SIMULATED ANNEALING APLICADO AO PROBLEMA DE ORDENAÇÃO EM LINHAS PARALAELAS

Bernardo De Polli Cellin
Mestrando em Informática - Universidade Federal do Espírito Santo
Av. Fernando Ferrari, Nº 514, CEP: 9060-900, Vitória, Espírito Santo
bcellin@hotmail.com

André Renato Sales Amaral
Programa de Pós-Graduação em Informática - Universidade Federal do Espírito Santo
Av. Fernando Ferrari, Nº 514, CEP: 9060-900, Vitória, Espírito Santo
amaral@inf.ufes.br

RESUMO

Problemas de layout de facilidade são NP-difíceis e possuem diversas aplicações na indústria. Um desses problemas é o Problema de Ordenação em Linhas Paralelas (PROP), no qual alguns departamentos que possuem características comuns devem ser atribuídos a uma mesma linha, enquanto os restantes são atribuídos a uma linha paralela. O objetivo do problema é encontrar uma ordenação de departamentos para cada linha a fim de minimizar o custo de fluxo de materiais entre todos os departamentos. Um algoritmo Simulated Annealing é proposto para o problema. Resultados computacionais para instâncias da literatura demonstram a eficiência do algoritmo.

PALAVRAS CHAVE. Ordenação em Linhas Paralelas, Layout de Facilidades, Simulated Annealing.

Áreas principais: Meta-heurísticas, Otimização combinatória.

ABSTRACT

Facility layout problems are NP-hard and have many applications in industry. One such problem is the Parallel Row Ordering Problem (PROP), in which some departments that have common features must be assigned to the same line, while the rest are assigned to a parallel line. The objective is to find an ordering of departments for each row in order to minimize the cost of material flow across all departments. A simulated annealing algorithm is proposed for the PROP. Computational results on instances from the literature demonstrate the algorithm efficiency.

KEYWORDS. Parallel Row Ordering Problem, Facility layout, Simulated Annealing.

Main areas: Metaheuristics, Combinatorial Optimization.

1. Introdução

O problema de ordenação em *k* linhas paralelas, ou *k*-Paralel Row Ordering Problem (*k*-PROP), considera um conjunto de *n* departamentos que estão distribuídos entre *k* linhas paralelas. Sabe-se previamente qual linha um determinado departamento irá ocupar, mas não se sabe a sua posição na linha. Portanto, os departamentos que ocupam uma dada linha não mudam, mas a sua ordenação na linha muda. Em cada linha *k*, o arranjo de departamentos deve começar a partir de um ponto comum e nenhum espaço é permitido entre dois departamentos adjacentes. Deseja-se ordenar os departamentos em cada linha de forma que uma função de custo seja minimizada. Quando o número de linhas *k* é igual a dois, chama-se o problema, simplesmente, de PROP (Amaral, 2013).

Entre as aplicações do *k*-PROP estão o arranjo de departamentos ao longo de duas ou mais linhas paralelas no chão de fábrica, a construção de edifícios com vários andares (Amaral, 2013) e o layout de máquinas em sistemas de manufatura flexível (FMS) (Hungerländer e Anjos, 2015).

O *k*-PROP é uma generalização do problema de ordenação de departamentos em linha única, ou Single Row Facility Layout Problem (SRFLP). Pesquisas sobre o SRFLP têm utilizado abordagens exatas e heurísticas.

Exemplos de abordagens exatas são: Amaral (2006) que apresentou um modelo de Programação Inteira Mista, ou Mixed Integer Programming (MIP), Amaral (2008) que propôs um modelo de programação linear 0-1 mista que é mais eficiente do que o MIP publicado anteriormente; Anjos e Vanelli (2008) que demonstraram que uma combinação de uma relaxação de programação semidefinida com planos de corte pode gerar layouts ótimos globais para instâncias com até 30 departamentos; Amaral(2009) que apresentou um algoritmo de planos de corte para o problema de layout, encontrando layouts ótimos para instâncias com 35 departamentos em tempo computacional razoável e Hungerlander e Hendl (2011) que usaram programação semidefinida e conseguiram resolver o SRFLP para instâncias com cerca de 40 departamentos.

Abordagens heurísticas têm sido usadas para o SRFLP. De Alvarenga et al. (2000) propuseram metaheurísticas baseadas em busca tabu e Simulated Annealing (SA); Datta et al. (2011) propuseram um Algoritmo Genético para o problema, conseguindo melhorar várias soluções tidas como melhores até o momento; e Kothari e Gosh (2013) apresentaram implementações eficientes para a busca tabu, conseguindo melhorar soluções em 23 de 43 instâncias testadas.

