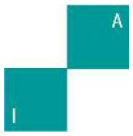
Inteligencia Artificial 24(67), 51-70
doi: 10.4114/intartif.vol24iss67pp51-70



INTELIGENCIA ARTIFICIAL

http://journal.iberamia.org/

Otimização por Enxame de Partículas híbrido de duas fases Aplicado o Problema de Layout em Linha Dupla

Two-phase hybrid Particle Swarm Optimization Applied to the Double Row Layout Problem

Gildásio Lecchi Cravo[1,A], Dayan de Castro Bissoli[2,B], André Renato Sales Amaral[1,C]

[1] Programa de Pós-Graduação em Informática (PPGI), Universidade Federal do Espírito Santo (UFES), Vitória, ES, Brasil
[2] Departamento de Computação (DCOMP), Universidade Federal do Espírito Santo (UFES), Alegre, ES, Brasil
lecchi2003@gmail.com[A], dayan.bissoli@ufers.br[B], amaral@inf.ufes.br[C]

Abstract The double-row layout problem (DRLP) consists in determining the location of facilities along both sides of a central corridor, with the objective of minimizing the weighted sum of the distances between all pairs of facilities. Facilities can be machines, work centers, manufacturing cells, building departments and robots in manufacturing systems. This paper proposes a purely heuristic approach, based on Particle Swarm Optimization (PSO) metaheuristic. To validate the proposed algorithm, it was submitted to computational tests with fifty-one instances, including instances considered large and the results found show the PSO as an excellent approach for DRLP having improved the known values for several instances available in the literature.

Resumo O problema de layout em linha dupla (DRLP) consiste em determinar a localização de facilidades ao longo de ambos os lados de um corredor central, tendo como objetivo a minimização da soma ponderada das distâncias entre todos os pares de facilidades. As facilidades podem ser máquinas, centros de trabalho, células de manufatura, departamentos de um edifício e robôs em sistemas de manufatura. Esse trabalho propõe uma abordagem puramente heurística, baseada na meta-heurística Particle Swarm Optimization (PSO). Para validar o algoritmo proposto, o mesmo foi submetido a testes computacionais com cinquenta e uma instâncias, incluindo instâncias consideradas de grande porte e os resultados encontrados mostram o PSO proposto como uma excelente abordagem para o DRLP, tendo melhorado os valores conhecidos para diversas instâncias disponíveis na literatura.

Keywords: Facilities layout, Computational Intelligence, Metaheuristics, Particle Swarm Optimization.
Palavras-chave: Layout de facilidades, Inteligência Computacional, Meta-heurísticas, Particle Swarm Optimization.

1 Introdução

Uma característica importante nos sistemas de manufatura modernos é a flexibilidade do layout. A flexibilidade é necessária para garantir uma alta taxa de utilização devido ao grande valor de investimento nessas instalações. A falta de flexibilidade torna as instalações ociosas quando ocorrem necessidades de reorganizar o layout. Além disso, um layout flexível proporciona uma resposta rápida às mudanças na demanda dos clientes [24].

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

Um sistema de manufatura consiste em um conjunto de máquinas e um dispositivo de manuseio de materiais que realiza o transporte dos materiais ao longo das máquinas. Dispositivos de manipulação automatizados, como veículos guiados automaticamente (VGA) são utilizados para aumentar a eficiência do sistema de manufatura. O VGA é superior ao transportador convencional no que diz respeito à utilização, custo e flexibilidade. Assim, estes são amplamente empregados em sistemas flexíveis de manufatura (FMS) [7]. Como os VGAs têm um melhor desempenho quando se movem em linhas retas, é interessante a organização das máquinas em linha reta em um FMS [43]. Desse contexto, surge o Problema de Layout em Linha (RLP, do inglês, *Row Layout Problem*), o qual objetiva determinar a ordenação de facilidades em arranjos em linhas retas.

Em reais contextos operacionais de indústrias, frequentemente as máquinas são organizadas em um layout em que as máquinas são dispostas em ambos os lados de um corredor em linha reta, com seus comprimentos paralelos ao corredor. Assim, surge o Problema de Layout de Linha Dupla (DRLP, do inglês, *Double-row layout problem*), que consiste em atribuir a localização de um conjunto de máquinas em ambos os lados do corredor, para que o custo total de transporte de materiais entre as máquinas seja minimizado [7]. A Figura 1 apresenta um exemplo de arranjo do DRLP, onde um VGA se move ao longo do corredor. Os pontos centrais de cada máquina é o local de carga e descarga de material pelo VGA.

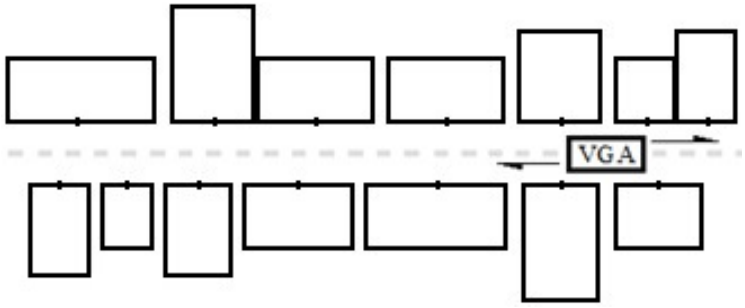


Figura 1. Exemplo de arranjo de máquinas em linha dupla.

O DRLP aparece em aplicações reais como na linha de produção de display de cristal líquido [17], na fabricação de semicondutores [34], entre outras. Possui uma participação significativa no processo produtivo na indústria, em termos de custo e tempo [33]. O problema vem sendo objeto de estudo por diversos autores [6, 7, 8, 17, 23, 26, 34].

Matematicamente o DRLP pode ser formulado como um conjunto de máquinas $N = \{1, 2, ..., n\}$, com seus respectivos comprimentos $\{l_1, ..., l_n\}$ e um fluxo não negativo c_{ij}. Em um layout viável para o DRLP, d_{ij} é a distância entre os centros dos pares (i, j) de máquinas. Dessa forma, o objetivo do DRLP é encontrar um layout viável que minimiza a Equação (1) [7].

$$min \sum_{1 \le i < j \le n} c_{ij} d_{ij} \tag{1}$$

Na Figura 2 são mostradas as variáveis envolvidas na representação do DRLP. A variável a_{ij} representa o espaçamento mínimo entre máquinas adjacentes, denominada folgas requeridas (*clearance*). As folgas são utilizadas nas situações onde as máquinas precisam de um espaço para manutenção ou operação. Heragu e Kusiak [25] supuseram que essa folga entre cada par de máquinas fosse dada. Entretanto, os valores das folgas podem ser considerados de diversas formas. Se a folga é uma característica da máquina para uso em manutenção, ela pode ser incluída no comprimento da máquina [7]. Mais especificamente, quando os valores das folgas são iguais a um dado valor a, o comprimento de cada máquina pode ser aumentado pelo valor a e assim, o problema pode ser tratado como um problema sem folgas, ou seja, a_{ij}=0 para todo par (i, j) de máquinas [25, 10, 14]. Se as folgas são incluídas em uma instância, é dito que o DRLP tem folgas explícitas. No entanto, quando os valores não aparecem nos dados da instância, porque eles já foram incluídos nos comprimentos das máquinas, é dito que a instância tem folgas implícitas. Se as folgas entre as máquinas são dependentes da ordenação, não sendo todas iguais, as folgas implícitas não poderão ser utilizadas [7].

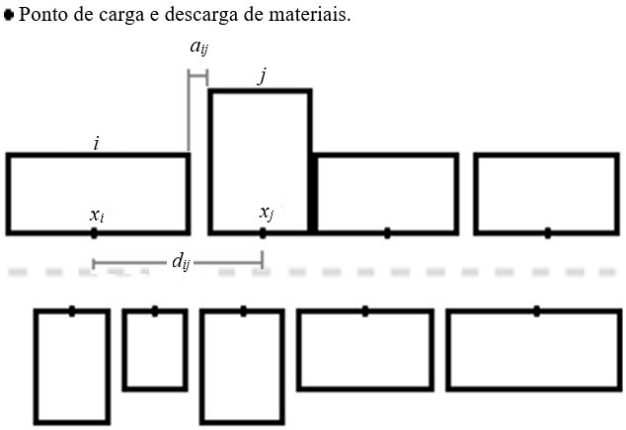


Figura 2. Representação do DRLP.

O DRLP apresenta aspectos combinatórios e contínuos, o que aumenta a complexidade para ser resolvido [23]. Sendo um problema de otimização combinatória NP-difícil [6], o uso de métodos baseados em heurísticas e/ou meta-heurísticas são indicados para que seja possível a resolução em tempos computacionais aceitáveis, possibilitando a resolução de instâncias de maior porte. Entretanto, não há muita pesquisa com abordagens puramente heurísticas para o DRLP [30].

Pelo exposto acima, para resolver o DRLP, esse trabalho implementa uma solução baseada na meta-heurística *Particle Swarm Optimization* (PSO). O PSO é uma meta-heurística evolutiva na qual um exame de partículas que interagem entre si trocando informações sobre o espaço de soluções de forma a convergir para uma solução localmente ótima. Essa meta-heurística é amplamente usada na resolução de problemas contínuos [13].

Para validar a qualidade do método proposto, realizou-se experimentos computacionais com instâncias com até n = 80 máquinas, que são considerados problemas de grande porte para o DRLP. Os resultados obtidos são comparados com os melhores resultados disponíveis na literatura.

O restante do trabalho apresenta a seguinte estrutura: a Seção 2 apresenta literatura relacionada sobre o DRLP; na Seção 3 é apresentada a meta-heurística proposta para solucionar o DRLP; na Seção 4 são mostrados os experimentos e os resultados computacionais; e na Seção 5, as conclusões, seguido do apêndice A e das referências bibliográficas.

2 Literatura Relacionada

Os problemas de layout de linha (RLP – *Row Layout Problem*) são muito interessantes do ponto de vista prático e têm recebido bastante atenção nos últimos anos, tendo diversas formulações e abordagens para sua solução [11, 19, 28]. Os problemas em linha incluem os problemas de layout de facilidades em linha única (SRFLP, do inglês, *Single-row facility layout problem*) [40], o problema de layout em múltiplas linhas (MRLP, do inglês, *Multi-row layout problem*) [22], o problema de layout em linha dupla (DRLP) [17], o problema de alocação em corredor (CAP, do inglês, *Corridor allocation problem*) [4] e o problema de ordenação em linhas paralelas (PROP, do inglês, *Parallel-row ordering problem*) [5]. Tais problemas são classificados como NP-difícil, logo, sendo caracterizados por agregar grande dificuldade em resolvê-los [44].

O DRLP é uma extensão do SRFLP sendo que suas formulações derivam dos modelos propostos para o SRFLP. Chung e Tanchoco [17] apresentam um modelo de programação inteira mista (MIP, do inglês *Mixed-Integer Programming*) para o DRLP derivado do modelo para o SRFLP de Heragu e Kusiak [26]. No modelo, é definido um parâmetro denominado a_{ij}. Se o valor de a_{ij}= 0, o modelo trata de problemas com folgas implícitas. Os autores também apresentam resultados de cinco heurísticas desenvolvidas baseadas no modelo proposto. Zhang e Murray [46] mostraram que o modelo MIP não estava correto quando os valores de a_{ij} não são todos zeros. Entretanto, o modelo para todo a_{ij}=0 é um modelo válido para o DRLP [7].

