UM ALGORITMO GRASP APLICADO AO PROBLEMA DE *LAYOUT* DE FACILIDADES EM FILA ÚNICA

Gildasio Lecchi Cravo
Faculdades Integradas de Aracruz - FAACZ
Rua Professor Berilo Basílio dos Santos, 180 - Centro, Aracruz - ES, 29194-910
gildasio@fsjb.edu.br

André Renato Sales Amaral
Universidade Federal do Espírito Santo - UFES
Av. Fernando Ferrari, 514 - Goiabeiras, Vitória - ES, 29075-910
amaral@inf.ufes.br

RESUMO

Este trabalho apresenta uma implementação da metaheurística GRASP para o problema de layout de facilidades em fila única (SRFLP – single-row facility layout problem). O SRFLP é um problema NP-difícil que consiste em arranjar facilidades ao logo de uma linha reta, objetivando a minimização da soma ponderada das distâncias entre todos os pares de facilidades. O algoritmo GRASP proposto foi testado com várias instâncias disponíveis na literatura e os resultados obtidos foram comparados com os melhores disponíveis na literatura. Nos experimentos realizados, o algoritmo GRASP apresentou soluções de alta qualidade e com pequeno desvio entre as diversas execuções para uma mesma instância. Das 43 instâncias testadas, os melhores valores de solução encontrados pelo algoritmo GRASP são iguais aos melhores conhecidos para 41 instâncias, e para 2 instâncias os desvios dos melhores valores ficaram em 0,003% e 0,006%.

PALAVARAS CHAVE. SRFLP, Metaheurística, GRASP.

Área principal (Metaheurística, Otimização Combinatória, PO na Indústria)

ABSTRACT

This paper presents an implementation of the GRASP metaheuristic to the single-row facility layout problem (SRFLP). The SRFLP is an NP-hard problem, which consists in arranging facilities along a straight line so as to minimizing the weighted sum of the distances between all pairs of facilities. The proposed GRASP algorithm was tested on several available instances in the literature and the results obtained were compared with the best-known ones. In the experiments, the GRASP algorithm presented high-quality solutions and with little deviation between different executions for the same instance. Out of the 43 instances tested, the best solution values found by the GRASP algorithm equal the best-known ones for 41 instances, and for two instances the deviations from the best-known values stood at 0.003% and 0.006%.

KEYWORDS. SRFLP. Metaheuristic, GRASP.

Main area (Metaheuristic, Optimization Combinatorial, Operations Research in Industry)

1. Introdução

O Problema de Layout de Facilidades em Fila Única, conhecido na literatura como *Single-row Facility Layout Problem* (SRFLP) pode ser descrito como a tarefa de arranjar n facilidades ao longo de uma linha em uma dada direção. Considera-se l_i o comprimento da facilidade i. Também são conhecidas as intensidades de fluxo entre as facilidades (por exemplo, tráfego de pessoas entre as facilidades) definidas por uma matriz $C = [c_{ij}]$ de tamanho $n \times n$, onde c_{ij} é a intensidade do fluxo entre a facilidade i e a facilidade j. A distância entre duas facilidades é definida como a distância entre os seus centroides. O objetivo é encontrar um arranjo de facilidades de modo a minimizar a soma ponderada das distâncias entre todos os pares de facilidades, isto é:

Encontrar uma permutação $\prod = \{\pi_1, \pi_2, ..., \pi_n\}$ do conjunto de facilidades $F=\{1, 2, ..., n\}$ que forneça o melhor valor para a expressão:

$$\sum_{1 \le i < j \le n} c_{\pi_i \pi_j} d_{\pi_i \pi_j} \quad (1)$$

Onde $d_{\pi_i \pi_j} = \frac{l_{\pi_i} + l_{\pi_j}}{2} + \sum_{i<k<j} l_{\pi_k}$ é a distância entre os centroides das facilidades π_i e π_j de F dispostas na permutação $\prod$ (Kothari e Ghosh, 2012).

O problema foi proposto por Simmons (1969) e possui uma grande gama de aplicações práticas como, por exemplo, arranjo de departamentos ao longo de um corredor em supermercados, construção de hospitais e escritórios (Simmons, 1969), arranjo de livros em uma prateleira (Kumar et al. 1995), entre outros.

Na literatura do SRFLP, são encontrados vários métodos para obter soluções exatas. Simmons (1969) propôs um algoritmo *branch-and-bound*; Picard e Queyranne (1981) apresentaram um algoritmo de programação dinâmica para o SRFLP. Amaral (2009) propôs um algoritmo de planos de corte; Hungerländer e Rendl (2013) testaram uma abordagem baseada em programação semidefinida; e Amaral e Letchford (2013) propuseram um algoritmo branch-and-cut. Outros autores apresentaram formulações de programação inteira (Love e Wong, 1976, Heragu e Kusiak, 1991 e Amaral, 2006, 2008). Os melhores métodos exatos são os dados por Amaral (2009) e Hungerländer e Rendl (2013). Entretanto, o maior tamanho de instância que pôde ser resolvida eficientemente, até a presente data, possui no máximo cerca de 40 facilidades. Para problemas de maior porte torna-se proibitiva a aplicação de métodos exatos.

Métodos para encontrar limitantes inferiores para o SRFLP são descritos em Anjos, Kennings e Vannelli (2005); Amaral (2009); Anjos e Yen (2009), Hungerländer e Rendl (2013); e Amaral e Letchford (2013).

Sendo o SRFLP um problema NP-Difícil (Amaral, 2006), também estão disponíveis soluções heurísticas, tais como, *heurística gulosa* (Kumar et al., 1995), *simulated annealing* (Heragu e Alfa, 1992; De Alvarenga et al., 2000), *busca tabu* (De Alvarenga et al., 2000; Samarghandi e Eshghi, 2010; Kothari e Ghosh, 2013a), *algoritmos genéticos* (Ozcelik, 2011, Datta et al., 2011; Kothari e Ghosh, 2014a), *Scatter search* (Kumar et al., 2008; Kothari e Ghosh, 2014b), *colônia de formigas* (Solimanpur, Vrat e Shankar, 2005); e *particle swarm* (Samarghandi, Taabayanb e Jahantighc, 2010). Os melhores resultados heurísticos até então são encontrados nos trabalhos de Kothari e Ghosh (2013a, 2013b, 2014a, 2014b) e Ozcelik (2011).

O presente trabalho propõe uma implementação de um algoritmo GRASP (*Greedy Randomized Adaptative Search Procedure*) para o SRFLP. Em testes realizados com 43 instâncias da literatura, o algoritmo GRASP proposto encontrou valores de solução iguais aos melhores conhecidos, exceto para 2 instâncias, onde os desvios dos melhores valores ficaram em 0,003% e 0,006%, como será visto na seção de resultados computacionais.

No restante do trabalho, são apresentadas a metaheurística GRASP (seção 2) e a implementação proposta (seção 3). Na seção 4 são apresentados os testes computacionais e a comparação com resultados da literatura. Em seguida, as conclusões.

2. Metaheurística GRASP

O GRASP, proposto por Feo e Resende (1995), é um método iterativo constituído de duas fases: a de construção e a de busca local. Na fase de construção, uma solução é construída elemento a elemento. Gera-se uma *lista de candidatos* (LC) contendo elementos de uma solução do problema, ordenados de acordo com sua contribuição na função objetivo (ou de acordo com algum critério estabelecido). Os melhores elementos de LC formam uma lista, denominada *lista de candidatos restrita* (LCR). A construção de uma solução é probabilística, pois a escolha de cada elemento que será retirado de LCR para ser adicionado à solução é feita aleatoriamente. A heurística também é adaptativa, pois, a cada iteração da fase de construção, são atualizadas as contribuições dos elementos restantes na LC a fim de refletir as mudanças ocasionadas pela seleção do elemento adicionado à solução na iteração anterior.