No caso do *k*-PROP, Amaral (2013) propôs uma modelagem MIP para o problema, a qual estende sua formulação MIP feita para o SRFL (Amaral, 2006) considerando que o layout de cada linha inicia-se de um ponto comum e que não há espaços entre os departamentos.

Hungerländer e Anjos (2015) usaram uma modelagem de relaxação semidefinida para o *k*-PROP com espaços entre os departamentos e com balanceamento do número de departamentos atribuídos a cada linha do layout.

Problemas de layout de facilidades são NP-difíceis (Amaral, 2013). Portanto, meta-heurísticas são utilizadas para obter boas soluções em tempo computacional viável. Aqui um algoritmo baseado no Simulated Annealing é proposto para resolver o PROP ($k = 2$).

Este artigo está organizado como segue. A Seção 2 mostra a formulação do PROP. A Seção 3 descreve o algoritmo Simulated Annealing proposto. A Seção 4 mostra os resultados obtidos. Por fim, a Seção 5 contém as considerações finais.

2. Formulação do PROP

No caso particular do *k*-PROP com $k = 2$, tem-se um conjunto $N = \{1, \dots, n\}$ de departamentos, dos quais *t* departamentos deverão ser arranjados na primeira linha e os outros (n-t) departamentos em uma segunda linha, paralela à primeira, formando os conjuntos $N_1 = \{1, \dots, t\}$ e $N_2 = \{t + 1, \dots, n\}$. Seja P_1 o conjunto de todas as permutações π_1 de N_1 e P_2 o conjunto de todas as permutações π_2 de N_2, então o PROP pode ser definido como:

$$\min_{\pi_1 \in P_1, \pi_2 \in P_2} \left\{ \left(\sum_{i,j \in N_1, i<j} c_{ij} d_{ij}^{\pi_1} \right) + \left(\sum_{i,j \in N_2, i<j} c_{ij} d_{ij}^{\pi_2} \right) + \left(\sum_{i \in N_1, j \in N_2, i<j} c_{ij} d_{ij}^{\pi_1, \pi_2} \right) \right\}$$

no qual c_{ij} é a interação, ou fluxo de materiais, entre os departamentos i e j, e d_{ij} é a distância de centro para centro entre os departamentos i e j ($i, j \in N$, $i<j$). No cálculo da distância d_{ij} somente a distância horizontal (ao longo do eixo x) é considerada, como pode ser observado na Figura 1.

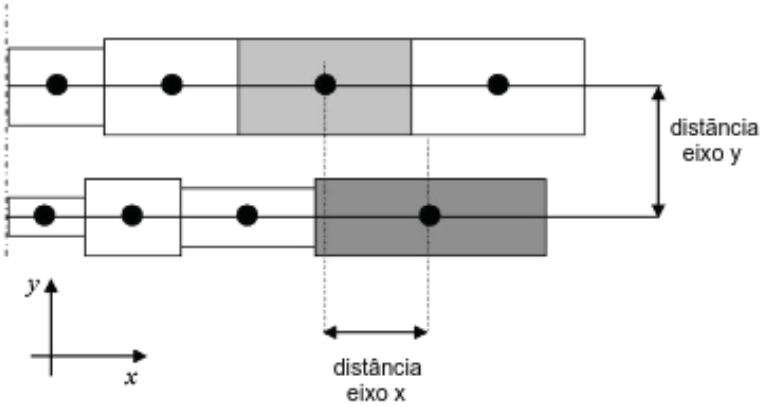

Figura 1. Cálculo da distância entre departamentos – adaptado de Amaral (2013)

3. Simulated Annealing

O Simulated Annealing (SA) é uma meta-heurística usada para resolver problemas de otimização combinatória. Esse método foi proposto originalmente por Kirkpatrick, Gellat e Vecchi (1983) e baseia-se em uma analogia com a teoria do resfriamento de materiais da termodinâmica para otimizar soluções para esses problemas.

Este método faz pequenas mudanças na solução iterativamente, buscando melhores soluções vizinhas. Uma característica notável do SA é o fato de que ele pode aceitar soluções piores para poder fugir de mínimos locais, buscando mínimos globais em iterações posteriores. Os parâmetros de entrada para o algoritmo SA são: a temperatura inicial T_0, o número máximo de iterações *SAmax* e a taxa de resfriamento α.