Amaral [6] propôs um modelo MIP para o DRLP onde são definidas desigualdades válidas, que são inseridas no modelo. O modelo apresenta um número menor de variáveis (tanto contínuas, quanto binárias). Com a adição das restrições definidas nas desigualdades, houve um ganho significativo em termo computacional em comparação com o modelo proposto por Chung e Tanchoco [17].

Posteriormente, Amaral [7] apresenta uma nova formulação via MIP para o DRLP, o qual compara com os modelos de Chung e Tanchoco [17] e de Amaral [6] para instâncias com até n=13 máquinas. Os resultados

computacionais mostram que o modelo proposto tem um desempenho superior ao de [17], sendo que, em relação ao modelo de [6], não apresentou, estatisticamente, diferença significativa. Outro modelo MIP para o DRLP, também baseado no modelo de [6], é apresentado em [37], que proporcionou melhora no desempenho, de modo que foi possível a resolução do problema para instâncias de até n=15 máquinas.

Na formulação MIP para o DRLP mais recente, proposta por Chae e Regan [16], os autores modificam o modelo de [37], introduzindo novas restrições. O novo modelo melhora o desempenho, em termos de tempo de execução, para resolver problemas com até n=16 máquinas. Os modelos propostos por [6], [7], [37] e [16] não consideram as folgas explícitas, ou seja, são válidos para o DRLP apenas quando a_{ij}=0.

Como dito anteriormente, o DRLP é um problema NP-Difícil, dessa forma, somente com o uso de heurísticas e/ou meta-heurísticas foi possível solucionar instâncias de grande porte desse problema. Entretanto, não há na literatura muitas pesquisas com abordagens puramente heurísticas para o DRLP [30]. As abordagens heurísticas normalmente utilizam-se de modelagens matemáticas para determinar das variáveis contínuas do problema. A Tabela 1 apresentas a soluções mais recentes propostas para o DRLP que tratam de problemas com número de máquinas $n > 15$, tendo na primeira coluna os nomes dos autores seguida da coluna com a abordagem utilizada.

Tabela 1: Métodos heurísticos da literatura para o DRLP

Autores	Soluções propostas
Chung e Tanchoco [17]	Heurísticas construtivas com Programação Linear
Murray, Smith e Zhang [34]	Heurísticas construtivas com MIP
Hungerländer e Anjos [30]	Relaxação de programação semi-definida
Permanhane [36]	*Iterated Local Search* (ILS) com MIP
Cellin [15]	*Variable Neabourhood Search (VNS*) com MIP, VNS/ILS com MIP
Guan et al. [23]	*Decomposition-based Algorithm (DBA)*
Amaral [8]	Heurísticas de melhoria com Programação Linear

Na Tabela 1 apresenta-se os estudos que utilizam-se de métodos exatos para a resolução das variáveis contínuas definidas no DRLP, com exceção do algoritmo DBA [23]. Chung e Tanchoco [17] propõem heurísticas construtivas para o DRLP, onde processo iterativo é guiado por uma das cinco regras denominadas MinLCF (*minimum location cost first*), MinFF (*minimum flow first*), MaxFF (*maximum flow first*), MinWF (*minimum width first*), e MaxWF (*maximum width first*). Após a determinação da ordem das máquinas no layout pela heurística construtiva, as posições em cada linha são determinadas por um modelo de programação linear. Hungerländer e Anjos [30] resolvem o DRLP via relaxação do modelo de programação semi-definida e obtém a viabilidade utilizando uma heurística proposta por [29], não considerando espaços entre os departamentos. Após uma solução viável ser obtida, para a determinação das posições das máquinas, espaços de comprimento 0,5 são adicionados a solução (departamentos "artificiais" de comprimento 0,5 e fluxo de custo zero). Murray, Smith e Zhang [34] fazem uma combinação de heurísticas construtivas e de busca local com o uso de soluções obtidas por um modelo MIP para máquinas na mesma linha. Os autores consideram valores explícitos das folgas ($a_{ij} \neq 0$) e matriz de fluxos assimétrica. Permanhane [36] resolve o DRLP utilizando meta-heurística ILS para fazer o posicionamento relativo nas linhas, fazendo uso de um modelo de MIP para fazer o posicionamento das facilidades no eixo-x. Cellin [15] apresenta uma heurística VNS (*Variable Neabourhood Search*) e uma hibridização do VNS com a meta-heurística ILS, em ambas estratégias, as posições das máquinas, ao longo do corredor, são determinadas pela resolução do modelo MIP. Tais estudos tratam o DRLP com instâncias de até 50 máquinas.

Guan et al. [23] propuseram um algoritmo denominado DBA (*Decomposition-based Algorithm*) que resolve o DRLP considerando a decomposição do mesmo em dois subproblemas. O primeiro problema considera a parte combinatória do DRLP, que consiste na ordenação das máquinas, desconsiderando as folgas entre as mesmas. O segundo subproblema consiste em determinar as folgas para a solução encontrada no primeiro subproblema, ou seja, resolve-se as variáveis contínuas do DRLP, determinando os valores das abscissas dos centros das máquinas ao longo do eixo-x. No DBA, inicialmente é gerado aleatoriamente um arranjo de máquinas considerando uma única linha. Em seguida, encontra-se um ponto de quebra para definir os arranjos em ambos os lados do corredor do DRLP e assim, calcula-se os valores iniciais das abscissas em ambas as linhas. Com a solução corrente, o algoritmo realiza três processos: aplicação de uma busca local com movimentos de troca 2-opt; determinação das folgas entre as máquinas, utilizando uma meta-heurística PSO, assim encontrando a solução completa para o DRLP; e por fim, a solução corrente discreta, desconsiderando as folgas, é perturbada para a execução da próxima

iteração do algoritmo. Nos testes computacionais, os autores consideram problemas com matriz de fluxos assimétricas e valores explícitos das folgas para algumas instâncias dos testes realizados, tal como [34].

Recentemente, Amaral [8] apresenta quatro algoritmos para resolver o DRLP. O autor propõe uma abordagem em duas fases nos quatro algoritmos, sendo que, na primeira fase é resolvido o problema de layout desconsiderando os espaços entre as máquinas no arranjo, definindo assim a ordem das máquinas no layout. No segundo momento, as soluções encontradas pelas heurísticas são melhoradas utilizando o modelo MIP proposto por [7], onde são definidos os espaços entre as máquinas no layout. O autor trata problemas com tamanho $n = 50$ máquinas, considerando valores implícitos e explícitos para as folgas requeridas e matriz de fluxos simétrica e assimétrica.

Considerando que na literatura somente Guan et al. [23] resolveram o DRLP com abordagens puramente heurísticas, gerando bons resultados. No presente estudo propõe-se o desenvolvimento de um método também baseado em heurística, porém com a inovação de não utilizar técnicas de decomposição do problema, isto é, não considerar o DRLP como dois subproblemas, como as abordagens até então propostas, mas sim, considerar na resolução do DRLP pelo PSO-DRLP tanto o posicionamento das máquinas quando suas posições no mesmo processo iterativo. Dessa forma, o PSO-DRLP proposto apresenta similaridade com o DBA [23], o qual também utiliza da meta-heurística PSO para a resolução das posições das máquinas ao longo do eixo-*x*. Entretanto, existem diferenças importantes nas duas abordagens.

No DBA, o DRLP é decomposto em dois subproblemas, sendo que no processo principal, o algoritmo realiza buscas por soluções considerando o primeiro subproblema, que é a parte discreta do DRLP (ordenação das máquinas no arranjo), desconsiderando as folgas entre as máquinas. Para solucionar o segundo subproblema, gerada a solução discreta, o DBA aplica o PSO para calcular as folgas entre as máquinas no arranjo. Por outro lado, o PSO-DRLP é baseado na meta-heurística PSO, onde o enxame de partículas é organizado em subpopulações e o processo iterativo alterna entre duas fases de otimização. Na primeira fase, busca-se encontrar as melhores soluções em cada subpopulação. Enquanto que, na segunda fase, a busca considera as melhores soluções de cada subpopulação como o exame de partículas, com o objetivo de encontrar a melhor solução global do enxame. Além disso, para acelerar a convergência do PSO-DRLP, a cada atualização das partículas pelo PSO (nova solução encontrada) é aplicado um procedimento de busca local. Outra diferença está no uso do procedimento de mutação. No PSO-DRLP não existe tal procedimento. Enquanto que, no DBA, o procedimento é utilizado como perturbação da solução corrente discreta, executada a cada iteração do algoritmo.

3 Meta-Heurística PSO proposta para o DRLP (PSO-DRLP)

A Otimização por Enxame de Partículas (PSO) tem sido amplamente utilizada para o tratamento de funções mal estruturadas, contínuas, discretas, restritas e não restritas, cujo comportamento é baseado em bandos de pássaros, cardumes de peixes ou enxames de abelhas, e até mesmo comportamento social humano [38] e apresenta uma abordagem muito parecida com paradigmas populares de computação evolutiva como Algoritmos Genéticos [27] e Evolução Diferencial [41].

A versão do PSO padrão usa uma população de partículas, onde cada partícula representa uma solução candidata do problema tratado. O sistema é inicializado com uma população de soluções aleatórias e busca por soluções melhores de acordo com alguma função de aptidão (*fitness*), gerando novas soluções pela atualização das posições das partículas através do espaço de busca multidimensional. As partículas acompanham sua própria posição com melhor aptidão, bem como, a melhor posição da sua vizinhança na população [32, 38].

Formalmente, cada partícula i é representada por um vetor posição x_i, que representa as variáveis do problema a ser resolvido, e um vetor velocidade, representado por um vetor v_i em um espaço multidimensional. Todas as partículas lembram de seu melhor vetor posição $pbest_i$ e conhecem o melhor vetor posição global $gbest$. A cada iteração, a partícula i tem seu vetor velocidade v_i e o vetor posição x_i atualizados de acordo com as Equações (2) e (3) [1].

$$v_i = v_i + r_1 c_1 (pbest_i - x_i) + r_2 c_2 (gbest - x_i) \quad (2)$$

$$x_i = x_i + v_i \quad (3)$$

Onde, r_1 e r_2 são números aleatórios, que são usados para manter a diversidade da população, sendo gerados por uma distribuição uniforme no intervalo (0, 1) para cada componente da partícula i. As constantes de aceleração c_1 e c_2, são constantes positivas, chamada de coeficiente da componente de auto reconhecimento e da componente social, respectivamente. Da Equação (2), a partícula decide para onde ir, considerando sua

experiência $pbest_i$, que é a memória de sua melhor posição passada e a experiência da partícula mais bem sucedida do enxame *gbest*. Para guiar as partículas efetivamente no espaço de busca, a distância máxima de movimento durante uma iteração pode ser fixada entre a velocidade máxima [v_{min}, v_{max}] [1].

