RESOLVENDO OS PROBLEMAS DE LEIAUTE DE FACILIDADES ESTÁTICO E DINÂMICO COM ÁREAS DIFERENTES USANDO ALGORITMO GENÉTICO

Frederico Galaxe Paes
Instituto Federal Fluminense – IFF
Av. Souza Mota, 350, Pq. Fundão, 28060-010, Campos dos Goytacazes, RJ
fpaes@iff.edu.br

Artur Alves Pessoa
Departamento de Engenharia de Produção – Universidade Federal Fluminense
Rua Passo da Pátria 156, São Domingos, 24210-240, Niterói, RJ
artur@producao.uff.br

RESUMO

Este artigo apresenta dois algoritmos genéticos que resolvem os problemas de leiaute de facilidades estático (PLFE) e dinâmico (PLFD), onde um conjunto de facilidades retangulares com áreas diferentes devem ser localizadas, sem sobreposição, em um espaço limitado. Dadas as dimensões das facilidades e o fluxo entre elas, o objetivo do PLFE é localizá-las no plano de modo a minimizar o custo de manuseio de material. Adicionando-se um custo que penaliza a realocação das facilidades de um período para o outro ao objetivo anterior, obtém-se o objetivo do PLFD. O algoritmo destinado a resolver o PLFE, utiliza também uma estratégia de deslocamento dos limites do espaço disponível sempre que possível, para permitir a inserção de uma determinada facilidade. Diversos testes computacionais foram conduzidos com instâncias da literatura para os dois problemas e os resultados comparados com as melhores abordagens da literatura. Como resultado, observou-se um desempenho superior dos algoritmos propostos para praticamente todas as instâncias, tanto na qualidade das soluções como no tempo de execução, com relação aos demais algoritmos.

PALAVRAS CHAVE. ***L*eiaute de Facilidades, Algoritmo Genético, Metaheurística.**

Tópicos: (MH–Metaheurísticas, OC–Otimização Combinatória)

ABSTRACT

This paper presents two genetic algorithms to solve the static facility layout problem (SFLP) and dynamic facility layout problem (DFLP), where a set of rectangular facilities with unequal-areas should to be placed, without overleap, in a limited plant floor. Given the facility dimensions and the flows among them, the objective of the SFLP is to place them in a plant floor in order to minimize the sum of the material handling costs. Adding a cost that penalize the rearrangement of the facilities of one period to other, to the previews objective, we have the DFLP objective. The algorithm proposed to solve the SFLP, also uses a strategy of displacement of the plant floor frontier whenever possible, to allow the insertion of one given facility. Several tests were conducted with instances from the literature for the two problems and the results were compared with the better approaches from the literature. As a result, the proposed approaches perform especially well for practically all instances, both in quality of the solutions and in CPU time, regarding other algorithms.

KEYWORDS. Facility Layout, Genetic Algorithm, Metaheuristic.

Paper topics: (MH–Metaheuristics, OC–Combinatorial Optimization)

1. Introdução

O problema de leiaute de facilidades estático (PLFE) é um conhecido problema da literatura, cujo objetivo é encontrar as posições dos departamentos no chão de fábrica (ou espaço disponível) sem que haja sobreposição, enquanto algum objetivo é otimizado, normalmente a minimização do custo de manuseio de material [Kusiak e Heragu, 1987; Drira et al., 2007]. No mercado atual, com o aumento da competição global e o curto ciclo de vida de produção, o fluxo de material entre as facilidades muda durante o horizonte de planejamento originando o problema de leiaute de facilidade dinâmico (PLFD). Neste caso, o horizonte de planejamento é dividido em um número de períodos, que podem ser anos, estações, meses ou semanas. Do mesmo modo que no PLFE, no PLFD as facilidades podem ter áreas iguais ou diferentes.

Assim, o PLFD consiste em encontrar a posição das facilidades dentro do chão de fábrica para múltiplos períodos, tal que não haja sobreposição entre as facilidades e a soma dos custos de manuseio de material e realocação de facilidades seja minimizada. O custo de realocação ocorre quando o centroide ou a orientação de uma facilidade muda em períodos consecutivos. Em outras palavras, para cada período do horizonte de planejamento o leiaute é determinado tal que a soma do custo de manuseio de material para cada leiaute e o custo de realocação dos departamentos entre períodos consecutivos sejam minimizados.

Rosenblatt [1986] foi o primeiro a estudar o PLFD com facilidades tendo áreas iguais, propondo uma heurística de otimização baseada em programação dinâmica (PD). Yang e Peters [1998] apresentou uma formulação e um procedimento heurístico baseado em um algoritmo de construção de leiaute para resolver o PLFD. Em Dunker et al. [2005] foi proposto um algoritmo híbrido combinando PD com um algoritmo genético (AG). Para cada estágio, o AG mantinha uma população de leiautes enquanto a PD fornecia a avaliação do *fitness* de cada leiaute (indivíduos da população). McKendall e Hakobyan [2010], através de uma modificação feita no algoritmo CBA [Imam e Mir, 1998], obteve uma heurística de construção chamada de *boundary search heuristic* (BSH). Após construir uma solução com o BSH, os autores utilizaram a heurística busca tabu para melhorar a solução e encontraram bons resultados para algumas instâncias do problema estático e dinâmico. Uma técnica híbrida robusta é proposta em Hosseini et al. [2014], a qual baseou-se em três heurísticas: *imperialist competitive algorithm* (ICA), *variable neighborhood search* (VNS) e simulated annealing (SA). Um algoritmo de otimização por dispersão de partículas foi proposto em Asl e Wong [2015] para resolver tanto o PLFE como o PLFD com áreas diferentes, bem como dois métodos de busca local para melhorar a qualidade das soluções.

