community, which assumes that only a single state transition is executed at a time. The main focus of their work lies on integrating the efficient ∃-step encoding with this representation of LTL formulae. In the ∃-step encoding operators can be executed simultaneously, as long as they are all applicable in the current state, the resulting state is uniquely determined, and there is an ordering in which they are actually executable. Mattmüller and Rintanen presented alterations to the ∃-step formula restricting the parallelism such that LTL formulas without the next-operator are handled correctly.

We point out an interesting relationship between the LTL encoding of Mattmüller and Rintanen and the Alternating Automaton encoding by Torres and Baier, showing that both use the same encoding technique, although derived by different means. This insight might prove useful in the future, e.g., to allow for optimisation of the propositional encoding using automata concepts. Next, we show how the propositional encoding by Mattmüller and Rintanen can be extended to also be able to handle the next-operator X. We introduce a new concept – partial evaluation traces – to capture the semantics of the encoding with respect to an LTL formula and show that our extension is correct. Based on partial evaluation traces, we show that the restrictions posed by Mattmüller and Rintanen [17] on allowed parallelism can be relaxed while preserving correctness. We provide an alteration of their encoding allowing for more parallelism. We present an alternative encoding, also based on partial evaluation traces, which allows for even more parallelism by introducing intermediate timepoints at which the formula is evaluated. Our empirical evaluation of all encodings shows that our new encodings outperform the original one.

2 Preliminaries

2.1 Planning

We consider propositional planning without negative preconditions. This is known to be equivalent to STRIPS allowing for negative preconditions via compilation. Also note that all our techniques are also applicable in the presence of conditional effects. We do not consider them in this paper to keep the explanation of the techniques as simple as possible. For the extension to conditional effects, see Mattmüller and Rintanen [17].

Let A be a set of proposition symbols and $Lit(A) = \{a, \neg a \mid a \in A\}$ be the set of all literals over A. An action a is a tuple $a = \langle p, e\rangle$, where p – the preconditions – is a subset of A and e – the effects – is a subset of $Lit(A)$. We further assume that the effects are not self-contradictory, i.e., that for no $a \in A$ both a and $\neg a$ are in e. A state s is any subset of A. An action $a = \langle p, e\rangle$ is executable in s, iff $p \subseteq s$. Executing a in s results in the state $(s \setminus \{a \mid \neg a \in e\}) \cup \{a \mid a \in e\}$. A planning problem $\mathcal{P} = \langle A, O, s_I, g\rangle$ consists of a set of proposition symbols A, a set of operators O, the initial state s_I, and the goal $g \subseteq A$. A sequence of actions $o_1, \ldots, o_n$ is a plan for $\mathcal{P}$ iff there exists a sequence of states $s_0, \ldots, s_{n+1}$ such that for every $i \in \{1, \ldots, n+1\}$, o_i is applicable in s_i, its application results in s_{i+1}, $s_0 = s_I$, and $g \subseteq s_{n+1}$. This sequence of states is called an execution trace.

2.2 Linear Temporal Logic

Formulae in Linear Temporal Logic (LTL) are constructed over a set of primitive propositions. In the case of planning these are the proposition symbols A. LTL formulae are recursively defined as any of the following constructs, where p is a proposition symbol and f and g are LTL formulae.

$$\bot \mid \top \mid p \mid \neg f \mid f \wedge g \mid f \vee g \mid Xf \mid \mathring{X}f \mid Ef \mid Gf \mid fUg$$

X, $\mathring{X}$, E, G, and U are called temporal operators. There are several further LTL-operators like $\mathring{U}$, R, or S and T [4]. Each of them can be translated into a formula containing only the temporal operators X, $\mathring{X}$, and U. The semantics of an LTL formula ϕ is given with respect to an execution trace. In general this trace can be infinitely long, as LTL can describe repeated behaviour. We consider only LTL over

finite traces, which is commonly called LTL_f [6]. The encodings we present can easily be extended to the infinite case (see Mattmüller and Rintanen [17]). The truth value of an LTL_f formula ϕ is defined over an execution trace $\sigma = (s_0, s_1, \ldots, s_n)$ as $[[\phi]](\sigma)$ where

$$\begin{aligned}
[[p]](s_0, \sigma) &= p \in s_0 \qquad \text{if } p \in A \\
[[\neg f]](\sigma) &= \neg[[f]](\sigma) \\
[[f \wedge g]](\sigma) &= [[f]](\sigma) \wedge [[g]](\sigma) \\
[[f \vee g]](\sigma) &= [[f]](\sigma) \vee [[g]](\sigma) \\
[[Xf]](s_0, \sigma) &= [[\mathring{X}f]](s_0, \sigma) = [[f]](\sigma) \\
[[Xf]](s_0) &= \bot \\
[[\mathring{X}f]](s_0) &= \top \\
[[Ef]](s_0, \sigma) &= [[f]](s_0, \sigma) \vee [[Ef]](\sigma) \\
[[Gf]](s_0, \sigma) &= [[f]](s_0, \sigma) \wedge [[Gf]](\sigma) \\
[[Ef]](s_0) &= [[Gf]](s_0) = [[f]](s_0) \\
[[fUg]](s_0, \sigma) &= [[g]](s_0, \sigma) \vee \\
&\quad ([[f]](s_0, \sigma) \wedge [[fUG]](\sigma)) \\
[[fUg]](s_0) &= [[g]](s_0)
\end{aligned}$$

The intuition of the semantics of temporal operators is: Ef – eventually f, i.e., f will hold at some time, now or in the future, Gf – globally f, i.e., f will hold from now on for ever, fUg – f until g, i.e., g will eventually hold and until that time f will always hold, and Xf – next f, i.e., f holds in the next state of the trace. Since we consider the case of finite LTL, we have – in addition to standard LTL – a new operator: weak next $\mathring{X}$. The formula Xf requires that there is a next state and that f holds in that state. In contrast, $\mathring{X}f$ asserts that f holds if a next state exists; if there is none, $\mathring{X}f$ is always true, taking care of the possible end of the state sequence.

As a preprocessing step, we always transform an LTL formula ϕ into negation normal form without increasing its size, i.e., into a formula where all negations only occur directly before atomic propositions. This can be done using equivalences like $\neg Gf = E\neg f$. Next, we add for each proposition symbol $a \in A$ a new proposition symbol $\overline{a}$. Its truth value will be maintained such that it is always the inverse of a. I.e. whenever an action has $\neg a$ as its effect, we add the effect $\overline{a}$ and when it has the effect a we add $\neg\overline{a}$. Lastly, we replace $\neg a$ in ϕ with $\overline{a}$, resulting in a formula not containing negation.

Given a planning problem $\mathcal{P}$ and an LTL formula ϕ, LTL planning is the task of finding a plan π whose execution trace σ will satisfy ϕ, i.e., for which $[[\phi]](\sigma)$. For a given LTL formula ϕ we define $A(\phi)$ as the set of predicates contained in ϕ and $\mathcal{S}(\phi)$ to be the set of all its subformulae. We write $[o]_e^\phi$ for the intersections of the effects of o and $A(\phi)$, i.e. all those effects that occur in ϕ.

3 State-of-the-art LTL→SAT encoding

As far as we are aware, there is only a single encoding of LTL planning problems into boolean satisfiability, developed by Mattmüller and Rintanen [17]. They adapted a propositional encoding for LTL developed by Latvala et al. [14] for bounded model checking. The main focus of Mattmüller and Rintanen's work lies on integrating modern, non-sequential encodings of planning problems into the formula. The encoding models evaluating the LTL formula in timesteps, which correspond to the states in a trace. In Latvala et al.'s encoding (which was not developed for planning, but for a more general automata setting) only a single action may be executed at each timestep in order to evaluate the formula correctly. Research in translating planning problems into propositional formulae has however shown that such sequential encodings perform significantly worse than those that allow for a controlled amount of parallel action execution [19]. Mattmüller and Rintanen addressed the question of how to use the LTL encoding by Latvala et al. in a situation where multiple state transitions take care in parallel – as is the case in these planning encodings. They used the property of stutter-equivalence which holds for LTL_{-X} (i.e. LTL without the X and $\mathring{X}$ operators) to integrate Latvala et al.'s encoding. To exploit stutter-equivalence, they had to restrict the allowed amount of parallelism to ensure correctness.

Their encoding, which we will denote with M&R'07, is based on the ∃-step encoding of propositional planning by Rintanen et al. [19]. As such, we start by reviewing the ∃-step encoding in detail. In this encoding the plan is divided into a sequence of timesteps $0, \ldots, n$. Each timestep t is assigned a resulting state using decision variables a^t for all $a \in A$ and $t \in \{1, \ldots, n+1\}$, each indicating that the proposition symbol a holds at timestep t, i.e. after executing the actions at timestep $t-1$. The initial state is represented by the variables a^0. Actions can be executed between two neighbouring timesteps t and $t+1$, which is represented by decision variables o^t for all $o \in O$ and $t \in \{0, \ldots, n\}$. If o^t is true the action o is executed at time t. The encoding by Kautz and Selman [13] is then used to determine which actions are executable in a^t and how the state a^{t+1} resulting from their application looks like. In a sequential encoding, one asserts for each timestep t that at most one o^t atom is true. Intuitively, this is necessary to ensure that the state a^{t+1} resulting from executing the actions o^t is uniquely determined. Consider, e.g., a situation where two actions `move-a-to-b` and `move-a-to-c` are simultaneously applicable, but result in conflicting effects. Executing these two actions in parallel has no well-defined result. Interestingly, the mentioned constraint is not necessary in this case, as the encoding by Kautz and Selman already leads to an unsatisfiable formula. There are however situations, where the resulting state is well-defined, but it is not possible to execute the actions in any order. Consider two actions `buy-a` and `buy-b`, both requiring money, spending it, and achieving possession of `a` and `b`, respectively. Both actions are applicable in the same state and their parallel effects are well-defined, as they don't conflict. It is not possible to find a sequential ordering of these two actions that is executable, as both need money, which won't be present before executing the second action. This situation must be prohibited, which can easily be achieved by forbidding parallel action execution at all, as in the sequential encoding.

In the ∃-step encoding, executing actions in parallel is allowed. Ideally, we would like to allow any subset S of o^t to be executable in parallel, as long as there exists a linearisation of S that is executable in the state s^t represented by a^t and all executable linearisations lead to the same state s^{t+1}. This property is however practically too difficult to encode [19]. Instead, the ∃-step encoding uses a relaxed requirement. Namely, (1) all actions in S must be executable in s^t, then it chooses a total order of all actions O, (2) asserts that if a set of actions S is executable, it must be executable in that order, and (3) that the state reached after executing them in this order is s^{t+1}. The encoding by Kautz and Selman ensures the first and last property. The ∃-step encoding has to ensure the second property. It however does not permit all subsets $S \subseteq O$ to be executable in parallel, but only those for which this property can be guaranteed.

As a first step, we have to find an appropriate order of actions in which as many subsets $S \subseteq O$ as possible can be executed. For this, the Disabling Graph (DG) is used. It determines which actions can be safely executed in which order without checking the truth of propositions inside the formula. In practice, one uses a relaxed version of the DG (i.e. one containing more edges), as it is easier to compute [19].

Definition 1. *Let $\mathcal{P} = \langle A, O, s_I, g\rangle$ be a planning problem. An action $o_1 = \langle p_1, e_1\rangle$* affects *another action $o_2 = \langle p_2, e_2\rangle$ iff $\exists l \in A$ s.t. $\neg l \in e_1$ and $l \in p_2$.*

A Disabling Graph DG($\mathcal{P}$) of a planning problem $\mathcal{P}$ is a directed graph $\langle O, E\rangle$ with $E \subseteq O \times O$ that contains all edges (o_1, o_2) where o_1 affects o_2 and a state s exists that is reachable from s_I in which both o_1 and o_2 are applicable.

The DG is tied to the domain itself and is not tied to any specific timestep, as such the restrictions it poses apply to every timestep equally. The DG encodes which actions disable the execution of other actions after them in the same timestep, i.e., we ideally want the actions to be ordered in the opposite way in the total ordering chosen by the ∃-step encoding. If the DG is acyclic, we can execute all actions in the inverted order of the disabling graph, as none will disable an action occurring later in that order. If so, the propositional encoding does not need any further clauses, as any subset S of actions can be executed at a timestep – provided that their effects do not interfere.

The DG is in practice almost never acyclic. Problematic are only strongly connected components (SCCs) of the DG, where we cannot find an ordering s.t. we can guarantee executability for all subsets of actions. Instead we fix some order $\prec$ for each SCC, asserting that the actions in it will always be executed in that order, and ensure in the SAT formula that if two actions o_1 and o_2 with $o_1 \prec o_2$ are executed, o_1 does not affect o_2. This way, we can safely ignore *some* of the edges of the DG – as their induced constraints are satisfied by the fixed ordered $\prec$ – while others have to be ensured in the formula. I.e. if we ensure for every edge (o_1, o_2) with $o_1 \prec o_2$ that if o_1 is part of the executed subset o_2 is not,

we know that there is a linearisation in which the chosen subset S is actually executable. To ensure this property, Rintanen et al. introduced *chains*. A chain $chain(\prec; E; R; l)$ enforces that whenever an action o in E is executed, all actions in R that occur after a in $\prec$ cannot be executed. Intuitively, E are those actions that produce some **e**ffect a, while the actions in R **r**ely on $\neg a$ to be true. The last argument l is a label that prohibits interference between multiple chains.

$$
\begin{aligned}
&chain(o_1, \ldots, o_n; E; R; l) = \\
&\quad \bigwedge \{o_i \rightarrow \mathtt{d}^{j,l} \mid i < j, o_i \in E, o_j \in R, \{o_{i+1}, .., o_{j-1}\} \cap R = \emptyset\} \\
&\quad \cup \{l^i \rightarrow a^{j,l} \mid i < j, \{o^i, o^j\} \subseteq R, \{o_{i+1}, .., o_{j-1}\} \cap R = \emptyset\} \\
&\quad \cup \{l^i \rightarrow \neg o_i \mid o^i \in R\}
\end{aligned}
$$

To ensure that for any SCC S of $DG(\mathcal{P})$ the mentioned condition holds for the chosen ordering $\prec$ of S, we generate for every proposition symbol $a \in A$ a chain with

$$
\begin{aligned}
E_a &= \{o \in S \mid o = \langle p, e \rangle \text{ and } \neg a \in e\} \\
R_a &= \{o \in S \mid o = \langle p, e \rangle \text{ and } a \in p\}
\end{aligned}
$$

Based on the $\exists$-step encoding, Mattmüller and Rintanen [17] added support for LTL formulae ϕ by exploiting the stutter-equivalence of LTL_{-X}. This stutter-equivalence ensures that if multiple actions are executed in a row but don nott change the truth of any of the predicates in $A(\phi)$, the truth of the formula is not affected, i.e., the truth of the formula does not depend on how many of these actions are executed in a row. Consequently the formula only needs to be checked whenever the truth of propositions in $A(\phi)$ changes. Their construction consists of two parts. First, they add clauses to the formula expressing that the LTL formula ϕ is actually satisfied by the trace restricted to the states where propositions in $A(\phi)$ change. These states are the ones represented in the $\exists$-step encoding by a^t atoms. Second, they add constraints to the $\exists$-step parallelism s.t. in every timestep the first action executed according to $\prec$ that changes proposition symbols in $A(\phi)$ is the only one to do so. Other actions in that timestep may not alter the state with respect to $A(\phi)$ achieved by that first action, but can assert the same effect.

In their paper, they provide a direct translation of ϕ into a proposition formula. In practice however, this formula cannot be given to a SAT solver, as it requires a formula in conjunctive normal form (CNF). The formula given in the paper is not in CNF and translating it into CNF can lead to a CNF of exponential size. They instead introduce additional variables [16], allowing them to generate (almost) a CNF[1]. For every sub-formula $\psi \in \mathcal{S}(\phi)$ and every timestep t they introduce the variable ψ^t_{LTL}, stating that ψ holds for the trace starting at timestep t. They then assert: (1) that ϕ^0_{LTL} holds and (2) that for every ψ^t_{LTL} the consequences must hold that make ψ true for the trace starting at time t. The latter is expressed by clauses $\psi^t_{LTL} \rightarrow [[\psi]]^t$, where $[[\psi]]^t$ is given in Tab. 1. Note that the M&R'07 encoding cannot handle the next operators X and $\mathring{X}$, as they are sensitive to stuttering, i.e., stutter-equivalence does not hold for formulae that contain X or $\mathring{X}$. We have added the encoding for X [14]. In addition, we have restricted the original encoding from infinite to finite LTL-traces and added a new encoding for $\mathring{X}$. Note that the M&R'07 encoding will lead to wrong results if used with the presented encoding of the X and $\mathring{X}$ operators. It is however correct, if used in conjunction with a sequential encoding [14]. We show in Sec. 5 how the M&R'07 encoding can be changed to handle X and $\mathring{X}$ correctly. Lastly, clauses need to be added in order to ensure that actions executed in parallel do not alter the truth of propositions in $A(\phi)$ – except for the first action that actually changes them. The extension of the $\exists$-step encoding achieving this is conceptually simple, as it consists of only two changes to the original encoding:

1. Add for every two actions o_1, o_2 which are simultaneously applicable the edge (o_1, o_2) to the DG iff $[o_2]^\phi_e \setminus [o_1]^\phi_e \neq \emptyset$, i.e., o_2 would change more than o_1 with respect to $A(\phi)$.

2. Add for every literal $l \in Lit(A(\phi))$ and SCC of $DG(\mathcal{P})$ with its total order $\prec$ the chain $chain(\prec; E^\phi_l; R^\phi_l; \phi_l)$ with

[1]The translations of $f \wedge g$, Gf, and fUg contain one conjunction each, which can transformed easily.

ϕ	$t < n$	$t = n$
$[[p]]^t \quad p \in A$	p^t	p^t
$[[f \wedge g]]^t$	$f^t_{LTL} \wedge g^t_{LTL}$	$f^t_{LTL} \wedge g^t_{LTL}$
$[[f \vee g]]^t$	$f^t_{LTL} \vee g^t_{LTL}$	$f^t_{LTL} \vee g^t_{LTL}$
$[[Xf]]^t$	f^{t+1}_{LTL}	$\bot$
$[[\mathring{X}f]]^t$	f^{t+1}_{LTL}	$\top$
$[[Ef]]^t$	$f^t_{LTL} \vee (Ef)^{t+1}_{LTL}$	f^t_{LTL}
$[[Gf]]^t$	$f^t_{LTL} \wedge (Gf)^{t+1}_{LTL}$	f^t_{LTL}
$[[fUg]]^t$	$g^t_{LTL} \vee ((fUg)^{t+1}_{LTL} \wedge f^t_{LTL})$	f^t_{LTL}

Table 1: Transition rules for LTL formulae

(a) $E^\phi_l = \{o \in O \mid o = \langle p, e\rangle \text{ and } l \notin e\}$

(b) $R^\phi_l = \{o \in O \mid o = \langle p, e\rangle \text{ and } l \in e\}$;

Mattmüller and Rintanen [17] have proven that these clauses suffice to ensure a correct and complete encoding.

4 Alternating Automata and M&R'07

In recent years, research on LTL planning focussed on translation based approaches. There, the original planning problem is altered in such a way that all solutions for the new problem adhere to the formula (e.g. Baier and McIlraith [2] and Torres and Baier [20], see Camacho et al. [5] for an overview). Traditionally, these approaches translate the LTL formula into a Büchi automaton (essentially a finite state machine, with a specific accepting criterion for infinite traces) and then integrate the automaton into the model. The major drawback of these translations is that the Büchi automaton for an LTL formula can have up to $2^{|\phi|}$ many states.

Torres and Baier [20] proposed a translation diverging from the classical LTL to Büchi translation. They instead based it on Alternating Automata, which are commonly used as an intermediate step when constructing the Büchi automaton for an LTL ϕ formula (see e.g. Gastin and Oddoux [10]). Alternating Automata have a guaranteed linear size in $|\phi|$, but have a more complex transition function.

Definition 2. *Given a set of primitive propositions A, an alternating automaton is a 4-tuple $\mathcal{A} = (Q, \delta, I, F)$ where*

- *Q is a finite set of states*
- *$\delta : Q \times 2^A \to \mathcal{B}^+(Q)$ is a transition function, where $\mathcal{B}^+(Q)$ is the set of positive propositional formulas over the set of states Q, i.e., those formulae containing only $\vee$ and $\wedge$.*
- *$I \subseteq Q$ is the initial state*
- *$F \subseteq Q$ is the set of final states.*

A run of an alternating automaton over a sequence of sets of propositions (execution trace) $(s_1, \ldots, s_n)$ is a sequence of sets of states $(Q_0, \ldots, Q_n)$ such that

- $Q_0 = I$
- $\forall i \in \{1, \ldots, n\} : Q_i \models \bigwedge_{q \in Q_{i-1}} \delta(q, s_i)$

The alternating automaton accepts the trace iff $Q_n \subseteq F$

Torres and Baier [20] generate an alternating automaton for an LTL formula ϕ as follows. They choose Q as the set of sub-expressions of ϕ starting with a temporal operator plus a state q_F representing that the end of the execution trace has been reached. Being in a state q means, that from the current

time on we have to fulfill the formula q. The automaton is given as $A_\phi = (Q, \delta, \{q_\phi\}, \{q_F\})$ where $Q = \{q_\alpha \mid \alpha \in \mathcal{S}(\phi)\} \cup \{q_F\}$ and the transition function δ is defined as follows:

$$\begin{aligned}
\delta(q_l, s) &= \begin{cases} \top & \text{if } l \in s \\ \bot & \text{if } l \notin s \end{cases} \\
\delta(q_F, s) &= \bot \\
\delta(q_{f \vee g}, s) &= \delta(q_f, s) \vee \delta(q_g, s) \\
\delta(q_{f \wedge g}, s) &= \delta(q_f, s) \wedge \delta(q_g, s) \\
\delta(q_{X f}, s) &= q_f \\
\delta(q_{\mathring{X} f}, s) &= q_F \vee q_f \\
\delta(q_{E f}, s) &= \delta(f, s) \vee q_{E f} \\
\delta(q_{G f}, s) &= \delta(f, s) \wedge (q_{G f} \vee q_F) \\
\delta(q_{f U g}, s) &= \delta(q_g, s) \vee (\delta(q_f, s) \wedge q_{f U g})
\end{aligned}$$

Note that we have to enumerate all states that are relevant to the formula, i.e., all states $s \subseteq 2^{A(\phi)}$, to construct the formula. Using the Alternating Automaton as the basis for a translation leads to a guaranteed linear increase in size when constructing the translated planning problem. This is due to the fact that the encoding does not actually has to construct the automaton, but only has to simulate its states. We will elaborate on this later. Also, it was demonstrated that the new encoding is more efficient than other current translation techniques [20].

A drawback of their translation was the need for introducing additional actions, performing bookkeeping on the current state of the alternating automaton. A translation into SAT, on the other hand, will not have this drawback, as we will show. We will again extend the ∃-step encoding and call the encoding AA (**A**lternating **A**utomaton). The restriction posed on parallelism by M&R'07 does not depend on the encoding of the formula itself as long as it does not contain the X or $\mathring{X}$ operators[2]. We introduce new decision variables q^t for each state $q \in Q$ and timestep t, signifying that the automaton A_ϕ is in state q after executing the actions of timestep t. To express the transition function of the Alternating Automaton, we use formulae of the form $\left(q^t \wedge \bigwedge_{a \in s} a\right) \rightarrow \delta(q, s)$ for each state q of the automaton and set of propositions s. We also replace each occurrence of a state q_f in $\delta(q, s)$ with the decision variable q_f^{t+1} and introduce intermediate decision variables to break down complex formulae as in the M&R'07 encoding. The following theorem holds by construction.

Theorem 1. *AA in conjunction with M&R'07's ∃-step encoding is correct for LTL_{-X}.*

We here want to point out that the AA encoding is not (as the one by Torres and Baier [20]) polynomial in the size of the formula. The reason lies in the explicit construction of the alternating automaton, which requires a single transition for every possible state that might be relevant to the formula, i.e., for every subset of $A(\phi)$. The translation encoding by Torres and Baier circumvents this construction by adding new operators, which can evaluate the necessary expression during planning. I.e. they have actions for each transition rule $\delta(\cdot, \cdot)$, which produce the right-hand sides of these above equations as their effects. Lastly, they introduce synchronisation actions to ensure that $\delta(\cdot, \cdot)$ is fully computed before another "real" action is executed.

If we apply this idea to the AA encoding, we would end up with the M&R'07 encoding. Since the states of the automaton are the sub-formulae of ϕ starting with a temporal operator, these decision variables are identical to the ψ^t_{LTL} variables of the M&R'07 encoding, where ψ starts with a temporal operator. The encoding by Torres and Baier also needs to introduce state variables for every sub formulae not starting with a temporal operator to represent the step-wise computation of $\delta(\cdot, \cdot)$ correctly. If translated into propositional variables, these correspond to the ψ^t_{LTL} variables of M&R'07, where ψ does not start with a temporal operator. Lastly, the transition rules for both encodings are identical.

As such, the M&R'07 encoding can also be interpreted as a direct translation of an Alternating Automaton into propositional logic using the compression technique of Torres and Baier [20]. Interestingly,

[2]The actual encoding of the formula can be exchanged in the proof of their main theorem as long as a similar version of their Theorem 2 can be proven, which is obvious in our case.

the original proof showing correctness of the LTL-part of the M&R'07 encoding by Latvala et al. [14] does not rely on this relationship to Alternating Automata, neither do they mention this connection. We think it is an interesting theoretical insight, as it might enable to further improve LTL encodings, e.g., based on optimisations of the Alternating Automaton.

5 X, Parallelism, and Partial Evaluation

We have noted that both M&R'07 and AA cannot handle LTL formulae containing the X or $\mathring{X}$ operators in conjunction with the $\exists$-step encoding. They are however correct if used together with the sequential encoding, where only a single action can be executed at each timestep. In order to derive extensions that can handle X and $\mathring{X}$, we first present a new theoretical foundation for both encodings. We will use the fact that in M&R'07, we know which parts of the formula are made true at which time and by which propositions. To formalise this, we introduce evaluation traces, which specify how an LTL formula is fulfilled over a trace.

Definition 3. *Let ϕ be an LTL formula. We call a sequence $\psi = (f_0, \ldots, f_n)$ with $f_i \subseteq \mathcal{S}(\phi)$ an evaluation trace for ϕ iff $\phi \in f_0$ and for all $i \in \{0, \ldots, n\}$ holds*

1. *if $f \vee g \in f_i$ then $f \in f_i$ or $g \in f_i$*
2. *if $f \wedge g \in f_i$ then $f \in f_i$ and $g \in f_i$*
3. *if $Xf \in f_i$ then $i < n$ and $f \in f_{i+1}$*
4. *if $\mathring{X}f \in f_i$ then $i = n$ or $f \in f_{i+1}$*
5. *if $Ef \in f_i$ then $f \in f_i$ or $i < n$ and $Ef \in f_{i+1}$*
6. *if $Gf \in f_i$ then $f \in f_i$ and if $i < n$ then $Gf \in f_{i+1}$*
7. *if $fUg \in f_i$ then $g \in f_i$ or $i < n$ and $f \in f_i$ and $fUg \in f_{i+1}$*

A trace $\pi = (s_0, \ldots, s_n)$ satisfies an evaluation trace $\psi = (f_0, \ldots, f_n)$ iff for all $a \in f_i \cap A$ also $a \in s_i$ holds.

The following theorem follows directly, as the definition just emulates checking an LTL formula.

Theorem 2. *An execution trace π satisfies an LTL formula ϕ iff an evaluation trace ψ for ϕ exists that satisfies π.*

In M&R'07, the LTL formula is only evaluated after a set of parallel actions have been executed. To capture this, we define partial evaluation traces.

Definition 4. *Let $\pi = (s_0, \ldots, s_n)$ be an execution trace and ϕ and LTL formula. We call an evaluation trace $\theta = (f_0, \ldots, f_l)$ with $l \leq n$ a partial evaluation trace (PET) for π if a sequence of indices $0 = i_0 < i_1 < ... < i_l = n + 1$ exists such that for each $k \in \{0, \ldots, l-1\}$ holds*

$$s_{i_k} \cap (f_k \cap A) = \cdots = s_{i_{k+1}-1} \cap (f_k \cap A)$$

and if $Xf \in f_k$ or $\mathring{X}f \in f_k$ and $k > 0$ then $i_{k-1} + 1 = i_k$. The PET θ is satisfied by the execution trace π, iff the execution trace $(s_{i_1-1} \cap f_0 \cap A, \ldots, s_{i_l-1} \cap f_l \cap A)$ satisfies θ in the sense of Def. 3.

A satisfying valuation of the M&R'07 encoding corresponds to a partial evaluation trace that satisfies the formula ϕ. M&R'07 also asserts that PET is satisfied by the execution trace corresponding to the sequential plan generated by the $\exists$-step formula. The main property of Def. 4, which is necessary for showing that we actually generate a PET, is ensured by the chains added to the original $\exists$-step encoding and the additional edges in the Disabling Graph. The following theorem states that every partial evaluation trace that is satisfied by an executed trace can be extended to a full evaluation trace and thus forms a witness that the execution trace satisfies the LTL formula. This gives us a second, independent proof of correctness for the M&R'07 encoding.

Theorem 3. *Let π be a trace and θ be an PET for the formula ϕ on π. If π satisfies θ, then π satisfies ϕ.*

Proof. We need to show that the PET $\psi = (f_0, \ldots, f_l)$ can be extended to a full evaluation trace on $\pi = (s_0, \ldots, s_n)$, s.t. π satisfies that evaluation trace. If so, we can apply Thm. 2 and conclude that π also satisfies ϕ. Let $i_0, \ldots, i_l$ be the indices of Def. 4 for which θ is a PET. We claim that

$$\theta^* = (\overbrace{f_0, \ldots f_0}^{i_1 - i_0 \text{ times}}, \overbrace{f_1, \ldots, f_1}^{i_2 - i_1 \text{ times}}, \ldots, \overbrace{f_l, \ldots, f_l}^{i_l - i_{l-1} \text{ times}})$$

is an evaluation trace that satisfies π, and that π satisfies ϕ. First we show that θ^* is an evaluation trace. We start by proving the enumerated properties of Def. 3. Consider the ith element f^* of θ^* and let f^{**} be the $i+1$th element (if such exists).

1. trivially satisfied
2. trivially satisfied
3. $Xf \in f^*$. We know that f^* is the only repetition some f_j in the trace, as θ is a PET. Also $f^{**} = f_{j+1}$. Consequently $f \in f^{**}$. In case f^{**} does not exist, θ cannot be an PET.
4. $\mathring{X}f \in f^*$. We know that f^* is the only repetition of some f_j in the trace, as θ is a PET. In case f^{**} does not exist, we have nothing to show. Else, $f^{**} = f_{j+1}$ and $f \in f^{**}$.

For the last three requirements relating to the temporal operators E, G, and U, we can distinguish three cases depending on where f^* is situated in the sequence θ^*

- f^* is the last element of θ^*:
  5. if $Ef \in f^*$ then $f \in f^*$, as θ is a PET.
  6. if $Gf \in f^*$ then $f \in f^*$, as θ is a PET.
  7. if $fUg \in f^*$ then $g \in f^*$, as θ is a PET.
- $f^* \neq f^{**}$, i.e., the last repetition of f^*. We know that $f^* = f_j$ and $f^{**} = f_{j+1}$ for some $j \in \{0, \ldots, l-1\}$.
  5. if $Ef \in f^*$ then either $f \in f_j = f^*$ or $Ef \in f_{j+1} = f^{**}$
  6. if $Gf \in f^*$ then $f \in f_j = f^*$ and $Gf \in f_{j+1} = f^{**}$
  7. if $fUG \in f^*$ then either $g \in f_{j+1} = f^{**}$ or $f \in f_j = f^*$ and $fUg \in f_{j+1} = f^{**}$
- if $f^* = f^{**}$
  5. if $Ef \in f^*$ then $Ef \in f^{**}$
  6. if $Gf \in f^*$ then $Gf \in f^{**}$ and $f \in f^*$
  7. if $fUg \in f^*$ then either $g \in f^*$, or $f \in f^*$, but then also $fUG \in f^{**}$, since $f^* = f^{**}$

$\phi \in f_0$ holds as θ is a PET, which concludes conclude the proof that θ^* is an evaluation trace.

Lastly, we have to show that θ^* satisfies π, i.e., we have to show, for each timestep $j \in \{0, \ldots n\}$ and every $a \in A$ which is true in the ith element of θ^*, that $a \in s_j$ holds. Consider first the indices of the last repetitions of each f_k, i.e., the states $s_{i_k - 1}$. Since θ is a PET, it satisfies $(s_{i_1 - 1} \cap f_0 \cap A, \ldots, s_{i_l - 1} \cap f_l \cap A)$, so it satisfies the required property for all time-steps $i_k - 1$. Consider any other timestep t and its next timestep in the PET $i_k - 1$ (which always exists, since the last index is equal to n). Since θ is a PET, we know that $s_t \cap (f_k - 1 \cap A) = s_{i_k - 1} \cap (f_k \cap A)$. We have chosen to set the tth element of θ^* to f_k. Since $s_{i_k - 1} \cap (f_k \cap A)$ satisfies the required property for f_k, so must $s_t \cap (f_k \cap A)$ and thus s_t itself (it can have only more true predicates). □

We can now use this result to integrate support for X and $\mathring{X}$ into the M&R'07 encoding. For that, we have to assert that the second last condition of Def. 4 holds, as all other requirements are already checked by the M&R'07 encoding. We first add four new variables per time step t.

- $exactOne^t$ – exactly one action is executed at time t
- $atLeastOne^t$ – at least one action is executed at time t
- $atMostOne^t$ – at most one action is executed at time t
- $none^t$ – no action is executed at any time $\geq t$

To enforce the semantics of these variables, we add the following clauses per timestep:

$$\begin{aligned} \forall o \in O : none^t &\rightarrow \neg o^t \\ none^t &\rightarrow none^{t+1} \\ atLeastOne^t &\rightarrow \bigvee_{o \in O} o^t \\ exactOne^t &\rightarrow atLeastOne^t \wedge atMostOne^t \end{aligned}$$

Encoding the $atMostOne^t$ atom is a bit more complicated. A native encoding requires $\mathcal{O}(|O|^2)$ clauses. There are however better encodings for the at most one constraint in SAT. We have chosen the log-counter encoding, which introduces $\log(|S|)$ new variables while only requiring $|S|\log(|S|)$ clauses [9]. To ensure the semantics of the atom $atMostOne^t$, we add it as a guard to the log-counter encoding, i.e., we add $\neg atMostOne^t$ to every clause. If $atMostOne^t$ is required to be true, the log-counter clauses have to be satisfied, i.e., at most one action atom can be true. If $atMostOne^t$ can be false, we can simply set it to $\bot$ thereby satisfying all log-counter clauses. We lastly set $exactOne^t$ for $t = -1$ to $\top$ and $atLeastOne^{n+1}$ to false. Based on these new variables, we can add the constraints necessary for evaluating X and $\mathring{X}$ correctly under the $\exists$-step encoding. For each timestep t, we add

$$\begin{aligned} (Xf)^t_{LTL} &\rightarrow exactOne^{t-1} \wedge f^{t+1} \wedge atLeastOne^t \\ (\mathring{X}f)^t_{LTL} &\rightarrow exactOne^{t-1} \\ (\mathring{X}f)^t_{LTL} &\rightarrow (f^{t+1} \wedge atLeastOne^t) \vee none^t \end{aligned}$$

A valuation of this encoding represents a partial evaluation trace satisfied by the execution trace of its actions. By applying Thm. 3, we know that a satisfying evaluation trace for the plan exists. Showing completeness for the encoding is trivial, since a sequential assignment satisfies the formula for every plan. We will also call this extended encoding M&R'07 in our evaluation, as it is exactly identical to this one, provided no X or $\mathring{X}$ operator is present.

