

Proposta de um algoritmo *Hill-Climbing* para solucionar o problema de arranjo físico com múltiplos corredores

Mayk Masayuki Yano Taniyama (UTFPR-LD) maykmasayuki002@gmail.com

Flavio Oliveira Tavares (UTFPR-LD) flaviootavares@hotmail.com

Rafael Henrique Palma Lima (UTFPR-LD) rafaelhlma@utfpr.edu.br

Resumo

O arranjo físico de instalações produtivas é um problema de grande importância para as organizações, pois os custos e o tempo envolvidos para reorganizar arranjos mal formulados geralmente são altos. Diversos problemas reais de *layout* podem ser caracterizados por diversas facilidades que precisam ser dispostas em uma quantidade de linhas a fim de minimizar o fluxo ponderado total entre elas. O objetivo deste trabalho é propor um algoritmo *Hill-Climbing* para solucionar esse problema, conhecido como MRFLP (*Multi-Row Facility Layout Problem*). O algoritmo proposto utiliza uma estratégia tendenciosa para construir soluções iniciais e emprega duas estruturas de vizinhança durante a busca local. Instâncias de *benchmark* para 2 e 3 linhas foram utilizadas para testar o algoritmo, com soluções iguais às melhores conhecidas sendo encontradas para quase todas as instâncias em um curto espaço de tempo.

Palavras chave: Problema de alocação de corredores. *Hill-Climbing*. Arranjo físico de facilidades.

Proposal of a Hill-Climbing algorithm to solve the multi-row facility layout problem

Abstract

Making decisions concerning the layout of production facilities is a problem of great importance to organizations because the costs and time involved to reorganize poorly formulated arrangements are usually high. Many industrial layout problems can be characterized by various facilities that need to be arranged in a number of rows in order to minimize the total weighted flow between them. The objective of this work is to propose a Hill-Climbing algorithm to solve this problem, known as MRFLP (*Multi-Row Facility Layout Problem*). The proposed algorithm uses a biased strategy to construct initial solutions and employs two neighborhood structures during the local search phase. Benchmark instances for 2 and 3 lines were used to test the algorithm. Almost all solutions found were equal to the best-known values reported in the literature and the computer time required to find them was very low.

Key-words: Corridor allocation problem. Hill-climbing. Facility layout.

1. Introdução

O *Multi-Row Facility Layout Problem* (MRFLP) consiste em escolher um determinado arranjo para facilidades retangulares, organizadas em linhas, de modo que o custo de comunicação total entre as facilidades seja minimizado (HUNGERLÄNDER E ANJOS, 2015).

Um exemplo simples deste problema seria uma escola em que as disciplinas são ministradas em salas fixas e os discentes se dirigem até elas. Dependendo da grade de horários, haverá um fluxo maior de deslocamento de alunos entre alguns pares de disciplinas em relação à outras. Neste caso, as salas seriam as facilidades, que estão organizadas em duas linhas paralelas ao corredor, e o custo de comunicação total seria a distância total percorrida por todos os discentes ao longo do dia.

Este problema também é conhecido na literatura científica como *Corridor Allocation Problem* (CAP), sendo o MRFLP uma generalização do CAP. O problema de alocação de corredores tradicional lida com problemas onde há apenas duas linhas para arranjar as facilidades e não há espaços adjacentes entre elas. O termo MRFLP é utilizado quando o problema trata de mais de duas linhas de organização de facilidades, podendo ou não haver espaços adjacentes.

Hungerländer e Anjos (2015) explicam a importância do MRFLP nas indústrias, em sistemas de manufatura flexíveis (FMSs), onde a produção é automatizada justamente por prezar pela tempo e custo de produção. Nesses sistemas, as máquinas de produção são alocadas em linhas e utilizam robôs como *gantry robots*, veículos guiados automatizados (AGVs), entre outros, para funcionar em conjunto com elas. Nesse contexto o layout das máquinas pode afetar significativamente a eficiência dos processos.

Este artigo tratará de um caso especial do MRFLP, que é o *Space-Free Multi Row Facility Layout Problem* (SF-MRFLP). O SF-MRFLP engloba problemas com múltiplas linhas para arranjo das facilidades, porém sem a possibilidade de existir espaços livres entre facilidades adjacentes. Vale ressaltar também que as facilidades são retangulares e possuem comprimentos diferentes e a largura, tanto do corredor quanto da facilidade, será desprezada nesta abordagem. Porém, há abordagens na literatura como a de Lee e Lee (2002) que resolveram o MRFLP com facilidades de áreas heterogêneas e consideraram as larguras, tendo utilizado uma hibridização de Algoritmo genético, *Simulated Annealing* e *Tabu Search* para resolução do problema.