Shi e Eberhart [39] introduziram o fator de inércia que elimina a necessidade de fixação de velocidade, mas ainda assim, restringe o comportamento divergente [39, 21]. O fator de inércia ω controla a atualização da partícula, ponderando a contribuição da velocidade anterior, ou seja, basicamente controla quanto da velocidade anterior irá influenciar a nova velocidade. A equação da velocidade então, foi alterada para:

$$v_i=\omega v_i+r_1c_1(pbest_i - x_i) + r_2c_2(gbest - x_i) \quad (4)$$

Estudos empíricos têm mostrado que o valor de ω é muito importante para garantir um comportamento convergente [20, 39]. Valores de $\omega > 1$, geram um incremento da velocidade ao longo do tempo, o que leva a um comportamento divergente. Para valores $\omega < 0$, partículas desaceleram até que suas velocidades se tornarem zero (dependendo dos valores de c_1 e c_2). Embora os valores de inércia estática tenham sido usados com sucesso, os valores de inércia adaptativa também mostraram levar a um comportamento convergente [42, 45].

Nas subseções seguintes são apresentadas as etapas que compõem a propostas de implementação da meta-heurística PSO para a resolução do DRLP, denominada PSO-DRLP.

3.1 Representação da Solução e Geração do Enxame de Partículas

No algoritmo proposto, uma partícula encapsula as seguintes informações: um número inteiro *t* e um arranjo com *n* máquinas. Cada máquina contém sua identificação, sua posição *x* (a abscissa do ponto central de carga e descarga de material) e velocidade *v*. O número inteiro *t* define o ponto de quebra do arranjo para representar as duas linhas do DRLP. Assim, em uma linha tem-se *t* máquinas e na outra linha, as (*n* - *t*) máquinas restantes. A aptidão de cada partícula é calculada pela função objetivo, conforme Equação (1). A Figura 3, exemplifica a representação da solução com um arranjo de n=10 máquinas e o ponto de quebra t=4.

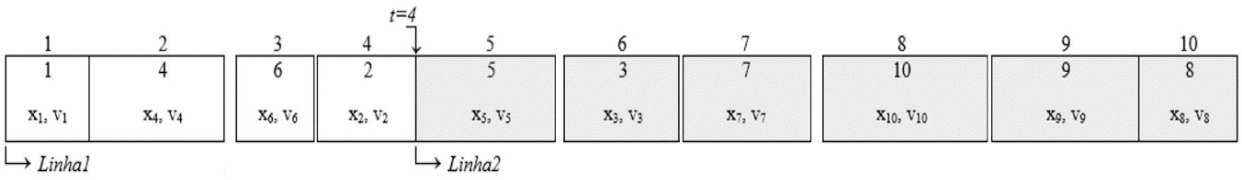


Figura 3. Representação do arranjo no PSO-DRLP

No PSO-DRLP, o exame de partícula é formado por subpopulações formadas por partículas com o mesmo valor para *t* (ponto de quebra do arranjo de máquinas). O ponto de quebra *t* são limitados pelo intervalo [t_{min}, t_{max}] e os valores de *t* não são redefinidos no processo iterativo do PSO-DRLP, ou seja, uma vez definido *t* na inicialização da partícula o valor não será mais alterado.

O processo de criação do enxame inicial de partículas é mostrado no pseudocódigo do Algoritmo 1. O algoritmo recebe como parâmetros o valor m_s, que define o total de partículas em cada subpopulação; n_s, que define a quantidade de subpopulações que irá compor o exame de partículas; *n*, número de máquinas do problema; e por fim, os valores de t_{min} e t_{max} que definem os limitantes para os valores de *t*. O procedimento retorna *P*, conjunto (enxame) contendo todas as partículas geradas; *Pbest*, que representa um conjunto contendo a melhor situação de cada partícula $p \in P$, inicialmente *Pbest* é igual a *P*; e *Gbest*, um conjunto com as melhores partículas de cada subpopulação do enxame.

Algoritmo 1 – Gerar Enxame de Partículas

Entrada: m_s, n_s, n, t_{min}, t_{max}
Saída: P, $Pbest$, $Gbest$

```
 1: t ← t_min
 2: Para s ← 1 até n_s faça
 3:  Para i ← 1 até m_s faça
 4:    p ← GerarParticulaAleatoriamente (t, n)
 5:    P^s ← P^s ∪ {p}
 6:    Pbest^s ← Pbest^s ∪ {p}
 7:    Se i = 1 ou f(p) < f(Gbest^s) então
 8:      Gbest^s ← p
 9:    Fim-Se
10:  Fim-Para
11: t ← t + 1
12: Se t > t_max então
13:   t ← t_min
14: Fim-Se
15: Fim-Para
16: retorne P, Pbest, Gbest
```

No Algoritmo 1, o valor de t é inicializado com t_{min}. No laço das Linhas 2 a 15, são geradas as subpopulações P^s que compõem o enxame de partículas P. No laço das Linhas 3 a 10, as partículas são geradas e adicionadas a subpopulação P^s do enxame. Na Linha 4, a variável p recebe a partícula gerada pelo procedimento *GerarParticulaAleatoriamente.* A partícula possui o ponto de quebra t; o arranjo de máquinas, gerado de forma aleatória; as posições x e as velocidades v de cada máquina do arranjo. Inicialmente, as posições das máquinas são calculadas considerando a ordenação do arranjo. Na Linha 5, a partícula é adicionada à subpopulação P^s do exame P e também no enxame $Pbest$ (Linha 6). Na Linha 7, é verificado se a nova partícula é a melhor que a melhor partícula da subpopulação P^s, denominada $Gbest^s$. Caso afirmativo, a partícula $Gbest^s$ é atualizada (Linha 8). Na Linha 11, o valor de t é incrementado de uma unidade e a Linha 12 verifica se o limite para t foi excedido, reiniciando para t_{min} se necessário. Ao final, o procedimento retorna P, $Pbest$ e o $Gbest$ (Linha 16).

No Algoritmo 1, o exame de partículas inicial é criado composto por n_s subpopulações. Vale ressaltar que, se $n_s = t_{max} - t_{min} + 1$, cada subpopulação está associada a um valor de t, ou seja, as subpopulações se distinguem por terem soluções com pontos de quebra (valores de t) distintos. Entretanto, a quantidade de subpopulações n_s poderá ser maior que $t_{max} - t_{min} + 1$ e assim, mais de uma subpopulação pode conter partículas com o mesmo valor de t. Em nossa implementação, n_s é no mínimo $t_{max} - t_{min} + 1$. Após essa etapa, o PSO-DRLP realizará a busca por soluções através das atualizações das velocidades e posições de cada partícula contida no enxame P fazendo uso das informações de $Pbest$ e $Gbest$.

3.2 Atualização da Velocidade e Posição das Partículas

No PSO-DRLP, o vetor velocidade é atualizado pela Equação (4), na qual é definido o fator de inércia ω, não sendo então utilizados os limitantes $[v_{min}, v_{max}]$ para a velocidade nas partículas.

No processo de atualização do vetor posições de cada partícula do PSO, as posições das máquinas são recalculadas, definindo assim, onde cada máquina será alocada ao longo do corredor, e implicitamente, são definidas as distâncias dos pares (i, j) de máquinas no arranjo. Após a atualização das posições, um procedimento verifica se existem sobreposições de máquinas nas duas linhas do arranjo utilizando um procedimento que percorre os pares de máquinas $(i, j;\ i < j)$ em cada linha do arranjo verificando se as máquinas i e j estão sobrepostas. Se as máquinas i e j estão sobrepostas, a posição de j é redefinida para um ponto adjacente à i, ou seja, $x_i = x_j + 0{,}5 \times (l_i + l_j)$, removendo assim a sobreposição.

3.3 Busca Local para o PSO-DRLP

No processo de atualização das partículas, as posições das máquinas ao longo do eixo-x são definidas e consequentemente, as distâncias entre as máquinas também são determinadas, gerando assim um layout viável para o DRLP. Contudo, a alteração da disposição das máquinas no arranjo não é afetada. Deste modo, para intensificar a busca por novas soluções foi adicionada ao processo iterativo do PSO uma busca local composta de duas fases.

As fases da busca local alternam a utilização de movimentos de inserção e 2-opt. Esses movimentos são amplamente utilizados em procedimento heurísticos propostos para os diversos problemas de layout de facilidades,

na tentativa de intensificar a busca no espaço de soluções [18, 31, 35]. A Figura 4 exemplifica os movimentos utilizados nas duas buscas local propostas para o PSO-DRLP.

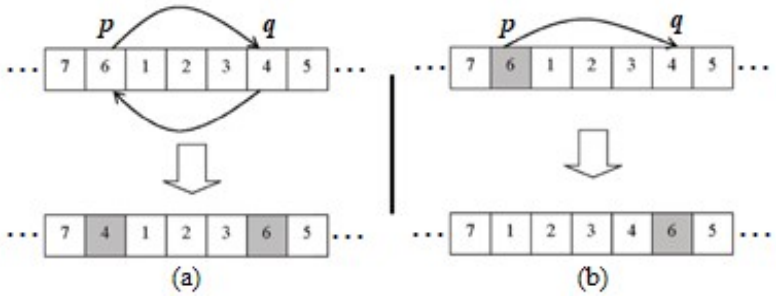


Figura 4. Movimentos de vizinhança (a) 2-opt e (b) inserção

Na Figura 4(a) é exemplificado um movimento de troca 2-opt, onde as máquinas das posições p e q têm suas posições trocadas no arranjo. Já na Figura 4(b), o movimento realizado é o de inserção, assim, a máquina que estava na posição p é inserida na nova posição q, e nesse caso, as máquinas intermediárias são deslocadas para que a máquina p seja inserida na nova posição. Em ambos os movimentos as posições são recalculadas de modo que não ocorreram sobreposições de máquinas.

Na primeira etapa da busca local, são geradas sequencialmente os pares de posições (r, s), na tentativa de inserir a máquina da posição $r = 1, ..., t$, na nova posição $s = 1, ..., t$ com $r \neq s$. Em seguida, são gerados os pares (r, s) para as posições $r = t+1, ..., n$, na tentativa de inserir a máquina da posição r nas posições $s = t+1, ..., n$; com $r \neq s$. Caso a nova solução vizinha melhore o valor da função objetivo corrente, a solução corrente é atualizada e a busca continua para a próxima solução vizinha, realizando o movimento de inserção para o próximo par (r, s). Para manter o ponto de quebra t inalterado na solução, nessa primeira fase, não há inserção de máquinas de uma linha em outra do arranjo, permanecendo assim, a partícula com o mesmo valor de t da sua subpopulação no exame de partículas.

Após o término da primeira etapa, a solução é submetida a uma segunda etapa da busca local onde são geradas soluções vizinhas realizando movimentos de troca 2-opt. Para isso, são percorridas de forma sequencial as posições $p = 1, ..., n$ e $q = 1, ..., n$; $p \neq q$, na tentativa de obter uma solução vizinha melhor que a solução corrente. Caso a nova solução vizinha tenha o valor da função objetivo melhor que a solução corrente, a solução corrente é atualizada e a busca continua para a próxima solução vizinha, realizando o movimento 2-opt para o próximo par (p, q). Nessa rotina, mesmo sendo realizados movimentos entre máquinas em linha opostas, não há alteração no ponto de quebra t, pois a quantidade de máquinas em ambas as linhas não é alterada. O Algoritmo 2 apresenta o pseudocódigo da busca local proposta. O procedimento recebe como parâmetros S_0 que representa a solução corrente do PSO-DRLP; o número de máquinas do problema, n; o valor de t da solução S_0; e retorna uma solução vizinha S.