So far, we have not used the – in our view – most significant improvement: Relaxing the restrictions on parallelism to only those actually needed by the current formula. Doing so based on our theorem is surprisingly easy. Consider a timestep of the M&R'07 encoding, in which actions can be executed in parallel, which are given an implicit order $\prec$ by the $\exists$-step encoding. For each literal $l \in Lit(A(\phi))$ a chain ensures that the action changing it is applied first, i.e., that the "first" action of a timestep performs all changes relevant to the whole formula and can thus check the formula only in the resulting state. By applying Def. 4 and Thm. 3, we have to ensure this property only for those proposition symbols that need to be evaluated after the actions have been executed, i.e., for all a^{t+1}_{LTL} that are true. We can do this simply by adding the literal $\neg a^t_{LTL}$ as a guard (simiar to $atMostOne^t$) to every clause in the chains $chain(\prec; E^\phi_l; R^\phi_l; \phi_a)$ and $chain(\prec; E^\phi_l; R^\phi_l; \phi_{\neg a})$. If we have to make the literal a^t_{LTL} true, the chains become active, if not they are inactive (as they are trivially satisfied by $\neg a^t_{LTL}$). We will denote this improved encoding with Improved-M&R'07.

To illustrate the effects of the improved M&R'07 encoding, consider the following planning problem. There are six proposition symbols a, b, c, d, e and f, of which a and b are true in the initial state and e has to hold in the goal state. There are five actions described in Tab. 2. We consider this domain in conjunction with the formula $\phi = G((a \wedge d) \rightarrow Ef) = G(\neg a \vee \neg d \vee Ef)$. The disabling graph, extended by the edges needed for the M&R'07 encoding, is depicted in Fig. 1. This planning problem has only a single solution, namely: Y, X, V, W, Z. Under the M&R'07 encoding, we need four timesteps to find a plan, i.e., {Y}, {X}, {V}, {W,Z}. This is due to the fact that most actions' effects contain one of the three predicates contained in ϕ, but not the others. Using the improved M&R'07 encoding, we only need three timesteps, as now Y and X can be executed together in the first timestep. The reason for this beeing possible is quite unintuitive, but it shows the strength of our approach. In the M&R'07 encoding,

	X	Y	Z	V	W
pre	a,b	a,b	c,d	c,d	g
add	c	d	e	g	f
del	a		c		

Table 2: Actions in the example domain

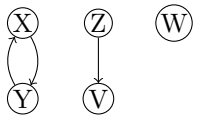


Figure 1: Extended disabling graph for the example domain.

the encoding correctly detects that there is a state in which a and d are true simultaneously[3] after the action Y has been executed and that thus Ef has to hold after executing Y. In the plan for the improved encoding, the solver can simply choose to achieve Ef after {Y,X} has been executed. This way the solver is never forced to achieve $\neg a$ or $\neg d$ and can thus completely ignore the associated constraints, i.e., chains.

6 Parallelism with Tracking

There is however still room for more parallelism even compared to the improved M&R'07 encoding. The key observation is that in many domains only a few actions actually influence the truth of variables in an LTL formula, and that those that do are usually close to each other in a topological ordering of the inverse disabling graph. The ∃-step semantic guarantees that if actions are executed in parallel, they can be sequentially executed in this order. Let this ordering be $(o_1, \ldots, o_n)$. We can divide it into blocks, such that for each block $(o_i, \ldots, o_j)$ it holds that

$$[o_i]_e^\phi \supseteq [o_{i+1}]_e^\phi \supseteq \cdots \supseteq [o_j]_e^\phi$$

Along the actions in a block the effects that contain predicates in $A(\phi)$ can only "decrease". A block forms a set of actions that can always – without further checking at runtime – be executed in parallel in the M&R'07 encoding. The number of such blocks is surprisingly small for most domains (see Tab. 3). We denote with $B = ((o_1, \ldots, o_i), \ldots, (o_j, \ldots, o_n))$ the sequence of blocks for a given ordering of actions.

If an action from a block has been executed, at least (usually more) the first action of the next block cannot be executed anymore in the same timestep, as it would change the truth of some $a \in A(\phi)$, even though it would be possible in the pure ∃-step encoding. We present a method to circumvent this restriction on parallelism. Instead of restricting the amount of parallel actions executing inside a timestep, we (partially) trace the truth of an LTL formula within that timestep to allow maximal parallelism. This is based on the insight that all proofs by Mattmüller and Rintanen [17] do not actually require an action to be present at any timestep, i.e., the set of actions executed in parallel can also be empty. So, conceptually, we split each timestep into $|B|$ many timesteps and restrict the actions in the ith splitted step to be those of the ith block. Then we use the M&R'07 encoding, without the need to add chain-clauses apart from those of the ∃-step encoding, as they are automatically fulfilled. The resulting encoding would be sound and complete for LTL formulae without X and $\mathring{X}$ by virtue of the results proven by Mattmüller and Rintanen [17].

We can however improve the formula even further. In the proposed encoding, we would compute the state after each block using the Kautz&Selman encoding. This is unnecessary, as we know from the ∃-step encoding, that we only need to compute the state again after all blocks of one original timestep have been executed. We only need to trace the truth of propositions in $A(\phi)$ between blocks. For that we don not need to check preconditions – they are already ensured by the ∃-step encoding. We end up with the ∃-step encoding, where we add at every timestep a set of intermediate timepoints at which the

[3]Technically both $\overline{a}$ and $\overline{d}$ are not true.

truth of propositions in $A(\phi)$ and the truth of the LTL formula ϕ is checked. Thus we call this encoding OnParallel.

As we have noted above, this construction works only for the original M&R'07 encoding, as supporting the X and $\mathring{X}$ requires to be able to specify that in the next timestep some action must be executed. This might not be possible with splitted timesteps, as the next action to be executed may only be contained in a timestep $|B|$ steps ahead. Luckily, we can fix this problem by slightly altering the encoding we used to track the truth of X and $\mathring{X}$ operators.

$$\begin{aligned}(Xf)^t \rightarrow atMostOne^{t-1} \wedge ((atLeastOne^{it} \wedge f_{LTL}^{t+1}) \vee \\ (nextNone^{t+1} \wedge (Xf)_{LTL}^{t+1}))\\ (\mathring{X}f)^t \rightarrow atMostOne^{t-1} \wedge ((f_{LTL}^{t+1} \wedge atLeastOne^{t}) \vee \\ (nextNone^{t+1} \wedge (\mathring{X}f)_{LTL}^{t+1}) \vee none^t))\end{aligned}$$

The semantics of $noneAt^t$ is ensured by clauses $nextNone^t \rightarrow \neg o^t$ for all $i \in O$. Lastly, we add $\neg X f_{LTL}^{n++11}$ for any $Xf \in \mathcal{S}$ stating that a next-formula cannot be made true at the last timestep. This would otherwise be possible, since $atMostOne^n$ could simply be made true. The OnParallel encoding is correct by applying Thm. 3.

7 Evaluation

We have conducted an evaluation in order to assess the performance of our proposed encodings. We used the same experimental setting as Mattmüller and Rintanen [17] in their original paper. We used the domains `trucks` and `rover` from the preference track of IPC5 (the original paper considered only `rover`), which contain temporally extended goals to specify preference. In these domains, temporally-extended goals are formulated using the syntax of PDDL 3.0 [11]. We parse the preferences and transform them into LTL formulae using the patterns defined by Gerevini and Long [11]. As did Mattmüller and Rintanen, we interpret these preferences as hard constraints and randomly choose a subset of three constraints per instance[4]. To examine the performance of our encoding for X and $\mathring{X}$, we have also tested the instances of the `trucks` domain with a formula that contains these operators. We have used the following formula[5]:

$$\phi = \forall ?l{-}Location ?t - Truck : G(at(?l, ?t) \rightarrow \mathring{X}(\neg at(?l, ?t) \vee \mathring{X} \neg at(?l, ?t)))$$

It forces each truck to stay at a location for at most one timestep – either it leaves the location right after entering it, or in the next timestep. The domain contains an explicit symbolic representation of time, which is used in temporal goals. When planning with ϕ, the number of timesteps is never sufficed to find a plan satisfying ϕ. We have therefore removed all preconditions, effects, and action parameters pertaining to the explicit representation of time. As a result, the domain itself is easier than the original one. We denote these instances in the evaluation with trucks-XY-ϕ.

Each planner was given 10 minutes of runtime and 4GB of RAM per instance on an Intel Xeon E5-2660 v3. We've used the SAT solver Riss6 [15], one of the best-performing solvers in the SAT Competition 2016. We have omitted results for all the trucks instances $11 - 20$, as no planner was able to solve them.

Table 3 shows the results of our evaluation. We show per instance the number of ground actions and blocks. The number of blocks is almost always significantly smaller than the number of ground operators. In the largest `rover` instance, only $\approx 1.4\%$ of operators start a new block. For the `trucks` domain this is $\approx 1.7\%$ for the largest instance.

For every encoding, we show both the number of parallel steps (i.e. timesteps) necessary to find a solution, as well as the time needed to solve the respective formula and the number of sequential plan steps found by the planner. In almost all instances the OnParallel encoding performs best, while there are some where the improved M&R'07 encoding is faster. Our improvement to the M&R'07 encoding almost always leads to a faster runtime. Also, the improved parallelism actually leads to shorter parallel plans.

[4]Mattmüller and Rintanen [17] noted that it is impossible to satisfy all constraints at the same time and that a random sample of more than three often leads to unsolvable problems. If a sample of 3 proved unsolvable we have drawn a new one.

[5]We handle these lifted LTL constraints by grounding them using the set of delete-relaxed reachable ground predicates.

In approximately half of the instances we can find plans with fewer parallel steps. In the experiments with the formula ϕ containing the $\mathring{X}$ operator, this is most pronounced. The OnParallel encoding cuts the number of timesteps by half and is hence significantly faster, e.g., on trucks-03-ϕ where the runtime is reduced from 165s to 6s. On the other hand, the sequential plans found are usually a few actions longer, although the same short plan could be found – this result is due to the non-determinism of the SAT solver.

8 Conclusion

In this paper, we have improved the state-of-the-art in translating LTL planning problems into propositional formulae in several ways. We have first pointed out an interesting theoretical connection between the propositional encoding by Mattmüller and Rintanen [17] and the compilation technique by Torres and Baier [20]. Next, we have presented a new theoretical foundation for the M&R'07 encoding – partial evaluation traces. Using them, we presented (1) a method to allow the X and $\mathring{X}$ operators in the M&R'07 encoding, (2) a method to further improve the parallelism in the M&R'07 encoding, and (3) a new encoding for LTL planning. In an evaluation, we have shown that both our improved M&R'07 encoding and the OnParallel encoding perform empirically better than the original encoding by Mattmüller and Rintanen. We plan to use the developed encoding in a planning-based assistant [3] for enabling the user to influence the instructions he or she is presented by the assistant, which in turn are based on the solution generated by a planner. Instructions given by the user can be interpreted as LTL goal and integrated into the plan using the presented techniques.

Acknowledgement

This work is funded by the German Research Foundation (DFG) within the technology transfer project "Do it yourself, but not alone: Companion Technology for Home Improvement" of the Transregional Collaborative Research Center SFB/TRR 62. The industrial project partner is the Corporate Sector Research and Advance Engineering of Robert Bosch GmbH.

Instance			M&R'07			Improved-M&R'07			OnParallel			AA		
	$\|O\|$	$\|B\|$	p. steps	s. steps	time	p. steps	s. steps	time	p. steps	s. steps	time	p. steps	s. steps	time
rover-01	63	10	7	16	**0.10**	7	16	0.11	7	16	**0.10**	7	**14**	0.23
rover-02	53	8	5	13	0.06	**4**	13	0.03*	**4**	14	**0.03**	4	**12**	0.04
rover-03	76	9	7	16	0.25	7	20	0.2*	**6**	17	**0.16**	7	**14**	0.35
rover-04	86	16	5	13	0.13	5	11	0.12*	**3**	**9**	**0.05**	5	14	0.12
rover-05	144	15	7	**26**	0.38	7	30	0.39	**6**	27	**0.29**	7	**26**	0.63
rover-06	178	16	10	42	1.08	**9**	42	0.73*	**8**	43	**0.6**	10	**39**	1.39
rover-07	151	7	5	31	0.37	5	29	0.33*	5	28	**0.31**	5	**21**	0.59
rover-08	328	9	7	40	1.06	7	43	1.11	7	45	**0.95**	7	**36**	1.24
rover-09	362	19	9	62	3.47	9	56	**3.13***	9	59	3.38	9	**36**	3.92
rover-10	382	14	6	54	1.73	**5**	46	1.17*	**4**	44	**0.77**	6	**40**	1.62
rover-11	436	9	10	**42**	3.91	10	**42**	3.77*	**9**	43	**2.93**	10	47	6.46
rover-12	366	15	5	27	1.28	5	29	1.18*	5	33	**1.07**	5	**25**	1.43
rover-13	749	11	8	**61**	4.51	8	65	4.75	8	66	**3.99**	8	62	5.27
rover-14	525	19	7	**40**	3.66	7	43	3.66	7	42	**3.31**	7	50	4.13
rover-15	751	12	8	64	6.18	**7**	60	4.85*	**7**	60	**4.36**	7	**47**	5.34
rover-16	671	14	6	44	3.66	6	47	3.41*	6	50	**3.32**	6	**43**	3.99
rover-17	1227	13	11	106	34.97	11	105	34.49*	11	103	**31.45**	11	**62**	37.17
rover-18	1837	49	5	65	10.63	5	65	11.04	5	**64**	**10.11**	5	70	10.59
rover-19	2838	17	8	91	20.28	8	94	19.63*	8	100	**19.00**	8	**86**	21.38
rover-20	3976	58	8	130	65.99	8	127	65.97*	8	119	**65.2**	8	**98**	66.38
trucks-01	333	82	8	15	0.69	**7**	**14**	**0.44***	**7**	15	0.50	8	15	0.98
trucks-02	624	56	9	20	1.73	**8**	**17**	1.26*	**8**	18	**1.21**	9	18	1.87
trucks-03	1065	68	10	24	2.43	10	**22**	2.47	**9**	**22**	**1.68**	10	23	6.34
trucks-04	1692	48	12	27	8.48	11	27	**6.68***	11	26	6.93	12	**25**	12.80
trucks-05	2541	85	12	29	13.70	11	30	11.07*	11	**27**	**10.56**	12	28	13.37
trucks-06	4928	112	14	37	42.01	14	36	**39.39***	14	**35**	41.37	14	**35**	88.09
trucks-07	7380	267	14	40	171.81	**13**	38	119.95*	**13**	39	**119.75**	14	**36**	247.5
trucks-08	9760	260	13	42	**151.22**	13	41	153.05	13	**39**	153.81	–	–	TLE
trucks-09	12628	203	14	45	432.99	14	**42**	373.30*	14	44	**362.54**	14	43	465.15
trucks-10	16032	267	–	–	TLE	–	–	TLE	–	–	TLE	–	–	TLE
trucks-01-ϕ	123	6	18	18	2.12	18	18	2.27	**9**	18	**0.24**	18	18	18.45
trucks-02-ϕ	162	6	24	24	14.44	24	24	14.55	**12**	24	**1.09**	24	24	498.93
trucks-03-ϕ	201	6	29	29	164.56	29	29	165.64	**15**	29	**5.97**	–	–	TLE
trucks-04-ϕ	240	6	–	–	TLE	–	–	TLE	**18**	36	**84.20**	–	–	TLE
trucks-05-ϕ	279	6	–	–	TLE	–	–	TLE	–	–	TLE	–	–	TLE

Table 3: SAT-solver runtime and solution length of several encodings. Per run, we give both the number of parallel steps (p. steps) and the number of sequential (s. steps). Minimum run-times and sequential plan lengths are marked in bold. If the number of parallel plan steps decreased for any of the two new encodings, they are also marked bold. Decreases in Runtime for the improve M&R'07 encoding compared to the original one are marked with an asterisk. TLE indicates exceeding the timelimit of 10 minutes.

References

[1] Fahiem Bacchus and Froduald Kabanza. Using temporal logics to express search control knowledge for planning. *Artificial Intelligence*, 116(1-2):123–191, 2000.

[2] Jorge Baier and Sheila McIlraith. Planning with first-order temporally extended goals using heuristic search. In *Proceedings of the 21st National Conference on AI (AAAI 2006)*, pages 788–795. AAAI Press, 2006.

[3] Gregor Behnke, Marvin Schiller, Matthias Kraus, Pascal Bercher, Mario Schmautz, Michael Dorna, Wolfgang Minker, Birte Glimm, and Susanne Biundo. Instructing novice users on how to use tools in DIY projects. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence and the 23rd European Conference on Artificial Intelligence (IJCAI-ECAI 2018)*, pages 5805–5807. IJCAI, 2018.

[4] Armin Biere, Keijo Heljanko, Tommi Junttila, Timo Latvala, and Viktor Schuppan. Linear encodings of bounded LTL model checking. *Logical Methods in Computer Science*, 2(5):1–64, 2006.

[5] Alberto Camacho, Eleni Triantafillou, Christian Muise, Jorge A. Baier, and Sheila A. McIlraith. Non-deterministic planning with temporally extended goals: Ltl over finite and infinite traces. In *Proceedings of the 31st National Conference on AI (AAAI 2017)*, pages 3716–3724. AAAI Press, 2017.

[6] Giuseppe De Giacomo and Moshe Y. Vardi. Linear temporal logic and linear dynamic logic on finite traces. In *Proceedings of the 23rd International Joint Conference on Artificial Intelligence (IJCAI 2013)*, pages 854–860. AAAI Press, 2013.

[7] Patrick Doherty and Jonas Kvarnström. TALPLANNER – A temporal logic-based planner. *The AI Magazine*, 22(3):95–102, 2001.

[8] Stefan Edelkamp. On the compilation of plan constraints and preferences. In *Proceedings of the 16th International Conference on Automated Planning and Scheduling (ICAPS 2006)*, pages 374–377. AAAI Press, 2003.

[9] Alan Frisch, Timothy Peugniez, Anthony Doggett, and Peter Nightingale. Solving non-boolean satisfiability problems with stochastic local search: A comparison of encodings. *Journal of Automated Reasoning (JAR)*, 35(1-3):143–179, 2005.

[10] Paul Gastin and Denis Oddoux. Fast ltl to büchi automata translation. In *Proceedings of the 13th International Conference on Computer Aided Verification (CAV 2001)*, pages 53–65. Springer-Verlag, 2001.

[11] Alfonso Gerevini and Derek Long. Plan constraints and preferences in PDDL3. Technical report, Department of Electronics for Automation, University of Brescia, 2005.

[12] Chih-Wei Hsu, Benjamin Wah, Ruoyun Huang, and Yixin Chen. Constraint partitioning for solving planning problems with trajectory constraints and goal preferences. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI 2007)*, pages 1924–1929. AAAI Press, 2007.

[13] Henry Kautz and Bart Selman. Pushing the envelope: Planning, propositional logic, and stochastic search. In *Proceedings of the 13th National Conference on Artificial Intelligence (AAAI)*, pages 1194–1201, 1996.

[14] Timo Latvala, Armin Biere, Keijo Heljanko, and Tommi Junttila. Simple bounded LTL model checking. In *Proceedings of the 5th Conference on Formal Methods in Computer-Aided Design (FMCAD 2004)*, pages 189–200. FMCAD Inc., 2004.

[15] Norbert Manthey, Aaron Stephan, and Elias Werner. Riss 6 solver and derivatives. In *Proceedings of SAT Competition 2016: Solver and Benchmark Descriptions*, pages 56–57. University of Helsinki, 2016.

[16] Robert Mattmüller. Erfüllbarkeitsbasierte Handlungsplanung mit temporal erweiterten Zielen, 2006. diploma thesis, Albert-Ludwigs-Universität, Freiburg, Germany.

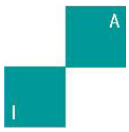
[17] Robert Mattmüller and Jussi Rintanen. Planning for temporally extended goals as propositional satisfiability. In *Proceedings of the 25th International Joint Conference on Artificial Intelligence (IJCAI 2007)*, pages 1966–1971. AAAI Press, 2007.

[18] Amir Pnueli. The temporal logic of programs. In *Proceedings of the 18th Annual Symposium on Foundations of Computer Science (SFCS 1977)*, pages 46–57. IEEE, 1977.

[19] Jussi Rintanen, Kejio Heljanko, and Ilkka Niemelä. Planning as satisfiability: parallel plans and algorithms for plan search. *Artificial Intelligence*, 170(12-13):1031–1080, 2006.

[20] Jorge Torres and Jorge A. Baier. Polynomial-time reformulations of LTL temporally extended goals into final-state goals. In *Proceedings of the 25th International Joint Conference on Artificial Intelligence (IJCAI 2015)*, pages 1696–1703. AAAI Press, 2015.

Inteligencia Artificial, 21(62) (2018), 91-102
doi: 10.4114/intartif.vol21iss62pp91-102



INTELIGENCIA ARTIFICIAL

http://journal.iberamia.org/

Deep Learning Applied on Refined Opinion Review Datasets

Ingo Jost[1], João Francisco Valiati[2]
[1]CWI Software, São Leopoldo, Brazil
ingo.jost@gmail.com
[2]Artificial Intelligence Engineers-AIE, Porto Alegre, Brazil
joao.valiati@ai-engineers.com

Abstract Deep Learning has been successfully applied in challenging areas, such as image recognition and audio classification. However, Deep Learning has not yet reached the same performance when employed in textual data classification, including Opinion Mining. In models that implement a deep architecture, Deep Learning is characterized by the automatic feature selection step. The impact of previous data refinement in the pre-processing step before the application of Deep Learning is investigated to identify opinion polarity. The refinement includes the use of a classical procedure of textual content and a popular feature selection technique. The results of the experiments overcome the results of the current literature with the Deep Belief Network application in opinion classification. In addition to overcoming the results, their presentation is broader than the related works, considering the change of parameter variables. We prove that combining feature selection with a basic preprocessing step, aiming to increase data quality, might achieve promising results with Deep Belief Network implementation.

Keywords: Deep Learning; Opinion Mining; Feature Selection; Deep Belief Networks.

1 Introduction

The continuous growth of data volume contributes to the improvement of techniques that seek the implicit knowledge of these data. The Knowledge Discovery in Database (KDD) area is furthered by the technological advances in recent years, standing out in its performance in different approaches such as Text Mining, a specific field of KDD that treats pattern recognition in textual data, like document classification[3].

When these textual data are about opinions, specific points arise and they are treated by Opinion Mining. It is a specialization of Text Mining that helps the decision making process and allows companies to understand what customers think about their products[4], and customers take better choices based on the purchase experience from other buyers. Different models have been applied in opinion classification problems. Among them stands out Deep Learning, which is employed in several fields of pattern recognition like image[10] and audio[14] identification, character classification[15], and face recognition[13]. The significant results in these areas open possibilities to apply Deep Learning in other fields like text mining. However, the Deep Learning application in opinion classification does not overcome the current literature results[1], motivating this investigation.

A major quality of Deep Learning is the feature selection from data[31], where the algorithm learns multiple levels of data representation and the learned representations can be considered as features[47].

ISSN: 1137-3601 (print), 1988-3064 (on-line)

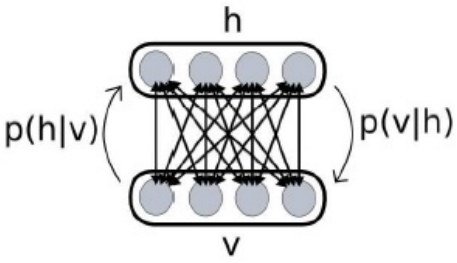


Figure 1: RBM's structure[20]

The aim of this research is to provide an analysis of the impact of data refinement, the use of a classical text pre-processing and a feature selection technique, exerts on polarity classification with the Deep Belief Network (DBN)[24], a specialization of Deep Learning. Although most researches use Deep Learning in raw data[16], this study demonstrates the benefits of pre-processing techniques.

The main contibution of this works is to demonstrate the effectiveness of the combination between classical preprocessing, with a feature selection method, and DBN can provide competitive results when compared with the current literature (see Table 6). Deep Learning, with focus on DBN, and Opinion Mining are introduced in the next sections, followed by related works that build models with deep architectures to apply on review data. After, the experiments and the obtained results are related. Finally, the conclusion is presented.

2 Deep Learning

Machine Learning is a research field open to the development and extension of methods with a continuous search for best practices, considering costs and results. This search favours the development of new areas, such as Deep Learning. Its concepts are based on Artificial Neural Networks (ANN)[49], having biological inspiration in the human brain and studies about the mammalian visual cortex[9].

In recent years researchers have tried to increase the quantity of layers of ANNs[27]. Success was not achieved until 2006, when an algorithm was proposed to train Deep Belief Networks[24]. This algorithm uses multiple layers composed of non-linear information for feature selection (supervised or unsupervised), transformation and pattern analysis, trying to identify relationships in the data[31].

The use of unsupervised learning algorithms to learn features from unlabeled data has contributed to Deep Learning[21] growth, evidencing its skill in feature selection and distributed representation on multiple levels[32]. Through various researches, the classifiers have been lead to achieve better results when using Deep Learning.

These results were also obtained in various Text Mining approaches: semantic and sense identification from terms[22], text clustering[3] and domain adaptation of Opinion Mining[23]. Ain et al.[52] presented a review of distinct approaches of Deep Learning in Opinion Mining. Similarly, Zhang et al.[53] provided a survey of Deep Learning applications on tasks of Sentiment Analysis. They were implemented by different architectures such as Stacked Auto-encoder, Convolutional Neural Networks[30, 51] and Deep Belief Networks.

The DBNs are probabilistic models composed by one visible and many hidden layers[30]. Each of these is compounded by Restricted Boltzmann Machines (RBM), learning statistical relationships among the lower level layers. The RBMs are a specific case of Boltzmann Machines formed by visible (v) and hidden (h) units, with the restriction of forbidden connections between neurons of the same layer[21], according to Figure 1. Their structure is bipartite graph, consisting of a stochastic neural network with only one hidden layer each, which tries to find a likeness from input data in an unsupervised way. The Boltzmann Machines (including RBMs) are Energy Based Models (EBM), associating an energy scalar for each joint configuration (pairs of visible and hidden units)[27]. This energy value allows calculating the distribution p from vector v, used for unit computing and updating of weights[20].

The DBN layers are formed by stacked RBMs hierarchically arranged, which are individually trained[19],

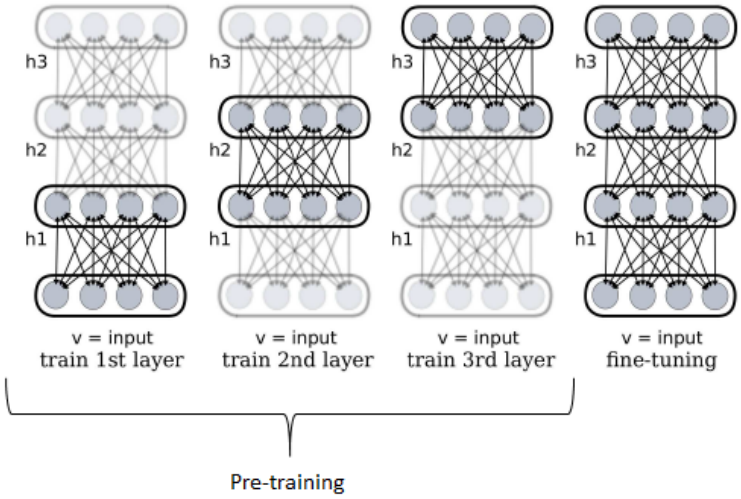


Figure 2: The DBN's training steps[20]

and the hidden units from each RBM are the visible units for the next layer (except for the last layer). This step, in which the RBMs are trained layer-by-layer, is called Pre-training. Thereafter, the whole model has the weights adjusted by backpropagation in the Fine-tuning stage. These steps characterize the algorithm proposed by Hinton et al.[24] and are illustrated in Figure 2, where each RBM is highlighted during its training in the Pre-training and the layers are connected in Fine-tuning.

The number of epochs in the Pre-training and Fine-tuning steps and the number of neurons for hidden layers in deep architectures are empirically determined, such as in shallow architectures. However, the researches that normally apply DBN use three hidden layers, with no references that recommend a higher number. Our work followed the premises of most of the works and employed three hidden layers (see Experimental Setup section).

3 Opinion Mining

Opinion Mining is a Text Mining specialization that deals with opinions[26]. Its growth is favoured by the web 2.0 scenario and social media content[25], a collaborative environment that makes possible the development of tools that allow users to send their opinion through discussion groups, forums, social networks, blogs, websites news, product sales, etc. Furthermore, the corporations increasingly need to understand the feelings of customers about their products and services[11]. This information helps in the decision making process and strengthens the need for studies on Opinion Mining.

Several challenges appear when the terms can have different meanings and a positive review can have terms frequently used in negative sentences, such as subjectivity. Moreover, there are sentiments that are very hard to identify, like irony and sarcasm, and the contradiction of ideas with the use of negation. In addition to these questions, the handling of the occurrence of implicit and conditional opinions is needed[26]. Although opinions belong to different domains (products or news) and are generated by different tools, studies for classifying opinions are often directed toward a common aim: polarity identification (opinion classification). This consists of identifying the class to which an opinion refers: positive or negative[11].

Since the Opinion Mining treats textual data, some specific procedures for handling texts are performed [43]: tokenization, the identifying words (tokens) in a text[44]; stemming, to group different terms but with the same radical[45]; and stopwords removal. Stopwords are terms that often occur in text but do not get to contribute to the process of classification or pattern recognition. In order to reduce the volume, these words can be removed without harming the meaning of the text. Besides, the feature selection (term selection) procedure is applied to reduce the data volume and enhance their quality. Among the

techniques for feature selection, such as Probability Ratio and Chi-squared, the IG is a popular feature selection approach[5], with a low computational cost and it has reached competitive results in the feature selection[41, 42]. The IG makes use of entropy and calculating the information gain for each term. More information about data is provided through this metric, enabling improvement in their quality and turning them into a refined form. These procedures are part of the pre-processing step.

After pre-processing there is a transformation step, in which the opinions are transformed into word vectors, where each position represents a term and storage value that will be used by the classifier algorithm. This value is a weight representation, such as TF-IDF[34] or the term frequency. In this way, the data is suited to the classifiers, like DBN, to be trained. The next section presents related works that implemented these applications using Deep Learning techniques.

4 Related Works

According to Xia et al.[46], the studies involving Opinion Mining usually follow a traditional text classification, where the documents are mapped into a feature vector through the bag-of-words (BOW) model and posteriorly classified by Machine Learning techniques, like Naïve Bayes (NB) or Support Vector Machines (SVM). Many works apply neural networks[1] with shallow architectures. Although there are recent studies that investigate several flavors of Deep Learning applied to Sentiment Analysis like exposed in [53], our focus is on the works that investigated the same datasets, such as the works presented in this section.

The work presented in Zhou et al.[7] uses the Active Learning model to classify opinions, comparing their results with the experiments done by other authors with the Deep Belief Networks implemented by Hinton[24]. Glorot et al.[9] develops a Deep Learning model with Denoising Autoencoder[27], adopting a rectifier function (and its smooth form, the Softplus function[12]) for neuron activation to image recognition and applies the same model to opinion classification. Both works produce experiments with the same datasets investigated in our work (movies and books reviews) and their results are explored as a benchmark. Glorot et al.[9] is an exception because their results are reported as an average of four datasets.

The following sections present the specifications of these works and determined aspects not appropriately explored, like the networks parameterization.

4.1 Active Learning

Zhou et al.[7] proposed the Active Deep Network, a semi-supervised learning method to classify opinions through Deep Learning architecture. The experiments were performed for datasets from Pang and Lee[8] and on another four datasets of different types of products from Amazon: books (BOO), DVDs (DVD), electronics (ELE), and kitchen's appliances (KIT). Each set was formed by 2,000 opinions equally divided into positive and negative classes.

The pre-processing was similar to Dasgupta[28], where vectors of unigrams represent opinions. The terms were selected by frequency, discarding the 1.5% used more frequently (assuming that these represent stopwords). The training was called semi-supervised because only part of the samples were labeled to adjust the weights. In the Pre-training step, all samples participated in the layer-by-layer unsupervised training architecture to generate the weight matrix. The supposed most difficult samples to classify were chosen from the training set through the technique of Active Learning, in order to consider the label for supervised training. This difficulty level was measured by the distance of the sample from the class separator on the hyperplane (the closer to the separator, the more difficult it was to classify). Finally, the model training considered the selected samples.

The authors performed the experiments with 10-folds randomly divided, which were tested with cross-validation. The results were compared with the other referenced models, highlighting the reached accuracy by Active Deep Learning (ADN) for books and movies datasets, 69% and 76%, respectively, overcoming the experiments of the authors with the original DBN for the same datasets (64% and 71%).

An extension for ADN was proposed in Zhou et al.[2]: the Information ADN (IADN), which used the information's density to choose the samples that will pass for supervised learning. Instead of considering

just the distance of the point from the separator point on the hyperplane, it also considered the distance from the center of the classes.

The experiments reached similar results to those obtained by the ADN when 100 labeled samples were used for the five datasets. However, the IADN produced better results when 10 labeled samples were used. Just one configuration of neurons per layer was used, the last layer being the output and the previous three layers corresponding to the hidden layers: 100-100-200-2. Furthermore, the training occurred with only 30 epochs.

The model demonstrated in this section improved the DBN implementation. The results were presented with just one setup of neurons per hidden layer, one number of epochs, and a fixed number of terms. In our work, an original DBN implementation was suited and the obtained accuracy was higher than experiments from Zhou et al.[7]. Moreover, it analysed the parameters change and its influence on results.

4.2 Rectifier Neural Network

Glorot et al.[9] proposed the use of the rectifier activation function for neurons, analysing the effects of using the Pre-training step in their Deep Learning framework. The work was an extension of Nair and Hinton[10].

The rectifier function generated a sparse representation of the data, according to biological inspiration, since studies have shown that neurons store information sparsely[29]. An advantage of this representation was the possibility of keeping data variability. When very dense, small changes can affect their vector representation[27]. Furthermore, sparse data have a tendency to be linearly separable.

Four datasets were used: MNIST, images of digits; CIFAR10, RGB images; NISTP, character images; and Norb, picture of toys. The experiments were realized with and without the Pre-training step, comparing the results. The experiments that performed the Pre-training step achieved the best results.

In addition to the sets of images, the model was applied to classify sentiments in reviews from the OpenTable website (10,000 labeled opinions and 300,000 not labeled). Each review was reduced through the BOW model and converted to binary vector, representing the presence or absence of the term. The 5,000 most frequent terms were considered. The experiments using the Pre-training step produced higher accuracy than experiments without Pre-training, proving its effectiveness.

Since no publication was related to the OpenTable data, the authors applied the rectifier neural network following the pre-processing setup defined by Zhou et al.[7], with the same datasets, using the four datasets from Amazon (BOO, DVD, ELE and KIT). The work reached an average accuracy of 78.9%, overcoming the 73.7% obtained by Zhou et al.[7]. Additionally, the results achieved by Ghosh et al.[54], which combined a two layered RBM to dimensionality reduction and a Probabilistic Neural Network (PNN), and Ruangkanokmas et al.[48], whom applied the Deep Belief Network to the Feature Selection (DBNFT) model, combining the Chi-square feature selection technique with common pre-processing before the DBN application, are also demonstrated. Table 1 presents the average accuracy obtained in the discussed works.

Table 1: Accuracy average (%) in related works in BOO, DVD, ELE and KIT datasets

Model	Average
DBN (Zhou et al. [7])	69.3
ADN (Zhou et al. [7])	73.7
IADN (Zhou et al. [2])	64.5
DBNFT (Ruangkanokmas et al. [48])	71.4
RNN (Glorot et al.[9])	78.9
PNN (Ghosh et al.[54])	80.1

These works demonstrated that Deep Learning offers different possibilities for application in opinion classification. Recent works have applied Deep Learning to obtain promising results in Sentiment Analysis considering data from social networks, such as the following propositions: Deep Recurrent Neural

Networks [36], Deep Memory Networks [38] and the proposed of Wang et al. [37], which combined Convolutional Neural Network and Recurrent Neural Network (CNN+RNN). Despite these last approaches, our work showed that the application of more effort in the pre-processing step lead to greater accuracy. The obtained accuracy of related works was presented for further comparison with our results (see Comparison of Results section).

5 Experiments

This investigation used two datasets: the classical dataset of opinions about movies from Pang and Lee[8] and opinions about books from Amazon. The employed evaluation made use of the 10-fold cross-validation method, and over each set of training and tests generated by these folds preprocessing and classification tasks were applied. The IG method was applied to find more representative terms after the use of a basic clean procedure: tokenization, removal stopwords list, and stemming (Snowball algorithm). We followed the premises of Moraes et al.[5], which created sets with different amounts of terms, then we evaluated the respective number of terms (300, 500, 1,000, 2,000, 3,000, 4,000, 5,000). Different from the related works[2, 7, 9], we also analyse the extension of network parameters, like the number of neurons by layer.