A calibragem destes parâmetros normalmente é feita empiricamente. Utilizar uma temperatura inicial muito alta faz com que um espaço muito grande de soluções seja visitado, aumentando o tempo computacional. O Algoritmo 1 (Ribeiro et al., 2011) é um pseudocódigo do Simulated Annealing:

Algoritmo 1 Simulated Annealing

1. obtém uma solução inicial *s*
2. $T \leftarrow T_0$
3. para $i \leftarrow 1$ até *SAmax*
4. perturba solução gerando solução vizinha *s′*
5. $\Delta \leftarrow f(S') - f(S)$
6. se ($\Delta < 0$)
7. $s = s'$
8. senão
9. faça $s = s'$ de acordo com probabilidade $e^{\frac{-\Delta}{T}}$
10. fim se
11. $T \leftarrow \alpha T$
12. fim para

3.1 Construção da solução inicial

A construção de uma solução inicial para o PROP é feita por um algoritmo guloso que inicia lendo o vetor $L[\ ]$ de atribuição de departamentos a linhas (e.g. se $L[i]=2$, então o departamento i deverá ser alocada na linha *2*). Em seguida o algoritmo escolhe aleatoriamente um primeiro departamento e o aloca na sua devida linha. A partir daí, a cada iteração, verifica para cada departamento ainda não alocado na sua devida linha, qual a contribuição na função objetivo ao se adicioná-lo na solução parcial em construção. O departamento que contribuir com o menor aumento da função objetivo é adicionado na sua devida linha. Note que a ordem em que cada departamento é inserido em uma linha k define a ordenação de departamentos S_k naquela linha.

Algoritmo 2 Construção da Solução Inicial

1. ler o vetor de atribuição de departamentos a linhas: $L[\ ]$
2. inicializa a ordenação na linha k: $S_k \leftarrow \emptyset$ para $k \in \{1, 2\}$
3. escolher aleatoriamente um departamento $i \in N$
4. inserir i no layout na sua devida linha: $S_{L[i]} \leftarrow S_{L[i]} + \{i\}$
5. $LC \leftarrow N - \{i\}$
6. enquanto $LC \neq \emptyset$
7. para cada departamento $j \in LC$
8. calcular a contribuição para o valor objetivo caso j fosse adicionado à solução parcial
9. fim para
10. seja j^* o departamento com a menor contribuição para o valor objetivo.
11. inserir j^* no layout na devida linha após o departamento anterior: $SL_{[j^*]} \leftarrow SL_{[j^*]} + \{j^*\}$
12. $LC \leftarrow LC - \{j^*\}$
13. fim enquanto

3.2 Simulated Annealing aplicado ao PROP

Após ter uma solução aleatória e gulosa já construída, o Algoritmo 3 mostra como o Simulated Annealing foi usado para resolver o PROP. Os seguintes parâmetros são recebidos pelo algoritmo SA: temperatura inicial T_0, temperatura final T_F, taxa de resfriamento α e número de perturbações P.

É feito um número determinado P de perturbações na solução para cada temperatura. A temperatura vai sendo reduzida por um fator de resfriamento α. A perturbação é feita de forma simples: trocando dois departamentos pertencentes a uma mesma linha.

Para a calibragem dos parâmetros, um dos parâmetros do SA é alterado enquanto os outros são mantidos fixos por um determinado número de execuções. Como o objetivo principal era ter uma boa solução no menor tempo possível, aceitou-se na calibragem um desvio nas soluções finais encontradas (desvio entre as soluções média e menor).

Após testes, os valores encontrados foram: temperatua inicial $T_0 = 1500$, temperatura final $T_F = 1$, número de perturbações $P = 400$ e taxa de resfriamento $\alpha = 0{,}975$.

Algoritmo 3 PROP-SA

1. constrói solução inicial s
2. $T \leftarrow T_0$
3. repita
4. repita
5. perturba solução gerando solução vizinha s'
6. $\Delta \leftarrow f(S') - f(S)$
7. se ($\Delta < 0$)
8. $s = s'$
9. senão
10. faça $s = s'$ de acordo com probabilidade $e^{\frac{-\Delta}{T}}$
11. fim se
12. $i \leftarrow i + 1$
13. até ($i > P$)
14. $T \leftarrow \alpha T$
15. até ($T > T_F$)

4. Experimentos Computacionais

O algoritmo SA proposto foi desenvolvido na linguagem de programação C. Os testes foram feitos em um notebook com processador intel core i7 2.0 GHz com 8 GB de memória RAM, utilizando o sistema operacional Ubuntu Linux 14.04 Kernel 3.13.0-48.

