Um algoritmo Simulated Annealing e um Híbrido VNS/ILS para o Problema de Ordenação em Linhas Paralelas

Bernardo De Polli Cellin
Programa de Pós-Graduação 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. O problema de ordenação em linhas paralelas, ou k-Parallel Row Ordering Problem (k-PROP), assume que várias facilidades estão atribuídas a k linhas paralelas. O objetivo do k-PROP é encontrar uma ordenação de facilidades para cada linha a fim de minimizar uma função de custo total. Neste artigo, implementações da meta-heurística *Simulated Annealing* e de um algoritmo híbrido VNS/ILS são propostas para resolver o k-PROP para duas a cinco linhas paralelas. Resultados computacionais com instâncias da literatura demonstram a eficiência dos algoritmos propostos.

PALAVRAS CHAVE. Problema de Ordenação em Linhas Paralelas, Simulated Annealing, Algoritmo Híbrido VNS/ILS.

Meta-heurísticas, Otimização combinatória.

ABSTRACT

Facility layout problems are NP-hard and have many applications in industry. The Parallel Row Ordering Problem (k-PROP) considers several facilities that are pre-assigned to k parallel rows. The objective of the k-PROP is to find an ordering of facilities for each row so as to minimize a total cost function. In this article implementations of the meta-heuristic Simulated Annealing and a hybrid algorithm VNS / ILS are proposed to solve the k-PROP for two to five parallel rows. Computational results on instances of the literature demonstrate the efficiency of the proposed algorithms.

KEYWORDS. Parallel Row Ordering Problem, Simulated Annealing, Hybrid VNS/ILS Algorithm.

Metaheuristics, Combinatorial Optimization.

1 - Introdução

O problema de Ordenação em k Linhas Paralelas, ou *k-Parallel Row Ordering Problem* (k-PROP), considera um conjunto de n facilidades alocadas de modo prévio em k linhas paralelas, sendo que cada facilidade pode mudar sua posição na linha à qual foi atribuída, mas a facilidade não pode mudar de linha.

O objetivo do problema é encontrar um arranjo de facilidades em cada linha que minimize a função de custo total, a qual é baseada na soma das distâncias horizontais entre facilidades multiplicadas pelo custo de fluxo entre cada uma delas. Quando k é igual a 2, o problema pode ser chamado apenas de PROP [Amaral, 2013].

Entre as aplicações do k-PROP estão o arranjo de facilidades ao longo de duas ou mais linhas em uma planta na indústria (que podem ser máquinas ao redor de uma esteira), o planejamento de edifícios com múltiplos andares [Amaral, 2013] e o Layout de facilidades em sistemas flexíveis de manufatura [Heragu e Kusiak, 1991].

O k-PROP é uma generalização do Problema de Arranjo de Facilidades em Fila Única, ou *Single Row Facility Layout Problem* (SRFLP). No SRFLP todas as facilidades devem ser arranjadas em uma única linha. O SRFLP tem sido resolvido por diversos métodos exatos: [Simmons, 1969] utilizou Branch-and-Bound; [Picard e Queyranne, 1981] resolveram o problema com Programação Dinâmica; [Amaral, 2006, 2008b; Heragu e Kusiak, 1991; Love e Wong, 1976] apresentaram modelos de Programação Inteira-mista, ou *Mixed Integer Programming* (MIP); [Amaral, 2009] introduziu um algoritmo de planos de corte; [Amaral e Letchford, 2013] apresentaram um algoritmo Branch-and-Cut; [Anjos e Vannelli, 2008] combinou o modelo de programação semi-definida proposto por [Anjos et al., 2005] utilizando planos de corte; [Hungerländer e Rendl, 2013] também propuseram um modelo de programação semi-definida.

Heurísticas e meta-heurísticas têm sido bastante utilizadas para resolver o SRFLP. [De Alvarenga et al., 2000] propuseram dois algoritmos: um baseado em *Simulated Annealing* (SA) e outro em Busca Tabu; [Amaral, 2008a] propôs um algoritmo baseado em *Enhanced Local Search*; [Samarghandi et al., 2010] propuseram um algoritmo que utilizou *Particle Swarming Optimization* (PSO); [Ozcelik, 2012] apresentou um algoritmo híbrido combinando Algoritmos Genéticos com Buscas Locais; [Kothari e Ghosh, 2013a] propuseram uma implementação de Algoritmo Genético; [Kothari e Ghosh, 2014] introduziram uma implementação do algoritmo Scatter Search; [Kothari e Ghosh, 2013b] apresentaram implementações eficientes da Busca Tabu. Recentemente, [Palubeckis, 2015] utilizou a meta-heurística Busca em Vizinhança Variável (*Variable Neighbohood Search* - VNS); [Guan e Lin, 2016] propuseram um algoritmo híbrido que faz uso das meta-heurísticas VNS e Colônia de Formigas com eficientes estruturas de vizinhança; [Palubeckis, 2017] desenvolveu um algoritmo baseado em *Simulated Annealing* com múltiplas inicializações (*Multi-start*).