The Pre-training stage was perceived in the studies related with textual data as an unsupervised feature selector[3, 23], following the approaches of other fields, like image recognition[50]. When these applications were made for opinion classification, a basic pre-processing, considering the frequency of terms, was applied[9, 7]. The purpose of applying IG for term selection in our investigation was to identify the influence that a basic and popular dimensionality reduction technique exerts over DBN. Before effectively applying the DBN, the data were transformed into vectors containing the Term Frequency from each selected term.

The implementation of a classifier with a deep architecture by Ruslan Salakhutdinov and Geoffrey Hinton[1] was used for experimenting. This classifier was a framework that implemented Deep Learning concepts, which trained a DBN with three hidden layers formed by RBMs. These layers were individually trained in the Pre-training stage and then the weights were adjusted in the Fine-tuning step.

The original framework's code was adapted to train and test opinion data, including modifications in input and output layers due to domain specification. An architecture with three hidden layers was maintained for all experiments, according to related works. The steps are illustrated in Figure 3. After the pre-processing (which prepared the refined data) and training steps, the model produced an output for each sample of the test set. This output was compared with the intended values, generating the confusion matrix, formed by the TP, TN, FP, and FN values.

The True Positives (TP) were the positive samples correctly classified and True Negatives (TN) the negative samples predicted as negative. The False Positives (FP) and False Negatives (FN) corresponded to the sample amount wrongly predicted as positive and negative, respectively[43]. From these metrics is calculated the accuracy (Eq. (1)) used in the related works and in our results.

$$accuracy = \frac{TP + TN}{TP + FP + TN + FN} \tag{1}$$

5.1 Experimental Setup

Maintaining a similar architecture found in related works, three hidden layers were adopted. The initial settings used in Zhou et al.[7] were replicated, i.e., three hidden layers with 100, 100, and 200 neurons and 30 epochs for training. Preliminary experiments demonstrated that a greater number of neurons per layer and more epochs contributed to improve the accuracy.

Seeking a suitable configuration of nodes in hidden layers that could produce better accuracy, experiments were realized beginning with the configuration from Zhou et al.[7], and gradually and proportionally the number of neurons was increased. These settings of neurons per hidden layer were used: 100, 100, 200; 200, 200, 400; 300, 300, 600; 400, 400, 800; and 500, 500, 1,000.

[1] http://www.cs.toronto.edu/~hinton/MatlabForSciencePaper.html

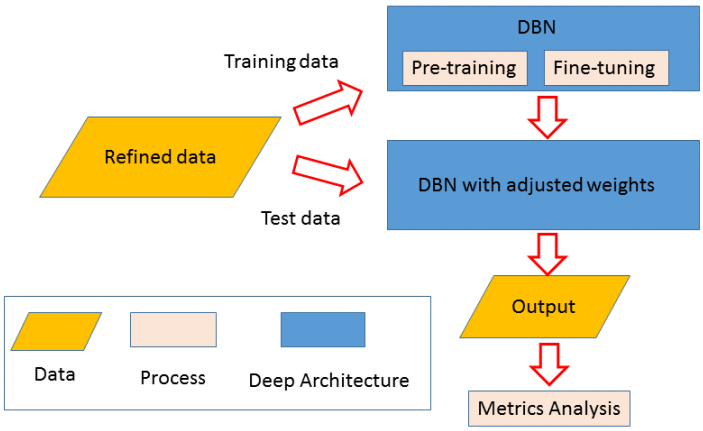


Figure 3: The flowchart of experiments

Since the Pre-training stage followed an unsupervised step, the end of training procedures was determined by the number of epochs. After preliminary tests, we chose to realize experiments with a higher number of epochs for training if compared to Zhou et al.[7]. We followed the approaches that applied Deep Learning in textual data, in which experiments were done with fewer epochs in the Pre-training[3], preventing a high number of epochs in this unsupervised step harmed the training. Different amounts of epochs were analysed for the Pre-training (PT) and Fine-tuning (FT) steps: 120 (PT) and 120 (FT); 160 (PT) and 160 (FT); and 30 (PT) and 120 (FT).

5.2 Obtained Results

Experiments were realized in movies and books datasets, varying the configuration of the three hidden layers and the sets with a different number of terms, as previously specified. Applying the configuration with 120 epochs in the Pre-training and Fine-tuning steps, the maximum accuracy obtained was 82% for movies (considering 2,000 terms) and 77.7% for books (considering 500 terms).

We repeated the same configurations with 160 epochs in Pre-training and Fine-tuning steps to check whether or not the continuous increase of epochs contributed for reaching better results. The accuracy obtained is presented in Tables 2 and 3, where it is verified that the movies dataset accuracy was greater in most of the experiments (comparing to experiments with 120 epochs in the Pre-training and Fine-tuning), while for books the results were closer to previous experiments.

Table 2: Accuracy (%) for movies - 160 epochs in Pre-training and Fine-tuning steps

Layers	Number of terms						
1st, 2nd, 3rd	300	500	1,000	2,000	3,000	4,000	5,000
100, 100, 200	79.0	75.9	75.2	75.5	75.2	75.2	75.3
200, 200, 400	**81.6**	81.9	82.0	80.8	80.4	77.3	76.3
300, 300, 600	81.1	**82.2**	**82.4**	**82.8**	**82.4**	82.3	**82.6**
400, 400, 800	80.7	81.8	82.1	82.7	**82.4**	82.1	82.5
500, 500, 1,000	79.6	81.1	82.1	82.3	**82.4**	**82.7**	82.2

Table 3: Accuracy (%) for books - 160 epochs in Pre-training and Fine-tuning steps

Layers	Number of terms						
1st, 2nd, 3rd	300	500	1,000	2,000	3,000	4,000	5,000
100, 100, 200	77.0	77.4	**76.8**	75.8	75.9	74.9	75.3
200, 200, 400	**77.3**	**77.7**	76.7	76.3	76.3	75.6	75.6
300, 300, 600	77.1	76.7	76.5	76.5	75.8	76.4	75.8
400, 400, 800	76.0	76.7	76.4	**76.6**	**76.6**	75.9	**76.0**
500, 500, 1,000	76.2	76.6	76.2	76.5	76.2	**76.5**	75.4

Although higher accuracy was obtained for the movies dataset, the improvement was not significant. For this reason, we adopted the strategy from Salakhutdinov and Hinton[3] that reduced the number of epochs in the Pre-training step. This was done in order to avoid unsupervised training with a high number of epochs, which caused the overfitting problem[39].

The obtained results in experiments with 30 epochs in Pre-training and 120 in Fine-tuning are shown in Tables 4 (movies) and 5 (books). The accuracy for the movies dataset did not exceed the obtained results in previous experiments (with 120 and 160 epochs in both steps). However, the strategy of a smaller number of epochs reached greater accuracy for the books dataset.

Table 4: Accuracy (%) for movies - 30 epochs in Pre-training and 120 in Fine-tuning

Layers	Number of terms						
1st, 2nd, 3rd	300	500	1,000	2,000	3,000	4,000	5,000
100, 100, 200	79.1	75.1	75.0	74.3	74.2	73.8	73.9
200, 200, 400	**81.1**	**81.6**	**81.5**	80.6	79.1	77.2	76.5
300, 300, 600	80.6	81.0	81.3	81.3	81.3	**81.5**	81.4
400, 400, 800	80.0	80.6	81.4	81.4	**81.5**	**81.5**	**81.7**
500, 500, 1,000	78.3	80.0	80.8	**81.8**	81.1	81.0	81.5

Table 5: Accuracy (%) for books - 30 epochs in Pre-training and 120 in Fine-tuning

Layers	Number of terms						
1st, 2nd, 3rd	300	500	1,000	2,000	3,000	4,000	5,000
100, 100, 200	**77.3**	77.6	**77.2**	76.3	75.8	75.1	**76.0**
200, 200, 400	77.0	**77.8**	**77.2**	76.4	**76.6**	75.8	75.5
300, 300, 600	76.6	76.7	76.7	**77.2**	76.4	**76.5**	75.9
400, 400, 800	75.9	76.4	76.3	76.1	75.6	76.2	75.7
500, 500, 1,000	76.0	75.9	76.1	75.5	75.7	75.7	74.9

In spite of the lower accuracy achieved for the movies dataset, the obtained results were competitive with the previous configurations. Moreover, when the Pre-training step with a smaller number of epochs was realized, the computational demand was significantly lower, turning this configuration to the recommended setting.

The data refinement strategy produced satisfactory results for the most experiments and settings, overcoming the related works, as discussed in the next section.

5.3 Comparison of Results

In Zhou et al.[7], the proposed Active Deep Learning (ADN) model applied the frequency for the terms selection and reached 76.3% accuracy for the movies dataset and 69% for the books dataset, overcoming

the 71.3% and 64.3% (movies and books, respectively) obtained with the same experiments with DBN implementation. The ADN results were adopted as a benchmark in the IADN[2], Rectifier Neural Network (RNN)[9] and Ruangkanokmas et al.[48].

In this investigation, a complementary experiment was realized submitting the movies reviews in raw format to the AlchemyAPI, an online commercial tool, recently aquired by IBM Watson, that receives documents and identifies through Deep Learning implementation (without informing the applied algorithm) whether or not they indicate a positive or negative sentiment.

Table 6 shows that the presented work achieved the best accuracy when compared with the results produced by Zhou et al.[7] in experiments with original DBN, with the ADN model, and its improvement (IADN)[2]. Moreover, the obtained accuracy in the AlchemyAPI experiments, the results from DBNFT[48], PNN[54] and CNN+RNN[37] were also related. The results of Glorot et al.[9] were not included because their presentation was an average (78.9%) of all datasets, considering different data from our work.

Table 6: Comparison of Results - Accuracy(%)

Model	Movies	Books
DBN Zhou et al. [7]	71.3	64.3
ADN Zhou et al. [7]	76.3	69.0
IADN Zhou et al. [2]	76.4	69.7
AlchemyAPI	77.8	-
DBNFT Ruangkanokmas et al. [48]	72.2	66.0
PNN Ghosh et al. [54]	80.8	81.0
CNN+RNN Wang et al. [37]	82.3	-
Present work	82.8	77.8

Although our DBN implementation did not exceed the results obtained in opinion classification researches, such as Bai[17] beating the 90% accuracy for the movies dataset from Pang and Lee[8], the strategy of refining data in the pre-processing step before the DBN application overcame the results found in related works that had applied a simple pre-processing. Moreover, the obtained accuracy was higher than the accuracy produced by the experiments with the AlchemyAPI using the raw data. Besides to exceeding the results, some network parameters were analysed to verify their impact in the deep architecture. Comparing Tables 2 and 4 for movies and Tables 3 and 5 for books, the variation of these parameters did not produce statistically significant improvements.

6 Conclusion

It was observed that the obtained results for opinion classification with the use of Deep Belief Networks, like reported by the current literature, do not exceed the results obtained with classical data mining techniques. However, this investigation opens possibilities of different approaches of Deep Learning application, such as the previous refinement of the data and the analysis of different parameters for the classifiers. The use of refinements in the original datasets, like the application of pre-processing techniques and feature selection[5], combined with the Deep Belief Network implementation for training and classifying of polarity classes, helps reach promising results.

Although the works compared from the current literature used a basic pre-processing, they only considered the frequency of terms. Our investigation proved that the strategy with data refinement - applying IG - allowed achieving and even overcoming the obtained results, increasing the accuracy in 6% and 8% for movies and books datasets, respectively. The experiments were realized with a wide setup, including a variety of parameterization. These experiments produced close results, confirming the successful data refinement strategy for the most experiments. Future extensions of the proposed work conducted with the use of this methodology to other datasets related to opinion classification, and the possibility to investigate a particular extension of Deep Learning, called Recursive Neural Tensor Network (RNTN)[6], have recently been recommended for Sentiment Analysis.

7 Acknowledgments

We thank CAPES (Coordenação de Aperfeiçoamento de Pessoal de Nível Superior) for the financial support.

References

[1] K. Ravi and V. Ravi, "A survey on opinion mining and sentiment analysis: Tasks, approaches and applications," *Knowledge-Based Systems* (2015). http://dx.doi.org/10.1016/j.knosys.2015.06.015

[2] S. Zhou, Q. Chen, X. Wang, "Active deep learning method for semi-supervised sentiment classification," *Neurocomputing* **120** (2016) 536–546.

[3] R. Salakhutdinov and G. Hinton, "Semantic hashing," International Journal of Approximate Reasoning **50** (2010) 969–978. doi:10.1016/j.ijar.2008.11.006

[4] S. Basari, B. Hussin, G. P. Ananta and J. Zeniarja, "Opinion Mining of Movie Review using Hybrid Method of Support Vector Machine and Particle Swarm Optimization," *Procedia Engineering* **53** (2013) 453–462. doi:10.1016/j.proeng.2013.02.059

[5] R. Moraes, J. F. Valiati and W. G. Neto, "Document-level sentiment classification: An empirical comparison between SVM and ANN," *Expert Systems with Appli cations* **40** (2013) 621–633. doi:10.1016/j.eswa.2012.07.059

[6] R. Socher, A. Perelygin, J. Y. Wu, C. D. Manning, A. Ng and C. Potts, "Recursive deep models for semantic compositionality over a sentiment treebank," *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2013, pp. 1631–1642.

[7] S. Zhou, Q. Chen and X. Wang, "Active deep networks for semi-supervised sentiment classification," *23rd International Conference on Computational Linguistics, Association for Computational Linguistics*, 2010, pp. 1515–1523.

[8] B. Pang and L. Lee, "A sentimental education: sentiment analysis using subjectivity summarization based on minimum cuts," *42nd Annual Meeting on Association for Computational Linguistics, Association for Computational Linguistics,* 2004, pp. 271–278. doi: 10.3115/1218955.1218990

[9] X. Glorot, A. Bordes and Y. Bengio, "Deep Sparse Rectifier Neural Networks," *International Conference on Artificial Intelligence and Statistics*, 2011, pp. 315–323.

[10] V. Nair and G. Hinton, "Rectified linear units improve restricted boltzmann machines," *27th International Conference on Machine Learning*, 2010, pp. 807–814.

[11] B. Liu, *Sentiment Analysis and Opinion Mining*, Morgan & Claypool Publishers, 2012.

[12] C. Dugas, Y. Bengio, F. Belisle, C. Nadeau and R. Garcia, "Incorporating second-order functional knowledge for better option pricing," *Advances in Neural Information Processing Systems*, 2001, pp. 472–478.

[13] H. Fan, Z. Cao, Y. Jiang, Q. Yin and C. Doudou, *Learning Deep Face Representation*, Cornell University Library, Computer Vision and Pattern Recognition, 2014. arXiv preprint arXiv:1403.2802

[14] M. Norouzi, *Convolutional Restricted Boltzmann Machines for Feature Learning*, Simon Fraser University, 2009.

[15] G. Hinton and R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science* **313** (2006) 504–507.

[16] Y. Bengio, P. Lamblin, D. Popovici and H. Larochelle, "Greedy layer-wise training of deep networks," *Advances in Neural Information Processing Systems* **19** (2007) 153–160.

[17] X. Bai, "Predicting consumer sentiments from online text, " *Decision Support Systems* **4** (2011) 732–742. doi:10.1016/j.dss.2010.08.024

[18] C. M. Bishop, *Neural Networks for Pattern Recognition*, Oxford University Press, New York, 1995.

[19] G. Hinton, "Deep Belief Nets," *NIPS Tutorial, Canadian Institute for Advanced Research and Department of Computer Science University of Toronto*, 2007.

[20] L. Arnold, S. Rebecchi, S. Chevallier and H. P. Moisy, "An introduction to deep-learning," *European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning*, 2011.

[21] Y. Yang, "Learning Hierarchical Representations for Video Analysis Using Deep Learning," Ph. D. Thesis, University of Central Florida, 2013.

[22] A. L. Maas, R. E. Daly, P. T. Pham, D. Huang, A. Ng and C. Potts, "Learning Word Vectors for Sentiment Analysis," *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies 1*, 2011, pp. 142–150.

[23] X. Glorot, A. Bordes and Y. Bengio, *Domain Adaptation for Large-Scale Sentiment Classification: A Deep Learning Approach*, Omnipress, 2011.

[24] G. Hinton, S. Osindero and Y. M. Teh, "A Fast Learning Algorithm for Deep Belief Nets," *Neural Computation* **18** (2006) 1527–1554.

[25] H. Chen and D. Zimbra, "AI and Opinion Mining," *IEEE Intelligent Systems* **25** (2010) 74–80. doi:10.1109/MIS.2010.75

[26] B. Pang and L. Lee, "Opinion mining and sentiment analysis," *Foundations and Trends in Information Retrieval* **2** (2008) 1–135.

[27] Y. Bengio, *Learning Deep Architectures for AI*, Now Publishers Inc, Boston, 2009.

[28] S. Dasgupta and V. Ng, "Mine the Easy, Classify the Hard, A Semi-supervised Approach to Automatic Sentiment Classification," *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, 2009.

[29] D. Attwell and S. B. Laughlin, "An Energy Budget for Signaling in the Grey Matter of the Brain," *Journal of Cerebral Blood Flow and Metabolism* **21** (2001) doi:10.1097/00004647-200110000-00001

[30] H. Lee, P. T. Pham, Y. Largman and A. Ng, "Unsupervised feature learning for audio classification using convolutional deep belief networks," *Advances in Neural Information Processing Systems 22: 23rd Annual Conference on Neural Information Processing Systems*, 2009.

[31] L. Deng and D. Yu, *Deep Learning: Methods and Applications*, NOW Publishers, 2014.

[32] R. Socher, Y. Bengio and C. D. Manning, "Deep learning for NLP (without magic)," *Tutorial Abstracts of ACL 2012, Association for Computational Linguistics*, 2012.

[33] Y. He, C. Lin and H. Alani, "Automatically Extracting Polarity-bearing Topics for Cross-domain Sentiment Classification," *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies* 1, 2011, pp. 123–131.

[34] G. Paltoglou and M. Thelwall, "A Study of Information Retrieval Weighting Schemes for Sentiment Analysis," *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, 2010, pp. 1386–1395.

[35] A. Yessenalina, Y. Yue and C. Cardie, "Multi-level Structured Models for Document-level Sentiment Classification," *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, 2010, pp. 1046–1056.

[36] C. Li, X. Guo and Q. Mei, "Deep Memory Networks for Attitude Identification" *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*, 2017, pp. 671-680. doi: 10.1145/3018661.3018714

[37] X. Wang, W. Jiang, Z. Luo, "Combination of convolutional and recurrent neural network for sentiment analysis of short texts" *Proceedings of the International Conference on Computational Linguistics*, 2016, doi: 10.1145/3155133.3155158

[38] Z. Zhao, H. Lu, D. Cai, X. He and Y. Zhuang, "Microblog Sentiment Classification via Recurrent Random Walk Network Learning" *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence*, 2017, pp. 3532–3538. doi: 10.24963/ijcai.2017/494

[39] C. Schittenkopf, G. Deco and W. Brauer, "Two Strategies to Avoid Overfitting in Feedforward Networks," *Neural Networks* **10** (1997) 505–516. doi:10.1016/S0893-6080(96)00086-X

[40] N. V. Chawla, "Data Mining for Imbalanced Datasets, An Overview," *Data Mining and Knowledge Discovery Handbook*, 2nd ed. Springer, 2010, pp. 853–867.

[41] Y. Yang and J. O. Pedersen, "A Comparative Study on Feature Selection in Text Categorization," *Fourteenth International Conference on Machine Learning*, 1997, pp. 412–420.

[42] A. Abbasi, S. France, Z. Zhang and H. Chen, "Selecting Attributes for Sentiment Classification Using Feature Relation Networks," *IEEE Transactions on Knowledge and Data Engineering* **23** (2011) 447–462. doi:10.1109/TKDE.2010.110

[43] S. M. Weiss, N. Indurkhya and T. Zhang, *Text mining. Predictive methods for analyzing unstructured information*, Springer, 2004.

[44] R. Feldman and J. Sanger, *Text Mining Handbook: Advanced Approaches in Analyzing Unstructured Data*, Cambridge University Press, 2006.

[45] M. W. Sholom, I. Nitin and Z. Tong, *Fundamentals of Predictive Text Mining*, Springer, 2006.

[46] S. M. Weiss, N. Indurkhya and T. Zhang, "Ensemble of Feature Sets and Classification Algorithms for Sentiment Classification," *Information Sciences* **181** (2011) 1138–1152. doi:10.1016/j.ins.2010.11.023

[47] D. Tang, B. Qin and T. Liu, "Deep learning for sentiment analysis: successful approaches and future challenges," *WIREs Data Mining Knowl Discov* **5** (2015) 292–303. doi: 10.1002/widm.1171

[48] P. Ruangkanokmas, T. Achalakul, K. Akkarajitsakul, "Deep Belief Networks with Feature Selection for Sentiment Classification," *7th International Conference on Intelligent Systems, Modelling and Simulation*, 2016.

[49] C. M. Bishop, *Neural Networks for Pattern Recognition*, Oxford University Press, New York, 1995.

[50] K. He, X. Zhang, S. Ren, J. Sun, "Deep Residual Learning for Image Recognition," 2015. arXiv 1512.03385v1

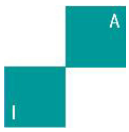
[51] A. Mishra, K. Dey, P. Bhattacharyya, "Learning cognitive features from gaze data for sentiment and sarcasm classification using convolutional neural network,", Proceedings of the Annual Meeting of the Association for Computational Linguistics 2017. doi: 10.18653/v1/P17-1035

[52] Q. T. Ain, M. Ali, A. Riaz, A. Noureen, M. Kamran, B. Hayat and A. Rehman, "Sentiment Analysis Using Deep Learning Techniques: A Review", *International Journal of Advanced Computer Science and Applications* **6** 2017. doi: 10.14569/IJACSA.2017.080657

[53] L. Zhang, S. Wang, B. Liu, "Deep Learning for Sentiment Analysis: A Survey", *WIREs Data Mining Knowl Discov.* **8** 2018. doi: 10.1002/widm.1253

[54] R. Ghosh, K. Ravi, V. Ravi, "A novel deep learning architecture for sentiment classification," *3rd Int'l Conf. on Recent Advances in Information Technology* 2016. doi: 10.1109/RAIT.2016.7507953

Inteligencia Artificial, 21(62) (2018), 103-113
doi: 10.4114/intartif.vol21iss62pp103-113



INTELIGENCIA ARTIFICIAL

http://journal.iberamia.org/

Compact Tree Encodings for Planning as QBF

Olivier Gasquet, Dominique Longin, Frédéric Maris, Pierre Régnier, Maël Valais
IRIT – University of Toulouse, Toulouse, France
{gasquet,longin,maris,regnier,valais}@irit.fr

Abstract Considerable improvements in the technology and performance of SAT solvers has made their use possible for the resolution of various problems in artificial intelligence, and among them that of generating plans. Recently, promising Quantified Boolean Formula (QBF) solvers have been developed and we may expect that in a near future they become as efficient as SAT solvers. So, it is interesting to use QBF language that allows us to produce more compact encodings. We present in this article a translation from STRIPS planning problems into quantified propositional formulas. We introduce two new Compact Tree Encodings: CTE-EFA based on Explanatory frame axioms, and CTE-OPEN based on causal links. Then we compare both of them to CTE-NOOP based on No-op Actions proposed in [3]. In terms of execution time over benchmark problems, CTE-EFA and CTE-OPEN always performed better than CTE-NOOP.

Keywords: Planning, Quantified Boolean Formula, Encodings

1 Introduction

An algorithmic approach for plans synthesis is automated compilation (i.e., transformation) of planning problems. In the SATPLAN planner [12], a planning problem is transformed into a propositional formula whose models, corresponding to solution plans, can be found using a SAT solver. The SAT approach searches for a solution-plan of fixed length k. In case of failure to find such a plan, this length is increased before restarting the search for a solution. In the classical framework, the complexity of finding a solution to any problem is PSPACE-hard, but the search for a fixed-size solution becomes NP-hard [2]. This compilation approach directly benefits from improvements in SAT solvers[1]. The most obvious example is the planner BLACKBOX [14, 15] (and its successors SATPLAN'04 [11] and SATPLAN'06 [16]). These planners won the optimal (in the number of plan steps) planning track of the International Planning Competitions[2] IPC-2004 and IPC-2006. This was unexpected because these planners were essentially updates of BLACKBOX and did not include any real novelty: improved performance was mainly due to progresses in the underlying SAT solver.

Numerous improvements of this original approach have been proposed since then, in particular via the development of more compact and efficient encodings: [13, 7, 18, 19, 22, 23, 24, 28].

Following these works, numerous other similar techniques for encoding planning problems have been developed: Linear Programming (LP) [30], Constraint Satisfaction Problems (CSP) [6], SAT Modulo Theories (SMT) [29, 20, 27]. More recently, a Quantified Boolean Formulas (QBF) approach had been proposed by [26, 3]. Currently SAT solvers outperform QBF solvers and the SAT approach is the most effective because SAT solvers and encodings have been greatly improved since 1992. However, over the past decade, there has been a growing interest in the QBF approach. The competitive evaluation of QBF solvers QBFEVAL[3] is now a joint event with the international SAT conference and QBF solvers improve regularly. QBFEVAL'16 had more participants than ever and QBF-related papers represented 27% of all papers published at SAT'16. Some promising techniques have been adapted

[1] http://www.satcompetition.org/
[2] http://www.icaps-conference.org/index.php/Main/Competitions
[3] http://www.qbflib.org/index_eval.php

ISSN: 1137-3601 (print), 1988-3064 (on-line)

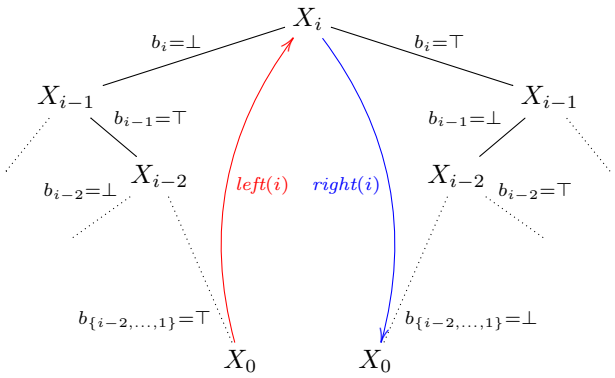


Figure 1: Both possible transitions in a CTE following the branching structure of a QBF: $X_0 \to X_i$ (from leaf to node on the left) and $X_i \to X_0$ (from node to leaf on the right). Note that i refers to any level (except for the leaf layer), not only the root.

to QBF solving such as counterexample guided abstraction refinement (CEGAR) [4, 10, 9, 21]. For comparable SAT / QBF encodings, the QBF approach also have the advantage to generate more compact formulas [3]. Even if the QBF approach is not as efficient as the SAT approach, it deserves the interest of the community.

Our paper shows that beyond the implementation of solvers, further work must be done to improve the encodings. In particular, we introduce two new QBF Compact Tree Encodings of STRIPS planning problems: CTE-EFA based on Explanatory frame axioms, and CTE-OPEN based on causal links. Then we compare both of them to [3] where CTE-NOOP based on No-op Actions is proposed. In terms of execution time over benchmark problems, CTE-EFA and CTE-OPEN always performed better than CTE-NOOP.

2 Planning as QBF

In [3], two different approaches of planning as QBF have been proposed: Flat Encoding, that was first introduced by [25] as an approach to general reachability, and Compact Tree Encoding (CTE). In [3], authors showed that Compact Tree Encodings outperform Flat Encodings. Both these planning encodings make use of the branching structure of the QBF to reuse a single set of clauses that describes a single step in the plan. The two assignments inside each universal variable represent the first and second half of the plan split around that branch. The assignments to each existential set represent action choices within a single step.

2.1 Preliminary Definitions

Let $\mathcal{F}$ be a finite set of *fluents* (atomic propositions). A STRIPS *planning problem* is a tuple $\langle I, \mathcal{A}, G\rangle$ where $I \subseteq \mathcal{F}$ is the set of initial fluents, $G \subseteq \mathcal{F}$ is the set of goal fluents and $\mathcal{A}$ is the set of actions. An action $a \in \mathcal{A}$ is a tuple $\langle Pre(a), Add(a), Del(a)\rangle$ where

- $Pre(a) \subseteq \mathcal{F}$ is the set of fluents required to be true in order to execute a,
- $Add(a) \subseteq \mathcal{F}$ and $Del(a) \subseteq \mathcal{F}$ are the sets of fluents respectively added and removed by the action a.

All QBF encodings studied in this paper use propositional variables for actions. The Compact Tree Encoding proposed in [3] is based on the *planning graph* introduced in [1] and uses additional no-op actions as frame axioms. We denote it by CTE-NOOP. Considering every action as a propositional variable, we define a set of propositional variables X, given by $X = \mathcal{A} \cup \{noop_f \mid f \in \mathcal{F}\}$.

In a CTE formula, we want to select two consecutive steps in order to define transitions (Figure 1). For each depth i of the tree, X_i denotes a copy of the set of variables X.

For CTE-NOOP, there exists a single variable $a_i \in X_i$ for each action and a single variable $noop_{f,i} \in X_i$ (no-op action) for each fluent used to determine a transition in the plan. At a same depth i, the value of these variables depends on the node (corresponding to a step in the plan) selected by the values of upper universal branching variables $b_{i+1} \ldots b_{depth}$. More details can be find in the slides[4].

An upper bound on the plan length is $2^{k+1} - 1$, where k is the number of alternations of quantifiers in the quantified boolean formula associated with the planning problem. In the case of CTE, k is also the compact tree

[4] `https://www.irit.fr/~Frederic.Maris/documents/coplas2018/slides.pdf`

depth. The number of possible states for a given planning problem is bounded by $2^{|\mathcal{F}|}$. Then, the existence of a plan can be determined using a linear QBF encoding with at most $k =\mid \mathcal{F} \mid$.

In the sequel, we propose two new encodings of planning problems into QBF. The first, denoted by CTE-OPEN, is based on causal links (plan-space). It has been introduced by [19] but needs to be adapted using additional variables for open conditions. The second, denoted by CTE-EFA, is based on explanatory frame axioms (state-space) first introduced by [12] and uses variables for fluents as well as for actions.

2.2 Causal Link Encoding: CTE-OPEN

The plan-space encodings of [19] cannot be directly adapted to the CTE. All these encodings refer to three indexed (not necessarily consecutive) steps of the plan. This is not possible in a CTE because each rule can refer to only one branch of the tree. To overcome this problem, it would be possible to duplicate the tree by adding, for each branching variable b_i, two more branching variables b'_i and b''_i, and for each node X_i, two node copies X'_i and X''_i, and equivalence rules $\bigwedge_{x_i \in X_i} \big((x_i \leftrightarrow x'_i) \wedge (x_i \leftrightarrow x''_i)\big)$. Unfortunately, this would increase the branching factor unnecessarily. So, we propose a new plan-space encoding which allows us to only refer to consecutive steps in the plan.

For every fluent $f \in \mathcal{F}$, we create a propositional variable $open_f$ to express that f holds in some previous step and must be protected at least until the current step. In Figure 2, the fluent f is an *open condition* in step S_i, entailing that either $f \in I$ or an action a' which adds f is executed in a previous step S_{i-k}. Open conditions are propagated backwards until the initial state or some step in which they are added by an action.

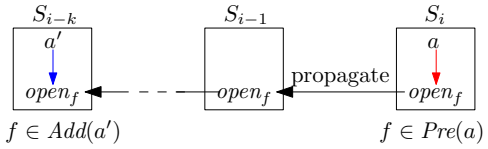


Figure 2: Causal link: a'produces f for a.

We define the set of "open" variables, denoted as Δ, as $\Delta = \{open_f \mid f \in \mathcal{F}\}$. Considering every action as a propositional variable, we define a set of propositional variables X, given by $X = \mathcal{A} \cup \Delta$.

Quantifiers For each depth i of the tree, X_i denotes a copy of the set of variables X. It exists a single variable $a_i \in X_i$ for each action used to determine last transition in the plan and a single variable $open_{f,i} \in X_i$ for each fluent used to determine if f is an open condition. At a same depth i, the value of these variables depends on the node (corresponding to a step in the plan) selected by the values of upper universal branching variables $b_{i+1} \dots b_{depth}$.

$$\begin{array}{l} \exists_{a \in \mathcal{A}}\, a_{depth}.\ \exists_{f \in \mathcal{F}}\, open_{f,depth}.\forall b_{depth}. \\ \exists_{a \in \mathcal{A}}\, a_{depth-1}.\ \exists_{f \in \mathcal{F}}\, open_{f,depth-1}.\forall b_{depth-1}. \\ \qquad \dots \\ \exists_{a \in \mathcal{A}}\, a_1.\ \exists_{f \in \mathcal{F}}\, open_{f,1}.\forall b_1.\ \exists_{a \in \mathcal{A}}\, a_0.\ \exists_{f \in \mathcal{F}}\, open_{f,0}. \end{array}$$

In the following, a *node* now refers to a non-leaf node (i.e., an inner node) and *depth* is the depth of the tree. The predecessor of a node at level i is the rightmost leaf of the left subtree. The successor of a node at level i is the leftmost leaf of the right subtree. In order to select these transitions, we introduce the leaf-to-node operator $left(i)$ defined as:

$$left(i) \equiv \neg b_i \wedge \bigwedge_{j=1}^{i-1} b_j.$$

Symmetrically, we introduce the node-to-leaf operator $right(i)$ defined as:

$$right(i) \equiv b_i \wedge \bigwedge_{j=1}^{i-1} \neg b_j.$$

Open conditions If an action a is executed in a step of the plan, then each precondition of a must be an open condition at this step (i.e., a causal link is required for this precondition).

$$\bigwedge_{i=0}^{depth} \bigwedge_{a \in \mathcal{A}} \left(a_i \Rightarrow \bigwedge_{f \in Pre(a)} open_{f,i} \right)$$

In the last plan step leading to the goal (i.e. the rightmost leaf of the tree), all the goal fluents must be either open conditions or added by actions executed in this step.

$$\bigwedge_{i=1}^{depth} b_i \Rightarrow \bigwedge_{f \in G} \left(open_{f,0} \vee \bigvee_{\substack{a \in \mathcal{A} \\ f \in Add(a)}} a_0 \right)$$

Propagate and close No conditions should remain open in the first plan step (i.e. the leftmost leaf of the tree) if it is not provided in the initial state.

$$\bigwedge_{i=1}^{depth} \neg b_i \Rightarrow \bigwedge_{f \in \mathcal{F} \setminus I} \neg open_{f,0}$$

Any open condition in a step must either remain open or be added (closed) by an action in the previous step.

$$\bigwedge_{i=1}^{depth} \bigwedge_{f \in \mathcal{F}} \left(\big(open_{f,i} \wedge left(i) \big) \Rightarrow \left(open_{f,0} \vee \bigvee_{\substack{a \in \mathcal{A} \\ f \in Add(a)}} a_0 \right) \right)$$

$$\bigwedge_{i=1}^{depth} \bigwedge_{f \in \mathcal{F}} \left(\big(open_{f,0} \wedge right(i) \big) \Rightarrow \left(open_{f,i} \vee \bigvee_{\substack{a \in \mathcal{A} \\ f \in Add(a)}} a_i \right) \right)$$

Protect open conditions An open condition in a given step cannot be removed in the previous step. This guarantees not to break any causal link in the plan.

$$\bigwedge_{i=1}^{depth} \bigwedge_{f \in \mathcal{F}} \left(\big(open_{f,i} \wedge left(i) \big) \Rightarrow \bigwedge_{\substack{a \in \mathcal{A} \\ f \in Del(a)}} \neg a_0 \right)$$

$$\bigwedge_{i=1}^{depth} \bigwedge_{f \in \mathcal{F}} \left(\big(open_{f,0} \wedge right(i) \big) \Rightarrow \bigwedge_{\substack{a \in \mathcal{A} \\ f \in Del(a)}} \neg a_i \right)$$

Prevent negative interactions In a given step, if an action removes a fluent which is needed or added by another action, then these two actions cannot be both executed in this step.

$$\bigwedge_{i=0}^{depth} \bigwedge_{a \in \mathcal{A}} \bigwedge_{f \in (Add(a) \cup Pre(a))} \bigwedge_{\substack{a' \in \mathcal{A} \\ a \neq a' \\ f \in Del(a')}} \left(\neg a_i \vee \neg a'_i \right)$$

2.3 State-Space Encoding: CTE-EFA

In this encoding, we define the set of propositional variables as $X = \mathcal{A} \cup \mathcal{F}$. Each step is now defined by a transition (as in CTE-OPEN) as well as the resulting state (valuation of the fluents in $\mathcal{F}$). The formula is an adaptation to the CTE of the well known state-space SAT encoding rules based on explanatory frame axioms of [12].