O objetivo deste trabalho é propor um algoritmo baseado na abordagem Hill-Climbing com vizinhança dupla para resolução do SF-MRFLP e verificar sua eficiência utilizando instâncias de *benchmark* comumente referenciadas na literatura, comparando os resultados obtidos com aqueles obtidos por outros pesquisadores.

O restante deste artigo está estruturado da seguinte maneira. A Seção 2 apresenta o referencial teórico e a Seção 3 descreve o algoritmo proposto. A Seção 4 apresenta os resultados numéricos obtidos pelo algoritmo e a Seção 5 faz as considerações finais do trabalho.

2. Referencial teórico

2.1. Formulação do SF-MRFLP

Uma instância do SF-MRFLP consiste em n facilidades unidimensionais, com comprimentos positivos, emparelhados, sem espaços adjacentes, organizados em k linhas, sendo o objetivo encontrar o arranjo de facilidades que minimiza a soma ponderada das distâncias entre os pontos centrais de todos os pares de facilidades vezes o fluxo de comunicação do par (HUNGERLÄNDER E ANJOS, 2012). A Figura 1 apresenta um exemplo de solução para uma instância de tamanho $n = 10$ usando 3 linhas.

1 12	2 12	3 10	4 16
5 16	6 20	7 6	
8 14	9 16	10 6	

Figura 1 – Exemplo de layout com 3 linhas e diferentes comprimentos de facilidades

Nesse problema o fluxo de comunicação poderia representar a quantidade movimentada de pessoas ou materiais entre cada par de facilidades. A Figura 2 apresenta um exemplo para uma instância com 10 facilidades.

	1	2	3	4	5	6	7	8	9	10
1	0	1	2	3	5	6	8	9	7	8
2	1	0	2	4	5	6	2	1	5	2
3	2	2	0	5	2	3	5	4	3	5
4	3	4	5	0	4	5	3	5	2	1
5	5	5	2	4	0	1	2	3	5	2
6	6	6	3	5	1	0	1	2	3	5
7	8	2	5	3	2	1	0	1	6	8
8	9	1	4	5	3	2	1	0	5	2
9	7	5	3	2	5	3	6	5	0	3
10	8	2	5	1	2	5	8	2	3	0

Figura 2 – Exemplo de matriz contendo os fluxos de comunicação entre as facilidades

2.2. Métodos de resolução do MRFLP

A literatura especializada sobre o MRFLP possui vários trabalhos publicados que propõem heurísticas e metaheurísticas para obter boas soluções para o problema usando pouco tempo computacional. A seguir serão citadas algumas dessas contribuições, enfatizando as técnicas de resolução utilizadas pelos autores.

Hungerländer e Anjos (2012) propuseram um modelo de relaxação semidefinida para resolver o SF-MRFLP através de heurísticas. Por outro lado, Amaral e Cellin (2016) resolveram o CAP, MRFLP e o SF-MRFLP utilizando um algoritmo *Simulated Annealing*, um algoritmo que se baseia na teoria termodinâmica para a otimização de soluções, e outro híbrido de *Variable Neighborhood Search* e *Iterated Local Search*, que são algoritmos que varia as estruturas de vizinhanças, no intuito de não estagnar a otimização da solução, e perturba a solução, explorando outros espaços de soluções a fim de encontrar melhora na solução, respectivamente, e obtiveram resultados equiparáveis à heurística SPD aplicada por Hungerländer e Anjos (2015) com tempo de execução muito menor.

Ahonen, Alvarenga e Amaral (2014) resolveram o CAP utilizando os algoritmos *Tabu Search*, um algoritmo que permite uma piora momentânea da solução no intuito de explorar outros espaços de soluções e conseguir melhora, e *Simulated Annealing*.

Santana e Santos (2016) resolveram o CAP utilizando um algoritmo *Iterated Local Search* e obtiveram resultados próximos aos algoritmos *Tabu Search* e *Simulated Annealing* aplicado por Ahonen (2014), porém, com um tempo de execução muito menor.