Algoritmo 2 – Busca Local do PSO-DRLP

Entrada: S_0, n, t
Saída: S

```
 1: S ← S0
 2: Faça
 3:   melhorou ← 0
     /* FASE 1 */
 4:  Para r ← 1 até t faça
 5:    Para s ←1 até t faça
 6:      Se r ≠ s então
 7:        S' ← ObterVizinhoMovimentoInserção (S, r, s)
 8:        Se f(S') < f(S) então
 9:          S ← S'
10:          melhorou ← 1
11:        Fim-Se
12:      Fim-Se
13:    Fim-Para
14:  Fim-Para
15:  Para r ← t + 1 até n faça
16:    Para s ← t + 1 até n faça
17:      Se r ≠ s então
18:        S' ← ObterVizinhoMovimentoInserção (S, r, s)
19:        Se f(S') < f(S) então
20:          S ← S'
21:          melhorou ← 1
22:        Fim-Se
23:      Fim-Se
24:    Fim-Para
25:  Fim-Para
     /* FASE 2 */
26:  Para p ← 1 até n faça
27:    Para q ← 1 até n faça
28:      Se p ≠ q então
29:        S' ← ObterVizinhoMovimento2-opt (S, p, q)
30:        Se f(S') < f(S) então
31:          S ← S'
32:          melhorou ← 1
33:        Fim-Se
34:      Fim-Se
35:    Fim-Para
36:  Fim-Para
37: Enquanto melhorou = 1
38: retorne S
```

Na Linha 1 do Algoritmo 2 a solução S é inicializada com a solução S_0. No laço das Linhas 2 a 37 o algoritmo repete enquanto uma solução vizinha melhor for encontrada. Na Linha 3, a *flag melhorou* é inicializada com zero. A primeira parte da busca local ocorre nas Linhas 4 a 25, onde são geradas e avaliadas soluções vizinhas utilizando os movimentos de inserção. Nos laços da Linha 4 e 5 são gerados os pares de posições referente a Linha 1 do arranjo, a serem avaliados pela busca local com movimentos de inserção. O procedimento *ObterVizinhoMovimentoInserção*, Linha 7, retorna uma solução vizinha de S que é atribuída à S'. Na Linha 8, é verificado se a solução S' é melhor que a solução corrente S. Caso afirmativo, a solução S é atualizada (Linha 9), e a *flag melhorou* recebe o valor 1 (Linha 10). De forma semelhante, nos laços das Linhas 15 e 16 são gerados os pares de posições referente a Linha 2 do arranjo, para serem avaliados. Na Linha 18 uma solução vizinha S' é obtida pelo movimento de inserção e na Linha 19 é verificado se a nova solução S' é melhor que a solução corrente S. Se verdadeiro, a solução S é atualizada na Linha 20 e a *flag melhorou* recebe o valor 1 (Linha 21). A segunda parte da busca local, onde são utilizados os movimentos 2-opt, ocorre nas Linhas 26 a 36, onde são gerados os valores das posições p. Para cada valor de p, o laço das Linhas 27-35 gera os valores das posições q. Na Linha 29, uma solução vizinha S' é obtida pelo procedimento *ObterVizinhoMovimento2-opt*, com a aplicação do movimento 2-opt, considerando as posições p e q. Na linha 30, é verificado se a nova solução S' é melhor que a solução corrente S. Caso afirmativo, a solução corrente S é atualizada e a *flag melhorou* recebe o valor 1 (Linhas 31 a 32). Ao final do procedimento, o algoritmo retorna à solução S (Linha 38).

Vale destacar dois pontos interessantes na busca local proposta. O primeiro é que, cada fase da busca local não é executada como uma rotina de busca local padrão, onde uma lista de pares (p, q) é gerada e a busca é realizada sobre a lista, até que não seja encontrada uma solução vizinha melhor. Como pode ser observado no Algoritmo 2, ambas as etapas realizam um número exato de movimentos, sem reinício, o que difere de uma busca local padrão, sendo que na primeira etapa são aplicados movimentos de inserção, e na segunda etapa, movimentos 2-opt. Após finalizar a segunda etapa, caso tenha ocorrido alguma melhora, o processo é reiniciado com a solução corrente.

O segundo ponto é que a busca local, sendo executada a cada nova solução obtida pelo PSO, também trabalha como um procedimento de diversificação no enxame de partículas pois, a cada atualização do vetor posição pela Equação (3) do PSO, uma nova solução é gerada com a mesma ordenação de máquinas, mas com alteração no posicionamento das mesmas, e no distanciamento dos pares no arranjo, ao longo do eixo-x. Com isso, um movimento de inserção ou 2-opt que antes não havia melhorado a solução corrente, agora, com o novo reposicionamento definido pela Equação (3) poderá melhorar e, se assim for, ocorrerá uma alteração na disposição das máquinas no arranjo, possibilitando então uma diversificação nas partículas do enxame.

3.4 O Algoritmo PSO-DRLP proposto

Como dito antes, o algoritmo proposto é composto por duas fases. Inicialmente são gerados os exames P e o correspondente *Pbest,* que armazena as melhores posições no espaço de busca de cada partícula. Além do conjunto *Gbest*, com as melhores partículas de cada subpopulação, a melhor solução (partícula) global, g_{best}, é obtida. Na primeira fase do PSO-DRLP, para cada subpopulação, são realizadas as atualizações das partículas (velocidade e posição das máquinas); a cada atualização de uma partícula p, é realizada a verificação e ajuste, se necessário, para eliminar possíveis sobreposições de máquinas no arranjo. Em seguida, como a partícula p contém uma nova solução, esta é submetida à busca local. Ao final de cada iteração é verificado se a partícula p melhorou em relação a sua melhor posição e também em relação à melhor solução da sua subpopulação. A cada iteração, a melhor partícula global, g_{best}, é mantida atualizada.

A ideia da primeira fase é que o PSO otimize cada subpopulação individualmente, não realizando troca de informação entre as subpopulações. Assim cada partícula leva em consideração sua melhor posição e a melhor partícula da subpopulação a qual pertence.

Na segunda fase do PSO-DRLP, o PSO irá otimizar as partículas pertencentes ao conjunto *Gbest*, que são os "guias" de cada subpopulação, introduzindo assim, uma comunicação entre as subpopulações. Para isso, são utilizadas as informações de cada partícula em *Gbest* e também a melhor partícula global g_{best}. PSO-DRLP é exemplificado no pseudocódigo apresentado no Algoritmo 3.

O Algoritmo 3, descrito a seguir, recebe como entrada os seguintes parâmetros:

- n: o tamanho do problema tratado;
- m_s, número de partículas para cada subpopulação;
- n_s, a quantidade de subpopulações;
- T_{max}, tempo máximo de execução do algoritmo;
- $tempo_{\omega}$, tempo de referência para o decremento do fator de inércia ω;
- ω_{max}, valor máximo inicial para o fator de inércia ω;
- ω_{min}, valor mínimo final para o fator de inércia ω;
- c_1: coeficiente da componente de auto reconhecimento;
- c_2: coeficiente da componente social; e
- k: valor inteiro para a definição do intervalo [t_{min}, t_{max}].

Na Linha 1, são definidos os valores mínimos e máximo, t_{min} e t_{max}, utilizados para os pontos de quebra dos arranjos. Na Linha 2, são inicializados o enxame de partículas inicial P, *Pbest* e *Gbest*. Na Linha 3 é inicializado o fator de inércia ω e a variável $tempo_{base}$ com o tempo atual. A variável $tempo_{base}$ armazena o tempo atual e é utilizado para a atualização do valor de ω na Linha 46. Na Linha 4, a melhor solução encontrada pelo PSO-DRLP, g_{best}, é inicializada com a melhor solução do conjunto *Gbest*. No laço das Linhas 5 a 51 ocorre o processo iterativo do PSO-DRLP.

Algoritmo 3 – Algoritmo PSO-DRLP proposto

Entrada: n, m_s, n_s, T_{max}, $tempo_{\omega}$, ω_{max}, ω_{min}, c_1, c_2, k
Saída: g_{best}

1: $t_{min} \leftarrow max\ \{1; \left\lfloor \frac{n}{2} \right\rfloor\}$, $t_{max} \leftarrow min\ \{n; \left\lfloor \frac{n}{2} \right\rfloor + k\}$
2: $[P, Pbest, Gbest] \leftarrow GerarEnxamePartículas\ (m_t, n_s, n, t_{min}, t_{max})$
3: $\omega \leftarrow \omega_{max}$, $tempo_{base} \leftarrow tempo_{corrente}$
4: $g_{best} \leftarrow \underset{1 \le s \le n_s}{argmin} \{Gbest^s\}$
5: **Enquanto** $tempo_{corrente} < T_{max}$ **faça**
/ FASE 1 */*
6: melhorou ← V
7: **Enquanto** melhorou = V **E** $tempo_{corrente} < T_{max}$ **faça**
8: melhorou ← F
9: **Para** $s \leftarrow 1$ **até** n_s **faça**
10: **Para cada** $p_i \in P^s$ **faça**
11: $p_{i,v} \leftarrow \omega p_v + r_1 c_1 (Pbest^s_{i,x} - p_{i,x}) + r_2 c_2 (Gbest^s_{i,x} - p_{i,x})$
12: $p_{i,x} \leftarrow p_{i,x} + p_{i,v}$
13: $p_i \leftarrow VerificarSobreposiçãoAjustar\ (p_i)$
14: $p_i \leftarrow BuscaLocal\ (p_i, n, t)$
15: **Se** $f(p_i) < f(Pbest^s_i)$ **então**
16: $Pbest^s_i \leftarrow p_i$
17: **Se** $f(p_i) < f(Gbest^s_i)$ **então**
18: $Gbest^s_i \leftarrow p_i$
19: **Se** $f(p_i) < f(g_{best})$ **então**
20: $g_{best} \leftarrow p_i$, $melhorou \leftarrow$ V
21: **Fim-Se**
22: **Fim-Se**
23: **Fim-Se**
24: **Fim-Para**
25: **Fim-Para**
26: **Fim-Enquanto**
/ FASE 2 */*
27: $P^{Gbest} \leftarrow Gbest$; $Pbest^{Gbest} \leftarrow Gbest$;
28: melhorou ← V
29: **Enquanto** melhorou = V **E** $tempo_{corrente} < T_{max}$ **faça**
30: melhorou ← F
31: **Para cada** $p_i \in P^{Gbest}$ **faça**
32: $p_{i,v} \leftarrow p_{i,v} + r_1 c_1 (Pbest^{Gbest}_{i,x} - p_{i,x}) + r_2 c_2 (g_{best,x} - p_{i,x})$
33: $p_{i,x} \leftarrow p_{i,x} + p_{i,v}$
34: $p_i \leftarrow VerificarSobreposiçãoAjustar\ (p_i)$
35: $p_i \leftarrow BuscaLocal\ (p_i, n, t)$
36: **Se** $f(p_i) < f(Pbest^{Gbest}_i)$ **então**
37: $Pbest^{Gbest}_i \leftarrow p_i$
38: **Se** $f(p_i) < f(g_{best})$ **então**
39: $g_{best} \leftarrow p_i$, $melhorou \leftarrow$ V
40: **Fim-Se**
41: **Fim-Se**
42: **Fim-Para**
43: **Fim-Enquanto**
44: $Gbest \leftarrow Pbest^{Gbest}$
45: $tempo_{decorrido} \leftarrow tempo_{corrente} - tempo_{base}$
46: $\omega \leftarrow \omega_{max} - (\omega_{max} - \omega_{min}) \times (tempo_{decorrido})/tempo_{\omega})$
47: **Se** $(tempo_{decorrido} \ge tempo_{\omega})$ **então**
48: $tempo_{base} \leftarrow tempo_{corrente}$
49: $\omega \leftarrow \omega_{max}$
50: **Fim-Se**
51: **Fim-Enquanto**
52: **retorne** g_{best}

Na Linha 6, a *flag melhorou* é iniciada com F. A primeira fase do PSO-DRLP é definida no laço das Linhas 7 a 26. Nessa primeira fase, são percorridas todas as subpopulações, laço das Linhas 9 a 25, identificadas pelo índice s superior nos conjuntos P, *Pbest* e *Gbest.* No laço das Linhas 10 a 24, as partículas pertencentes a cada subpopulação P^s são atualizadas. As velocidades ($p_{i,v}$) das máquinas de cada partícula p_i são recalculadas na Linha 11 e, na Linha 12, as posições ($p_{i,x}$) são atualizadas. Após a atualização, é verificado se ocorreu alguma sobreposição de máquinas no arranjo e, se necessário, as posições são ajustadas (Linha 13). Na Linha 14 a partícula p_i é atualizada com a solução retornada da busca local. Na Linha 15, é verificado se a nova partícula p_i melhora a melhor solução corrente, armazenada em $Pbest^s_i$. Se verdadeiro, $Pbest^s_i$. é atualizada na Linha 16. Na Linha 17 é verificado se a partícula corrente p_i é melhor em relação a melhor partícula, $Gbest^s_i$, da subpopulação s.