Quantifiers At each depth i of the tree, it exists a single variable a_i for each action used to determine last transition in the plan and a single variable f_i for each fluent used to determine the state. At a same depth i , the values of these variables depend on the node (corresponding to a transition in the plan and the resulting state) selected by the values of upper universal branching variables $b_{i+1} \dots b_{depth}$.

$$\begin{array}{c} \exists\limits_{a\in\mathcal{A}} a_{depth}.\ \exists\limits_{f\in\mathcal{F}} f_{depth}.\forall b_{depth}. \\ \exists\limits_{a\in\mathcal{A}} a_{depth-1}.\ \exists\limits_{f\in\mathcal{F}} f_{depth-1}.\forall b_{depth-1}. \\ \dots \\ \exists\limits_{a\in\mathcal{A}} a_1.\ \exists\limits_{f\in\mathcal{F}} f_1.\forall b_1.\ \exists\limits_{a\in\mathcal{A}} a_0.\ \exists\limits_{f\in\mathcal{F}} f_0. \end{array}$$

Goal In the state after the last plan transition (i.e. the rightmost leaf of the tree), all goal fluents must be achieved.

$$\bigwedge_{i=1}^{depth} b_i \Rightarrow \bigwedge_{f\in G} f_0$$

Conditions and effects of actions If an action a is executed in a transition of the plan, then each effect of a occurs in the resulting state and each condition of a is required in the previous state.

$$\bigwedge_{i=0}^{depth} \bigwedge_{a\in\mathcal{A}} \left(a_i \Rightarrow \left(\bigwedge_{f\in Add(a)} f_i \right) \wedge \left(\bigwedge_{f\in Del(a)} \neg f_i \right) \right)$$

$$\bigwedge_{i=1}^{depth} \bigwedge_{a\in\mathcal{A}} \left(a_i \wedge \mathit{left}(i) \Rightarrow \bigwedge_{f\in Pre(\mathcal{A})} f_0 \right)$$

$$\bigwedge_{i=1}^{depth} \bigwedge_{a\in\mathcal{A}} \left(a_0 \wedge \mathit{right}(i) \Rightarrow \bigwedge_{f\in Pre(\mathcal{A})} f_i \right)$$

Moreover, an action which do not have all conditions in initial state cannot be executed in the first plan transition (i.e. the leftmost leaf of the tree):

$$\bigwedge_{i=1}^{depth} \neg b_i \Rightarrow \bigwedge_{\substack{a\in\mathcal{A} \\ Pre(a)\not\subset I}} \neg a_0$$

Explanatory frame axioms If the value of a fluent changes between two consecutive states, then an action which produces this change is executed in the plan transition between these states.

$$\bigwedge_{i=1}^{depth} \bigwedge_{f\in\mathcal{F}} \left((\neg f_0 \wedge f_i \wedge \mathit{left}(i)) \Rightarrow \left(\bigvee_{\substack{a\in\mathcal{A} \\ f\in Add(a)}} a_i \right) \right)$$

$$\bigwedge_{i=1}^{depth} \bigwedge_{f\in\mathcal{F}} \left((\neg f_i \wedge f_0 \wedge \mathit{right}(i)) \Rightarrow \left(\bigvee_{\substack{a\in\mathcal{A} \\ f\in Add(a)}} a_0 \right) \right)$$

$$\bigwedge_{i=1}^{depth} \bigwedge_{f\in\mathcal{F}} \left((f_0 \wedge \neg f_i \wedge \mathit{left}(i)) \Rightarrow \left(\bigvee_{\substack{a\in\mathcal{A} \\ f\in Del(a)}} a_i \right) \right)$$

$$\bigwedge_{i=1}^{depth} \bigwedge_{f \in \mathcal{F}} \left((f_i \wedge \neg f_0 \wedge right(i)) \Rightarrow \left(\bigvee_{\substack{a \in \mathcal{A} \\ f \in Del(a)}} a_0 \right) \right)$$

An extra rule is also required to describe explanatory frame axioms for the first plan transition from initial state (i.e. the leftmost leaf of the tree):

$$\bigwedge_{f \in \mathcal{F} \setminus I} \left(\left(f_0 \wedge \bigwedge_{i=1}^{depth} \neg b_i \right) \Rightarrow \bigvee_{\substack{a \in \mathcal{A} \\ f \in Add(a) \\ Pre(a) \subset I}} a_0 \right)$$

$$\bigwedge_{f \in I} \left(\left(\neg f_0 \wedge \bigwedge_{i=1}^{depth} \neg b_i \right) \Rightarrow \bigvee_{\substack{a \in \mathcal{A} \\ f \in Del(a) \\ Pre(a) \subset I}} a_0 \right)$$

Prevent negative interactions Unlike in CTE-NOOP and CTE-OPEN, contradictory effects are already disallowed by previous rules (effects of actions). Then, this rule only need to prevent interactions between conditions and deletes of actions. If an action removes a fluent which is needed by another action, then these two actions cannot be both executed in a same plan transition.

$$\bigwedge_{i=0}^{depth} \bigwedge_{a \in \mathcal{A}} \bigwedge_{f \in Pre(a)} \bigwedge_{\substack{a' \in \mathcal{A} \\ a \neq a' \\ f \in Del(a')}} (\neg a_i \vee \neg a'_i)$$

3 Experimental Trials

To compare these three encodings on a same basis we used our translator TouIST[5] [5] that can use several QBF solvers. We ran all available STRIPS IPC benchmarks (1 through 8, except for the 7th which was not available and authors did not answer) on an Intel Xeon CPU E7-8890 v4 @ 2.20GHz, 512 GB of RAM. The domains tested include Gripper, Logistics, Mystery, Blocks, Elevator, Depots, DriverLog, ZenoTravel, FreeCell, Airport, Pipesworld-NoTankage, Pipesworld-Tankage, PSR, Satellite, OpenStacks, Pathways, Rovers, Storage, TPP, Trucks, ChildSnack, Hiking, VisitAll and the non-IPC Ferry.

We tried to consider as many QBF solvers as possible using the QBFEval 2017 as a reference. Qute (version of 2017-07-09, based on dependency learning QCDCL) and CaQE (version of 2017-07-08, based on CEGAR clausal abstraction) were not able to give a valuation for the outer existential quantifier. AIGSolve and Qell weren't available for download. GhostQ was skipped (but we should have included it). DepQBF (version 6.03 of 2017-08-02, based on "generalized Q-resolution", described in [17]) and RAReQS (version 1.1 of 2013-05-07, based on a CEGAR approach, detailed in [8]) were the only solvers left. RAReQS was consistently twice as fast as DepQBF, we thus dismissed DepQBF and only shown results for RAReQS. Finally, we did not apply any QBF preprocessor (e.g., Bloqqer).

[5] `https://www.irit.fr/touist`

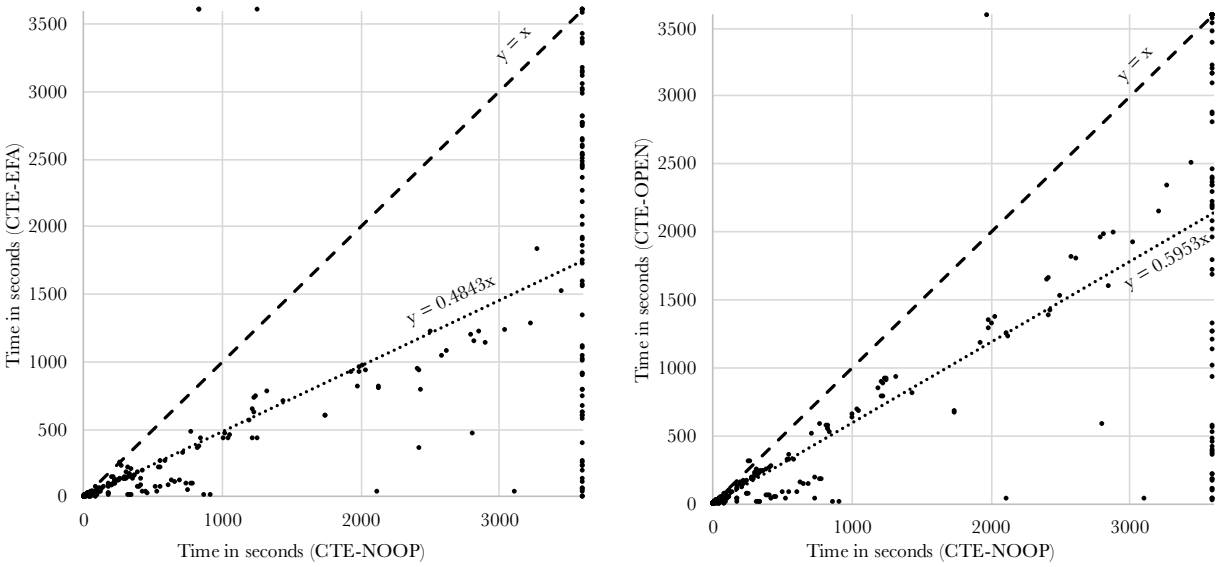


Figure 3: Plan decision time (EFA/OPEN vs NOOP).

Encoding	Solved problems	Decision Time	Literals	Clauses	Transitions-over-nodes ratio
CTE-NOOP	412 over 2112 (20%)	0%	0%	0%	30%
CTE-EFA	463 over 2112 (22%)	-55%	-26%	+15%	47%
CTE-OPEN	445 over 2112 (21%)	-41%	-2%	-28%	17%

Table 1: Comparison of the presented encodings across 65 STRIPS domains from IPC 1 through 8 (IPC 7 excepted) with a total of 2112 problems. Decision time, literals count, clauses count and the transitions-over-nodes ratio are averages. The transitions-over-nodes ratio measures the quantity (in average) of transition-based constraints over node-based constraints.

We ran these benchmarks using our new encodings CTE-EFA and CTE-OPEN as well as the state-of-the-art compact tree encoding (CTE-NOOP). We compared them two-by-two by considering the time needed to prove the existence of a plan (decision time, Figure 3) and the overall time required to obtain a plan (extraction time, Figure 5). The "decision" step consists of launching incrementally the QBF solver on a CTE of increasing depth until the solver returns true or reaches the upper bound (total number of fluents). The "extraction" step consists of one solver launch per node of the tree in order to retrieve the plan. Each experiment had a 60 minutes[6] timeout for searching the plan and 60 minutes for extracting it. The benchmark results are available as an Excel file[7].

The results show that our encodings CTE-EFA and CTE-OPEN are more efficient than CTE-NOOP both in plan existence as well as in plan extraction. CTE-EFA by a factor of 2.1 (1/0.4843) and CTE-OPEN by a factor of 1.7 (1/0.5953). Also, the comparison between CTE-EFA and CTE-OPEN (Figure 4, Figure 6) consistently shows that CTE-EFA outperforms CTE-OPEN by a factor of 1.4 (1/0.7266). Table 1 gives a summary of the benchmark results.

Contrary to what happens with flat encodings, the gain over CTE-NOOP cannot be explained by the difference in quantifier alternations as the depth is the same in the three encodings. However, the way actions are represented in these encodings may explain this difference.

4 Discussion

In order to identify the source of these improvements, we propose two hypothesis:

[6]The grounding step (i.e., action instantiation) is not included in the elapsed time.

[7]`https://www.irit.fr/~Frederic.Maris/documents/coplas2018/results.xls`

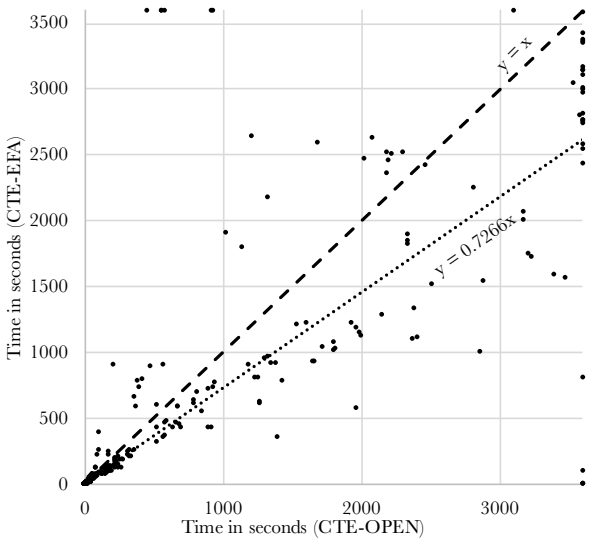


Figure 4: Plan decision time (EFA vs OPEN).

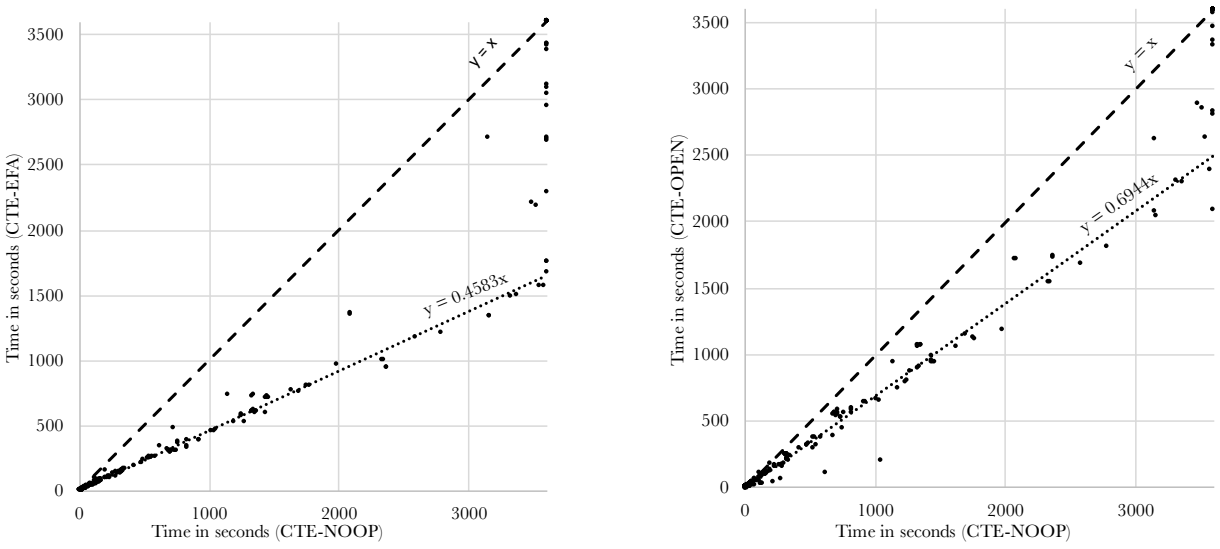


Figure 5: Plan extraction time (EFA/OPEN vs NOOP).

Hypothesis 1 "The performance gain is correlated to a decrease in the number of clauses and/or literals across encodings". Although the size of the problem is known to be noticeably non-correlated with its hardness in SAT, we wondered if we could see the same non-correlation. As shown in Table 1, we do not observe any clear tendency: CTE-EFA tends to have a slightly higher number of clauses (+15%) than CTE-NOOP although having less variables (-26%). CTE-OPEN has the same number of literals and much less clauses than CTE-NOOP, but resulting in a lower performance gain (-41%) than CTE-EFA (-55%). This non-correlation leads us to reject this hypothesis.

Hypothesis 2 "The performance gain is due to a difference in the number of transition-based constraints compared to the number of node-based constraints". Intuitively, one can think that a lower ratio of transition-based constraints over node-based constraints would ease the solving process: in node-based constraints, a clause has the same context[8] across the whole QBF expansion. In branch constraints, the corresponding

[8]Context and expansion are defined in [3]. Intuitively, the expansion is a tree representing the QBF and a context is a leaf in that tree.

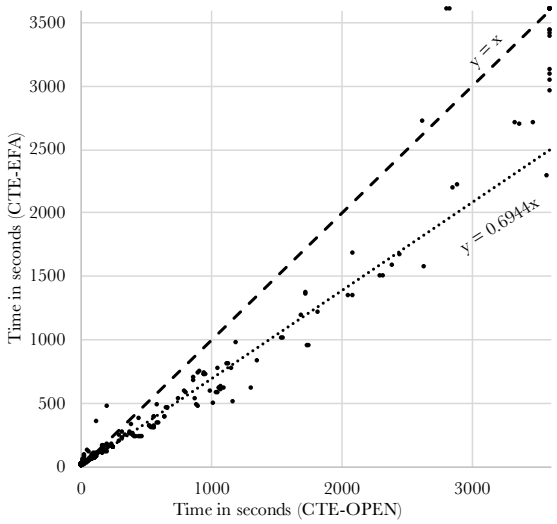


Figure 6: Plan extraction time (EFA vs OPEN).

clause has different contexts depending on the selected branch. The idea is that clauses based on different contexts slow the solver down. As displayed in Table 1, this hypothesis does not appear to be correct experimentally: although CTE-OPEN shows a lower transitions-to-nodes ration, it does not lead to the best performance gain. On the contrary, CTE-EFA has a poorer ratio, although being the most efficient compared to CTE-NOOP. We thus refute this hypothesis as we did not see any noticeable correlation supporting it, although we noticed a slight tendency where the decrease of time and of number of clauses were correlated.

Through these hypotheses, we tried to understand the causes of these enhancements. None of these hypothesis proved to be useful.

5 Conclusion

We have proposed two new QBF Compact Tree Encodings: CTE-OPEN based on causal links (plan-space) and CTE-EFA based on explanatory frame axioms (state-space). We compared these encodings with the state-of-the-art QBF encoding CTE-NOOP. In average over all available STRIPS IPC benchmarks, CTE-EFA performed twice as fast as CTE-NOOP (respectively 1.7 times faster for CTE-OPEN).

Through experiments, we refuted the two hypotheses we had formulated in order to explain the causes of this enhancement: neither the difference in the number of literals and clauses nor the transitions-to-nodes ratio between the three encodings allow us to draw conclusions.

Although it is fair to say that the work we are presenting lacks explanations on the reasons for the gain in performance, we think that this paper aims at showing the interest of systematically studying the statistical properties of the various action representations in encodings in order to understand the ontological choices related to these action representations.

Furthermore, it is noticeable that the performance ranking of the various action representations of SAT encodings (e.g., No-op performs better than EFA) is different in QBF (as we showed, EFA performs better than No-op in the CTE encoding). It would be interesting to study more broadly the methods used in SAT for encoding actions and see how their QBF counterpart behave.

References

[1] A. Blum and M. Furst. Fast planning through planning-graphs analysis. *Artificial Intelligence*, 90(1-2):281–300, 1997.

[2] Tom Bylander. The computational complexity of propositional STRIPS planning. *Artif. Intell.*, 69(1-2):165–204, 1994.

[3] Michael Cashmore, Maria Fox, and Enrico Giunchiglia. Planning as quantified boolean formula. In Luc De Raedt, Christian Bessière, Didier Dubois, Patrick Doherty, Paolo Frasconi, Fredrik Heintz, and Peter J. F. Lucas, editors, *ECAI 2012 - 20th European Conference on Artificial Intelligence. Including Prestigious Applications of Artificial Intelligence (PAIS-2012) System Demonstrations Track, Montpellier, France, August 27-31 , 2012*, volume 242 of *Frontiers in Artificial Intelligence and Applications*, pages 217–222. IOS Press, 2012.

[4] Edmund M. Clarke, Orna Grumberg, Somesh Jha, Yuan Lu, and Helmut Veith. Counterexample-guided abstraction refinement for symbolic model checking. *J. ACM*, 50(5):752–794, 2003.

[5] Alexis Comte, Olivier Gasquet, Abdelwahab Heba, Olivier Lezaud, Frederic Maris, Khaled Skander-Ben-Slimane, and Mael Valais. Twist your logic with touist. *CoRR*, abs/1507.03663, 2015.

[6] Minh Binh Do and Subbarao Kambhampati. Planning as constraint satisfaction: Solving the planning graph by compiling it into CSP. *Artif. Intell.*, 132(2):151–182, 2001.

[7] M. Ernst, T. Millstein, and D.S. Weld. Automatic SAT-compilation of planning problems. In *Proc. IJCAI-97*, 1997.

[8] Mikolás Janota, William Klieber, João Marques-Silva, and Edmund M. Clarke. Solving QBF with counterexample guided refinement. In Alessandro Cimatti and Roberto Sebastiani, editors, *Theory and Applications of Satisfiability Testing - SAT 2012 - 15th International Conference, Trento, Italy, June 17-20, 2012. Proceedings*, volume 7317 of *Lecture Notes in Computer Science*, pages 114–128. Springer, 2012.

[9] Mikolás Janota, William Klieber, Joao Marques-Silva, and Edmund M. Clarke. Solving QBF with counterexample guided refinement. *Artif. Intell.*, 234:1–25, 2016.

[10] Mikolás Janota and Joao Marques-Silva. Solving QBF by clause selection. In Qiang Yang and Michael Wooldridge, editors, *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015*, pages 325–331. AAAI Press, 2015.

[11] H. Kautz. Satplan04: Planning as satisfiability. In *Abstracts of the 4th International Planning Competition, IPC-04*, 2004.

[12] H. Kautz and B. Selman. Planning as satisfiability. In *ECAI-92*, pages 359–363, 1992.

[13] H. Kautz and B. Selman. Pushing the envelope: Planning, propositional logic and stochastic search. In *Proc. AAAI-96*, pages 1194–1201, 1996.

[14] H. Kautz and B. Selman. BLACKBOX: A new approach to the application of theorem proving to problem solving. In *Proceedings of AIPS-98 Workshop on Planning as Combinatorial Search*, 1998.

[15] H. Kautz and B. Selman. Unifying SAT-based and Graph-based planning. In *Proc. IJCAI-99*, pages 318–325, 1999.

[16] H. Kautz, B. Selman, and J. Hoffmann. Satplan04: Planning as satisfiability. In *Abstracts of the 5th International Planning Competition, IPC-06*, 2006.

[17] Florian Lonsing and Uwe Egly. Depqbf 6.0: A search-based QBF solver beyond traditional QCDCL. In Leonardo de Moura, editor, *Automated Deduction - CADE 26 - 26th International Conference on Automated Deduction, Gothenburg, Sweden, August 6-11, 2017, Proceedings*, volume 10395 of *Lecture Notes in Computer Science*, pages 371–384. Springer, 2017.

[18] A. Mali and S. Kambhampati. Refinement-based planning as satisfiability. In *Proc. Workshop planning as combinatorial search, AIPS-98*, 1998.

[19] A. Mali and S. Kambhampati. On the utility of plan-space (causal) encodings. In *Proc. AAAI-99*, pages 557–563, 1999.

[20] Frederic Maris and Pierre Régnier. TLP-GP: new results on temporally-expressive planning benchmarks. In *(ICTAI 2008), Vol. 1*, pages 507–514, 2008.

[21] Markus N. Rabe and Leander Tentrup. CAQE: A certifying QBF solver. In Roope Kaivola and Thomas Wahl, editors, *Formal Methods in Computer-Aided Design, FMCAD 2015, Austin, Texas, USA, September 27-30, 2015.*, pages 136–143. IEEE, 2015.

[22] J. Rintanen. Symmetry reduction for sat representations of transition systems. In *Proc. ICAPS-03*, 2003.

[23] J. Rintanen, K. Heljanko, and Niemelä. Parallel encodings of classical planning as satisfiability. In *Proc. European Conference on Logics in Artificial Intelligence, JELIA-04*, 2004.

[24] J. Rintanen, K. Heljanko, and Niemelä. Planning as satisfiability: parallel plans and algorithms for plan search. *Artificial Intelligence*, 170(1213):1031–1080, 2006.

[25] Jussi Rintanen. Partial implicit unfolding in the davis-putnam procedure for quantified boolean formulae. In *Logic for Programming, Artificial Intelligence, and Reasoning, 8th International Conference, LPAR 2001, Havana, Cuba, December 3-7, 2001, Proceedings*, pages 362–376, 2001.

[26] Jussi Rintanen. Asymptotically optimal encodings of conformant planning in QBF. In *Proceedings of the Twenty-Second AAAI Conference on Artificial Intelligence, July 22-26, 2007, Vancouver, British Columbia, Canada*, pages 1045–1050, 2007.

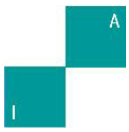
[27] Jussi Rintanen. Discretization of temporal models with application to planning with SMT. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, January 25-30, 2015, Austin, Texas, USA.*, pages 3349–3355, 2015.

[28] Jussi Rintanen, Bernhard Nebel, J. Christopher Beck, and Eric A. Hansen, editors. *Proceedings of the Eighteenth International Conference on Automated Planning and Scheduling, ICAPS 2008, Sydney, Australia, September 14-18, 2008*. AAAI, 2008.

[29] Ji-Ae Shin and Ernest Davis. Processes and continuous change in a sat-based planner. *Artif. Intell.*, 166(1-2):194–253, 2005.

[30] Steven A. Wolfman and Daniel S. Weld. The LPSAT engine & its application to resource planning. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence, IJCAI 99, Stockholm, Sweden, July 31 - August 6, 1999. 2 Volumes, 1450 pages*, pages 310–317, 1999.

Inteligencia Artificial, 21(62) (2018), 114-133
doi: 10.4114/intartif.vol21iss62pp114-133



INTELIGENCIA ARTIFICIAL

http://journal.iberamia.org/

Fuzzy Neural Networks based on Fuzzy Logic Neurons Regularized by Resampling Techniques and Regularization Theory for Regression Problems

Paulo Vitor de Campos Souza[1,2], Augusto Junio Guimarães[2], Vanessa Souza Araújo[2], Thiago Silva Rezende[2], Vinicius Jonathan Silva Araújo[2]

[1]Federal Center for Technological Education of Minas Gerais, Faculty UNA of Betim, Minas Gerais Brazil.
goldenpaul@informatica.esp.ufmg.br, pauloc@prof.una.br

[2]Faculty UNA of Betim Minas Gerais, Brazil
augustojunioguimaraes@gmail.com, v.souzaaraujo@yahoo.com.br, silvarezendethiago@gmail.com,vinicius.j.s.a22@hotmail.com

Abstract This paper presents a novel learning algorithm for fuzzy logic neuron based on neural networks and fuzzy systems able to generate accurate and transparent models. The learning algorithm is based on ideas from Extreme Learning Machine, to achieve a low time complexity, and regularization theory, resulting in sparse and accurate models. A compact set of incomplete fuzzy rules can be extracted from the resulting network topology. Experiments considering regression problems are detailed. Results suggest the proposed approach as a promising alternative for pattern recognition with good accuracy and some level of interpretability, in addition to a more compact network architecture to work with regression problems.

Keywords: Bootstrap Lasso, Extreme Learning Machine, Fuzzy Logic Neurons, Fuzzy Neural Networks, Regression Problems.

1 Introduction

Fuzzy neural networks are networks based on fuzzy logic neurons [50]. These models are a synergy between fuzzy sets theory, as a mechanism for knowledge representation and information compaction, and neural networks. The main feature of these models is transparency since a set of fuzzy rules can be extracted from the network structure after training [10]. Thus, the neural network is now seen as a linguistic system, preserving the learning capacity of ANN. Its fuzzy neurons are composed of triangular norms, which generalize the union and intersection operations of classical clusters to the theory of fuzzy sets. Furthermore, these models have a neural network topology which enables the utilization of a large variety of existing machine learning algorithms for structure identification and parameter estimation. Fuzzy neural networks have already been used to solve several distinct problems including pattern classification [10] and [60], time series prediction [4] [27] [8] and dynamic system modeling [26] [19] [41] [42]. Examples of fuzzy neurons are and and or neurons [54]. These logic based neurons are nonlinear mappings of the form $[0, 1]^N \rightarrow [0, 1]$, where N is the number of inputs. The processing of these neurons is performed in two steps. Firstly, the input signals are individually combined with connection weights. Next, an overall aggregation operation is performed over the results obtained in the first step and and or neurons use t-norms and s-norms (t-conorms) to performed output processing. Several learning algorithms for fuzzy neural networks have already been proposed in the literature. Usually, learning is performed in two steps. Firstly, the network topology is defined. This step involves defining fuzzy sets for each input variable, selecting a suitable number

ISSN: 1137-3601 (print), 1988-3064 (on-line)

of neurons and defining network connections. The most commonly used methods for structure definition are clustering [10], [4], [8], [26], [41], [42] and evolutionary optimization [51] [52] [43]. Once the network structure is defined, free parameters are estimated. A number of distinct methods have already been used in this step including reinforcement learning [10], [4], [26], gradient-based methods [52], [53], genetic algorithms [41], least squares [8], [42] and hybrid strategy [60] through a grid partition of the data (without grouping) [37]. Regarding network structure optimization, the two most commonly used methods may present significant deficiencies. Clustering has a low computational cost when compared with evolutionary optimization. Nevertheless, interpretable fuzzy rules usually canât be extracted from the resulting network, since fuzzy sets generated by clustering are typically challenging to be interpreted [21]. Evolutionary optimization based methods may be able to generate interpretable fuzzy rules. However, they have a high computational complexity. This paper proposes a novel learning algorithm for fuzzy neural networks able to generate compact and accurate models. The learning is performed using ideas from Extreme Learning Machine [34], to speed-up parameter tuning, and regularization theory [18]. First, fuzzy sets are generated for each input variable. Next, a large number of candidate fuzzy neurons are created with randomly assigned weights. In this step, only a random fraction of the input variables are used in each candidate neuron. Following, the bootstrap Lasso algorithm [3] is used to define the network topology by selecting a subset of the candidate neurons. Finally, the remaining network parameters are estimated through least squares. The resulting network is a sparse model and can be expressed as a compact set of incomplete fuzzy rules, that is, rules with antecedents defined using only a fraction of the available input variables [21]. A similar approach was used for classification problems [60], but taking advantage of the universal approximation capability of ELM [36], the model was adapted to perform resolutions for regression problems. The paper is organized as follows. Next section reviews the necessary fundamental concepts about fuzzy logic neurons and fuzzy neural networks. Section III details the proposed new learning algorithm able to generate small and transparent networks. Section IV presents the experimental results for regression problems and comparison with alternative classifiers. Finally, the conclusions and further developments are summarized in section V.

2 Fuzzy Neural Networks

2.1 Neural Networks

A neural network is a distributed, and parallel processor made up of less complicated processing units, in order to store knowledge about a theme or a set of characteristics, making it available for use in similar tasks human beings [25]. Already [9] complements that the neural network can be connected entirely where each neuron is connected to all the neurons of the next layer, partially connected where each neuron is, or locally connected where there is a partial connection oriented for each type of functionality. To perform the training of a neural network requires a set of data that contains patterns for training and desired outputs. As artificial neural networks seek to simulate the behavior of the human brain, knowledge is acquired by the network through the environment that surrounds it, through a learning process. Already in the storage of knowledge, the analogy is made with the forces of connections between the neurons, known as synaptic weights.

2.2 Fuzzy Systems

The fuzzy systems are based on fuzzy logic, developed by [62]. His work was motivated by a wide variety of vague and uncertain information in human decision making. Some problems cannot be solved with standard Boolean logic. The fuzzy model's systems can be divided into three main concepts [28]:
-Fuzzy Linguistic Models;
-Relational Fuzzy Models;
-Functional Fuzzy Models.
The fuzzy linguistic models are those that use a rule base if-then and an inference method for relating inputs and outputs. This model is called linguistic to make direct use of the linguistic representation of rules [28]. In turn, the nebulous relational models, as the name itself says, represent the mapping between the fuzzy sets of input and output through nebulous relations [54]. Moreover, finally, the fuzzy functional models (Takagi-Sugeno) are models with terms in the antecedent of a rule and a function of the entrances in the consequent.

2.3 Fuzzy logic Neurons

Fuzzy logic neurons are functional units able to perform multivariate nonlinear operations in unit hypercube [0, 1]^N [0, 1] [50], where N is the number of inputs. The term logic is associated with the logic disjunction or and conjunction and operations performed by these neurons using, in this work, t-norms and s-norms. The *or* and *and* neurons aggregates input signals $\mathbf{a} = [a_1, a_2, ..., a_N\ ]$ by firstly combining them individually with the weights

$\mathbf{w} = [w_1, w_2, ..., w_N]$, where $a_j \in [0, 1]$ and $w_j \in [0, 1]$ for $j = 1, ... , N$, and subsequently globally aggregating these results. They were initially defined as follows [50]:

$$h = AND(a; w) = T_i^n{}_{=}1(a_i s w_i) \tag{1}$$

$$h = OR(a; w) = S_i^n{}_{=}1(a_i t w_i) \tag{2}$$

where S and s are s-norm and T and are t-norm.

The or-neuron is interpreted as a logical expression that performs an and-type local aggregation of the inputs and the weights using a t-norm, followed by an or-type global aggregation of the results using as s-norm. The and-neuron is interpreted similarly, with an or-type local aggregation and an and-type global aggregation. Figure 1 illustrates the neurons, as defined in [50].

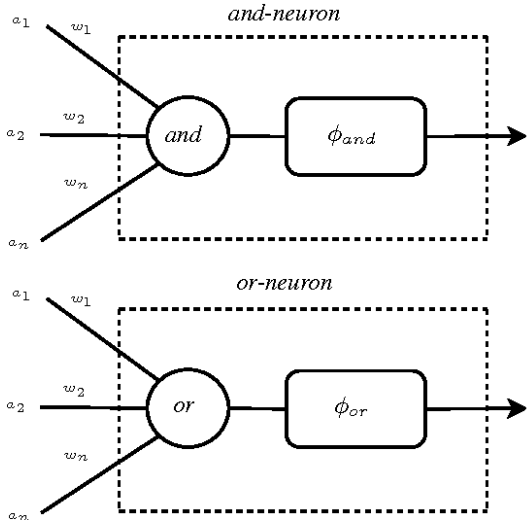


Figure 1: Fuzzy Logic Neurons.

The activation functions $\phi$and and ϕor can, in general, be nonlinear mappings. In this paper $\phi$and $(\varepsilon) = \phi$or $(\varepsilon) = \varepsilon$, i.e., they are defined as the identity function. According to [51], the local aggregations performed by these neurons can be interpreted as weighting operations of the inputs, since the role of the weights is to differentiate among distinct levels of impact that individual inputs might have on the global aggregation result. In the case of the or-neuron, lower values of wj reduces the impact of the corresponding input, while higher values do not affect the original value of the corresponding input. In limit, if all weights are set to 1, the neuron output is a plain or combination of the inputs. For the and-neuron, the interpretation of the weightâs values is inverse, i.e., higher values of w_j reduces the impact of the corresponding input. For this neuron, in the limit, if all weights are set to 0, the output is a plain and combination of the inputs. In order to unify the interpretation of the weights for the and and or neurons, (i.e., a low value of $\mathbf{w}$ reduces the impact of the associated input) the and-neuron used in this paper is defined as:

$$h = AND(a; w) = T_i^n{}_{=}1(a_i s(1 - w)_i) \tag{3}$$

2.4 Fuzzy Neural Networks concepts

Fuzzy neural networks use the structure of an artificial neural network, where artificial neurons are replaced by classics fuzzy neurons [50]. These neurons are implemented by triangular norms that generalize the operations of union and intersection of sets classics for the theory of fuzzy sets. Thus, the neural network is now seen as a linguistic system, preserving the learnability of RNA [26]. They provide a network with topology and allows the use of a wide variety of learning processes with the database. The main characteristic of these networks is your transparency, allowing the use of a priori information to define the initial network topology and allowing the extraction of valuable information from the resulting topology after training in the form of a set of fuzzy rules [26].

2.4.1 Characteristics of Fuzzy Neural Networks

Fuzzy neural networks can be classified concerning how their neurons are connected. This form of connection sets as the signals will be transmitted on the network. In general, there is the feedforward where fuzzy neurons are grouped into layers and the signal across the network in a single direction, usually the model until your output generating an expected result. Fuzzy neurons in the same layer that have no connection and its networks are also known as no feedback networks [24]. This type of connection is the most common among the models of fuzzy neural networks [8], [41] and [42]. Finally, there are the networks that are used with feedback, also called recurring (recurrent). In this type of fuzzy network, neurons are also assembled in layers, but there is information power in neurons in the same layer, and it may even happen to own fuzzy neuron in question, or even previous layers, if former. Figure 2 shows plainly the difference between a recurrent network and a forward network.