Neste artigo, são abordados os problemas de leiaute estático (PLFE) e dinâmico (PLFD), com área disponível limitada e contínua, envolvendo facilidades rígidas com orientação livre (localização vertical ou horizontal) e com áreas diferentes. Tanto o PLFE como o PLFD são difíceis de serem resolvidos de forma exata, pois apresentam características similares ao problema quadrático de alocação (PQA) [Sahni e Gonzalez, 1976]. Portanto, métodos analíticos, heurísticas e metaheurísticas têm sido utilizados para resolver este problema. Deste modo, são propostos duas abordagens baseadas em algoritmo genético (GA) que utilizam um procedimento conhecido de avaliação para a construção gulosa da solução, chamado de espaço máximo vazio (EMV)(*Empty Maximal-Spaces*–EMS) [Paes et al., 2017], para resolver o PLFE e o PLFD. Além disso, no algoritmo destinado a resolver o PLFE, quando uma determinada facilidade não couber no EMV disponível, uma estratégia de deslocamento dos limites do espaço disponível é utilizada sempre que possível, no intuito de permitir sua inserção. Os algoritmos são então testados em instâncias de referência da literatura e comparado com as melhores abordagens disponíveis destinadas a resolver os problemas.

O restante do artigo é organizado como segue. A Seção 2 apresenta a definição do problema. A Seção 3 descreve como é gerada a população inicial do GA, bem como seus operadores de *crossover* e mutação e o pseudocódigo do algoritmo proposto. Na Seção 4 são comparados e discutidos os resultados computacionais dos algoritmos propostos com outros resultados relevantes da literatura e finalmente, na Seção 5 são apresentadas as conclusões.

2. Definição do Problema

Seja o problema de localizar para cada período $p = 1, \ldots, P$, onde P é o número de períodos, os centroides (x_{pi}, y_{pi}) de n facilidades retangulares com áreas diferentes em um plano limitado e contínuo de dimensões L (largura) e H (altura), sem sobreposição. Cada facilidade $i = 1, \ldots, n$ é definida por sua largura l_i, sua altura h_i, sua área $A_i = l_i \times h_i$ e pode ser localizada tanto na posição horizontal (maior lado paralelo ao eixo x), como na vertical (maior lado paralelo ao eixo y). Desta forma, para cada período p um leiaute fica determinado pelas coordenadas dos centroides e pelas dimensões (largura e altura) de cada facilidade i. A função custo a ser minimizada é:

$$Custo = \sum_{p=1}^{P} \sum_{i=1}^{n} \sum_{j=i+1}^{n} f_{pij}(|x_{pi} - x_{pj}| + |y_{pi} - y_{pj}|) + \sum_{p=1}^{P} \sum_{i=1}^{n} R_{pi} r_{pi} \quad (1)$$

Na equação (1), f_{ij} é o fluxo de material entre as facilidades i e j, R_{pi} é o custo de realocação de qualquer facilidade i no início de qualquer período p e r_{pi} é uma variável binária, onde $r_{pi} = 1$ se a facilidade i for realocada no período p ($x_{p-1,i} \neq x_{pi}$ ou $y_{p-1,i} \neq y_{pi}$ ou se a orientação (vertical/horizontal) mudar de um período para o outro). Um leiaute inicial/existente ($p = 0$) ainda é considerado. Assim, ao custo de manuseio de material do leiaute obtido para o primeiro período pode ser adicionado um custo de realocação, caso alguma facilidade seja realocada. Desconsiderando o índice p e o segundo termo da equação (1) (custo de realocação), obtém-se a função objetivo do PLFE. Formulações para ambos os problemas podem ser encontradas em Yang e Peters [1998], Dunker et al. [2005] e McKendall e Hakobyan [2010]. Um pequeno exemplo com 12 facilidades ($n = 12$) e três períodos ($p = 3$) é apresentado na Figura 1.

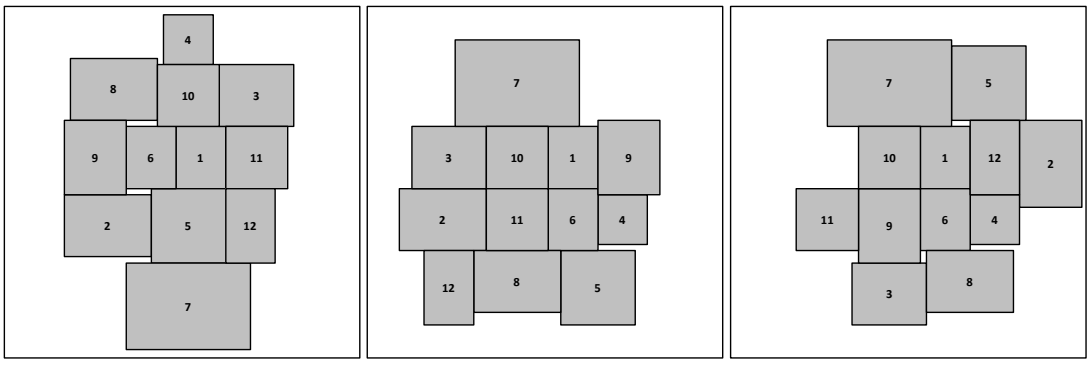

Figura 1: Exemplo de uma solução do PLFD com n=12 e p=3.

3. Algoritmo Genético para o PLFE/PLFD

Um algoritmo genético (GA) é uma conhecida metaheurística pertencente à classe dos algoritmos evolucionários (AE), que imita o processo de evolução natural usando técnicas tais como herança, mutação, seleção e *crossover* e é frequentemente usada para gerar

boas soluções em problemas de otimização. GAs trabalham com uma família de soluções, conhecida como a população atual $P(t)$, da qual se obtém a próxima geração de soluções $P(t+1)$. O GA apresentado neste artigo é similar ao GA proposto em Paes et al. [2017] no que se refere à geração da população inicial, ao *crossover* e à construção da solução para cada período p, diferenciando-se apenas pela inclusão do operador de mutação, como será visto na Seção 3.3.