Para o PROP, [Amaral, 2013] propôs um modelo MIP que obteve soluções ótimas para instâncias com até 23 facilidades. [Cellin e Amaral, 2015] propuseram um algoritmo *Simulated Annealing* para o problema com duas linhas paralelas; [Hungerländer e Anjos, 2015] utilizaram um modelo de programação semidefinida para o k-PROP, com até 5 linhas paralelas, que permitia espaços entre as facilidades. Para adaptar instâncias do SRFLP para o k-PROP, os autores fizeram um balanceamento do comprimento total das linhas do layout.

Problemas de Layout de Facilidades são NP-difíceis (*NP-hard*) [Garey e Johnson, 1979]. Portanto, meta-heurísticas são largamente utilizadas para resolvê-los em tempo computacionalmente viável. Neste trabalho, um algoritmo *Simulated Annealing* (SA) e um algoritmo híbrido que utiliza as meta-heurísticas *Variable Neighborhood Search* (VNS) e Busca Local Iterada (*Iterated Local Search* - ILS) são utilizados para resolver o k-PROP.

Este artigo está organizado da seguinte maneira: a Seção 2 apresenta a formulação do k-PROP, a Seção 3 descreve os algoritmos propostos, a Seção 4 mostra os resultados computacionais. Finalmente na seção 5 são mostradas as conclusões.

2 - Formulação do k-PROP

Na versão do k-PROP tratada aqui, assumindo que se tem k linhas paralelas horizontais em um layout, o arranjo de facilidades precisa iniciar-se de uma linha vertical de referência, e não são permitidos espaços entre as facilidades adjacentes. A distância entre os centros das facilidades é calculada apenas ao longo do eixo horizontal. Arranjar as facilidades significa encontrar as posições x dos centros de cada facilidade em sua respectiva linha do Layout.

No formulação do PROP, tem-se duas linhas paralelas horizontais {Linha 1, Linha 2} e o conjunto $N = \{1, ..., n\}$ de facilidades. t dessas facilidades $\{1, ..., t\}$ devem ser alocadas na primeira linha (Linha 1) e o restante (n - t) das facilidades $\{t + 1, ..., n\}$ devem ser alocadas na segunda linha (Linha 2). Π_t^1 será o conjunto de todas as permutações de $N_1 = \{1, ..., t\}$ e Π_{n-t}^2 será o conjunto de todas as permutações de $N_2 = \{t + 1, ..., n\}$.

Para calcular a distância entre duas facilidades, apenas o eixo horizontal x é levado em conta. Então, o PROP pode ser definido como:

$$\min_{\pi_1 \in \Pi_1, \pi_1 \in \Pi_2} \left\{ \left(\sum_{i,j \in N_1, i<j} c_{ij} d_{ij}^{\pi_1} \right) + \left(\sum_{i,j \in N_2, i<j} c_{ij} d_{ij}^{\pi_2} \right) + \left(\sum_{i \in N_1, j \in N_2, i<j} c_{ij} d_{ij}^{\pi_1, \pi_2} \right) \right\} \quad (1)$$

no qual:

c_{ij} é o custo de fluxo entre as facilidades $i, j \in N$;

$d_{ij}^{\pi_1}$ é a distância no eixo x entre as facilidades $i, j \in N_1$ com relação à permutação π_1;

$d_{ij}^{\pi_2}$ é a distância no eixo x entre as facilidades $i, j \in N_2$ com relação à permutação π_2;

$d_{ij}^{\pi_1, \pi_2}$ é a distância no eixo x entre as facilidades $i \in N_1$ e $j \in N_2$ com relação às permutações π_1 e π_2;

Quando k é maior que 2, a formulação também pode ser expandida e aplicada. Então, a permutação π_r de um conjunto N_r é considerado em cada linha *r*, $(1 \leq r \leq k)$.

3 - Solução Inicial

A construção de uma solução inicial para o k-PROP é feita por um algoritmo guloso que inicia lendo o vetor $L[\ ]$ de atribuição de departamentos a linhas (por exemplo, se $L[i]$=2, então a facilidade i deverá ser alocada na linha 2). Em seguida o algoritmo escolhe aleatoriamente um primeiro departamento e o aloca na sua devida linha. A cada iteração a partir deste ponto, o algoritmo verifica para cada departamento ainda não alocado, qual a contribuição na função objetivo ao se adicioná-lo na solução parcial em construção. O departamento que contribuir com o maior aumento da função objetivo é adicionado na sua devida linha (a fim de tentar colocar facilidades com maior fluxo juntas). Note que a ordem em que cada departamento é inserido em uma linha r define a ordenação de departamentos naquela linha. O pseudocódigo pode ser visualizado no Algoritmo 1.

4 - Movimentos de Troca e Busca Local

Neste trabalho foram utilizados dois tipos de movimentos de troca: a *troca simples* e a *troca de coluna*, que podem ser visualizados na Figura 1.

A *troca simples* consiste em trocar duas facilidades de uma mesma linha de posição. Como não deve haver sobreposições nem espaços entre facilidades, deve-se corrigir o layout caso isso ocorra. Corrigir significa deslocar as facilidades que estão sobrepostas ou afastadas das facilidades vizinhas a fim de tornar o layout viável para o problema.