Provavelmente, uma solução gerada ao final da fase de construção, não será localmente ótima. Daí a importância da segunda fase do GRASP, que tenta melhorar a solução construída na fase anterior, explorando a sua vizinhança.

Existem na literatura diversas aplicações práticas do método GRASP em problemas de otimização na indústria. Dentre esses estão inclusos problemas de roteirização, lógica, particionamento, localização, teoria dos grafos, transporte, telecomunicações, projeto VLSI, problema de rotulação cartográfica de pontos (Cravo, Ribeiro e Lorena, 2008), projeto de rede de transmissão (Faria et al., 2005). Outras aplicações do GRASP podem ser vistas em Resende e Ribeiro (2005).

3. GRASP proposto para o SRFLP

O pseudocódigo do GRASP proposto para o SRFLP é apresentado na Figura 1. O parâmetro *Iter* define o número de iterações do GRASP e o parâmetro *LCR_tam* define o tamanho da LCR (i.e. os melhores *LCR_tam* elementos de LC formam a LCR). A cada iteração é construída uma solução gulosa aleatória *S* e é aplicada a essa solução uma heurística de busca local que busca, na vizinhança de *S*, uma melhora na função objetivo. Se a solução melhorou (linha 5) a melhor solução encontrada até então é atualizada (linhas 6 e 7). Ao final do procedimento a melhor solução está armazenada em *S**.

```
Procedimento GRASP
Dados: Iter, LCR_tam
Resultado: Solução S*
01: f* ← ∞
02: Para i=1 até Iter Faça
03: S ← ConstruçãoGulosaAleatoria(LCR_tam);
04: S ← BuscaLocal(S);
05:    Se f(S) < f* Então
06:       f* ←f(S);
07:       S* ← S;
08:    Fim-Se
09: Fim-Para
10: Retorne S*;
11: Fim-GRASP
```

Figura 1 – pseudocódigo do GRASP proposto para o SRFLP

No algoritmo GRASP proposto, cada elemento da lista LC é um par de facilidades. Dessa lista, é sorteado, dentre os *LCR_tam* melhores elementos de LC, um par que irá fazer parte da solução (linha 3). A cada par selecionado a LC é atualizada para refletir o custo dos pares ainda não adicionados à solução parcial em construção. Quando não houver mais pares para adicionar à solução, a etapa de construção da solução inicial estará concluída. Assim, todas as facilidades têm suas posições definidas na permutação e, é aplicado um procedimento de busca local, consistindo de três estruturas de vizinhanças: uma baseada no movimento de Inserção e duas baseadas no movimento 2-OPT. O procedimento é repetido *Iter* vezes (linha 2).

3.1 Fase Construtiva do GRASP

A ideia da fase construtiva é criar uma lista LC com os possíveis pares de facilidades que poderão compor uma solução a partir da posição livre mais a esquerda *p* e da posição livre mais a direta *q* na solução parcial corrente.

Quando uma solução *S* está vazia, o custo w_{ij} do par de facilidades *i* e *j*, tal que *i* será colocada na posição livre *p*=1 e *j* será colocada na posição livre *q*=*n*, é calculado como:

$$w_{ij} = c_{ij}\left(\frac{l_i + l_j}{2} + \sum_{1\le k\le n; k\ne i, k\ne j} l_k\right), \quad (1 \le i < j \le n) \tag{2}$$

Assim, a LC é criada, ordenada pelo custo w_{ij}. Os *LCR_tam* melhores elementos de LC definem a LCR e então, um elemento da lista LCR é sorteado e adicionado à solução *S* vazia nas posições *p* e *q*.

A partir deste momento, como a solução não é mais vazia, os custos w_{ij} serão calculados levando em consideração os elementos já pertencentes à solução *S*.

Seja *S'* o conjunto dos elementos à esquerda da próxima posição livre *p* na solução *S* e *S''* o conjunto dos elementos à direita da próxima posição livre *q* na solução *S*. O custo w_{ij} é calculado conforme a expressão:

$$w_{ij} = c_{ij}\left(\frac{l_i + l_j}{2} + \sum_{k\ne i,\, k\ne j,\, k\notin S} l_k\right) + \sum_{k\in S'}(d_{ki}c_{ki} + d_{kj}\, c_{kj}) + \sum_{k\in S''}(d_{ki}c_{ki} + d_{kj}\, c_{kj}), \quad \forall\, i, j \notin S \tag{3}$$

Na expressão (3), para um par candidato (*i*, *j*), $i, j \notin S$, será escolhida a melhor configuração, pois tem-se a possibilidade de colocar a facilidade *i* na posição *p* e a facilidade *j* na posição *q* ou *i* na posição *q* e *j* na posição *p*. A Figura 2 ilustra a aplicação das expressões (2) e (3) para um problema com 6 facilidades.

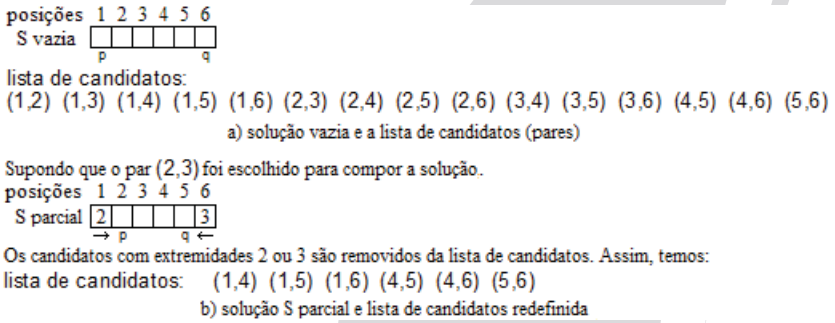

Figura 2 – aplicação das expressões (2) e (3) para geração de uma solução inicial

A Figura 2(a) apresenta uma solução *S* vazia com 6 posições livres. Assim, a lista de candidatos é gerada e os pesos dos pares de facilidades são calculados conforme expressão (2):

$w_{12} = c_{12}((l_1 + l_2)/2 + l_3 + l_4 + l_5 + l_6)$ $\quad w_{26} = c_{26}((l_2 + l_6)/2 + l_1 + l_3 + l_4 + l_5)$
$w_{13} = c_{13}((l_1 + l_3)/2 + l_2 + l_4 + l_5 + l_6)$ $\quad w_{34} = c_{34}((l_3 + l_4)/2 + l_1 + l_2 + l_5 + l_6)$
$w_{14} = c_{14}((l_1 + l_4)/2 + l_2 + l_3 + l_5 + l_6)$ $\quad w_{35} = c_{35}((l_3 + l_5)/2 + l_1 + l_2 + l_4 + l_6)$
$w_{15} = c_{15}((l_1 + l_5)/2 + l_2 + l_3 + l_4 + l_6)$ $\quad w_{36} = c_{36}((l_3 + l_6)/2 + l_1 + l_2 + l_4 + l_5)$
$w_{16} = c_{16}((l_1 + l_6)/2 + l_2 + l_3 + l_4 + l_5)$ $\quad w_{45} = c_{45}((l_4 + l_5)/2 + l_1 + l_2 + l_3 + l_6)$
$w_{23} = c_{23}((l_2 + l_3)/2 + l_1 + l_4 + l_5 + l_6)$ $\quad w_{46} = c_{46}((l_4 + l_6)/2 + l_1 + l_2 + l_3 + l_5)$
$w_{24} = c_{24}((l_2 + l_4)/2 + l_1 + l_3 + l_5 + l_6)$ $\quad w_{56} = c_{56}((l_5 + l_6)/2 + l_1 + l_2 + l_3 + l_4)$
$w_{25} = c_{25}((l_2 + l_5)/2 + l_1 + l_3 + l_4 + l_6)$