3.1. Representação do Cromossomo e Geração da População Inicial

Um cromossomo do PLFD será representado por uma sequência de P (nº de períodos) permutações π pertencentes ao conjunto $\Pi(n)$ de todas as permutações de $N = \{1, \ldots, n\}$, onde n é o número de facilidades que serão localizadas no plano por meio de um procedimento construtivo. Deste modo, um cromossomo será representado por um vetor do tipo $(\pi_1, \ldots, \pi_p, \ldots, \pi_P)$, onde $\pi_p = (\pi_p(1), \pi_p(2), \ldots, \pi_p(n))$ representa a parte do cromossomo referente ao período p, com $\pi_p(i)$ sendo a i-ésima facilidade a ser inserida ($i = 1, \ldots, n$ e $p = 1, \ldots, P$). Para representar um cromossomo do PLFE, basta considerar apenas um único período.

$$Cromossomo = (\underbrace{(\pi_1(1), \ldots, \pi_1(n))}_{p=1}, \ldots, \underbrace{(\pi_k(1), \ldots, \pi_k(n))}_{p=k}, \ldots, \underbrace{(\pi_P(1), \ldots, \pi_P(n))}_{p=P}) \quad (2)$$

Os indivíduos da população inicial são gerados a partir de uma ordenação prévia das n facilidades (para cada período p), em ordem crescente do valor do quociente A_i/FT_i, onde A_i é a área da facilidade i e FT_i é a soma dos fluxos entre a facilidade i e as demais facilidades $j = 1, \ldots, n$, com $i \neq j$. Em seguida, os indivíduos são perturbados de modo a gerar uma certa aleatoriedade. O processo de geração da população inicial usa uma lista de candidatos restrita (LCR) [Feo e Resende, 1995], cujo tamanho é ajustado por um parâmetro α. Iterativamente, para $k = 1, \ldots, n$, o $k - ésimo$ elemento da sequência será aleatoriamente selecionado da LCR, considerando uma distribuição uniforme. Para cada iteração k, sejam $\mathcal{S}_k$ o conjunto das facilidades que já foram inseridas e $\bar{\mathcal{S}}_k$ o conjunto das facilidades que ainda não foram inseridas. A LCR é inicializada e atualizada para conter as $\ell = \min\{\alpha, |\bar{\mathcal{S}}_k|\}$ facilidades com menor A_i/FT_i em $\bar{\mathcal{S}}_k$ [Paes et al., 2017].

A Figura 2 ilustra a geração de um cromossomo da população inicial com $\alpha = 5$. Uma sequência com facilidades ordenadas em ordem crescente de A_i/FT_i para cada período p, aparece no topo da figura seguida pela primeira LCR. Então, a partir do primeiro período ($p = 1$), a facilidade 4 é aleatoriamente selecionada e inserida na primeira posição. Em seguida, a facilidade 5 é aleatoriamente selecionada e inserida na segunda posição. O cromossomo final gerado aparece no final da figura.

3.2. Seleção e *Crossover*

Nos algoritmos propostos, a seleção dos pais para recombinação é feita de maneira totalmente aleatória com probabilidade uniforme. Desta forma, permite-se que todos os indivíduos, independente de sua aptidão, tenham as mesmas chances de serem selecionados.

Quanto ao operador de *crossover* utilizado, sempre que dois pais forem selecionados serão submetidos ao processo de recombinação para gerar um novo filho. Em função da representação dos cromossomos, optou-se pelo *Position-based Crossover* (PX) [Syswerda e Palmucci, 1991], um operador que procura preserva a sequência dos genes no filho. O modo como o operador PX atua é descrito no pseudo-código a seguir:

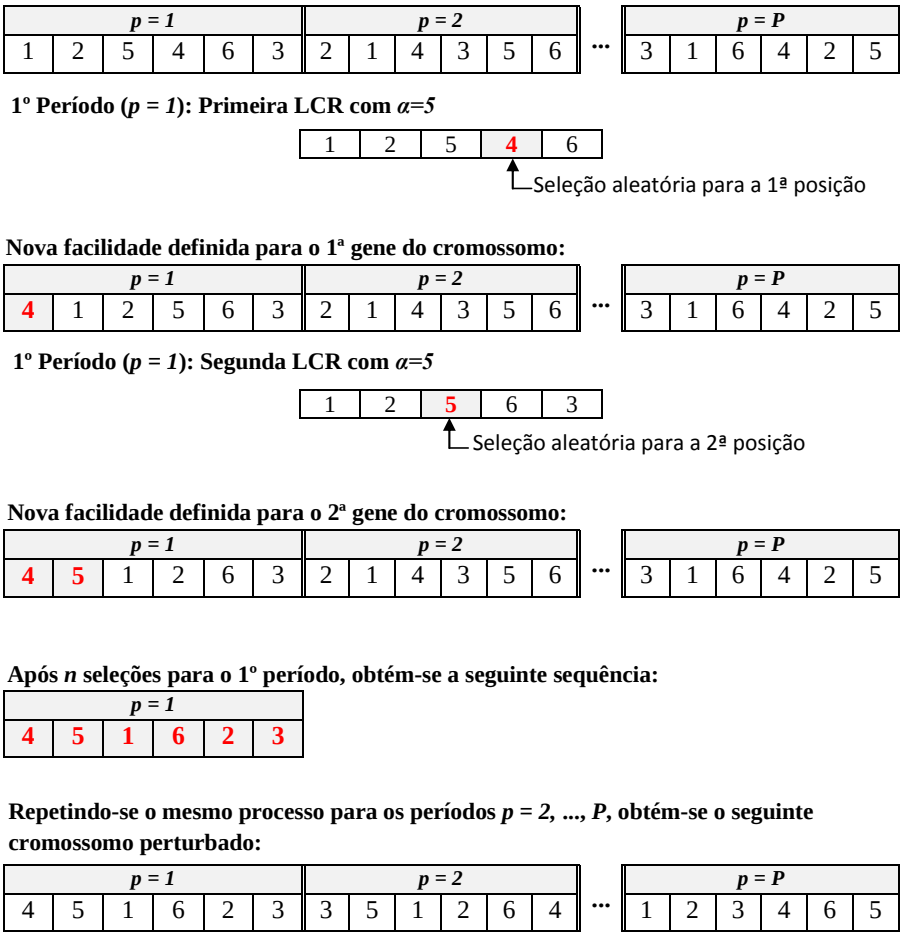

Figura 2: Exemplo de um cromossomo sendo perturbado com $\alpha = 5$.

Passo 1: Escolha aleatoriamente um pai atribuindo a mesma probabilidade a cada um dos dois indivíduos que serão recombinados;

Passo 2: Para cada período $p = 1, \ldots, P$ faça:

Passo 2.1: Selecione N_p^{herd} genes deste pai aleatoriamente, onde $N_p^{herd} = \frac{n}{2} + \frac{N_p^{dif}}{4}$, com n representando o número de genes e N_p^{dif} o número de facilidades diferentes nos dois pais referentes ao mesmo período p;

Passo 2.2: Copie o conteúdo destes genes para os genes correspondentes no filho ;

Passo 2.3: Remova os genes que foram selecionados no **Passo 2.1**, no segundo pai. A sequência resultante contém os genes que o filho precisa;

Passo 2.4: Copie o conteúdo na mesma ordem da sequência resultante, da esquerda para a direita, para as posições vazias no filho.

Esta estratégia baseia-se na observação de que no início das gerações, quando os indivíduos ainda são muito diferentes, gerar um filho que possua 75% do pai selecionado ajuda a evitar um tempo excessivo até a convergência. Por outro lado, quando a população começar a convergir e os indivíduos ficarem muito parecidos, o valor de N_p^{herd} se aproximará de 50% de n e, desta forma, selecionando-se 50% de cada pai evita-se a geração de muitos filhos repetidos [Paes et al., 2017].

3.3. Operador de Mutação

A mutação será aplicada a todos os indivíduos sempre que a população convergir, isto é, quando uma das três condições seguintes ocorrer:

- o *gap* entre o custo médio e o custo mínimo de todos os indivíduos em $P(t)$ é menor que $0,05\%$;
- todos os indivíduos gerados na iteração atual são iguais a algum indivíduo em $P(t-1)$;
- $P(t) = P(t-1) = \ldots = P(t-50)$.

O operador de mutação escolhido baseia-se na ordem em que as facilidades de cada período aparecem no cromossomo. O operador funciona do seguinte modo: para cada período p, sorteiam-se aleatoriamente dois pontos de corte dentro do cromossomo, que delimitarão uma sub-lista. Em seguida, faz-se uma permutação aleatória dos elementos desta sub-lista. A Figura 3 mostra um exemplo de mutação em um cromossomo com 6 facilidades e P períodos.

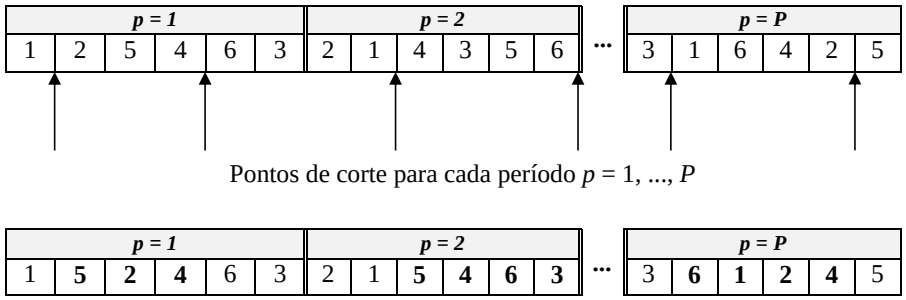

Figura 3: Exemplo do operador de mutação baseado em ordem.

3.4. Pseudocódigo do GA para o PLFE/PLFD

Os algoritmos propostos para resolver ambos os problemas, utilizam um procedimento construtivo guloso para localizar as facilidades uma de cada vez, segundo a ordem em que aparecem no cromossomo, em uma posição no plano que não gere sobreposição e que produza o menor custo parcial. Assim, a estratégia de construção baseia-se na seleção de um espaço máximo vazio EMV a partir de uma lista $\mathcal{L}$ de todos os EMV's disponíveis, tal que produza o menor custo parcial (mais detalhes em Paes et al. [2017] e Gonçalves e Resende [2015]). Basicamente, o pseudocódigo é o mesmo para ambos os problemas, diferenciando-se apenas quanto ao número de períodos P do caso dinâmico. Para o problema estático o algoritmo foi chamado de GAE e para o dinâmico de GAD, ambos recebendo como parâmetros o valor de α e o tamanho da população $nPop$.