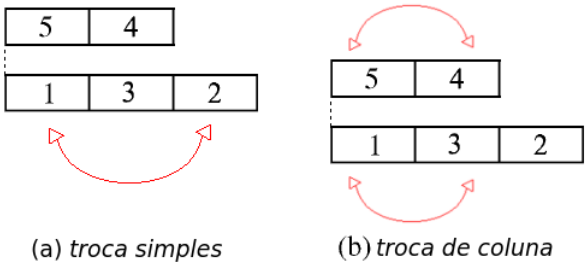
A *troca de coluna* consiste em selecionar duas posições relativas e trocar todos os pares de facilidades que estão nessas posições em todas as linhas, também respeitando as restrições de que não pode haver sobreposições e espaços entre facilidades.

Algoritmo 1 Construindo uma solução inicial - PROP

Entrada: lista de atribuição de cada linha $L[\ ]$
Saída: solução s

inicializa a ordenação na linha $r \in \{1, 2, ..., k\}$: $\pi^r \leftarrow \emptyset$ para cada linha
aleatoriamente seleciona $i \in N$
inserir i em sua devida linha no layout: $\pi^{L[i]} \leftarrow \pi^{L[i]} + \{i\}$
$LC \leftarrow N - \{i\}$
enquanto $LC \neq \emptyset$ **faça**
    para para cada facilidade $j \in LC$ **faça**
        calcular o aumento na função objetivo caso a facilidade j fosse alocada ao layout parcial
    fim para
    seja $j*$ a facilidade com maior contribuição de valor para a função objetivo
    inserir $j*$ na sua devida linha imediatamente após a facilidade anterior naquela linha: $\pi^{L[i]} \leftarrow \pi^{L[i]} + \{j*\}$
    $LC \leftarrow LC - \{j*\}$
fim enquanto

Figura 1: Funcionamento do ILS



De acordo com o Algoritmo 2, a busca local recebe um parâmetro y identificando o movimento de troca a ser utilizado. Caso y seja igual a 1, o movimento escolhido é a troca simples; caso seja igual a 2 o movimento escolhido é a troca de coluna. Nos dois laços mais internos, para cada linha r, os índices i e j representam todos os pares de posições relativas de facilidades a serem trocados na linha selecionada com o devido movimento de troca, com i e j indo de 1 até no máximo $N[linhaAtual]$, o qual é número de facilidades na linha corrente. No caso do movimento de troca simples, o algoritmo gera uma nova solução s' trocando as facilidades nas posições i e j; caso o movimento seja o de troca de coluna, o algoritmo gera s' trocando todos os pares nas posições relativas i e j de cada linha r. O laço mais externo mantém a busca até que a solução não possa ser mais melhorada. A busca local é utilizada no algoritmo híbrido VNS/ILS.

5 - *Simulated Annealing*

O algoritmo *Simulated Annealing* utilizado para resolver o k-PROP recebe os seguintes parâmetros: o número máximo de iterações SA_{max}, a taxa de resfriamento α e o fator β. A perturbação é feita escolhendo-se aleatoriamente um dos movimentos de troca para ser aplicado na solução corrente e gerar nova solução s'. Se nenhuma solução for aceita no laço interno, ou seja, caso $\Delta < 0$ seja falso para todas as novas soluções geradas, o valor da temperatura atual é aumentado por um fator β. Tal procedimento é usado para ajudar o algoritmo a escapar de ótimos locais. O pseudocódigo pode ser visualizado no Algoritmo 3.

Algoritmo 2 Busca Local

Entrada: solução s, identificador de movimento y
Saída: solução $sBest$

$sBest \leftarrow s$
$melhorou \leftarrow$ FALSO
enquanto $melhorou =$ FALSO **faça**
  $melhorou \leftarrow$ FALSO
  para $linhaAtual \leftarrow 1$ **to** k **faça**
    para $i \leftarrow 1$ **to** $N[linhaAtual] - 1$ **faça**
      para $j \leftarrow i + 1$ **to** $N[linhaAtual]$ **faça**
        gera solução s' aplicando o movimento de troca y com os parâmetros i e j
        se $f(f(s')) < f(s)$ **então**
          $s \leftarrow s'$
          $improved \leftarrow$ VERDADEIRO
          se $f(s) < f(sBest)$ **então**
            $sBest \leftarrow s$
          fim se
        fim se
      fim para
    fim para
  fim para
fim enquanto

Algoritmo 3 Simulated Annealing

Entrada: solução inicial s, T_0, α, β
Saída: solução $sBest$

$sBest \leftarrow s$
$T \leftarrow T_0$
para $i \leftarrow 1$ **to** $SAmax$ **faça**
  para $j \leftarrow 1$ **to** P **faça**
    perturba solução gerando solução vizinha s'
    $\Delta \leftarrow f(s') - f(s)$
    se $\Delta < 0$ **então**
      $s \leftarrow s'$
      se $f(s') < f(sBest)$ **então**
        $sBest \leftarrow s'$
      fim se
    senão
      faça $s \leftarrow s'$ de acordo com probabilidade $e^{\frac{-\Delta}{T}}$
    fim se
  fim para
  se nenhuma solução melhor foi aceita **então**
    $T \leftarrow \beta T$
  fim se
  $T \leftarrow \alpha T$
fim para