Com os pesos calculados, a lista de candidatos é ordenada, de forma não decrescente,

pelos pesos w_{ij} e um elemento é sorteado para compor a solução. Na Figura 2(b), na solução parcial *S* as facilidades 2 e 3 foram colocadas nas posições 1 e 6, respectivamente. Assim, a lista de candidatos é redefinida como mostrado na Figura 2(b) e os pesos w_{ij} são recalculados conforme expressão (3):

$$w_{14} = c_{14}((l_1 + l_4)/2 + l_5 + l_6) + (d_{21}c_{21} + d_{24}c_{24}) + (d_{31}c_{31} + d_{34}c_{34})$$
$$w_{15} = c_{15}((l_1 + l_5)/2 + l_4 + l_6) + (d_{21}c_{21} + d_{25}c_{25}) + (d_{31}c_{31} + d_{35}c_{35})$$
$$w_{16} = c_{16}((l_1 + l_6)/2 + l_4 + l_5) + (d_{21}c_{21} + d_{26}c_{26}) + (d_{31}c_{31} + d_{36}c_{36})$$
$$w_{45} = c_{45}((l_4 + l_5)/2 + l_1 + l_6) + (d_{24}c_{24} + d_{25}c_{25}) + (d_{34}c_{34} + d_{35}c_{35})$$
$$w_{46} = c_{46}((l_4 + l_6)/2 + l_1 + l_5) + (d_{24}c_{24} + d_{26}c_{26}) + (d_{34}c_{34} + d_{36}c_{36})$$
$$w_{56} = c_{56}((l_5 + l_6)/2 + l_1 + l_4) + (d_{25}c_{25} + d_{26}c_{26}) + (d_{35}c_{35} + d_{36}c_{36})$$

Com os pesos recalculados, ordena-se novamente a lista, um novo elemento é sorteado para compor a solução *S*, a lista de candidatos é redefinida e os pesos recalculados. O procedimento é repetido até que se tenha uma solução *S* completa. A Figura 3 apresenta o pseudocódigo para fase construtiva do GRASP.

Uma observação a ser feita é que quando temos *n* (quantidade de facilidades) ímpar coloca-se a facilidade que ficou sem par (que não foi alocada) no último passo, na posição central da solução/permutação.

```
Procedimento ConstruçãoGulosaAleatoria
Dados: n, LCR_Tam //Tamanho da lista de candidatos restrita
Resultado: S
01: Alocados = {"F", ...,"F"} // F- não alocado, V-alocado
02: S ← {} //Lista de facilidades, ou seja, a solução
03: p ← 1, q ← n // posições livres
04: LC ← CriarLC_Inicial();
05: Enquanto ( LC ≠ {} ) Faça
06:    par ← Sortear (LCR_tam, LC);
07:    S(p) ← par.i;
08:    S(q) ← par.j;
09:    p ← p + 1;
10:    q ← q - 1;
11:    Alocados(par.i) ← Alocados(par.j) ← " V ";
12:    RemoverConflitos(par, LC);
13:    AtualizarListaCandidatos(S, LC);
14:    OrdenarListaCandidatos(LC);
15: Fim-Enquanto
16:    Se (n é ímpar) Então
17:       S((n/2)+1) ← BuscaNãoAlocado (Alocados);
18:    Fim-Se
19: Retorne S;
20: Fim-ConstruçãoGulosaAleatoria
```

Figura 3 – pseudocódigo para a fase de construção do GRASP

O procedimento mostrado na Figura 3 recebe *n* (quantidade de facilidades) e *LCR_tam* (quantos elementos da LC fazem parte da LCR). Na linha 1, 2 e 3 são inicializados: o vetor "*Alocados*" que controla as facilidades já alocadas na solução *S* colocando "F" (Falso) para todas as facilidades, a solução *S* como vazia e as posições livres *p* e *q*, sendo *p* a posição mais esquerda e *q* a posição mais a direita na solução *S* e na linha 4, a LC é criada conforme a expressão (2).

O laço *Enquanto* definido nas linhas 5 até 15 executa até que não se tenha mais candidatos na LC. Na linha 6, o candidato para compor a solução é sorteado e, são adicionadas em *S* as facilidades nas posições livres (linha 7 e 8); incrementa-se a posição livre *p* e decrementa-se a posição livre *q* (linhas 9 e 10) e marcam-se como alocadas as facilidades *i* e *j* do par selecionado, na linha 11. Na linha 12, removem-se os pares que têm extremidade conflitante; na linha 13, atualizam-se os custos dos candidatos remanescentes na LC e na linha 14 ordena-se a LC novamente pelos custos (atualizados conforme expressão 3).

O bloco da linha 16 verifica se a instância possui uma quantidade ímpar de facilidades, pois, se possuir, a solução deverá ser acertada, buscando a facilidade ainda não alocada para colocá-la na posição central da permutação *S*.

O procedimento ConstruçãoGulosaAleatoria é um algoritmo guloso, aleatório e a propriedade adaptativa reside no fato de que a cada par de facilidades adicionado à solução *S*, os elementos conflitantes são removidos e os custos dos elementos restantes são atualizados, sendo recalculados levando-se em consideração a solução parcial *S* que está sendo construída. Ao final dessa etapa uma solução viável inicial *S* é devolvida (linha 19).

Como a construção da permutação *S* é feita de forma gulosa e aleatória é provável que a solução não seja localmente ótima. A fim de melhorar a solução inicial encontrada nessa primeira fase, uma busca local é aplicada.

3.2 Segunda Fase do GRASP

A segunda fase do GRASP aplica uma Busca Local baseada nas estruturas de vizinhanças *2-OPT* e *Inserção*, ambas bastante usadas na literatura para o SRFLP (Ozcelik, 2011; Kothari e Ghosh, 2013a). A Figura 4 mostra o pseudocódigo da busca local.

```
Procedimento BuscaLocal
Dados: S // Solução Inicial
Resultado: S* // Solução localmente ótima
01: S* ← S;
02: melhorou ← " V ";
03: Enquanto melhorou= " V " Faça
04:    melhorou ← " F ";
05:    2_Opt_Pares(S);
06:    Inserção(S);
07:    2_Opt_N(S);
08:    Se (f(S) < f(S*)) Então
09:       S*← S;
10:       melhorou ← " V ";
11:    Fim-Se
12: Fim-Enquanto
13: Retorne S*;
14: Fim-BuscaLocal
```

Figura 4 – pseudocódigo para a fase de busca local do GRASP

A busca local segue uma estrutura comum em que os procedimentos para melhoria da solução corrente são executados até que não se encontre um vizinho com um custo melhor.

A busca local principal (Figura 4) trabalha com três buscas locais internamente. Essas buscas são definidas como ***2_Opt_Pares*** (linha 5), ***Inserção*** (linha 6) e ***2_Opt_N*** (linha 7). Note que a vizinhança é explorada na mesma sequência, mas sempre a partir de uma solução inicial diferente o que evita uma busca tendenciosa.

A estrutura de vizinhança ***2_Opt_Pares*** (linha 5) aplicada à permutação *S* é um procedimento *2-OPT* em que dois pares de facilidades da solução atual são trocados. A ideia é tentar gerar soluções vizinhas efetuando as trocas dos pares que foram escolhidos e atribuídos nas posições na primeira fase do GRASP. A Figura 5 mostra um exemplo e as trocas entre os pares.

