Bernardo De Polli Cellin

Métodos para Resolução eficiente de Problemas de Layout

Vitória, ES

2017

Bernardo De Polli Cellin

Métodos para Resolução eficiente de Problemas de Layout

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Informática da Universidade Federal do Espírito Santo, como requisito parcial para obtenção do Grau de Mestre em Informática.

Universidade Federal do Espírito Santo – UFES

Centro Tecnológico

Programa de Pós-Graduação em Informática

Orientador: Prof. Dr. André Renato Sales Amaral

Vitória, ES

2017

Bernardo De Polli Cellin
Métodos para Resolução eficiente de Problemas de Layout/ Bernardo De Polli Cellin. – Vitória, ES, 2017-
72 p. : il. (algumas color.) ; 30 cm.

Orientador: Prof. Dr. André Renato Sales Amaral

Dissertação de Mestrado – Universidade Federal do Espírito Santo – UFES
Centro Tecnológico
Programa de Pós-Graduação em Informática, 2017.

1. Otimização Combinatória. 2. Meta-heurística.

CDU 02:141:005.7

Bernardo De Polli Cellin

Métodos para Resolução eficiente de Problemas de Layout

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Informática da Universidade Federal do Espírito Santo, como requisito parcial para obtenção do Grau de Mestre em Informática.

Vitória, ES, 30 de março de 2017:

Prof. Dr. André Renato Sales Amaral
Orientador

Professor
Renato Elias Nunes de Moraes

Professor
Jorge Pinho de Sousa

Vitória, ES
2017

Agradecimentos

Agradeço primeiramente aos meus pais, José e Angela, e minha irmã, Laurana, por acreditarem em mim e incentivarem minhas escolhas em todos os momentos, tornando possível a realização deste trabalho.

Agradeço ao Professor André Renato Sales Amaral, meu orientador, pelo acompanhamento ao longo de todo o curso de mestrado.

Agradeço à Fundação de Amparo à Pesquisa e Inovação do Espírito Santo (FAPES) pelo importante apoio financeiro.

Agradeço à Rayanne pelo apoio durante a realização deste trabalho sendo uma constante fonte de carinho e motivação.

Finalmente, agradeço a todos que diretamente ou indiretamente contribuíram para a realização deste trabalho.

Resumo

Em sistemas produtivos onde a diminuição dos custos de produção é vista como peça chave na estratégia competitiva, a otimização entra como uma importante ferramenta para auxiliar as empresas nesse processo de sobrevivência e expansão no mercado. Na indústria surgem muitos problemas de Layout, os quais são problemas de otimização que se caracterizam pelo arranjo físico de facilidades ao longo de uma determinada área, formando um Layout. Problemas de Layout são difíceis e complexos de serem resolvidos do ponto de vista computacional. Por motivação de origem econômica e acadêmica, os problemas de Layout vêm sendo estudados há décadas. Esta dissertação propõe métodos heurísticos e híbridos para a resoluções da alguns destes problemas encontrados na literatura, como os Layouts de facilidades em duas ou mais linhas paralelas. Mais especificamente, são implementados métodos baseados nas meta-heurísticas Simulated Annealing, Variable Neighborhood Search e Iterated Local Search para resolver tais problemas. Além de meta-heurísticas, também é utilizado um modelo de Programação Linear.

Palavras-chaves: Problemas de Layout. Simulated Annealing. VNS. Meta-heurísticas. Otimização Combinatória.

Abstract

In production systems where a decrease in production costs is seen as a key part of the competitive strategy, Optimization is an important tool to assist companies in this process of survival and expansion in the market. There are many Layout problems in Industry, which are optimization problems that are characterized by the physical arrangement of facilities along a given area, forming a layout. Layout problems are difficult and complex to solve, from a computational point of view. For economic and academic reasons, Layout Problems have been studied for decades. This dissertation proposes heuristic and hybrid methods for solving some of these problems found in the literature, such as Facility Layout Problems on two or more parallel lines. More specifically, methods based on the meta-heuristics Simulated Annealing, Variable Neighborhood Search and Iterated Local are implemented to solve such problems. In addition to meta-heuristics, a Linear Programming model is also used.

Keywords: Layout Problems. Simulated Annealing. VNS. Metaheuristics. Combinatorial Optimization.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

SRFLP	Single Row Facility Layout Problem
SREFLP	Single Row Equidistant Facility Layout Problem
FMS	Flexible Manufacturing Systems
AGV	Automated Guided Vehicle
DRFLP	Double-row Facility Layout Problem
CAP	Corridor Allocation Problem
PROP	Parallel Row Ordering Problem
MRFLP	Multi-row Facility Layout Problem
SF-MRFLP	Space-free Multi-row Facility Layout Problem
VNS	Variable Neighborhood Search
SA	Simulated Annealing
ELS	Enhanced Local Search
QAP	Quadratic Assignment Problem
PSO	Particle Swarm Optimization
ILS	Iterated Local Search

Sumário

1 Introdução

1.1 O que são e por que estudar problemas de layout

Problemas de Layout possuem incontáveis aplicações práticas na indústria e podem ser caracterizados pelo arranjo físico de facilidades ao longo de uma determinada área, formando um Layout. Facilidades tais que podem ser máquinas, centros de trabalho, células de manufatura, departamentos de um edifício, armazéns, robôs em sistemas de manufatura, equipamentos e pessoal de produção ou até mesmo cilindros de um disco magnético (HERAGU, 1997; NEARCHOU, 2006; PICARD; QUEYRANNE, 1981).

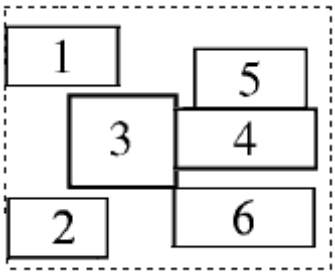
Definir o arranjo físico é decidir onde colocar todas as instalações, máquinas, equipamentos, pessoal da produção, enfim, todas as facilidades, para que haja um fluxo harmonioso dos recursos do processo, evitando, dessa forma, os desperdícios de produção (OLIVEIRA et al., 2014 apud SLACK, 2006).

De acordo com (BRITO; LOPES, 2014 apud MARQUES, 2012), é necessário considerar os seguintes aspectos no momento de planejar um arranjo físico:

- Especificações gerais como espaços necessários, distâncias a serem cobertas e circulação de pessoas ou materiais;
- Estimativa de demanda de cada produto no sistema;
- Requisitos de processo, quantidade de operações, fluxos entre os vários componentes;
- Necessidade de espaços para a operação e manutenção dos equipamentos e tarefas;
- Disponibilidade de espaço para possíveis novas configurações;

De modo geral, o objetivo do problema é justamente decidir a localização física de cada facilidade no layout de modo a minimizar uma função de custo, que pode ser calculada com base nas distâncias entre as facilidades e os fluxos entre as mesmas, sejam eles de pessoas ou materiais. Essa é uma das primeiras definições dessa classe de problemas, encontrada em (KOOPMANS; BECKMANN, 1955). Outros autores definiram de forma similar, como Meller, Narayanan e Vance (1998), que definem o problema de layout como encontrar localizações de centros de facilidades que não se sobrepõem em um arranjo ortogornal de n facilidades retangulares em um local com o objetivo de minimizar uma função de custo; e os autores Azadivar e Wang (2000), que consideram o problema como a determinação de posições relativas e de alocação de espaço para um dado número de facilidades.

Figura 1 – Exemplo de um Problema de Layout



Há autores que consideram outros parâmetros no problema de layout, como Lee e Lee (2002), que levam em consideração a área total alocada para o layout. Neste caso o problema passa a ser multiobjetivo, com a finalidade de diminuir simultaneamente o custo total e a área ocupada ou área livre.

No caso de um Problema de Layout com área delimitada e várias facilidades com áreas diferentes, um layout viável pode ser mostrado pela Figura 1.

Nas atuais circunstâncias de um mundo globalizado, as empresas estão sujeitas a constantes mudanças de mercado como a crescente diversificação e personalização de produtos, demandas flutuantes e imprevisíveis, flutuações do mercado financeiro e mudanças no clico de vida dos produtos. Para se manterem competitivas e sobreviver ao mercado, elas necessitam aumentar a produtividade e diminuir custos provenientes de manutenção e manufatura, ou seja, aumentar a eficiência (BENJAAFAR; SHEIKHZADEH, 2000).

Tais fatores podem ser alcançados com a utilizaçao de layouts de facilidades otimizados na indústria (DRIRA; PIERREVAL; HAJRI-GABOUJ, 2007). O estudo do layout está diretamente ligado à programação e controle da produção e aos custos de produção, por meio do melhor aproveitamento do espaço físico, melhoria das condições de trabalho e gerência no fluxo de materiais e pessoas, determinando, assim, o grau de eficiência dos recursos produtivos (OLIVEIRA et al., 2014 apud SLACK, 2006).

Tompkins et al. (1996) faz uma aproximação afirmando que um bom layout de facilidades pode levar a uma redução de até 50% de todo o custo operacional.

Além das motivações de origem econômica, o problema de Layout vem sendo estudado por décadas porque é computacionalmente difícil e complexo de ser resolvido. De acordo com as classificações de complexidade computacional, a classificação dos problemas de layout, em geral, é NP-hard (GAREY; JOHNSON, 1979).

Dizer que o Problema de Layout é NP-hard significa que ele é bastante sensível à entrada, isto é, o problema exige cada vez mais recursos computacionais à medida que aumenta-se o número de facilidades a serem arranjadas no layout. O problema pode

até se tornar inviável de ser resolvido com métodos exatos quando esse valor se torna grande o suficiente para exigir tempos de execução na ordem de séculos ou milênios. Resolver o problema de forma exata significa encontrar a melhor solução possível em tempo determinado (OLIVEIRA, 2012), ou seja, um layout com o menor custo total possível.

Em casos em que resolver um problema da classe NP-hard de forma exata se torna inviável, o que vem sendo utilizado é a implementação de heurísticas. Um algoritmo heurístico apoia-se no conhecimento de um problema específico para construir uma estratégia eficiente para percorrer o conjunto de soluções possíveis. Os métodos heurísticos não garantem a otimalidade de uma solução, no entanto, muitas vezes fornecem soluções subótimas de boa qualidade com custo computacional factível (TALBI, 2009).

De modo mais geral, existem as meta-heurísticas, que são heurísticas, ou métodos abstratos, que podem ser aplicados a diversos tipos de problema a fim de encontrar eficientemente boas soluções subótimas. Glover e Kochenberger (2003) definem metaheurísticas como métodos de solução que coordenam procedimentos de busca locais com estratégias de mais alto nível, de modo a criar um processo capaz de escapar de mínimos locais e realizar uma busca robusta no espaço de soluções de um problema. Neste trabalho utiliza-se meta-heurística para encontrar layouts viáveis para os problemas de layout que foram estudados e que serão apresentados nas seções a seguir.

São vários os Problemas de Facilidades que podem ser encontrados na literatura. Nas próximas seções serão apresentados alguns deles.

1.2 Problemas de Facilidade em Linha Única

O Problema de Ordenação de Facilidades em linha única, ou Single-row Facility Layout Problem (SRFLP), considera a ordenação de várias facilidades em uma única linha, de modo a minimizar os custos totais.

A formulação do problema é feita levando-se em conta a distância d_{ij} entre os centros de cada par distinto de facilidades i e j e o fluxo total c_{ij} entre as mesmas. Sendo Π o conjunto de todas as permutações π de $N = \{1, 2, ..., n\}$, a função de custo resultante deste problema de minimização pode ser visualizada na Eq. (1.1) (AMARAL; LETCHFORD, 2013). Cada facilidade do layout possui seu comprimento $l_i, ..., l_n$ conhecido previamente.

$$\min_{\pi \in \Pi} \sum_{(i,j) \subset N} c_{ij} d_{ij}^{\pi} \tag{1.1}$$

Normalmente, a configuração de Layout de Linha Única é feita de modo linear, em uma única linha reta. Há casos, porém, em que o Layout assume uma forma semicircular, em serpentina ou em forma de U (KELLER; BUSCHER, 2015 apud HASSAN, 1994).

Um caso especial do SRFLP é o Single-row Equidistant Facility Layout Problem (SREFLP), no qual as facilidades são todas de mesmo tamanho (PALUBECKIS, 2012).

Exemplos de layouts do SRFLP e SREFLP podem ser vistos na Figura 3 em (a) e (b).

O SRFLP possui um grande número de aplicações práticas como o arranjo de departamentos, quartos ou lojas em um corredor de um supermercado, hospital ou edifício (SIMMONS, 1969); o arranjo de livros em uma prateleira (AMARAL, 2006); o posicionamento de arquivos em cilindros de discos magnéticos (PICARD; QUEYRANNE, 1981); a ordenação de máquinas em Sistemas Flexíveis de Manufatura, ou Flexible Manufacturing Systems (FMS), ao longo de um caminho reto percorrido por Veículos Automáticos Guiados, ou Automated Guided Vehicles (AGVs) (HERAGU; KUSIAK, 1988); o escalonamento de aviões em portões de embarque (SURYANARAYANAN; GOLDEN; WANG, 1991); ordenação em layouts de armazéns (PICARD; QUEYRANNE, 1981).

O SRFLP contém o conhecido Problema do Arranjo Linear Mínimo (MinLA) em um caso especial obtido quando l_i é igual a 1 para todas as facilidades $i \in N$ e $c_{ij} \in \{0, 1\}$ para todo $i, j \in N$ (AMARAL; LETCHFORD, 2013). Como o MinLA é NP-hard (GAREY; JOHNSON, 1979), percebe-se que o SRFLP também se encaixa nessa categoria de complexidade.

1.3 Problemas de Facilidade em Linha Dupla

Uma extensão do SRFLP é o Problema de Facilidade em Linha Dupla, ou Double-row Facility Layout Problem (DRFLP), que considera a ordenação de facilidades em duas linhas paralelas. Assim como o SRFLP, o Double-row estuda a ordenação de facilidades com o objetivo de minimizar os custos totais.

Sua formulação é parecida com a do SRFLP. De modo geral, considerando um conjunto de facilidades $N = 1, 2, ..., n$ com seus respectivos comprimentos $\{li, ..., ln\}$, um fluxo não negativo c_{ij} entre cada par $i, j \in N$, a distância mínima entre o par d_{ij} e a coordenada do centro de cada facilidade x_i, o problema busca a permutação que minimiza a função de custo total que pode ser visualizada na Eq. 1.2. Na função objetivo há uma multiplicação entre o fluxo e a diferença absoluta entre as coordenadas dos centros de cada par de facilidades. A restrição 1.3 certifica-se de que não haverão sobreposições no layout e de que as distâncias mínimas entre facilidades serão respeitadas. Esta formulação pode ser encontrada em (HERAGU; KUSIAK, 1991).

$$\min \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} c_{ij}|x_i - x_j| \tag{1.2}$$

$$\text{sujeito a: } |x_i - x_j| \geq \frac{(l_i + l_j)}{2} + d_{ij}, i = 1, ..., n-1, j = i+1, ..., n \tag{1.3}$$

De modo geral, trabalhos encontrados na literatura tratam a distância entre os centros de cada par de facilidades com base somente no eixo horizontal. Por isso, pode-se observar o componente do eixo da abscissa aparecendo unicamente na equação.

Além do Double-row Facility Layout Problem, há outros problemas de ordenação em duas linhas paralelas, como o Problema de Alocação em Corredor, ou Corridor Allocation Problem (CAP), e o Problema de Ordenação em Linhas Paralelas, ou Parallel Row Ordering Problem (PROP). A diferença entre eles é que no CAP e no PROP não há espaços entre as facilidades dispostas em uma mesma linha e todas as alocações de facilidades nas linhas iniciam-se de um ponto comum (AMARAL, 2012); já no DRFLP, pode haver espaços entre as facilidades (AMARAL, 2013a). O PROP ainda difere do CAP por ter uma característica especial na ordenação que é a alocação prévia de facilidades em ambas as linhas, permitindo trocas entre facilidades somente em uma mesma linha.

O espaço considerado pelo DRFLP entre essas duas facilidades é chamado na literatura de *clearance*. As clearances são utilizadas em aplicações práticas nas quais máquinas precisam de um espaço para manutenção ou operação. Essa característica faz com que o DRFLP seja de natureza contínua, diferentemente do CAP, SRFLP e PROP.

Exemplos de layouts do DRFLP e CAP podem ser vistos na Figura 3.

O DRFLP é relevante no contexto prático na indústria por otimizar fluxo de material e uso de espaço em situações em que há máquinas dos dois lados de um caminho percorrido por um Veículo Automático Guiado em um Sistema Flexível de Manufatura (ANJOS; FISCHER; HUNGERLÄNDER, 2015). O CAP tem aplicações na ordenação de departamentos em edifícios comerciais, hospitais, shoppings e escolas (AMARAL, 2012), e no arranjo de layouts nos quais facilidades devem ser divididas em duas linhas e posicionadas ao longo de uma disposição em forma de espinha, como pode ser notado em instalações de fabricação de semicondutores (AHONEN; DE ALVARENGA; AMARAL, 2014 apud YANG; PETERS, 1997). Uma aplicação prática para o PROP é o arranjo de departamentos previamente alocados em andares de um edifício (AMARAL, 2013b).

1.4 Problemas de Facilidade em Múltiplas Linhas

Mais geral que os problemas de layout em linha discutidos anteriormente é o Problema de ordenação de Facilidades em múltiplas linhas, ou Multi-row Facility Layout Problem (MRFLP). Este problema considera um conjunto de máquinas retangulares, um determinado número de linhas e os custos de fluxo definidos entre pares distintos de facilidades. O objetivo do MRFLP é posicionar todas as facilidades nas linhas de forma a

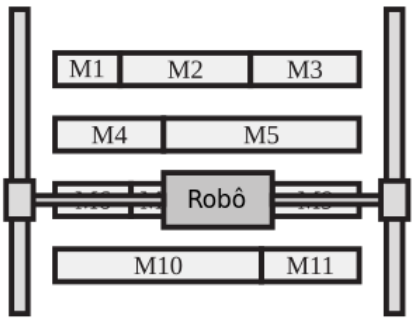
minimizar uma função de custo total.

A formulação mostrada na equação 1.2 e restrição 1.3 também se encaixam para o problema de ordenação de facilidades em múltiplas linhas. Ainda sujeito às restrições de não sobreposição e respeito às distâncias mínimas entre facilidades vizinhas, a função objetivo do MRFLP é a multiplicação dos fluxos pelas distâncias entre centros de cada par de facilidade do layout. Assim como o Double-row, no cálculo das distâncias entre os centros das facilidades pode-se generalizar e levar em conta somente as distâncias horizontais, do eixo das abscissas. Há autores, porém, que consideram o eixo y e utilizam outras métricas como (HERAGU, 1992).

Um caso particular do MRFLP é k-PROP que, assim como o PROP, considera n departamentos alocados previamente nas linhas do layout, todavia o posicionamento é feito em não apenas duas linhas mas k linhas paralelas (AMARAL, 2013b). Outra variação do MRFLP é o Problema de Facilidades em Múltiplas Linhas sem Espaçamento, ou Space-free Multi-row Facility Layout Problem (SF-MRFLP), que considera um conjunto de departamentos que deve ser arranjado em múltiplas linhas de forma a não permitir espaços entre os mesmos (HUNGERLÄNDER; ANJOS, 2015). Layouts viáveis do MRFLP e SF-MRFLP podem ser vistos na Figura 3.