Caso verdadeiro, a partícula $Gbest_i^s$ é atualizada na Linha 18. Na Linha 19, a partícula p_i é comparada com a melhor solução corrente g_{best}. Se verdadeiro, a solução g_{best} e *flag melhorou* são atualizadas na Linha 20. Ao final da primeira fase, que termina quando não houver melhoria na solução g_{best} ou o tempo de execução tenha se esgotado, tem-se a melhor solução do PSO-DRLP, g_{best} e as melhores soluções de cada subpopulação *Gbest*, que serão utilizados na segunda fase do PSO-DRLP.

A segunda fase do PSO-DRLP inicia na Linha 27, com a definição dos conjuntos/enxames P^{Gbest} e $Pbest^{Gbest}$ que são inicializados com as partículas contidas no conjunto *Gbest*. Na Linha 28, a *flag melhorou* é reiniciada com V. A partir desse ponto, no laço das Linhas 29 a 43, tem-se a segunda fase do PSO-DRLP. No laço das Linhas 31 a 42, são atualizas as partículas em P^{Gbest}, $Pbest^{Gbest}$ e a melhor solução global g_{best} utilizando o princípio da meta-heurística PSO. Na Linha 32, o vetor velocidade v de cada partícula é atualizada e na Linha 33, o vetor posições x é recalculado. Na Linha 34, verifica se ocorre sobreposições no novo arranjo, e se necessário efetua a correção. Na Linha 35, a partícula é submetida à busca local. Na Linha 36 é verificado se a solução atual melhora em relação a sua melhor posição em $Pbest^{Gbest}$. Se verdadeiro, atualiza a partícula $Pbest_i^{Gbest}$ (Linha 37). Na Linha 38, é verificado se a partícula corrente é melhor que a melhor solução global do PSO-DRLP. Se verdadeiro, g_{best} e a *flag melhorou* são atualizadas na Linha 39. A final da segunda fase, encerrada quando não ocorrer melhoria na solução g_{best} ou pelo esgotamento do tempo de execução, atualiza-se o conjunto *Gbest* com as soluções obtidas pela segunda fase do PSO-DRLP (Linha 44), que serão utilizadas na primeira fase do PSO-DRLP na próxima iteração.

Na Linha 45, obtém-se o tempo decorrido. O valor de ω é recalculado, levando em consideração a variação do tempo de execução da atualização de todas as partículas ($tempo_{decorrido}$) e o tempo de referência $tempo_{\omega}$ (Linha 46). Na Linha 47 o tempo decorrido é verificado em relação ao limite de referência $tempo_{\omega}$. Se verdadeiro, as variáveis $tempo_{base}$ e ω são reiniciadas, respectivamente nas Linhas 48 e 49. Ao final, o algoritmo PSO-DRLP retorna a melhor solução global g_{best} encontrada (Linha 52).

O uso da informação do tempo decorrido e o tempo de referência $tempo_{\omega}$ é utilizado para manter o padrão de decrescimento de ω, independentemente do tempo máximo de execução do algoritmo (T_{max}).

4 Experimentos Computacionais

Para a realização dos experimentos computacionais, foi utilizado um *laptop* com processador Intel i7-7500U 2,7 GHz com 8GB de memória com sistema operacional MS Windows 10. O PSO-DRLP foi implementado em linguagem de programação C e compilados no compilador GCC 5.1.0 x64 com as *flags* de otimização -O3.

Nos experimentos foram utilizadas trinta e nove instâncias com matriz de fluxos simétricas e folgas implícitas, sendo seis instâncias para cada $n = \{11, 12, 13\}$ de [7]; uma com $n = 15$ [2]; duas com $n = 16$ [5]; uma com $n = 17$ e uma com $n = 18$ de [3]; cinco instâncias com $n = 30$ [10]; sete com $n = 40$ [8]; e quatro instâncias com tamanhos $n = \{60, 70, 75, 80\}$ de [9]. Também são consideradas doze instâncias com matriz de fluxos assimétricos e folgas explícitas, sendo sete para cada $n = \{15, 20, 25, 30, 35, 40, 45\}$ e cinco com $n = 50$ de [34]. Tais instâncias foram consideradas na presente análise por serem as mais complexas e utilizadas nas comparações dos estudos mais recentes. As tabelas apresentam os nomes das instâncias, os tamanhos das instâncias, as quantidades de instâncias para cada tamanho e por último a origem das instâncias na literatura.

4.1 Calibração dos Parâmetros

No PSO-DRLP tem-se os seguintes parâmetros que precisam ser definidos: m_s, número de partículas do enxame; n_s, número de subpopulações do exame de partículas; $tempo_{\omega}$, tempo de referência para o decremento do fator de inércia ω; ω_{max} e ω_{min}, que formam o intervalo para o fator de inércia ω; e as constantes c_1 e c_2. Além disso, os parâmetros n e k são fixos, sendo n definido pelo tamanho da instância e $k = 4$ [8].

Para determinar os parâmetros do PSO-DRLP, utilizou-se a metodologia apresentada em [18], onde os autores utilizam a meta-heurística PSO para a calibração do algoritmo GRASP-F para o SRFLP. Assim, o PSO calibrador [18] foi adaptado para a calibração do PSO-DRLP, com o objetivo de obter a calibração dos valores do vetor posição x_i e a definir os limites para os possíveis valores de cada parâmetro a serem gerados. Nesse trabalho, definiu-se como entrada do PSO calibrador o seguinte vetor de parâmetros e seus limites: m_s, $5 \leq x_{i1} \leq 100$; n_s, $5 \leq x_{i2} \leq 100$; $tempo_{\omega}$, $30 \leq x_{i3} \leq 120$; ω_{max}, $0{,}4 \leq x_{i4} \leq 0{,}9$; ω_{min}, $0{,}4 \leq x_{i5} \leq \omega_{max}$; c_1, $0 \leq x_{i6} \leq 2$; e c_2, $0 \leq x_{i7} \leq 2$ para cada partícula i do PSO de calibração. Na calibração, $T_{max} = tempo_{\omega}$.

O PSO de calibração foi executado para um subconjunto contendo quatro instâncias de diferentes complexidades, sendo duas instâncias com matriz de fluxos simétricas e folgas implícitas (N30_03 e DRLP_02) e duas instâncias com matriz de fluxos assimétricas e folgas explícitas (Murray243 e Murray322). Os parâmetros usados no PSO de calibração foram: $m = 10$ (dez partículas) e $IterMax = 36$ (total de iterações do PSO de calibração). Os resultados da calibração são mostrados na Tabela 2.

Tabela 2: Parâmetros encontrados para o PSO-DRLP.

Instâncias	Parâmetros	$tempo_{\omega}$	m_s	n_s	ω_{max}	ω_{min}	c_1	c_2
N30_02	P1	79	76	86	0,708602	0,510998	1,251,112	0,050853
N30_03	P2	91	48	99	0,405524	0,404195	1,565,725	1,133,444
DRLP_02	P3	52	28	75	0,483970	0,475621	0,873352	0,754753
DRLP_03	P4	92	75	59	0,643440	0,499749	0,977237	0,266861

Os parâmetros encontrados são diferentes para cada instância. Assim, para determinar o melhor conjunto de parâmetros baseada nos valores das soluções média encontrada para cada conjunto de parâmetros [18], o PSO-DRLP foi executado novamente, com o subconjunto de instâncias para cada configuração de parâmetro com T_{max} = 120 segundos por instância e as médias dos resultados são mostrados na Tabela 3. A Tabela 3 é estruturada da seguinte forma: a primeira coluna apresenta o nome do conjunto de parâmetros e a segunda coluna mostra a média das soluções encontradas pelo PSO-DRLP, para cada parâmetro. A qualidade do conjunto de parâmetros é medida pelo valor da coluna Média. Quanto menor o valor, melhor o parâmetro [18].

Tabela 3: Desempenho do PSO-DRLP para cada conjunto de parâmetros.

Parâmetros	*Média*
P1	1476081,8
P2	1476405,9
P3	1476734,5
P4	**1475791,7**

Pelos resultados da Tabela 3, o conjunto de parâmetros denominado P4 (em negrito) apresentou o melhor desempenho, tendo o menor valor dentre os demais, 1475791,7. Assim, o conjunto de parâmetros denominado P4 ($tempo_{\omega}$ = 92; m_s = 75; n_s = 59; ω_{max} = 0,643440; ω_{min} = 0,499749; c_1 = 0,977237; c_2 = 0,266861) foi utilizado nos experimentos computacionais para o PSO-DRLP. Ressalta-se que a média para os quatro conjuntos de parâmetros testados se apresenta com valores muito próximos, o que evidencia a robustez do algoritmo proposto.

4.2 Comparação do PSO-DRLP com a literatura

Para a avaliação do PSO-DRLP, foram realizadas dez execuções para cada instância. A melhor solução obtida pelo PSO-DRLP é denotada por *Melhor*. A média dos valores das soluções encontradas pelo PSO-DRLP é denominada *Média*. Também foram calculadas as medidas de desempenho *DP* e *Desv* (%), sendo *DP* o desvio padrão e *Desv* (%) o desvio médio relativo calculado como *Desv* (%) = $100 \times \frac{Média - v}{v}$. O tempo limite de execução do PSO-DRLP, T_{max}, é informado em segundos.

As seções seguintes apresentam os resultados do PSO-DRLP, sendo que na Seção 4.2.1 são indicados os resultados considerados de médio porte e, para as Seções 4.2.2 a 4.2.4, são expostos os resultados para as instâncias consideradas de grande porte para o DRLP. Os resultados são comparados com as melhores soluções conhecidas na literatura.