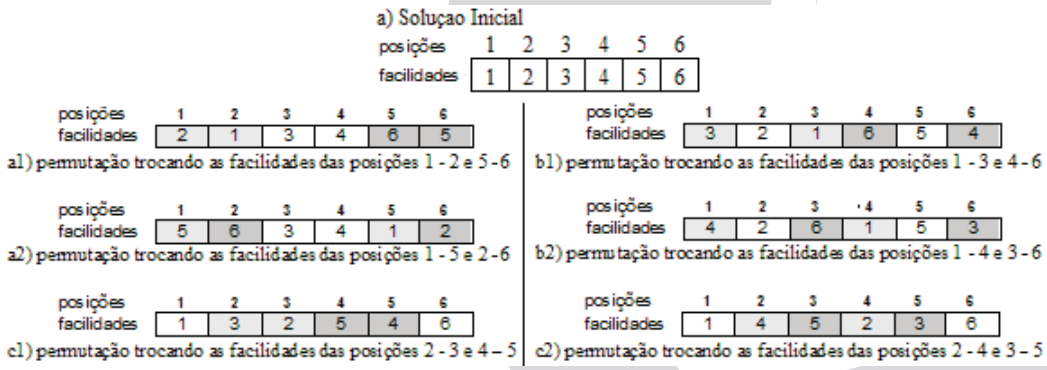

Figura 5 – aplicação da *2_Opt_Pares* a uma solução com 6 facilidades

As soluções vizinhas geradas pela *2_Opt_Pares* proposta neste trabalho, são obtidas pelas trocas entre os pares de facilidades das posições (conforme Figura 5):

- (1, 6) x (2, 5): trocar as facilidades das posições a1=(1-2 e 5-6) e a2=(1-5 e 2-6).
- (1, 6) x (3, 4): trocar as facilidades das posições b1=(1-3 e 4-6) e b2=(1-4 e 3-6).
- (2, 5) x (3, 4): trocar as facilidades das posições c1=(2-3 e 4-5) e c2=(2-4 e 3-5).

A cada movimento (nova vizinhança gerada) verifica-se se o custo da solução melhorou. O movimento é desfeito se o custo da solução não melhorou.

Após a aplicação da *2_Opt_Pares* é aplicada a busca local *Inserção* (Figura 4, linha 6). A vizinhança por *Inserção* é bastante utilizada na literatura para o SRFLP (Ozcelik, 2011; Kothari e Ghosh, 2013a). A Figura 6 mostra como a vizinhança por inserção trabalha com uma solução.

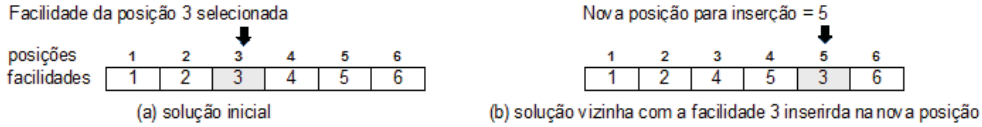

Figura 6 – movimento da estrutura de vizinhança por inserção

Como pode ser visto na Figura 6, a facilidade que estava na posição 3 da solução foi selecionada e inserida na posição 5. Para isso, as facilidades das posições 4 e 5 foram deslocadas e a facilidade selecionada foi inserida na posição 5.

A *Inserção* é feita selecionando a facilidade da posição i=1,...,n da solução tentando inserir na posição j=1, ..., n com $j \neq i$. Se o movimento melhora o custo da solução, ele é aceito, senão é desfeito. Após a aplicação da *Inserção* a busca local *2_Opt_N* é aplicada (Figura 4, linha 7).

A estrutura de vizinhança *2_Opt_N* gera soluções vizinhas realizando trocas de pares de facilidades definidas em esquemas de trocas, baseando-se na ideia apresentada em Ahonen et. al (2014). A Figura 7(a) apresenta os esquemas para uma instância com 6 facilidades, sendo: "origem", o primeiro elemento a ser trocado, "destino", a posição onde a troca irá começar e *t*, quantos elementos, a partir da origem, serão trocados.

Supondo os esquemas definidos na Figura 7(a), a aplicação do esquema destacado (em cinza) geraria uma solução vizinha, a partir da solução da Figura 5(a), como mostrada na Figura 7(b).

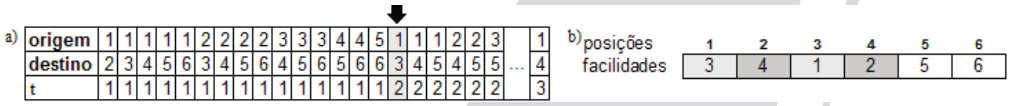

Figura 7 – (a) esquemas de trocas para uma instância com 6 facilidades e (b) uma solução vizinha

No esquema selecionado, t = 2 indica que a partir da "origem", no caso a posição 1 (inclusive) devem-se trocar 2 elementos consecutivos, ou seja, a facilidade que estava na posição 1 foi trocada com a facilidade da posição 3 e a facilidade que estava na posição 2 trocada pela facilidade da posição 4.

Os esquemas são gerados inicialmente levando em consideração o tamanho da instância. Em testes preliminares, com as instâncias Sko64, os esquemas com *t* de 1 (um) a, no máximo, 8% (oito por cento) do tamanho da instância apresentaram melhora no valor da função objetivo (para uma mesma solução inicial) e tempo computacional razoável. Com *t* acima 8% a melhora não acontece mais, só aumentando o tempo de execução da busca local.

Os esquemas são armazenados em uma lista e são tentados na ordem que foram gerados. Se uma troca aplicada melhora o valor do custo total da solução ela é aceita, senão é desfeita.

4. Experimentos Computacionais

Para a validação do algoritmo GRASP proposto para o SRFLP foram executados testes com instâncias apresentadas por Anjos, Kennings e Vannelli (2005), Amaral e Letchford (2011) e as instâncias "SKO" de Anjos e Yen, (2009).

O GRASP tem um importante parâmetro que é o tamanho da lista de candidatos restrita (*LCR_Tam*, na Figura 1). Para a calibração desse parâmetro foram utilizadas as cinco instâncias com n=64 facilidades. O GRASP foi executados 10 (dez) vezes para cada uma dessas instâncias, fazendo $LCR\_tam = \gamma \cdot n$, onde $\gamma \in \{5\%, 10\%, 15\%, \ldots, 95\%, 100\%\}$ como mostrado na Figura 8.

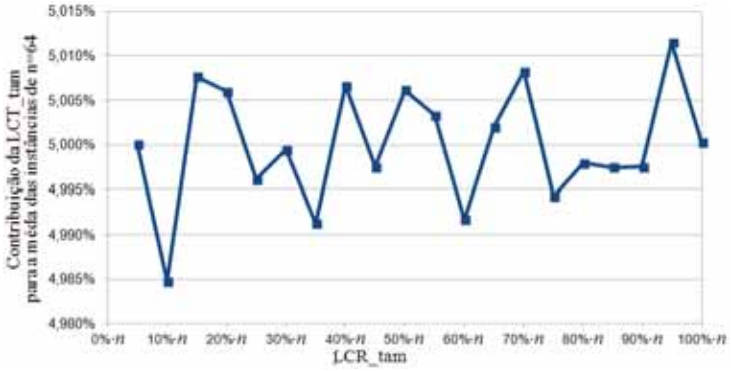

Figura 8 – resultado do teste para calibração do tamanho da LCR

O gráfico da Figura 8, mostra, para cada valor da *LCR_tam*, a contribuição das soluções encontradas para a média das soluções encontradas (dentre todas as instâncias de n=64).

Assim, como pode-se observar, fazer $LCR\_tam = 10\% \cdot n$ levou a soluções com custos menores, sendo então este valor de *LCR_tam* escolhido para os testes computacionais realizados.

O GRASP foi executado com quinhentas iterações. Cada instância foi executada dez vezes, usando-se um *laptop* com processador Intel Core 2 Duo de 2.1Ghz com 3GB de memória DDR2.

A Tabela 1 mostra os resultados encontrados nas dez execuções do GRASP, com $LCR\_tam = 10\% \cdot n$ e 500 iterações, para as vinte instâncias de Anjos, Kennings e Vannelli (2005). A primeira coluna contém o nome da instância, a segunda mostra o tamanho da instância, a terceira é o melhor valor de solução encontrado, a quarta contém a média das dez execuções e a coluna "Desvio Médio" apresenta a média dos desvios entre os valores de soluções e a média encontradas nas 10 execuções do GRASP (em percentual). A sexta coluna contém a média das iterações em que foi obtida o melhor valor de solução, a sétima coluna apresenta o tempo médio decorrido para encontrar o melhor valor de solução, a oitava coluna apresenta o tempo médio das 10 execuções do GRASP e por último os valores da melhor solução na literatura (Kothari e Ghosh, 2014b) e a coluna "*Gap*" mostra o erro, em percentual, entre a melhor valor de solução do GRASP e a melhor valor da literatura.