Na literatura podem-se encontrar várias aplicações gerais para o MRFLP como o problema da disposição de componentes em computadores, ou Computer Backboard Wiring Problem (STEINBERG, 1961), Layouts com Múltiplos Andares com Elevador (Multi-floor Layout Problem) (KOCHHAR; HERAGU, 1998), planejamento de Campus (DICKEY; HOPKINS, 1972), Scheduling (GEOFFRION; GRAVES, 1976), design de teclados (POLLATSCHEK; GERSHONI; RADDAY, 1976), Layout de Hospitais (ELSHAFEI, 1977), Balanceamento de Rotores Hidráulicos de Turbinas (LAPORTE; MERCURE, 1988), Análise Numérica (BRUSCO; STAHL, 2000) e Geração de Layouts de Memória em Processadores Digitais de Sinal (WESS; ZEITLHOFER, 2004). Assim como aplicações na indústria, como em Layout de Máquinas em Sistemas Automatizados de Manufatura (HERAGU; KUSIAK, 1988), Layout de Facilidades com áreas diferentes (LEE; LEE, 2002) e em Sistemas Flexíveis de Manufatura (FMS), nas quais operam AGVs ou Gantry Robots (NEARCHOU, 2006). A Figura 2 mostra como seria um Layout de uma instância do MRFLP considerando um *Gantry Robot* posicionado acima das máquinas controlando o fluxo de material entre essas máquinas.

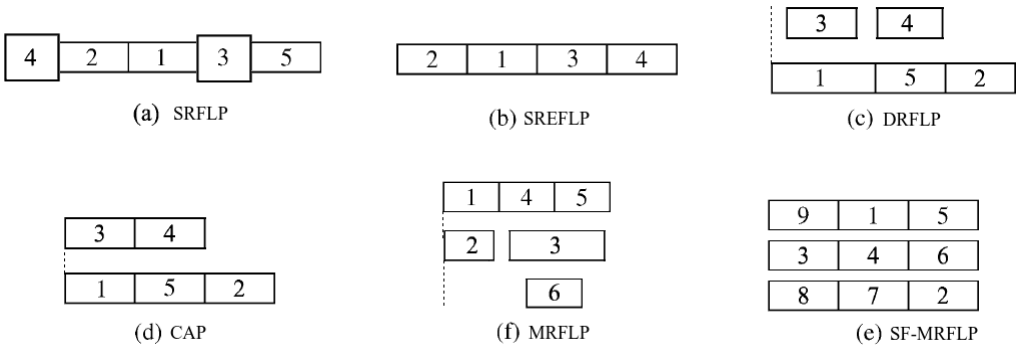
Figura 2 – Gantry Robot em um FMS



Fonte: (HUNGERLÄNDER; ANJOS, 2015)

As características e diferenças de cada um dos Problemas discutidos podem ser visualizadas nos exemplos da Figura 3.

Figura 3 – Exemplos de Problemas de Layouts



Devido às aplicações na indústria e motivações científicas, este trabalho tem como foco resolver alguns dos problemas de layout de facilidades, mais especificamente o Problema de Alocação em Corredor (CAP), o Problema de Ordenação em Linhas Paralelas (PROP), o Problema de ordenação de Facilidades em múltiplas linhas (MRFLP) e o Problema de Facilidades em Múltiplas Linhas sem Espaçamento (SF-MRFLP). Tais problemas não possuem grande número de trabalhos publicados na literatura, sendo assim escolhidos para serem estudados ao longo do curso de mestrado.

No capítulo 2 é feita uma breve revisão da literatura, mostrando trabalhos feitos para alguns dos problemas de layout discutidos. No capítulo 3 são apresentadas as meta-heurísticas utilizadas na resolução dos problemas abordados, as quais são Simulated Annealing (SA), Variable Neighborhood Search (VNS) e Iterated Local Search (ILS).

Nos capítulos seguintes (4, 5 e 6) são apresentados os métodos e algoritmos utilizados para resolver cada um dos problemas propostos; sempre com resultados computacionais exibidos ao final de cada capítulo.

Por fim, no capítulo 7 são feitas conclusões sobre os resultados obtidos pelos algoritmos e sobre o que foi feito no trabalho, assim como previsões para trabalhos futuros.

2 Revisão da Literatura

Pesquisas sobre problemas de facilidades têm utilizado abordagens exatas ou meta-heurísticas.

O Single Row Facility Layout Problem (SRFLP) foi resolvido de forma exata em vários estudos como, por exemplo, com Branch-and-bound por Simmons (1969); Programação Dinâmica por Picard e Queyranne (1981); Programação Linear Inteira Mista, ou Mixed Integer Programming (MIP) por Heragu e Kusiak (1991), Amaral (2006), Amaral (2008b), Love e Wong (1976); método de planos de corte (AMARAL, 2009b), revisitados por um estudo poliedral em (SANJEEVI; KIANFAR, 2010); combinação de relaxação de programação semidefinida com planos de corte em (ANJOS; VANNELLI, 2008); programação semi-definida em (HUNGERLÄNDER; RENDL, 2013) e Branch-and-cut em (AMARAL; LETCHFORD, 2013). Para o SREFLP, tem-se Branch-and-bound em (PALUBECKIS, 2012) e Programação Semi-definida em (HUNGERLÄNDER, 2014).

Abordagens heurísticas têm sido usadas para o SRFLP. De Alvarenga, Negreiros-Gomes e Mestria (2000) propuseram metaheurísticas baseadas em Busca Tabu e Simulated Annealing (SA); Amaral (2008a) propôs um método heurístico chamado Enhanced Local Search (ELS), que faz uso de vários tipos de buscas locais para o problema, conseguindo soluções subótimas para instâncias de até 30 facilidades; Amaral (2009a) resolveu o clássico Problema do Arranjo Linear Mínimo, o qual está contido no SRFLP, com um modelo de Programação Linear Mista 0-1; Ozcelik (2012) usou uma hibridização de Algoritmos Genéticos com buscal local e Datta, Amaral e Figueira (2011) propuseram um Algoritmo Genético.

Kothari e Ghosh utilizaram as seguintes estratégias heurísticas: implementaram uma heurística chamada de LK-INSERT, que utiliza uma eficiente estrutura de vizinhança Lin-Kernighan para movimentos de inserções em (KOTHARI; GHOSH, 2013b), propuseram uma implementação de Algoritmo Genético em (KOTHARI; GHOSH, 2013a), propuseram a implementação do algoritmo Scatter Search em (KOTHARI; GHOSH, 2014) e, em (KOTHARI; GHOSH, 2013c), apresentaram implementações eficientes da busca tabu. Recentemente, Palubeckis (2015a) trata de heurísticas de vizinhança variável, ou Variable Neighbohood Search (VNS), para resolver o SRFLP com instâncias de tamanho de até 300 facilidades em tempo razoável; Guan e Lin (2016) propuseram um algoritmo híbrido que faz uso as meta-heurísticas VNS e Colônia de Formigas com eficientes estruturas de vizinhança e Palubeckis (2017) desenvolveu um algoritmo Simulated Annealing com Multi-start que consegue resolver instâncias de até 1000 facilidades.

Já para o Single Row Equidistant Facility Layout Problem (SREFLP), Palubeckis

(2015b) utilizou Simulated Annealing com eficiente estrutura de vizinhança para resolver instâncias de até 300 facilidades.

O Corridor Allocation Problem (CAP) já foi resolvido por Amaral (2012), com o uso de Programação Inteira Mista; por Ghosh e Kothari (2012) com uso de Algoritmos Genéticos com Buscal Local e Scatter Search com Path Relinking; por Hungerländer e Anjos (2012) com otimização semidefinida e heurísticas para obter soluções viáveis de relaxação de programação semidefinida e por Ahonen, De Alvarenga e Amaral (2014), com Simulated Annealing e Busca Tabu.

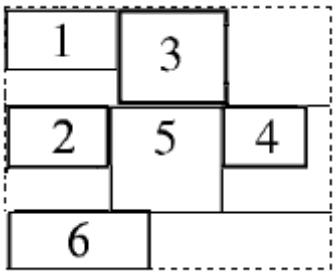
O Double Row Facility Layout Problem (DRFLP) foi considerado por Chung e Tanchoco (2010), que propuseram um modelo de programação inteira mista para o problema. Pouco tempo depois, Zhang e Murray (2012) publicaram um trabalho corrigindo este modelo. Amaral (2011) propôs uma formulação de programação linear inteira-mista para resolver o problema Minimum Duplex Arrangement, ou MinDA, que é equivalente a um caso especial do DRFLP no qual todas as facilidades possuem tamanhos e áreas iguais. Murray, Smith e Zhang (2013) utilizaram uma combinação de heurísticas construtivas e de busca local que fazem uso de soluções obtidas por um modelo de programação inteira mista para máquinas da mesma linha com matriz de distâncias mínimas assimétrica. Hungerländer e Anjos (2015) resolveram o SF-DRFLP, ou Space-free Double Row Layout Problem, um caso particular do DRFLP. Permanhane (2016) resolveu o DRFLP utilizando meta-heurística ILS para fazer o posicionamento relativo nas linhas e um Programa Linear para fazer o posicionamento das facilidades no eixo horizontal.

Sobre o k-Parallel Row Ordering Problem (k-PROP), uma formulação de programação inteira foi proposta por Amaral (2013b) e uma meta-heurística Simulated Annealing foi apresentada por Cellin e Amaral (2015) para $k = 2$. Hungerländer e Anjos (2015) usaram um modelo de relaxação semidefinida para o k-PROP com espaços entre os departamentos. Durante a atribuição de facilidades no PROP, os autores fizeram um balanceamento do número de departamentos atribuídos a cada linha do layout. Em seus testes eles consideram 2<= k <= 5.

Para o Multi-row Facility Layout Problem (MRFLP), muitos trabalhos já foram publicados, como pode ser visto em Surveys sobre Problemas de Layout de Facilidades como (DRIRA; PIERREVAL; HAJRI-GABOUJ, 2007), (JAIN; KHARE; MISHRA, 2013) e (MAVRIDOU; PARDALOS, 1997).

Modelos de MRFLP em que máquinas são separadas exatamente pela distância mínima, normalmente são resolvidos com modelos de Problemas Quadráticos de Alocação, ou Quadratic Assignment Problem (QAP), como em (SINGH; SHARMA, 2008). Além disso, alguns autores não consideram distâncias mínimas entre linhas, como as formulações QAP de HASSAN (1994) e Tate e Smith (1995), nas quais facilidades com mesma área são alocadas de modo prévio e as linhas não precisam ser uniformes. Heragu e Kusiak

Figura 4 – Layout com áreas diferentes



(1988) afirmaram que modelos QAP não podem ser usados para layouts facilidades com áreas desiguais e propuseram dois algoritmos construtivos para resolver o problema de múltiplas linhas.

Heragu e Alfa (1992) propuseram uma hibridização envolvendo Simulated Annealing para Layouts em Múltiplas Linhas de mesma área. Heragu (1992) apresentou formulações para o Multi-row Facility Layout Problem com áreas iguais e diferentes considerando a distância retilinear entre os centros de facilidades, com expressões com valores absolutos na função objetivo e restrições. Esse modelo não considerou corredores entre as linhas. El-Baz (2004) utilizou a meta-heurística Algoritmo Genético para resolver vários tipos de problemas de layouts, inclusive o Problema de Layout com múltiplas linhas com áreas iguais. O autor considera a distância entre centros retilinear no eixo x e y.

Lee e Lee (2002) utilizaram uma hibridização com Algoritmos Genéticos, Simulated Annealing e Busca Tabu para resolver o problema de múltiplas linhas com facilidades de áreas heterogêneas. Neste trabalho, a função objetivo era calculada em função da distância retilinear entre os centros de pares distintos de facilidades multiplicada pelos fluxos totais entre as mesmas, acrescido do valor de desperdício de área total do layout. Um layout válido para esse trabalho pode ser visualizado na Figura 4.

Ficko, Brezocnik e Balic (2004) propuseram um Algoritmo Genético para resolver o MRFLP baseando-se em um sistema FMS com um robô AGV que circula em corredores entre as linhas. As áreas das máquinas são diferentes e as distâncias consideradas são retilineares. A distância centro-a-centro no eixo y leva em consideração a altura da linha e a separação mínima entre linhas. Nesse trabalho só foi usada uma única instância para testar o algoritmo.

Miao e Xu (2009) propuseram uma hibridização de Algoritmo Genético com Busca Tabu para resolver um modelo para o MRFLP. Nesta modelagem, foram assumidas as seguintes hipóteses: o espaço de alocação total e cada facilidade possuem forma retangular com comprimentos e larguras conhecidos, cada facilidade é alocada de forma paralela ao eixo horizontal, as distâncias verticais e horizontais entre facilidades vizinhas são conhecidas e facilidades que estejam em uma mesma linha possuem mesma coordenada no

eixo y.

Jankovits et al. (2011) resolveram o Problema de Layout com áreas desiguais. Neste problema, o objetivo é organizar vários departamentos retangulares em um Layout de forma a minimizar a função de custo total, que considera as distâncias de centro-a-centro e um vetor de custos de fluxo. Nesse trabalho, encontrar a largura e comprimento dos departamentos fazia parte do problema. A abordagem feita foi uma modelagem com otimização convexa e também uma modelagem com relaxação semi-definida. Tasadduq, Imam e Ahmad (2011) utilizaram a meta-heurística Algoritmo Memético, que envolve buscas locais no procedimento do Algoritmo Genético. Com essa meta-heurística resolveram o FLP com áreas desiguais. Compararam as soluções encontradas com um algoritmo comercial chamado VIP-PLANOPT conseguindo encontrar melhores resultados em algumas instâncias.

Ficko et al. (2010) fizeram uma implementação eficiente da meta-heurística Particle Swarm Optimization (PSO) para encontrar soluções para o problema de layout sem restrições de linhas em um FMS, no qual as facilidades podem estar dispostas de qualquer forma no layout. O algoritmo calcula as posições das facilidades em um layout tendo como base as dimensões das facilidades, os seus pontos de serviço e os custos de fluxos entre os mesmos. Dependendo das dimensões das facilidades de entrada, o layout inicial pode ficar grande. Nesse trabalho, foram implementadas formas de penalizar soluções com caminhos bloqueados ou inválidos entre as facilidades.

Relacionado ao MRFLP, o Multiple-floor Layout Problem foi discutido primeiramente por (JOHNSON, 1982 apud JAIN; KHARE; MISHRA, 2013). O problema envolve um layout com múltiplos andares e várias linhas de facilidades. Johnson (1982) resolveu o problema usando uma heurística de um estágio chamada SPACECRAFT. Em (BOZER; MELLER; ERLEBACHER, 1994) e (MELLER; BOZER, 1996) o problema é abordado com heurísticas. Meller e Bozer (1997) mostraram a diferença entre heurísticas de um estágio e dois estágios e desenvolveram em seu trabalho uma heurística de dois estágios baseada em Simulated Annealing. Posteriormente, Lee, Roh e Jeong (2005) usaram Algoritmo Genético para resolver o problema. Em seu trabalho, as posições dos elevadores são determinadas previamente, assim como o número de andares que serão usados.

Hungerländer e Anjos (2012) usaram relaxação semi-definida para encontrar soluções ótimas relaxadas, e heurísticas para torná-las viáveis para o Space-free Multi-row Facility Layout Problem (SF-MRFLP). Já em (ANJOS; FISCHER; HUNGERLÄNDER, 2015) propuseram dois modelos para o SF-MRFLP: um de programação linear inteira e um de otimização semi-definida, para resolver o problema de múltiplas linhas com facilidades de mesmo tamanho mostrando resultados com até 5 linhas paralelas.

Continuando os estudos anteriores, Hungerländer e Anjos (2015) fizeram uma extensão do modelo criado para o SF-MRFLP e propuseram uma nova modelagem para o

MRFLP, tratando-o como um problema discreto. Para o problema, eles consideram que cada máquina pode ser atribuída a qualquer linha, que todas as linhas possuem a mesma altura, que todas as distâncias entre linhas são iguais e que todos as facilidades possuem a mesma altura, igual à da linha. Usaram uma relaxação semi-definida e uma heurística para tornar as soluções viáveis para resolver não só o SF-MRFLP mas também problemas relacionados como o MRFLP, PROP e DRFLP.

Meta-heurísticas não têm sido muito utilizadas para problemas de facilidades com variáveis contínuas (HUNGERLÄNDER; ANJOS, 2015). O fato de que não há espaços entre máquinas e o fato das distâncias entre linhas serem fixos fazem com que problemas como o SF-MRFLP possuam apenas variáveis inteiras e binárias, tornando-os mais simplificados e aplicáveis para meta-heurísticas.

Para o PROP, este trabalho fez uso da meta-heurística Simulated Annealing como em (CELLIN; AMARAL, 2015) mas com diferenças no pseudocódigo. O algoritmo foi comparado com a hibridização entre VNS e ILS para instâncias maiores.

Para o CAP e SF-MRFLP, o Simulated Annealing também teve a adição de uma função chamada *SA_reverso*, que produz uma temperatura inicial para o algoritmo. Esta função pode ser encontrada no trabalho de Ahonen, De Alvarenga e Amaral (2014). Além do *SA_reverso*, o algoritmo SA implementado neste trabalho também teve a adição de uma rotina que aumenta a temperatura caso novas soluções não estejam sendo aceitas, algo que será discutido nas próximas sessões.

Já para o DRFLP e MRFLP, as meta-heurísticas implementadas foram utilizadas em conjunto com um Programa Linear, assim como foi feito em (PERMANHANE, 2016), o qual usou a meta-heurística Iterated Local Search (ILS) em conjunto com um Programa Linear. Os métodos utilizados também serão discutidos mais adiante no trabalho.

3 Meta-heurísticas Utilizadas

Os problemas de layout são difíceis e complexos de serem resolvidos. A sua resolução utilizando-se métodos exatos muitas vezes é inviável de ser utilizada na prática. Por isso, neste trabalho foram utilizadas heurísticas de construção e meta-heurísticas para conseguir boas soluções subótimas em tempo computacional viável. A seguir, serão discutidas as meta-heurísticas utilizadas nos problemas que são o foco deste trabalho. Tais meta-heurísicas são: Simulated Annealing, VNS e ILS. Essas meta-heurísticas foram escolhidas por terem sido estudadas e investigadas ao longo do curso de mestrado e por mostrarem bons resultados em trabalhos anteriores na literatura.

3.1 Otimização Combinatória

Problemas de otimização combinatória possuem um conjunto de elementos que podem ser selecionados e agrupados de acordo com regras estipuladas, e uma função de custo associada a cada um dos subconjuntos possíveis, normalmente chamada de função objetivo. O problema resume-se então a encontrar, dentre todos os subconjuntos possíveis, o que produz a melhor função objetivo. Caso o problema seja de maximização, a função objetivo deve ser a maior possível; caso seja um problema de minimização, a função objetivo deve ser a menor possível. Os problemas de layout normalmente são problemas de minimização, já que possuem o objetivo de minimizar a função de custo total.