6 - Algoritmo Híbrido VNS/ILS

Também foi implementado um algoritmo híbrido que é baseado nas meta-heurísticas ILS e VNS. De acordo com o Algoritmo 4, a cada iteração do laço interno, o algoritmo perturba a solução de acordo com uma intensidade dada pela variável p, que aumenta no caso de nenhuma solução melhor ser encontrada no laço interno até atingir um valor p_{max}. Perturbar a solução com intensidade p significa aplicar p movimentos de troca aleatórios na solução. Após a perturbação, o algoritmo faz buscas locais com o movimento de troca selecionado, indicado pelo valor de y. No caso do k-PROP, são dois movimentos de troca então y tem valor máximo de 2. Se uma busca local é bem sucedida, y aponta novamente para o primeiro movimento de troca e as buscas recomeçam. Para prevenir laços na busca local, há um movimento de *shake* do VNS, que aplica movimentos de troca aleatórios na solução, incluindo um movimento de deslocar uma facilidade para o final da sua devida linha.

Algoritmo 4 Algoritmo Híbrido VNS/ILS

Entrada: solução s, p_{max}, i_{max}
Saída: solução $sBest$

```
1:  sBest ← s
2:  para i ← 1 to i_max faça
3:      p ← 1
4:      enquanto p ≤ p_max faça
5:          perturbar a solução atual s com intensidade p
6:          enquanto y ≤ 2 faça
7:              s' ← shake(s)
8:              s' ← local search(s', y)
9:              y ← y + 1
10:             se f(s') < f(s) então
11:                 s ← s'
12:                 y ← 1
13:             fim se
14:         fim enquanto
15:         se f(s) < f(sBest) então
16:             sBest ← s
17:             p ← 1
18:         senão
19:             p ← p + 1
20:         fim se
21:     fim enquanto
22:     i ← i + 1
23: fim para
```

7 - Resultados Computacionais

Para os testes computacionais foram utilizadas 19 instâncias do PROP com tamanhos que vão de 16 até 23 facilidades com soluções ótimas calculadas pelo modelo MIP de [Amaral, 2013]. Os algoritmos são então validados de acordo com esses valores encontrados na literatura. Além disso, as instâncias, com tamanhos que vão de 60 ate 80 facilidades, usadas por [Anjos et al., 2005] para o SRFLP foram adaptadas para o k-PROP. Para adaptá-las, o número de facilidades em cada linha do k-PROP foi tomado igual a $\lfloor \frac{n}{k} \rfloor$. Na lista de facilidades da instância do SRFLP, são tomadas as $\lfloor \frac{n}{k} \rfloor$ primeiras facilidades para a linha 1, as $\lfloor \frac{n}{k} \rfloor$ facilidades seguintes para a linha 2, e assim por diante.

Tabela 1: Instâncias com tamanhos entre 16 e 23 facilidades - PROP

	MIP Amaral	SA			VNS/ILS		
Instância	Valor Ótimo	Mínimo	Média	DP	Mínimo	Média	DP
P16_a	**7630,0**	**7630,0**	7634,0	12,0	**7630,0**	7648,4	29,3
P16_b	**6239,5**	**6239,5**	6239,5	0,0	**6239,5**	6239,5	0,0
P20_a	**12609,5**	**12609,5**	12612,8	9,9	**12609,5**	12623,9	4,8
P20_b	**12936,0**	**12936,0**	12938,6	5,2	**12936,0**	12936,0	0,0
P21_a	**7006,5**	**7006,5**	7006,5	0,0	**7006,5**	7013,8	9,0
P21_b	**11705,0**	**11705,0**	11705,3	0,9	**11705,0**	11705,0	0,0
P21_c	**11434,0**	**11434,0**	11434,0	0,0	**11434,0**	11437,3	6,6
P21_d	**12289,0**	**12289,0**	12289,0	0,0	**12289,0**	12289,4	1,2
P21_e	**13112,5**	**13112,5**	13112,7	0,6	**13112,5**	13112,9	0,8
P22_a	**8874,0**	**8874,0**	8874,2	0,6	**8874,0**	8889,4	23,9
P22_b	**15714,0**	**15714,0**	15714,0	0,0	**15714,0**	15714,0	0,0
P22_c	**14693,0**	**14693,0**	14693,0	0,0	**14693,0**	14723,5	21,3
P22_d	**16355,0**	**16355,0**	16366,7	17,9	**16355,0**	16360,5	5,5
P22_e	**14815,5**	**14815,5**	14822,5	21,0	**14815,5**	14815,5	0,0
P23_a	**10242,0**	**10242,0**	10242,0	0,0	**10242,0**	10250,9	9,6
P23_b	**15802,5**	**15802,5**	15802,5	0,0	**15802,5**	15809,9	22,2
P23_c	**15542,0**	**15542,0**	15542,0	0,0	**15542,0**	15542,0	0,0
P23_d	**17174,0**	**17174,0**	17174,0	0,0	**17174,0**	17188,0	9,2
P23_e	**16481,5**	**16481,5**	16513,3	63,6	**16481,5**	16481,5	0,0
			12669,3	6,9		12672,7	7,5

Os parâmetros utilizados no algoritmo *Simulated Annealing*, T_0, SA_{max} e α, foram iguais a respectivamente 3500.0, 650 e 0.975 para $n \leq 30$; 10000.0, 2500 e 0.99 para $31 \leq n \leq 45$ e 20000.0, 4000 e 0.995 para $n \geq 46$. O número de perturbações P é igual a 400 e o fator β é igual a $\frac{1}{\alpha}$. Já os parâmetros do VNS/ILS i_{max} e p_{max} são iguais ao número de facilidades n.

