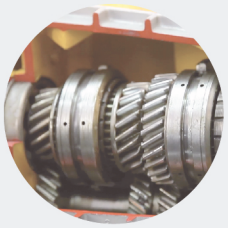
INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO
MESTRADO EM ENGENHARIA E GESTÃO INDUSTRIAL






Programação da produção: Otimização de Layouts Industriais

NUNO MANUEL BOLÉO TELES DE JESUS
Setembro de 2017

POLITÉCNICO
DO PORTO

Instituto Politécnico do Porto
Instituto Superior de Engenharia do Porto

Nuno Manuel Boleo Teles de Jesus

Programação da produção: Otimização de Layouts Industriais

Dissertação de Mestrado
Mestrado em Engenharia e Gestão Industrial
Orientação: Professora Doutora Isabel Cristina Lopes
Co-orientação: Professor Doutor Manuel Cruz

Porto, Setembro de 2017

Instituto Politécnico do Porto
Instituto Superior de Engenharia do Porto

Nuno Manuel Boleo Teles de Jesus

Programação da produção: Otimização de Layouts Industriais

Dissertação de Mestrado
Mestrado em Engenharia e Gestão Industrial
Orientação: Professora Doutora Isabel Cristina Lopes
Co-orientação: Professor Doutor Manuel Cruz

Porto, Setembro de 2017

Nuno Manuel Boleo Teles de Jesus

Programação da produção: Otimização de Layouts Industriais

Dissertação de Mestrado

Mestrado em Engenharia e Gestão Industrial

Membros do Júri

Presidente

Professor(a) Doutor(a) [Nome do Presidente do Júri]

Escola Superior de Estudos Industriais e de Gestão – Instituto Politécnico do Porto

Professor(a) Doutor(a) [Nome do elemento do Júri]

[Designação da Unidade Orgânica] – [Designação da Instituição]

Professor(a) Doutor(a) [Nome do elemento do Júri]

[Designação da Unidade Orgânica] – [Designação da Instituição]

Porto, Setembro de 2017

AGRADECIMENTOS

Gostaria de agradecer em primeiro lugar à minha orientadora, Professora Doutora Isabel Cristina Lopes pelo apoio, disponibilidade, motivação e confiança transmitida durante este estudo.

Ao meu coorientador, Professor Doutor Manuel Cruz pelo apoio e disponibilidade.

Aos meus colegas de mestrado, pela partilha de conhecimento ao longo destes dois anos.

À Escola Superior de Estudos Industriais e de Gestão, pelas condições proporcionadas para a realização deste mestrado.

Aos meus Pais e irmão que sempre acreditaram em mim, pelos seus ensinamentos e conselhos transmitidos ao longo da vida.

A todos os meus amigos pela paciência e tolerância com que me apoiaram e compreensão manifestada pelos momentos de companhia e dedicação que lhes retirei, no decurso da elaboração deste trabalho.

E a todos que de certa forma contribuíram para a realização desta tese.

RESUMO ANALÍTICO

O problema conhecido na literatura como "*Facility layout problem* (FLP)", em que se pretende determinar a disposição de recursos de produção e a sua interação num determinado espaço, é um problema estratégico para a implementação do chão de fábrica de uma empresa pelo impacto que tem na performance da produção.

O problema consiste em encontrar um posicionamento único entre instalações (departamentos, máquinas, células de produção, armazéns, etc.) e localizações no chão de fábrica, de forma a otimizar um ou mais objetivos de produção. O objetivo da criação de *layout* consiste na otimização do espaço existente, minimização do tempo de produção, redução do custo de manuseamento de matérias, aumento do grau de flexibilidade, entre outros.

A solução do problema deverá especificar a localização relativa de cada departamento (*layout em bloco*) e numa fase posterior poderá especificar o *layout* detalhado dentro de cada departamento.

Na presente tese serão apresentados alguns modelos matemáticos para criação de um *layout*, neste caso vamos usar uma formulação matemática *Quadratic Assignment Problem* (QAP), uma formulação matemática *Mixed Integer Programming* (MIP) e uma heurística de *Particle Swarm Optimization (*PSO*)* para resolver problemas de *layout*. Todas estas formulações e modelos serão postos em prática para a resolução de problemas fictícios.

Numa primeira abordagem iremos resolver problemas fictícios onde abordaremos a formulação QAP para problemas de atribuição de espaço de duas dimensões (x,y) e MIP e em seguida iremos usar a heurística PSO para a resolução de problemas em escala maior e real.

Palavras-chave: *Facility layout problem, Quadratic Assignment Problem, Mixed Integer Programming Problem e Particle Swarm Optimization*

ABSTRACT

The problem known in the literature as "Facility layout problem (FLP)", which is intended to determine the physical layout of industrial facilities, is a strategic problem for the implementation of a company by the impact it has on the production performance.

The problem is to find an unambiguous allocation between facilities (departments, machines, production cells, warehouses, etc.) and locations on the shop floor in order to optimize one or more production goals. The objectives often considered are the optimization of the space, minimizing production time, reduce the handling costs of materials, increased flexibility, among others.