Juntos, todos esses subconjuntos do problema formam todas as soluções possíveis, formando um espaço de soluções. Nem sempre todas as soluções são consideradas viáveis por causa das regras e restrições do problema, portanto procura-se encontrar soluções viáveis dentro desse espaço de soluções.

No processo de investigar o espaço por soluções viáveis, pode-se utilizar busca local. Russell e Norvig (2009) explicam que no procedimento de Busca Local, tendo o estado atual como referência, tenta-se encontrar soluções melhores em sua vizinhança. Quando uma solução melhor é encontrada, ela se torna a atual e o algoritmo continua a buscar soluções melhores em sua vizinhança. A vizinhança de uma solução é o conjunto de todas as soluções alcançáveis pela solução atual utilizando movimentos de troca em seus elementos.

Na busca local quando é encontrada uma solução atual que é a melhor entre as vizinhas, o algoritmo chega ao fim e fornece a solução encontrada como resposta. Muitas vezes essa melhor solução encontrada consiste no que é chamado de um ótimo local, e não é a melhor solução que poderia ser encontrada no espaço de soluções do problema,

chamada de ótimo global. Isso acontece porque o algoritmo fica preso naquele ponto por não conseguir encontrar vizinhos com solução melhor.

As meta-heurísticas são desenvolvidas para tentar escapar desses ótimos locais, fazendo buscas no espaço de soluções de forma inteligente.

3.2 Simulated Annealing

O Simulated Annealing (SA) é uma meta-heurística usada para resolver problemas de otimização combinatória. Esse método foi proposto originalmente por Kirkpatrick, Gelatt e Vecchi (1983) e desde então tem sido bastante aplicado em diversos tipos de problemas, inclusive problemas classificados como NP-hard, por ter implementação considerada rápida.

O SA baseia-se em uma analogia com a teoria do resfriamento de materiais da termodinâmica para otimizar soluções para esses problemas. Annealing, ou Recozimento, é um processo que consiste em aquecer um metal até o ponto de fusão e então resfriá-lo lentamente, permitindo que as moléculas alcancem uma configuração de baixa energia e formem uma estrutura com nível mínimo de energia (GLOVER; KOCHENBERGER, 2003).

Em (GLOVER; KOCHENBERGER, 2003) também é afirmado que se o resfriamento for suficientemente lento, a configuração final resulta em um sólido com alta integridade estrutural. O método computacional, o Simulated Annealing, estabelece uma conexão entre este comportamento termodinâmico e a busca pelo ótimo global de um problema de otimização.

Este método faz pequenas mudanças na solução iterativamente, buscando melhores soluções vizinhas. Assim como outras meta-heurísticas, uma característica 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 SA_{max} e a taxa de resfriamento α. A calibragem destes parâmetros normalmente é feita empiricamente. Na analogia com a teoria da termodinâmica, a função objetivo seria o nível de energia, a solução viável seria o estado do sistema, a solução vizinha seria uma mudança de estado, o parâmetro de controle seria a própria temperatura e a melhor solução seria o estado de solidificação.

O pseudocódigo pode ser visualizado no Algoritmo 1. A cada iteração, da linha 2 até a 11, o algoritmo diminui a temperatura atual de acordo com o fator de redução de temperatura α. Esse decaimento pode ser feito de forma linear, aritmético ou pode ser feito de forma geométrica, como é feito na linha 10 ($T \leftarrow \alpha T$).

A cada uma dessas iterações de temperatura, são realizadas várias perturbações na

solução, como na linha 3, que são trocas de estado. Essas perturbações levam a soluções consideradas vizinhas e podem produzir soluções melhores ou piores. Caso a solução vizinha obtida seja melhor do que a atual em memória, o algoritmo a aceita e faz com que essa solução seja a atual (linha 6). Caso contrário, o método também pode aceitar a solução, de acordo com uma probabilidade $e^{\frac{-\Delta}{T}}$, no qual Delta é a diferença de valor de função objetivo entre a solução obtida e a solução atual e T é a temperatura atual (linha 8). Isso faz com que o algoritmo possa aceitar soluções não desejáveis no momento para escapar de ótimos locais, conseguindo encontrar melhores soluções eventualmente.

A temperatura controla a convergência do algoritmo: quando ela está alta, a energia do sistema está elevada e várias soluções piores são aceitas, fazendo com que o algoritmo busque novas soluções em um grande espaço. Quando ela está próxima de zero, a energia do sistema está baixa e o algoritmo passa a aceitar somente soluções que melhorem, aproximando-se de uma busca local.

Algoritmo 1 Simulated Annealing

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

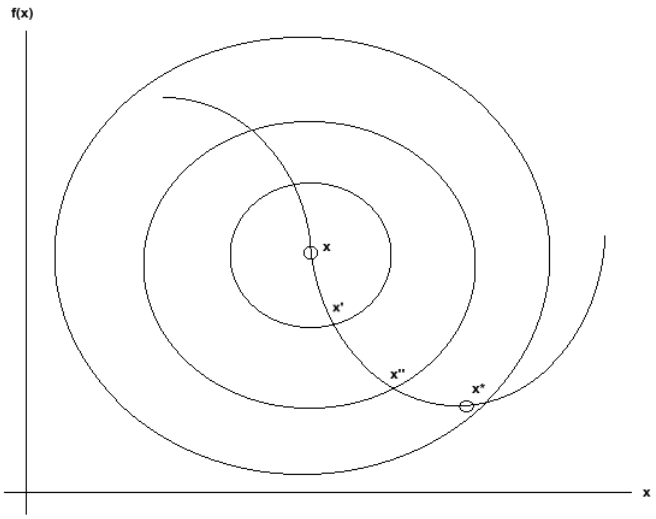
Quando comparado com outros métodos, o Simulated Annealing tem a implementação considerada simples. Por isso, consegue trazer bons resultados de forma rápida.

3.3 VNS

O Variable Neighborhood Search (VNS), foi proposto por Mladenović e Hansen (1997). O método consiste na realização de buscas locais no espaço de soluções e a ideia principal é fazer mudanças sistemáticas nas estruturas de vizinhanças para explorar de forma mais profunda o espaço de soluções e escapar de ótimos locais. Normalmente as meta-heurísticas aceitam a degradação da solução corrente para fugir de ótimos locais mas o VNS não trabalha com tal procedimento.

De acordo com Glover e Kochenberger (2003), o VNS baseia-se em três princípios

Figura 5 – Mudanças de vizinhança no VNS



básicos:

- Um ótimo local com relação a uma vizinhança não necessariamente corresponde a um ótimo com relação à outra vizinhança;
- Um ótimo global corresponde a um ótimo local para todas as estruturas de vizinhança;
- Frequentemente, ótimos locais relativos a estruturas de vizinhança semelhantes estão relativamente próximos.

O funcionamento do VNS básico pode ser observado na Figura 5. A solução inicial x não é ótima e pode ser melhorada. As mudanças na estrutura de vizinhança exploram o espaço de soluções de forma cada vez mais abrangente e que vão encontrando soluções cada vez melhores até estagnar em um suposto ponto ótimo, em x^*.

O algoritmo mostrado em (GLOVER; KOCHENBERGER, 2003) pode ser visualizado no Algoritmo 2. O algoritmo geral do VNS combina mudanças determinísticas e estocásticas na solução e vizinhança. Primeiramente o algoritmo recebe um conjunto de vizinhanças $N_k(x)$ e uma solução inicial x gerada por alguma heurística construtiva. Para cada iteração do laço interno, que vai da linha 3 até a 12, é feito o *shake*, que consiste em uma pequena perturbação na solução x, gerando uma solução x' de acordo com a estrutura de vizinhança atual, a qual é identificada por uma variável k no algoritmo. Esse passo é feito na linha 4 para evitar mínimos locais. Também no laço interno, na linha 5, é feita uma busca local na solução gerando um mínimo local x''. A busca local pode ser um algoritmo best-improvement ou first-improvement, por exemplo. Esta nova solução é comparada com a solução inicial na linha 6 e é aceita de acordo com critérios do algoritmo. Caso a solução seja aceita, a busca continua com $k = 1$; caso contrário, k é acrescido de

valor para apontar para outras vizinhanças no algoritmo. Esses passos se repetem em um laço externo de acordo com um critério de aceitação e vão da linha 1 até a 13.

Algoritmo 2 VNS

Entrada: conjunto de vizinhanças $N_k(x)$, solução inicial x
Saída: solução x

1: **repita**
2: $k \leftarrow 1$
3: **repita**
4: gerar uma solução x' através de uma vizinhança k de x, $x' \in N_k(x)$
5: aplicar uma busca local com x' como solução inicial, gerando x'' como ótimo local
6: **se** $f(x'') \leq f(x)$ **então**
7: $x \leftarrow x''$
8: $k \leftarrow 1$
9: **senão**
10: $k \leftarrow k + 1$
11: **fim se**
12: **até que** $k \leq k_{max}$
13: **até que** condição de parada seja satisfeita

O VNS tem extensões que possuem algumas diferenças em relação ao VNS básico. Algumas delas podem ser de verificadas em (MLADENOVIĆ; HANSEN, 1997):

- Variable Neighborhood Descent (VND): é uma busca local e extensão do VNS no qual ocorrem mudanças na vizinhança de modo determinístico. Normalmente utiliza vizinhanças menores antes de utilizar as maiores;
- Reduced VNS (RVNS): extensão que é obtida quando pontos são selecionados de forma totalmente aleatória, somente utilizando o *shake* para gerar novas soluções;
- Variable Neighborhood Decomposition Search (VNDS): extende o VNS básico em um esquema multi-nível baseado na decomposição do problema. A diferença é que na fase de busca local, ao invés de buscar soluções pelo espaço de soluções inteiro, cada iteração é resolvida em um subespaço de soluções;
- Skewed VNS (SVNS): é uma extensão no qual explora soluções longe das soluções correntes. Pode chegar ao ponto, no pior caso, de se comportar como uma heurística Multi-start para buscar soluções longe no espaço de soluções. Faz uso de uma função específica que mede a distância entre a solução corrente e mínimos locais encontrados;
- Parallel VNS (PVNS): uma extensão que transforma o VNS em um algoritmo paralelizável. Isso acontece com a paralelização da busca local simplesmente ou com o aumento de soluções obtidas da vizinhança atual, nas quais são aplicadas buscas locais em paralelo.

O VNS pode ser alvo de hibridizações e utilizado em combinação com outras metaheurísticas tais como o Busca Tabu, GRASP e *Constraint Programming*, por exemplo (MLADENOVIĆ; HANSEN, 1997).

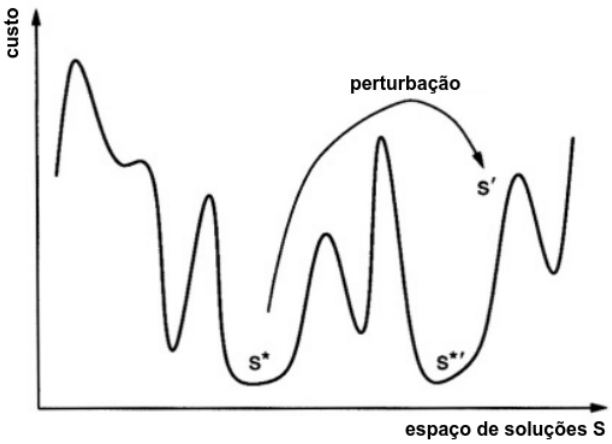
3.4 ILS

Iterated Local Search, ou ILS, é uma meta-heurística que tem o objetivo de, iterativamente, aplicar buscas locais e degradar a solução com perturbações (LOURENÇO; MARTIN; STÜTZLE, 2003). A busca local leva a um ótimo local e, assim como outras meta-heurísticas, o ILS tenta escapar destruindo a solução corrente com uma perturbação que seja capaz de levar a outras soluções no espaço de busca.

Dado um problema de otimização qualquer P e sendo S o conjunto com todas as soluções possíveis para P, uma heurística de busca local pode ser definida como um mapeamento muitos para um do conjunto S para o conjunto menor S* contendo todos os ótimos locais do problema (LOURENÇO; MARTIN; STÜTZLE, 2003). A ideia do ILS é a seguinte: dada a busca local L, tentar encontrar uma melhor solução, sem necessitar alterar a função de busca local.

O que o ILS faz é utilizar uma busca tendenciosa pelo espaço de soluções (CHIARANDINI; STÜTZLE, 2002). Isso é feito realizando-se uma perturbação em um ótimo local, obtendo $s' \in S$ e, logo depois, aplicando-se uma busca local no resultado da perturbação, obtendo $s*' \in S*$. A nova solução obtida deve então passar por um critério de aceitação para se tornar, ou não, a nova solução do problema. O funcionamento prático do ILS pode ser visto na Figura 6, na qual é mostrada uma perturbação em uma solução $s*$ chegando a uma nova solução $s*'$. A ideia principal do algoritmo é evitar ficar preso em ótimos locais durante a realização de buscas no espaço de solução.

Figura 6 – Funcionamento do ILS



Fonte: (LOURENÇO; MARTIN; STÜTZLE, 2003)

Para implementar o ILS é necessário definir os quatro blocos principais do algoritmo (CHIARANDINI; STÜTZLE, 2002):

- Geração da Solução inicial: que fornece uma solução inicial s_0;
- Perturbação: que recebe uma solução e a transforma em uma solução s';
- Busca Local: que recebe uma solução s' e a transforma em uma solução ótima local $s*'$;
- Critério de Aceitação: que decide em qual solução a perturbação será aplicada na próxima iteração.

Essas características podem ser visualizadas no pseudo-código do ILS, mostrado no Algoritmo 3.

Algoritmo 3 ILS

1: $s_0 \leftarrow$ solução inicial gerada
2: $s* \leftarrow$ busca_local(s_0)
3: **repita**
4: $\quad s' \leftarrow$ perturbação($s*$, memoria)
5: $\quad s*' \leftarrow$ busca_local(s')
6: $\quad s* \leftarrow$ critério de aceitação($s*$, $s*'$,memoria)
7: **até que** condição de parada seja satisfeita

Basicamente, a solução inicial na linha 1 pode ser gerada aleatoriamente ou com uma heurística construtiva. De acordo com Stützle (1998), a utilização de soluções aleatórias fazem com que o algoritmo demore mais para convergir do que com uma solução obtida por heurística.

A busca local encontada nas linhas 2 e 5 pode ser considerada como um dos blocos mais importantes, visto que é ela quem determina o comportamento e desempenho do algoritmo (LOURENÇO; MARTIN; STÜTZLE, 2003). A heurística escolhida para o bloco da busca local não precisa ser necessariamente uma busca local, podendo ser substituído, por exemplo, por uma meta-heurística como Busca Tabu ou Simulated Annealing, como visto em (LOURENÇO; ZWIJNENBURG, 1996).

O principal objetivo da perturbação, encontrada na linha 4, é escapar de ótimos locais. Dependendo da intensidade da perturbação, isto é, da quantidade de mudanças na estrutura da solução introduzidas pela perturbação, é possível diversificar ou intensificar a busca em uma determinada região do espaço de soluções. De maneira geral a intensidade da perturbação deve ser o suficiente para que a busca local não retorne ao ponto de partida. Perturbações muito fortes tendem a tornar o algoritmo aleatório. Normalmente uma boa

perturbação deve incorporar elementos específicos do problema e complementar a busca local (LOURENÇO; MARTIN; STÜTZLE, 2003).

É possível utilizar uma perturbação adaptativa, alterando a intensidade durante a busca, ou utilizando um histórico de buscas, como em (BATTITI; PROTASI, 1997), no qual o autor utiliza uma busca local para perturbação.

O critério de aceitação, encontrado na linha 6, determina se a nova solução será, ou não, aceita. Juntamente com a perturbação, definem a diversificação e a intensificação do espaço de buscas, pois de nada adianta a perturbação tentar diversificar novas soluções se o critério de aceitação não permiti-las.

De um lado extremo o critério de aceitação pode permitir apenas soluções que melhorem a solução atual, resultando em intensificação, enquanto no outro extremo permite qualquer solução gerada pela perturbação, resultando em diversificação. Podem ser criadas soluções intermediárias, como proposto a utilização de um critério baseado no Simulated Annealing em (MARTIN; OTTO; FELTEN, 1991), onde a nova solução é aceita com uma probabilidade que diminui com o andar do algoritmo. Ainda é possível levar em consideração o histórico de buscas realizando, por exemplo, o reinicio do algoritmo caso não tenha sido encontrada uma solução melhor nas últimas iterações. A perturbação, busca local e critério de aceitação se repetem em um laço que vai da linha 3 até a linha 7.

Neste trabalho fez-se uma hibridização entre as meta-heurísticas Variable Neighborhood Search e Iterated Local Search, utilizando elementos do ILS, como a perturbação, e a mudança de estrutura de vizinhança do VNS. A cada iteração o algoritmo proposto faz buscas locais utilizando diferentes estruturas de vizinhança, já que um ótimo local para uma estrutura de vizinhança pode não ser um ótimo local em relação a outra. Após serem aplicadas as buscas, a solução é degradada com uma perturbação, para poder explorar outras soluções no espaços de soluções. A força da perturbação varia de acordo com uma variável que conta o número de iterações executadas sem encontrar soluções melhores.

4 Métodos utilizados na resolução do PROP

Neste trabalho são apresentados algoritmos para resolver o PROP considerando que a distância de centro a centro entre um par de máquinas distinto pode ser calculada somente com as distâncias no eixo horizontal. Como as abordagens na literatura não consideram espaços entre as linhas, não considerou-se também neste trabalho. As posições nas mesmas iniciam-se de um ponto comum e não há espaços entre as máquinas.

Como se trata de um problema NP-hard, aqui serão implementados algoritmos que utilizam meta-heurística para encontrar boas soluções, que não diferem muito da solução ótima, em tempo computacional viável. As meta-heurísticas que foram utilizadas neste trabalho para comparação são: Simulated Annealing e um algoritmo híbrido VNS/ILS.

4.1 Solução Inicial

Na construção da solução inicial, tenta-se posicionar o mesmo número de facilidades em cada linha. No caso do PROP, foi adotado um número de facilidades $\lfloor n/2 \rfloor$ na primeira linha, e o restante atribuído para a segunda linha. A solução inicial é construída por um algoritmo guloso e aleatório que inicia recebendo uma lista de atribuições $L[\,]$ que aponta em qual linha estarão alocadas cada uma das facilidades e inicia a ordenação para cada linha, de acordo com a linha 1 do Algoritmo 4. Por exemplo, se $L[i] = 2$, a facilidade i deve ser posicionada na segunda linha.

Posteriormente, nas linhas 2 e 3, o algoritmo aleatoriamente seleciona uma facilidade e a posiciona na sua devida linha. Na linha 4 ele retira a facilidade alocada da lista de candidatas a entrar no layout LC. Então, a cada iteração, o algoritmo verifica para cada facilidade não alocada da lista LC a contribuição para o valor da função objetiva caso a facilidade fosse alocada no layout parcial em sua própria linha. Dentre elas, a facilidade que tiver a maior contribuição, ou seja, com maior fluxo entre as facilidades já alocadas, é a escolhida para entrar no layout e é alocada, de acordo com a linha 7. Isso é feito para poder posicionar facilidades com maior fluxo entre si mais próximas em um layout inicial. Nas linhas 9, 10 e 11 a facilidade escolhida é inserida no layout e retirada da lista LC. Nota-se que a ordem na qual cada facilidade é inserida na linha do layout define uma permutação de facilidades na linha. O laço externo se repete até que todas as facilidades tenham entrado no layout e vai da linha 5 até a 12 no algoritmo. Todos os passos utilizados na construção da solução inicial podem ser visualizados no Algoritmo 4. Como o PROP tem apenas duas linhas paralelas, k assume valor 2 no algoritmo.