Tabela 1 – Resultados do GRASP para instâncias de Anjos, Kennings e Vannelli(2005)

Instância	n	Valor Objetivo		Desvio	Última Melhora		Tempo (s)	Literatura	*Gap*
		Melhor	Média	Médio (%)	Iter. Média	Tempo(s) Médio	Médio Total	Melhor Valor	
Anjos_60_1	60	1477834	1477834	0,000%	7	5	304	1477834	0,000%
Anjos_60_2	60	841776	841776	0,000%	8	6	332	841776	0,000%
Anjos_60_3	60	648337,5	648337,5	0,000%	97	67	348	648337,5	0,000%
Anjos_60_4	60	398406	398406	0,000%	22	16	345	398406	0,000%
Anjos_60_5	60	318805	318805	0,000%	4	4	346	318805	0,000%
Anjos_70_1	70	1528537	1528537	0,000%	9	14	768	1528537	0,000%
Anjos_70_2	70	1441028	1441028	0,000%	4	8	714	1441028	0,000%
Anjos_70_3	70	1518993,5	1518993,5	0,000%	4	8	789	1518993,5	0,000%
Anjos_70_4	70	968796	968796	0,000%	20	35	832	968796	0,000%
Anjos_70_5	70	4218002,5	4218002,5	0,000%	2	5	806	4218002,5	0,000%
Anjos_75_1	75	2393456,5	2393456,5	0,000%	10	27	1210	2393456,5	0,000%
Anjos_75_2	75	4321190	4321190	0,000%	51	117	1165	4321190	0,000%
Anjos_75_3	75	1248423	1248423	0,000%	25	61	1193	1248423	0,000%
Anjos_75_4	75	3941816,5	3941816,5	0,000%	11	27	1181	3941816,5	0,000%
Anjos_75_5	75	1791408	1791408	0,000%	51	115	1150	1791408	0,000%

Anjos_80_1	80	2069097,5	2069097,5	0,000%	7	25	1527	2069097,5	0,000%
Anjos_80_2	80	1921136	1921136	0,000%	11	38	1570	1921136	0,000%
Anjos_80_3	80	3251368	3251368	0,000%	7	26	1612	3251368	0,000%
Anjos_80_4	80	3746515	3746515	0,000%	8	32	1692	3746515	0,000%
Anjos_80_5	80	1588885	1588885	0,000%	45	150	1597	1588885	0,000%

Como nota-se na Tabela 1, o GRASP encontrou soluções com os mesmos valores que aqueles das melhores soluções conhecidas na literatura para as vinte instâncias de Anjos, Kennings e Vannelli (2005). O algoritmo apresenta uma robustez que pode ser observada pelos valores dos desvios médios que permanecem 0,000% para todas as instâncias.

Na Tabela 2 estão dispostos os resultados encontrados pelo GRASP para as vinte instâncias SKO. Os resultados estão dispostos como na Tabela 1.

Tabela 2 – Comparação dos resultados do GRASP com os da literatura, instâncias "SKO"

Instância	n	Valor Objetivo		Desvio	Última Melhora		Tempo (s)	Literatura	
		Melhor	Média	Médio (%)	Iter. Média	Tempo(s) Médio	Médio Total	Melhor Valor	*Gap*
Sko64-1	64	96881	96884,1	0,003%	306	157	421	96881	0,000%
Sko64-2	64	634332,5	634332,5	0,000%	55	43	379	634332,5	0,000%
Sko64-3	64	414323,5	414323,5	0,000%	344	67	378	414323,5	0,000%
Sko64-4	64	297129	297135	0,003%	213	150	386	297129	0,000%
Sko64-5	64	501922,5	501923,7	0,000%	202	47	384	501922,5	0,000%
Sko72-1	72	139150	139151,9	0,002%	265	325	760	139150	0,000%
Sko72-2	72	711998	712001,5	0,000%	343	234	647	711998	0,000%
Sko72-3	72	1054110,5	1054110,5	0,000%	283	91	633	1054110,5	0,000%
Sko72-4	72	919586,5	919586,5	0,000%	331	219	628	919586,5	0,000%
Sko72-5	72	428226,5	428234,6	0,002%	252	343	674	428226,5	0,000%
Sko81-1	81	205114	205142,3	0,010%	416	690	1315	205108,5	0,003%
Sko81-2	81	521391,5	521392,8	0,000%	342	294	1191	521391,5	0,000%
Sko81-3	81	970796	970816,2	0,002%	319	474	1185	970796	0,000%
Sko81-4	81	2031803	2031832	0,002%	340	555	1192	2031803	0,000%
Sko81-5	81	1302711	1302788,9	0,008%	296	409	1150	1302711	0,000%
Sko100-1	100	378258	378350,7	0,009%	383	1504	4269	378234	0,006%
Sko100-2	100	2076008,5	2076022,9	0,000%	459	2246	3611	2076008,5	0,000%
Sko100-3	100	16145614,5	16145732,5	0,001%	349	1666	3449	16145614,5	0,000%
Sko100-4	100	3232522	3232526,3	0,000%	231	729	3333	3232522	0,000%
Sko100-5	100	1033080,5	1033130,6	0,006%	349	1353	3475	1033080,5	0,000%

Novamente, pode-se notar que os desvios médios para as instâncias "SKO" também permaneceram baixos, mostrando também a robustez do algoritmo nesse conjunto de instâncias. Na coluna "*Gap*" tem-se erros de 0% para 18 instâncias, e para as instâncias Sko81-1 e Sko100-1 os erros de apenas 0,003% e 0,006%, respectivamente.

Na Tabela 3 estão dispostos os resultados encontrados pelo algoritmo GRASP para as instâncias do SRFLP com 110 facilidades dadas por Amaral e Letchford (2011).

Tabela 3 – Comparação dos resultados do GRASP com os da literatura, instâncias SRFLP-110

Instância	n	Valor Objetivo		Desvio	Última Melhora		Tempo (s)	Literatura	
		Melhor FO	Média FO	Médio (%)	Iter. Média	Tempo(s) Médio	Médio Total	Melhor Valor	*Gap*
SRFLP-110-1	110	144296664,5	144299856,6	0,003%	286	4351	7600	144296664,5	0,000%
SRFLP-110-2	110	86050037	86051075,2	0,002%	187	3001	7730	86050037	0,000%
SRFLP-110-3	110	2234743,5	2234756,6	0,001%	235	3691	7876	2234743,5	0,000%

Para as instâncias "SRFLP-110" o algoritmo GRASP, mais uma vez, obteve soluções com valores iguais aos melhores encontradas na literatura e para essas instâncias os desvios médios também permanecem baixos.

No apêndice A, estão disponibilizadas as melhores soluções encontradas pelo algoritmo GRASP para cada instância utilizada.

5. Conclusões

O algoritmo aqui proposto é a primeira implementação do GRASP para o SRFLP. Foram propostas duas estruturas de vizinhança: a *2_Opt_X* que tenta explorar uma solução através da troca de pares de elementos escolhidos na fase de construção da solução e a *2_Opt_N* onde são definidos esquemas de trocas de um par de elementos por outro par (ao invés da troca de um elemento por outro elemento como em uma busca 2-OPT convencional). Entendemos que as ideias discutidas nessa implementação poderão auxiliar futuros estudos que tratem da aplicação do GRASP para o problema.

O algoritmo GRASP proposto obteve soluções robustas para todas as execuções, como podem ser observadas pelos baixos desvios médios. Na comparação com os resultados disponíveis na literatura, o algoritmo GRASP perde somente nas instâncias Sko81-1 e Sko100-1 tendo um G*ap* em relação ao melhor valor de solução da literatura de 0,003% e 0,006%, respectivamente.