The solution of the problem should specify the relative location of each department (block layout) and at a later stage it can specify the detailed layout within each department.

In this thesis will be presented some methods of resolution in this case we use a discrete Quadratic Assignment formulation (QAP), a Mixed Integer Linear Programming formulation (MIP) and a Particle Swarm Optimization heuristic (PSO) to solve layout problems. All these heuristics will be implemented for solving fictitious problems.

In a first approach we will solve simpler problems where we use the QAP and MIP formulation and following we will use the PSO heuristic to solve problems on a larger scale.

Keywords: *Facility layout problem, Quadratic Assignment Problem, Mixed Integer Programming Problem e Particle Swarm Optimization*

SUMÁRIO

Lista de tabelas/ilustrações/siglas

Lista de tabelas

Lista de siglas

GT - *Group technology*;

JIT - *Just-in-time*;

FMS - *Flexible Manufacturing Systems*;

FLP - *Facility layout problem*;

QAP – *Quadratic Assignment Problem;*

MIP – *Mixed Integer Programming;*

PSO - *Particle Swarm Optimization;*

FMS - Flexible Manufacturing Systems

FDMS - Sistema Fuzzy de Tomada de Decisões

Capítulo 1 - Introdução

1. Introdução

A necessidade de uma maior flexibilidade dos ambientes industriais face a fatores como as exigências e instabilidade dos mercados, conjunturas económicas, sociais, inovação tecnológica, a própria concorrência, e a exigência dos consumidores, levam à necessidade das empresas procederem a reestruturações externas e internas.

Na planificação de um *layout* de raiz ou até numa reestruturação de *layout* já existente, existe o problema de alocar corretamente máquinas, trabalhadores entre outros. Problema esse mais conhecido na literatura como "*Facility layout problem (FLP)*", em que se pretende determinar um arranjo físico nas instalações industriais. Este problema pretende a implementação de um *layout* numa empresa visando a que tenha um melhor impacto na performance da produção.

O problema (FLP) consiste em encontrar uma associação clara entre instalações (departamentos, máquinas, células de produção, armazéns, etc) e localizações no chão de fábrica, de forma a otimizar um ou mais objetivos de produção. Os objetivos frequentemente considerados são a otimização do espaço existente, minimização do tempo de produção, redução dos custos de manuseamento de materiais, aumento do grau de flexibilidade, etc.

A solução deste tipo de problemas passa por se especificar a localização relativa de cada departamento, no qual numa fase inicial usar-se-á o *layout* em blocos e posteriormente poderá especificar o *layout* detalhado dentro de cada departamento.

Os problemas de otimização de *layout* industriais são bastante complexos e na sua generalidade são NP-difíceis o que é necessário uma elevada complexidade computacional. Para estes problemas, existem vários modelos matemáticos, entre os quais o modelo clássico de programação quadrática proposto por Koopmans and Beckman (1957) um modelo de programação inteira mista proposto por (Montreuil, 1991), e modelos exatos baseados em grafos, bem como diversos algoritmos que permitem obter boas soluções baseados em procedimentos heurísticos, tais como tabu *search*, *simulated annealing*, algoritmos genéticos, *Particle Swarm Optimization entre outros*.

1.1. Objetivo

O estudo principal deste trabalho incide sobre a atualidade do *facility layout problem* e quais os seus modelos de resolução nas empresas industriais. Desta forma, foi feita uma vasta pesquisa baseada em referências teóricas acerca de sistemas de produção, formas e dimensões, sistemas de manuseamento, formulações de problemas, heurísticas, entre outros.

Apos uma vasta pesquisa, verificou-se que se pode dividir as resoluções de f*acility layout problem* em dois tipos de categorias: métodos exatos, métodos heurísticos aproximados. Nos métodos exatos, temos modelos nos quais nos dão a melhor solução

possível, modelos esses que podem ser resolvidos através do método de programação dinâmica e método *branch-and-bound*. Neste caso vamos tratar com mais relevância a *Quadratic Assignment Problem* (QAP) e *Mixed Integer Programming* (MIP), já os métodos heurísticos temos vários modelos tais como, Simulated Annealing, Tabu Search, algoritmo genético, Particle Swarm Optimization (PSO).

De todos os modelos falados anteriormente, iremos fazer uma abordagem prática de (QAP), (MIP) e (PSO) no qual serão postos a resoluções de problemas fictícios.

1.2. Estrutura da dissertação

Esta dissertação foi estruturada de forma a permitir uma leitura fluída e clara, dividindo-se em 5 partes distintas.

No primeiro capítulo apresentam-se os objetivos que se pretende atingir com esta tese, assim como a estrutura da dissertação. De seguida no segundo capítulo é apresentada uma abordagem genérica sobre o conceito de *layout* e a aplicação do mesmo em instalações industriais, visando os objetivos, restrições, condicionalismos e tipos de *layout*.

No terceiro capítulo, apresentamos algumas formulações matemáticas para a resolução de questões a nível de *layout* estático ou dinâmico. Dependendo da formulação e restrições do *layout* estes podem ser resolvidos de formas diferentes.