Algoritmo 4 Construindo uma solução inicial - PROP

Entrada: lista de atribuição de cada linha $L[\ ]$
Saída: solução s
1: inicializa a ordenação na linha $r \in \{1, 2, ..., k\}$: $\pi^r \leftarrow \emptyset$ para cada linha
2: aleatóriamente seleciona $i \in N$
3: inserir i em sua devida linha no layout: $\pi^{L[i]} \leftarrow \pi^{L[i]} + \{i\}$
4: $LC \leftarrow N - \{i\}$
5: **enquanto** $LC \neq \emptyset$ **faça**
6: **para** cada facilidade $j \in LC$ **faça**
7: calcular o aumento na função objetivo caso a facilidade j fosse alocada ao layout parcial
8: **fim para**
9: seja $j*$ a facilidade com maior contribuição de valor para a função objetivo
10: inserir $j*$ na sua devida linha imediatamente após a facilidade anterior naquela linha: $\pi^{L[i]} \leftarrow \pi^{L[i]} + \{j*\}$
11: $LC \leftarrow LC - \{j*\}$
12: **fim enquanto**

4.2 Movimentos de Troca e Busca Local

Para o problema do PROP, foram utilizados alguns tipos de movimentos de troca: a *troca simples* e a *troca de coluna*.

A troca simples consiste em trocar duas facilidades aleatoriamente de uma mesma linha de posição. Após a troca, muitas vezes desloca-se muitas facilidades ou a linha inteira porque não deve haver sobreposições nem espaços entre facilidades.

A troca de coluna consiste em selecionar duas posições relativas aleatoriamente 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. Por exemplo, se o algoritmo selecionar as posições 2 e 4, as facilidades que estão na segunda e quarta posição de todas as linhas serão trocadas entre si.

A busca local utilizada no algoritmo híbrido segue os passos do Algoritmo 5. Ela recebe um parâmetro y identificando o movimento de troca a ser utilizado. Nos laços mais internos, para cada linha, i e j representam todos os pares de posições relativas a serem trocados na linha selecionada com o devido movimento de troca, com i e j indo de 1 até no máximo $N[linhaAtual]$, que é uma variável que conta número de facilidades na linha corrente. Os laços podem ser vistos nas linhas 6 e 7. Na linha 8, caso seja o 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. O loop mais externo, que vai da linha 3 até a 19, mantém a busca até que a solução não possa ser mais melhorada, de acordo com a variável *melhorou*.

Algoritmo 5 Busca Local

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

```
1:  sBest ← s
2:  melhorou ← FALSO
3:  enquanto melhorou = FALSO faça
4:    melhorou ← FALSO
5:    para linhaAtual ← 1 to k faça
6:      para i ← 1 to N[linhaAtual] − 1 faça
7:        para j ← i + 1 to N[linhaAtual] faça
8:          gera solução s' aplicando o movimento de troca y com os parâmetros i e j
9:          se f(f(s')) < f(s) então
10:           s ← s'
11:           melhorou ← VERDADEIRO
12:           se f(s) < f(sBest) então
13:             sBest ← s
14:           fim se
15:         fim se
16:       fim para
17:     fim para
18:   fim para
19: fim enquanto
```

4.3 Simulated Annealing

O algoritmo Simulated Annealing utilizado para resolver o PROP recebe os seguintes parâmetros: uma solução inicial s, a temperatura inicial T_0, o número máximo de iterações SA_{max}, a taxa de resfriamento α, um fator β e o número de perturbações P. A perturbação é feita escolhendo-se aleatoriamente um dos movimentos de troca anteriormente citados para ser aplicado na solução corrente e gerar nova solução s', de acordo com a linha 5 do algoritmo 6. Caso a solução seja menor, ela é imediatamente aceita na linha 8; caso contrário ela pode ser aceita na linha 13 de acordo com uma probabilidade. Se a nova solução for a melhor de todas até agora, ela é salva numa variável *sBest* na linha 10. Caso nenhuma solução seja 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. Isso é feito para ajudar o algoritmo a escapar de ótimos locais. O laço externo é executado um número vezes igual a SA_{max}.

Algoritmo 6 Simulated Annealing

Entrada: solução inicial s, T_0, α, β, SA_{max}, P
Saída: solução $sBest$

1: $sBest \leftarrow s$
2: $T \leftarrow T_0$
3: **para** $i \leftarrow 1$ **to** $SAmax$ **faça**
4: **para** $j \leftarrow 1$ **to** P **faça**
5: perturba solução gerando solução vizinha s'
6: $\Delta \leftarrow$ f(s') – f(s)
7: **se** $\Delta < 0$ **então**
8: $s \leftarrow s'$
9: **se** $f(s') < f(sBest)$ **então**
10: $sBest \leftarrow s'$
11: **fim se**
12: **senão**
13: faça $s \leftarrow s'$ de acordo com probabilidade $e^{\frac{-\Delta}{T}}$
14: **fim se**
15: **fim para**
16: **se** nenhuma solução melhor foi aceita **então**
17: $T \leftarrow \beta T$
18: **fim se**
19: $T \leftarrow \alpha T$
20: **fim para**

4.4 Algoritmo Híbrido

Neste trabalho foi feito um algoritmo híbrido que contém elementos das metaheurísticas ILS e VNS. O pseudocódigo é mostrado no Algoritmo 7.

A cada iteração do laço interno, que vai da linha 4 até a 21, o algoritmo perturba a solução de acordo com uma intensidade dada pela variável p, que inicializa com valor 1 e aumenta em caso de nenhuma solução melhor ser encontrada no loop interno até atingir um valor p_{max}. Se uma solução melhor for encontrada, a variável p é reinicializada e a solução é salva, de acordo com as linhas 16 e 17. Perturbar a solução com intensidade p significa aplicar p movimentos de troca na solução, os quais são escolhidos aleatoriamente dentre os movimentos de troca possíveis. Após a perturbação, o algoritmo faz buscas locais com o movimento de troca selecionado y. A busca local é realizada com o Algoritmo 5. Como no caso do PROP há apenas dois movimentos de troca então y tem valor máximo de 2. Se uma busca local é bem sucedida na linha 10, 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 utilizado na linha 7, que aplica movimentos de troca aleatórios na solução, incluindo um movimento de movimentar uma facilidade para o final da sua devida linha.

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

4.5 Resultados Computacionais

Os testes computacionais dos algoritmos na resolução do PROP foram feitos utilizando as instâncias usadas por (HUNGERLÄNDER; ANJOS, 2015) e (AMARAL, 2013b). Dentre elas, há um conjunto de 21 instâncias consideradas pequenas com tamanhos de 11 até 23 facilidades com Layouts ótimos calculados pelo modelo MIP em (AMARAL, 2013b). Os algoritmos são então validados de acordo com esses valores encontrados na literatura. Há um outro conjunto de instâncias consideradas grandes com tamanhos que vão de 60 até 110 facilidades. Juntos, foram um total de 44 instâncias testadas neste trabalho para o PROP. Como não há resultados para instâncias grandes na literatura, os resultados dos algoritmos aqui testados são comparados para tais instâncias. O número de facilidades em cada linha é balanceado, sendo igual a $\lfloor \frac{n}{k} \rfloor$.

Os parâmetros utilizados no algoritmo do 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. Os parâmetros foram calibrados de forma empírica com testes nos quais uma variável era calibrada enquanto as outras eram mantidas fixas.

Não há comparações com os resutados de (HUNGERLÄNDER; ANJOS, 2015) porque a abordagem é diferente já que permite espaços entre facilidades, além de haver um balanceamento de facilidades antes da ordenação. Neste trabalho, a escolha das facilidades em cada linha é feito de acordo com a ordem das mesmas na instância.

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.

4.5.1 PROP com instâncias de 11 a 23 facilidades

Os resultados de ambos algoritmos são validados utilizando os valores ótimos obtidos na literatura e podem ser vistos na tabela 1. Esse conjunto de instâncias consiste em um conjunto de 21 instâncias com tamanhos iguais a 15, 16, 20, 21, 22 e 23 facilidades.

A primeira coluna da Tabela 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. Os valores em negrito indicam as melhores soluções encontradas para cada instância.

Tabela 1 – Instâncias de 11 a 23 facilidades - PROP

	MIP Amaral	SA			VNS/ILS		
Instância	Valor Ótimo	Mínimo	Média	DP	Mínimo	Média	DP
S_11	**3895,5**	**3895,5**	3895,5	0,00	**3895,5**	3895,5	0,00
Am15	**3435,0**	**3435**	3435	0,00	**3435**	3435	0,00
P16_a	**7630,0**	**7630**	7630	0,00	**7630**	7630	0,00
P16_b	**6239,5**	**6239,5**	6239,5	0,00	**6239,5**	6239,5	0,00
P20_a	**12609,5**	**12609,5**	12612,7	6,40	**12609,5**	12620,8	10,45
P20_b	**12936,0**	**12936**	12953,8	35,60	**12936**	12949,3	39,90
P21_a	**7006,5**	**7006,5**	7006,5	0,00	**7006,5**	7009	7,50
P21_b	**11705,0**	**11705**	11705,6	1,20	**11705**	11705,3	0,90
P21_c	**11434,0**	**11434**	11434	0,00	**11434**	11435,6	4,80
P21_d	**12289,0**	**12289**	12291,7	8,10	**12289**	12289	0,00
P21_e	**13112,5**	**13112,5**	13112,5	0,00	**13112,5**	13113,1	0,92
P22_a	**8874,0**	**8874**	8877,8	11,40	**8874**	8878,7	11,18
P22_b	**15714,0**	**15714**	15714	0,00	**15714**	15714	0,00
P22_c	**14693,0**	**14693**	14697	12,00	**14693**	14710,2	21,32
P22_d	**16355,0**	**16355**	16358,9	11,70	**16355**	16362,7	5,04
P22_e	**14815,5**	**14815,5**	14829,5	28,00	**14815,5**	14830,4	29,87
P23_a	**10242,0**	**10242**	10244,4	7,20	**10242**	10247,3	6,50
P23_b	**15802,5**	**15802,5**	15802,5	0,00	**15802,5**	15802,5	0,00
P23_c	**15542,0**	**15542**	15542	0,00	**15542**	15542	0,00
P23_d	**17174,0**	**17174**	17174	0,00	**17174**	17184	10,00
P23_e	**16481,5**	**16481,5**	16498,3	47,48	**16481,5**	16481,5	0,00

Os tempos computacionais médios do algoritmo exato e dos algoritmos heurísticos podem ser observados na Tabela 2.

Tabela 2 – Tempos computacionais (em segundos) para instâncias de 11 a 23 facilidades - PROP

Instância	MIP Amaral	SA	VNS/ILS
S_11	0,23	0,368	0,001
Am15	1,79	0,523	0,009
P16_a	26,18	0,584	0,011
P16_b	46,69	0,542	0,011
P20_a	4461,36	0,816	0,063
P20_b	1080,51	0,894	0,062
P21_a	1316,27	0,866	0,051
P21_b	2095,24	0,894	0,045
P21_c	568,32	0,821	0,051
P21_d	3701,92	0,861	0,046
P21_e	5754,71	0,908	0,044
P22_a	5114,44	1,151	0,073
P22_b	7506,63	0,915	0,064
P22_c	63825,77	1,053	0,066
P22_d	41713,32	0,918	0,060
P22_e	55591,31	0,975	0,064
P23_a	52515,75	1,026	0,091
P23_b	28371,18	1,027	0,087
P23_c	79860,19	1,105	0,073
P23_d	61151,19	1,097	0,088
P23_e	96033,18	1,087	0,076

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. O Simulated Annealing mostrou um desvio padrão máximo na instância P23_e com valor 47.48 e o Algoritmo Híbrido mostrou um desvio padrão máximo na instância P_20b de 39.90. De forma geral, os algoritmos obtiveram desvios padrão relativamente (relativos aos valores mínimos) 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 exibiu menor tempo computacional para todas as instâncias.

4.5.2 Instâncias grandes para o PROP

Como citado anteriormente, para as grandes instâncias os resultados de ambos algoritmos são mostrados lado-a-lado a fim de comparação.

Na Tabela 3 podem ser observados os resultados de ambas meta-heurísticas para um conjunto de 20 instâncias encontradas em (ANJOS; KENNINGS; VANNELLI, 2005).

Esse conjunto é chamado de AKV e possui 5 instâncias de cada tamanho sendo eles 60, 70, 75 e 80. Na tabela podem ser verificados em cada coluna os nomes das instâncias, os campos com melhores soluções encontradas pelos algoritmos em todas as execuções (Mínimo), Solução Média dos algoritmos (Média) e desvios padrão em relação a todas as execuções (DP). Os valores em negrito indicam as melhores soluções encontradas para a instância. Na última linha são exibidas as médias dos valores das colunas Média e Desvio Padrão.

Tabela 3 – Instâncias grandes (AKV) - PROP

	SA			VNS/ILS		
Instância	Mínimo	Média	DP	Mínimo	Média	DP
AKV-60-01	**772202,0**	772369,1	204,4	**772202**	772389	196,9
AKV-60-02	**430380,0**	430450,4	123,5	430384	430433,4	37,4
AKV-60-03	**331103,5**	331538,2	447,0	331140,5	331562,2	523,0
AKV-60-04	**201052,0**	201179	155,1	201128	201361,5	143,5
AKV-60-05	**165096,0**	165111,9	31,1	**165096**	165101,8	3,7
AKV-70-01	**779130,0**	779905,3	689,3	779151	779766,2	286,5
AKV-70-02	**737919,0**	738997,1	548,5	738749	739437,4	324,2
AKV-70-03	**764463,5**	764942,6	473,5	**764463,5**	765248,6	262,2
AKV-70-04	**491028,0**	491110,5	97,9	491041	491098,1	47,8
AKV-70-05	**2187780,5**	2188087,5	897,3	**2187780,5**	2187882,1	124,2
AKV-75-01	**1214584,5**	1215166,6	860,7	1214737,5	1214865,5	186,0
AKV-75-02	**2173260,0**	2174634,1	1647,0	2173443	2173612,3	115,5
AKV-75-03	**634119,0**	634544,9	209,8	634155	634576,6	182,4
AKV-75-04	2009064,5	2010043,5	1068,3	**2008998,5**	2009269,6	121,0
AKV-75-05	**914833,0**	915573,7	978,3	**914833**	914981,2	249,8
AKV-80-01	**1043521,5**	1044245,9	562,5	1043768,5	1043925,7	147,1
AKV-80-02	**986822,0**	987166,1	278,4	986904	987333,9	234,4
AKV-80-03	**1638732,0**	1640013,8	1732,5	1639900	1641148,6	1483,9
AKV-80-04	1902717,0	1904540,4	1237,0	**1902594**	1903595,6	902,8
AKV-80-05	**800673,0**	800796,7	118,3	**800673**	801080,9	357,2
	1008924,1		618,0	1009057,1		296,5

Na Tabela 4 podem ser verificados em cada coluna os nomes das instâncias e tempos computacionais utilizados por cada algoritmo, em segundos.

Tabela 4 – Tempos computacionais em segundos para instâncias grandes (AKV) - PROP

Instância	SA	VNS/ILS
AKV-60-01	34,2	43,9
AKV-60-02	27,3	53,1
AKV-60-03	30,4	53,3
AKV-60-04	31,1	62,1
AKV-60-05	28,8	57,3
AKV-70-01	38,6	137,5
AKV-70-02	40,3	185,1
AKV-70-03	38,9	177,6
AKV-70-04	39,7	177,3
AKV-70-05	50,8	127,0
AKV-75-01	45,6	225,0
AKV-75-02	47,3	191,5
AKV-75-03	46,0	228,6
AKV-75-04	48,3	263,1
AKV-75-05	46,4	282,8
AKV-80-01	57,1	377,5
AKV-80-02	57,4	398,3
AKV-80-03	47,2	432,4
AKV-80-04	58,1	448,3
AKV-80-05	57,9	480,6

Pode-se ver que o algoritmo Simulated Annealing obteve soluções mínimas iguais ou melhores que o Algoritmo Híbrido na maioria das 20 instâncias do conjunto, apesar de que o Algoritmo Híbrido tenha chegado a soluções mínimas próximas. As médias de todas as soluções mínimas dos dois algoritmos, exibidas na última linha, ficaram próximas (diferença de 0,01%). O Algoritmo Híbrido somente conseguiu melhor solução mínima nas instâncias AKV-75-04 e AKV-80-04. Os desvio padrão encontrados foram relativamente baixos (relativos às soluções mínimas), sendo que a média apresentada pelo Algoritmo Híbrido foi menor. O menor desvio padrão do Simulated Annealing ocorreu para a instância AKV-60-05 com valor 31.1 enquanto que o menor desvio padrão obtido pelo Algoritmo Híbrido foi 3.7 para a mesma instância. Já o maior desvio padrão obtido para o SA foi para a instância AKV-80-03, com valor 1732.5, enquanto que o Algoritmo Híbrido oteve o maior desvio padrão na instância AKV-80-03, com valor 1483.9.

Na Tabela 4 pode-se observar os tempos computacionais médios exibidos por ambos algoritmos, em segundos. Enquanto que a proporção de tempo de execução dos dois algoritmos nas instâncias de tamanho 60 é aproximadamente o dobro, pode-se observar uma relação de tempo de até 8 para 1 nas instâncias de tamanho 80. De forma geral o SA obteve menores tempos computacionais médios do que o VNS/ILS.

Na Tabela 5 pode-se observar os resultados de ambos algoritmos para um conjunto de 20 instâncias encontradas em (ANJOS; YEN, 2009). Esse conjunto é chamado de sko e

suas instâncias testadas neste trabalho foram as de tamanho 64, 72, 81 e 100, sendo que foram testadas 5 de cada tamanho. Na Tabela também podem ser verificados em cada coluna os campos com melhores soluções (Mínimo), desvios padrão (DP) e Solução Média (Média). Novamente na última linha é feita uma média de todos os valores obtidos nas colunas citadas e os valores em negrito indicam as melhores soluções encontradas para a instância.