Na implementação apresentada, a cada iteração, é realizada a reavaliação completa da função objetivo das soluções vizinhas geradas. Assim, pretendemos trabalhar nesse aspecto para reduzir o tempo de processamento. Outro ponto a ser trabalhado é o aumento da diversificação das soluções geradas, pois nos testes das instâncias de Anjos, Kennings e Vannelli (2005) nota-se que rapidamente o algoritmo encontra a solução final e a partir desse ponto não se observa melhoramentos.

Apêndice A – Instâncias e disposição das facilidades para as melhores soluções encontradas pelo GRASP proposto

Instância	Solução
Anjos_60_1	27 11 48 56 28 24 15 21 3 40 25 51 6 46 54 7 57 12 47 9 36 44 16 13 35 4 32 33 8 26 2 17 14 42 22 49 41 29 38 5 23 45 31 43 37 50 53 10 52 59 19 30 58 60 55 20 39 18 34 1
Anjos_60_2	20 11 44 35 57 22 19 4 2 23 48 12 3 31 49 7 47 38 60 16 58 55 26 51 59 43 6 27 52 36 18 45 46 17 39 54 1 24 5 25 37 56 21 28 33 40 32 34 14 8 42 15 9 30 50 29 41 13 10 53
Anjos_60_3	4 6 59 53 49 36 47 27 22 54 13 1 32 34 31 10 7 51 58 23 28 56 25 45 60 17 39 26 30 46 43 8 50 33 40 35 9 15 18 12 57 37 55 52 14 2 21 44 41 11 20 48 24 16 29 38 42 3 19 5
Anjos_60_4	13 20 31 19 55 57 10 39 7 44 24 22 36 33 56 11 21 6 23 17 46 40 59 53 5 3 45 2 50 35 14 27 18 15 58 8 16 60 29 26 42 37 54 47 30 51 32 48 52 9 49 38 28 1 4 41 34 25 43 12
Anjos_60_5	12 33 49 44 52 34 38 3 11 1 26 43 29 32 14 57 23 22 18 5 51 15 58 31 45 24 56 42 21 27 10 50 6 48 25 8 28 40 60 36 20 30 35 17 46 55 2 13 9 54 41 19 4 59 47 16 7 37 53 39
Anjos_70_1	50 37 35 39 57 29 17 48 43 25 46 45 34 3 66 20 7 52 59 6 58 12 11 54 27 24 21 18 13 60 55 23 36 30 5 49 42 33 19 14 26 10 44 70 67 56 38 41 61 64 16 4 1 68 51 69 31 63 9 32 8 15 22 62 28 40 2 65 47 53
Anjos_70_2	56 25 57 64 61 51 59 41 48 66 70 60 38 17 65 12 35 55 32 31 34 37 15 27 5 22 46 54 20 4 2 40 8 16 42 24 43 44 49 6 68 14 11 63 30 18 62 29 45 23 33 67 9 58 69 13 19 3 7 1 26 36 21 47 10 52 53 50 39 28
Anjos_70_3	8 30 63 51 54 27 29 43 21 32 23 24 20 1 41 67 6 47 9 26 11 65 22 4 61 7 70 19 2 42 37 68 50 10 14 49 53 5 25 56 18 28 66 39 36 40 3 35 62 60 45 13 55 48 12 59 58 38 33 46 15 64 69 34 17 31 57 52 44 16
Anjos_70_4	16 25 7 15 51 30 47 52 9 63 54 6 64 22 11 29 61 42 33 12 14 23 58 60 68 53 24 26 5 46 2 34 18 36 50 1 28 44 27 37 49 56 57 69 59 19 70 17 21 67 55 62 40 20 48 43 3 13 32 10 45 65 31 39 38 4 8 35 66 41
Anjos_70_5	67 28 17 44 64 32 69 33 18 13 52 47 25 57 48 45 55 15 59 42 66 3 46 60 34 29 7 11 53 21 4 12 36 30 5 70 20 35 14 24 56 43 41 61 8 26 49 27 63 65 16 54 6 10 39 22 50 37 58 40 2 51 31 62 23 68 1 9 38 19
Anjos_75_1	38 47 46 16 29 9 63 72 54 28 51 5 13 33 21 8 45 17 22 25 11 75 39 62 53 43 3 68 70 4 66 1 10 31 34 12 19 74 20 42 32 24 2 37 40 55 71 44 41 7 35 6 65 50 23 30 49 15 61 56 18 26 59 27 52 36 57 73 67 69 48 60 14 64 58
Anjos_75_2	25 26 43 56 37 72 47 70 66 30 39 45 35 24 69 54 33 10 41 4 19 31 9 74 17 48 36 46 55 13 68 16 38 27 15 62 8 12 23 22 11 67 34 3 71 65 75 20 53 5 52 58 73 29 1 57 59 7 18 50 60 42 40 63 14 6 44 28 2 21 61 32 64 51 49
Anjos_75_3	47 69 42 10 19 33 15 17 43 51 41 46 29 23 68 26 60 4 39 74 64 61 56 20 36 12 27 13 48 71 11 65 57 5 67 45 21 28 35 24 9 75 58 73 40 7 32 6 49 52 59 34 3 16 62 31 30 44 37 2 38 66 70 18 72 8 25 55 53 63 14 1 54 50 22
Anjos_75_4	36 60 5 14 15 50 7 75 10 42 62 37 8 70 30 47 22 57 20 41 29 40 33 39 46 12 3 64 35 65 16 52 28 53 44 73 34 18 24 45 13 32 1 67 2 19 55 48 56 63 66 26 23 58 59 54 43 71 4 31 11 74 61 51 6 25 27 68 69 38 72 49 9 17 21
Anjos_75_5	24 20 37 72 52 51 50 60 35 43 58 1 68 71 44 47 74 32 7 65 73 66 23 31 22 38 49 8 42 10 28 3 5 30 17 70 12 55 29 25 46 45 34 36 59 11 57 54 48 15 67 4 56 21 69 75 39 61 18 41 16 53 14 40 64 62 2 6 13 19 63 9 26 33 27
Anjos_80_1	55 70 2 60 80 67 44 27 65 40 21 16 29 43 19 46 14 63 61 39 53 52 12 68 47 45 75 49 51 34 62 56 66 54 48 41 32 20 10 50 58 15 42 76 9 31 36 6 22 23 74 33 13 17 24 37 18 7 69 26 72 73 30 5 59 35 3 8 77 38 64 28 1 4 71 79 11 57 25 78
Anjos_80_2	11 66 61 45 48 43 49 63 69 51 80 60 78 64 52 31 71 37 79 7 70 30 73 76 59 19 68 26 13 75 42 62 44 50 2 65 8 34 40 9 36 39 38 4 20 53 3 67 6 21 35 25 23 28 27 72 1 77 33 16 29 41 54 12 47 74 32 56 15 46 14 58 5 55 17 10 24 18 57 22
Anjos_80_3	57 63 54 53 37 62 61 26 29 32 64 35 48 72 30 22 25 76 67 77 10 50 52 51 8 3 39 78 27 17 80 44 21 14 11 59 23 7 41 19 73 75 15 1 18 2 71 55 60 66 9 38 31 6 12 28 79 65 56 4 42 24 5 45 47 46 43 13 70 69 49 34 36 68 20 33 16 58 40 74
Anjos_80_4	69 60 64 75 71 38 35 78 54 30 46 28 32 42 66 80 12 43 74 76 14 29 9 62 27 17 6 4 18 67 15 3 7 16 2 20 72 31 19 79 52 26 1 33 48 37 53 77 13 49 8 51 65 70 5 59 11 39 45 68 23 25 73 63 34 24 55 10 44 36 21 58 57 22 56 61 40 50 41 47
Anjos_80_5	2 7 47 52 38 27 61 73 21 67 68 10 37 74 11 22 1 70 8 31 19 9 76 33 36 62 30 34 79 75 54 44 6 43 53 35 71 66 55 65 59 3 32 39 77 18 4 80 20 45 40 63 23 46 50 57 5 56 15 24 28 26 64 14 51 16 29 12 72 42 49 13 78 69 60 58 41 48 17 25
Sko64-1	31 57 35 18 60 63 2 56 25 50 37 17 26 59 7 1 45 33 44 21 14 8 58 24 22 9 29 52 23 13 49 10 16 53 42 4 64 19 20 62 55 48 38 6 47 51 39 40 5 30 43 28 11 54 32 46 3 34 41 61 36 27 12 15
Sko64-2	47 18 32 27 30 5 40 51 3 43 54 34 41 39 12 11 14 46 36 61 52 15 38 6 28 55 48 60 20 23 29 45 7 1 37 50 16 56 35 17 21 49 59 4 24 44 2 13 33 31 9 10 58 26 22 57 63 62 64 42 25 19 53 8
Sko64-3	15 12 61 9 56 41 42 49 13 29 4 52 22 23 16 46 36 51 64 55 21 27 31 3 44 14 58 57 24 53 10 25 63 43 18 47 30 35 17 38 34 45 1 39 5 60 26 28 40 11 54 2 8 33 37 19 32 48 20 7 50 59 62 6