Lee e Lee (2002) utilizaram uma hibridização de Algoritmo genético, um algoritmo que se baseia na teoria da evolução humana de Darwin, gerando populações de soluções e as melhorando ao passar das “gerações”, *Simulated Annealing* e *Tabu Search* para resolver um MRFLP com facilidades de áreas heterogêneas.

Para resolução do CAP, Gosh e Kothari (2012) utilizaram um algoritmo genético híbrido com busca local e outro *Scatter Search*, um algoritmo que orienta uma exploração em conjunto de soluções, chamados de referências, criados por outros métodos, através de uma técnica evolutiva, de modo a encontrar conjuntos melhores, híbrido com *Path Relinking*, que intensifica a busca explorando em espaços não explorados pelo *Scatter Search*. Os resultados computacionais mostraram que ambos obtiveram resultados semelhantes, porém, o algoritmo híbrido de *Scatter Search* e *Path Relinking* obteve um tempo menor.

Anjos e Yen (2009) propuseram uma nova formulação baseada em matriz que produza um novo relaxamento semidefinido com menos restrições lineares, mas ainda produzindo limites inferiores globais de alta qualidade, para resolução do *Single Row Facility Layout Problem* (SRFLP), problema similar ao CAP mas que trata apenas com uma única linha de organização, com até 100 facilidades.

3. Algoritmo proposto

O *Hill-Climbing* é uma meta-heurística que realiza buscas locais a partir de uma solução inicial, com o objetivo de encontrar o ótimo local usando estruturas de vizinhança. Uma das principais deficiências do *Hillclimbing* é o fato de que ao chegar em ótimos locais, o método não prossegue na busca por melhorias (VIEIRA, MARQUES E CASTRO, 2015). Para superar essa limitação, o algoritmo desenvolvido neste artigo utiliza duas estruturas de vizinhança sequenciais diferentes, que são geradas à cada iteração de busca, visando o aumento do conjunto de vizinhança para encontrar soluções ainda melhores.

Pode se verificar semelhança entre o algoritmo proposto neste artigo e um VNS (*Variable Neighborhood Search*) devido à utilização de mais de uma vizinhança. Porém, a diferença está no fato de que o VNS realizaria as iterações de buscas por melhorias e trocaria de estrutura de geração de vizinhança somente se estagnasse em um ótimo local, enquanto que, no proposto neste artigo, o algoritmo verificaria a possibilidade de troca à cada iteração, pois como as duas vizinhanças são geradas simultaneamente, há a possibilidade de comparação e utilização da que obteve a melhor melhoria.

3.1. Representação da solução

As facilidades serão enumeradas por número inteiros diferentes de zero, conforme ilustra a Figura 3. Dessa forma, a ordem da solução será dada por um vetor unidimensional que conterá estes números, representando as facilidades. A quantidades de facilidades em cada linha serão

agrupadas em um segundo vetor unidimensional. Dessa forma, as facilidades serão designadas para cada linha na ordem da solução contida no primeiro vetor.

Ordem	1	2	3	4	5	6	7	8	9	10
Cortes	4	3	3							

Figura 3 – Exemplo de representação do layout da Figura 1

3.2. Estruturas de vizinhanças

O algoritmo utilizará duas estruturas de vizinhanças para explorar o espaço de soluções próximo à uma dada solução. A primeira estrutura de vizinhança consiste em realizar todas as trocas possíveis entre pares de facilidades. A quantidade de facilidades em cada linha se mantém inalterado. A Figura 4 mostra o funcionamento dessa estrutura de vizinhança.

A segunda vizinhança, ilustrada na Figura 5, consiste em realizar o aumento e a diminuição, de uma única unidade de facilidade, de um par de linhas para todos os pares possíveis, além de realizar o inverso, diminuindo e aumentando. A posição da facilidade em relação à ordem da solução se mantém nesta estrutura.