4.2.1 Comparação dos Resultados para instâncias com $11 \leq n \leq 18$ máquinas com matriz de fluxos simétrica e folgas implícitas

A Tabela 4 apresenta a comparação dos resultados dos experimentos com o PSO-DRLP para as instâncias com tamanhos $11 \leq n \leq 18$ máquinas. Essas instâncias apresentam uma matriz de fluxos simétrica e os valores das folgas implícitos. Para as instâncias com tamanhos $11 \leq n \leq 16$, os melhores valores conhecidos são ótimos, sendo $11 \leq n \leq 13$ obtidos em [7], $n = 15$ em [37] e $n = 16$ em [16]. Para a instância Am17, o melhor valor conhecido foi obtido pelas heurísticas em [8]. O valor ótimo para a instância Am18 não é conhecido, sendo o valor disponível (5372,5), obtido pelo PSO-DRLP com T_{max} = 3600s.

A Tabela 4 está estruturada da seguinte forma: a primeira coluna mostra os nomes das instâncias; a segunda coluna, os tamanhos das instâncias; a coluna *v* apresenta os melhores valores conhecidos; os valores das melhores soluções obtidas pelo PSO-DRLP nas 10 execuções estão indicados na coluna *Melhor*, seguido dos valores médios das soluções obtidas; a coluna *Desv* (%) apresenta os desvios relativos entre a média das soluções e a melhor solução conhecida; os desvios padrões são mostrado na coluna *DP*, seguido da coluna T_{avg}, que apresenta a

média dos tempos máximos (T_{max}) de execução utilizados para cada instância, em segundos. A última linha da Tabela 4 apresenta a média sobre todas as colunas da tabela.

Tabela 4: Resultados para instâncias de pequeno porte com $11 \leq n \leq 15$ máquinas.

Instâncias	*n*	*v*	*Melhor*	*Média*	*Desv* (%)	*DP*	T_{avg}
Am11a	11	**5559,0**	**5559,0**	5559,0	0,000	0	5,4
Am11b	11	**3655,5**	**3655,5**	3655,5	0,000	0	5,2
Am11c	11	**3832,5**	**3832,5**	3832,5	0,000	0	5,2
Am11d	11	**906,5**	**906,5**	906,5	0,000	0	5,3
Am11e	11	**578,0**	**578,0**	578,0	0,000	0	5,3
Am11f	11	**825,5**	**825,5**	825,5	0,000	0	5,2
Am12a	12	**1493,0**	**1493,0**	1493,0	0,000	0	5,4
Am12b	12	**1606,5**	**1606,5**	1606,5	0,000	0	5,2
Am12c	12	**2012,5**	**2012,5**	2012,5	0,000	0	5,4
Am12d	12	**1107,0**	**1107,0**	1107,0	0,000	0	5,4
Am12e	12	**1066,0**	**1066,0**	1066,0	0,000	0	5,2
Am12f	12	**997,5**	**997,5**	997,6	0,005	0,2	5,3
Am13a	13	**2456,5**	**2456,5**	2456,5	0,000	0	5,5
Am13b	13	**2864,0**	**2864,0**	2864,0	0,000	0	5,3
Am13c	13	**4136,0**	**4136,0**	4136,0	0,000	0	5,4
Am13d	13	**6164,5**	**6164,5**	6164,5	0,000	0	5,3
Am13e	13	**6502,5**	**6502,5**	6502,5	0,000	0	5,6
Am13f	13	**7699,5**	**7699,5**	7699,5	0,000	0	5,8
Am15	15	**3195,0**	**3195,0**	3195,6	0,019	0,5	5,0
P16a	16	**7365,5**	**7365,5**	7365,5	0,000	0	5,3
P16b	16	**5870,5**	**5870,5**	5870,8	0,005	0,9	5,2
Am17	17	**4655,0**	**4655,0**	4655,0	0,000	0	5,6
Am18	18	**5372,5**	**5372,5**	5374,9	0,045	2,1	5,5
Média		3506,4	3508,3	3508,6	0,053	0,2	5,4

Valores ótimos para $11 \leq n \leq 13$ foram obtidos em [7], $n = 15$ em [37] e $n = 16$ em [16]; o valor de Am17 é apresentado em [8] e Am18 foi obtido com o PSO-DRLP com T_{max}=3600s. Os valores mínimos estão destacados em negrito.

Para o conjunto de testes da Tabela 4, o PSO-DRLP encontrou as melhores soluções da literatura para todas as instâncias. O algoritmo apresenta robustez, como pode ser notado pela baixa média dos valores de desvios médios relativos e de desvio padrão, apresentados respectivamente nas colunas *Desv* (%) e DP, a um custo computacional extremamente baixo, conforme indicado na coluna T_{avg}.

4.2.2 Comparação dos Resultados para instâncias com $30 \leq n \leq 40$ máquinas com matriz de fluxos simétricas e folgas implícitas

A Tabela 5 apresenta os resultados para as instâncias do DRLP com $30 \leq n \leq 40$ máquinas com matriz de fluxos simétrica e folgas implícitas. Os resultados são comparados com os melhores resultados disponíveis até a presente data. Os valores em *v* foram obtidos em [8] a partir de um computador Intel Core i7-3770 CPU 3,40 GHZ com 8 GB de RAM e usando o solver CPLEX 12.7.1.0 para resolver o modelo de programação linear. A Tabela 5 está estruturada como a Tabela 4.

Para tornar a comparação entre computadores com processadores diferentes mais justa, os tempos de execução do PSO-DRLP foram ajustados de acordo com os parâmetros *CPU Mark*, disponível no *site cpubenchmark* (*https://www.cpubenchmark.net*). O processador utilizado em [8] possui o valor 6360 contra 3701 do processador utilizado nesse trabalho. Assim, os tempos de execução, T_{max}, utilizados pelo PSO-DRLP são iguais aos tempos do algoritmo *Heuristica2+LP* [8] multiplicados pelo fator $\left(\frac{6360}{3701}\right)$. Na Tabela 5, os tempos apresentados são as médias dos tempos de execução do PSO-DRLP, podem sofrer variações dependendo do tempo de execução de cada iteração do algoritmo.

Tabela 5: Comparação dos resultados para as instâncias com $30 \leq n \leq 40$ máquinas.

Instâncias	*n*	*v*	*Melhor*	*Média*	*Desv* (%)	*DP*	T_{avg}
N30_01	30	**4115,0**	**4115,0**	4115,0	0,000	0	3413,9
N30_02	30	**10771,0**	**10771,0**	10771,0	0,000	0	2031,9
N30_03	30	**22692,0**	**22692,0**	22694,1	0,009	2,7	1665,1
N30_04	30	**28390,0**	**28390,0**	28391,5	0,005	1,4	1676,0
N30_05	30	**57393,5**	**57393,5**	57393,7	0,000	0,6	1463,1
DRLP_01	40	99525,5	**99508,0**	99524,9	-0,001	11,0	5436,6
DRLP_02	40	300973,5	**300967,0**	300990,9	0,006	22,0	5322,6
DRLP_03	40	**416257,0**	416260,0	416278,1	0,005	12,0	5404,6
DRLP_04	40	**207510,0**	207517,0	207530,5	0,010	9,6	5602,0
DRLP_05	40	**193748,0**	**193748,0**	193768,6	0,011	16,1	5371,5
DRLP_06	40	**1881277,0**	1881361,0	1881412,8	0,007	47,0	5282,0
DRLP_07	40	545239,0	**545099,0**	545377,1	0,025	197,3	5196,1
Média		313991,0	313985,1	314020,7	0,006	26,6	3988,8

Melhores soluções disponíveis em [8]. Os valores mínimos estão destacados em negrito.

Para as instâncias da Tabela 5, o PSO-DRLP alcançou as melhores soluções da literatura para todas as instâncias com $n = 30$, sendo que para as instâncias N30_01 e N30_02, o PSO-DRLP encontrou o melhor valor conhecido nas 10 execuções, como pode ser observado na coluna *DP*. Para as instâncias com $n = 40$, o PSO-DRLP alcançou quatro melhores soluções conhecidas, sendo que melhorou três novas melhores soluções conhecidas, nas instâncias DRLP_01, DRLP_02 e DRLP_07.

O PSO-DRLP é capaz de gerar soluções de alta qualidade, conforme os baixos valores dos desvios médios relativos. Além disso, pelos baixos valores dos desvios padrões obtidos, o algoritmo apresenta uma convergência consistente para esse conjunto de instâncias. Mesmo o algoritmo proposto não tendo encontrado as melhores soluções para as instâncias DRLP_03, DRLP_04 e DRLP_06, as soluções encontradas para essas três instâncias estão bem próximas dos melhores valores conhecidos. De modo geral, o PSO-DRLP apresentou a média das melhores soluções obtidas melhor que as soluções de referência.

4.2.3 Comparação dos Resultados para instâncias com $60 \leq n \leq 80$ máquinas com matriz de fluxos simétrica e folgas implícitas

A Tabela 6 apresenta os resultados encontrados para as instâncias do PSO-DRLP com $60 \leq n \leq 80$ máquinas que, até a presente data, são as maiores instâncias consideradas na literatura para o DRLP. Essas instâncias também possuem a matriz de fluxos simétrica e os valores implícitos das folgas. As soluções do PSO-DRLP são comparadas com os melhores valores da literatura, obtidos em [23], executando o algoritmo denominado DBA, em um computador com processador Intel Core i5-4460 3,20 GHz. A Tabela 6 está estruturada como a Tabela 4.

Novamente, os tempos de execução do PSO-DRLP foram ajustados de acordo com os parâmetros de comparação entre processadores disponível no *site cpubenchmark*. O processador utilizado em [23] possui um *CPU Mark* de 4777. Assim, os tempos utilizados em [23] foram multiplicados pelo fator $\left(\frac{4777}{3701}\right)$ e os valores obtidos, foram utilizados como tempos limites de execução do PSO-DRLP.

Tabela 6: Comparação dos resultados para instâncias com $60 \leq n \leq 80$ máquinas.

Instância	*n*	*v*	*Melhor*	*Média*	*Desv* (%)	*DP*	T_{avg}
AKV60_01	60	739233,0	**739199,0**	739395,6	0,022	93,0	395,3
AKV70_01	70	**764511,0**	764797,0	765016,7	0,066	136,6	783,1
AKV75_01	75	**1197240,0**	1197608,5	1197885,2	0,054	181,5	1037,4
AKV80_01	80	**1034735,5**	1034985,5	1035191,5	0,044	143,5	1385,0
Média		933929,9	934147,5	934372,2	0,047	138,6	900,2

Melhores soluções disponíveis em [23]. Os valores mínimos estão destacados em negrito.

Para os testes mostrados na Tabela 6, o PSO-DRLP conseguiu melhorar o melhor valor conhecido de uma instância, AKV60_01. Para as outras instâncias o PSO-DRLP obteve valores muito próximos dos melhores disponíveis, tendo o maior desvio percentual de 0,066%, para a instância AKV70_01. Os baixos valores dos

desvios padrões mostram que o PSO-DRLP mantém uma capacidade de convergência consistente para esse conjunto de testes com as maiores instância tratadas nesse trabalho.