Não há comparações com os resutados de [Hungerländer e Anjos, 2015] porque a abordagem deles é diferente já que permite espaços entre facilidades.

Os algoritmos foram implementados utilizando a linguagem de programação C e compilados com gcc 5.4. Os testes foram feitos em um computador com processador Intel core i7 2.0 GHz com 8 GB de RAM e sistema operacional Linux Ubuntu 16.04. Para cada instância, os algoritmos foram executados 10 vezes.

7.1 - PROP com instâncias de 11 a 23 facilidades

Os resultados de ambos os algoritmos são validados utilizando os valores ótimos obtidos na literatura e podem ser vistos na tabela 1. Para tal são utlizadas instâncias com tamanhos iguais a 16, 20, 21, 22 e 23 facilidades.

A primeira coluna da Tabela 1 mostra os nomes das instâncias. A segunda coluna mostra os valores ótimos encontrados na literatura. A terceira, quarta e quinta colunas mostram respectivamente os valores mínimos (Mínimo), os valores médios (Média) e desvios-padrão (DP) obtidos em todas as execuções do *Simulated Annealing*. Já a sexta, sétima e oitava colunas mostram respectivamente os valores mínimos (Mínimo), os valores médios (Média) e desvios-padrão (DP) obtidos nas execuções do Algoritmo Híbrido VNS/ILS. Na última linha são mostradas as médias das colunas DP e Média dos dois algoritmos propostos. Os valores em negrito indicam as melhores soluções encontradas para cada instância.

Os tempos computacionais, em segundos, do algoritmo exato e dos algoritmos heurísticos podem ser observados na Tabela 2. Também podem ser observados os valores médios na última linha da tabela.

Tabela 2: Tempos computacionais em segundos para instâncias com tamanhos entre 16 e 23 facilidades - PROP

Instância	MIP Amaral	SA	VNS/ILS
P16_a	26,18	0,56	0,01
P16_b	46,69	0,51	0,01
P20_a	4461,36	0,79	0,05
P20_b	1080,51	0,82	0,05
P21_a	1316,27	0,88	0,06
P21_b	2095,24	0,75	0,04
P21_c	568,32	0,68	0,06
P21_d	3701,92	0,92	0,05
P21_e	5754,71	0,81	0,06
P22_a	5114,44	1,07	0,08
P22_b	7506,63	0,77	0,06
P22_c	63825,77	0,89	0,06
P22_d	41713,32	0,78	0,10
P22_e	55591,31	1,01	0,07
P23_a	52515,75	1,01	0,10
P23_b	28371,18	0,90	0,08
P23_c	79860,19	1,05	0,09
P23_d	61151,19	0,96	0,08
P23_e	96033,18	1,13	0,07
	26880,75	0,86	0,06

Os dois algoritmos conseguiram atingir o valor ótimo conhecido pelo menos em uma execução para todas as instâncias, com valores médios próximos ou iguais ao mínimo encontrado. De acordo com a última linha da tabela, o *Simulated Annealing* mostrou desvios padrão e soluções médias menores que o VNS/ILS. De forma geral, os algoritmos obtiveram desvios padrão relativamente pequenos nos testes.

Devido a diferenças de hardware, não se pode comparar diretamente os tempos computacionais dos algoritmos com a abordagem exata mas a Tabela 2 exibe os tempos computacionais de ambos algoritmos e da resolução pela abordagem exata para dar uma ideia de seus valores. Comparando-se os dois algoritmos, pode-se ver que o Algoritmo Híbrido VNS/ILS exibiu menor tempo computacional para todas as instâncias.

7.2 - Instâncias grandes para o PROP

Na Tabela 3 podem ser observados os resultados de ambas as meta-heurísticas para um conjunto de 20 instâncias encontradas em [Anjos et al., 2005]. Esse conjunto é chamado de AKV e possui 5 instâncias para cada tamanho: n = 60, 70, 75 e 80. Na tabela podem ser verificados valores mínimos, médios, desvios padrão bem como o tempo computacional médio em segundos. Os valores em negrito indicam as melhores soluções encontradas para a instância. Na última linha são mostradas as médias das colunas.

Pode-se ver que o algoritmo *Simulated Annealing* obteve valores mínimos iguais ou melhores que os do Algoritmo Híbrido VNS/ILS na maioria das 20 instâncias do conjunto, apesar de que o Algoritmo Híbrido tenha chegado a valores mínimos próximos. O Algoritmo Híbrido somente conseguiu melhor valor mínimo na instância AKV-75-03. Os desvios padrão encontrados foram relativamente baixos sendo que as médias das colunas Média e DP apresentadas pelo algoritmo híbrido VNS/ILS foram menores que as do SA.