Tabela 5 – Instâncias grandes (sko) - PROP

	SA			VNS/ILS		
Instância	Mínimo	Média	DP	Mínimo	Média	DP
sko64-1	**48557**	48571,2	40,3	**48557**	48606,7	31,5
sko64-2	**321145,5**	321170,9	47,8	**321145,5**	321222,6	79,8
sko64-3	**207998,5**	208000,6	4,2	**207998,5**	208048,7	46,7
sko64-4	**149135**	149390,1	215,9	149143	149290,4	166,9
sko64-5	**252067,5**	252071,2	8,1	252069,5	252093,8	13,9
sko72-1	**70072**	70077,3	8,5	70086	70104,1	15,7
sko72-2	**358510**	358684	260,8	358557	358667	79,4
sko72-3	**530000,5**	530355	704,0	**530000,5**	530692,5	831,8
sko72-4	**474818,5**	475183,9	538,4	**474818,5**	474932,5	68,2
sko72-5	**216275,5**	216537,9	283,4	216300,5	216420,9	65,3
sko81-1	**102987**	103039,9	29,9	103095	103154,4	41,5
sko81-2	**262590,5**	263153,9	471,8	262594,5	262678,5	196,7
sko81-3	**491801**	492471	667,6	491818	492153,7	289,9
sko81-4	**1018548**	1019096	385,4	1018651	1019487,4	377,5
sko81-5	**656020**	656463,9	417,6	**656020**	656302,9	348,1
sko100-1	**189988**	190009,7	17,3	190001	190130,3	77,6
sko100-2	**1043726,5**	1044600,9	456,7	1044735,5	1045322,1	442,3
sko100-3	**8092502,5**	8093612	1347,5	8092686,5	8094334,3	1852,0
sko100-4	**1681236**	1682862,5	2330,2	1682004	1682816,3	1029,1
sko100-5	518379,5	518835,2	345,5	**518366,5**	518650,3	141,1
	834318,0		429,0	834432,4		309,7

Na Tabela 6 podem ser verificados os tempos computacionais utilizados por cada algoritmo, em segundos.

Tabela 6 – Tempos computacionais em segundos para instâncias grandes (sko) - PROP

Instância	SA	VNS/ILS
sko64-1	32,2	126,0
sko64-2	41,7	101,0
sko64-3	34,8	99,7
sko64-4	37,1	122,3
sko64-5	37,3	111,3
sko72-1	46,6	225,7
sko72-2	42,8	237,3
sko72-3	40,9	228,5
sko72-4	40,0	246,9
sko72-5	43,2	229,9
sko81-1	48,0	568,6
sko81-2	60,7	481,5
sko81-3	53,9	580,8
sko81-4	53,3	567,2
sko81-5	53,7	531,5
sko100-1	88,1	2526,4
sko100-2	78,5	2803,7
sko100-3	88,4	2059,4
sko100-4	81,8	2266,8
sko100-5	85,8	2058,1

Pode-se observar novamente que o Simulated Annealing obteve melhores soluções mínimimas para quase todas as 20 instâncias, obtendo uma solução mínima maior somente na instância sko100-5. Em algumas instâncias como sko72-1 e sko100-2 até mesmo a solução média obtida pelo SA superou a mínima encontrada pelo Algoritmo Híbrido. A média das soluções mínimas do Simulated Annealing foi menor que a do Híbrido. Já a média da coluna referente ao desvio padrão mostra um valor menor para o Algoritmo Híbrido. O SA obteve valores bem pequenos (em relação às soluções mínimas) de desvio padrão nas instâncias sko64-5 e sko72-1: 8.1 e 8.5 respectivamente, porém obteve um desvio padrão máximo de 2330.2 na instância sko100-4, o que significa 0,13% da solução mínima. Já o Algoritmo Híbrido mostrou desvio padrão mínimo de 13.9 na instância sko64-5 e máximo de 1852.0 na instância sko100-3.

Na tabela 6 pode-se ver os tempos computacionais médios exibidos pelos algoritmos. Enquanto que os tempos médios do Simulated Annealing aumentam devagar com o aumento do tamanho da instância, os exibidos pelo Algoritmo Híbrido cresceram muito rapidamente chegando a utilizar um tempo de execução até 24 vezes maior que o Simulated Annealing na última instância.

Finalmente foram rodados os algoritmos para um conjunto de 3 instâncias grandes com tamanho igual a 110 facilidades utilizados em (AMARAL; LETCHFORD, 2013). Na Tabela 7 também podem ser verificados em cada coluna os campos com nomes das

instâncias, melhores soluções encontradas (Mínimo), desvios padrão (DP) e Solução Média (Média). Novamente os valores em negrito indicam as melhores soluções encontradas para a instância e a última linha indica uma média dos valores de Média e Desvio Padrão.

Tabela 7 – Instâncias grandes (Am) - PROP

	SA			VNS/ILS		
Instância	Mínimo	Média	DP	Mínimo	Média	DP
Am_110_01	**72323216,5**	72357804,7	31371,4	72327687,5	72340003,9	9891,8
Am_110_02	**1120526,5**	1121080	395,0	1120532,5	1120689,4	335,0
Am_110_03	**101926381,0**	101956358,1	22616,1	101930623,0	101939722,6	13024,5
	58456708,0		429,0	58459614,3		309,7

Na Tabela 8 podem ser verificados em cada coluna os nomes das instâncias e tempos computacionais utilizados por cada algoritmo, em segundos.

Tabela 8 – Tempos computacionais em segundos para instâncias grandes (Am) - PROP

Instância	SA	VNS/ILS
Amaral_110_01	128,32	4099,8
Amaral_110_02	107,56	4447,3
Amaral_110_03	97,81	3203,4

O Simulated Annealing novamente continuou a obter melhores exibições em termos de solução mínima e solução média. Os algoritmos SA e Híbrido obtiveram respectivamente um desvio padrão de 395.0 e 335.0 na instância Am_110_02, valores que podem ser considerados baixos para tal instância (0,03% e 0,02% respectivamente em relação á solução mínima).

Na tabela 8 pode-se ver os tempos computacionais médios exibidos pelos algoritmos. Os tempos do Algoritmo Híbrido aumentaram significativamente para tal conjunto de instâncias exibindo resultados relativamente altos. O tempo computacional médio do Algoritmo Híbrido foi aproximadamente 33 vezes maior que o exibido pelo Simulated Annealing na instância Am_110_03.

Pode-se observar em todos os testes com grandes instâncias que os tempos computacionais do Algorítimo Híbrido são maiores que os obtidos pelo Simulated Annealing. Com o aumento do tamanho das instâncias o espaço de soluções aumenta muito rapidamente fazendo com que as buscas locais do VNS tomem mais tempo.

De forma geral o algoritmo Simulated Annealing mostrou ser mais eficaz obtendo melhores soluções em menor tempo computacional nos conjuntos testados. O algoritmo parece ser competitivo para instâncias grandes do PROP.

5 Métodos utilizados na resolução do CAP e SF-MRFLP

Neste capítulo são apresentados algoritmos para resolver o CAP e SF-MRFLP. Da mesma forma que o PROP foi abordado, somente o eixo-horizontal foi levado em conta na distância de centro a centro entre um par de facilidades distinto. As linhas necessitam iniciar de um ponto comum e não há espaços entre as máquinas. Como um problema de facilidade NP-hard, as meta-heurísticas Simulated Annealing, VNS e ILS foram utilizadas.

5.1 Solução Inicial

A construção da solução inicial é feita de forma gulosa e aleatória. Os passos utilizados na construção da solução inicial podem ser visualizados no Algoritmo 8.

O algoritmo inicia alocando uma facilidade aleatória no Layout e retirando-a da lista de candidatos LC, como pode ser visto nas linhas 1, 2 e 3. Posteriormente, a cada iteração do laço externo, o algoritmo verifica em um laço interno, para cada facilidade não alocada, a contribuição para o valor da função objetivo caso a facilidade fosse alocada no layout parcial na linha de menor comprimento. Essa verificação é feita na linha 7. Nas linhas 9, 10 e 11, a facilidade $j*$ que tiver a maior contribuição, ou seja, com maior fluxo entre as facilidades já alocadas, é a escolhida para entrar no layout e é introduzida imediatamente após a facilidade anterior, respeitando as restrições de sobreposição e espaço entre facilidades; assim como é retirada da lista de candidatas LC.

Algoritmo 8 Construindo uma solução inicial - SF-MRFLP

1: aleatóriamente seleciona $i \in N$
2: inserir i na primeira linha do layout
3: $LC \leftarrow N - \{i\}$
4: **enquanto** $LC \neq \emptyset$ **faça**
5: selecionar a linha r com menor comprimento total
6: **para** para cada facilidade $j \in LC$ **faça**
7: calcular o aumento na função objetivo caso a facilidade j fosse alocada ao layout parcial na linha r
8: **fim para**
9: seja $j*$ a facilidade com maior contribuição de valor para a função objetivo
10: inserir $j*$ na linha r imediatamente após a facilidade anterior naquela linha
11: $LC \leftarrow LC - \{j*\}$
12: **fim enquanto**

5.2 Movimentos de Troca e Busca Local

Para o problema do SF-MRFLP, foram utilizados alguns tipos de movimentos de troca: a troca simples e a troca de linha.

A *troca simples* consiste em trocar duas facilidades de posição selecionadas aleatoriamente. Após a troca, muitas vezes desloca-se muitas facilidades porque não deve haver sobreposições nem espaços entre facilidades.

A *troca de linha* consiste em mover uma facilidade de uma linha para outra no layout, mantendo sua posição relativa na mesma. Por exemplo, se uma facilidade está posicionada na segunda posição da terceira linha e é envolvida em uma troca de linha para a primeira linha, ela continuará na segunda posição da primeira linha, com as devidas correções no layout. Esse movimento também respeita as restrições de que não pode haver sobreposições e espaços entre facilidades no SR-MRFLP.

A busca local utilizada no Algoritmo Híbrido para o SF-MRFLP segue os mesmos passos do Algoritmo 5. A diferença é que os movimentos de troca utilizados são a troca simples e a troca de linha.

5.3 Simulated Annealing

O algoritmo Simulated Annealing utilizado para resolver o SF-MRFLP recebe os mesmos parâmetros do PROP, visualizado no Algoritmo 6. A perturbação é feita escolhendo-se aleatoriamente a troca simples ou a troca de coluna para ser aplicado na solução corrente e gerar nova solução s'.

Diferentemente do que foi utilizado no PROP, a temperatura inicial não é um valor fixo dado como parâmetro ao algoritmo. Foi implementada uma função chamada SA_reverso que calcula a temperatura inicial de acordo com o tamanho da entrada. O método foi implementado baseado no pseudocódigo encontrado em (AHONEN; DE ALVARENGA; AMARAL, 2014), já que obteve resultados excelentes para o CAP. O pseudocódigo pode ser observado em 9. O algoritmo segue basicamente os mesmos passos do Simulated Annealing clássico. A diferença é que, na linha 17, a temperatura vai aumentando a cada iteração. Nesta função, a temperatura inicial é inicializada com valor baixo e vai aumentando de acordo com o número de melhorias na solução a cada iteração de temperatura até que um valor estipulado $max\_aceitas$ de novas soluções $sol\_aceitas$ seja aceito pelo algoritmo. Além de produzir uma temperatura inicial para o algoritmo Simulated Annealing, essa função também já produz uma solução apurada $sBEst$, por armazenar as melhore soluções encontradas, de acordo com a linha 11.

Algoritmo 9 SA_reverso

Entrada: solução inicial s, $max\_aceitas$, α
Saída: solução $sBest$, T_0

```
1: sol_aceitas ← 0
2: T_0 ← 0.01
3: enquanto sol_aceitas < max_aceitas faça
4:    para j ← 1 to P  faça
5:       perturba solução gerando solução vizinha s'
6:       Δ ← f(s') – f(s)
7:       se Δ < 0 então
8:          sol_aceitas ← sol_aceitas + 1
9:          s ← s'
10:         se f(s') < f(sBest) então
11:            sBest ← s'
12:         fim se
13:      senão
14:         faça s ← s' de acordo com probabilidade e^(−Δ/T)
15:      fim se
16:   fim para
17:   T_0 ← T_0/α
18: fim enquanto
```

O Simulated Annealing implementado difere-se do SA de Ahonen, De Alvarenga e Amaral (2014) por implementar um aumento de temperatura a cada iteração do laço externo de acordo com um fator β com o objetivo de escapar de mínimos locais.

5.4 Algoritmo Híbrido

Na aplicação do Algoritmo Híbrido para o SF-MRFLP, foram aplicados os mesmos passos aplicados para o PROP. Como no SF-MRFLP também são utilizados dois movimentos de troca, y tem valor máximo de 2 no Algoritmo 7. Se uma busca local é bem sucedida, y aponta novamente para o primeiro movimento de troca e as buscas recomeçam. Além da *troca simples* e *troca de linha*, o movimento *shake* utilizado no SF-MRFLP e CAP pode deslocar uma facilidade de uma linha para o final de outra linha aleatória.

5.5 Resultados Computacionais

Os experimentos computacionais para o CAP e SF-MRFLP foram realizados baseados em instâncias usadas em (HUNGERLÄNDER; ANJOS, 2015) e (AHONEN; DE ALVARENGA; AMARAL, 2014). Compondo o conjunto de 40 instâncias testadas há um grupo de 20 instâncias consideradas pequenas, com tamanhos que vão de 5 a 15 facilidades, e um grupo de 20 instâncias consideradas grandes para o problema, que possuem de 30 a

56 facilidades.

Os parâmetros utilizados no algoritmo do Simulated Annealing SA_{max} e α foram iguais a respectivamente 650 e 0.975 para $n \leq 30$; 2500 e 0.99 para $31 \leq n \leq 45$ e 4000 e 0.995 para $n \geq 46$. O número de perturbações P é igual a 400 e o fator β é igual a $\frac{1}{\alpha}$. No *SA_reverso* os parâmetros SA_{max} e α utilizados foram os mesmos do Simulated Annealing. Como entrada no algoritmo, o parâmetro *max_aceitas* foi igual a 8000 para $n \leq 30$; 13000 para $31 \leq n \leq 45$ e 20000 para $n \geq 46$. Já os parâmetros do VNS/ILS i_{max} e p_{max} são iguais ao número de facilidades n.

Os testes computacionais para o CAP e SF-MRFLP foram feitos nas mesmas condições dos testes feitos para o PROP: algoritmos compilados com gcc 5.4 utilizando a linguagem C e rodados em um computador com processador Intel core i7 2.0 GHz com 8 GB de RAM utilizando sistema operacional Linux Ubuntu 16.04. Para cada instância, os algoritmos foram executados 10 vezes.

5.5.1 CAP para instâncias de 5 a 15 facilidades

Primeiramente resolveu-se o CAP e comparou-se os resultados dos algoritmos implementados com os valores obtidos em (HUNGERLÄNDER; ANJOS, 2015), que utilizaram programação semi-definida (SDP), e (AHONEN; DE ALVARENGA; AMARAL, 2014), que utilizaram uma eficiente implementação do Simulated Annealing. Os resultados podem ser observados na Tabela 9.

A primeira coluna da tabela mostra os nomes das instâncias. A segunda coluna mostra os resultados mínimos obtidos pela heurística SDP nas instâncias que vão de H_5 até S_11. A terceira coluna mostra os resultados mínimos obtidos pelo Simulated Annealing de Ahonen, De Alvarenga e Amaral (2014) nas instâncias S_9, SH_9, S_10, S_11, Am12a, Am12b, Am13a, Am13b e Am15. Nas três colunas seguintes pode-se observar os resultados mínimo (Mínimo), médio (Média) e desvio padrão (DP) do algoritmo Simulated Annealing implementado para o problema. Finalmente, nas três últimas colunas pode-se observar os resultados mínimo, médio e desvio padrão do algoritmo híbrido. Os valores em negrito indicam as melhores soluções encontradas para cada instância.

Tabela 9 – Instâncias de 5 a 15 facilidades - CAP

	SDP	SA (Ahonen et al., 2014)	SA (2017)			VNS/ILS		
Instância	Mínimo	Mínimo	Mínimo	Média	DP	Mínimo	Média	DP
H_5	**450,0**	-	**450,0**	450,0	0,0	**450,0**	450,0	0,0
Rand_5	**52,5**	-	**52,5**	52,5	0,0	**52,5**	52,5	0,0
H_6	**720,0**	-	**720,0**	720,0	0,0	**720,0**	720,0	0,0
Rand_6	**190,5**	-	**190,5**	190,5	0,0	**190,5**	190,5	0,0
H_7	**1700,0**	-	**1700,0**	1700,0	0,0	**1700,0**	1700,0	0,0
Rand_7	**166,0**	-	**166,0**	166,0	0,0	**166,0**	166,0	0,0
H_8	**2385,0**	-	**2385,0**	2385,0	0,0	**2385,0**	2385,0	0,0
S_8	**408,0**	-	**408,0**	408,0	0,0	**408,0**	408,0	0,0
SH_8	**1135,5**	-	**1135,5**	1135,5	0,0	**1135,5**	1135,5	0,0
Rand_8	**205,0**	-	**205,0**	205,0	0,0	**205,0**	205,0	0,0
S_9	**1181,5**	**1181,5**	**1181,5**	1181,5	0,0	**1181,5**	1181,5	0,0
SH_9	**2294,5**	**2294,5**	**2294,5**	2294,5	0,0	**2294,5**	2294,5	0,0
Rand_9	**492,5**	-	**492,5**	492,5	0,0	**492,5**	492,5	0,0
S_10	**1374,5**	**1374,5**	**1374,5**	1374,5	0,0	**1374,5**	1374,5	0,0
Rand_10	**838,0**	-	**838,0**	838,0	0,0	**838,0**	838,0	0,0
S_11	**3439,5**	**3439,5**	**3439,5**	3439,5	0,0	**3439,5**	3439,5	0,0
Am12a	-	**1529,0**	**1529,0**	1530,2	3,0	**1529,0**	1529,0	0,0
Am12b	-	**1609,5**	**1609,5**	1609,5	0,0	**1609,5**	1609,5	0,0
Am13a	-	**2467,5**	**2467,5**	2468,6	1,4	**2467,5**	2467,5	0,0
Am13b	-	**2870,0**	**2870,0**	2870,1	0,3	**2870,0**	2870,0	0,0
Am15	-	**3195,0**	**3195,0**	3195,0	0,0	**3195,0**	3195,0	0,0

Os tempos computacionais utilizados pela heurística SDP e pelos algoritmos implementados, em segundos, podem ser observados na tabela 10.

Tabela 10 – Tempos computacionais em segundos para instâncias de 5 a 15 facilidades - CAP

Instância	Heurística SDP	SA (Ahonen et al., 2014)	SA (2017)	VNS/ILS
H_5	4,80	-	0,210	0,005
Rand_5	4,50	-	0,185	0,006
H_6	11,10	-	0,220	0,009
Rand_6	13,80	-	0,166	0,008
H_7	25,20	-	0,223	0,015
Rand_7	31,70	-	0,163	0,014
H_8	91,30	-	0,179	0,025
S_8	91,60	-	0,174	0,025
SH_8	87,40	-	0,169	0,027
Rand_8	82,20	-	0,168	0,024
S_9	253,40	1,14	0,215	0,048
SH_9	251,90	1,15	0,193	0,046
Rand_9	252,60	-	0,188	0,041
S_10	713,00	1,58	0,247	0,075
Rand_10	698,20	-	0,251	0,071
S_11	2127,00	2,02	0,276	0,116
Am12a	-	2,43	0,345	0,164
Am12b	-	2,43	0,390	0,161
Am13a	-	2,99	0,445	0,250
Am13b	-	3,01	0,346	0,251
Am15	-	4,70	0,518	0,504

Todos dois algoritmos conseguiram atingir os valores mínimos conhecidos da literatura em pelo menos uma execução para todas as instâncias, com valores médios próximos ou iguais ao mínimo encontrado. O Simulated Annealing mostrou um desvio padrão máximo na instância Am12a com valor 3.0 e o algoritmo híbrido obteve desvios padrão iguais a zero para todas as instâncias testadas.