4.2.4 Comparação dos Resultados para instâncias com $15 \leq n \leq 50$ máquinas com matriz de fluxos assimétrica e folgas explicitas

Nesse conjunto de testes, foram consideradas nove instâncias apresentadas em [34]. Essas instâncias têm tamanhos de $15 \leq n \leq 50$ máquinas com matriz de fluxos assimétricas e folgas explícitas. Para as sete instâncias com $15 \leq n \leq 45$ máquinas, os melhores resultados são apresentados em [23] e para as seis instâncias com $n = 50$ máquinas, os melhores resultados estão em [8]. A Tabela 7 apresenta a comparação dos resultados encontrados pelo PSO-DRLP com as melhores soluções da literatura e tem a mesma estrutura da Tabela 4. Os tempos computacionais utilizados pelo PSO-DRLP foram ajustados como descritos nas subseções anteriores.

Tabela 7: Comparação dos resultados para instâncias com matriz de fluxos assimétricas e folgas explícitas

Instâncias	*n*	*v*	*Melhor*	*Média*	*Desv* (%)	*DP*	T_{avg}
Murray155	15	**105578,4**	105578,9	105581,9	0,003	2,5	8,0
Murray206	20	**311751,1**	311751,9	311758,6	0,002	13,1	25,7
Murray218	25	**624653,3**	624661,4	625005,9	0,056	299,4	65,7
Murray231	30	**918759,6**	919559,3	920146,8	0,151	543,9	145,6
Murray243	35	**1257631,8**	1258197,9	1260047,1	0,192	869,7	316,4
Murray256	40	**2118158,1**	2120279,3	2122357,8	0,198	1287,8	461,1
Murray309	45	**2628585,5**	2628712,5	2631780,5	0,122	1993,3	432,5
Murray322	50	**4018716,2**	4023454,3	4026030,2	0,182	1636,4	251,6
Murray324	50	4786649,1	**4783908,5**	4794911,3	0,173	4351,8	254,0
Murray325	50	**4006421,4**	4011891,5	4015394,7	0,224	2713,7	252,8
Murray327	50	4237417,9	**4232807,5**	4240714,4	0,078	3325,9	255,8
Murray328	50	**5273616,5**	5275271,5	5278987,2	0,102	2564,0	253,2
Média		2523994,9	2524672,9	2527726,4	0,124	1633,5	226,9

Melhores soluções para $15 \leq n \leq 45$ estão disponíveis em [23] e $n = 50$ em [8]. Os valores mínimos estão destacados em negrito.

No geral, para esse conjunto de instâncias, o PSO-DRLP obteve soluções muito próximas dos melhores valores encontrados na literatura, como pode ser visto na coluna *Desv* (%), sendo 0,224% o maior desvio relativo para a instância Murray325, em comparação com a melhor solução obtida em [8]. Evidencia-se que o PSO-DRLP melhorou os melhores valores da literatura para duas instâncias com $n = 50$, Murray324 e Murray327.

5 Conclusões e Recomendações Futuras

Esse trabalho apresentou uma abordagem puramente heurística para solucionar o problema de layout em duas linhas (DRLP), baseada na meta-heurística PSO. O PSO-DRLP proposto é constituído de duas fases e com o enxame de partículas organizados em subpopulações. O algoritmo proposto foi avaliado utilizando diversas instâncias do DRLP. Os testes realizados variam de instâncias com $n = \{11, 12, 13, 15, 16, 17, 18\}$ a instâncias consideradas de grande porte com até $n = 80$ máquinas.

Os resultados obtidos mostraram que o PSO-DRLP possui uma convergência consistente para soluções de alta qualidade com baixos valores dos desvios relativos e desvios padrões, mesmo com o aumento dos tamanhos das instâncias. No geral, a média das melhores soluções encontradas pelo algoritmo proposto em relação aos melhores valores de soluções encontradas na literatura possui um desvio de apenas 0,02%, sendo que o PSO-DRLP melhorou a melhor solução em seis instâncias. De modo geral, pelos resultados obtidos, o PSO-DRLP pode ser considerado uma boa estratégia para a resolução do DRLP, sendo capaz de obter soluções de alta qualidade em tempos computacionais relativamente baixos.

Agradecimentos

O segundo autor agradece à Fundação de Amparo à Pesquisa e Inovação do Espírito Santo - FAPES (processo 85308307/2019) pelo auxílio financeiro. O último autor foi financiado em parte pela Coordenação de

Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Código Financeiro 001; e em parte pela Fundação de Amparo à Pesquisa e Inovação do Espírito Santo (FAPES).

Apêndice A – Melhores resultados encontrados pelo PSO-DRLP

Nesse apêndice são apresentadas, em detalhes, os arranjos de máquinas das instâncias em que o PSO-DRLP encontrou soluções melhores que a literatura. A Tabela A.1 mostra as soluções onde, na primeira coluna, tem-se os nomes das instâncias, na segunda coluna os tamanhos das instâncias, na terceira coluna, o valor de *t* (ponto de quebra), na quarta coluna o valor da função objetivo e por fim, o arranjo encontrado, com a disposição das máquinas em duas linhas, tendo os elementos separados por ponto-e-vírgula e cada elemento representado pela máquina e o centro no eixo-*x*.

Tabela A.1 - Detalhes das soluções melhoradas pelo PSO-DRLP

Instâncias	*n*	*t*	FO	Arranjo
Am18	18	10	5372,5	L1=[17 25,0; 5 32,0; 4 37,0; 11 41,0; 12 45,0; 2 48,0; 14 51,0; 18 55,0; 3 62,0; 9 71,0] L2=[1 22,5; 7 36,0; 16 41,0; 8 45,0; 15 51,0; 10 57,5; 6 62,0; 13 68,0]
DRLP_01	40	21	99508,0	L1=[8 452,5; 19 467,0; 34 481,5; 21 494,0; 22 504,5; 2 514,5; 6 523,5; 33 531,5; 38 540,5; 26 552,0; 37 564,5; 35 574,0; 14 581,0; 36 588,0; 30 594,0; 12 600,0; 23 606,0; 1 614,0; 5 626,0; 18 639,0; 28 652,500000] L2=[40 449,5; 3 459,5; 39 470,0; 25 481,0; 31 489,0; 7 498,0; 24 508,5; 16 520,0; 11 532,0; 32 540,5; 17 551,0; 13 562,0; 9 571,5; 29 581,5; 15 589,0; 27 599,0; 10 614,0; 20 628,0; 4 641,500000]
DRLP_02	40	20	300967,0	L1=[27 279,0; 37 293,5; 24 306,0; 26 318,0; 1 328,0; 20 338,5; 29 348,0; 7 357,5; 18 369,0; 14 376,0; 23 381,0; 19 389,5; 32 399,5; 21 409,0; 25 417,0; 39 426,0; 36 437,0; 4 445,0; 8 454,5; 34 467,500000] L2=[28 279,0; 6 292,0; 9 305,0; 13 317,5; 33 328,5; 22 338,5; 10 347,5; 16 355,0; 35 362,5; 3 370,0; 31 375,5; 15 382,0; 2 389,0; 40 397,0; 38 407,0; 12 416,5; 17 426,0; 30 438,0; 11 452,0; 5 466,000000]
DRLP_07	40	21	545099,0	L1=[38 399,0; 17 409,5; 24 425,0; 19 439,5; 33 452,5; 1 465,5; 23 472,5; 26 481,0; 3 489,0; 9 492,0; 34 496,5; 13 503,5; 6 514,0; 14 528,5; 15 547,0; 35 559,5; 32 566,5; 2 573,5; 21 584,0; 10 597,5; 11 611,000000] L2=[36 400,5; 7 419,0; 8 433,0; 37 441,0; 22 452,0; 4 466,5; 39 479,0; 29 495,0; 40 508,5; 5 514,0; 25 517,0; 27 527,0; 28 545,0; 12 556,0; 18 562,0; 16 573,5; 20 586,5; 30 601,5; 31 620,500000]
AKV60_01	60	31	739199,0	L1=[1 155,5; 39 212,5; 55 266,0; 58 302,5; 30 344,0; 52 390,5; 10 430,0; 37 476,0; 45 512,0; 23 531,5; 5 559,0; 29 582,0; 41 589,5; 22 599,5; 42 611,0; 2 619,5; 26 631,0; 4 646,5; 13 655,5; 16 662,0; 36 679,0; 9 702,0; 57 739,0; 7 778,5; 46 812,5; 25 853,0; 40 886,0; 21 919,5; 24 966,0; 48 1015,0; 11 1066,500000] L2=[34 149,5; 18 196,0; 20 249,0; 60 301,0; 19 343,0; 59 389,5; 53 435,0; 50 456,5; 43 474,0; 31 511,0; 38 553,5; 49 590,5; 14 608,5; 17 614,5; 33 624,5; 8 630,0; 32 633,0; 35 650,0; 44 682,5; 47 723,0; 12 755,5; 54 778,5; 51 813,0; 6 848,0; 3 885,5; 15 939,0; 28 979,0; 56 1018,0; 27 1073,000000]
Murray324	50	27	4783908,5	L1=[27 43,76503; 34 63,295029; 39 82,745026; 20 99,580025; 21 113,235023; 46 126,080025; 45 138,165024; 25 151,27002; 48 163,050018; 47 169,350021; 32 174,055023; 4 179,535019; 50 184,265015; 3 187,87001; 7 191,365005; 36 194,345001; 11 198,714996; 12 204,175003; 28 207,99501; 24 215,490005; 9 226,375; 1 238,085007; 15 253,350006; 33 269,035004; 6 285,924988; 35 303,929993; 23 322,315002] L2=[29 49,875; 17 69,485001; 2 87,895004; 13 104,660004; 49 119,800003; 18 134,790009; 42 148,585007; 37 159,49501; 14 167,01001; 16 173,585007; 22 180,520004; 41 186,800003; 10 193,555008; 30 200,695007; 40 206,850006; 8 213,930008; 43 224,280014; 31 237,205017; 26 251,12001; 44 267,550018; 19 285,830017; 5 303,735016; 38 322,970032]
Murray327	50	26	4232807,5	L1=[12 43,895004; 33 64,335007; 21 83,960007; 25 101,475006; 32 117,550003; 28 131,160004; 3 142,940002; 19 152,330002; 16 160,865005; 39 169,12001; 40 173,560013; 15 175,76001; 44 179,555008; 18 184,190002; 11 187,175003; 22 189,130005; 2 194,059998; 29 200,319992; 43 206,579987; 6 215,679993; 24 224,784988; 27 235,134995; 35 249,949997; 37 267,48999; 1 285,869995; 5 304,924988] L2=[13 51,445004; 45 69,795006; 36 85,085007; 20 101,615005; 38 119,230003; 31 134,410004; 26 145,930008; 47 154,080002; 42 162,770004; 30 169,88501; 14 176,395004; 8 181,100006; 41 183,785004; 9 187,470001; 10 189,960007; 17 193,655014; 4 201,060013; 50 209,045013; 49 218,535019; 23 232,325012; 34 248,505005; 48 266,350006; 46 283,640015; 7 302,085022]

Referências

[1] A. Abraham, H. Guo and H. Liu. *Swarm Intelligence: Foundations, Perspectives and Applications*. Springer Berlin Heidelberg, Berlin, Heidelberg, v. 26, p. 3-25, 2006. doi: https://doi.org/10.1007/978-3-540-33869-7_1

[2] A. R. S. Amaral. *On the exact solution of a facility layout problem*. European Journal of Operational Research v. 173, n. 2, p. 508-518, 2006. doi: https://doi.org/10.1016/j.ejor.2004.12.021