Ordem	2	1	3	4	5	6	7	8	9	10
Cortes	4	3	3							

Ordem	1	3	2	4	5	6	7	8	9	10
Cortes	4	3	3							

Ordem	1	2	4	3	5	6	7	8	9	10
Cortes	4	3	3							

Ordem	1	2	3	5	4	6	7	8	9	10
Cortes	4	3	3							

Ordem	1	2	3	4	6	5	7	8	9	10
Cortes	4	3	3							

Ordem	1	2	3	4	5	7	6	8	9	10
Cortes	4	3	3							

Ordem	1	2	3	4	5	6	8	7	9	10
Cortes	4	3	3							

Ordem	1	2	3	4	5	6	7	9	8	10
Cortes	4	3	3							

Ordem	1	2	3	4	5	6	7	8	10	9
Cortes	4	3	3							

Figura 4 – Exemplo de soluções vizinhas geradas pela primeira estrutura de vizinhança

Ordem	1	2	3	4	5	6	7	8	9	10
Cortes	5	2	3							

Ordem	1	2	3	4	5	6	7	8	9	10
Cortes	4	4	2							

Ordem	1	2	3	4	5	6	7	8	9	10
Cortes	3	4	3							

Ordem	1	2	3	4	5	6	7	8	9	10
Cortes	4	2	4							

Figura 5 – Exemplo de soluções vizinhas geradas pela segunda estrutura de vizinhança

3.3. Método de geração da solução inicial

Utilizou-se um método de geração de solução aleatória, porém tendenciosa, dando probabilidade maior para que facilidades com maior fluxo de comunicação entre si ficassem mais próximas logo na primeira solução.

Primeiramente, designa-se uma facilidade aleatória para a primeira posição da primeira linha. A partir disto, escolhesse aleatoriamente a facilidade a ser alocada na primeira posição da segunda linha, porém, dando uma probabilidade maior para as facilidades que tiverem um fluxo de comunicação maior, respectivamente, com esta primeira facilidade. O processo se repete para terceira linha, porém, utilizando agora a facilidade escolhida para segunda linha como referência e assim sucessivamente.

Definido as facilidades em cada linha, insere-se o restante das facilidades: Escolhe-se uma linha aleatoriamente, verifica-se qual foi a última facilidade inserida e insere-se uma facilidade aleatoriamente, porém, dando uma probabilidade maior para as facilidades que tiverem um fluxo de comunicação maior, respectivamente, com esta facilidade já inserida na linha. O processo se repete até que todas as facilidades sejam alocadas.

Algoritmo 1 Heurística para solução inicial

- 1: v =facilidades do layout
- 2: m =fluxo de comunicação entre as facilidades
- 3: Escolhe uma facilidade aleatória f contida no vetor v e designa à segunda coluna da primeira linha de s'
- 4: $s'(0,1)=1$
- 5: $i=2$
- 6: **Para** i **até** número de linhas do layout **faça**
- 7: Escolhe uma facilidade aleatória f' contida em v que ainda não foi escolhida, dando maior probabilidade para facilidades que tiverem maior fluxo de comunicações com f , e designa à segunda coluna da linha i de s'
- 8: $s'(0,i)=1$
- 9: $f=f'$
- 10: **Fim Para**
- 11: n' =número de facilidades- número de linhas do layout
- 12: $n=1$
- 13: **Para** n **até** n' **faça**
- 14: Escolhe uma linha aleatória L
- 15: Escolhe uma facilidade aleatória contida em v que ainda não foi escolhida, dando maior probabilidade para facilidades que tiverem maior fluxo de comunicação com $s'(1,L)$, e designa para $s'(s'(0,L)+1,L)$
- 16: $s'(0,L)=s'(0,L)+1$
- 17: **Fim Para**
- 18: $n''=1$
- 19: $n'''=1$
- 20: **Para** n'' **até** número de linhas do layout **faça**
- 21: $s(n'',1)=s'(0, n'')$
- 22: **Para** n''' **até** $(n''' + s'(0, n'')-1)$ **faça**
- 23: $s(n''',0)=s'(n''',n'')$

24: **Fim Para**
 25: $n''' = n''' + s'(0, n'')$
 26: **Fim Para**

3.4. Avaliação da solução

A avaliação da solução é realizada analisando o custo de comunicação total entre as facilidades:

Custo de comunicação total entre n facilidades

$$= \sum_{i=1}^{n-1} \sum_{j=i+1}^n \text{Distância entre as facilidades } i \text{ e } j \times \text{Fluxo de comunicação entre } i \text{ e } j$$

Vale destacar que a distância utilizada foi apenas as distâncias entre os centros de i e j no eixo das abscissas em um plano cartesiano, levando em consideração que as linhas estão em paralelo com este eixo e começam no ponto zero. Esta distância foi utilizada para que fosse possível utilizar as mesmas instâncias utilizadas por Amaral e Cellin (2016) e realizar uma comparação entre os resultados computacionais. Utilizando esta distância, verifica se que as larguras dos corredores são desprezadas, porém Hungerländer e Anjos (2015) utilizam a mesma distância, sendo um método aceitável de relaxamento que aumenta consideravelmente a velocidade das iterações do algoritmo.