Não se pode comparar diretamente os tempos computacionais dos algoritmos implementados com os algoritmos encontrados na literatura por serem executados em diferentes tipos de hardware mas a Tabela 2 exibe os tempos computacionais de todos algoritmos para dar uma ideia de seus valores. Comparando-se os dois algoritmos implementados, pode-se ver que o Algoritmo Híbrido exibiu menor tempo computacional em todas as instâncias.

5.5.2 CAP para instâncias grandes

Posteriormente o CAP foi resolvido para instâncias grandes encontradas na literatura e os resultados dos algoritmos implementados foram comparados com os valores obtidos pelo algoritmo Simulated Annealing implementado em (AHONEN; DE ALVARENGA;

AMARAL, 2014). Os resultados podem ser observados na Tabela 11. Um conjunto de 20 instâncias foi utilizado nessa bateria de testes.

A primeira coluna da tabela mostra os nomes das instâncias. A segunda coluna mostra os resultados mínimos obtidos pela heurística Simulated Annealing de Ahonen, De Alvarenga e Amaral (2014). As três colunas seguintes mostram os resultados mínimo (Mínimo), médio (Média) e desvio padrão (DP) do algoritmo Simulated Annealing implementado neste trabalho. Nas três últimas colunas pode-se observar os resultados mínimo, médio e desvio padrão do algoritmo híbrido. Os valores em negrito indicam as melhores soluções encontradas para cada instância.

Tabela 11 – Instâncias grandes - CAP

	SA (Ahonen, 2014)	SA			VNS/ILS		
Instância	Mínimo	Mínimo	Média	DP	Mínimo	Média	DP
N30_01	**4115,0**	**4115,0**	4115,0	0,0	**4115,0**	4115,8	1,2
N30_02	**10779,5**	**10779,5**	10784,1	3,7	10786,5	10800,8	10,5
N30_03	**22702,0**	**22702,0**	22705,2	5,4	22704,0	22728,6	12,9
N30_04	**28401,5**	**28401,5**	28413,0	10,0	28444,5	28470,6	21,9
N30_05	**57400,0**	**57400,0**	57451,5	124,1	57483,0	57565,9	57,8
sko42-1	**12731,0**	**12731,0**	12733,8	3,4	12736,0	12747,7	7,7
sko42-2	**108016,5**	108044,5	108096,6	64,9	108122,5	108184,8	50,5
sko42-3	**86646,5**	86649,5	86760,2	112,4	86730,5	86800,5	44,4
sko42-4	**68708,0**	68725,0	68787,4	38,5	68829,0	68910,2	50,3
sko42-5	**124017,5**	124032,5	124059,4	35,4	124162,5	124256,6	66,3
sko49-1	**20470,0**	**20470,0**	20480,3	10,2	20482,0	20500,3	9,9
sko49-2	**208079,0**	208104,0	208444,1	365,4	208382,0	208516,8	104,4
sko49-3	162196,0	**162189,0**	162337,5	225,9	162402,0	162537,1	97,4
sko49-4	**118260,5**	118275,5	118337,2	86,6	118363,5	118542,5	85,8
sko49-5	332853,0	**332836,0**	332920,2	44,3	333232,0	333470,7	141,9
sko56-1	**31972,0**	**31972,0**	31976,4	5,4	31983,0	32017,5	17,4
sko56-2	**248225,0**	248226,0	248254,3	26,2	248500,0	248703,0	83,0
sko56-3	**85190,0**	85202,0	85240,4	53,5	85285,0	85405,8	48,3
sko56-4	**156666,0**	156691,0	156745,6	33,6	156867,0	157017,7	90,8
sko56-5	**296176,5**	296214,5	296275,8	42,2	296504,5	296661,2	100,0

Os tempos computacionais utilizados pelos algoritmos heurísticos, em segundos, podem ser observados na tabela 12.

Tabela 12 – Tempos computacionais em segundos para instâncias grandes - CAP

Instância	SA (Ahonen, 2014)	SA (2017)	VNS/ILS
N30_01	25,59	4,89	7,10
N30_02	26,13	5,86	7,84
N30_03	26,12	6,04	7,80
N30_04	25,39	6,03	8,71
N30_05	25,29	5,68	8,40
sko42-1	25,48	9,14	48,28
sko42-2	25,13	9,86	54,60
sko42-3	21,67	11,80	55,56
sko42-4	21,80	10,92	59,45
sko42-5	22,12	10,03	59,11
sko49-1	47,74	23,35	27,62
sko49-2	47,41	19,07	33,47
sko49-3	47,80	19,90	32,92
sko49-4	47,63	20,88	31,64
sko49-5	47,22	22,04	31,64
sko56-1	92,61	24,82	64,80
sko56-2	92,59	29,77	75,72
sko56-3	93,35	28,69	65,03
sko56-4	128,69	30,38	62,25
sko56-5	142,37	28,24	68,25

Pode-se ver que o algoritmo Simulated Annealing obteve soluções mínimas melhores que o Algoritmo Híbrido na maioria das 20 instâncias do conjunto. O Simulated Annealing conseguiu obter uma solução com desvio padrão igual a 0 na instância N30_01. Além disso, conseguiu alcançar a melhor solução conhecida na literatura em 10 instâncias, conseguindo encontrar a melhor solução conhecida em duas instâncias: sko49-3 e sko49-5. O maior desvio padrão encontrado pelo SA foi de 365.4 na instância sko49-2 e o maior desvio encontrado pelo VNS/ILS foi de 141.9 na instância sko49-5. As médias do Simulated Annealing permaneceram relativamente próximas às melhores soluções encontradas.

Na Tabela 12 pode-se observar os tempos computacionais médios exibidos por ambos algoritmos, além do Simulated Annealing de Ahonen, De Alvarenga e Amaral (2014) em segundos. Os tempos exibidos pelo Simulated Annealing foram menores do que os tempos utilizados pelo pelo VNS/ILS. Os tempos computacionais do algoritmo Simulated Annealing de Ahonen, De Alvarenga e Amaral (2014) e o Simulated Annealing implementado neste trabalho não podem ser comparados por terem sido executados em hardwares diferentes, mas pode-se afirmar que cresceram de forma similar de acordo com o tamanho da instância. Já o tempo computacional exibido pelo algoritmo híbrido VNS/ILS cresceu muito com o aumento do tamanho das instâncias.

5.5.3 SF-MRFLP de 3 a 5 linhas paralelas

Neste trabalho, o problema do SF-MRFLP foi resolvido com mais de duas linhas paralelas, com k variando de 3 a 5. Os resultados obtidos pelos algoritmos implementados neste trabalho foram comparados com os resultados obtidos pela heurística SDP implementada por Hungerländer e Anjos (2015). Os resultados podem ser observados na Tabela 13.

A primeira coluna da tabela mostra os nomes das instâncias. A segunda coluna mostra o número de linhas paralelas k utilizadas por cada instância. A terceira coluna mostra os resultados mínimos obtidos pela heurística SDP nas instâncias que vão de H_5 até Rand_9 ou Rand_8. A quarta coluna mostra o tempo computacional utilizado pela heurística SDP para resolver cada instância. Nas três colunas seguintes pode-se observar os resultados mínimo (Mínimo), tempo computacional médio utilizado (Tempo) e desvio padrão (DP) do algoritmo Simulated Annealing implementado neste trabalho. Finalmente, nas três últimas colunas pode-se observar os resultados mínimo, tempo computacional médio utilizado e desvio padrão do algoritmo híbrido. Os valores em negrito indicam as melhores soluções encontradas para cada instância.

Tabela 13 – SF-MRFLP com 3 a 5 linhas paralelas

		SDP		SA			VNS/ILS		
Instância	k	Mínimo	Tempo (s)	Mínimo	Tempo (s)	DP	Mínimo	Tempo (s)	DP
H_5	3	**290,0**	8,0	**290,0**	0,184	0,0	**290,0**	0,005	0,0
Rand_5		**34,5**	7,6	**34,5**	0,138	0,0	**34,5**	0,006	0,0
H_6		**560,0**	32,8	**560,0**	0,142	0,0	**560,0**	0,010	0,0
Rand_6		**110,5**	36,4	**110,5**	0,139	0,0	**110,5**	0,011	0,0
H_7		**1190,0**	139,2	**1190,0**	0,155	0,0	**1190,0**	0,020	0,0
Rand_7		**108,0**	157,0	**108,0**	0,150	0,0	**108,0**	0,015	0,0
H_8		**1515,0**	638,0	**1515,0**	0,170	0,0	**1515,0**	0,025	0,0
S_8		**260,0**	709,1	**260,0**	0,197	0,0	**260,0**	0,027	0,0
SH_8		**740,5**	633,6	**740,5**	0,173	0,0	**740,5**	0,027	0,0
Rand_8		**138,0**	653,3	**138,0**	0,161	0,0	**138,0**	0,025	0,0
S_9		**771,5**	3074,5	**771,5**	0,181	0,0	**771,5**	0,048	0,0
SH_9		**1431,5**	2883,7	**1431,5**	0,187	0,0	**1431,5**	0,049	0,0
Rand_9		**317,5**	3055,0	**317,5**	0,233	0,0	**317,5**	0,045	0,0
H_5	4	**140,0**	3,5	**140,0**	0,125	0,0	**140,0**	0,005	0,0
Rand_5		48,5	4,5	**34,5**	0,133	0,0	**34,5**	0,005	0,0
H_6		**410,0**	34,9	**410,0**	0,132	0,0	**410,0**	0,013	0,0
Rand_6		**98,5**	40,0	**98,5**	0,128	0,0	**98,5**	0,010	0,0
H_7		**820,0**	232,2	**820,0**	0,146	0,0	**820,0**	0,021	0,0
Rand_7		**77,0**	190,1	**77,0**	0,141	0,0	**77,0**	0,013	0,0
H_8		**1135,0**	1211,1	**1135,0**	0,161	0,0	**1135,0**	0,026	0,0
S_8		**188,0**	1461,4	**188,0**	0,160	0,0	**188,0**	0,031	0,0
SH_8		**491,5**	1292,7	**491,5**	0,167	0,0	**491,5**	0,026	0,0
Rand_8		**98,0**	1411,1	**98,0**	0,155	0,0	**98,0**	0,023	0,0
H_5	5	200,0	0,2	**140,0**	0,129	0,0	**140,0**	0,005	0,0
Rand_5		44,5	0,2	**34,5**	0,134	0,0	**34,5**	0,005	0,0
H_6		**260,0**	7,0	**260,0**	0,121	0,0	**260,0**	0,012	0,0
Rand_6		**94,5**	6,7	**94,5**	0,118	0,0	**94,5**	0,011	0,0
H_7		**650,0**	87,6	**650,0**	0,135	0,0	**650,0**	0,021	0,0
Rand_7		**61,0**	83,3	**61,0**	0,131	0,0	**61,0**	0,012	0,0
H_8		**875,0**	981,6	**875,0**	0,155	0,0	**875,0**	0,025	0,0
S_8		**146,0**	1194,8	**146,0**	0,159	0,0	**146,0**	0,026	0,0
SH_8		**413,5**	911,8	**413,5**	0,146	0,0	**413,5**	0,025	0,0
Rand_8		**82,0**	967,2	**82,0**	0,147	0,0	**82,0**	0,022	0,0

Os algoritmos SA e VNS/ILS conseguiram alcançar os valores mínimos obtidos por Hungerländer e Anjos (2015). Em relação às instâncias Rand 5 (para k igual a 4 e 5) e H 5 (para k igual a 5), os valores encontrados são menores devido ao fato de que os algoritmos propostos não obrigaram a alocação de todas as k linhas, permitindo uma linha sem nenhuma facilidade a fim de obter melhor solução para a instância. Ambos algoritmos obtiveram a mesma solução em todas as execuções para todas as instâncias, ou seja, desvio padrão igual a 0. Os tempos computacionais permaneceram baixos, sendo que o Algoritmo Híbrido obteve tempos muito menores em comparação ao Simulated Annealing.

6 Métodos utilizados na resolução do DRFLP e MRFLP

Neste capítulo são apresentados algoritmos para resolver o DRFLP e MRFLP. A métrica utilizada considerou somente o eixo-horizontal no cálculo de distância de centro a centro entre um par de facilidades distinto. Diferentemente dos capítulos anteriores, o MRFLP permite espaços entre facilidades e não precisa iniciar de um ponto comum. Para resolvê-lo, as meta-heurísticas Simulated Annealing, VNS e ILS foram utilizadas, além de um modelo de Programação Linear.

6.1 Solução Inicial

A construção da solução inicial é feita de forma gulosa e aleatória, assim como a construção da solução inicial do SF-MRFLP mostrado no Algoritmo 5. O algoritmo inicia alocando uma facilidade aleatória no Layout. A solução inicial possui linhas que iniciam do eixo de origem e não possuem espaço, mas à medida que a heurística de melhoramento é aplicada, começam a aparecer espaços entre as facilidades.

6.2 Movimentos de Troca e Busca Local

Para o problema do MRFLP, foram utilizados os dois movimentos de troca utilizados no SF-MRFLP.

A troca simples consiste em trocar duas facilidades de lugar. Após a troca, possíveis sobreposições devem ser corrigidas mas podem haver espaços entre facilidades devido a essa troca. Tal característica é permitida no MRFLP por ser um problema de natureza contínua.

A troca de linha consiste em mover uma facilidade de uma linha para outra no layout, mantendo sua posição relativa na mesma. Por exemplo, se uma facilidade está posicionada na segunda posição da terceira linnha e é envolvida em uma troca de linha para a primeira linha, ela continuará na segunda posição da primeira linha. Esse movimento também respeita as restrições de que não pode haver sobreposições.

Assim como a busca local utilizada no Algoritmo Híbrido para o PROP e SF-MRFLP, a busca local implementada no MRFLP segue os mesmos passos do Algoritmo 5. A diferença é que os movimentos de troca utilizados são a troca simples e a troca de linha.

6.3 Programa Linear

Como o problema do Multi-row é de natureza contínua e não apenas binária e inteira, sua resolução é ainda mais difícil. Além das meta-heurísticas implementadas neste trabalho, foi aqui utilizado também um programa linear para obter melhores soluções.

Nesse caso especial, o objetivo principal das meta-heurísticas foi de encontrar uma boa permutação de facilidades nas linhas, ou seja, as posições relativas de cada facilidade em cada linha. Com esse layout, um programa linear é chamado para obter o posicionamento ótimo das facilidades no eixo horizontal de cada linha do layout.

O modelo de programação linear utilizado neste trabalho pode ser encontrado em (HERAGU; KUSIAK, 1991). A restrição 6.2 indica que não devem haver sobreposições no Layout. As restrições de 6.3 a 6.8 são utilizadas para tratar o valor absoluto encontrado na função objetivo e restrição 6.2. Para isso, a equação $|x_i - x_j|$ é transformada em uma soma de duas variáveis especiais chamadas $x_{ij}^+ + x_{ij}^-$. Para determinar se x_i é menor ou maior que x_j, nas restrições 6.5 e 6.8, são utilizadas as posições relativas fornecidas pelas meta-heurísticas, estipulando qual facilidade se encontra antes da outra no layout. c_{ij} é o fluxo entre cada par de facilidades i e j. Nos algoritmos implementados, o programa linear foi resolvido utilizando-se o software CPLEX 12.6.

$$\min_{i<j} \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} c_{ij}|x_i - x_j| \tag{6.1}$$

$$\text{sujeito a } |x_i - x_j| \geq \frac{1}{2}(l_i + l_j) + d_{ij}, i = 1, ..., n-1, j = i+1, ..., n \tag{6.2}$$

$$|x_i - x_j| = x_{ij}^+ + x_{ij}^- \tag{6.3}$$

$$(x_i - x_j) = x_{ij}^+ - x_{ij}^- \tag{6.4}$$

$$x_{ij}^+ = x_i - x_j \text{ se } x_i - x_j > 0 \tag{6.5}$$

$$x_{ij}^+ = 0 \text{ se } x_i - x_j \leq 0 \tag{6.6}$$

$$x_{ij}^- = x_i - x_j \text{ se } x_i - x_j \leq 0 \tag{6.7}$$

$$x_{ij}^- = 0 \text{ se } x_i - x_j > 0 \tag{6.8}$$

$$x_{ij}^-, x_{ij}^+, x \geq 0 \tag{6.9}$$

6.4 Simulated Annealing

O algoritmo Simulated Annealing utilizado para resolver o MRFLP também recebe os mesmos parâmetros utilizados no PROP: o número máximo de iterações SA_{max}, a taxa de resfriamento α e a temperatura inicial T_0. A perturbação é feita escolhendo-se aleatoriamente um dos movimentos de troca citados anteriormente para ser aplicado na solução corrente e gerar nova solução s'.

Com a solução s' gerada após a perturbação do método, uma nova permutação de posições relativas é atribuída à solução. Com isso, um programa linear é chamado para calcular as posições de cada facilidade no Layout. O pseudocódigo pode ser verificado em 10.

Algoritmo 10 Simulated Annealing

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

1: $sBest \leftarrow s$
2: $T \leftarrow T_0$
3: **para** $i \leftarrow 1$ **to** $SAmax$ **faça**
4: **para** $j \leftarrow 1$ **to** P **faça**
5: perturba solução gerando solução vizinha s'
6: resolve o programa linear com solução s'
7: $\Delta \leftarrow$ f(s') – f(s)
8: **se** $\Delta < 0$ **então**
9: $s \leftarrow s'$
10: **se** $f(s') < f(sBest)$ **então**
11: $sBest \leftarrow s'$
12: **fim se**
13: **senão**
14: faça $s \leftarrow s'$ de acordo com probabilidade $e^{\frac{-\Delta}{T}}$
15: **fim se**
16: **fim para**
17: **se** nenhuma solução melhor foi aceita **então**
18: $T \leftarrow \beta T$
19: **fim se**
20: $T \leftarrow \alpha T$
21: **fim para**

6.5 Algoritmo Híbrido

Na aplicação do Algoritmo Híbrido para o SF-MRFLP, foram aplicados os mesmos passos aplicados para o PROP e SF-MRFLP. Como no SF-MRFLP também são utilizados dois movimentos de troca, y tem valor máximo de 2 no Algoritmo 11. Se uma busca local é bem sucedida, y aponta novamente para o primeiro movimento de troca e as buscas recomeçam. A diferença é que no algoritmo implementado para resolver o MRFLP, foi

feita uma chamada do Programa Linear a cada iteração para fazer o posicionamento das facilidades nas linhas.