Um laço principal (linha 3) controla o número de vezes que o GA será executado. Assim, após o GA convergir 3(três) vezes o mesmo é encerrado. Na linha 4 do Algoritmo 1, uma população inicial $P(0)$ de tamanho $nPop$ é gerada do mesmo modo como foi descrito na Seção 3.1. Para cada cromossomo gerado, uma solução é construída e avaliada na linha 6 utilizando o algoritmo de construção guloso, respeitando as dimensões L e H do espaço disponível. A referida solução terá um único leiaute, caso o algoritmo seja o GAE, e terá P leiautes, caso seja o GAD. O laço interno7–16 controla o número de gerações, sendo executado até a convergência da população. Para cada iteração deste laço, o algoritmo

Algoritmo 1 Algoritmo Genético para o PLFE/PLFD

```
1: Procedimento GAE/GAD(α, nPop)
2:     t ← 0 ;
3:     repita
4:         Gere a população P(t) com base no parâmetro α ;
5:         Construa e avalie cada indivíduo de P(t) usando o algoritmo de construção, conforme
6:         o problema tratado;
7:         enquanto (P(t) não convergiu) faça        {Laço principal. Controla o nº de gerações }
8:             t = t + 1 ;
9:             para i ← 1 até nPop faça        {Gera descendentes por crossover}
10:                Selecione dois pais p1, p2 ∈ P(t − 1) aleatoriamente;
11:                d ← Crossover(p1, p2) ;
12:                Construa e avalie o indivíduo d usando o algoritmo de construção, conforme
13:                o problema tratado;
14:            fim para
15:            Selecione indivíduos sobreviventes em P(t) ;
16:        fim enquanto
17:        para i ← 1 até nPop faça        {Aplica mutação em todos os indivíduos i}
18:            se i ≠ melhor então
19:                Mutação(i) ;
20:            fim se
21:        fim para
22:    até que critéro de parada = TRUE
23: Retorne a melhor solução S* encontrada;
24: fim Procedimento
```

seleciona aleatoriamente dois pais da população $P(t-1)$ (linha 10) para então ser realizado o *crossover* (linha 11), como mostrado na Seção 3.2. Novamente, uma solução é construída e avaliada para o indivíduo d utilizando o algoritmo guloso (linha 13). Deste modo, $nPop$ filhos gerados serão inseridos na população. Após o laço 9–14 ser executado, a população contará com $2 \times nPop$ indivíduos e então, $nPop$ indivíduos sobreviventes deverão ser selecionados para compor a próxima geração $P(t)$ (linha 15). Após a convergência do GA, todos os indivíduos, exceto o indivíduo com o menor custo (melhor), sofrerão mutação conforme descrito na Seção 3.3.

O algoritmo de construção utilizado para resolver o PLFE difere do algoritmo usado para resolver o PLFD em um aspecto. Sempre que uma nova facilidade i é selecionada do cromossomo para ser inserida, ela é testada nos EMV's de modo a encontrar o ponto $(X_i^{mín}, Y_i^{mín})$ mais próximo do ótimo irrestrito (OI) [Heragu, 1997] e que produza a menor soma parcial dos fluxos, ponderada pelas distâncias retilíneas entre os centroides da facilidade inserida e das facilidades já alocadas (primeira termo da equação (1)). A diferença está em considerar o deslocamento da fronteira do espaço disponível em quatro direções possíveis: *direita*, *esquerda*, *para cima* e *para baixo*. Assim, se não for possível inserir a facilidade i em um determinado EMV, tenta-se deslocar o espaço disponível em uma das quatro direções de modo que a facilidade se ajuste ao EMV com uma determinada orientação. Em seguida, se o ponto $(X_i^{mín}, Y_i^{mín})$ pertencer a um EMV expandido, os novos limites do espaço disponível, bem como a lista $\mathcal{L}$ de EMV's são atualizados, conforme mostrado na Figura 4.

Quanto ao algoritmo de construção destinado a resolver o PLFD, além de construir um leiaute para cada período p, ele considera o custo de realocação que será adicionado ao custo da solução sempre que $x_{p-1,i} \neq x_{pi}$ ou $y_{p-1,i} \neq y_{pi}$ ou se a orientação (vertical/horizontal) mudar de um período para o outro. Deste modo, ao inserir uma facilidade no local selecionado em um período p, deve-se verificar as coordenadas e a orientação da

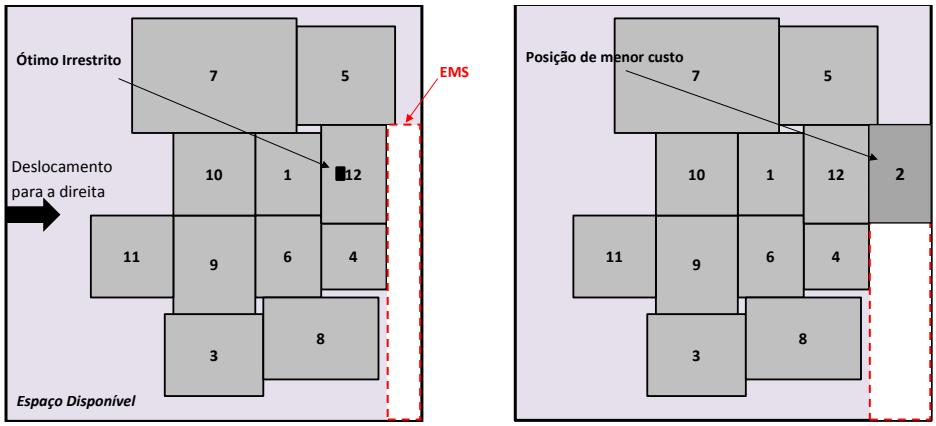

Figura 4: Construção de uma solução para o PLFE com deslocamento do espaço disponível.

mesma facilidade no período $p-1$. Se houver mudança, deve-se avaliar se o custo parcial da nova posição/orientação adicionado ao custo de realocação, é melhor que o custo parcial considerando o local do período $p-1$, caso este ainda esteja disponível.