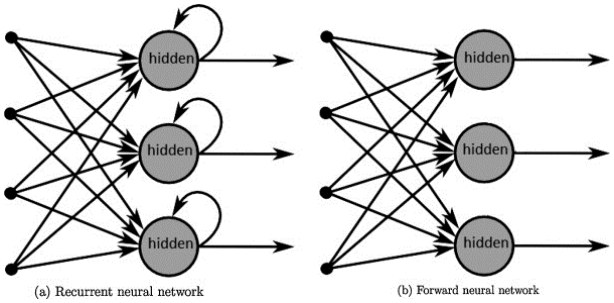


Figure 2: Example of networks feedforward and networks recurrent. [49]

The number of layers is also a factor that can differentiate fuzzy neural networks. In the models, each layer is responsible for a specific task or function. In general, the first layer is responsible for dealing with and the last inputs for bringing the network response. Between these two layers, there are other intermediates, that can be hidden or not. Depending on the model and what he proposes, each layer has a specific function. Examples of the fuzzy neural networks that have three layers in some models the first two are based on concepts of fuzzy systems and the third is an artificial neural network. The models of [26], [41] and [12] using fuzzy neurons (unineurons) to aggregate the values of the first layer. When we evaluate the networks on the types of neurons that compose their layers, we can highlight the models that have fuzzy neurons and/or type. These models generally use these structures to use operators of t-norm and s-norms. As an example of fuzzy neural networks that use these types of neuron the models [60], [55], [38] and [14]. Evaluating the type of fuzzy neural networks training can highlight that these algorithms are a set of well-defined rules for resolving learning problems. These methodologies are seeking training, in General, to simulate human learning to learn or refresh their new concepts, working mainly in network factors, for example, the values of the synaptic weights. In General, both for artificial neural networks and fuzzy neural networks learning can be classified according to the following paradigms:

-*Supervised*-when using some external factor that indicates to the network which is the desired result to the problem in question.

-*Unsupervised*-when no external agent is indicating the network response input standards submitted to the models.

-*Strengthening*-when an external evaluator unit measures the response of the model [24].

A model employee for training and upgrading of network elements involved in the network are the extreme learning machine (ELM) [34], which has been shown to be a fast and precise method for parameter adjustment, being used even by several models of different networks [42], [7] and [59]. In addition to these concepts, genetic algorithms can be used to perform network training. The model proposed by [41] uses a genetic algorithm called real-coded genetic algorithm [29], finding the model parameters that minimize the extent of the error mean square using the training database to update critical variables of a network. Each is generated by genetic algorithm encodes the following network attributes: dispersion of the pertinence of fuzzy Gaussian; weights that link the first and second layer; identity elements of the unineurons of the second layer; singletons fuzzy [41]. In the fuzzy neural network [55] training involves iterative adjustment of system parameters using a hybrid learning procedure to map each vector to your training target output vector with minimum error value quadratic. The neuro-fuzzy inference system Adaptive trained is used to process all the vectors. The way inputs are addressed in

the fuzzy neural networks also have featured, because the way they act reflects directly on the variables involved in the model. The method with which the entries are handled can set parameters such as activation functions, membership functions of fuzzy sets, network topology, among others [41]. Fuzzy neural networks can use grouping methods, such as the c-means clustering and your fuzzy version: fuzzy c-means [15], [5], density-based methods of data (clouds) [2], online group (eClustering [1]), the ePL [44], F-scores [13], and ANFIS approach [60], [12] and [14], [11] among others.

2.5 Fuzzy Neural Networks architecture proposes

The fuzzy logic neurons described in the previous section can be used to construct fuzzy neural networks and solve pattern recognition problems. Figure 3 illustrates the feed-forward topology of the fuzzy neural networks considered in this paper.

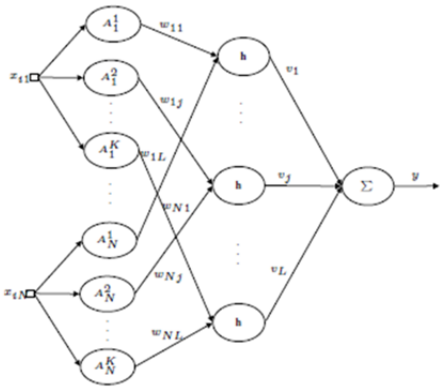


Figure 3: Feedforward fuzzy neural network

The first layer is composed of neurons whose activation functions are membership functions of fuzzy sets defined for the input variables. For each input variable x_{ij}, K fuzzy sets are defined A^k, $k = 1...\ K$ whose membership functions are the activation functions of the corresponding neurons. Thus, the outputs of the first layer are the membership degrees associated with the input values, i.e., $a_{jk} = \mu_{A^k}$. for $j = 1...$, N and $k = 1$, ... , K, where N is the number of inputs and K is the number of the membership functions of each input. For this, the ANFIS algorithm [37] is used.All the neurons of the first layer are of the Gaussian type, formed through the centers obtained by the functions of Gaussian pertinence and with the value of sigma defined in a random way.

The second layer is composed of L fuzzy logic neurons. Each neuron performs a weighted aggregation of some of the first layer outputs. This aggregation is performed using the weights w_{il} (for $i = 1...\ N$ and $l = 1...\ L$). For each input variable j, only one first layer output a_j^k is defined as the input of the l-th neuron. Furthermore, in favor of generating sparse topologies, each second layer neuron is associated with only $nl \leq N$ input variables, that is, the weight matrix $\mathbf{w}$ is sparse. Finally, the output layer uses a classic linear perceptron neuron to compute the network output:

$$y = \sum_{j=0}^{L} h_l v_l \tag{4}$$

where h_l for $l = 1$, ... , L are the outputs of second layer neurons, v_l are the output layer weights and $h_0 = 1$.

As discussed, incomplete fuzzy rules can be extracted from the network topology. Figure 4 illustrates an example of a fuzzy network composed by and-neurons. This network has 2 input variables, 2 membership functions for each variable and 3 neurons, i.e., $M = 2$, $K = 2$ and $L = 3$. The following if-then rules can be extracted from the network structure:

Rule$_1$: If x_{i1} is A_1^1 with certainty w_{11}...
and x_{i2} is A_2^1 with certainty w_{21}...
Then y_1 is v_1
Rule$_2$: If x_{i1} is A_1^2 with certainty w_{12}...
Then y_2 is v_2

$Rule_3$: If x_{i2} is A_2^2 with certainty w_{23}...
Then y_3 is v_3 (5)

where each free parameter v_l for $l = 1, ..., 3$ can be interpreted as a singleton.

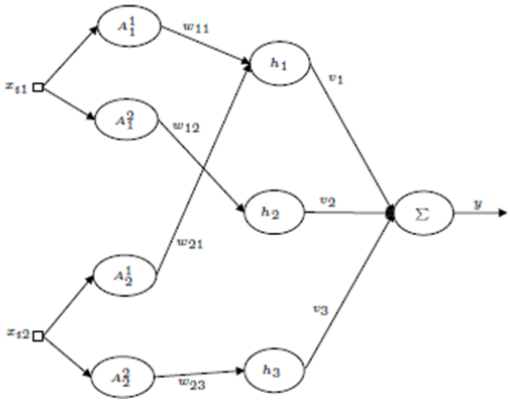


Figure 4: Example of a fuzzy neural network

3 Fuzzy Neural Networks Training Algorithm

The learning algorithm detailed in [41] for fuzzy neural networks uses clustering for topology definition and genetic algorithms for parameter tuning. This learning procedure can generate accurate models. However, the resulting network is usually not interpretable, since the fuzzy sets for each input variable are defined using clustering [21]. Furthermore, the learning algorithm has a high time complexity, given the nature of the parameter tuning algorithm.

In [42] the genetic algorithm is replaced by a new parameter tuning procedure based on ideas from ELM. Extreme Learning Machine [36] is a learning algorithm developed initially for single hidden layer feed-forward neural networks (SLFNs). The algorithm assigns random values for the first layer weights and analytically estimates the output layer weights. When compared with traditional methods for SLFNs learning, this algorithm has good generalization performance and shallow time complexity, since only the output layer parameters are tuned [35]. It has been proved that SLFNs trained using this approach have a universal approximation property, for a Fig. 5. Input domain partition using five triangular membership functions given a choice of the hidden nodes [34].

Inspired by this SLFN learning scheme, a new fast learning algorithm for fuzzy neural networks is described in [42]. This approach assigns random values for the neuron parameters and estimates the parameters of the output layer neuron using least squares. This learning algorithm has a low computational cost, but still generates networks which are not easily interpretable, since the fuzzy sets are still defined using clustering.

This paper proposes an improvement in the learning algorithm described in [42] implemented in [60] in order to improve the interpretability of the resulting networks for regression problems. A variable selection algorithm based on regularization theory is used to define the network topology. This algorithm can generate a sparse topology which can be interpreted as a compact set of incomplete fuzzy rules. The proposed learning algorithm initially defines first layer neurons by dividing each input variable domain interval into K fuzzy sets [37], where K is usually a small number. Next, L_c candidate neurons are randomly selected from the L neurons, where $L_c \leq L$. If L_c chosen is greater than L, all L neurons created are considered as L_c. Usually L_c is initialized with 200 like [60]. For each candidate neuron $l = 1... \ L_c$, first, a random fraction of the input variables are selected. A random value n_l is sampled from a discrete uniform distribution defined over the interval [1, N]. The value n_l represents the number of input variables associated with the l-th neuron. Next, n_l input variables are randomly selected. For each input variable is chosen, a random fuzzy set (first layer neuron) is selected as the input of the l-th neuron and the corresponding weight w_{jl} is sampled from a uniform distribution over the interval [0, 1]. One must note that, after this step, for each neuron l, only n_l weights w_{jl} will have nonzero values, i.e., only the weights associated with the selected input variables. Once the candidate neurons are created, the final network topology is defined by selecting an optimal subset of these neurons. This procedure can be seen as a variable selection problem in which one has to find the optimum subset of variables (in this case, neurons) for a given

cost function. The approach of [60] was also used for the model that will act on linear regression problems. The learning algorithm assumes that the output of a network composed by all L_s selected neurons can be written as [60]:

$$f(x_i) = \sum_{i=0}^{L_s} v_i h_i(x_i) = h(x_i)v \tag{6}$$

where $\mathbf{v} = [v_0, v_1, v_2, ..., v_{Lc}]^T$ is the weight vector of the output layer and $\mathbf{h}\ (x_i) = [z_0, z_1(x_i), z_2(x_i)\hat{a}z_{L_{\ddot{I}(x_i}}$)] is the augmented output (row) vector of the second layer, for $z_0 = 1$. In this context, $\mathbf{h}\ (x_i)$ is considered as the non-linear mapping of the input space for a space of fuzzy characteristics of dimension $Ł_s$.

Note that Equation (6) can be seen as a simple linear regression model since the weights connecting the first two layers are randomly assigned and the only parameters left to be estimated the weights of the output layer. According to ELM learning theory [36], a general type of feature mappings $\mathbf{h}\ (x_i)$ can be used in ELM. This theory says so ELM can approximate any continuous target functions (Universal Approximation Capability [33] and [32] for details). What this feature allows the ELM can be used for regression problems [31]. A learning machine with a feature mapping which does not satisfy the universal approximation condition cannot approximate all continuous target functions. Thus, the universal approximation condition is not only a sufficient condition but also a necessary condition for a feature mapping to be widely used [31]. Thus, the problem of selecting the best candidate neurons subset can be seen as an ordinary linear regression model selection problem [17]. One commonly used approach for model selection is the Least Angle Regression (LARS) algorithm [16]. The LARS is a regression algorithm for high-dimensional data which can estimate not only the regression coefficients but also a subset of candidate regressors to be included in the final model. Given a set of D distinct observations (x_i, y_i), where $x_i = [x_{i1}, x_{i2}, ..., x_{iN}]^T \in \mathbb{R}^N$ and $y_i \in \mathbb{R}$ for $i = 1, \ldots, D$, the cost function of this regression algorithm is defined as:

$$\sum_{i=1}^{D} \|h(x_i)v - y_i\|_{.2} + \lambda\|v\|_{.1} \tag{7}$$

where λ is a regularization parameter, commonly estimated by cross-validation and $\|v\|$ is a weight norm. The first term of (7) corresponds to the sum of the squares of the residues (RSS). This term decreases as the training error decreases. The second term is an L_1 regularization term. This term is used for two reasons. First, it improves the network generalization, avoiding over fitting [18]. Second, it can be used to generate sparse models [57]. In order to understand why LARS can be used as a variable selection algorithm, Equation (7) is rewritten as: [16].

$$\begin{aligned} \min_{\beta} \quad & RSS(v) \\ \text{s.t.} \quad & ||v||_1 \leq \beta \end{aligned} \tag{8}$$

where β is an upper-bound on the L_1-norm of the weights. A small value of β corresponds to a high value of λ, and vice versa. This equation is known as least absolute shrinkage and selection operator [22], or lasso. As in the ridge regression [30], we can re-parameterize the constant β_0, standardizing the predictors. Comparing with the ridge regression, we can say that it replaces the penalty term β_j^2 with $\|\beta_j\|$. This constraint makes the solution nonlinear as a function of y_i, not obtaining a complete expression for the calculation of the coefficients as in the ridge regression [23]. β can be calculated by:

$$\beta_{lasso} = \arg\min_{\beta} \sum_{i=1}^{n} (y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j)^2 \tag{9}$$

$$subject \sum_{i=1}^{N} |\beta_j| \leq t \tag{10}$$

Figure 5 illustrates the contours of the RSS objective function, as well as the l_1 constraint surface. It is well known from the theory of constrained optimization that the optimum solution corresponds to the point where the lowest level of the objective function intersects the constraint surface. Thus, one should note that, when β grows until it meets the objective function, the points in the corners of the l_1 surface (i.e., the points on the coordinate axes) are more likely to intersect the RSS ellipse than one of the sides [57]. Figure 5 shows the estimated prediction error curves and the standard error for the lasso method for the example of an any synthetic database.

Figure 6 shows the estimated prediction error curves and the standard error for the lasso method example of an any synthetic database.

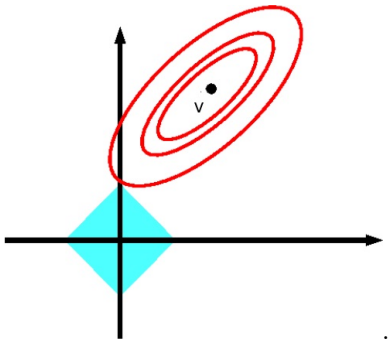


Figure 5: l1 regularization

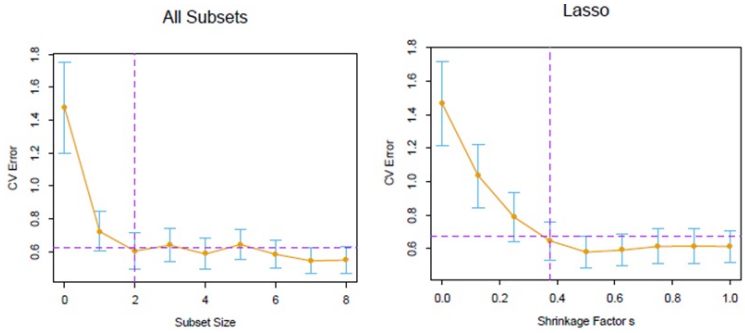


Figure 6: Estimated prediction error curves and their standard errors for the lasso regression method according to subset size [17].

The LARS algorithm can be used to perform model selection since for a given value of λ only a fraction (or none) of the regressors have corresponding non-zero weights. If $\lambda = 0$, the regression problem becomes unconstrained, and all weights are non-zero. As λ increases from 0 to a specific λ_{max} value, the number of non-zero weights decreases from N to 0. For the problem considered in this paper, the regressors $\mathbf{h}_l$ are the outputs of the candidate neurons. Thus, the LARS algorithm can be used to select an optimum subset of the candidate neurons which minimizes (7) for a given value of λ, selected using cross-validation. One must note that this is not a novel approach and has already been used for SLFN pruning [56], [46] and fuzzy model identification [56]. However, one disadvantage of using the LARS algorithm for model selection is that it can give entirely different results for small perturbations in the dataset [57]. If this algorithm is executed for two distinct noisy datasets drawn from the same problem very distinct network topologies may be selected, compromising model interpretability. One existing approach to increase the stability of this model selection algorithm is to use bootstrap resampling. The LARS algorithm is executed on several bootstrap replicates of the training dataset. For each replicate considered, a distinct subset of the regressors is selected. The regressors to be included in the final model are defined according to how often each one of them is selected across different trials. A consensus threshold is defined, say $\gamma = 80\%$, and a regressor is included if it is selected in at least 80% of the trials. This algorithm is known as bootstrap lasso [3]. In this paper, the bootstrap lasso algorithm is used for topology definition.

When we use the regression lasso for the regularization of models we find that the method leads to results with solutions (loose), general of the resulting vectors with many zeros, which represent data of no importance to the result of the variables analyzed, enabling a better selection of models [3]. In this paper, Bach explains that in the studies carried out, they evaluated the consistency of the Lasso model when the data submitted to the model were generated through a vector of load, checking if lasso still managed to perform its actions as the amount of data grew. It was verified that in the case of a fixed number of covariates the lasso method recovers the dispersion pattern, if and only if, a satisfied covariance matrix generation condition is verified. When the lasso method is submitted to data with low correlation its performance is satisfactory, just as when the correlation is high the performance of the actions of the method is impaired [3]. After performing a detailed asymptotic analysis of the selection procedure of the lasso method estimation procedures, where the efforts were focused on a specific decay parameter, Bach concluded that when the trim is proportional to the value of $n^{-1/2}$ where n is the number of observations, the lasso method will select all variables relevant to the model with probability tending

to an exponentially fast with n, while it selects all other variables with strictly positive probability. This new methodology can select the variables most relevant to the model by discarding the least essential variables. When using resampling methods such as the bootstrap, it is possible to imitate the availability of several datasets even if it is using a single data set. Using the bootstrap concept and realizing the intersection between supports. Bach [3] developed a consistent estimating model of regularization, without the conditions of consistencies required by the lasso method. To this new procedure, it gave the name of Bolasso (bootstrap-enhanced least absolute shrinkage operator). This new framework can be seen as a voting scheme applied to the supports of lasso method estimates. However, Bolasso can be seen as a regime of consensus combinations where the most significant subset of variables on which all regressors agree when the aspect is the selection of variables is maintained. [3]. Bolasso procedure is summarized in Algorithm 1.

Algorithm 1: Bolasso- bootstrap-enhanced least absolute shrinkage operator

(b1) Let n be the number of examples, (lines) in X:
(b2) Show n examples of (**X, Y**), uniformly and with substitution, called here ($Xsamp$, $Ysamp$).
(b3) Determine which weights are nonzero given a λ value.
(b4) Repeat steps b1: b3 for a specified number of bootstraps b.
(b5) Take the intersection of the non-zero weights indexes of all bootstrap replications. Select the resulting variables.
(b6) Revise using the variables selected via non-regularized least squares regression (if requested).
(b7) Repeat the procedure for each value of b bootstraps and λ (actually done more efficiently by collecting interim results).
(b8) Determine optimal values for λ and b.

Finally, once the network structure is defined, the learning algorithm has only to estimate the output layer vector $\mathbf{v} = [v_0, v_1, v_2 ... v_L]^T$ which best fits the desired outputs. In this paper these parameters are computed using the Moore- Penrose pseudo-inverse [20]:

$$v = (H^T H)^{-1} H^T Y = H^+ Y \tag{11}$$

where H is defined as:

$$H = \begin{bmatrix} h_0 & h_1(x_1) & ... & h_L(x_1) \\ h_0 & h_1(x_2) & ... & h_L(x_2)) \\ h_0 & h_1(x_N) & ... & h_L(x_N)) \end{bmatrix}_{N \times k} \tag{12}$$

The learning procedure is summarized in Algorithm 2. The algorithm has four parameters:
- the number of membership function, K;
- the number of candidate neurons, L_c;
- the number of bootstrap replications, b;
- the consensus threshold, γ.

4 Experiments

The fuzzy neural network learning algorithm described in the previous section is evaluated using regression problems. For all experiments in this section, only networks composed by and-neurons are considered. All and-neurons use the product (t-norm) and the probabilistic sum (s-norm) and only Gaussian membership functions are used. The network used in the experiments was named R-ANDNET. In all tests the value $K = 3$ and L_c=200 like [60].

The accuracy of the R-ANDNET is evaluated for benchmark regression problems. To perform the comparison between the proposed models in this paper, algorithms were chosen to make two comparisons.

The first group of algorithms will verify the ability of the model to be a universal approximation, comparing its results with other models of fuzzy neural networks based on ELM to perform the network training. The models proposed in [8], [41] and [42] called here respectively of eXUninet, N-Uninet and FL-Uninet were chosen. Already for the second group of models, we chose algorithms that are based on ELM, but can perform an adaptation of the internal architecture of the neural network, through regularization or pruning methods. For this purpose, the models [56], [40] and [58] called EFOP-ELM (TakagiâSugenoâKang based fuzzy inference system is obtained based on the concept of OP-ELM, the state of art in pruning ELM), R-ELANFIS (regularized extreme learning

Algorithm 2: FNN training

(1) Define the number of equally spaced fuzzy sets for each input variable, k.
(2) number of candidate neurons, L_c.
(3) Define bootstrap replications, b.
(4) Define the consensus threshold, γ
(5) Construct L fuzzy neurons with Gaussian membership functions constructed with center values derived from ANFIS and sigma defined at random.
(6) Define the weights of fuzzy neurons at random.
(7) Construct L_c andneurons with random weights and bias =1 on the second layer of the network by welding the L_c fuzzy neurons of the first layer.
(8) **For all** N inputs **do**
(8.1) Calculate the mapping h (x_i)
end for
(9) Select significant L_s neurons using the lasso bootstrap according to the settings of b and γ.
(10) Estimate the weights of the output layer (11)
(11) Calculate the output of the model using an artificial neuron (4).

adaptive neuro fuzzy for regression problems) and OS-FELM (an online sequential fuzzy extreme learning machine has been proposed for function approximation and classification problems) were chosen.

L_c used for the proposed model will serve as reference as L for the other models used in the test. That is, if the R-ANDNET after the pre-filtering method of the neurons set a value of $L_c = 200$ neurons, the value used as L in the other models submitted to the tests for the said base will have the value of 200. However, the performance of the proposed model and of the other models that perform linear regression was evaluated using the root mean square error (RMSE). The RMSE was calculated in the same way as in [41]:

$$RMSE = (\frac{1}{N}(\sum_{k=1}^{n} y^k - y^{'k})^{\frac{1}{2}}) \tag{13}$$

where y^k is the response provided by the model and $y^{'k}$ is the expected output for the test in question.

4.1 Benchmark Regression Problems Dataset

A total of 11 benchmark regression problems were chosen to study the performance of the proposed approach. The configuration of said test follows the assumptions defined in [31], even with the same database where data sets were standardized to zero mean and unit variance. One-third of each data set was selected randomly for validating, and the remaining for training. This set of databases is used in several tests related to regression problems in machine learning. In general, these bases come from specific research and are made available to the community of machine learning researchers to test the accuracy of their models in regression problems. For each test (50 in total for each base) the samples are swapped randomly to avoid trends. Some informationâs of the datasets are shown in Table I. The 11 regression data sets (cf. Table I) can be classified into three groups of data:

-data sets with relatively small size and low dimensions, e.g., Basketball, Strike [47], Cloud, and Auto price [6];
-data sets with relatively small size and medium dimensions, e.g., Pyrim, Housing [6], Bodyfat, and Cleveland [47];
-data sets with relatively large size and low dimensions, e.g., Balloon, Quake [47], and Abalone [6].

4.2 Linear Regression Tests

In this section will be evidenced the results obtained in the tests. In the second test and third test, cross-validation was used [39] for estimate the parameters, $b = \{8, 16, 32\}$ and $\gamma = \{70\%, 80\%, 90\%\}$ where 3 partitions (K) are used.

4.2.1 Fuzzy neural network models for linear regression problems

Table II shows the results of each of the algorithms for each of the databases tested. In this table are presented the results of comparison between the models of fuzzy neural networks to verify the ability of the model to act as a universal approximation.

Table 1: Dataset used in the experiments of regression problems

Dataset	**Init.**	**Feature**	**Train**	**Test**	L_c/**L**
Basketball	BAS	4	64	32	81
Strike	STK	6	416	209	200
Cloud	CLO	9	72	36	200
Auto price	AUT	9	106	53	200
Pyrim	PIR	27	49	25	200
Boston Housing	HOU	13	337	169	200
Body Fat	BOD	14	168	84	200
Cleveland	CLE	13	202	101	200
Ballon	BAL	2	1334	667	9
Quake	QUA	3	1452	726	27
Abalone	ABA	8	2874	1393	200

Table 2: Performance Comparison of Regression Datasets.

Dataset	**R-ANDNET**	**eXUninet**	**N-Uninet**	**FL-Uninet**
BAS	**0.1619 (0.0089)**	0.1842 (0.0421)	0.1766 (0.0027)	0.1673 (0.0042)
STK	0.2356 (0.0432)	**0.1908 (0.0331)**	0.2187 (0.0422)	0.2009 (0.0014)
CLO	0.3245 (0.0254)	**0.2987 (0.0651)**	0.3148 (0.0541)	0.3287 (0.0125)
AUT	0.1876 (0.0461)	0.2098 (0.0567)	0.1786 (0.0021)	**0.1687 (0.0531)**
PIR	**0.1036 (0.0113)**	0.1421 (0.0256)	0.1199 (0.0062)	0.1345 (0.0123)
HOU	0.0865 (0.0103)	0.0765 (0.0442)	**0.0620 (0.0011)**	0.0789 (0.0140)
BOD	0.0341 (0.0148)	0.0654 (0.0112)	0.0476 (0.0239)	**0.0212 (0.0099)**
CLE	0.1652 (0.0008)	0.1754 (0.0643)	**0.1415 (0.0108)**	0.1654 (0.0003)
BAL	**0.0076 (0.0014)**	0.0165 (0.0076)	0.1002 (0.0144)	0.0308 (0.0123)
QUA	**0.1817 (0.0108)**	0.1876 (0.0334)	0.1986 (0.0895)	0.1982 (0.0108)
ABA	0.0712 (0.0065)	**0.0699 (0.0012)**	0.0754 (0.0017)	0.0878 (0.0107)

At first, we verified that the performance of the evaluated algorithms approximates similar behavior, where each of the evaluated models stood out more in a specific context. The proposed model was able to stand out when the number of dimensions of the problem was small. The eXUninet model had its most notable performance when there was a balance between the number of features and a quantity of not so high samples. Another essential factor to be highlighted is that the FL-Uninet model had a better performance in performing the regression when the dimensions were higher (above eight features). As the results were close and it is impossible to identify a trend to the naked eye, statistical tests were used to verify if for each regression set of solving the algorithms had the same behavior [17]. As the test conditions were planned to avoid interfering with undesirable values in the results, we will perform a block variance analysis test [48] to answer the following question: Do the evaluated algorithms have the same RMSE to perform the regression of the 11 proposed classes to the test? As we evaluate the accuracy of the algorithms in 11 bases of different characteristics, we must avoid that the variation of the difficulties of the inputs becomes a source of uncertainty for the problem. Therefore all the source of spurious variation must be controlled. To perform the statistical tests, the analysis of variance (ANOVA) [48] on the results of each of the groups (algorithm x block factor) is used in the test. In general, it is verified that the test has 44 (4 algorithms and 11 bases) groups. Through this test, we intend to conclude if the performance of the algorithms proposed in this paper presents an average performance equal to the other models that are the reference in the literature.

In order to aid in statistical evaluations, we will define as our null hypothesis (H_0) that there is no difference in performance of the algorithms when comparing the RMSE of each of the test blocks in the databases. The alternative hypothesis (H_1) is that the algorithms present different RMSE when acting in the linear regression in the test bases. The confidence interval adopted for the evaluation of the tests is $\alpha = 95$In Figure 7 and 8 we can see the plot of the relationship between the evaluated algorithms and the RMSE of the linear regression test.

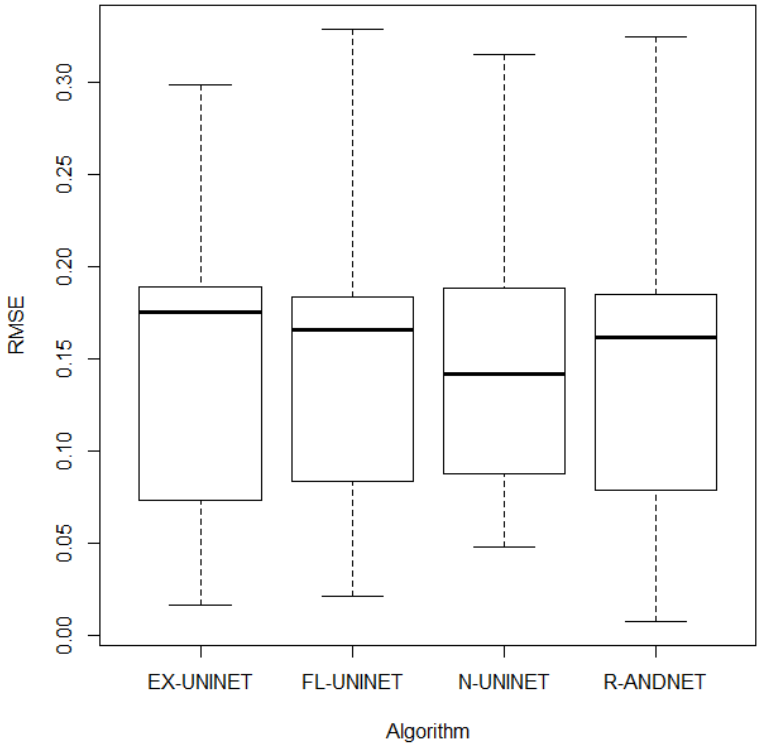


Figure 7: Plot RMSE x Algorithm.

After the information collected, we created graphs with the information obtained to verify the behavior of the models. In the graphical verification of the data we can verify that the models maintain similar behavior in relation to the RMSE, however to resolve any doubts about the issues of accepting or rejecting the null hypothesis, we performed a test of analysis of variances with the data, obtaining as a response that we must accept the null hypothesis (H_0), worth noting that the p-value found was very high: 0.9981. Therefore, we can conclude based on a value $\alpha = 0.05$, that the models analyzed in the test are statistically the same as performing the linear regression in the 11 bases analyzed.

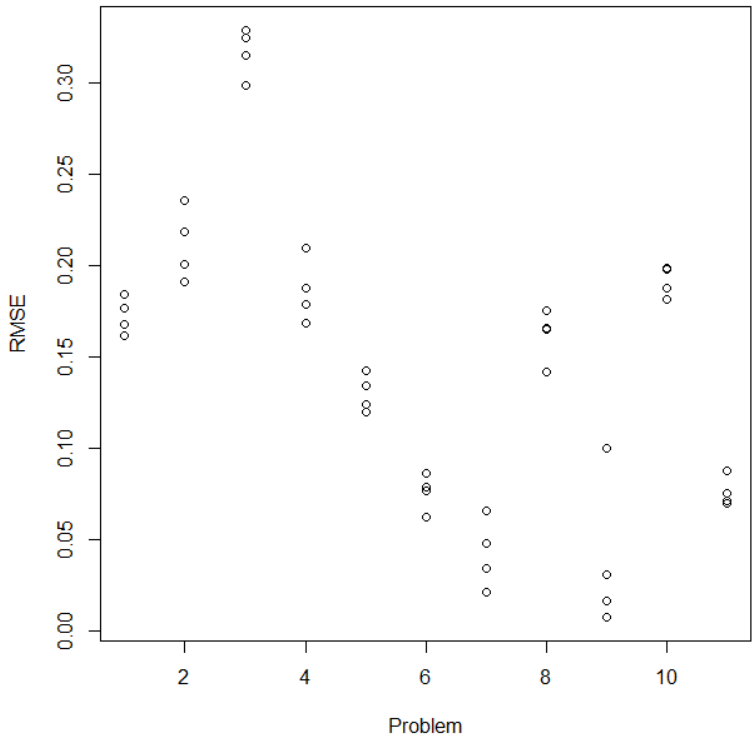


Figure 8: Plot RMSE x Problem.

To confirm the result of the analysis of the variances, three tests were used to validate the normality respectively, homoscedasticity and independence of the data collected. To confirm the conditions of normality of the data, we used the Shapiro-Wilk test [48] that resulted in the confirmation of the characters found analyzed. In the homoscedasticity test, which verifies the equality of variances of the residues, the acceptance of the null hypothesis for the Fligner-Killeen test [48] was verified, which informs us that the variances of the residues involved in the analysis are the same. Finally, the Durbin-Watson test [48] confirmed that the null hypothesis of the test could not be rejected, so the data collected have independence.

Figure 9 shows the graphical results of the validation tests of the ANOVA premises.

In order to present in a more detailed way the behavior of the proposed model in comparison to the other models of fuzzy neural networks submitted to the test, we performed the post-hoc test of multiple comparisons of Tukey [48]. The result of this test is shown in Table III.

Table 3: Multiple Comparisons of Means: Tukey Test

Algorithm	diff	lwr	upr.	Pr adj
FL-UNINET-EX-UNINET	-0.003109	-0.086463	0.080245	0.999634
N-UNINET-EX-UNINET	0.001545	-0.081809	0.084900	0.999955
R-ANDNET-EX-UNINET	-0.003400	-0.086754	0.079547	0.999522
N-UNINET-FL-UNINET	0.004654	-0.078700	0.088245	0.998779
R-ANDNET-FL-UNINET	-0.002909	-0.086463	0.080245	0.999999
R-ANDNET-N-UNINET	-0.004945	-0.088302	0.078409	0.998530

Since no rejections of H_0 occurred with any of the premise verification tests, we can conclude, with a 95% confidence interval, that we must accept our null hypothesis for the statistical tests used.

4.2.2 Regularized or pruned methods for regression problems

In this test, we verified the ability to regularize the proposed model in comparison to the regularized models or intended for pruning of neurons in the hidden layer of its structure. The objective is to verify if the fuzzy neural

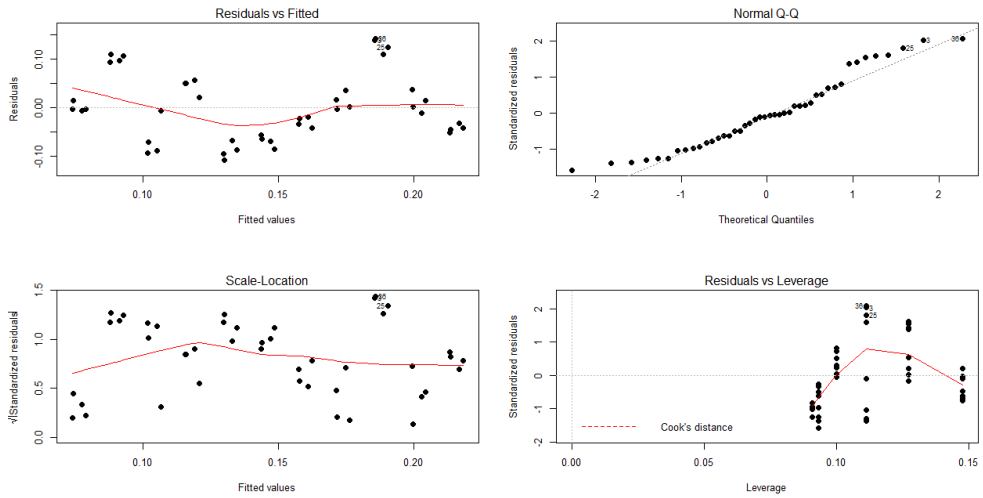


Figure 9: ANOVA test check.

network can maintain the capacity to perform linear regression and how it behaves in the face of the pruning of neurons through the bolasso. Table IV and Table V present the result of tests.

Table 4: Performance Comparison of Regression Datasets for Regularized Pruning Methods.