Sko64-4	15 59 57 53 11 54 32 41 30 8 19 10 33 56 25 17 2 38 50 26 34 37 1 7 45 47 35 44 58 13 49 21 24 4 64 39 62 5 16 23 12 29 55 22 9 28 48 14 40 61 52 18 6 3 51 46 20 63 27 43 42 36 60 31
Sko64-5	36 9 18 22 29 63 49 20 46 61 12 52 14 10 64 13 58 55 42 16 6 5 43 51 47 4 27 31 3 23 40 28 60 48 24 30 56 21 35 45 41 54 11 39 33 34 38 32 8 7 57 37 1 44 2 19 17 26 50 53 25 15 59 62
Sko72-1	12 53 31 64 60 18 35 8 56 10 70 27 30 17 67 21 2 28 26 14 22 47 65 52 23 61 29 1 7 38 59 46 49 33 50 13 37 63 32 39 16 9 6 71 44 55 57 36 51 68 42 40 48 41 15 19 58 20 66 43 4 62 72 69 25 11 5 45 54 24 3 34
Sko72-2	34 5 58 41 45 11 62 64 20 9 40 71 17 1 19 4 66 54 6 16 72 61 15 36 69 68 57 26 44 55 24 67 39 33 25 29 48 42 51 8 50 31 21 49 7 30 27 2 38 46 60 22 35 28 10 70 52 47 65 53 3 43 63 37 32 13 59 23 14 56 18 12
Sko72-3	31 14 56 60 18 39 17 67 9 28 26 30 10 70 59 61 62 21 8 50 46 13 63 65 22 64 23 48 33 54 27 16 35 47 53 6 43 37 12 25 69 52 42 24 68 19 66 32 11 45 40 57 44 29 36 4 1 34 71 20 55 41 51 38 49 2 3 5 15 72 7 58
Sko72-4	12 3 56 64 24 72 45 50 36 41 20 11 65 61 15 4 66 43 62 63 37 58 34 69 25 6 53 18 5 19 39 9 68 67 16 54 48 51 42 40 44 57 55 31 29 38 60 46 23 49 22 14 35 33 71 70 13 10 30 7 27 28 1 47 52 26 21 2 32 59 17 8
Sko72-5	51 29 34 40 71 9 44 23 1 7 33 38 55 16 41 58 59 46 65 6 19 15 20 14 49 66 68 61 36 4 57 5 45 69 25 11 24 26 13 43 32 47 52 42 62 63 22 50 37 21 12 2 35 72 17 31 67 64 60 39 48 30 28 54 27 70 10 18 56 3 53 8
Sko81-1	74 66 3 61 30 25 69 67 23 65 50 9 48 56 33 17 4 6 41 21 81 14 15 19 38 49 31 7 39 36 26 70 35 42 22 55 11 57 78 44 2 45 52 32 68 13 62 1 5 27 58 20 18 73 43 59 24 51 37 63 12 79 75 80 77 71 72 29 64 46 28 10 40 34 76 16 53 54 60 47 8
Sko81-2	8 28 77 60 64 57 76 16 47 72 54 62 7 63 27 10 39 20 18 24 43 53 75 71 40 59 73 37 68 13 46 51 78 1 58 5 79 29 80 32 12 45 2 44 19 31 26 34 9 11 4 42 65 30 36 67 23 50 41 70 74 66 33 17 69 3 21 61 15 6 56 48 52 14 81 35 25 49 38 55 22
Sko81-3	47 77 53 16 44 54 10 34 40 76 60 59 63 20 37 51 13 18 64 68 57 21 27 62 24 78 43 7 49 22 45 52 58 69 75 73 1 8 71 32 46 56 4 67 42 23 17 50 65 41 66 81 26 36 80 19 12 33 38 70 3 25 61 2 14 15 31 9 5 6 48 72 28 11 55 29 79 74 39 35 30
Sko81-4	74 66 8 25 26 65 69 23 51 3 14 81 20 50 2 30 61 21 12 28 6 44 9 11 19 17 15 70 48 33 46 39 72 40 34 32 49 60 4 67 37 56 78 42 54 10 18 35 36 55 1 45 52 27 62 58 75 64 47 63 73 31 13 68 57 76 77 24 53 38 16 43 59 7 41 5 29 22 79 71 80
Sko81-5	40 34 54 79 4 22 64 29 38 47 75 65 30 80 35 26 42 60 43 19 18 12 63 37 76 23 33 73 69 77 67 58 51 24 20 59 25 10 49 71 53 14 41 28 81 32 11 46 1 50 17 21 44 16 13 31 57 70 78 39 62 7 45 2 27 68 61 72 55 3 5 9 8 15 6 36 52 48 56 74 66
Sko100-1	3 35 44 29 17 21 7 76 53 45 12 36 50 41 48 61 23 49 2 70 82 92 38 72 81 64 90 66 47 51 46 26 100 69 20 40 43 30 85 99 94 67 73 96 4 8 98 68 56 89 93 24 42 59 19 14 9 84 87 86 52 22 62 16 31 34 75 80 54 55 13 27 10 60 28 57 32 63 5 88 77 25 58 74 95 11 6 15 33 79 39 83 37 71 91 1 97 18 78 65
Sko100-2	33 29 17 99 3 53 12 61 77 41 36 76 48 50 94 38 72 7 26 40 68 8 81 43 14 59 100 70 69 20 46 35 67 45 92 49 51 52 21 2 98 47 66 90 44 86 19 1 82 23 4 56 55 31 28 15 89 16 80 6 57 37 74 87 34 22 79 84 75 13 83 71 63 54 62 96 30 85 9 64 60 95 11 32 25 88 73 18 93 10 65 42 24 39 5 27 58 97 91 78
Sko100-3	44 78 97 35 39 89 24 80 54 21 5 37 71 82 91 31 63 27 13 83 75 11 16 14 9 62 65 73 30 85 99 96 12 93 56 95 10 4 55 74 25 1 18 84 79 57 6 32 77 20 88 34 81 22 15 17 42 98 26 59 23 64 46 7 52 51 67 50 94 29 48 36 38 72 19 86 87 68 66 90 2 43 69 70 41 47 100 61 40 92 53 49 45 76 60 33 8 28 58 3
Sko100-4	49 42 39 79 71 25 32 93 97 18 5 94 52 68 8 98 83 9 16 88 22 33 43 21 27 75 80 24 60 67 86 28 31 74 19 89 54 15 1 56 96 65 4 91 85 55 13 11 78 63 57 62 37 77 59 81 61 50 92 48 90 100 38 46 26 82 69 53 35 72 66 70 36 51 10 40 20 30 47 73 14 41 87 34 64 95 44 29 23 12 17 99 2 76 58 45 84 6 3 7
Sko100-5	76 3 50 48 45 86 72 52 92 38 17 12 23 62 19 69 51 59 61 29 98 47 7 21 35 100 8 44 70 49 66 82 36 40 68 94 53 26 41 46 99 67 20 2 14 87 64 73 34 81 56 93 32 1 77 33 43 22 88 5 79 57 18 91 9 16 89 42 39 83 11 97 71 37 95 15 31 27 24 80 10 75 54 85 28 74 96 65 4 60 84 30 6 58 13 25 63 55 90 78
SRFLP-110-1	8 80 53 83 98 6 49 17 54 100 57 48 2 44 52 29 94 106 41 23 18 34 30 107 78 13 70 32 31 12 105 89 81 61 59 38 45 66 68 85 37 27 103 101 104 72 36 82 10 58 87 63 21 60 96 95 20 51 102 47 25 35 43 42 88 99 97 46 90 26 50 91 71 28 5 64 16 74 77 15 56 84 69 1 93 39 24 22 55 79 7 14 92 108 4 33 110 86 3 109 9 11 65 76 67 75 73 62 19 40
SRFLP-110-2	64 62 1 13 33 11 104 69 74 25 54 16 5 57 86 96 47 55 34 80 14 58 3 70 101 91 75 48 51 9 46 99 56 26 15 88 39 76 77 84 107 2 30 73 7 93 28 22 42 35 92 108 23 100 71 68 4 90 85 94 52 6 72 78 83 82 60 24 49 10 97 17 19 31 63 53 43 66 45 38 61 29 41 20 27 89 44 105 95 81 79 110 50 36 106 59 109 102 98 37 65 103 18 21 67 87 32 40 12 8
SRFLP-110-3	84 20 61 51 27 92 38 33 68 45 34 99 8 25 55 98 35 3 63 80 44 88 22 4 75 30 26 107 11 76 100 23 52 56 14 109 50 6 110 31 93 5 103 69 59 36 72 105 15 57 82 78 37 13 85 79 64 32 28 49 70 43 21 19 39 48 67 77 65 47 2 81 101 54 60 12 66 97 53 96 87 29 74 9 41 18 83 1 46 42 89 108 90 104 106 86 16 73 71 58 40 94 95 102 17 7 91 10 24 62