Vale lembrar que as dimensões das facilidades, a quantidade de linhas de organização e o fluxo de comunicação entre as facilidades são conhecidos, sendo parâmetros do problema. A Figura 6 apresenta um exemplo de cálculo da distância entre as facilidades 1 e 9. O custo total da solução na Figura 6, considerando as distâncias e os fluxos entre todos os pares de facilidades é igual a 2603.

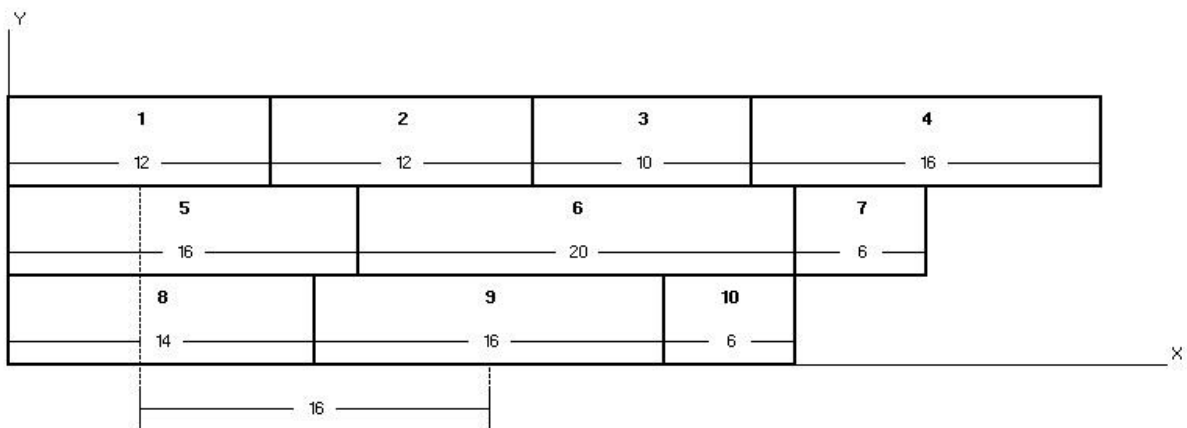


Figura 6 – Exemplo do cálculo da distância entre as facilidades 1 e 9

3.5. Síntese do algoritmo

A partir de uma dada solução inicial, que é considerada uma solução global neste momento, o algoritmo gera um conjunto de soluções vizinhas utilizando as duas estruturas de vizinhança, agrupando em um único conjunto para tal geração, avaliará todo o conjunto, verificando se há alguma solução com melhoria, e substitui a solução global pela mesma caso haja melhoria. O processo se repete até que o conjunto de vizinhança não represente uma melhoria, o que

caracteriza o ótimo local e encerra a iteração. Sendo assim, dado uma quantidade i_{max} de iterações desejada, o algoritmo gerará i_{max} soluções iniciais e realizará a busca local, até a estagnação da otimização, em todas, substituindo a solução global de melhoria somente ao se deparar com uma solução melhor.

Algoritmo 2 Hillclimbing

```
1: Para  $i=1$  até  $i_{max}$  faça
2:   Gera uma solução aleatória  $s$ 
3:   Gera soluções vizinhas de  $s$  utilizando as 2 estruturas de vizinhança
4:   Avalia as soluções vizinhas e retorna uma solução candidata  $s'$ 
5:   Enquanto  $s' < s$  então
6:      $s = s'$ 
7:   Se  $s' \geq s$ 
8:     Fim Enquanto
9:   Fim Se
10: Fim Enquanto
11: Fim Para
```

4. Resultados computacionais

O algoritmo foi implementado na linguagem VBA e os teste foram realizados em um computador com processador Intel Core i5 1.7 GHz com 4 GB de memória RAM e sistema operacional Windows 8.1 PRO 64 bits.