[3] A. R. S. Amaral. *An exact approach to the one-dimensional facility layout problem*. Operations Research, 56(4), 1026–1033, 2008. https://doi.org/10.1287/opre.1080.0548

[4] A. R. S. Amaral. *The corridor allocation problem*. Computers & Operations Research, v. 39, n. 12, p. 3325-3330, 2012. doi: http://dx.doi.org/10.1016/j.cor.2012.04.016

[5] A. R. S. Amaral. *A parallel ordering problem in facilities layout*. Computers & Operations Research, v. 40, n. 12, p. 2930-2939, 2013a. doi: https://doi.org/10.1016/j.cor.2013.07.003

[6] A. R. S. Amaral. *Optimal solutions for the double row layout problem*. Optimization Letters, v. 7, n. 2, p. 407-413. 2013b. doi: http://dx.doi.org/10.1007/s11590-011-0426-8

[7] A. R. S. Amaral. *A mixed-integer programming formulation for the double row layout of machines in manufacturing systems*. International Journal of Production Research, v. 57, n. 1, p. 34-47, 2018. doi: https://doi.org/10.1080/00207543.2018.1457811

[8] A. R. S. Amaral. *A heuristic approach for the double row layout problem*. Annals of Operations Research, 2020. doi: https://doi.org/10.1007/s10479-020-03617-5

[9] M. F. Anjos, A. Kennings and A. Vannelli. *A semidefinite Optimization approach for the single-row layout problem with unequal dimensions*. Discrete Optimization, v. 2, n. 2, p. 113-122, 2005. doi: https://doi.org/10.1016/j.disopt.2005.03.001

[10] M. F. Anjos and A. Vannelli. *Computing globally optimal solutions for single-row layout problems using semidefinite programming and cutting planes*. INFORMS Journal on Computing, v. 20, n. 4, p. 611-617, 2008. doi: https://doi.org/10.1287/ijoc.1080.0270

[11] M. F. Anjos and M. V. Vieira. *Mathematical Optimization approaches for facility layout problems: The state-of-the-art and future research directions*. European Journal of Operational Research, v. 261, n. 1, p. 1-16, 2017. doi: https://doi.org/10.1016/j.ejor.2017.01.049

[12] M. F. Anjos and G. Yen. *Provably near-optimal solutions for very large single-row facility layout problems*. Optimization Methods and Software, v. 24, n. 4, p. 805-817, 2009. doi: http://dx.doi.org/10.1080/10556780902917735

[13] M. R. Bonyadi and Z. Michalewicz. Particle Swarm Optimization for Single Objective Continuous Space Problems: A Review. In *Evolutionary Computation*, v. 25, n. 1, p. 1-54, 2017. doi: https://doi.org/10.1162/EVCO_r_00180

[14] P. A. Brunese and J. M. A. Tanchoco. *On Implied Within-building Constraints for Machine Layout*. International Journal of Production Research, v. 51, n. 6, p. 1937-1952, 2013. doi: https://doi.org/10.1080/00207543.2012.721571

[15] B. Cellin. *Métodos para Resolução eficiente de Problemas de Layout*. Dissertação (Dissertação em informática) - Universidade Federal do Espirito Santo - UFES, Vitória, 2017

[16] J. Chae and A. C. Regan. *A mixed integer programming model for a double row layout problem*. Computers & Industrial Engineering 140, 106-244, 2020. doi: https://doi.org/10.1016/j.cie.2019.106244

[17] J. Chung and J. M. A. Tanchoco. *The double row layout problem, International Journal of Production Research*, v. 48, n. 3, p. 709-727, 2010. Disponível em http://webbuild.knu.ac.kr/~chung/research/DRLP.pdf

[18] G. L. Cravo and A. R. S. Amaral. *A GRASP algorithm for solving large-scale single row facility layout problems*. Computers and Operations Research, v. 106, p. 49-61, 2019. doi: https://doi.org/10.1016/j.cor.2019.02.009

[19] A. Drira, H. Pierreval and S. Hajri-Gabouj. *Facility layout problems: A survey*. Annual Reviews in Control, v. 31, n. 2, 255-267, 2007. doi: https://doi.org/10.1016/j.arcontrol.2007.04.001

[20] R. C. Eberhart and Y. Shi. Comparing inertia weights and constriction factors in particle swarm optimization, In *Proceedings of the 2000 Congress on Evolutionary Computation (CEC00 Cat. No.00TH8512)*, La Jolla, CA, USA, 2000, p. 84-88. doi: https://doi.org/10.1109/CEC.2000.870279

[21] R. C. Eberhart and Y. Shi. Particle swarm optimization: developments, applications and resources, In *Proceedings of the 2001 Congress on Evolutionary Computation (IEEE Cat. No.01TH8546)*, Seoul, South Korea, 2001, p. 81-86. doi: https://doi.org/10.1109/CEC.2001.934374

[22] M. Gen, K. Ida and C. Cheng. *Multirow machine layout problem in fuzzy environment using genetic algorithms*. Computers & Industrial Engineering, v. 29, n 1-4, p. 519-523, 1995. doi: https://doi.org/10.1016/0360-8352(95)00127-M

[23] J. Guan, G. Lin, H.-B. Feng and Z.-Q. Ruan. *A decompositionbased algorithm for the double row layout problem*. Applied Mathematical Modelling 77, 963-979, 2020. doi: https://doi.org/10.1016/j.apm.2019.08.015 2

[24] M. M. D. Hassan. *Machine layout problem in modern manufacturing facilities*. International Journal of Production Research, v. 32, n. 11, p. 2559-2584, 1994. doi: http://dx.doi.org/10.1080/00207549408957084

[25] S. S. Heragu and A. Kusiak. *Machine layout problem in flexible manufacturing systems*. Operations Research, v. 36, n. 2, p. 258-268, 1988. doi: https://doi.org/10.1287/opre.36.2.258

[26] S. S. Heragu and A. Kusiak. *Efficient models for the facility layout problem*. European Journal of Operational Research, v. 53, n. 1, p. 1-13, 1991. doi: https://doi.org/10.1016/0377-2217(91)90088-D

[27] J. H. Holland. *Adaptation in Natural and Artificial Systems*; University of Michigan Press: Editora Ann Arbor, 1975.

[28] H. Hosseini-Nasab, S. Fereidouni, S.M.T. Fatemi Ghomi et al. *Classification of facility layout problems: a review study*. The International Journal of Advanced Manufacturing Technology, v. 94, n. 1-4, p. 957–977, 2018. doi: https://doi.org/10.1007/s00170-017-0895-8

[29] P. Hungerländer and M. F. Anjos. *A semidefinite optimization approach to space-free multi-row facility layout*. Group for Research in Decision Analysis, 2012. Disponível em: https://www.gerad.ca/en/papers/G-2012-03/view

[30] P. Hungerländer and M. F. Anjos. *A semidefinite optimization-based approach for global optimization of multi-row facility layout*. European Journal of Operational Research, v. 245, n. 1, p. 46 - 61, 2015. doi: https://doi.org/10.1016/j.ejor.2015.02.049

[31] R. Kothari and D. Ghosh. *A scatter search algorithm for the single row facility layout problem*. Journal of Heuristics, v. 20, n. 2, p. 125-142, 2014. doi: https://doi.org/10.1007/s10732-013-9234-x

[32] V. Kothari, J. Anuradha, S. Shah and P. Mittal. A Survey on Particle Swarm Optimization in Feature Selection. In: *Krishna P.V., Babu M.R., Ariwa E. (eds) Global Trends in Information Systems and Software Applications. ObCom 2011*. Communications in Computer and Information Science, v. 270. Springer, Berlin, Heidelberg, 2012. doi: https://doi.org/10.1007/978-3-642-29216-3_22

[33] M. Mohammadi and K. Forghani. *Designing cellular manufacturing systems considering S-shaped layout*. Computers & Industrial Engineering, 98, 221–236, 2016. doi: https://doi.org/10.1016/j.cie.2016.05.041

[34] C. C. Murray, A. E. Smith and Z. Zhang. *An efficient local search heuristic for the double row layout problem with asymmetric material flow*. International Journal of Production Research, v. 51, n. 20, p. 6129-6139, 2013. doi: https://doi.org/10.1080/00207543.2013.803168

[35] G. Palubeckis. *Fast local search for single row facility layout*. European Journal of Operational Research, v. 246, n. 3, p. 800-814, 2015. doi: https://doi.org/10.1016/j.ejor.2015.05.055

[36] R. M. Permanhane. *Problemas de Layout*. Dissertação (Dissertação em informática) - Universidade Federal do Espírito Santo - UFES, Vitória, 2016.

[37] L. D. Secchin and A. R. S. Amaral. *An improved mixed-integer programming model for the double row layout of facilities*. Optim Lett (2018), v. 13, n. 1, p. 193-199, 2018. doi: https://doi.org/10.1007/s11590-018-1263-9

[38] S. Sengupta, S. Basak and R.A. Peters II. *Particle Swarm Optimization: A Survey of Historical and Recent Developments with Hybridization Perspectives*. Mach. Learn. Knowl. Extr. 2019, v. 1, n. 1, p. 157-191, 2019. doi: https://doi.org/10.3390/make1010010

[39] Y. Shi and R. C. Eberhart. Parameter selection in particle swarm optimization, In *Proceedings of the 7th Annual Conference on Evolutionary Programming*, New York, USA, 1998, p. 591-600, 1998. Disponível em: http://dl.acm.org/citation.cfm?id=647902.738978

[40] D. M. Simmons. *One-dimensional space allocation: An ordering algorithm*. Operations Research, v. 17, n. 5, p. 812-826, 1969. doi: https://doi.org/10.1287/opre.17.5.812

[41] R. Storn and K. Price. *Differential Evolution – A simple and efficient heuristic for global optimization over continuous spaces*. J. Glob. Optim. 1997, v. 11, n. 4, p. 341-359, 1997. doi: https://doi.org/10.1023/A:1008202821328

[42] P. N. Suganthan. Particle swarm optimiser with neighborhood operator, In *Proceedings of the IEEE Congress on Evolutionary Computation*, IEEE Press, Piscataway, USA, 1999, p. 1958-1962, 1999. doi: https://doi.org/10.1109/CEC.1999.785514

[43] A. Tubaileh and J. Siam. *Single and multi-row layout design for flexible manufacturing systems*. International Journal of Computer Integrated Manufacturing, v. 30, n. 12, p. 1316-1330, doi: https://doi.org/10.1080/0951192X.2017.1314013

[44] X. Yang, W. Cheng,A. E. Smith and A. R. S. Amaral. *An improved model for the parallel row ordering problem*, Journal of the Operational Research Society, p. 1-14. doi: https://doi.org/10.1080/01605682.2018.1556570

[45] H. Yoshida, Y. Fukuyama, S. Takayama and Y. Nakanishi. A particle swarm optimization for reactive power and voltage control in electric power systems considering voltage security assessment, In *IEEE SMC'99 Conference Proceedings*. 1999 IEEE International Conference on Systems, Man, and Cybernetics (Cat. No.99CH37028), Tokyo, Japan, 1999, p. 497-502 v.6. doi: https://doi.org/10.1109/ICSMC.1999.816602

[46] Z. Zhang and C. Murray. *A Corrected Formulation for the Double Row Layout Problem*. International Journal of Production Research, v. 50, n. 15, p. 4220-4223, 2012. doi: https://doi.org/10.1080/00207543.2011.603371