4. Experimentos Computacionais

Para avaliar a performance das abordagens propostas para resolver tanto o PLFE, como o PLFD, foram realizados alguns experimentos computacionais. Os algoritmos foram codificados em C++ e os experimentos conduzidos em uma máquina Intel Core $i7-3517U$, com 1,9 GHz de CPU e 6 GB de memória RAM usando o sistema operacional Windows 8. As instâncias testadas do PLFD são compostas por 6 facilidades e 6 períodos (P6) e 12 facilidades e 4 períodos (P12), que devem ser localizadas em um espaço disponível com dimensões de 30×30 e 50×50, respectivamente. Observou-se que o custo de realocação utilizado para uma mesma instância não é o mesmo em todas as abordagens da literatura. Deste modo, o GAD foi comparado com os algoritmos: HGA [Dunker et al., 2005], TS/BSH [McKendall e Hakobyan, 2010] e PSO [Asl e Wong, 2015], conforme mostrado na Tabela 1, os quais utilizaram custos de realocação de 19 e 50 unidades para as instâncias P6 e P12, respectivamente. O algoritmo HC [Yang e Peters, 1998] usou um custo de realocação de 100 unidades para as duas instâncias. Deste modo, o GAD foi comprado separadamente com o HC, usando o mesmo valor de R_{pi} (Tabela 3). Como na literatura só existem duas instâncias do PLFD, três outras instâncias do caso estático (PLFE) também foram testadas. Tais instâncias são compostas por 8 facilidades (P8), 11 facilidades (P11) e 20 facilidades (P20), que devem ser localizadas em um espaço disponível com dimensões de 12×12, 15×15 e 14×14, respectivamente. Assim, o GAE foi comparadas com os seguintes algoritmos da literatura: TOPOPT [Imam e Mir, 1989], FLOAT [Imam e Mir, 1993], HOT [Mir e Imam, 2001] e PSO [Asl e Wong, 2015], conforme mostrado na Tabela 4. Todas as instâncias testadas, para ambos os casos, foram obtidas de Asl e Wong [2015].

Os parâmetros utilizados no algoritmo genético foram fixados em $nPop$=1000 (tamanho da população) e α=5 (população inicial) para ambos os problemas. A seguir, serão apresentados os resultados comparativos entre as abordagens baseadas em GA e os principais algoritmos da literatura.

4.1. Avaliação dos Resultados

Os resultados dos testes computacionais realizados são apresentados nas Tabelas 1, 3 e 4. A Tabela 1 compara os resultados do GAD com o HGA, TS/BSH e PSO através dos valores dos custos mínimos C(mín) e custos médios C(méd), obtidos em 10 rodadas do

algoritmo GAD. A tabela também apresenta o tempo de processamento da melhor solução obtida T(s), o custo de realocação total (CRT), bem como os *Gaps* entre o C(mín) do algoritmo proposto e os demais algoritmos (Equação 3). Em McKendall e Hakobyan [2010] não foi fornecido o C(mín) para o algoritmo TS/BSH, assim, o *Gap* foi calculado com relação ao C(méd). Na Equação 3, $C(mín)_{alg}$ é o custo mínimo obtido pelo algoritmo comparado para as instâncias da literatura e $C(mín)_{GAD/GAE}$ é o custo mínimo obtido por um dos dois algoritmos propostos.

$$Gap = \frac{(C(mín)_{alg} - C(mín)_{GAD/GAE})}{C(mín)_{alg}} \times 100 \tag{3}$$

Heur./Inst.	P6					P12				
	C(mín)	CRT	C(méd)	T(s)	Gap(%)	C(mín)	CRT	C(méd)	T(s)	Gap(%)
HGA	**6507,50**		6569	504	-0,39	29098,50		27748	3240	+9,52
TS/BSH	-	-	6648,30	1666,20[a]	+1,71[b]	-	-	26845,50	4920[a]	+1,57[b]
PSO	6668,75	95	6883,18	2354,58	+2,03	27626,67	1200	29752,83	11105,56	+4,70
GAD	6533,50	285	**6534,20**	**17,66**	-	**26328,50**	1750	**26422,95**	**70,67**	-

[a] Tempo médio de execução do algoritmo;
[b] Desvio com relação ao custo médio C(méd).

Tabela 1: Comparação do GAD com os melhores algoritmos da literatura.

De acordo com a Tabela 1, para a instância $P6$ o GAD apresentou um C(mín) = 6533, 50, o qual é composto por um custo de manuseio de material de 6248,50 e um custo de realocação total de 285, sendo superado apenas pelo HGA que apresentou um C(mín) = 6507, 50. Por outro lado, o C(méd) do GAD superou todas os demais algoritmos. Além disso, o tempo necessário para obter a melhor solução do GAD foi bem inferior aos demais algoritmos. Com relação à instância P12 o GAD superou todas os demais algoritmos com um C(mín) = 26328, 50, o qual é composto por um custo de manuseio de material de 24578,50 e um custo de realocação total de 1750, representando uma redução no custo mínimo (*Gap*) de até 9, 52%. Além disso, o tempo de processamento do GAD para estas instâncias foi inferior aos dos demais algoritmos. Cabe ressaltar, que o HGA é uma heurística baseada em uma formulação de programação linear inteira mista (PLIM) sendo, portanto, um método exato. Dunker et al. [2005] usou uma formulação relaxada, na qual as únicas variáveis binárias são as variáveis usadas para orientação e realocação das facilidades. Segundo os autores, o número de variáveis binárias no problema relaxado aumenta linearmente com o tamanho da instância, o qual poderia resultar em um aumento exponencial do tempo de processamento. Por outro lado, o GAD é capaz de resolver instâncias maiores em um tempo computacional razoável. A Figura 5 mostra o leiaute obtido para cada período da melhor solução de P12. O leiaute inicial é dado na Tabela 2.