Dataset	R-ANDNET	EFOP-ELM	OSF-ELM	R-ELAFIS
BAS	**0.1817 (0.0111)**	0.1908 (0.0165)	0.2021 (0.0448)	0.2062 (0.0414)
STK	0.2816 (0.0546)	**0.1799 (0.0423)**	0.1987 (0.0074)	0.2941 (0.0364)
CLO	0.8642 (0.0642)	0.7654 (0.0115)	**0.6668 (0.0712)**	0.7681 (0.1008)
AUT	0.2071 (0,0876)	**0.1815 (0.0532)**	0.1987 (0.0101)	0.2245 (0.0436)
PIR	**0.1276 (0.0008)**	0.1343 (0.0876)	0.1619 (0.0421)	0.1513 (0.0778)
HOU	0.2019 (0.0876)	0.1876 (0.0477)	0.1987 (0.0722)	**0.1543 (0.0765)**
BOD	**0.0259 (0.0123)**	0.0324 (0.0176)	0.3531 (0.0044)	0.2780 (0.0654)
CLE	0.2034 (0.0076)	**0.1124 (0.0163)**	0.2173 (0.0345)	0.1876 (0.0089)
BAL	**0.0156 (0.0053)**	0.0311 (0.0048)	0.0221 (0.0017)	0.0421 (0.0654)
QUA	**0.1973 (0.0301)**	0.2134 (0.0412)	0.2396 (0.0972)	0.3498 (0.0498)
ABA	0.0567 (0.0126)	0.0341 (0.0103)	0.0442 (0.0112)	**0.0245 (0.0341)**

Table 5: Final Network Neurons (l_s)

Dataset	R-ANDNET	EFOP-ELM	OSF-ELM	R-ELAFIS
BAS	**12.650 (0.756)**	22.650 (13.165)	16.761 (6.543)	42.677 (23.158)
STK	**48.965 (13.985)**	67.821 (14.875)	89.087 (9.812)	64.997 (14.833)
CLO	67.834 (18.998)	**64.560 (9.764)**	71.714 (8.701)	89.221 (19.723)
AUT	**56.412 (12.008)**	0.1815 (0.0532)	0.1987 (0.0101)	0.2245 (0.0436)
PIR	**101.965 (5.983)**	106.331 (12.316)	118.650 (14.876)	136.347 (31.331)
HOU	**78.943 (7.698)**	79.516 (9.142)	79.017 (8.765)	81.424 (6.165)
BOD	**69.113 (15.657)**	71.912 (0.987)	70.761 (15.321)	91.817 (0.441)
CLE	**55.437 (16.842)**	62.980 (7.678)	56.111 (8.742)	77.180 (12.016)
BAL	**5.098 (0.007)**	8.000 (0.000)	7.876 (0.987)	7.0870 (0.007)
QUA	12.761 (2.098)	**10.762 (0.661)**	15.531 (4.841)	14.678 (0.890)
ABA	**32.365 (8.431)**	54.812 (12.012)	49.067 (12.150)	44.832 (12.043)

In the second test, we verified that there are no significant differences in the RMSE results among the three

methods analyzed. However, when comparing the number of neurons used. It can be seen that the model proposed in the paper worked on average with an architecture to perform the linear regression of the database.

4.2.3 Fuzzy Rules and interpretation of the results obtained by the model

In order to identify the interpretability of the results. A test with synthetic data was proposed so that the predictive capacity of the model is demonstrated graphically and through fuzzy rules. The following code was generated in Matlab to perform the creation of a synthetic. Random and independent base. Seeking to define the better visibility of the facts.

```
X=[(100:200)'+randn(101.1) (300:400)'+randn(101.1) ones(101.1)];
P1=2; P2=1; P3=100;
Y=P1*X(:.1)+P2*X(:.2)+P3;
Y=Y+randn(101.1);
```

For the test to be performed. consider $K = 2$ for Gaussian Membership Function. $b = 16$ and $\gamma = 0.9$. In figure 10 presents the Gaussian Membership Functions create in this tests.

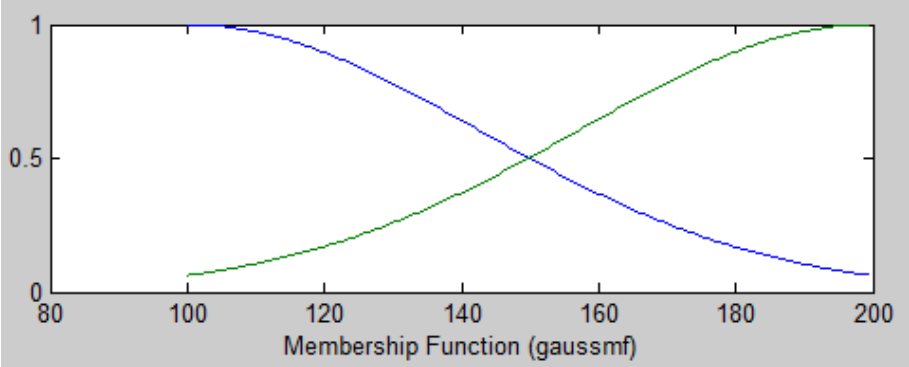


Figure 10: Gaussian Membership Function.

The model generated four fuzzy rules in the first layer. This type of approach allows saying that in a group of two characteristics on a Cartesian axis (x_1 and x_2). Equally spaced membership functions can denote literal values as small and large. In figure 11 below, consider the abscissa axis as x_1 and the ordinate axis as x_2. In figure 12 we can see how the membership functions can help in the division of the data into space.

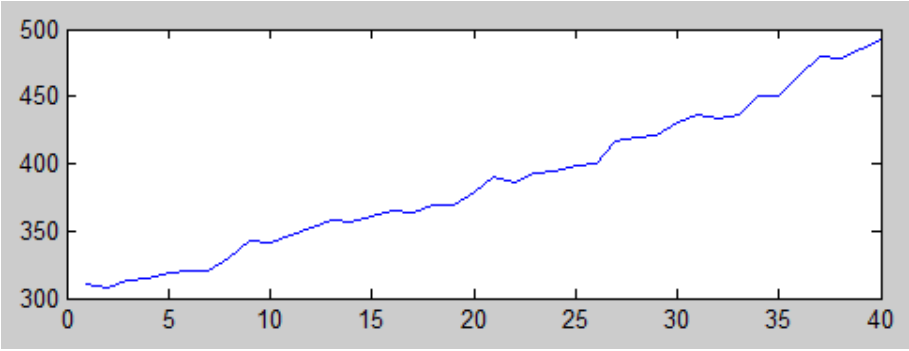


Figure 11: Hypothetical data used in the regression

In Figure 12 we can see how the membership functions can delimit more representative spaces for the problem. In this presented context, each one of the Gaussian membership functions was given names of *small* and *large*. Note that when the ordinate axis has a *small* membership function, and the abscissa axis has a *large* membership function, there is no data representative of the model.

The same happens when in the x_1 axis the *large* membership function is represented, and the *small* membership function represents the x_2 axis. Thus the resampling regularization method can eliminate these two neurons

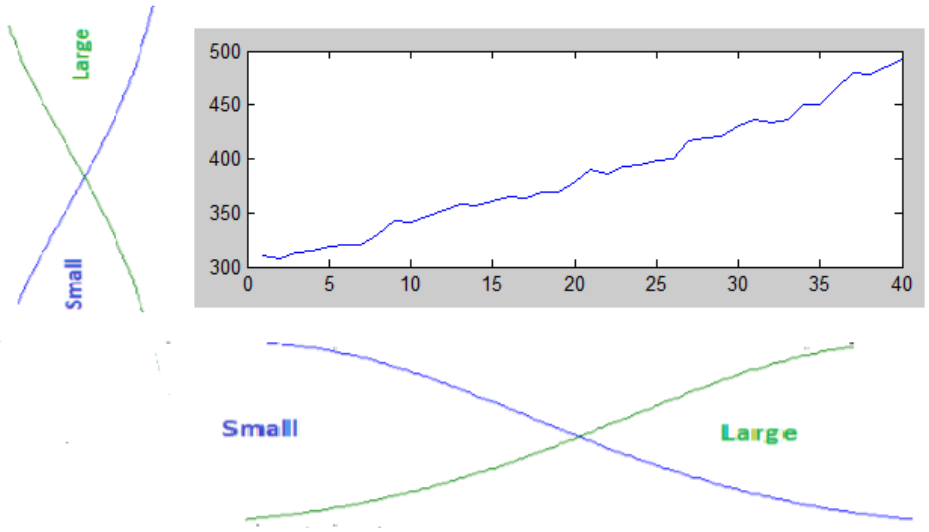


Figure 12: Presentation of the hypothetical data used in the regression problem and the Gaussian membership functions.

from the structure of the model. Allowing it to have more representative answers to solve the problem. In this example model we have four fuzzy neurons represented by the following combinations:

1. First Fuzzy Neuron: x_1 small and x_2 small
2. Second Fuzzy Neuron: x_1 small and x_2 large
3. Third Fuzzy Neuron: x_1 large and x_2 small
4. Fourth Fuzzy Neuron: x_1 large and x_2 large

The method of regularization will eliminate two fuzzy neurons that do not present data in this problem (second and third), will have two neurons that are more representative of the problem, thus allowing to build fuzzy rules (5) of each of the andneuron.

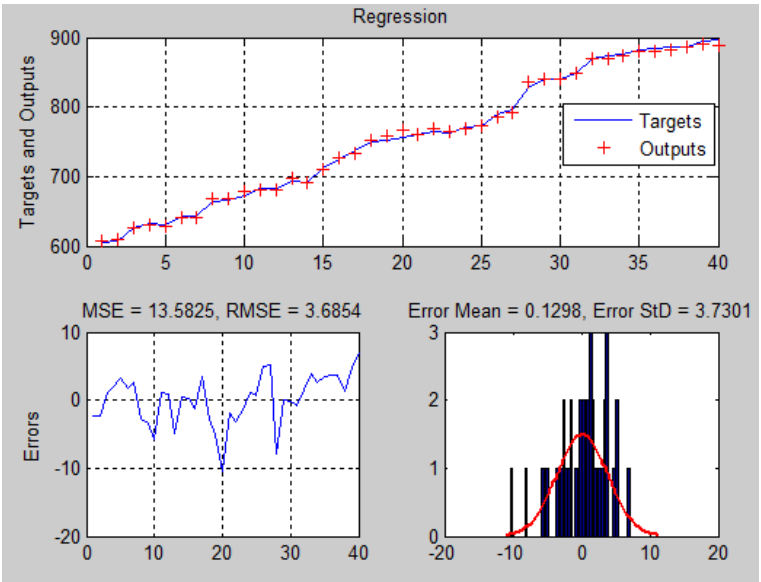


Figure 13: Results of regression problem.

For the four fuzzy neurons resulting from this experiment, we can conclude the following fuzzy rules:

Rule$_1$: If x_{i1} is *small* with certainty 0.7139
and x_{i2} is *small* with certainty 0.5207
Then y_1 is 0.6180
$Rule_2$: If x_{i1} is *small* with certainty 0.2607
and x_{i2} is *large* with certainty 0.8759
Then y_2 is 0.1968
$Rule_3$: If x_{i1} is *large* with certainty 0.6297
and x_{i2} is *small* with certainty 0.4699
Then y_3 is -1.0633
$Rule_4$: If x_{i2} is *large* with certainty 0.4196
and x_{i2} is *large* with certainty 0.5207
Then y_4 is 0.2851 (14)

In this context, we can demonstrate several relationships, for example, suppose that x_1 is the value of an invoice and x_2 represents taxes. The response variable may be the value of a person's debts. When the value of invoices and taxes are low the value of the person's debts will be low. Likewise, when invoice and tax amounts are high, the person's spending forecast will be high. This type of rule assists in the creation of expert systems, to assist in the prediction of factors that may or may not impact the linear regression of a studied context.

5 Conclusion

This paper has introduced a new learning algorithm for fuzzy neural networks based on ideas from Extreme Learning Machine and regularization theory for regression problems. Random parameters are defined for the second layer neurons, and the bootstrap lasso algorithm is used to generate sparse models. The experiments performed and the results suggest the network as a promising alternative to build accurate and transparent models. The statistical tests confirmed that the fuzzy neural network could behave as a universal approximation, performing linear regressions equivalent to models widely used in the literature and maintains a leaner architecture with regularization techniques, allowing the hidden layer of the model to use the lowest mean number of neurons to perform their activities. As the cross-validation method becomes a little expensive in the choice of network parameters, new optimization techniques for the choice of input parameters of the model may be the subject of future research. Future work shall address methods to improve the network interpretability and evaluation on multi-class classification problems. In future work, the model can be adapted to incorporate convolutions, deep learning concepts and think of new techniques of data fuzzification [61] and [45] for example, so that the model does not have its performance on creating membership functions proportional to the number of dimensions and/or samples.

Acknowledgements

The thanks of this work are for CEFET-MG and UNA Betim.

References

[1] Plamen Angelov. *Evolving Takagi-Sugeno Fuzzy Systems from Streaming Data (eTS+)*, chapter 2, pages 21–50. Wiley-Blackwell, 2010.

[2] Plamen Angelov and Ronald Yager. A new type of simplified fuzzy rule-based system. *International Journal of General Systems*, 41(2):163–185, 2012.

[3] Francis R Bach. Bolasso: model consistent lasso estimation through the bootstrap. In *Proceedings of the 25th international conference on Machine learning*, pages 33–40. ACM, 2008.

[4] Rosangela Ballini and Fernando Gomide. Learning in recurrent, hybrid neurofuzzy networks. In *Fuzzy Systems, 2002. FUZZ-IEEE'02. Proceedings of the 2002 IEEE International Conference on*, volume 1, pages 785–790. IEEE, 2002.

[5] James C. Bezdek. *Objective Function Clustering*, pages 43–93. Springer US, Boston, MA, 1981.

[6] Catherine Blake. Uci repository of machine learning databases. *http://www. ics. uci. edu/˜ mlearn/MLRepository. html*, 1998.

[7] F. Bordignon and F. Gomide. Extreme learning for evolving hybrid neural networks. In *2012 Brazilian Symposium on Neural Networks*, pages 196–201, Oct 2012.

[8] Fernando Bordignon and Fernando Gomide. Uninorm based evolving neural networks and approximation capabilities. *Neurocomputing*, 127:13–20, 2014.

[9] A de P Braga, APLF Carvalho, and Teresa Bernarda Ludermir. *Redes neurais artificiais: teoria e aplicações*. Livros Técnicos e Científicos Rio de Janeiro, 2000.

[10] Walmir M Caminhas, Hermano Tavares, Fernando AC Gomide, and Witold Pedrycz. Fuzzy set based neural networks: Structure, learning and application. *JACIII*, 3(3):151–157, 1999.

[11] P. V. de Campos Souza and P. F. A. de Oliveira. Regularized fuzzy neural networks based on nullneurons for problems of classification of patterns. In *2018 IEEE Symposium on Computer Applications Industrial Electronics (ISCAIE)*, pages 25–30, April 2018.

[12] P. V. de Campos Souza, G. R. L. Silva, and L. C. B. Torres. Uninorm based regularized fuzzy neural networks. In *2018 IEEE Conference on Evolving and Adaptive Intelligent Systems (EAIS)*, pages 1–8, May 2018.

[13] Paulo Vitor de Campos Souza. Pruning fuzzy neural networks based on unineuron for problems of classification of patterns. *Journal of Intelligent and Fuzzy Systems*, 35:2597–2605, 2018.

[14] Paulo Vitor de Campos Souza and Luiz Carlos Bambirra Torres. Regularized fuzzy neural network based on or neuron for time series forecasting. In Guilherme A. Barreto and Ricardo Coelho, editors, *Fuzzy Information Processing*, pages 13–23, Cham, 2018. Springer International Publishing.

[15] J. C. Dunn. A fuzzy relative of the isodata process and its use in detecting compact well-separated clusters. *Journal of Cybernetics*, 3(3):32–57, 1973.

[16] Bradley Efron, Trevor Hastie, Iain Johnstone, Robert Tibshirani, et al. Least angle regression. *The Annals of statistics*, 32(2):407–499, 2004.

[17] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. *The elements of statistical learning*, volume 1. Springer series in statistics New York, NY, USA:, 2001.

[18] Federico Girosi, Michael Jones, and Tomaso Poggio. Regularization theory and neural networks architectures. *Neural computation*, 7(2):219–269, 1995.

[19] Adam F Gobi and Witold Pedrycz. Logic minimization as an efficient means of fuzzy structure discovery. *IEEE Transactions on Fuzzy Systems*, 16(3):553–566, 2008.

[20] Gene Golub and William Kahan. Calculating the singular values and pseudo-inverse of a matrix. *Journal of the Society for Industrial and Applied Mathematics, Series B: Numerical Analysis*, 2(2):205–224, 1965.

[21] Serge Guillaume. Designing fuzzy inference systems from data: An interpretability-oriented review. *IEEE transactions on fuzzy systems*, 9(3):426–443, 2001.

[22] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. Unsupervised learning. In *The elements of statistical learning*, pages 485–585. Springer, 2009.

[23] Trevor Hastie, Robert Tibshirani, and Jerome H. Friedman. *The elements of statistical learning: data mining, inference, and prediction, 2nd Edition*. Springer series in statistics. Springer, 2009.

[24] S. Haykin. *Redes Neurais: Princípios e Prática*. Artmed, 2007.

[25] Simon Haykin. *Neural networks: a comprehensive foundation*. Prentice Hall PTR, 1994.

[26] Michel Hell, Pyramo Costa, and Fernando Gomide. Participatory learning in power transformers thermal modeling. *IEEE Transactions on Power Delivery*, 23(4):2058–2067, 2008.

[27] Michel Hell, Fernando Gomide, Rosangela Ballini, and Pyramo Costa. Uninetworks in time series forecasting. In *Fuzzy Information Processing Society, 2009. NAFIPS 2009. Annual Meeting of the North American*, pages 1–6. IEEE, 2009.

[28] Michel Bortolini Hell et al. Abordagem neurofuzzy para modelagem de sistemas dinamicos não lineares. 2008.

[29] F. Herrera, M. Lozano, and J.L. Verdegay. Tackling real-coded genetic algorithms: Operators and tools for behavioural analysis. *Artificial Intelligence Review*, 12(4):265–319, Aug 1998.

[30] Arthur E Hoerl and Robert W Kennard. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 12(1):55–67, 1970.

[31] G. Huang, H. Zhou, X. Ding, and R. Zhang. Extreme learning machine for regression and multiclass classification. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 42(2):513–529, April 2012.

[32] Guang-Bin Huang and Lei Chen. Convex incremental extreme learning machine. *Neurocomputing*, 70(16):3056 – 3062, 2007. Neural Network Applications in Electrical Engineering Selected papers from the 3rd International Work-Conference on Artificial Neural Networks (IWANN 2005).

[33] Guang-Bin Huang, Lei Chen, and Chee-Kheong Siew. Universal approximation using incremental constructive feedforward networks with random hidden nodes. *Trans. Neur. Netw.*, 17(4):879–892, July 2006.

[34] Guang-Bin Huang, Lei Chen, Chee Kheong Siew, et al. Universal approximation using incremental constructive feedforward networks with random hidden nodes. *IEEE Trans. Neural Networks*, 17(4):879–892, 2006.

[35] Guang-Bin Huang, Qin-Yu Zhu, and Chee-Kheong Siew. Extreme learning machine: a new learning scheme of feedforward neural networks. In *Neural Networks, 2004. Proceedings. 2004 IEEE International Joint Conference on*, volume 2, pages 985–990. IEEE, 2004.

[36] Guang-Bin Huang, Qin-Yu Zhu, and Chee-Kheong Siew. Extreme learning machine: theory and applications. *Neurocomputing*, 70(1-3):489–501, 2006.

[37] J-SR Jang. Anfis: adaptive-network-based fuzzy inference system. *IEEE transactions on systems, man, and cybernetics*, 23(3):665–685, 1993.

[38] J. Kim and N. Kasabov. Hyfis: adaptive neuro-fuzzy inference systems and their application to nonlinear dynamical systems. *Neural Networks*, 12(9):1301 – 1319, 1999.

[39] Ron Kohavi. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence - Volume 2*, IJCAI'95, pages 1137–1143, San Francisco, CA, USA, 1995. Morgan Kaufmann Publishers Inc.

[40] Shihabudheen KV and G.N. Pillai. Regularized extreme learning adaptive neuro-fuzzy algorithm for regression and classification. *Know.-Based Syst.*, 127(C):100–113, July 2017.

[41] Andre Lemos, Walmir Caminhas, and Fernando Gomide. New uninorm-based neuron model and fuzzy neural networks. In *Fuzzy Information Processing Society (NAFIPS), 2010 Annual Meeting of the North American*, pages 1–6. IEEE, 2010.

[42] Andre Paim Lemos, Walmir Caminhas, and Fernando Gomide. A fast learning algorithm for uninorm-based fuzzy neural networks. In *Fuzzy Information Processing Society (NAFIPS), 2012 Annual Meeting of the North American*, pages 1–6. IEEE, 2012.

[43] Xiaofeng Liang and Witold Pedrycz. Logic-based fuzzy networks: a study in system modeling with triangular norms and uninorms. *Fuzzy sets and systems*, 160(24):3475–3502, 2009.

[44] E. Lima, F. Gomide, and R. Ballini. Participatory evolving fuzzy modeling. In *2006 International Symposium on Evolving Fuzzy Systems*, pages 36–41, Sept 2006.

[45] Jingjing Ma, Xiangming Jiang, and Maoguo Gong. Two-phase clustering algorithm with density exploring distance measure. *CAAI Transactions on Intelligence Technology*, 3(1):59–64, 2018.

[46] José M MartíNez-MartíNez, Pablo Escandell-Montero, Emilio Soria-Olivas, José D MartíN-Guerrero, Rafael Magdalena-Benedito, and Juan GóMez-Sanchis. Regularized extreme learning machine for regression problems. *Neurocomputing*, 74(17):3716–3721, 2011.

[47] M Mike. Statistical datasets. *Dept. Statist., Univ. Carnegie Mellon, Pittsburgh, PA*, 1989.

[48] D.C. Montgomery. *Design and Analysis of Experiments*. Student solutions manual. John Wiley & Sons, 2008.

[49] Wim De Mulder, Steven Bethard, and Marie-Francine Moens. A survey on the application of recurrent neural networks to statistical language modeling. *Computer Speech Language*, 30(1):61 – 98, 2015.

[50] Witold Pedrycz. Fuzzy neural networks and neurocomputations. *Fuzzy Sets and Systems*, 56(1):1–28, 1993.

[51] Witold Pedrycz. Heterogeneous fuzzy logic networks: fundamentals and development studies. *IEEE Transactions on Neural Networks*, 15(6):1466–1481, 2004.

[52] Witold Pedrycz. Logic-based fuzzy neurocomputing with unineurons. *IEEE Transactions on Fuzzy Systems*, 14(6):860–873, 2006.

[53] Witold Pedrycz and Rafik A Aliev. Logic-oriented neural networks for fuzzy neurocomputing. *Neurocomputing*, 73(1-3):10–23, 2009.

[54] Witold Pedrycz and Fernando Gomide. *Fuzzy systems engineering: toward human-centric computing.* John Wiley & Sons, 2007.

[55] Alok Porwal, E. J. M. Carranza, and M. Hale. A hybrid neuro-fuzzy model for mineral potential mapping. *Mathematical Geology*, 36(7):803–826, Oct 2004.

[56] Federico Montesino Pouzols and Amaury Lendasse. Evolving fuzzy optimally pruned extreme learning machine for regression problems. *Evolving Systems*, 1(1):43–58, 2010.

[57] Christian Robert. Machine learning, a probabilistic perspective, 2014.

[58] H. Rong, G. Huang, N. Sundararajan, and P. Saratchandran. Online sequential fuzzy extreme learning machine for function approximation and classification problems. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 39(4):1067–1072, Aug 2009.

[59] R. Rosa, F. Gomide, and R. Ballini. Evolving hybrid neural fuzzy network for system modeling and time series forecasting. In *2013 12th International Conference on Machine Learning and Applications*, volume 2, pages 378–383, Dec 2013.

[60] Paulo Vitor C Souza. Regularized fuzzy neural networks for pattern classification problems. *International Journal of Applied Engineering Research*, 13(5):2985–2991, 2018.

[61] Jun Wu and Xin Xu. Decentralised grid scheduling approach based on multi-agent reinforcement learning and gossip mechanism. *CAAI Transactions on Intelligence Technology*, 3(1):8–17, 2018.

[62] Lotfi A Zadeh. Fuzzy sets. *Information and control*, 8:3, 1965.

Inteligencia Artificial 21(62), 134-144
doi: 10.4114/intartif.vol21iss62pp134-144

INTELIGENCIA ARTIFICIAL

http://journal.iberamia.org/

Artificial Neural Network (ANN) in a Small Dataset to determine Neutrality in the Pronunciation of English as a Foreign Language in Filipino Call Center Agents

Rey Benjamin M. Baquirin[1], Proceso L. Fernandez Jr.[2]
[1,2]Ateneo de Manila University, Quezon City, Philippines
[1]rey.baquirin@obf.ateneo.edu
[2]pfernandez@ateneo.edu

Abstract Artificial Neural Networks (ANNs) have continued to be efficient models in solving classification problems. In this paper, we explore the use of an ANN with a small dataset to accurately classify whether Filipino call center agents' pronunciations are neutral or not based on their employer's standards. Isolated utterances of the ten most commonly used words in the call center were recorded from eleven agents creating a dataset of 110 utterances. Two learning specialists were consulted to establish ground truths and Cohen's Kappa was computed as 0.82, validating the reliability of the dataset. The first thirteen Mel-Frequency Cepstral Coefficients (MFCCs) were then extracted from each word and an ANN was trained with Ten-fold Stratified Cross Validation. Experimental results on the model recorded a classification accuracy of 89.60% supported by an overall F-Score of 0.92.

Keywords: Automatic Speech Classification, Artificial Intelligence, Neural Networks, Mel-Frequency Cepstral Coefficients, Machine Learning

1 Introduction

As spoken language proficiency remains to be the most valuable skill call centers look for in their employees, the use of technologies that aid in its assessment and training have become a norm especially in the Philippine Business Process Outsourcing (BPO) industry [1]. Pearson's Versant and Berlitz's Spoken Language Test for example, are the most commonly used technologies that companies administer to screen prospective employees in language comprehension and speaking proficiency, including pronunciation [2].

Pronunciation is the act of producing sounds of speech in accordance to accepted standards of a language [3]. Published dictionaries indicate how words in a specific language should be pronounced to be properly understood by listeners. Nevertheless, these pronunciation guidelines are not rigidly followed as words that are deemed to have 'acceptable pronunciations' change in actual conversation and are affected by many factors including education, geography, social status, race, and culture [4]. This change leads to the acceptability of pronunciation to just be perceived as either 'neutral' or 'not neutral' depending on the circumstances surrounding the speakers. For most Filipino call centers, this pertains to their employees' ability to speak English in a way that is neutral enough for their clients to understand.

However, the technologies previously mentioned, Versant and Berlitz, are often costly and companies only ever utilize them once or twice per employee or applicant. Consequently, all other assessments conducted to measure spoken language proficiency rely on the individual knowledge and experience of recruiters and trainers leading to results with unavoidable personal bias. Call centers therefore suffer from having a subjectively varied standard for assessing spoken language proficiency, especially pronunciation.

ISSN: 1137-3601 (print), 1988-3064 (on-line)

This paper contributes to the knowledge space by proposing a model that can accurately classify whether call center agents' utterances are pronounced neutral or not. The goal is to train an Artificial Neural Network (ANN) that captures a uniform, objective standard for pronunciation neutrality assessment specific to a call center's standards albeit using a small dataset.

2 Literature Review

The decision to use an ANN for this study was drawn from the many publications that have used Machine Learning (ML) techniques in Automatic Speech Recognition (ASR). However, there is only a limited amount of literature regarding the use of ANNs to classify pronunciation neutrality as far as the researchers are aware of.

Studies like [5] used Deep Neural Networks (DNN) to detect mispronunciations of Mandarin and English words to enhance the performance of a Computer-Aided Language Learning (CALL) system. In addition, the authors of [6] used Hidden Markov Models (HMM) and DNNs to detect pronunciation errors of Japanese students learning Chinese to provide instructive feedback when using a Computer-Aided Pronunciation Training (CAPT) system. Both undertakings used ML for ASR but were more concerned on detecting pronunciation errors than generalizing whether an utterance is neutral or not. However, both also used Neural Network-based architectures as classifiers. Although HMM has been the most common model used in modern ASR systems, these studies exemplify that using Neural Networks as classifiers for speech or signal processing problems yield good results too.

[7] also focused on pronunciation but explored other areas. This research involved the use of crowd-sourcing techniques to generate pronunciations for named-entities. The study used the Google Voice Search production recognition engine, which runs on a DNN, to learn crowd-sourced business names and street names from a database of voice search queries in Google Maps. Rutherford and his team used a Grapheme-to-Phoneme dictionary mapping to facilitate pronunciation learning, which is a common technique used to evaluate the correctness of the model's predictions phonetically. For example, the string 'Iowa' can be mapped in the dictionary to be worded phonetically as [AY OW WUH] to successfully learn pronunciation. Like the previously mentioned studies, Neural Networks were also used as classifiers in this undertaking.

Adding to the fact that these studies used Neural Network-based classifiers, the use of Mel Frequency Cepstral Coefficients (MFCCs) as features for classifier training was also a noted commonality amongst them. MFCC extraction is one of the most used techniques in ASR research as it accurately approximates how humans generate speech sounds. Hence in this experiment, an ANN is trained entirely on MFCC features extracted from a small dataset of isolated speech utterances.

3 Methodology

Procedures, techniques, tools and algorithms used in the study are discussed in three sub-sections: Dataset Preparation, Feature Extraction, and Training and Validation.

3.1 Dataset Preparation

The dataset was collected from a Filipino call center catering to American clients. It contains 110 audio files, with each file representing one of ten words as follows: 'actually', 'basically', 'broadband', 'computer', 'Genie', 'internet', 'mobile', 'mobility', 'unfortunately', and 'wireless'. These words are the 10 most commonly used words in the call center where the data was collected. There are 11 utterances per word, with each utterance recorded from a different speaker.

Each utterance was recorded via Audacity and a condenser mic with 8000Hz sampling rate stored as 16-bit integers. All utterances were then saved as WAV files in a mono channel with the amplitude centred at 0dB. The trailing silences before and after the actual utterance were also removed in Audacity. The files have a typical duration of 0.3 – 1.5 seconds.

Ground truth labels of "Neutral" or "Not Neutral" were determined for each utterance with the help of two Learning Specialists as raters. The 110 utterances were labelled as 68 'Neutral' and 42 'Not Neutral' instances. To ensure the dataset's reliability and assess interrater agreement, Cohen's Kappa was computed as given by Equation 1.

$$\kappa \equiv \frac{p_o - p_e}{1 - p_e} = 1 - \frac{1 - p_o}{1 - p_e}, \tag{1}$$

where p_o is the observed proportionate agreement between the raters across all categories, and p_e is the probability that the raters will agree on a label [8].

Computations resulted to a kappa score of 0.82 for the dataset; showing strong agreement between the two raters and approximating 64-81% of the data as reliable to use as per table 1. The kappa score also reinforced the viability of the dataset to represent a baseline standard for pronunciation assessment specific to the company's requirements.

Table 1: Interpretation of Cohen's Kappa [9]

Value	Level of agreement	Sample of reliable data
0-0.20	None	0-0.4%
.21–.39	Minimal	4–15%
.40–.59	Weak	15–35%
.60–.79	Moderate	35–63%
.80–.90	Strong	64–81%
Above.90	Almost Perfect	82–100%

3.2 Feature Extraction

Feature vectors were extracted from each audio file in the form of Mel-Frequency Cepstral Coefficients (MFCCs) through Python. MFCC extraction is one of the most used techniques in ASR research as it accurately approximates how humans generate speech sounds [10]. MFCC extraction assumes that although a speech sound constantly changes over time, it can be represented by a series of power spectrums captured in very short time frames [11]. It was implemented in the study as per the process flow in figure 1.

Figure 1. The MFCC extraction process flow used in the study.

Each audio file was passed through a pre-emphasis filter to center the low and high frequency readings. It was then windowed using Hamming Windows with a window length of 25ms and a window step of 10ms, thereby framing the signal into short frames. Figure 2 shows an example of a windowed portion of an input audio file.

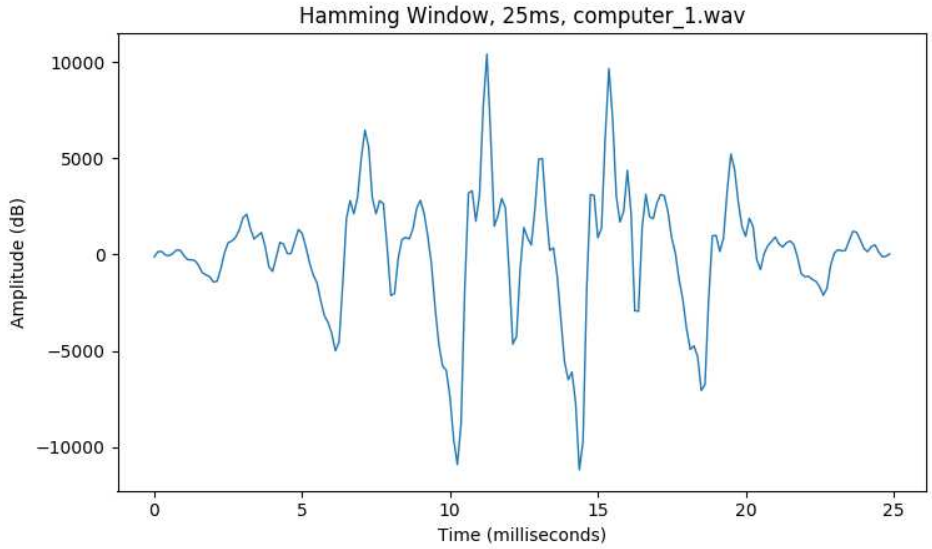


Figure 2. A 25ms hamming-windowed portion of the file 'computer1.wav'.

Windowing resulted to each frame of the input signal having 200 samples, derived from the original 8000Hz sample rate. The Fast Fourier Transform (FFT) algorithm was used in each frame to calculate spectral density, creating individual spectrums for each frame and approximating the periodogram of the power spectrum. Figure 3 shows the periodogram spectrum of figure 2 calculated through FFT.

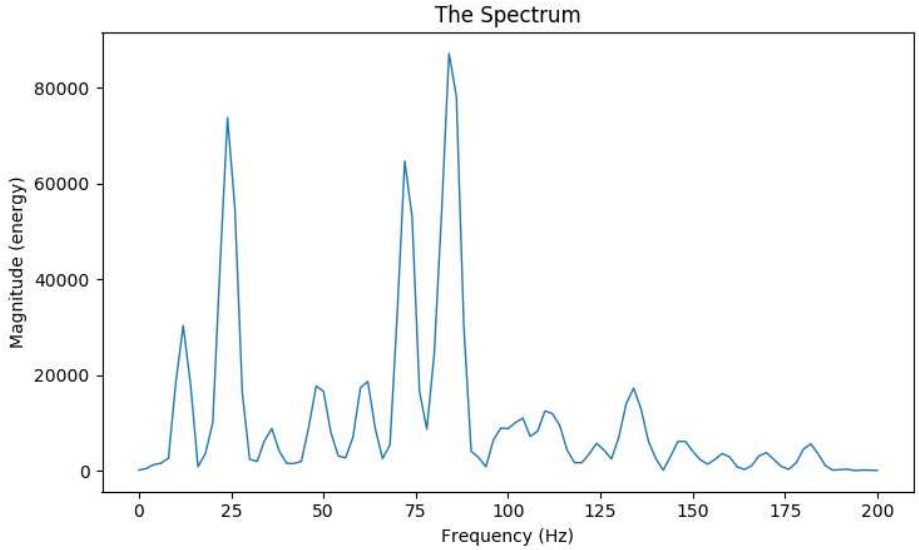


Figure 3. The spectrum generated after FFT.

The spectrum represents the identity of the input signal by detecting which frequencies are present in each frame [12]. To mimic human hearing and improve training results, a mel filterbank was used to discard information on higher frequency bands. This was done by spacing filters using the mel scale with more filters in lower to mid frequency bands as they are more relevant to human hearing and are where speech signals are commonly found [13]. Twenty-six (26) filters were used in the filterbank, separated across the spectrum via the computed mel scale as shown in figure 4.