De forma geral, o SA executou em tempos computacionais bem menores do que o VNS/ILS, chegando a ser até mais de vezes menor para a instância AKV-80-04.

Tabela 3: Instâncias (AKV) - PROP

	SA				VNS/ILS			
Instância	Mínimo	Média	DP	Tempo	Mínimo	Média	DP	Tempo
AKV-60-01	**772202,0**	772392,7	190,9	24,2	**772202,0**	772428,0	230,8	55,9
AKV-60-02	**430380,0**	430417,5	89,7	22,0	**430380,0**	430436,0	29,2	74,0
AKV-60-03	**331103,5**	331439,9	614,3	21,3	331103,5	331211,8	156,5	83,3
AKV-60-04	**201052,0**	201117,1	67,2	22,8	201128,0	201401,4	108,2	80,9
AKV-60-05	**165096,0**	165195,4	141,2	20,4	165099,0	165103,6	4,4	80,2
AKV-70-01	**779130,0**	779586,0	482,9	39,3	**779130,0**	779416,1	270,5	208,3
AKV-70-02	**738112,0**	739180,4	479,3	35,2	737919,0	738976,0	700,7	211,5
AKV-70-03	**764463,5**	764774,5	401,2	33,9	**764463,5**	765158,8	333,0	157,6
AKV-70-04	**491028,0**	491267,5	602,6	35,9	**491028,0**	491072,8	36,8	194,4
AKV-70-05	**2187780,5**	2188679,9	1373,9	43,6	**2187780,5**	2187883,9	127,6	134,2
AKV-75-01	**1214737,5**	1215792,9	1569,5	36,2	**1214737,5**	1215251,1	685,0	249,3
AKV-75-02	**2172972,0**	2174698,8	1899,5	40,3	2173363,0	2173684,2	497,0	194,2
AKV-75-03	634169,0	634631,3	413,3	41,1	**634130,0**	634555,1	233,7	313,6
AKV-75-04	**2008998,5**	2009659,5	848,4	48,7	**2008998,5**	2009287,1	212,4	227,5
AKV-75-05	**914833,0**	915427,6	823,0	47,2	**914833,0**	914853,2	59,3	257,8
AKV-80-01	**1043521,5**	1043910,8	309,6	61,6	1043533,5	1043818,1	179,9	448,3
AKV-80-02	**986822,0**	987127,8	272,4	56,9	**986822,0**	987072,0	290,0	380,3
AKV-80-03	**1638732,0**	1640339,9	2140,2	46,6	1640029,0	1640584,9	859,4	482,0
AKV-80-04	**1902566,0**	1903904,9	638,6	44,9	1902594,0	1903391,8	604,1	532,9
AKV-80-05	**800668,0**	800759,5	151,2	46,3	800679,0	801152,1	322,8	482,0
	1008918,4	1009515,2	675,4	38,4	1008997,7	1009336,9	297,1	242,4

7.3 - *k*-PROP com 3 a 5 linhas paralelas

O k-PROP também foi resolvido com k variando de 3 a 5. Os resultados obtidos pelos algoritmos implementados foram comparados entre si e podem ser observados na Tabela 4, que possui a mesma estrutura da Tabela 3 além de uma coluna indicando o número k de linhas paralelas. Após os valores de cada instância para cada linha k são mostrados os valores médios de cada coluna.

Pode-se ver que o algoritmo híbrido VNS/ILS obteve valores mínimos iguais ou melhores que os do *Simulated Annealing* para todas as instâncias do conjunto e para todos os valores de k. Os desvios padrão encontrados foram iguais a zero ou relativamente baixos para o *Simulated Annealing* (relativos ao desvio padrão do VNS/ILS). Mesmo com maior desvio padrão médio o algoritmo híbrido VNS/ILS mostrou ser mais eficiente para o problema em múltiplas linhas, obtendo os menores valores mínimos e médios em menor tempo computacional.

8 - Conclusões

O problema de ordenação em linhas paralelas é um problema NP-difícil com várias aplicações na indústria. Este artigo propôs dois algoritmos para resolvê-lo: um algoritmo *Simulated Annealing* e um híbrido VNS/ILS.

As performances dos algoritmos foram avaliadas de acordo com várias instâncias encontradas na literatura.

A comparação de resultados entre os algoritmos mostrou que para instâncias grandes o algoritmo *Simulated Annealing* mostrou-se melhor, produzindo os melhores resultados para quase todas as instâncias. Já para as instâncias menores e com múltiplas linhas paralelas, o algoritmo VNS/ILS mostrou-se mais eficiente, produzindo melhores resultados em tempo computacional menor.

Para trabalhos futuros, pretende-se aplicar outras meta-heurísticas ao k-PROP.