Fac.	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
x	6	11,5	17,5	22	11,5	22	11,5	23	17	17,5	17,5	6,5
y	16,5	27,5	16	18	22	27,5	14	22,5	22	32,5	27,5	22
l	4	7	5	4	6	4	7	7	5	5	5	4
h	5	5	6	4	6	5	10	5	6	5	5	6

Tabela 2: Centroides e dimensões das facilidades do leiaute inicial (p=0) de P12.

Na Tabela 3 o GAD é comparado com o algoritmo HC, considerando um custo de realocação de 100 para ambas as instâncias. De acordo com a tabela, o GAD apresenta

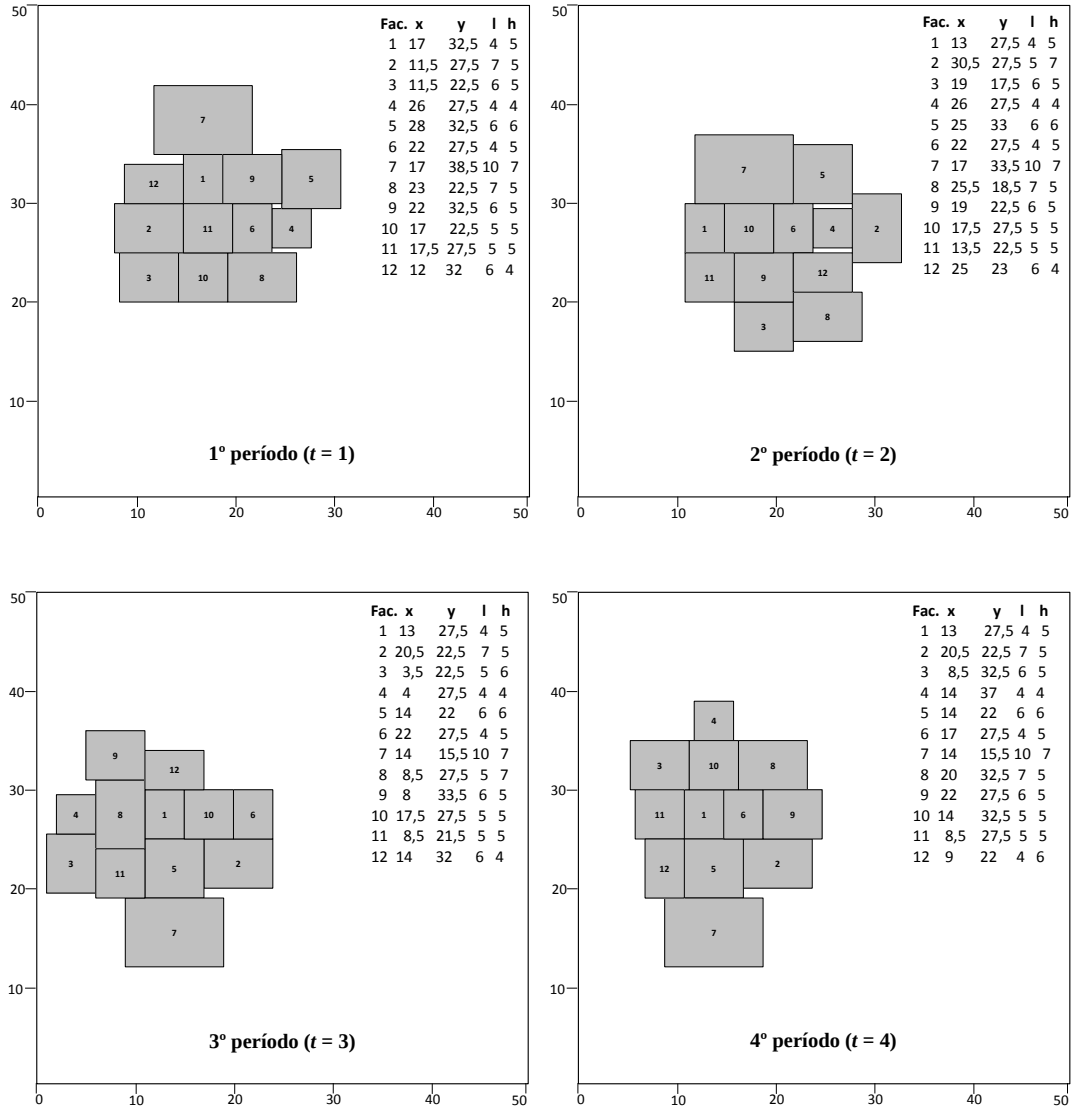

Figura 5: Melhor solução encontrada para o problema P12, com Custo=26328,5 e t=70,67seg.

Heur./Inst.	P6					P12				
	C(mín)	CRT	C(méd)	T(s)	Gap(%)	C(mín)	CRT	C(méd)	T(s)	Gap(%)
HC	7657	900	-	20	+8,99	30344	2300	-	2014	+9,43
GAD	**6968**	400	6968	**18,13**	-	**27481**	2000	27677,10	**272,51**	-

Tabela 3: Comparação do GAD com o algoritmo de Yang e Peters [1998].

C(mín) menores que o HC para ambas as instâncias com tempos de processamentos inferiores. De fato, o GAD apresenta reduções nos custos mínimos de 8,99% (P6) e 9,43% (P12).