Algoritmo 11 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:        resolve o programa linear com solução s'
10:       y ← y + 1
11:       se f(s') < f(s) então
12:         s ← s'
13:         y ← 1
14:       fim se
15:     fim enquanto
16:     se f(s) < f(sBest) então
17:       sBest ← s
18:       p ← 1
19:     senão
20:       p ← p + 1
21:     fim se
22:   fim enquanto
23:   i ← i + 1
24: fim para
```

6.6 Resultados Computacionais

Os experimentos computacionais para o DRFLP e MRFLP foram realizados baseados em instâncias usadas em (HUNGERLÄNDER; ANJOS, 2015) totalizando um conjunto de 31 instâncias testadas.

Os parâmetros utilizados no algoritmo do Simulated Annealing e VNS/ILS foram os mesmos utilizados para o PROP.

Os testes computacionais para o DRFLP e MRFLP também foram feitos nas mesmas condições dos testes feitos para o PROP e SRFLP: algoritmos compilados com gcc 5.4 utilizando a linguagem C e rodados em um computador com processador Intel core i7 2.0 GHz com 8 GB de RAM utilizando sistema operacional Linux Ubuntu 16.04. Para cada instância, os algoritmos foram executados 5 vezes.

6.6.1 DRFLP e MRFLP com 3 a 5 linhas paralelas

Os algoritmos implementados foram executados para resolver instâncias pequenas de duas a cinco linhas do Multi-row Facility Layout Problem. Os resultados, mostrados em 14, foram comparados com os obtidos em (HUNGERLÄNDER; ANJOS, 2015), os quais utilizaram programação semi-definida (SDP) para resolver o DRFLP, para k igual a 2, e MRFLP, para k de 3 até 5. A primeira coluna mostra o nome de cada instância utilizada nos testes. A segunda coluna mostra o número de linhas k utilizados no Layout. A terceira e quarta coluna mostram, respectivamente, os valores mínimos (Min) encontrados pela heurística SDP de Hungerländer e Anjos (2015) e o tempo computacional médio (Tempo) utilizado para cada instância. Nas três colunas seguintes pode-se observar respectivamente os valores mínimos (Min), desvios padrão (DP) e tempos computacionais médios do algoritmo Simulated Annealing para cada instância, em segundos (Tempo). Nas três últimas colunas pode-se observar estes mesmos dados para o Algoritmo Híbrido.

Tabela 14 – DRFLP e MRFLP com 3 a 5 linhas paralelas

		Heurística SDP		SA			VNS/ILS		
Instância	k	Valor	Tempo (s)	Min	DP	Tempo (s)	Min	DP	Tempo (s)
H_5	2	**350,0**	4,8	**350,0**	0,0	21,05	**350,0**	0,0	0,35
Rand_5		**52,5**	4,5	**52,5**	0,0	21,35	**52,5**	0,0	0,27
H_6		**640,0**	11,1	**640,0**	15,0	28,96	**640,0**	0,0	0,59
Rand_6		**190,5**	13,8	**190,5**	0,0	29,55	**190,5**	0,0	0,60
H_7		**1660,0**	25,2	**1660,0**	0,0	32,74	**1660,0**	4,0	1,29
Rand_7		**159,0**	31,7	**159,0**	0,0	32,70	**159,0**	0,4	1,04
H_8		**2265,0**	91,3	**2265,0**	57,1	33,40	**2265,0**	0,0	1,71
S_8		**396,0**	91,6	**396,0**	0,0	33,15	**396,0**	1,0	1,40
SH_8		1125,5	87,4	**1123,0**	0,0	32,64	**1123,0**	0,2	1,94
Rand_8		**189,5**	82,2	**189,5**	0,0	29,45	**189,5**	1,6	1,35
S_9		**1179,0**	253,4	**1179,0**	0,0	27,47	**1179,0**	1,3	1,91
SH_9		2294,5	251,9	**2293,0**	0,0	29,29	**2293,0**	6,6	1,82
Rand_9		**486,5**	252,6	**486,5**	0,0	29,13	**486,5**	9,5	2,05
S_10		1353,5	713,0	**1351,0**	5,9	34,27	**1351,0**	7,3	2,89
Rand_10		**821,0**	698,2	**821,0**	8,5	30,12	**821,0**	6,0	3,69
S_11		**3424,5**	2127,0	**3424,5**	38,7	33,75	3427,5	12,4	4,02
Rand_11		**697,0**	2048,5	773,5	0,0	29,76	773,5	3,8	4,23
H_12		**8875,0**	5943,3	**8875,0**	208,9	34,83	9015,0	20,4	5,41
Rand_12		**788,0**	6389,5	1021,0	6,4	34,39	1035,0	16,0	5,64
H_5	3	**175,0**	680,6	**175,0**	0,0	18,47	**175,0**	0,0	0,30
Rand_5		**18,0**	893,2	**18,0**	0,0	13,49	**18,0**	0,0	0,18
H_5	4	**105,0**	2287,4	**105,0**	0,0	17,46	**105,0**	0,0	0,30
Rand_5		**10,0**	4360,1	**10,0**	0,0	10,19	**10,0**	0,0	0,13
H_5	5	**0,0**	271,3	**0,0**	0,0	10,60	**0,0**	0,0	0,22
Rand_5		**0,0**	6252,4	**0,0**	0,0	7,89	**0,0**	0,0	0,10

Ambos os algoritmos mostraram-se eficientes para o MRFLP para as instâncias testadas, alcançando a maioria dos valores obtidos na literatura. O algoritmo SA conseguiu alcançar os resultados da abordagem semi-definida para várias instâncias ou atingiu valores próximos, sendo que na média o Simulated Annealing conseguiu melhores resultados que o algoritmo híbrido. Os algoritmos conseguiram melhores soluções do que as conhecidas na literatura para três instâncias: SH_8, SH_9 e S_10, todas três com k igual a 2. Na média, o tempo computacional utilizado pelo Algoritmo Híbrido foi bem menor do que o do Simulated Annealing. A heurística SDP não foi executada no mesmo hardware que os algoritmos implementados, então não há como comparar os tempos diretamente.

6.6.2 DRFLP para instâncias grandes

Posteriormente o DRFLP foi resolvido para instâncias grandes encontradas na literatura e os resultados dos algoritmos implementados foram comparados com os valores obtidos pelo algoritmo ILS-LP implementado em (PERMANHANE, 2016), utilizando-se um total de 12 instâncias. Os resultados podem ser observados na Tabela 15. As instâncias N_30_01-05 possuem 30 facilidades e podem ser encontradas em (AHONEN; DE ALVARENGA; AMARAL, 2014); já as instâncias DRLP_01-07 podem ser encontradas em (AMARAL, 2016) e possuem tamanho igual a 40 facilidades. A primeira coluna mostra o nome de cada instância utilizada nos testes. As outras colunas mostram respectivamente os valores mínimos (Min), os desvios padrão (DP) e valores médios (Média) encontrados pelos algoritmos.

Tabela 15 – Instâncias grandes - DRFLP

	ILS			SA			VNS/ILS		
Instância	Min	DP	Média	Min	DP	Média	Min	DP	Média
N30_01	**4115,0**	0,0	4115,0	**4115,0**	3,6	4116,8	4121,0	15,2	4145,2
N30_02	10778,5	2,6	10781,3	**10772,0**	22,7	10794,9	10812,5	25,7	10844,0
N30_03	22697,5	2,4	22701,9	**22697,0**	3,7	22704,0	22784,0	102,0	22897,9
N30_04	28436,5	11,3	28453,5	**28407,5**	9,5	28414,9	28586,5	131,7	28737,2
N30_05	**57400,0**	26,9	57429,1	57422,0	282,2	57662,0	57865,0	184,6	58032,3
DRLP_01	**99525,0**	25,9	99554,2	99545,0	203,9	99733,1	100540,0	590,2	101406,1
DRLP_02	301023,0	23,7	301051,2	**300978,5**	1164,8	301940,5	301325,0	881,7	302318,4
DRLP_03	**416266,0**	31,1	416289,0	416324,0	900,9	416778,2	417006,0	1611,0	418725,3
DRLP_04	207573,0	8,0	207581,0	**207536,5**	670,8	208100,7	207993,0	777,1	208892,1
DRLP_05	193817,0	7,3	193824,2	**193748,0**	147,3	194041,8	194832,0	352,4	195266,6
DRLP_06	1881379,0	77,3	1881456,2	**1881349,5**	18585,3	1907604,6	1882369,5	3991,6	1887387,5
DRLP_07	545421,5	475,5	545897,0	**545110,5**	2550,1	547861,4	550291,0	3119,7	553808,4

Os tempos computacionais utilizados pelos algoritmos, em segundos, podem ser observados na tabela 16.

Tabela 16 – Tempos computacionais para instâncias grandes - DRFLP

Instância	ILS	SA	VNS/ILS
N30_01	14243,01	8945,33	311,25
N30_02	16365,29	9737,80	389,06
N30_03	15767,46	9376,22	336,88
N30_04	17557,01	9084,36	293,50
N30_05	16985,47	8951,31	328,24
DRLP_01	41657,89	11432,30	966,16
DRLP_02	72215,77	19551,16	1438,35
DRLP_03	77250,05	27328,30	1929,74
DRLP_04	30935,37	14797,62	1161,37
DRLP_05	25908,29	12814,87	1038,74
DRLP_06	50122,51	27062,44	1924,18
DRLP_07	22301,57	10786,92	845,37

O algoritmos SA conseguiu alcançar os valores mínimos da literatura na maioria das instâncias, conseguindo melhores resultados mínimos em 9 instâncias do conjunto. Já o algoritmo VNS/ILS não conseguiu alcançar nenhum valor mínimo encontrado na literatura. Os valores de desvio padrão dos algoritmos SA e VNS/ILS são maiores do que os obtidos pelo ILS, sendo que o desvio padrão do Simulated Annealing é menor do que o do híbrido na maioria das instâncias. Os tempos computacionais do Algoritmo Híbrido foram menores em comparação ao Simulated Annealing. Os tempos computacionais do algoritmo Simulated Annealing foram maiores por utilizar mais vezes o programa linear resolvido pelo CPLEX. A cada perturbação do SA o programa linear era chamado, diferentemente do algoritmo híbrido VNS/ILS, no qual o programa linear era chamado apenas depois de uma busca local ser realizada. Não há comparação dos tempos dos algoritmos implementados com o algoritmo ILS por não serem executados em mesmo hardware.

7 Considerações Finais

O problema de Layout, no geral, possui muitas aplicações práticas. Foram discutidos alguns destes problemas, assim como suas aplicações práticas e abordagens de resolução encontradas na literatura.

Em especial, os problemas PROP, CAP, DRFLP, MRFLP e SF-MRFLP foram resolvidos utilizando-se algoritmos heurísticos os quais são baseados nas conhecidas meta-heurísticas Simulated Annealing, VNS e ILS. Foi implementado um algoritmo híbrido VNS/ILS para resolver os problemas estudados. Para os problemas de natureza contínua, como o DRFLP e MRFLP, foi utilizado também um modelo de programação linear para fazer o posicinamento das facilidades nas linhas.

Para o PROP, os algoritmos conseguiram as mesmas melhores soluções conhecidas da literatura para todas as instâncias pequenas testadas, sendo que em testes posteriores o Simulated Annealing saiu-se melhor do que o Algoritmo Híbrido.

Para o CAP, o algoritmo Simulated Annealing implementado atingiu soluções boas, obtendo, de forma geral, valores próximos das soluções que podem ser encontradas na literatura. O algoritmo também conseguiu obter melhores soluções em duas instâncias, para os testes com instâncias grandes. Já para o SF-MRFLP, os algoritmos alcançaram os valores da literatura e conseguiram melhores resultados em duas instâncias.

Para o MRFRLP, foram atingidos quase todos os valores mínimos encontrados na literatura para pequenas instâncias, com melhores soluções encontradas para três instâncias. Para instâncias maiores, no DRFLP, o algoritmo SA saiu-se melhor e encontrou soluções semelhantes à da literatura, conseguindo melhores soluções em oito instâncias.

Em trabalhos futuros, serão buscadas novas alternativas para a modelagem e estrutura de vizinhança dos problemas estudados a fim de obter melhor eficiência nos cálculos da função objetivo e movimentos de troca, assim como a exploração de novas meta-heurísticas ou combinações das mesmas.

Referências

AHONEN, H.; DE ALVARENGA, A.; AMARAL, A. Simulated annealing and tabu search approaches for the corridor allocation problem. *European Journal of Operational Research*, v. 232, n. 1, p. 221 – 233, 2014. ISSN 0377-2217. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0377221713005808>. Citado 9 vezes nas páginas 16, 21, 24, 47, 48, 49, 52, 53 e 61.

AMARAL, A. R. S. On the exact solution of a facility layout problem. *European Journal of Operational Research*, v. 173, n. 2, p. 508 – 518, 2006. ISSN 0377-2217. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0377221705002031>. Citado 2 vezes nas páginas 15 e 20.

AMARAL, A. R. S. Enhanced local search applied to the single-row facility layout problem. *Simpósio Brasileiro de Pesquisa Operacional*, 2008. Disponível em: <http://www.din.uem.br/sbpo/sbpo2008/pdf/arq0026.pdf>. Citado na página 20.

AMARAL, A. R. S. An exact approach to the one-dimensional facility layout problem. *Operations Research*, v. 56, n. 4, p. 1026–1033, 2008. Disponível em: <http://dx.doi.org/10.1287/opre.1080.0548>. Citado na página 20.

AMARAL, A. R. S. A mixed 0-1 linear programming formulation for the exact solution of the minimum linear arrangement problem. *Optimization Letters*, v. 3, n. 4, p. 513–520, 2009. ISSN 1862-4480. Disponível em: <http://dx.doi.org/10.1007/s11590-009-0130-0>. Citado na página 20.

AMARAL, A. R. S. A new lower bound for the single row facility layout problem. *Discrete Applied Mathematics*, v. 157, n. 1, p. 183 – 190, 2009. ISSN 0166-218X. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0166218X0800259X>. Citado na página 20.

AMARAL, A. R. S. On duplex arrangement of vertices. Vitoria, Brasil, 2011. Citado na página 21.

AMARAL, A. R. S. The corridor allocation problem. *Computers & Operations Research*, v. 39, 2012. Citado 2 vezes nas páginas 16 e 21.

AMARAL, A. R. S. Optimal solutions for the double row layout problem. *Optimization Letters*, v. 7, n. 2, 2013. Citado na página 16.

AMARAL, A. R. S. A parallel ordering problem in facilities layout. *Computers & Operations Research*, v. 40, 2013. Citado 4 vezes nas páginas 16, 17, 21 e 37.

AMARAL, A. R. S. A heuristic approach for the double row layout problem. Vitoria, Brasil, 2016. Citado na página 61.

AMARAL, A. R. S.; LETCHFORD, A. N. A polyhedral approach to the single row facility layout problem. *Mathematical Programming*, Springer Berlin Heidelberg, v. 141, n. 1-2, p. 453–477, 2013. ISSN 0025-5610. Disponível em: <http://dx.doi.org/10.1007/s10107-012-0533-z>. Citado 4 vezes nas páginas 14, 15, 20 e 44.

ANJOS, M. F.; FISCHER, A.; HUNGERLÄNDER, P. Solution approaches for equidistant double- and multi-row facility layout problems. *Group for Research in Decision Analysis*, 2015. Disponível em: <https://www.gerad.ca/en/papers/G-2015-06>. Citado 2 vezes nas páginas 16 e 23.

ANJOS, M. F.; KENNINGS, A.; VANNELLI, A. A semidefinite optimization approach for the single-row layout problem with unequal dimensions. *Discrete Optimization*, v. 2, n. 2, p. 113 – 122, 2005. ISSN 1572-5286. Disponível em: <http://www.sciencedirect.com/science/article/pii/S1572528605000174>. Citado na página 40.

ANJOS, M. F.; VANNELLI, A. 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, 2008. Disponível em: <http://dx.doi.org/10.1287/ijoc.1080.0270>. Citado na página 20.

ANJOS, M. F.; YEN, G. Provably near-optimal solutions for very large single-row facility layout problems. *Optimization Methods and Software*, v. 24, n. 4-5, p. 805–817, 2009. Disponível em: <http://dx.doi.org/10.1080/10556780902917735>. Citado na página 42.

AZADIVAR, F.; WANG, J. Facility layout optimization using simulation and genetic algorithms. *International Journal of Production Research*, v. 38, n. 17, p. 4369–4383, 2000. Disponível em: <http://dx.doi.org/10.1080/00207540050205154>. Citado na página 12.

BATTITI, R.; PROTASI, M. Reactive search, a history-sensitive heuristic for max-sat. *J. Exp. Algorithmics*, ACM, New York, NY, USA, v. 2, jan. 1997. ISSN 1084-6654. Disponível em: <http://doi.acm.org/10.1145/264216.264220>. Citado na página 32.

BENJAAFAR, S.; SHEIKHZADEH, M. Design of flexible plant layouts. *IIE Transactions*, v. 32, n. 4, p. 309–322, 2000. ISSN 1573-9724. Disponível em: <http://dx.doi.org/10.1023/A:1007691303186>. Citado na página 13.

BOZER, Y. A.; MELLER, R. D.; ERLEBACHER, S. J. An improvement-type layout algorithm for single and multiple-floor facilities. *Management Science*, INFORMS, v. 40, n. 7, p. 918–932, 1994. ISSN 00251909, 15265501. Disponível em: <http://www.jstor.org/stable/2632922>. Citado na página 23.

BRITO, F. T. de; LOPES, H. dos S. O méodo slp integrado ao algoritmo cna para maior eficácia no planejamento de layout: estudo de caso em uma movelaria na cidade de dom eliseu - pa. *Simpósio Brasileiro de Pesquisa Operacional*, p. 36–47, 2014. Citado na página 12.

BRUSCO, M. J.; STAHL, S. Using quadratic assignment methods to generate initial permutations for least-squares unidimensional scaling of symmetric proximity matrices. *Journal of Classification*, v. 17, n. 2, p. 197–223, 2000. Citado na página 17.

CELLIN, B. D. P.; AMARAL, A. R. S. Simulated annealing aplicado ao problema de ordenação em linhas paralelas. *XLVII Simpósio Brasileiro de Pesquisa Operacional*, 2015. Citado 2 vezes nas páginas 21 e 24.

CHIARANDINI, M.; STÜTZLE, T. An application of iterated local search to graph coloring. p. 112–125, 2002. Disponível em: <http://iridia.ulb.ac.be/~stuetzle/publications/ChiStu02.COLOR.pdf>. Citado 2 vezes nas páginas 30 e 31.

CHUNG, J.; TANCHOCO, J. M. A. The double row layout problem. *International Journal of Production Research*, v. 48, n. 3, p. 709–727, 2010. Disponível em: <http://webbuild.knu.ac.kr/~chung/research/DRLP.pdf>. Citado na página 21.

DATTA, D.; AMARAL, A. R.; FIGUEIRA, J. R. Single row facility layout problem using a permutation-based genetic algorithm. *European Journal of Operational Research*, v. 213, n. 2, p. 388 – 394, 2011. ISSN 0377-2217. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0377221711002712>. Citado na página 20.

DE ALVARENGA, A.; NEGREIROS-GOMES, F. J.; MESTRIA, M. Metaheuristic methods for a class of the facility layout problem. *Journal of Intelligent Manufacturing*, Kluwer Academic Publishers, v. 11, n. 4, p. 421–430, 2000. ISSN 0956-5515. Disponível em: <http://dx.doi.org/10.1023/A%3A1008982420344>. Citado na página 20.

DICKEY, J. W.; HOPKINS, J. W. Campus building arrangement using topaz. *Transportation Research*, v. 6, n. 1, p. 59–68, 1972. Citado na página 17.

DRIRA, A.; PIERREVAL, H.; HAJRI-GABOUJ, S. Facility layout problems: A survey. *Annual Reviews in Control*, v. 31, n. 2, p. 255 – 267, 2007. ISSN 1367-5788. Disponível em: <http://www.sciencedirect.com/science/article/pii/S1367578807000417>. Citado 2 vezes nas páginas 13 e 21.

EL-BAZ, M. A. A genetic algorithm for facility layout problems of different manufacturing environments. *Computers & Industrial Engineering*, v. 47, n. 2–3, p. 233 – 246, 2004. ISSN 0360-8352. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0360835204001159>. Citado na página 22.