Tabela 4: k-PROP com 3 a 5 linhas paralelas

		SA				VNS/ILS			
Instância	k	Mínimo	DP	Média	Tempo	Mínimo	DP	Média	Tempo
P21_a	3	**4867,5**	0,0	4867,5	0,766	**4867,5**	25,2	4881,7	0,015
P21_b		**8139,0**	0,0	8139,0	0,692	**8139,0**	17,1	8151,6	0,014
P21_c		**7846,0**	0,0	7846,0	0,685	**7846,0**	0,0	7846,0	0,014
P21_d		8260,0	0,0	8260,0	0,763	**8160,0**	61,9	8269,4	0,017
P21_e		**9156,5**	0,0	9156,5	0,716	**9156,5**	106,0	9230,3	0,015
P22_a		6213,0	0,0	6213,0	0,843	**5596,0**	214,5	6087,6	0,018
P22_b		10398,0	0,0	10398,0	0,812	**9548,0**	284,8	10100,7	0,028
P22_c		**10194,0**	6,6	10196,2	0,851	**10194,0**	17,7	10220,0	0,024
P22_d		**11053,0**	0,0	11053,0	0,761	**11053,0**	6,0	11056,0	0,019
P22_e		**10336,5**	0,0	10336,5	0,821	**10336,5**	2,4	10338,5	0,025
P23_a		**7321,0**	0,0	7321,0	0,864	**7321,0**	37,2	7336,7	0,037
P23_b		**11493,5**	0,0	11493,5	0,852	**11493,5**	20,8	11503,9	0,030
P23_c		**11293,0**	1,5	11294,2	0,967	**11293,0**	8,0	11297,3	0,037
P23_d		**11908,0**	11,7	11911,9	1,030	**11705,0**	72,8	11811,1	0,028
P23_e		**11199,5**	0,0	11199,5	1,084	**11199,5**	5,4	11206,1	0,027
		9311,9	1,3	9312,4	0,834	9193,9	58,7	9289,1	0,023
P21_a	4	**3921,5**	0,0	3921,5	0,687	**3921,5**	0,0	3921,5	0,009
P21_b		**6282,0**	0,0	6282,0	0,676	**6282,0**	8,1	6284,7	0,008
P21_c		**6804,0**	0,0	6804,0	0,652	**6804,0**	13,6	6817,8	0,011
P21_d		6940,0	0,0	6940,0	0,682	**6012,5**	282,0	6442,1	0,012
P21_e		7351,5	0,0	7351,5	0,665	**7147,5**	61,2	7329,4	0,009
P22_a		**5476,0**	0,0	5476,0	0,743	**5476,0**	14,0	5483,0	0,014
P22_b		8168,0	0,0	8168,0	0,793	**7878,0**	124,7	8075,8	0,020
P22_c		**8564,0**	0,0	8564,0	0,756	**8564,0**	2,8	8565,4	0,013
P22_d		**8308,0**	0,0	8308,0	0,739	**8308,0**	0,0	8308,0	0,015
P22_e		**8801,5**	0,0	8801,5	0,777	**8801,5**	0,0	8801,5	0,013
P23_a		**6583,0**	0,0	6583,0	0,816	**6583,0**	2,4	6584,2	0,018
P23_b		**9645,5**	0,0	9645,5	0,826	**9645,5**	0,0	9645,5	0,015
P23_c		**9533,0**	0,0	9533,0	0,813	**9533,0**	0,0	9533,0	0,015
P23_d		**10152,0**	3,9	10153,3	0,861	**10152,0**	19,7	10164,9	0,018
P23_e		**9299,5**	2,4	9300,3	0,837	**9299,5**	28,0	9333,8	0,014
		7722,0	0,4	7722,1	0,755	7627,2	37,1	7686,0	0,014
P21_a	5	**3032,5**	0,0	3032,5	0,803	**3032,5**	0,0	3032,5	0,005
P21_b		**5030,0**	0,0	5030,0	0,685	**5030,0**	6,4	5033,2	0,005
P21_c		**5323,0**	0,0	5323,0	0,628	**5323,0**	0,0	5323,0	0,005
P21_d		**5268,0**	0,0	5268,0	0,630	**5268,0**	0,0	5268,0	0,006
P21_e		**5541,5**	0,0	5541,5	0,634	**5541,5**	0,0	5541,5	0,006
P22_a		**4286,0**	0,0	4286,0	0,760	**4286,0**	0,0	4286,0	0,007
P22_b		**6615,0**	0,0	6615,0	0,721	**6615,0**	0,0	6615,0	0,008
P22_c		**6673,0**	0,0	6673,0	0,724	**6673,0**	2,7	6677,2	0,007
P22_d		**6624,0**	0,0	6624,0	0,758	**6624,0**	0,0	6624,0	0,008
P22_e		**7004,5**	0,0	7004,5	0,833	**7004,5**	0,0	7004,5	0,008
P23_a		**5293,0**	0,0	5293,0	0,772	**5293,0**	6,3	5295,1	0,009
P23_b		**8213,5**	0,0	8213,5	0,784	**8213,5**	2,0	8214,8	0,010
P23_c		**8001,0**	0,0	8001,0	0,745	**8001,0**	0,3	8001,1	0,009
P23_d		**8267,0**	0,0	8267,0	0,849	**8267,0**	0,0	8267,0	0,008
P23_e		**7778,5**	0,0	7778,5	0,831	**7778,5**	10,7	7782,3	0,011
		6196,7	0,0	6196,7	0,744	6196,7	1,9	6197,7	0,007

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*, 173(2):508 – 518. ISSN 0377-2217. URL http://www.sciencedirect.com/science/article/pii/S0377221705002031.