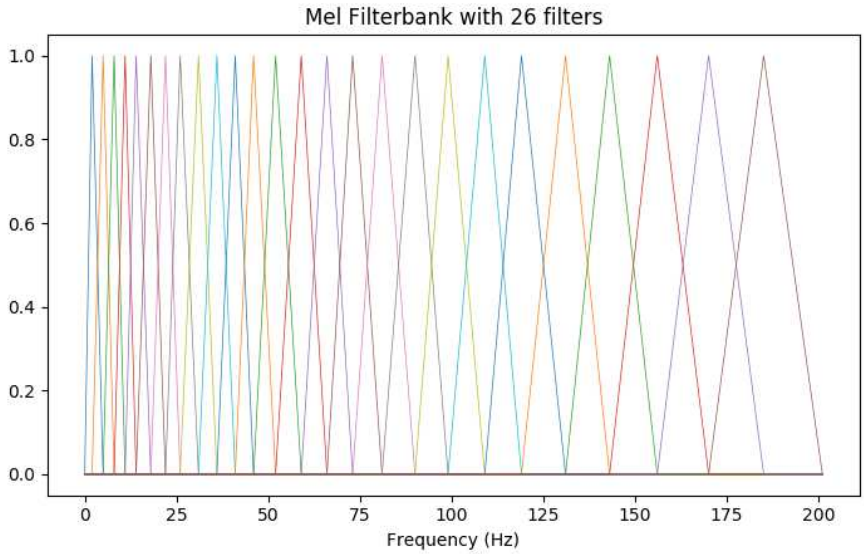


Figure 4. The mel filterbank applied to the spectrum in figure 3.

Filtering the power spectrum through the mel filterbank yielded to the mel frequency spectrum which contains energy readings that more closely represent what humans hear than the previous power spectrum. This is evident in the decreased energy readings at the higher frequencies shown in figure 5.

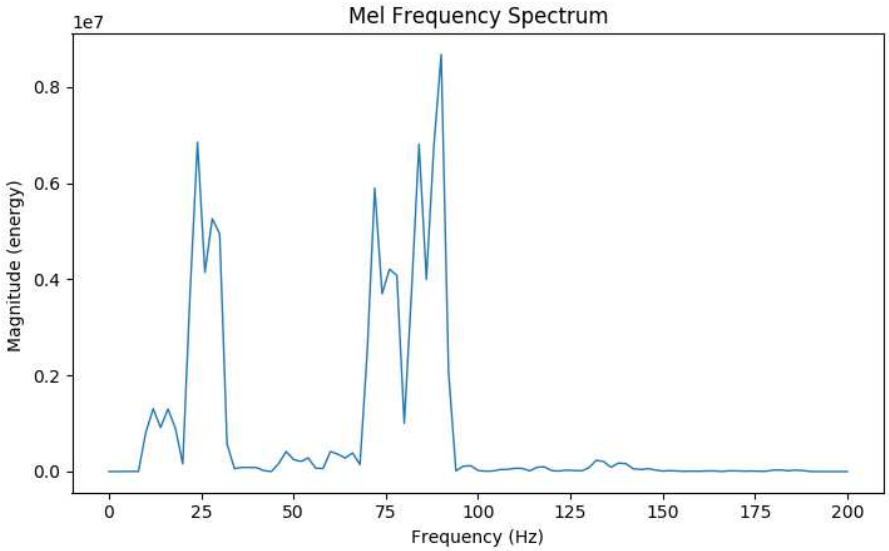


Figure 5. The mel frequency spectrum containing filterbank energies.

The mel frequency spectrum's logarithm was then computed as in figure 6. This is to further improve the features to be extracted as variation in energy levels has been proven to have little to no effect on how humans perceive sound [14]. Thus, taking the logarithm of the mel frequency spectrum still encapsulates the input signal accurately.

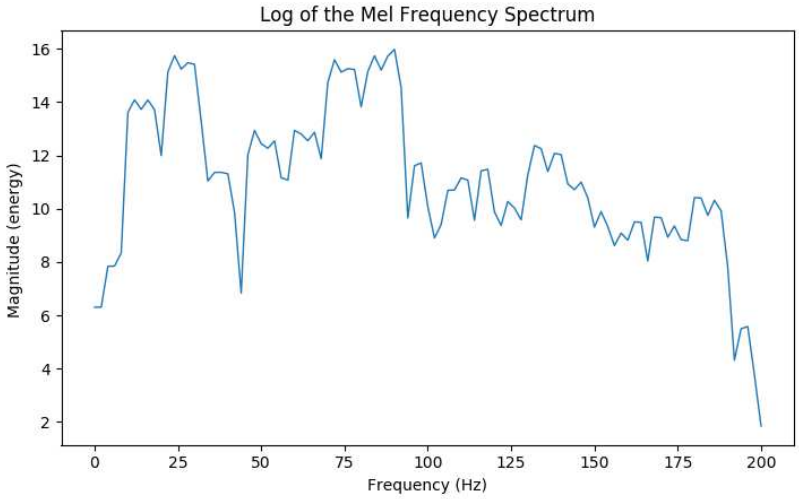


Figure 6. The logarithm of the mel frequency spectrum.

Finally, the Discrete Cosine Transform (DCT) algorithm was used on the log mel frequency values to get the MFCCs from the generated cepstrum. General implementations of MFCC extraction as in [15] result to 12 cepstral coefficients plus its energy coefficient, 13 delta cepstral coefficients, and 13 double delta or acceleration coefficients. Hence a total of 39 MFCCs can be used as features.

For this experiment, only the first 13 of the 39 MFCCs were extracted as feature vectors from each audio file; the same feature size used in the study of [16]. The vectors were then flattened by computing the mean of every feature across all frames so that each audio file can be represented as one row in the dataset. In the end, MFCC extraction resulted to a dataset of 110 files x 13 features before the ground truths of each file were appended, 110 files x 14 features after.

A summary of all the pre-processing steps done to facilitate training of the ANN is shown in figure 7.

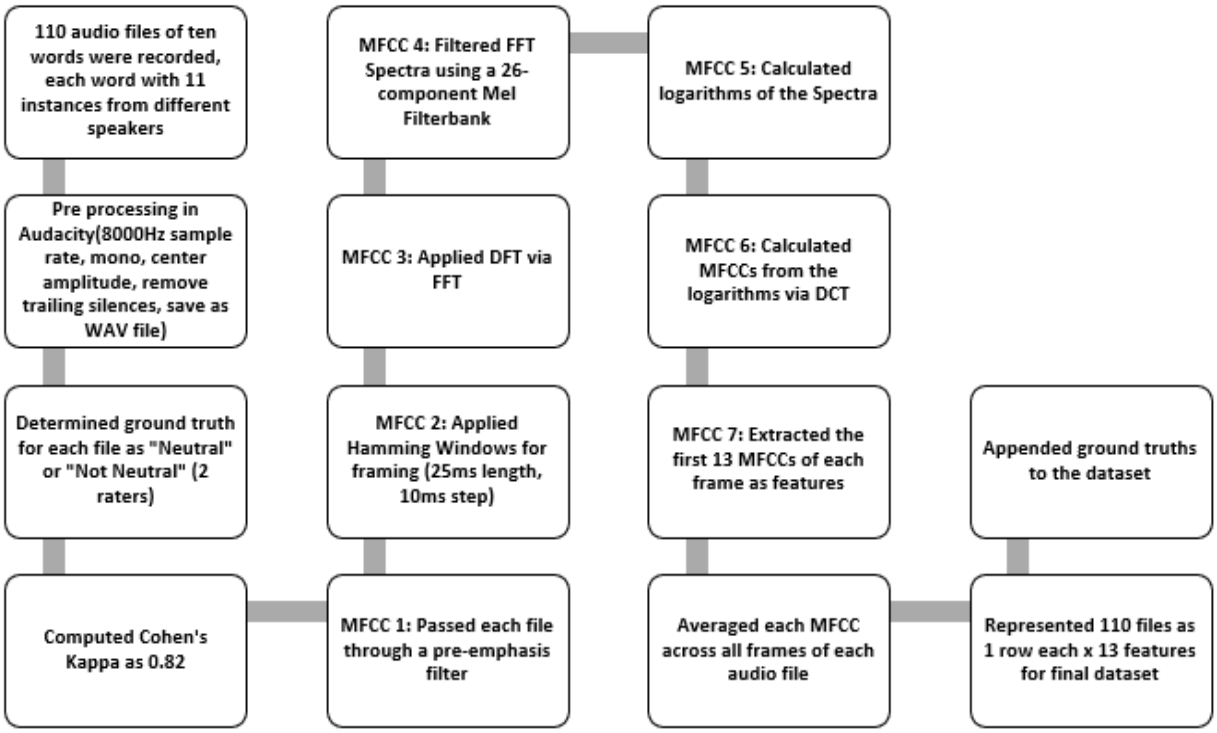


Figure 7. The pre-processing steps conducted in the study.

3.3 Training and Validation

Training and Validation was done through Sequential modelling in Keras. An ANN was created with an input layer of 13 nodes, a hidden layer of 110 nodes, a second hidden layer of 60 nodes and an output layer of 1 node as shown in figure 8.

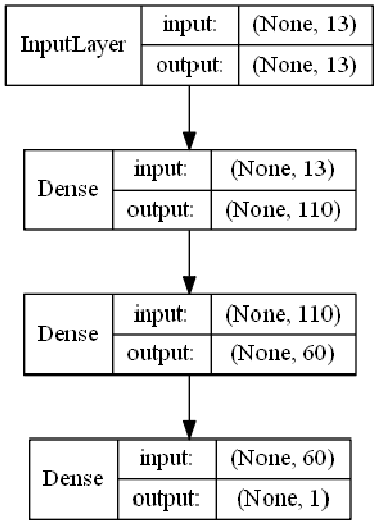


Figure 8. The ANN's architecture.

The figure describes the structure of the input and output for each layer with the notation (batch size, number of nodes). A "None" batch size as in the figure indicates that any batch size can be used during training for flexibility of experiments.

The model's structure amounts to 8,261 trainable parameters as shown in figure 9. The initial weights of each parameter were generated by the Random Normal kernel initializer in Keras. Rectified Linear Unit (ReLU) activation functions were used in the first two hidden layers of the model while a Sigmoid activation function was implemented on the output layer. The model was compiled with a Binary Cross-Entropy loss function and a Stochastic Gradient Descent optimizer.

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_1 (Dense)              (None, 110)               1540
_________________________________________________________________
dense_2 (Dense)              (None, 60)                6660
_________________________________________________________________
dense_3 (Dense)              (None, 1)                 61
=================================================================
Total params: 8,261
Trainable params: 8,261
Non-trainable params: 0
_________________________________________________________________
```

Figure 9. The total number of trainable parameters.

Stratified Ten-fold Cross Validation was used to prevent the model from overfitting during training with a batch size of 4 across 110 epochs. We also implemented a stopping condition using Keras callbacks for training to stop automatically when the minimum training loss for each fold has been reached. Accuracy of each fold was then computed and averaged to capture the ANN's overall performance. Furthermore, all misclassified files were identified for comparison with the dataset and confirm whether these files were among the files that the raters disagreed on.

4 Results and Discussion

Figure 10 shows a visualization of how the binary cross entropy loss function was minimized for both training and validation sets across all folds. Although there were noticeable fluctuations in the validation loss minimization, there were no observable deviances in the training loss minimization, and the model was still able to learn the desired weights accurately.

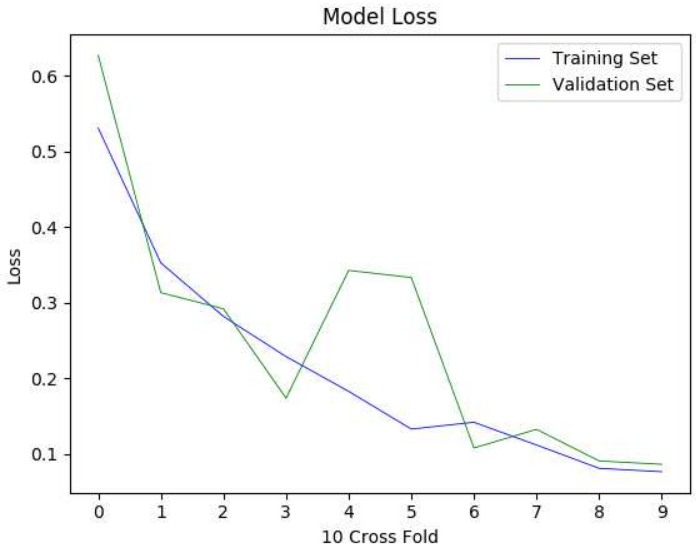


Figure 10. Binary cross entropy loss function minimization across 10 folds.

The ANN was able to correctly classify 33 out of 42 'Not Neutral' utterances and 65 out of 68 'Neutral' utterances as shown in the confusion matrix in table 2. In addition, only 1 out of the 12 misclassified files was among the utterances that had conflicting labels from the raters. This suggests that the model's errors were most likely due to the parameters not being learned properly during training and not based on the dataset's reliability.

Table 2: The ANN's confusion matrix

	Neutral	Not neutral
Neutral	33	9
Not neutral	3	65

The ANN has achieved an overall classification accuracy of 89.60% on the dataset, computed from the confusion matrix and the summary of training and validation results shown in table 3.

Table 3: Training and validation results (accuracy and loss)

Fold	Training accuracy	Validation accuracy	Training loss	Validation loss
1	0.7428	0.6625	0.5313	0.6276
2	0.8600	0.8571	0.3528	0.3134
3	0.8787	0.8484	0.2819	0.2920
4	0.9191	0.9272	0.2287	0.1737
5	0.9393	0.8636	0.1827	0.3428
6	0.9747	0.8484	0.1328	0.3333

7	0.9595	0.9696	0.1418	0.1077
8	0.9730	1.0	0.1117	0.1324
9	0.9833	0.9833	0.0808	0.0906
10	0.9775	1.0	0.07635	0.0861
Average:	0.92	0.8960		

In addition, F-Scores were also computed in every fold to back the model's reported accuracy as given by

$$H = \frac{2x_1 x_2}{x_1 + x_2}. \tag{2}$$

where H is the harmonious mean or F1 Score, x_1 is precision, and x_2 is recall. The average F-Score computed was 0.92 across all folds as shown in table 4, supporting the classifier's overall performance.

Table 4: Training and validation results (precision, recall, and f-score)

Fold	Precision	Recall	F-score
1	0.6792	0.8571	0.7504
2	0.8602	0.9183	0.8835
3	0.8101	1.0	0.8944
4	0.9250	0.9714	0.9446
5	0.8660	0.9285	0.8952
6	0.8312	0.9761	0.8940
7	1.0	0.9523	0.9743
8	1.0	1.0	1.0
9	1.0	0.9722	0.9848
10	1.0	1.0	1.0
Average:			0.9221

These findings show that the model can be expected to do well in classifying future data even without a pronunciation lexicon or dictionary used during training. In addition, the findings proved that the Stratified Ten-Fold Cross Validation can still yield good results despite the dataset having a slightly higher count of 'Neutral' utterances.

5 Conclusion

In this study, an ANN was developed to classify the neutrality of call center agents' pronunciations and develop an objective standard that a company can use for assessing their employees' or applicants' pronunciations using a small dataset of speech recordings. After ensuring the reliability of the dataset, training, and validation, the ANN achieved an accuracy of 89.60% in detecting whether utterances of 10 specific words are 'Neutral' or 'Not Neutral'. This accuracy was supported by an average F-Score of 0.92.

Therefore, the model can be expected to accurately serve as a standard that caters specifically to the call center's requirements as far as pronunciation assessment of specific words is concerned. It is important to note

however, that this performance can only be expected on new utterances that fit the context of classifying pronunciation neutrality definitive to the call center involved. Varied results may be observed if the model is tested outside of this context. Consequently, this allows for the use of the model strictly in assessing pronunciations within the call center where the dataset was collected from.

Results have also shown that the use of a standard ANN or Multilayer Perceptron for speech classification is provably effective when working with a relatively small dataset. Although a difficulty arose with the full utilization of the extracted MFCC features as standard ANNs have fixed inputs. This was addressed by flattening the MFCC vectors across all frames per individual audio file, but theoretically, other Neural Network architectures that can handle variable-length inputs and time-series data could outperform the standard ANN model and is therefore recommended for future work.

Other recommendations include creating a larger dataset with more raters for kappa computation. The use of deeper neural networks is also recommended as well as comparing the performance of different neural network architectures and feature extraction techniques.

Acknowledgements

The researchers would like to thank Jose Bien B. Tejo for facilitating the recording of audio samples from members of his team.

References

[1] Lockwood, J., Forey, G., & Price, H. (2008). English in Philippine call centers and BPO operations: Issues, opportunities and research. In *Philippine English: Linguistic and Literary* (pp. 219-241). Hong Kong University Press, HKU. doi: 10.5790/hongkong/9789622099470.003.0012

[2] Foote, J. A. and Trofimovich, P. (2017). Second language pronunciation learning: an overview of theoretical perspectives. In *The Routledge Handbook of Contemporary English Pronunciation* (pp. 93-108). doi: 10.1002/9781118346952.ch20

[3] Wotschke, I. (2014). *How Educated English Speak English: Pronunciation as Social Behaviour* (pg. 165). Frank & Timme.

[4] Copeland, J. E., Fries, P. H., Lockwood, D. G. (2000). *Functional approaches to language, culture, and cognition* (pg. 515). John Benjamins Publishing Company. doi: 10.1075/cilt.163

[5] Hu, W., Qian, Y., Soong, F. K., Wang, Y. (2015). Improved mispronunciation detection with deep neural network trained acoustic models and transfer learning based logistic regression classifiers. In *Speech Communication, Volume 67* (pp. 154-166). European Association for Signal Processing (EURASIP) and International Speech Communication Association (ISCA). doi: 10.1016/j.specom.2014.12.008

[6] Gao, Y., Xie, Y., Cao, W., Zhang J. (2015). A study on robust detection of pronunciation erroneous tendency based on deep neural network. In *INTERSPEECH-2015* (pp. 693-696).

[7] Rutherford, A. T., Peng, F., Beaufays, F. (2014). Pronunciation learning for named-entities through crowd-sourcing. In *INTERSPEECH-2014* (pp. 1148-1452).

[8] Fleiss, J. L., & Cohen, J. (1973). The equivalence of weighted kappa and the intraclass correlation coefficient as measures of reliability. *Educational and Psychological Measurement 33* (pp.613-619). doi: 10.1177/001316447303300309

[9] McHugh, M.L. (2012). Interrater reliability: the kappa statistic. *Biochemia Medica 22* (pp.276-282). doi: 10.11613/bm.2012.031

[10] Sung, Y.H. and Jurafsky, D. (2009). Hidden conditional random fields for speech recognition. *IEEE Workshop on Automatic Speech Recognition & Understanding*. doi: 10.1109/asru.2009.5373329

[11] Zheng, F., Zhang G., Song, Z. (2001). Comparison of different implementations of MFCC. *Journal of Computer Science and Technology. Volume 16, Issue 6* (pp. 582-589). doi: 10.1007/bf02943243

[12] Mecklenbrauker, W. (1989). A tutorial on non-parametric bilinear time-frequency signal representations. In *Time and Frequency representation of signals and systems* (pg.12). International Centre for Mechanical Sciences. Springer. doi: 10.1007/978-3-7091-2620-2_2

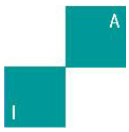
[13] Tkachenko, M., Yamshinin, A., Kotov, M., Nasatasenko, M. (2017). Speech Enhancement for Speaker Recognition Using Deep Recurrent Neural Networks. In *Lecture Notes in Computer Science* (pp. 690-696). doi: 10.1007/978-3-319-66429-3_69

[14] Malmierca, M. S., Irvine, D. R. (2005). Auditory Spectral Processing. In *International Review of Neurobiology* (pg.129). Elsevier Academic Press. doi: 10.1016/s0074-7742(05)70015-5

[15] Alshutayri, A. and Albarhamtoshy, H. (2011). Arabic Spoken Language Identification System (ASLIS): A Proposed System to Identifying Modern Standard Arabic (MSA) and Egyptian Dialect. In Communications in Computer and Information Science on (pp375 to 385). Springer. doi: 10.1007/978-3-642-25453-6_33

[16] Hirsch, H.G., Pearce, D., Deutschland, Ericsson, E. (2000). The Aurora Experimental Framework for the performance evaluation of speech recognition systems under noisy conditions. In ASR2000 - Automatic Speech Recognition: Challenges for the new Millennium Paris, France September 18-20, 2000 Proceedings.

Inteligencia Artificial 21(62) (2018), 145-158
doi: 10.4114/intartif.vol21iss62pp114-158



INTELIGENCIA ARTIFICIAL

http://journal.iberamia.org/

Learning with ensembles from non-stationary data streams

Aprendiendo con ensambles a partir de flujos de datos no estacionarios

Alberto Verdecia-Cabrera[1], Isvani Frías-Blanco[2], Luis Quintero-Domínguez[3], Yanet Rodríguez Sarabia[4]

[1]Universidad de Granma
Centro de Investigaciones de la Informática, Universidad Central "Marta Abreu" de Las Villas, Cuba
averdeciac@gmail.com

[2]LexisNexis Risk Solutions. Sao Paulo, Brazil
justisvani@gmail.com

[3]Universidad de Sancti Spíritus, Cuba
lquintero@uclv.cu

[4]Universidad Central "Marta Abreu" de Las Villas, Cuba
yrsarabia@uclv.edu.cu

Abstract Nowadays many sources generate massive data continuously, without control of the arrival order and at high speed. Internet, cell-phones, cars, and security sensors are examples of such sources. Because of the temporal dimension of the data (they are constantly arriving over time), and the dynamism of many real-world situations, the target function to be learned can change over time. This situation, known as concept drift, complicates the task of estimating this target function, because a previous learning model can become outdated, or even contradictory regarding the most recent data. There are several algorithms to manipulate concept drift, and among them are the classifier ensembles. In this article, we present a new ensemble algorithm called ADCE, able for learning from data streams with concept drift. ADCE manipulates these changes using a change detector in each base classifier. When the detector estimates a change, the classifier in which the change was estimated is replaced by a new one. ADCE combines the simplicity of the bagging algorithm to train base classifiers with methods used in batch learning to combine the output of the base classifiers. The proposed algorithm is compared empirically with several bagging family ensemble algorithms for data streams. The experimental results show that the proposed algorithm constitutes a viable option for learning from concept drifting data streams.

Resumen En la actualidad muchas fuentes generan grandes volúmenes de datos constantemente, sin control del orden de llegada y a altas velocidades. Internet, teléfonos celulares, automóviles y sensores de seguridad son ejemplos de tales fuentes. Debido a la dimensión temporal de los datos (que están llegando constantemente a lo largo del tiempo) y al dinamismo de muchas situaciones del mundo real, la función objetivo que se debe aprender puede cambiar con el tiempo. Esta situación, conocida como cambio de concepto, complica la tarea de estimar esta función objetivo, porque un modelo de aprendizaje anterior puede quedar desactualizado o incluso contradictorio con respecto a los datos más recientes. Existen varios algoritmos para manipular cambios de concepto, entre los cuales se encuentran los ensambles de clasificadores. En este artículo se presenta un nuevo algoritmo de ensamble llamado ADCE, capaz de aprender a partir de flujos de datos con cambios de concepto. ADCE manipula estos cambios utilizando un detector de cambios en cada clasificador base. Cuando el detector estima un cambio, el clasificador en que se estimó el cambio es remplazado por uno nuevo. ADCE combina la simplicidad del algoritmo

ISSN: 1988-3064(on-line)

bagging para entrenar clasificadores base con métodos utilizados en el aprendizaje por lotes para combinar la salida de los clasificadores base. El algoritmo propuesto se compara empíricamente con varios algoritmos de ensamble de la familia bagging para flujos de datos. Los resultados experimentales muestran que el algoritmo propuesto constituye una opción viable para el aprendizaje de flujos de datos con cambios de concepto.

Keywords: Data stream, classifier ensemble, concept drift.
Palabras Clave: Flujos de datos, ensambles de clasificadores, cambio de concepto.

Introducción

En la actualidad, el volumen de los datos generados por sensores, Internet, dispositivos de localización, telefonía y muchos otros, está en constante aumento. El tamaño de estos datos es potencialmente infinito debido a su constante generación, por lo que es necesario procesarlos con recursos limitados de cómputo. Para este procesamiento es factible el uso de técnicas de aprendizaje automático. Dependiendo de cómo se presenten los ejemplos de entrenamiento, el aprendizaje automático se puede clasificar en dos tipos [26]: aprendizaje por lotes y aprendizaje en línea. En el aprendizaje por lotes, los datos deben ser recolectados y almacenados antes de ser procesados, por lo que su uso no es factible para procesar datos generados constantemente en el tiempo. Sin embargo, el aprendizaje en línea permite procesar flujos de datos de manera secuencial [17].

En las tareas de clasificación, un flujo de datos es comúnmente definido como una secuencia muy grande (potencialmente infinita) de pares que se van adquiriendo a lo largo del tiempo. Estos pares, llamados instancias o ejemplos, están compuestos por un conjunto de atributos y una etiqueta de clase. Debido a la dimensión temporal de los datos (estos son adquiridos en el tiempo) y la dinámica de muchas situaciones reales, la distribución de probabilidad que regula a los mismos (también llamada concepto) puede cambiar con el tiempo, un problema conocido comúnmente como cambio de concepto. Consecuentemente, los algoritmos de aprendizaje para la minería de flujos de datos deben ser actualizados con respecto a los conceptos más recientes [16].

Los métodos de ensambles de clasificadores han recibido en los últimos tiempos gran atención para el modelado y la clasificación de flujos de datos no estacionarios [5]. Los métodos de ensambles combinan las predicciones de los clasificadores base con el objetivo de mejorar la precisión obtenida por estos clasificadores de forma individual. Con el fin de manipular cambios de concepto se utilizan medidas de rendimiento para monitorizar la consistencia del ensamble en relación con los nuevos datos. Variaciones significativas en los valores de rendimiento se interpretan como un cambio de concepto y los métodos de ensamble eliminan, reactivan o añaden nuevos clasificadores base dinámicamente en respuesta a estas variaciones.

El mecanismo de adaptación a los cambios de concepto de varios algoritmos de ensambles propuestos en la literatura consiste en eliminar el peor clasificador base cuando se detecta un cambio, seguido por el entrenamiento de un nuevo clasificador base y su posterior inclusión dentro del ensamble [3, 4]. Por lo que, estos algoritmos solo manipulan cambios de concepto a nivel de ensamble y no en los clasificadores base. El peor clasificador en el ensamble puede no ser el afectado por el cambio de concepto actual. Por lo tanto, la estrategia adoptada por algoritmos de ensambles anteriores puede conducir a una adaptación ineficaz a los cambios. Además, en el aprendizaje en línea, los métodos para combinar las predicciones de los clasificadores base se han enfocado en el voto ponderado. Sin embargo la relación subyacente entre las predicciones de los clasificadores y la clase verdadera puede ser más compleja que una combinación lineal de las predicciones. Por lo que en este trabajo se utilizan otros métodos que han sido utilizados en el aprendizaje por lotes para combinar las salidas de los clasificadores base.

En este trabajo se presenta un nuevo algoritmo que combina la simplicidad del algoritmo Bagging [28] para entrenar clasificadores en flujos de datos, con métodos utilizados en el aprendizaje por lotes para combinar clasificadores y con un nuevo mecanismo de adaptación a los cambios de concepto. Para detectar los cambios de concepto en los clasificadores base se utilizó HDDM (Hoeffding Drift Detection Method) [12] como detector de cambios de concepto y estimador de error con el objetivo de emitir tres señales diferentes de cambio durante el proceso de aprendizaje. HDDM emite la señal *en-control* cuando el concepto actual permanece estable, *aviso* cuando es probable que se aproxime un cambio, y *fuera-de-control* cuando se detecta el cambio. Entonces, cuando el detector estima una cambio, se sustituye el clasificador en el que se detectó el cambio por uno nuevo.

El resto de este artículo está estructurado como sigue. Primero, en la Sección 1 se presentan las definiciones fundamentales y tipos de cambios de concepto más comunes. Luego, en la Sección 2 se describen varios algoritmos basados en ensambles de clasificadores. Posteriormente en la Sección 3 se describen los métodos utilizados en este trabajo para combinar clasificadores y en la Sección 4 se describe el algoritmo propuesto. En la Sección 5 se describe la configuración del estudio empírico para evaluar el rendimiento del algoritmo propuesto y se muestran y discuten los resultados obtenidos sobre datos sintéticos y datos reales, los cuales demuestran la validez del algoritmo propuesto. Finalmente en la Sección 6 se presentan las conclusiones de este artículo.

1. Cambio de concepto

Dentro del aprendizaje en línea [33], el problema de clasificación se define generalmente para una secuencia (posiblemente infinita) de ejemplos $S = e_1, e_2, ..., e_i...$ que se obtienen en el tiempo, normalmente uno a la vez y no necesariamente dependientes del tiempo. Cada ejemplo de entrenamiento $e_i = (\vec{a_i}, c_i)$ está formado por un vector $\vec{a_i}$ y un valor discreto c_i; donde cada vector $\vec{a_i} \in \vec{\mathcal{A}}$ tiene las mismas dimensiones y cada dimensión se llama atributo. El valor discreto c_i es llamado etiqueta y tomado de un conjunto finito de clases $\mathcal{C}$.

Concepto se refiere a la función de distribución de probabilidad del problema en un punto determinado en el tiempo [23]. Este concepto puede ser caracterizado por la distribución de probabilidad conjunta $P\left(\vec{\mathcal{A}}, \mathcal{C}\right)$ donde $\vec{\mathcal{A}}$ representa los atributos y $\mathcal{C}$ es la clase. Por tanto un cambio en la distribución del problema (también conocido como contexto) [15] implica un cambio de concepto. Gama y otros [16] distinguen dos tipos fundamentales de cambios de concepto que están presentes en muchos problemas reales:

1. *Cambios de concepto reales:* se refieren a cambios en la distribución de probabilidad a posteriori de las clases $p(\mathcal{C} \mid \mathcal{A})$. Estos cambios pueden ocurrir sin que ocurran cambios en la distribución de los datos de entrada $p(\mathcal{A})$.

2. *Cambios de concepto virtuales:* ocurren si la distribución de los datos de entrada cambia (es decir $p(\mathcal{A})$ cambia) sin afectar $p(\mathcal{C} \mid \mathcal{A})$.

Los tipos de cambio de concepto también pueden clasificarse en cuanto a su extensión (*o tasa de cambio*) [21, 18, 19, 20]: cambios locales y globales. En cambios locales, la distribución cambia solo en una región limitada del espacio de instancias. En el caso de los cambios globales, la distribución cambia sobre toda la región del espacio de instancias, es decir, para todos los valores posibles de las clases y los atributos. Otro aspecto importante a considerar es el período de transición entre conceptos consecutivos (*velocidad del cambio*), y en este caso tenemos cambios abruptos y graduales. Los cambios abruptos ocurren cuando la transición entre conceptos consecutivos es instantánea y los cambios graduales ocurren cuando el período de transición contiene cierto número de ejemplos. Otro tipo común de cambio es el recurrente, que ocurre cuando los conceptos pueden reaparecer [27].

2. Trabajos Relacionados

Los algoritmos basados en ensambles de clasificadores combinan la predicción de los clasificadores base y se pueden clasificar por la forma en que actualizan sus clasificadores base. La primera clasificación está dada por la utilización de bloques de instancias. Los ensambles basados en bloques de instancias dividen el flujo de datos en pequeños bloques de igual tamaño y entrenan clasificadores con cada uno de estos bloques. La adaptación a los cambios de concepto de estos algoritmos generalmente es lenta porque tienen que esperar a que se llene el bloque para actualizar los clasificadores base. La segunda clasificación de los ensambles es por la actualización de los clasificadores de forma incremental. Estos métodos utilizan como clasificadores base clasificadores incrementales, es decir actualizan los modelos en la medida que se obtienen las instancias. Los métodos propuestos en este artículo están dentro de esta clasificación.

2.1. Ensambles que utilizan bloques de instancias

Uno de los primeros algoritmos de ensamble para el procesamiento de flujos de datos fue SEA (*Streaming Ensemble Algorithm*) [31]. SEA crea los clasificadores base a partir de pequeños subconjuntos de los datos, leídos secuencialmente en bloques de un tamaño fijo. El tamaño de esos bloques es un parámetro importante porque es responsable del equilibrio entre precisión y flexibilidad. Cada bloque se utiliza para entrenar un nuevo clasificador, el cual es comparado con los demás miembros del ensamble. Si alguno de los miembros es peor que el nuevo clasificador, éste se sustituye por el nuevo. Para evaluar los clasificadores, Street y Kim proponen utilizar la precisión de clasificación obtenida en el bloque de datos más reciente. El algoritmo cuenta con un límite máximo de clasificadores que actúa como mecanismo de adaptación. Al ser alcanzado este límite máximo obliga al algoritmo sustituir a clasificadores base anteriores siguiendo cierto criterio de reemplazo. Para combinar las predicciones de los clasificadores base utiliza voto mayoritario y como clasificador base utiliza el algoritmo C4.5. SEA presenta problemas para adaptarse a los cambios de conceptos abruptos; en estos resultados influye el mecanismo de votación utilizado ya que los clasificadores dejan de aprender una vez que son creados.

Bajo el mismo esquema de entrenamiento de SEA, Wang et al. [32] propusieron el método AWE (*Accuracy Weighted Ensemble*). AWE combina las respuestas de los clasificadores base a través del voto mayoritario ponderado. La ponderación de los clasificadores base está en función de la precisión obtenida por los mismos, al utilizar como instancias de prueba las propias instancias del bloque de entrenamiento actual. Al igual que SEA, su rendimiento frente a cambios de conceptos graduales es aceptable, pero esto no ocurre así cuando los cambios son abruptos. Eso se debe fundamentalmente a que AWE espera al próximo bloque de entrenamiento para actualizar los pesos de los clasificadores base.

BWE (*Batch Weighted Ensemble*) [9] también divide el conjunto de entrenamiento en bloques de igual tamaño y utiliza el voto mayoritario ponderado para combinar la salida de los clasificadores base. A diferencia de los métodos anteriores, este incorpora un detector de cambios al algoritmo, BDDM (*Batch Drift Detection Method*), y utiliza un modelo de regresión para estimar cambios de concepto. El detector de cambios se utiliza básicamente para determinar cuando crear un nuevo clasificador base, debido a los cambios de concepto o si el concepto permanece estable el ensamble no varía. La idea fundamental de esta propuesta es combinar la capacidad de los ensambles para adaptarse a los cambios graduales con el detector de cambios para manipular cambios abruptos.

La mayoría de los ensambles que actualizan los clasificadores base mediante bloques presentan problemas para adaptarse a los cambios de concepto, esto se debe a que no tienen mecanismos explícitos para detectar cambios y generalmente esperan a que se complete un bloque de instancias para actualizar los clasificadores base.

2.2. Ensambles incrementales

Bagging [6] y Boosting [11] son dos de los algoritmos más conocidos para entrenar clasificadores base. Bagging aplica muestreo con remplazamiento al conjunto de datos original para crear M conjuntos de entrenamientos del mismo tamaño que el original y crea M clasificadores base con estos conjuntos de entrenamiento. A diferencia de Bagging, Boosting entrena secuencialmente un conjunto de clasificadores con instancias ponderadas en el cual el peso asociado a cada instancia depende del desempeño del clasificador anterior. El objetivo de Boosting es asignarle mayor peso a las instancias mal clasificadas.

Oza y Rusell [28] propusieron versiones incrementales de los algoritmos Bagging y Boosting, pero estas propuestas no tienen mecanismos para adaptarse a los cambios de concepto. Así, su adaptación al cambio depende del algoritmo de clasificación utilizado para generar los clasificadores base. Para manipular cambios de concepto, Bifet y otros [4] propusieron un nuevo algoritmo basado en Bagging llamado OzaBagAdwin. La idea de esta variante es agregar a la versión incremental del algoritmo Bagging el detector de cambios de concepto ADWIN (*Adaptive Windowing*) [1]. El mecanismo de adaptación utilizado se basa en la sustitución del peor de los clasificadores, cuando se estima un cambio, por un nuevo clasificador base creado más recientemente. Otro ensamble basado en Bagging es DDD (*Dealing with Diversity for Drifts*) [24]. DDD alterna entre dos estados: (1) antes de la detección del cambio y (2) después de la detección del cambio. DDD varía la diversidad del ensamble con el objetivo de adaptarse a los cambios de concepto rápidamente.

3. Métodos para combinar clasificadores

La manera en que se combinan los resultados de los clasificadores individuales es una cuestión fundamental en el estudio de los ensambles (combinaciones) de clasificadores. La idea aquí es optimizar la decisión que se va a tomar a partir de las decisiones individuales. Múltiples métodos de combinación de clasificadores han sido utilizados con éxito en el aprendizaje por lotes tradicional. En este trabajo se seleccionaron tres de los métodos más conocidos para la combinación de clasificadores base en el aprendizaje a partir de flujos de datos no estacionarios. Además, se propone el uso del método de control estadístico de procesos EWMA *(Exponentialy Weighted Moving Average)* ya que permite asignar más o menos importancia a la precisión actual y al peso anterior del clasificador. A continuación se describen los métodos de combinación empleados.

Naıve Bayes (NB)

La regla de Bayes ha tenido múltiples aplicaciones en la teoría de las probabilidades y en el aprendizaje automático. Usando la regla de Bayes, se puede extender la idea de Naive Bayes para combinar varios clasificadores [29] de la siguiente manera:

$$class\,(x) = \underset{\substack{c_j \in dom(y) \\ P(y,=c_j)>0}}{\operatorname{argmax}} \hat{P}\,(y = c_j) \times \prod_{k=1} \frac{\hat{P}_{M_k}\,(y = c_j|x)}{\hat{P}\,(y = c_j)} \tag{1}$$

donde M_k denota al clasificador k y $\hat{P}_{M_k}\,(y = c_j|x)$ denota la probabilidad de que y obtenga el valor c_j dada una instancia x.

EWMA

EWMA (*Exponentialy Weighted Moving Average*) es un método de control estadístico de procesos. La característica esencial de este método es que otorga menos peso a las observaciones cuanto más alejadas están en el tiempo. El estadístico que se representa en el gráfico es:

$$Ewma_{M_k} \rightarrow W_{M_k} = \beta x_{M_k} + (1 - \beta)\, W_{M_k} \tag{2}$$

donde x_{M_k} es una variable aleatoria , β es una constante para determinar el peso de las observaciones, $(0 < \beta < 1)$.

En este trabajo, la ecuación 2 se utiliza para determinar el peso de los clasificadores base. En este caso, W_{M_k} sería el peso de cada clasificador y x_{M_k} la precisión actual. Como cada clasificador utiliza un detector de cambio, se puede estimar en cualquier momento el error ϵ_m de cada clasificador base, por lo que la precisión sería $1 - \epsilon_m$. La constante β preestablecida representa el nivel de importancia que se les otorga al funcionamiento de los clasificadores base frente a los datos antiguos y frente a los datos actuales respectivamente. Un valor elevado de β significa que se le dará más importancia al funcionamiento histórico del clasificador que al funcionamiento frente a los datos actuales; la adaptación al cambio será un poco más lenta pero el proceso será más robusto frente a datos ruidosos.

Entropy Weighting (EW)

La idea de este método de combinación es asociarle a cada clasificador un peso que es inversamente proporcional a la entropía de su vector de clasificación [29]:

$$class\,(x) = \underset{c_i \in dom(y)}{\operatorname{argmax}} \sum_{k:c_i = \underset{c_j \in dom(y)}{\operatorname{argmax}}\, \hat{P}_{M_k}(y=c_j|x)} Ent\,(M_k, x) \tag{3}$$

donde:

$$Ent\,(M_k, x) = -\sum_{c_j \in dom(y)} \hat{P}_{M_k}\,(y = c_j|x) \times \log\left(\hat{P}_{M_k}\,(y = c_j|x)\right) \tag{4}$$

permite calcular la entropía para el clasificador M_k y la instancia x.

Dempster–Shafer (DS)

La idea de usar la teoría de la evidencia de Dempster–Shafer para combinar modelos fue propuesta por [30]. Este método utiliza la noción de asignación de probabilidades básica definida para una cierta clase c_i dada la instancia x:

$$bpa\left(c_i, x\right) = 1 - \prod_k \left(1 - \hat{P}_{M_k}\left(y = c_i | x\right)\right) \tag{5}$$

En consecuencia, la clase seleccionada es la que maximiza el valor de la función:

$$Bel\left(c_i, x\right) = \frac{1}{A} \times \frac{bpa\left(c_i, x\right)}{1 - bpa\left(c_i, x\right)} \tag{6}$$

donde A es un factor de normalización definido como:

$$A = \sum_{\forall c_i \in dom(y)} \frac{bpa\left(c_i, x\right)}{1 - bpa\left(c_i, x\right)} + 1 \tag{7}$$

Algoritmo 1: ADCE

Entrada:
ejemplos: ejemplos de entrenamiento con etiquetas de clase $y_i \in Y = 1, \ldots, L$
K: tamaño del ensamble
Comb: método de combinación de clasificadores
Salida :
$\mathfrak{C}\left(x\right)$ = combinar los clasficadores base M_k utilizando *Comb* para todo $k \in \{1, 2, \ldots, K\}$