ELSHAFEI, A. N. Hospital layout as a quadratic assignment problem. v. 28, n. 1, p. 167–179, 1977. Citado na página 17.

FICKO, M.; BREZOCNIK, M.; BALIC, J. Designing the layout of single- and multiple-rows flexible manufacturing system by genetic algorithms. *Journal of Materials Processing Technology*, v. 157–158, p. 150 – 158, 2004. ISSN 0924-0136. Achievements in Mechanical and Materials Engineering Conference. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0924013604010544>. Citado na página 22.

FICKO, M. et al. Intelligent design of an unconstrained layout for a flexible manufacturing system. *Neurocomputing*, v. 73, n. 4–6, p. 639 – 647, 2010. ISSN 0925-2312. Bayesian Networks / Design and Application of Neural Networks and Intelligent Learning Systems (KES 2008 / Bio-inspired Computing: Theories and Applications (BIC-TA 2007). Disponível em: <http://www.sciencedirect.com/science/article/pii/S0925231209004056>. Citado na página 23.

GAREY, M. R.; JOHNSON, D. S. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. New York, NY, USA: W. H. Freeman & Co., 1979. ISBN 0716710447. Citado 2 vezes nas páginas 13 e 15.

GEOFFRION, A.; GRAVES, G. Scheduling parallel production lines with changeover costs: Practical applications of a quadratic assignment/lp approach. *Operations Re- search*, v. 24, p. 595–610, 1976. Citado na página 17.

GHOSH, D.; KOTHARI, R. Population heuristics for the corridor allocation problem. *Indian Institute of Management*, p. 1–13, Setembro 2012. Disponível em: <http://vslir.iimahd.ernet.in:8080/xmlui/bitstream/handle/123456789/11391/2012-09-02.pdf?sequence=1>. Citado na página 21.

GLOVER, F.; KOCHENBERGER, G. A. *Handbook of Metaheuristics*. Boston: Kluwer Academic Publishers, 2003. Citado 4 vezes nas páginas 14, 26, 27 e 28.

GUAN, J.; LIN, G. Hybridizing variable neighborhood search with ant colony optimization for solving the single row facility layout problem. *European Journal of Operational Research*, v. 248, n. 3, p. 899 – 909, 2016. ISSN 0377-2217. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0377221715007316>. Citado na página 20.

HASSAN, M. M. D. Machine layout problem in modern manufacturing facilities. *International Journal of Production Research*, v. 32, n. 11, p. 2559–2584, 1994. Disponível em: <http://dx.doi.org/10.1080/00207549408957084>. Citado 2 vezes nas páginas 14 e 21.

HERAGU, S. S. Recent models and techniques for solving the layout problem. *European Journal of Operational Research*, v. 57, n. 2, p. 136 – 144, 1992. ISSN 0377-2217. Disponível em: <http://www.sciencedirect.com/science/article/pii/037722179290038B>. Citado 2 vezes nas páginas 17 e 22.

HERAGU, S. S. Facilities design. Boston, 1997. Citado na página 12.

HERAGU, S. S.; ALFA, A. S. Experimental analysis of simulated annealing based algorithms for the layout problem. *European Journal of Operational Research*, v. 57, n. 2, p. 190 – 202, 1992. ISSN 0377-2217. Disponível em: <http://www.sciencedirect.com/science/article/pii/0377221792900428>. Citado na página 22.

HERAGU, S. S.; KUSIAK, A. Machine layout problem in flexible manufacturing systems. *Operations Research*, v. 36, n. 2, p. 258–268, 1988. Disponível em: <http://dx.doi.org/10.1287/opre.36.2.258>. Citado 3 vezes nas páginas 15, 17 e 22.

HERAGU, S. S.; KUSIAK, A. Efficient models for the facility layout problem. *European Journal of Operational Research*, v. 53, n. 1, p. 1 – 13, 1991. ISSN 0377-2217. Disponível em: <http://www.sciencedirect.com/science/article/pii/037722179190088D>. Citado 3 vezes nas páginas 15, 20 e 57.

HUNGERLÄNDER, P. Single-row equidistant facility layout as a special case of single-row facility layout. *International Journal of Production Research*, v. 52, n. 5, p. 1257–1268, 2014. Disponível em: <http://dx.doi.org/10.1080/00207543.2013.828163>. Citado na página 20.

HUNGERLÄNDER, P.; ANJOS, M. F. A semidefinite optimization approach to space-free multi-row facility layout. *Group for Research in Decision Analysis*, 2012. Disponível em: <https://www.gerad.ca/en/papers/G-2012-03>. Citado 2 vezes nas páginas 21 e 23.

HUNGERLÄNDER, P.; ANJOS, M. F. A semidefinite optimization-based approach for global optimization of multi-row facility layout. *European Journal of Operational Research*, v. 245, n. 1, p. 46 – 61, 2015. ISSN 0377-2217. Disponível em:

<http://www.sciencedirect.com/science/article/pii/S0377221715001691>. Citado 13 vezes nas páginas 17, 18, 21, 23, 24, 37, 38, 48, 49, 54, 55, 59 e 60.

HUNGERLÄNDER, P.; RENDL, F. A computational study and survey of methods for the single-row facility layout problem. *Computational Optimization and Applications*, Springer US, v. 55, n. 1, p. 1–20, 2013. ISSN 0926-6003. Disponível em: <http://dx.doi.org/10.1007/s10589-012-9505-8>. Citado na página 20.

JAIN, A. K.; KHARE, V. K.; MISHRA, P. M. Facility planning and associated problems: A survey. *Innovative Systems Design and Engineering*, v. 4, n. 6, p. 1–8, 2013. Disponível em: <www.iiste.org/Journals/index.php/ISDE/article/download/6015/6054>. Citado 2 vezes nas páginas 21 e 23.

JANKOVITS, I. et al. A convex optimisation framework for the unequal-areas facility layout problem. *European Journal of Operational Research*, v. 214, n. 2, p. 199 – 215, 2011. ISSN 0377-2217. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0377221711003560>. Citado na página 23.

JOHNSON, R. V. Spacecraft for multi-floor layout planning. *Management Science*, v. 28, n. 4, p. 407–417, 1982. Disponível em: <http://dx.doi.org/10.1287/mnsc.28.4.407>. Citado na página 23.

KELLER, B.; BUSCHER, U. Single row layout models. *European Journal of Operational Research*, v. 245, n. 3, p. 629 – 644, 2015. ISSN 0377-2217. Disponível em: <http://www.sciencedirect.com/science/article/pii/S037722171500209X>. Citado na página 14.

KIRKPATRICK, S.; GELATT, C. D.; VECCHI, M. P. Optimization by simulated annealing. *Science*, American Association for the Advancement of Science, v. 220, n. 4598, p. 671–680, 1983. ISSN 0036-8075. Disponível em: <http://science.sciencemag.org/content/220/4598/671>. Citado na página 26.

KOCHHAR, J. S.; HERAGU, S. S. Multi-hope: A tool for multiple floor layout problems. *International Journal of Production Research*, v. 36, n. 12, p. 3421–3435, 1998. Citado na página 17.

KOOPMANS, T.; BECKMANN, M. J. *Assignment Problems and the Location of Economic Activities*. [S.l.], 1955. Disponível em: <http://EconPapers.repec.org/RePEc:cwl:cwldpp:4>. Citado na página 12.

KOTHARI, R.; GHOSH, D. An efficient genetic algorithm for single row facility layout. *Optimization Letters*, v. 8, n. 2, p. 679–690, 2013. ISSN 1862-4480. Disponível em: <http://dx.doi.org/10.1007/s11590-012-0605-2>. Citado na página 20.

KOTHARI, R.; GHOSH, D. Insertion based lin–kernighan heuristic for single row facility layout. *Computers & Operations Research*, v. 40, n. 1, p. 129 – 136, 2013. ISSN 0305-0548. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0305054812001219>. Citado na página 20.

KOTHARI, R.; GHOSH, D. 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, 2013. ISSN 0377-2217. Disponível em:

<http://www.sciencedirect.com/science/article/pii/S0377221712005942>. Citado na página 20.

KOTHARI, R.; GHOSH, D. A scatter search algorithm for the single row facility layout problem. *Journal of Heuristics*, v. 20, n. 2, p. 125–142, 2014. ISSN 1862-4480. Disponível em: <http://dx.doi.org/10.1007/s11590-012-0605-2>. Citado na página 20.

LAPORTE, G.; MERCURE, H. Balancing hydraulic turbine runners: A quadratic assignment problem. *European Journal of Operational Research*, v. 35, n. 3, p. 378–381, 1988. Citado na página 17.

LEE, K.-Y.; ROH, M.-I.; JEONG, H.-S. An improved genetic algorithm for multi-floor facility layout problems having inner structure walls and passages. *Computers & Operations Research*, v. 32, n. 4, p. 879 – 899, 2005. ISSN 0305-0548. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0305054803002752>. Citado na página 23.

LEE, Y. H.; LEE, M. H. A shape-based block layout approach to facility layout problems using hybrid genetic algorithm. *Computers & Industrial Engineering*, v. 42, n. 2–4, p. 237 – 248, 2002. ISSN 0360-8352. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0360835202000189>. Citado 3 vezes nas páginas 13, 17 e 22.

LOURENÇO, H. R.; MARTIN, O. C.; STÜTZLE, T. Iterated local search. In: _____. *Handbook of Metaheuristics*. Boston, MA: Springer US, 2003. p. 320–353. ISBN 978-0-306-48056-0. Disponível em: <http://dx.doi.org/10.1007/0-306-48056-5_11>. Citado 3 vezes nas páginas 30, 31 e 32.

LOURENÇO, H. R.; ZWIJNENBURG, M. Combining the large-step optimization with tabu-search: Application to the job-shop scheduling problem. In: _____. *Meta-Heuristics: Theory and Applications*. Boston, MA: Springer US, 1996. p. 219–236. ISBN 978-1-4613-1361-8. Disponível em: <http://dx.doi.org/10.1007/978-1-4613-1361-8_14>. Citado na página 31.

LOVE, R. F.; WONG, J. Y. On solving a one-dimensional space allocation problem with integer programming. *INFOR*, v. 14, p. 139 – 144, 1976. Citado na página 20.

MARQUES, C. F. *Estratégia da gestão da produção e operações*. Curitiba: Digital, 2012. Citado na página 12.

MARTIN, O.; OTTO, S. W.; FELTEN, E. W. Large-step markov chains for the traveling salesman problem. 1991. Disponível em: <http://digitalcommons.ohsu.edu/csetech/16>. Citado na página 32.

MAVRIDOU, T. D.; PARDALOS, P. M. Simulated annealing and genetic algorithms for the facility layout problem: A survey. *Computational Optimization and Applications*, Kluwer Academic Publishers, v. 7, n. 1, p. 111–126, 1997. ISSN 0926-6003. Disponível em: <http://dx.doi.org/10.1023/A%3A1008623913524>. Citado na página 21.

MELLER, R. D.; BOZER, Y. A. A new simulated annealing algorithm for the facility layout problem. *International Journal of Production Research*, v. 34, n. 6, p. 1675–1692, 1996. Disponível em: <http://dx.doi.org/10.1080/00207549608904990>. Citado na página 23.

MELLER, R. D.; BOZER, Y. A. Alternative approaches to solve the multi-floor facility layout problem. *Journal of Manufacturing Systems*, v. 16, n. 3, p. 192 – 203, 1997. ISSN 0278-6125. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0278612597888875>. Citado na página 23.

MELLER, R. D.; NARAYANAN, V.; VANCE, P. H. Optimal facility layout design1. *Operations Research Letters*, v. 23, n. 3–5, p. 117 – 127, 1998. ISSN 0167-6377. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0167637798000248>. Citado na página 12.

MIAO, Z.; XU, K. lin. Research of multi-rows facility layout based on hybrid algorithm. In: *Information Management, Innovation Management and Industrial Engineering, 2009 International Conference on*. [S.l.: s.n.], 2009. v. 2, p. 553–556. Citado na página 22.

MLADENOVIĆ, N.; HANSEN, P. Variable neighborhood search. *Computers & Operations Research*, v. 24, n. 11, p. 1097 – 1100, 1997. ISSN 0305-0548. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0305054897000312>. Citado 3 vezes nas páginas 27, 29 e 30.

MURRAY, C. C.; SMITH, A. E.; ZHANG, Z. An efficient local search heuristic for the double row layout problem with asymmetric material flow. *International Journal of Production Research*, v. 51, n. 20, p. 6129–6139, 2013. Disponível em: <http://dx.doi.org/10.1080/00207543.2013.803168>. Citado na página 21.

NEARCHOU, A. C. Meta-heuristics from nature for the loop layout design problem. *International Journal of Production Economics*, v. 101, n. 2, p. 312–328, 2006. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0925527305000484>. Citado 2 vezes nas páginas 12 e 17.

OLIVEIRA, C. *Heurísticas para o Problema de Rotulação Cartográfica de Pontos e de Coloração de Vértices com Pesos*. dissertação de mestrado — Universidade Federal de Minas Gerais - UFMG, 2012. Citado na página 14.

OLIVEIRA, F. D. de et al. Propostas de melhorias no layout e no fluxo de processo de uma empresa brasileira de alimentos. *Simpósio Brasileiro de Pesquisa Operacional*, p. 72–83, 2014. Citado 2 vezes nas páginas 12 e 13.

OZCELIK, F. A hybrid genetic algorithm for the single row layout problem. *International Journal of Production Research*, v. 50, n. 20, p. 5872–5886, 2012. Disponível em: <http://dx.doi.org/10.1080/00207543.2011.636386>. Citado na página 20.

PALUBECKIS, G. A branch-and-bound algorithm for the single-row equidistant facility layout problem. *OR Spectrum*, v. 334, p. 1–21, 2012. Citado 2 vezes nas páginas 15 e 20.

PALUBECKIS, G. Fast local search for single row facility layout. *European Journal of Operational Research*, v. 246, n. 3, p. 800 – 814, 2015. ISSN 0377-2217. Disponível em: <http://www.sciencedirect.com/science/article/pii/S037722171500466X>. Citado na página 20.

PALUBECKIS, G. Fast simulated annealing for single-row equidistant facility layout. *Applied Mathematics and Computation*, v. 263, p. 287 – 301, 2015. ISSN 0096-3003. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0096300315005354>. Citado na página 21.

PALUBECKIS, G. Single row facility layout using multi-start simulated annealing. *Computers & Industrial Engineering*, v. 103, p. 1 – 16, 2017. ISSN 0360-8352. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0360835216303667>. Citado na página 20.

PERMANHANE, R. M. *Problemas de Layout*. dissertação de mestrado — Universidade Federal do Espirito Santo - UFES, 2016. Citado 3 vezes nas páginas 21, 24 e 61.

PICARD, J.-C.; QUEYRANNE, M. On the one-dimensional space allocation problem. *Operations Research*, v. 29, n. 2, p. 371–391, 1981. Disponível em: <http://dx.doi.org/10.1287/opre.29.2.371>. Citado 3 vezes nas páginas 12, 15 e 20.

POLLATSCHEK, M.; GERSHONI, N.; RADDAY, Y. Optimization of the typewriter keyboard by computer simulation. *Angewandte Informatik*, v. 10, p. 438–439, 1976. Citado na página 17.

RUSSELL, S. J.; NORVIG, P. *Artificial Intelligence: A modern approach*. [S.l.]: Prentice Hall, 2009. Citado na página 25.

SANJEEVI, S.; KIANFAR, K. A polyhedral study of triplet formulation for single row facility layout problem. *Discrete Applied Mathematics*, v. 158, n. 16, p. 1861 – 1867, 2010. ISSN 0166-218X. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0166218X1000257X>. Citado na página 20.

SIMMONS, D. M. One-dimensional space allocation: An ordering algorithm. *Operations Research*, v. 17, n. 5, p. 812–826, 1969. Disponível em: <http://dx.doi.org/10.1287/opre.17.5.812>. Citado 2 vezes nas páginas 15 e 20.

SINGH, S. P.; SHARMA, R. R. K. Two-level modified simulated annealing based approach for solving facility layout problem. *International Journal of Production Research*, v. 46, n. 13, p. 3563–3582, 2008. Disponível em: <http://dx.doi.org/10.1080/00207540601178557>. Citado na página 21.

SLACK, N. *Administração da Produção, edição compacta*. São Paulo: Atlas, 2006. Citado 2 vezes nas páginas 12 e 13.

STEINBERG, L. The backboard wiring problem: A placement algorithm. *SIAM Review*, v. 3, n. 1, p. 37–50, Jan 1961. Citado na página 17.

STÜTZLE, T. Applying iterated local search to the permutation flow shop problem. 1998. Citado na página 31.

SURYANARAYANAN, J. K.; GOLDEN, B. L.; WANG, Q. A new heuristic for the linear placement problem. *Computers & Operations Research*, v. 18, n. 3, p. 255 – 262, 1991. ISSN 0377-2217. Citado na página 15.

TALBI, E. *Metaheuristics: From Design to Implementation*. [S.l.]: Wiley Publishing, 2009. ISBN 9780470278581. Citado na página 14.

TASADDUQ, I. A.; IMAM, M.; AHMAD, A.-R. A novel metasearch algorithm for facility layout optimization. *International Conference on Computers & Industrial Engineering*, v. 41, p. 854–859, 2011. Disponível em: <http://www.usc.edu/dept/ise/caie/Checked%20Papers%20%5Bruhi%2012th%20sept%5D/word%20format%20papers/

REGISTRATION%20PAID%20PAPERS%20FOR%20PROCEEDINGS/pdf/282%2013%20A%20NOVEL%20METASEARCH%20ALGORITHM%20FOR%20FACILITY%20LAYOUT%20OPTIMIZATION.pdf>. Citado na página 23.

TATE, D. M.; SMITH, A. E. A genetic approach to the quadratic assignment problem. *Computers & Operations Research*, v. 22, n. 1, p. 73 – 83, 1995. ISSN 0305-0548. Disponível em: <http://www.sciencedirect.com/science/article/pii/0305054893E0020T>. Citado na página 21.

TOMPKINS, J. A. et al. Facilities planning. New York, 1996. Citado na página 13.

WESS, B.; ZEITLHOFER, T. On the phase coupling problem between data memory layout generation and address pointer assignment. In: SCHEPERS, H. (Ed.). *Software and Compilers for Embedded Systems*. Springer Berlin Heidelberg, 2004, (Lecture Notes in Computer Science, v. 3199). p. 152–166. ISBN 978-3-540-23035-9. Disponível em: <http://dx.doi.org/10.1007/978-3-540-30113-4_12>. Citado na página 17.

YANG, T.; PETERS, B. A. A spine layout design method for semiconductor fabrication facilities containing automated material-handling systems. *International Journal of Operations & Production Management*, Emerald, v. 17, n. 5, p. 490–501, 05 1997. ISSN 0144-3577. Citado na página 16.

ZHANG, Z.; MURRAY, C. C. A corrected formulation for the double row layout problem. *International Journal of Production Research*, v. 50, n. 15, p. 4220–4223, 2012. Disponível em: <http://dx.doi.org/10.1080/00207543.2011.603371>. Citado na página 21.