Amaral, A. R. S. (2008a). Enhanced local search applied to the single-row facility layout problem. *Simpósio Brasileiro de Pesquisa Operacional*. URL http://www.din.uem.br/sbpo/sbpo2008/pdf/arq0026.pdf.

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

Amaral, A. R. S. (2009). A new lower bound for the single row facility layout problem. *Discrete Applied Mathematics*, 157(1):183 – 190. ISSN 0166-218X. URL http://www.sciencedirect.com/science/article/pii/S0166218X0800259X.

Amaral, A. R. S. e Letchford, A. N. (2013). A polyhedral approach to the single row facility layout problem. *Mathematical Programming*, 141(1):453–477. ISSN 1436-4646. URL http://dx.doi.org/10.1007/s10107-012-0533-z.

Amaral, A. R. (2013). A parallel ordering problem in facilities layout. *Computers & Operations Research*, 40(12):2930 – 2939. ISSN 0305-0548. URL http://www.sciencedirect.com/science/article/pii/S0305054813001767.

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(2):113 – 122. ISSN 1572-5286. URL http://www.sciencedirect.com/science/article/pii/S1572528605000174.

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*, 20(4):611–617. URL http://dx.doi.org/10.1287/ijoc.1080.0270.

Cellin, B. D. P. e Amaral, A. R. S. (2015). Simulated annealing aplicado ao problema de ordenação em linhas paralelas. *XLVII Simpósio Brasileiro de Pesquisa Operacional*.

De Alvarenga, A., Negreiros-Gomes, F. J., e Mestria, M. (2000). Metaheuristic methods for a class of the facility layout problem. *Journal of Intelligent Manufacturing*, 11(4):421–430. ISSN 0956-5515. URL http://dx.doi.org/10.1023/A%3A1008982420344.

Garey, M. R. e Johnson, D. S. (1979). *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman and Co., New York, NY, USA. ISBN 0716710447.

Guan, J. e Lin, G. (2016). Hybridizing variable neighborhood search with ant colony optimization for solving the single row facility layout problem. *European Journal of Operational Research*, 248(3):899 – 909. ISSN 0377-2217. URL http://www.sciencedirect.com/science/article/pii/S0377221715007316.

Heragu, S. S. e Kusiak, A. (1991). Efficient models for the facility layout problem. *European Journal of Operational Research*, 53(1):1 – 13. ISSN 0377-2217. URL http://www.sciencedirect.com/science/article/pii/037722179190088D.

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*, 245(1):46 – 61. ISSN 0377-2217. URL http://www.sciencedirect.com/science/article/pii/S0377221715001691.

Hungerländer, P. e Rendl, F. (2013). A computational study and survey of methods for the single-row facility layout problem. *Computational Optimization and Applications*, 55(1):1–20. ISSN 0926-6003. URL http://dx.doi.org/10.1007/s10589-012-9505-8.

Kothari, R. e Ghosh, D. (2013a). An efficient genetic algorithm for single row facility layout. *Optimization Letters*, 8(2):679–690. ISSN 1862-4480. URL http://dx.doi.org/10.1007/s11590-012-0605-2.

Kothari, R. e Ghosh, D. (2013b). Tabu search for the single row facility layout problem using exhaustive 2-opt and insertion neighborhoods. *European Journal of Operational Research*, 224 (1):93 – 100. ISSN 0377-2217. URL http://www.sciencedirect.com/science/article/pii/S0377221712005942.

Kothari, R. e Ghosh, D. (2014). A scatter search algorithm for the single row facility layout problem. *Journal of Heuristics*, 20(2):125–142. ISSN 1862-4480. URL http://dx.doi.org/10.1007/s11590-012-0605-2.

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. (2012). A hybrid genetic algorithm for the single row layout problem. *International Journal of Production Research*, 50(20):5872–5886. URL http://dx.doi.org/10.1080/00207543.2011.636386.

Palubeckis, G. (2015). Fast local search for single row facility layout. *European Journal of Operational Research*, 246(3):800 – 814. ISSN 0377-2217. URL http://www.sciencedirect.com/science/article/pii/S037722171500466X.

Palubeckis, G. (2017). Single row facility layout using multi-start simulated annealing. *Computers & Industrial Engineering*, 103:1 – 16. ISSN 0360-8352. URL http://www.sciencedirect.com/science/article/pii/S0360835216303667.

Picard, J.-C. e Queyranne, M. (1981). On the one-dimensional space allocation problem. *Operations Research*, 29(2):371–391. URL http://dx.doi.org/10.1287/opre.29.2.371.

Samarghandi, H., Taabayan, P., e Jahantigh, F. F. (2010). A particle swarm optimization for the single row facility layout problem. *Computers & Industrial Engineering*, 58(4):529 – 534. ISSN 0360-8352. URL http://www.sciencedirect.com/science/article/pii/S0360835209003027.

Simmons, D. M. (1969). One-dimensional space allocation: An ordering algorithm. *Operations Research*, 17(5):812–826. URL http://dx.doi.org/10.1287/opre.17.5.812.