Os experimentos computacionais foram realizados baseados em instâncias usadas para o PROP em Hungerländer e Anjos (2015) e Amaral (2013). Para cada instância, o algoritmo Simulated Annealing foi executado 10 vezes. Não foram feitas comparações com resultados encontrados em Hungerländer e Anjos (2015) pois sua abordagem do PROP é diferente – nela são permitidos espaços entre os departamentos. Para instâncias grandes do PROP, são feitas comparações de diferentes versões do SA obtidas usando-se diferentes configurações de parâmetros.

4.1 PROP com instâncias de 16 a 23 departamentos

Primeiramente comparou-se os resultados do algoritmo com os valores ótimos obtidos pela abordagem exata de Amaral (2013) para instâncias do PROP de 16 a 23 departamentos.

Para comparar estes resultados, usou-se $t = \lfloor n/2 \rfloor$. Há 19 instâncias ao todo. Os resultados podem ser observados na Tabela 1.

O algoritmo Simulated Annealing conseguiu atingir o valor ótimo conhecido pelo menos em uma execução para todas as instâncias. Também obteve valores médios próximos ao valor mínimo, com desvios médios menores que 1%. Não se pode comparar os tempos computacionais diretamente pois foram utizados hardware diferentes. Assim, as duas últimas colunas da Tabela 1 apresentam tempos apenas para referência. Pode-se afirmar que o algoritmo SA é bem rápido para todas as instâncias.

4.2 PROP com instâncias de 60 a 80 departamentos

Instâncias de 60,70 e 80 departamentos encontradas em Hungerländer e Anjos (2015) também foram testadas pelo algoritmo. Os resultados do algoritmo Simulated Annealing são mostrados na Tabela 2.

Tabela 1. Comparação dos resultados do algoritmo com abordagem exata

Instância	*n*	Solução Ótima	Menor Valor	Valor Médio	Desvio (%)	Tempo Médio (s)	Tempo Amaral (s)
P16_a	16	**7630**	**7630**	7638	0,104849279	0,12169	26,18
P16_b		**6239,5**	**6239,5**	6241,9	0,038464621	0,111331	46,69
P20_a	20	**12609,5**	**12609,5**	12637,8	0,224433959	0,156431	4461,36
P20_b		**12936**	**12936**	12969	0,255102041	0,170877	1080,51
P21_a	21	**7006,5**	**7006,5**	7038,1	0,451009777	0,163601	1316,27
P21_b		**11705**	**11705**	11718,1	0,111917984	0,164771	2095,24
P21_c		**11434**	**11434**	11434	0	0,152378	568,32
P21_d		**12289**	**12289**	12345,2	0,457319554	0,15846	3701,92
P21_e		**13112,5**	**13112,5**	13131,1	0,14184938	0,164313	5754,71
P22_a	22	**8874**	**8874**	8885,2	0,126211404	0,170023	5114,44
P22_b		**15714**	**15714**	15714	0	0,1775	7506,63
P22_c		**14693**	**14693**	14707,1	0,095964065	0,160904	63825,77
P22_d		**16355**	**16355**	16381,5	0,16202996	0,176657	41713,32
P22_e		**14815,5**	**14815,5**	14867,2	0,348958861	0,174682	55591,31
P23_a	23	**10242**	**10242**	10300,7	0,573130248	0,172884	52515,75
P23_b		**15802,5**	**15802,5**	15806,2	0,023414017	0,184683	28371,18
P23_c		**15542**	**15542**	15542	0	0,180813	79860,19
P23_d		**17174**	**17174**	17176,6	0,015139164	0,201502	61151,19
P23_e		**16481,5**	**16481,5**	16572,1	0,549707248	0,18011	96033,18

Tabela 2. Resultados do SA para instâncias grandes

Instância	*n*	Menor Valor	Desvio (%)	Valor Médio	Tempo Médio (s)
Anjos60_05	60	165096	0,27402239	165548,4	0,803113
Anjos70_05	70	2193147,5	0,18307022	2197162,5	1,09368
Anjos80_05	80	800890	0,21004133	802572,2	1,39609

Para essas instâncias com n=60,70,80, os desvios das soluções do algoritmo SA permaneceram baixos, assim como os tempos permaneceram baixos.