Os resultados computacionais obtidos foram comparados com resultados obtidos por Gosh e Kothari (2012) e Cellin e Amaral (2016), que por sua vez compararam com resultados de outros artigos, utilizando instâncias cujas fontes estão devidamente identificadas nas Tabelas 1 e 2 e são disponibilizadas na FLPLIB por Anjos (2018).

A coluna de tempo médio foi calculada considerando o tempo que o algoritmo levou até chegar ao resultado exato, ou aproximado, do resultado encontrado nos dois artigos utilizados como comparativos. O algoritmo foi executado 5 vezes para cada instância. Para os resultados aproximados, também foi calculado a média dos resultados obtidos, pois os mesmo se divergiam entre si nas 5 buscas realizadas.

As numerações na coluna “Fonte” e da coluna “Melhor conhecido”, das Tabelas 1 e 2 correspondem à referência onde foi encontrada a informação: [1]: Simmons (1969); [2]: Heragu e Kusiak (1991) ; [3]: Hungerländer e Anjos (2015); [4]: Amaral (2008); [5]: Anjos e Vanelli (2008); [6]: Gosh e Kothari (2012); [7]: Cellin e Amaral (2016).

A Tabela 1 apresenta os resultados obtidos para problemas em que as facilidades devem ser organizadas em 2 linhas. Por sua vez, a Tabela 2 mostra os resultados obtidos para problemas com 3 linhas.

Instância	Fonte	Tamanho (n)	Melhor conhecido	Resultado	Gap(%)	Tempo (s)
S_8	[1]	8	408 [7]	408	0%	0,035
SH_8	[1]	8	1135,5 [7]	1135,5	0%	0,017
S9	[1]	9	1181,5 [6,7]	1181,5	0%	0,039
SH_9	[1]	9	2294,5 [6,7]	2294,5	0%	0,023
S10	[1]	10	1374,5 [6,7]	1374,5	0%	0,055
S11	[1]	11	3439,5 [6,7]	3439,5	0%	0,031
H_5	[2]	5	450 [7]	450	0%	0,005
H_6	[2]	6	720 [7]	720	0%	0,007
H_7	[2]	7	1700 [7]	1700	0%	0,011
H_8	[2]	8	238,5 [7]	238,5	0%	0,051
Rand_5	[3]	5	52,5 [7]	52,5	0%	0,006
Rand_6	[3]	6	190,5 [7]	190,5	0%	0,009
Rand_7	[3]	7	166 [7]	166	0%	0,017
Rand_8	[3]	8	205 [7]	205	0%	0,014
Rand_9	[3]	9	492,5 [7]	492,5	0%	0,023
Rand_10	[3]	10	838 [7]	838	0%	0,035
P15	[4]	15	3195 [6]	3195	0%	20,188
N-30-05	[5]	30	57400 [6]	57516	0,2%	196,133

Tabela 1 – Resultados obtidos para layouts com 2 linhas de organização

Instância	Fonte	Tamanho (n)	Melhor conhecido	Resultado	Gap(%)	Tempo (s)
S_8	[1]	8	260 [7]	260	0%	0,023
SH_8	[1]	8	740,5 [7]	740,5	0%	0,015
S_9	[1]	9	771,5 [7]	771,5	0%	0,035
SH_9	[1]	9	1431,5 [7]	1431,5	0%	0,017

H_5	[2]	5	290	[7]	290	0%	0,005
H_6	[2]	6	560	[7]	560	0%	0,011
H_7	[2]	7	1190	[7]	1190	0%	0,008
H_8	[2]	8	1515	[7]	1515	0%	0,022
Rand_5	[3]	5	34,5	[7]	34,5	0%	0,005
Rand_6	[3]	6	110,5	[7]	110,5	0%	0,006
Rand_7	[3]	7	108	[7]	108	0%	0,008
Rand_8	[3]	8	138	[7]	138	0%	0,014
Rand_9	[3]	9	317,5	[7]	317,5	0%	0,028

Tabela 2 – Resultados obtidos para layouts com 3 linhas de organização

Com base nos resultados obtidos, é possível concluir que o algoritmo desenvolvido conseguiu encontrar soluções de alta qualidade com baixo tempo computacional. Em apenas uma das instâncias o algoritmo não encontrou a melhor solução existente, mas mesmo assim o gap foi baixo.

5. Conclusão

Este artigo propôs um algoritmo *Hill-Climbing* para solucionar o MRFLP. O algoritmo usa um método aleatório tendencioso para construção de soluções iniciais e duas estruturas de vizinhança para melhorar a exploração do espaço de soluções durante a busca local.