```
1 inicio
2     Inicializar los clasficadores base M_k, detectores D_k para todo k ∈ {1, 2, ..., K}
3     para cada ejemplo de entrenamiento hacer
4         para k ← 1 hasta K hacer
5             K ← P(λ)    // P(λ) es la distribución de Poisson con media λ
6             para j ← 1 hasta K hacer
7                 Actualizar M_k y D_k con el ejemplo actual
8             si D_k estima un cambio entonces
9                 reiniciar M_k y D_k
```

4. Algoritmo propuesto

El algoritmo propuesto llamado ADCE utiliza la versión en línea del algoritmo Bagging para entrenar los clasificadores base y uno de los métodos de combinación de clasificadores descritos en la sección anterior para determinar los pesos de los clasificadores. ADCE manipula cambios de concepto de una manera simple y eficiente (ver algoritmo 1). Cada clasificador base M_k $(1 < k < K)$ utiliza un detector de cambios $\mathfrak{D}_k$ para estimar la tasa de error ϵ_k de cada clasificador. Cuando $\mathfrak{D}_k$ estima un cambio, el clasificador M_k es remplazado por uno nuevo. Para combinar la salida de los clasificadores base se puede utilizar cualquiera de los métodos mencionados anteriormente. El algoritmo solo recibe como parámetros el detector de cambios y el número de clasificadores base.

La tasa de error de los clasificadores base se monitoriza constantemente a medida que llega cada ejemplo de entrenamiento. Por lo tanto, este seguimiento también debe realizarse con recursos computacionales controlados. En los últimos años se han propuesto varios métodos en la comunidad estadística para detectar cambios en línea [25]. Sin embargo, estos suponen que los datos de entrada están regulados por una distribución de probabilidad conocida. ADCE utiliza HDDM (*Hoeffding Drift Detection Method*)

[12] como detector de cambios de concepto y estimador de error. HDDM procesa cada valor entrante con una complejidad computacional constante y proporciona garantías matemáticas para las tasas de falsos positivos y falsos negativos.

Tabla 1: Principales características de los conjuntos de datos utilizados en los experimentos.

Conjunto de Datos	Acrónimo	Eejemplos	Nominal	Numérico	Valores perdidos	Clases
LED display	LED	1,000,000	24	0	no	10
SEA	SEA	1,000,000	0	3	no	2
Radial base functions	RBF	1,000,000	0	10	no	2
Waveform	WAV	1,000,000	0	40	no	3
Agrawal	AGR	1,000,000	3	6	no	2
Stagger	STA	1,000,000	3	0	no	2
Hyperplane	HYP	1,000,000	0	10	no	2
Usenet 1	USE1	1,500	100	0	no	2
Usenet 2	USE2	1,500	100	0	no	2
Segment	SEG	2,310	0	19	no	7
Mushroom	MUS	8,124	22	0	yes	2
Spam	SPA1	4,601	1	57	mo	2
Spam corpus 2	SPA2	9,323	500	0	no	2
Nursery	NUR	12,960	8	0	no	5
EEG Eye State	EYE	14,980	0	14	no	2
Weather	WEA	18,159	0	8	no	2
Bank marketing	BAN	41,188	9	7	no	2
Electricity	ELE	45,312	1	7	yes	2
Connect-4	CON	67,557	21	0	no	3
KDD Cup 10 %	KDD	494,021	7	34	no	2
Forest Cover	COV	581,012	44	10	no	7
Poker Hand	POK	1,000,000	10	0	no	10

5. Evaluación experimental

Esta sección describe la evaluación experimental realizada al algoritmo propuesto con respecto a los algoritmos basados en Bagging más relevantes encontrados en la literatura. El algoritmo propuesto tiene cuatro variantes, una por cada método de combinación, en lo adelante ADCE+NB, ADCE+WMV, ADCE+EW y ADCE+DS. Por lo que, el estudio experimental tiene dos objetivos fundamentales. El primero es seleccionar la mejor variante del algoritmo propuesto, ya que éste utiliza diferentes métodos para combinar la salida de los clasificadores base. El segundo objetivo es comparar la mejor variante con los algoritmos más conocidos basados en Bagging. Para realizar esta comparación todos los algoritmos se evaluaron en presencia de cambios de concepto (abruptos y graduales) y frente a conjuntos de datos reales.

5.1. Configuración de los experimentos

El rendimiento de los algoritmos considerados se evaluó utilizando los conjuntos de datos más utilizados en la literatura. La Tabla 1 resume sus principales características. Los conjuntos de datos presentan ruido, atributos irrelevantes, atributos nominales y numéricos, y diferentes tipos de funciones objetivo. Como se muestra en la Tabla 1, los algoritmos se evaluaron frente a 15 conjuntos de datos de problemas reales

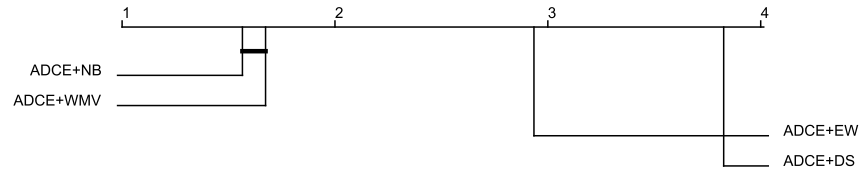


Figura 1: Ranking de los algoritmos. Los algoritmos que no son significativamente diferente están conectados.

y 7 conjuntos de datos de problemas artificiales. Los conjuntos de datos artificiales permiten evaluar los algoritmos controlando los conceptos estables, cambios abruptos, graduales; los datos artificiales también tienen la ventaja de modelar escenarios donde los algoritmos pueden demostrar su desempeño.

En los conjuntos de datos artificiales, se generaron 10 cambios de concepto para cada experimento. Los cambios se generaron cada 25,000 instancias. Los cambios graduales se simularon usando un período de transición entre conceptos consecutivos. Ese período de transición consistió en instancias de entrenamiento con diferentes funciones objetivos. El número de instancias utilizados varió entre experimentos para evaluar el comportamiento de los algoritmos con diferentes tasas de cambio. Más específicamente, se usaron 100, 500, y 1000 instancias de entrenamiento en períodos de transición.

Todos los experimentos fueron ejecutados en MOA [2], una herramienta para evaluar los algoritmos que aprenden a partir de flujos de datos. MOA provee una gran colección de herramientas de evaluación, varios algoritmos de aprendizaje y métodos para generar flujos de datos artificiales con la posibilidad de incluir cambios de concepto. Para la experimentación se seleccionaron los algoritmos basados en Bagging, tales como, OzaBag (versión en línea del algoritmo bagging) [28], OzaBagAdwin [4] y LeveragingBag [3]. Los algoritmos fueron utilizados con la configuración por defecto en MOA. En el detector de cambio HDDM, el nivel de significación para el cambio fue de $0,001$ y para la alerta de $0,005$. Se utilizaron 10 clasificadores base en todos los algoritmos. Como clasificador base se eligió Naive Bayes por ser uno de los algoritmos más exitosos para aprender de los flujos de datos [22, 7, 14], además tiene un costo computacional bajo y una semántica clara.

La medida de rendimiento utilizada para evaluar los algoritmos fue la precisión de la clasificación (accuracy). Esta medida se calculó en línea, con el objetivo de medir cómo evoluciona el proceso de aprendizaje en el tiempo. En este artículo se utilizó para evaluar los algoritmos el enfoque *test-then-train*, también conocido como *prequential* (*predictive sequential*) que se deriva del error predictivo secuencial [8]. Este enfoque consiste básicamente en calcular las medidas de interés (usualmente la precisión) para cada instancia nueva (etapa de prueba) y después utilizar el ejemplo para seguir con el entrenamiento del algoritmo (etapa de entrenamiento) [10]. Por lo tanto, en cada ejemplo nuevo, el clasificador primero se probó y luego se entrenó. Durante el proceso de aprendizaje, la precisión se calculó con respecto a una ventana deslizante de tamaño 100 [2].

5.2. Selección de la mejor variante del algoritmo propuesto

El algoritmo propuesto ADCE utiliza la versión en línea del algoritmo bagging para entrenar los clasificadores base. Primeramente se evaluó el algoritmo propuesto en combinación con diferentes métodos para combinar clasificadores: EWMA, NB, EW, DS.

Las combinaciones de ADCE con los diferentes métodos de combinación de clasificadores se evaluaron sobre todos los conjuntos de datos descritos en la sección anterior. Los resultados obtenidos se analizaron utilizando el test de Friedman y el procedimiento de Holm para el análisis post hoc, con un valor de significación de $0,05$. La Figura 1 muestra los rankings de los algoritmos con respecto a la precisión de la clasificación, en la cual las combinaciones entre las que no existen diferencias significativas están conectadas por una línea horizontal. En la misma se muestra que el algoritmo propuesto en combinación con NB obtiene los mejores resultados aunque sin superar significativamente a la combinación con EWMA.

5.3. Resultados frente a cambios abruptos y graduales

En este experimento los algoritmos se ejecutaron frente a cambios abruptos y graduales utilizando los conjuntos de datos artificiales descritos en la Tabla 1. Para simular cambios graduales utilizamos la función sigmoide que está implementada en MOA [2], aumentando la probabilidad de que las nuevas instancias pertenecen al nuevo concepto. Se generaron 10 cambios cada 25,000. Para evaluar el algoritmo frente a cambios graduales el período de transición entre conceptos consecutivos fue de 100, 500, 1000. Las Tabla 2 resume el rendimiento de los algoritmos sobre cambios abruptos en términos de la media y la desviación estándar para la precisión de la clasificación. Los niveles más altos de precisión están resaltados en negrita. Como se puede observar en la Tabla 2 el algoritmo propuesto obtiene los mejores resultados en cuanto a precisión.

En las Tablas 3, 4 y 5 se muestran los resultados de los algoritmos frente a cambios de conceptos graduales. Las Tablas 3, 4 y 5 muestran que el algoritmo también obtiene buenos resultados frente a cambios graduales. Adicionalmente la Figura 2 muestra la precisión de los algoritmos sobre los conjuntos de datos utilizados. Podemos ver que la precisión del algoritmo propuesto no cae drásticamente cuando ocurre el cambio de concepto. Esto se debe principalmente a la eficacia del método utilizado para adaptarse a los cambios de concepto en combinación con NB para combinar la salida de los clasificadores base.

Algoritmo	ADCE+NB	LeveragingBag	OzaBagAdwin	OzaBag
AGR	**83,99 ±12,49**	82,98 ±13,29	80,57 ±17,28	63,38 ±15,64
HYP	**92,98 ±02,79**	85,45 ±13,87	85,41 ±15,59	65,77 ±23,49
LED	**73,53 ±02,90**	70,16 ±08,86	71,34 ±06,46	46,87 ±20,45
RBF	71,96 ±01,54	**72,00 ±01,48**	71,98 ±01,50	71,95 ±01,51
SEA	**87,76 ±01,63**	87,69 ±01,56	87,04 ±02,03	84,42 ±03,65
STA	**99,95 ±00,28**	87,10 ±19,18	87,28 ±18,98	65,78 ±20,05
WAV	**80,52 ±01,28**	80,49 ±01,28	80,48 ±01,30	80,48 ±01,30

Tabla 2: Resultados los algoritmos frente a cambios abruptos.

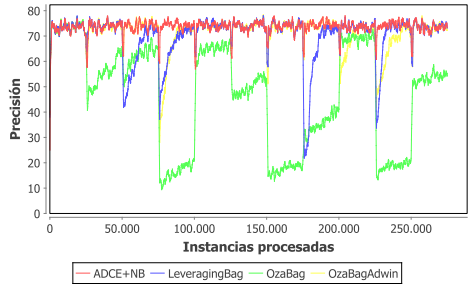
Algoritmo	ADCE+NB	LeveragingBag	OzaBagAdwin	OzaBag
AGR	**83,96 ±12,38**	81,89 ±15,24	82,20 ±14,94	63,38 ±15,63
HYP	**91,92 ±03,06**	87,12 ±14,18	90,40 ±07,44	62,12 ±20,64
LED	**73,47 ±03,06**	73,24 ±03,61	73,19 ±03,72	47,95 ±19,20
RBF	71,96 ±01,54	72,00 ±01,48	**72,02 ±01,49**	72,00 ±01,50
SEA	**87,76 ±01,64**	87,47 ±01,86	87,28 ±01,94	84,43 ±03,65
STA	**99,87 ±00,66**	91,79 ±12,37	91,48 ±12,52	65,75 ±19,99
WAV	**80,53 ±01,29**	80,50 ±01,28	80,50 ±01,30	80,50 ±01,30

Tabla 3: Resultados de los algoritmos frente a cambios graduales. El número de instancias en el período de transición fue 100.

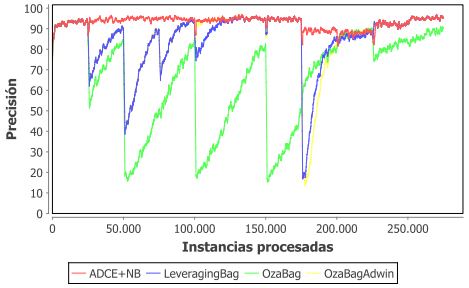
5.4. Resultados con conjuntos de datos reales

En muchos escenarios reales se ha detectado la presencia de cambios de concepto. En estos escenarios no se sabe que tipos de cambios están presentes por lo que es importante realizar experimentos con datos de distintos escenarios. Los conjuntos de datos reales seleccionados (ver Tabla 1) se han usado en diferentes estudios sobre aprendizaje a partir de flujos de datos no estacionarios [13, 27].

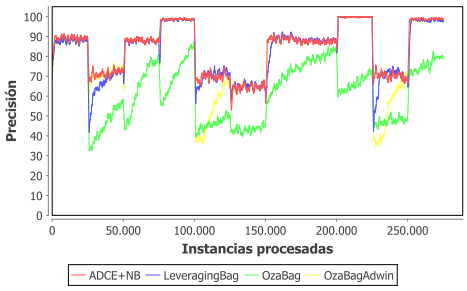
La Tabla 6 muestra el rendimiento de los algoritmos en estos conjuntos de datos reales. Los niveles más altos de precisión están resaltados en negrita. Como se puede ver en la Tabla 6 el algoritmo propuesto obtiene los mejores resultados. Por lo que reemplazar el clasificador en el que se detectó el cambio por uno nuevo es más eficiente que eliminar siempre el peor clasificador. Al eliminar el peor clasificador puede que no se elimine el clasificador donde se detectó el cambio, por lo que el ensamble se adapta más lento



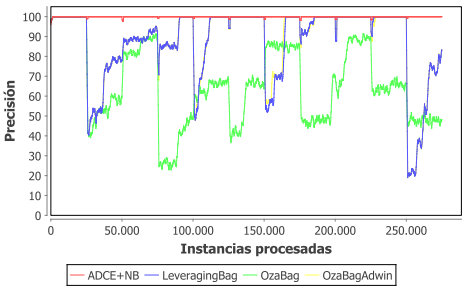
(a) HYP



(b) HYP

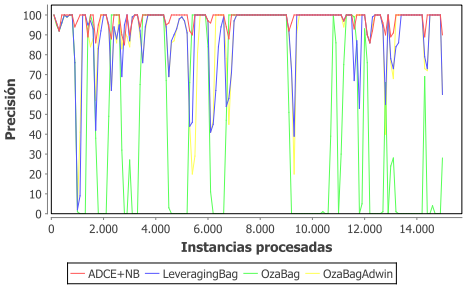


(c) SEA

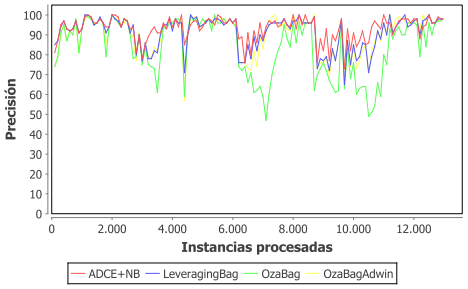


(d) STA

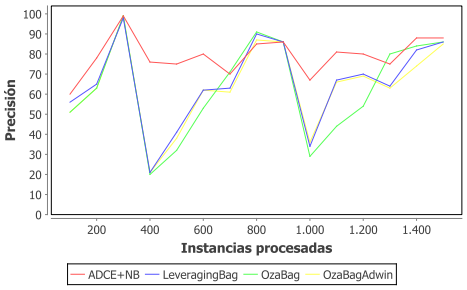
Figura 2: Precisión de los algoritmos frente a cambios abruptos. Se generaron 10 cambios cada 25,000 instancias.



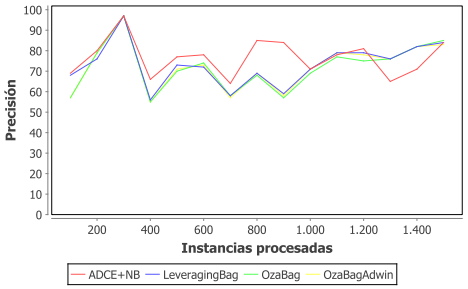
(a) EYE



(b) NUR



(c) USE1



(d) USE2

Figura 3: Precisión de los algoritmos en conjuntos de datos reales.

Algoritmo	ADCE+NB	LeveragingBag	OzaBagAdwin	OzaBag
AGR	**83,76 ±12,42**	83,72 ±12,35	83,72 ±12,30	63,38 ±15,57
HYP	**92,42 ±2,57**	91,72 ±3,94	91,71 ±4,02	63,21 ±18,53
LED	73,32 ±3,46	**73,50 ±2,87**	73,44 ±3,14	46,06 ±19,36
RBF	71,91 ±1,53	**71,97 ±1,48**	71,92 ±1,50	71,91 ±1,50
SEA	**87,73 ±1,64**	87,63 ±1,61	87,01 ±1,91	84,44 ±3,62
STA	**99,56 ±2,08**	99,32 ±3,19	99,11 ±4,17	65,82 ±19,95
WAV	**80,54 ±1,29**	80,50 ±1,28	80,50 ±1,30	80,50 ±1,30

Tabla 4: Resultados de los algoritmos frente a cambios graduales. El número de instancias en el período de transición fue 500.

Algoritmo	ADCE+NB	LeveragingBag	OzaBagAdwin	OzaBag
AGR	83,46 ±12,48	83,47 ±12,38	**83,49 ±12,35**	63,36 ±15,47
HYP	**92,75 ±03,14**	92,52 ±03,48	92,35 ±03,87	63,84 ±34,24
LED	73,20 ±03,90	73,30 ±03,60	**73,22 ±03,77**	44,66 ±19,23
RBF	**72,01 ±01,55**	71,99 ±01,47	72,00 ±01,49	71,96 ±01,51
SEA	**87,67 ±01,72**	87,41 ±01,68	87,00 ±01,80	84,46 ±03,54
STA	**99,08 ±03,70**	99,01 ±04,30	98,86 ±04,89	65,86 ±19,88
WAV	**80,54 ±01,30**	80,51 ±01,29	80,51 ±01,31	80,51 ±01,31

Tabla 5: Resultados de los algoritmos frente a cambios graduales. El número de instancias en el período de transición fue 1000.

a los cambios. La Figura 3 muestra la precisión de los algoritmos frente a los conjuntos de datos EYE, NUR, USE1 y USE2. En la Figura 3 se puede ver que la precisión del algoritmo propuesta no disminuye significativamente cuando ocurre un posible cambio de concepto.

Para comprobar diferencias significativas entre los algoritmos se utilizó el test de Friedman y el procedimiento de Holm para el análisis post hoc, con un valor de significación de $0,05$. El ranking se calculó con respecto a todos los conjuntos de datos (artificiales y reales). La Figura 4 muestra que, en los conjuntos de datos considerados, el algoritmo propuesto fue significativamente más preciso que el resto de los algoritmos competidores.

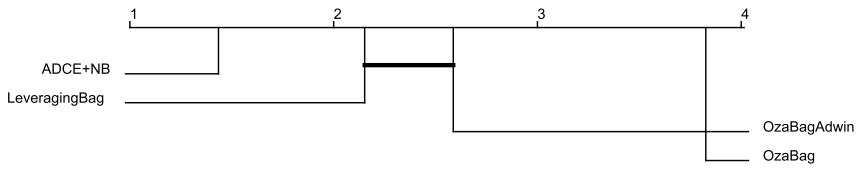


Figura 4: Ranking de los algoritmos. Los algoritmos que no son significativamente diferente están conectados. El ranking se calculó con respecto a las Tablas 2, 3, 4, 5 y 6.

Algoritmo	ADCE+NB	LeveragingBag	OzaBagAdwin	OzaBag
ADU	**83,49 ±03,80**	**83,49 ±03,84**	83,29 ±03,89	83,29 ±03,89
BAN	88,74 ±11,74	**89,94 ±11,17**	89,79 ±11,39	89,15 ±12,58
CAR	**85,56 ±10,11**	82,56 ±10,42	80,67 ±10,78	80,56 ±11,71
CON	74,83 ±12,77	**75,07 ±13,72**	74,87 ±13,99	69,17 ±17,33
COV	**87,36 ±07,70**	83,20 ±11,92	83,07 ±11,96	60,55 ±21,76
ELE	**84,75 ±06,54**	78,84 ±11,97	78,91 ±12,13	74,26 ±14,63
EYE	**98,17 ±03,71**	90,61 ±17,88	90,91 ±18,26	47,31 ±46,41
KDD	**99,80 ±01,29**	99,62 ±03,14	99,66 ±02,88	97,88 ±11,01
MUS	98,82 ±01,75	**99,28 ±01,45**	98,65 ±02,17	98,41 ±02,29
NUR	**93,34 ±05,75**	91,19 ±08,13	90,30 ±09,27	84,11 ±14,18
OUT	**63,20 ±06,69**	60,83 ±10,53	58,33 ±11,80	55,95 ±15,50
POK	**75,69 ±09,71**	73,08 ±12,58	73,48 ±12,35	59,55 ±21,95
SEG	**78,00 ±07,21**	77,83 ±06,88	77,58 ±07,19	77,58 ±07,19
SPA1	**99,38 ±04,04**	97,64 ±05,75	98,00 ±05,19	83,55 ±14,46
SPA2	**93,05 ±05,95**	91,67 ±10,08	90,53 ±10,84	90,44 ±10,97
USE1	**79,20 ±09,24**	65,67 ±20,78	64,07 ±20,50	62,87 ±23,92
USE2	**76,67 ±08,78**	73,27 ±10,40	72,33 ±11,38	71,93 ±11,43
WEA	**72,90 ±09,81**	71,20 ±11,73	72,26 ±11,06	69,95 ±11,90

Tabla 6: Resultados de los algoritmos frente a conjuntos de datos reales.

6. Conclusiones

En este artículo se presentó un nuevo algoritmo capaz de aprender de flujos de datos no estacionarios. El nuevo algoritmo combina la simplicidad de bagging para entrenar clasificadores base con los métodos de combinación de clasificadores: EWMA, NB, DS y EW. Los resultados experimentales muestran los mejores resultados utilizando NB como método de combinación. El nuevo algoritmo ADCE procesa los datos de entrada con complejidad temporal y espacial constante, y solo procesa cada ejemplo de entrenamiento una vez. Para manipular cambios de concepto, ADCE utiliza un detector de cambios en cada clasificador base para estimar su tasa de error. Cuando se estima un cambio en un clasificador base, este es remplazado por uno nuevo.

ADCE fue comparado empíricamente con varios algoritmos de ensamble de la familia bagging, utilizando Naive Bayes como clasificador base. Los experimentos incluyeron conjuntos de datos artificiales y reales. Todos los algoritmos fueron probados frente a los tipos de cambios comunes (abruptos y graduales). Los experimentos mostraron que el nuevo algoritmo es una buena opción para el aprendizaje a partir de flujos de datos con cambios de concepto.

Agradecimientos

Los autores queremos agradecer al editor y a los revisores anónimos por sus comentarios y sugerencias útiles.

Referencias

[1] Albert Bifet and Ricard Gavalda. Learning from time-changing data with adaptive windowing. In *In SIAM International Conference on Data Mining*, 2007. 2.2

[2] Albert Bifet, Geoff Holmes, Richard Kirkby, and Bernhard Pfahringer. Moa: Massive online analysis. *The Journal of Machine Learning Research*, 11:1601–1604, 2010. 5.1, 5.3

[3] Albert Bifet, Geoff Holmes, and Bernhard Pfahringer. Leveraging bagging for evolving data streams. In *Machine learning and knowledge discovery in databases*, pages 135–150. Springer, 2010. (document), 5.1

[4] Albert Bifet, Geoff Holmes, Bernhard Pfahringer, Richard Kirkby, and Ricard Gavalda. New ensemble methods for evolving data streams. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 139–148. ACM, 2009. (document), 2.2, 5.1

[5] Isvani Inocencio Frias Blanco, Agustin Alejandro Ortiz Diaz, Gonzalo Ramos Jimenez, Rafael Morales Bueno, and Yaile Caballero Mota. Clasificadores y multiclasificadores con cambio de concepto basados en arboles de decision. *Inteligencia Artificial, Revista Iberoamericana de Inteligencia Artificial*, 14(45):32–43, 2010. (document)

[6] Leo Breiman. Bagging predictors. *Machine learning*, 24(2):123–140, 1996. 2.2

[7] Bojan Cestnik and others. Estimating probabilities: a crucial task in machine learning. In *ECAI*, volume 90, pages 147–149, 1990. 5.1

[8] A. P. Dawid. Present Position and Potential Developments: Some Personal Views: Statistical Theory: The Prequential Approach. *Journal of the Royal Statistical Society. Series A (General)*, 147(2):278–292, 1984. 5.1

[9] Magdalena Deckert. Batch weighted ensemble for mining data streams with concept drift. In *Foundations of Intelligent Systems*, pages 290–299. Springer, 2011. 2.1

[10] Pedro Domingos and Geoff Hulten. Mining high-speed data streams. In *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 71–80. ACM, 2000. 5.1

[11] Yoav Freund and Robert E. Schapire. A Short Introduction to Boosting. In *In Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence*, pages 1401–1406. Morgan Kaufmann, 1999. 2.2

[12] Isvani Frias-Blanco, Jose del Campo-Avila, Gonzalo Ramos-Jimenez, Rafael Morales-Bueno, Agustin Ortiz-Diaz, and Yaile Caballero-Mota. Online and Non-Parametric Drift Detection Methods Based on Hoeffding Bounds. *IEEE Transactions on Knowledge and Data Engineering*, 27(3):810–823, March 2015. (document), 4

[13] Isvani Frías-Blanco, José del Campo-Ávila, Gonzalo Ramos-Jiménez, Andre CPLF Carvalho, Agustín Ortiz-Díaz, and Rafael Morales-Bueno. Online adaptive decision trees based on concentration inequalities. *Knowledge-Based Systems*, 104:179–194, 2016. 5.4

[14] Isvani Frías-Blanco, Alberto Verdecia-Cabrera, Agustín Ortiz-Díaz, and Andre Carvalho. Fast adaptive stacking of ensembles. In *Proceedings of the 31st Annual ACM Symposium on Applied Computing*, pages 929–934. ACM, 2016. 5.1

[15] João Gama, Pedro Medas, Gladys Castillo, and Pedro Pereira Rodrigues. Learning with Drift Detection. In *Advances in Artificial Intelligence - SBIA 2004, 17th Brazilian Symposium on Artificial Intelligence, São Luis, Maranhão, Brazil, September 29 - October 1, 2004, Proceedings*, pages 286–295, 2004. 1

[16] Joao Gama, Indre Zliobaite, Albert Bifet, Mykola Pechenizkiy, and Abdelhamid Bouchachia. A Survey on Concept Drift Adaptation. *ACM Comput. Surv.*, 46(4):44:1–44:37, March 2014. (document), 1

[17] Haibo He, Sheng Chen, Kang Li, and Xin Xu. Incremental Learning From Stream Data. *IEEE Transactions on Neural Networks*, 22(12):1901–1914, December 2011. (document)

[18] David P Helmbold and Philip M Long. Tracking drifting concepts using random examples. In *Proceedings of the fourth annual workshop on Computational learning theory*, pages 13–23. Morgan Kaufmann Publishers Inc., 1991. 1

[19] David P. Helmbold and Philip M. Long. Tracking Drifting Concepts By Minimizing Disagreements. *Machine Learning*, 14(1):27–45, 1994. 1

[20] Elena Ikonomovska and João Gama. Learning Model Trees from Data Streams. In *Discovery Science, 11th International Conference, DS 2008, Budapest, Hungary, October 13-16, 2008. Proceedings*, pages 52–63, 2008. 1

[21] Anthony Kuh, Thomas Petsche, and Ronald L. Rivest. Learning Time-varying Concepts. In *Proceedings of the 1990 Conference on Advances in Neural Information Processing Systems 3*, NIPS-3, pages 183–189, San Francisco, CA, USA, 1990. Morgan Kaufmann Publishers Inc. 1

[22] Pat Langley, Wayne Iba, and Kevin Thompson. An analysis of Bayesian classifiers. In *Aaai*, volume 90, pages 223–228, 1992. 5.1

[23] Leandro L Minku, Allan P White, and Xin Yao. The impact of diversity on online ensemble learning in the presence of concept drift. *Knowledge and Data Engineering, IEEE Transactions on*, 22(5):730–742, 2010. 1

[24] Leandro L. Minku and Xin Yao. DDD: A new ensemble approach for dealing with concept drift. *Knowledge and Data Engineering, IEEE Transactions on*, 24(4):619–633, 2012. 2.2

[25] Douglas C Montgomery. *Introduction to statistical quality control*. John Wiley & Sons, 2007. 4

[26] Kyosuke Nishida. *Learning and detecting concept drift*. PhD thesis, School of Information Science and Technology, Hokkaido University, 2008. (document)

[27] Agustín Ortíz Díaz, José del Campo-Ávila, Gonzalo Ramos-Jiménez, Isvani Frías Blanco, Yailé Caballero Mota, Antonio Mustelier Hechavarría, and Rafael Morales-Bueno. Fast Adapting Ensemble: A New Algorithm for Mining Data Streams with Concept Drift. *The Scientific World Journal*, 2014. 1, 5.4

[28] Nikunj C. Oza and Stuart Russell. Online Bagging and Boosting. In Tommi Jaakkola and Thomas Richardson, editors, *Eighth International Workshop on Artificial Intelligence and Statistics*, pages 105–112, Key West, Florida. USA, January 2001. Morgan Kaufmann. (document), 2.2, 5.1

[29] Lior Rokach. Ensemble Methods for Classifiers. In *Data Mining and Knowledge Discovery Handbook*. 2005. DOI: 10.1007/0-387-25465-X_45. 3, 3

[30] S. Shilen. Multiple binary tree classifiers. *Pattern Recognition*, 23(7):757–763, 1990. 3

[31] W. Nick Street and YongSeog Kim. A Streaming Ensemble Algorithm (SEA) for Large-scale Classification. In *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '01, pages 377–382, New York, NY, USA, 2001. ACM. 2.1

[32] Haixun Wang, Wei Fan, Philip S. Yu, and Jiawei Han. Mining concept-drifting data streams using ensemble classifiers. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 226–235. ACM, 2003. 2.1

[33] Gerhard Widmer and Miroslav Kubat. Learning in the presence of concept drift and hidden contexts. *Machine learning*, 23(1):69–101, 1996. 1