Referências

Ahonen H., De Alvarenga A.G. e Amaral A. R. S. (2014), Simulated annealing and tabu search approaches for the Corridor Allocation Problem, European Journal of Operational Research, 232, 221-233.

Amaral A. R. S. (2006), On the exact solution of a facility layout problem, European Journal of Operational Research, 173, 508-518.

Amaral A. R. S. (2008), An exact approach for the one-dimensional facility layout problem, Operations Research, 56, 1026-1033.

Amaral A. R. S. (2009), A New Lower Bound for the Single Row Facility Layout Problem, Discrete Applied Mathematics, 157, 183-190.

Amaral A. R. S. e Letchford A. N. (2011), A polyhedral approach to the single row facility layout problem. Technical Report. Department of Management Science, Lancaster University. Disponível em: http://www.optimization-online.org/DB_FILE/2008/03/1931.pdf

Amaral A. R. S. e Letchford A. N. (2013), A polyhedral approach to the single row facility layout problem, Mathematical Programming, 141, 453-477.

Anjos M. F., Kennings A. e Vannelli A. (2005), A semidefinite optimization approach for the single-row layout problem with unequal dimensions, Discrete Optimization, 2, 113-122.

Anjos M. F. e Yen G. (2009), Provably near-optimal solutions for very large single row facility layout problems, Optimization Methods and Software, 24, 805-817.

Cravo G. L., Ribeiro G.M. e Lorena L.A. N. (2008), A greedy randomized adaptive search procedure for the point-feature cartographic label placement, Computer & Geosciences, 34, 373-386.
Datta D., Amaral A. R. S. e Figueira J. R. (2011), Single row facility layout problem using a permutation-based genetic algorithm, European Journal of Operational Research, 213, 388-394.
De Alvarenga A. G., Negreiros-Gomes F. J. e Mestria M. (2000), Metaheuristic methods for a class of the facility layout proble, Journal of Intelligent Manufacturing, 11, 421-430.
Faria H. Jr., Binato S., Resende M. G. C. e Falcão D. J. (2005), Transmission network design by a greedy randomized adaptive path relinking approach, IEEE Trans Power Systems, 20, 43-49.
Feo, T. A. e Resende, M. G. C. (1995), Greedy Randomized Adaptive Search Procedures, Journal of Global Optimization, 6, 109-133.
Heragu, S. S. e Alfa, A. S. (1992), Experimental analysis of simulated annealing based algorithms for the layout problem, European Journal of Operational Research, 57, 190-202.
Heragu S. S. e Kusiak A. (1988), Machine layout problem in flexible manufacturing systems, Operations Research, 36, 258-268.
Heragu, S. S. e Kusiak, A. (1991), Efficient models for the facility layout problem, European Journal Of Operational Research, 53, 1-13.
Hungerländer P. e Rendl F. (2013), A Computational Study for the Single-Row Facility Layout Problem, Computational Optimization and Applications, 55, 1-20.
Kothari R. e Ghosh D. (2012), The Single Row facility Layout Problem: state of the art, OPSEARCH, 49, 442-462.
Kothari, R. e Ghosh, D. (2013a), Tabu Search for the single row facility layout problem using exhaustive 2-opt and insertion neighborhoods, European Journal of Operational Research, 224, 93-100.
Kothari, R. e Ghosh, D. (2013b), Insertion based Lin-Kernighan heuristic for single row facility layout, Computers & Operations Research, 40, 129-136.
Kothari, R. e Ghosh, D. (2014a), An efficient genetic algorithm for single row facility layout, Optimization Letters, 8, 679-690.
Kothari, R. e Ghosh, D. (2014b). A Scatter search algorithms for the single row facility layout problem, Journal of Heuristics, 20, 125-142.
Kumar K. R., Hadjinicola G.C. e Lin T.L. (1995), A heuristic procedure for the single-row facility layout problem, European Journal of Operation Research, 87, 65-73.
Kumar, S., Asokan, P., Kumanan, S. e Varma, B. (2008), Scatter search algorithm for single row layout problem in FMS, Advances in Production Engineering and Management, 3, 193-204.
Love, R. F. e Wong, J. Y. (1976), On solving a one-dimensional space allocation problem with integer programming, INFOR, 14, 139-144.
Ozcelik, F. (2011), A hybrid genetic algorithm for the single row layout problem, International Journal of Production Research , 50, 5872-5886.
Picard J. e Queyranne M. (1981), On the one dimensional space allocation problem, Journal of Operation Research, 29, 371-391.
Resende, M. G. C. e Ribeiro, C. C., GRASP with path-relinking: recent advances and applications. In: Ibaraki, T., Nonobe, K. and Yagiura, M. (Ed.). Metaheuristics: Progress as Real Problem Solvers. Kluwer Academic Publishers, 2005, 29-63.
Samarghandi H. e Eshghi K. (2010), An Efficient Tabu Algorithm for the Single Row Facility Layout Problem, European Journal of Operational Research, 205, 98-105.
Samarghandi H., Taabayanb P. e Jahantighc F. F. (2010), A particle swarm optimization for the single row facility layout problem, Computers & Industrial Engineering, 58, 529–534.
Simmons D. M. (1969), One-dimensional space allocation: An ordering algorithm, Operations Research, 17, 812-826.
Solimanpur M., Vrat P. e Shankar R. (2005), An Ant Algorithm for the Single Row Layout Problem in Flexible Manufacturing Systems, Computers & Operations Research, 32, 583-598.