A partir desses resultados, foram feitas comparações com configurações alternativas do algoritmo SA as quais foram obtidas variando-se parâmetros. Na versão SA-1, o parâmetro de temperatura inicial T_0 foi aumentado para 10000 enquanto os outros parâmetros foram mantidos, e na versão SA-2, o parâmetro de número de perturbações foi aumentado, também com os outros inalterados. Os resultados das versões alternativas podem ser observados nas Tabela 3 e 4.

Tabela 3. Resultados do SA-1 com T_0=10000

		Alternativa 1			
Instância	*n*	Menor Valor	Desvio (%)	Valor Médio	Tempo Médio (s)
Anjos60_05	60	165100	0,31403193	165620,1	1,032508
Anjos70_05	70	2187780,5	0,19000474	2191945,3	1,375265
Anjos80_05	80	800669	0,17630343	802083,1	1,776282

Tabela 4. Resultados do SA-2 com *P*=1000

Instância	*n*	Menor Valor	Desvio (%)	Valor Médio	Tempo Médio (s)
Anjos60_05	60	165096	0,29073993	165576	1,993883
Anjos70_05	70	2187780,5	0,33035307	2195007,9	2,747511
Anjos80_05	80	800703	0,19646486	802276,1	3,472707

Observando os resultados das versões alternativas, conclui-se que ao aumentar a temperatura inicial, os tempos aumentaram em até 28,5% enquanto que os valores médios da solução melhoraram em no máximo 0,23%. Da mesma forma, ao aumentar o número máximo de perturbações, percebeu-se um aumento de até 151,21% no tempo computacional com uma diminuição do valor médio da solução de no máximo 0,09%.

5. Conclusões

O Problema de Ordenação em Linhas Paralelas (PROP), é um problema computacionalmente difícil e com muitas aplicações na indústria. Este artigo apresentou um algoritmo Simulated Annealing para resolvê-lo.

A comparação de resultados do algoritmo com os resultados ótimos obtidos por uma abordagem exata mostrou que o Simulated Annealing é eficiente na resolução do PROP. Resultados para instâncias maiores do PROP confirmaram sua eficiência e serviram para comparação da influência dos parâmetros na solução do algoritmo.

Como trabalhos futuros, pretende-se fazer uma adaptação do algoritmo proposto para tratar outros problemas de layout, como o problema de layout com múltiplas linhas, ou Multi-Row Facility Layout Problem, ou o próprio *k*-PROP com mais de duas linhas paralelas.

6. Agradecimentos

Bernardo Cellin agradece a bolsa de mestrado da FAPES.

Referências

Amaral, A.R.S. (2006), On the exact solution of a facility layout problem. European Journal of Operational Research, v. 173, n. 2, p. 508–518.

Amaral, A.R.S. (2008), An exact approach to the one-dimensional facility layout problem. Operations Research, v. 56, n. 4, p. 1026–1033.

Amaral A.R.S. (2009), A new lower bound for the single row facility layout problem, Discrete Applied Mathematics, v. 157, n. 1, p. 183-190.

Amaral, A.R.S. (2013), A parallel ordering problem in facilities layout. Computers & Operations Research, v. 40, n. 12, p. 2930–2939.

Anjos, M.F. e Vannelli A. (2008), Computing globally optimal solutions for single-row layout problems using semidefinite programming and cutting planes. INFORMS Journal on Computing, v. 20, n. 4, p. 611-617.

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, v. 213, n. 2, p. 388-394.

De Alvarenga, A.G., Negreiros-Gomes, F.J. e Mestria, M. (2000), Metaheuristic methods for a class of the facility layout problem, v. 11, n. 4, p. 421-430.

Hungerländer, P. e Anjos, M.F. (2015), A Semidefinite Optimization-Based Approach for Global Optimization of Multi-Row Facility Layout, European Journal of Operational Research.

Hungerländer, P. e Rendl, F. (2011), A computational study and survey of methods for the single-row facility layout problem, Technical Report, Alpen-Adria-Universitaet Klagenfurt.

Kirkpatrick, S., Gellat, D. C. e Vecchi, M. P. (1983). Optimization by simulated annealing. Science, v. 220, p. 671–680.

Kothari, R. e Ghosh, D. (2013), Tabu search for the single row facility layout problem using exhaustive 2-opt and insertion neighborhoods, European Journal of Operational Research, v. 224, n. 1, p. 93-100.
Ribeiro, G.M., Mauri, G.R. e Lorena, L.A.N. (2011), A simple and robust Simulated Annealing algorithm for scheduling workover rigs on onshore oil fields, Computers & Industrial Engineering, v. 60, p. 519-526.