Finalmente, a Tabela 4 apresenta os resultados para as instâncias do caso estático. Conforme descrito anteriormente, o algoritmo GAE utilizado para resolver estas instâncias considera apenas um único período. Este algoritmo também foi executado 10 vezes e comparado com as melhores abordagens da literatura. De acordo com a tabela, o GAE superou todas os demais algoritmos em todas as três instâncias com reduções nos custos mínimos que chegaram a 14,1% (TOPOPT – P20), apresentando um tempo de processamento bem inferior ao PSO. Este tempo reduzido de processamento, deve-se ao fato de o GAE utilizar

Heur./Inst.	P8				P11				P20			
	C(mín)	C(méd)	Gap(%)	T(s)	C(mín)	C(méd)	Gap(%)	T(s)	C(mín)	C(méd)	Gap(%)	T(s)
TOPOPT	-	-	-	-	-	-	-	-	1320,7	1395,6	+14,1	-
FLOAT	-	-	-	-	-	-	-	-	1264,9	1333,8	+10,2	-
HOT	-	-	-	-	-	-	-	-	1225,4	1287,3	+7,4	-
PSO	193,7	208,7	+1,2	220,7	1286,1	1335,6	+6,7	888,3	1206,6	1264,2	+5,9	2352,1
GAE	**191,5**	**192**	-	**0,7**	**1200,3**	**1200,3**	-	**4,9**	**1135**	**1135,1**	-	**21,4**

Tabela 4: Comparação do GAE com os melhores algoritmos da literatura.

um algoritmo de construção guloso eficiente, enquanto que o PSO depende de dois métodos de busca local para melhorar a qualidade das soluções e prevenir ótimos locais [Asl e Wong, 2015]. A mesma diferença de tempo pode ser observada entre os algoritmos GAD e PSO na Tabela 1, uma vez que o PSO para o caso dinâmico também faz uso de busca local. Não foi possível uma comparação de tempo com os outros algoritmos, porque os tempos não foram apresentados pelos mesmos.

5. Conclusões

Neste artigo são propostos dois algoritmos que, ao contrário das abordagens da literatura, utilizam um método guloso eficiente para a construção de uma solução tanto do PLFE, como do PLFD. O primeiro problema tem como objetivo minimizar a soma do custo de manuseio de material, enquanto o segundo busca minimizar a soma do custo de manuseio de material mais o custo de realocação de facilidades, ambos respeitando as restrições impostas. Estes algoritmos são então embutidos em um algoritmo genético GA, originando o algoritmo GAE para o caso estático e o GAD para o caso dinâmico. O método de construção do algoritmo GAE, conta ainda com uma estratégia adicional que permite o deslocamento da fronteira do espaço disponível para permitir a inserção de uma determinada facilidade, quando esta não for possível. Diversos testes computacionais foram conduzidos com instâncias da literatura, obtendo-se reduções nos custos mínimos acima de 9% para o problema dinâmico e acima de 14% para o problema estático, além de um menor esforço computacional se comparados com as melhores abordagens da literatura.

Como propostas de trabalhos futuros, sugere-se o desenvolvimento de um algoritmo mais geral para o PLFD onde sejam considerados os *aspect ratios* das facilidades (facilidade flexível), permitindo que elas se ajustem ao espaço disponível. Outra proposta interessante, seria gerar novas instâncias de médio e grande porte para avaliar melhor as abordagens propostas para o PFLD. Tal fato justifica-se pela importância do PLFD, corroborado pela quantidade de publicações na literatura internacional.

Referências

Asl, A. D. e Wong, K. Y. (2015). Solving unequal-area static and dynamic facility layout problems using modified particle swarm optimization. *Journal of Intelligent Manufacturing*, p. 1–20.

Drira, A., Pierreval, H., e Hajri-Gabouj, S. (2007). Facility layout problems: A survey. *Annual Reviews in Control*, 31(2):255–267.

Dunker, T., Radons, G., e Westkämper, E. (2005). Combining evolutionary computation and dynamic programming for solving a dynamic facility layout problem. *European Journal of Operational Research*, 165(1):55–69.

Feo, T. A. e Resende, M. G. (1995). Greedy randomized adaptive search procedures. *Journal of global optimization*, 6(2):109–133.

Gonçalves, J. F. e Resende, M. G. (2015). A biased random-key genetic algorithm for the unequal area facility layout problem. *European Journal of Operational Research*, 246(1): 86–107.

Heragu, S. (1997). *Facilities design*. PWS Publishing, Boston, MA.

Hosseini, S., Al Khaled, A., e Vadlamani, S. (2014). Hybrid imperialist competitive algorithm, variable neighborhood search, and simulated annealing for dynamic facility layout problem. *Neural Computing and Applications*, 25(7-8):1871–1885.

Imam, M. H. e Mir, M. (1993). Automated layout of facilities of unequal areas. *Computers & industrial engineering*, 24(3):355–366.

Imam, M. H. e Mir, M. (1998). Cluster boundary search algorithm for building-block layout optimization. *Advances in Engineering Software*, 29(2):165–173.

Imam, M. e Mir, M. (1989). Nonlinear programming approach to automated topology optimization. *Computer-Aided Design*, 21(2):107–115.

Kusiak, A. e Heragu, S. S. (1987). The facility layout problem. *European Journal of operational research*, 29(3):229–251.

McKendall, A. R. e Hakobyan, A. (2010). Heuristics for the dynamic facility layout problem with unequal-area departments. *European Journal of Operational Research*, 201(1):171–182.

Mir, M. e Imam, M. (2001). A hybrid optimization approach for layout design of unequal-area facilities. *Computers & Industrial Engineering*, 39(1):49–63.

Paes, F. G., Pessoa, A. A., e Vidal, T. (2017). A hybrid genetic algorithm with decomposition phases for the unequal area facility layout problem. *European Journal of Operational Research*, 256(3):742–756.

Rosenblatt, M. J. (1986). The dynamics of plant layout. *Management Science*, 32(1):76–86.

Sahni, S. e Gonzalez, T. (1976). P-complete approximation problems. *Journal of the ACM (JACM)*, 23(3):555–565.

Syswerda, G. e Palmucci, J. (1991). The application of genetic algorithms to resource scheduling. In *ICGA*, p. 502–508.

Yang, T. e Peters, B. A. (1998). Flexible machine layout design for dynamic and uncertain production environments. *European Journal of Operational Research*, 108(1):49–64.