O baixo *gap* encontrado nos testes computacionais, que na realidade foi igual a zero para maioria das instâncias, comprova o bom desempenho do algoritmo criado em relação aos propostos em outros artigos. A proposta deste artigo foi a criação de um algoritmo *Hill-Climbing* com estrutura de vizinhança dupla no intuito de superar a deficiência de um *Hill-Climbing* tradicional, que geralmente utiliza apenas uma estrutura de vizinhança na busca por melhorias na solução corrente.

A utilização dessa estrutura de vizinhança dupla demonstrou-se capaz de evitar a estagnação precoce na busca local e permitiu que as soluções ótimas do problema fossem encontradas rapidamente. Assim, os resultados obtidos pelo algoritmo aqui proposto são equiparáveis ao de meta-heurísticas que também possuem estratégias mais sofisticadas de busca, como o algoritmos genéticos, *Scatter Search*, *Simulated Annealing* e o algoritmo híbrido de VNS e ILS desenvolvido por Cellin e Amaral (2016).

Para trabalhos futuros, sugere-se o desenvolvimento de novos algoritmos que utilizam o método de geração de soluções iniciais e as estruturas de vizinhança descritos neste artigo, porém em outras metaheurísticas como o *Iterated Local Search* (ILS), *Variable Neighborhood Search* (VNS) ou algoritmos genéticos. Além disso, recomenda-se buscar na literatura instâncias maiores para testar o desempenho de novos algoritmos em instâncias mais complexas.

Referências

ANJOS, M. *FLPLIB: Facility Layout Database*. Disponível em: <<https://www.miguelanjós.com/flplib>>, Acesso em: 06/08/2018.

AHONEN, H., DE ALVARENGA, A., E AMARAL, A. *Simulated Annealing and Tabu Search Approaches for the Corridor Allocation Problem*. *European Journal of Operational Research*, 232(1):221 – 233. ISSN 0377-2217, 2014.

AMARAL, A. R. S. *An Exact Approach to the One-Dimensional Facility Layout Problem*. *Operations Research*, 56(4):1026–1033, 2008.

ANJOS, M. F. AND YEN, G. *Provably Near-Optimal Solutions for Very Large Single-row Facility Layout Problems*. *Optimization Methods and Software*, 24(4-5):805–817, 2009.

CELLIN, B. D. P. E AMARAL, A. R. S. *Um Algoritmo Simulated Annealing e um Híbrido VNS/ILS para Resolver o Problema de Layout em Múltiplas Linhas*. XLVIII Simpósio Brasileiro de Pesquisa Operacional, 2016.

GHOSH, D. E KOTHARI, R. *Population Heuristics for the Corridor Allocation Problem*. *Indian Institute of Management*, p. 1–13, 2012.

HERAGU, S. S. E KUSIAK, A. *Efficient Models for the Facility Layout Problem*. *European Journal of Operational Research*, 53(1), 1–13, 1991.

HUNGERLÄNDER, P. E ANJOS, M. F. *A Semidefinite Optimization-Based Approach for Global Optimization of Multi-Row Facility Layout*. *European Journal of Operational Research*, 245(1):46 – 61. ISSN 0377-2217, 2015.

HUNGERLÄNDER, P. E ANJOS, M.F. *A Semidefinite Optimization Approach to Space-Free Multi-Row Facility Layout*. *Group for Research in Decision Analysis*, 2012.

LEE, Y. H. E LEE, M. H. *A Shape-Based Block Layout Approach to Facility Layout Problems Using Hybrid Genetic Algorithm*. *Computers and Industrial Engineering*, 42(2-4):237 – 248. ISSN 0360-8352, 2002.

SANTANA, C. A. E SANTOS A. G. *Uma Heurística Baseada na Busca Local Iterada para o Problema de Alocação de Corredor*. XLVIII Simpósio Brasileiro de Pesquisa Operacional, 2016.

SIMMONS, D. M. *One-Dimensional Space Allocation: An Ordering Algorithm*. *Operations Research*, 17(5):812–826, 1969.

VIEIRA A. H., MARQUES R. S. E CASTRO P. A. L. *Otimização Paramétrica Através da Busca Hill Climbing Distribuída em Sistemas Multiagentes*. *Computer on the Beach*, p. 161-168, 2015.