No quarto capítulo expõem-se alguns métodos de otimização de f*acility layout problem* no qual nos métodos exatos explicaremos o funcionamento do método de *branch-and-bound*. Nos métodos heurísticos explicaremos o principal funcionamento dos métodos de *Simulated Annealing*, *Tabu Search*, Algoritmo Genético, *Particle Swarm Optimization* (PSO).

No quinto capítulo descrevem-se as experiências computacionais realizadas, analisando e comparando os resultados obtidos, de várias formulações possíveis para *layout*. No sexto e último capítulo apresentam-se as conclusões obtidas com a realização deste estudo e as perspetivas de desenvolvimento futuro.

Capítulo 2 – Conceitos genéricos de *layout*

2. Conceitos genéricos de layout

O conceito de *layout*, ou arranjo físico, está normalmente associado a instalações industriais. No entanto, podemos encontrar a aplicação do conceito noutros tipos de instalações, tais como, armazéns, escritórios, lojas comerciais, etc. Podemos dizer então que o significado de *layout* é a forma como os componentes de um determinado espaço se encontram distribuídos nessa área. Desta forma, Heragu (1997) classifica o *layout* de uma instalação como sendo a organização de tudo o que é necessário para a produção de produtos ou prestação de serviços. Sendo a instalação, a área que facilita o desempenho de qualquer trabalho.

O *layout* é uma característica inerente a qualquer operação, visto que determina a sua forma, aparência e a maneira como materiais, informações e clientes fluem através dessa operação (Slack, et al., 1997). Tompkins, et al. (1996), define o *layout* de um armazém como a melhor forma de aproveitar a área de armazenamento, atendendo à estruturação entre os vários operadores, equipamentos e espaço disponível.

Normalmente, o estudo do planeamento de um l*ayout* é abordado como fator quantitativo, tendo como objetivo minimizar o custo de movimentação de pessoas e materiais numa instalação (Mecklenburgh, 1985 e Francis et al.,1992). No entanto, segundo Francis e White (1974), o estudo de um *layout* deve considerar fatores qualitativos como a segurança da instalação, a flexibilidade do *layout*, o ruído e a estética.

O conceito de *layout* converge em tantas aplicações no que respeita ao seu objetivo final, ou seja, criar a melhor disposição dos constituintes humanos e materiais para que o processo específico se desenvolva ao mais baixo custo, obtendo a maior segurança, eficiência e rentabilidade. A fins de atingir os objetivos propostos no planeamento do *layout* existe uma grande variedade de restrições a ter em conta para determinar a melhor solução a aplicar.

Quando uma nova instalação é projetada de raiz, a disposição das instalações deve ser integrada no projeto arquitetónico. O tamanho das divisões e as suas próprias formas são muitas vezes limitações que podem influenciar fortemente as configurações do *layout*. Em outras situações, como empresas que retifiquem ou modifiquem o seu *layout*, muitas vezes torna-se necessário mudar os tamanhos de divisões e formas.

2.1. Objetivos de planeamento e implementação de *layout*

Os objetivos básicos no desenvolvimento de uma disposição das instalações devem ser, a funcionalidade e a economia de custos. No geral, a funcionalidade inclui aspetos de um *layout* que podem não ser imediatamente quantificáveis, tais como facilitar a comunicação e melhorar a moral do pessoal.

As economias de custos incluem a redução dos tempos de viagem entre as áreas, o que minimiza o espaço necessário e permite a redução de pessoal, colocando

funções de trabalho semelhantes próximas umas das outras. Segundo Muther, (1978) dois elementos chave destas metas são economia de espaço e a redução do tempo de viagem entre os departamentos.

A quantidade de espaço alocado para um determinado departamento, muitas vezes é definida por fatores fora do controlo do planificador, cujo trabalho, sendo assim, é aproveitar ao máximo esse espaço, Mason (1989). Um espaço de trabalho mal projetado prejudica a produtividade e qualidade do mesmo.

Outro aspeto prejudicial é a distância de viagem entre os departamentos, a qual tem um custo que pode chegar a proporções enormes ao longo do tempo. O que pode parecer para o planificador uma curta distância, pode tornar-se, ao longo da vida de uma instalação, em dias perdidos a viajar. Isso não só aumenta os custos, mas também enfraquece o moral do pessoal.

O *layout* tem implicações de grande relevo perante a qualidade, produtividade e competitividade de uma empresa. As decisões de *layout* afetam significativamente a eficiência com que os trabalhadores podem realizar o seu trabalho, como a produção mais rápida e eficiente dos produtos. A automatização de um sistema de *layout*, muitas vezes pode ser difícil, uma vez que o sistema possa ter mudanças de produtos ou design de serviço, um novo mix de produtos ou um grande volume de fabricação.

A disposição das instalações é um processo complexo no qual engloba muitas variáveis, variáveis essas, que podem mudar com tempo, disponibilidade de espaço, recursos ilimitados tanto em matéria como mão-de-obra, escala de produtividade entre outras.

Assim sendo, para a criação de um *layou*t de origem ou modificar um já existente, teremos que ter em consideração os seguintes aspetos dos sistemas de manufatura apresentados na Figura 1.

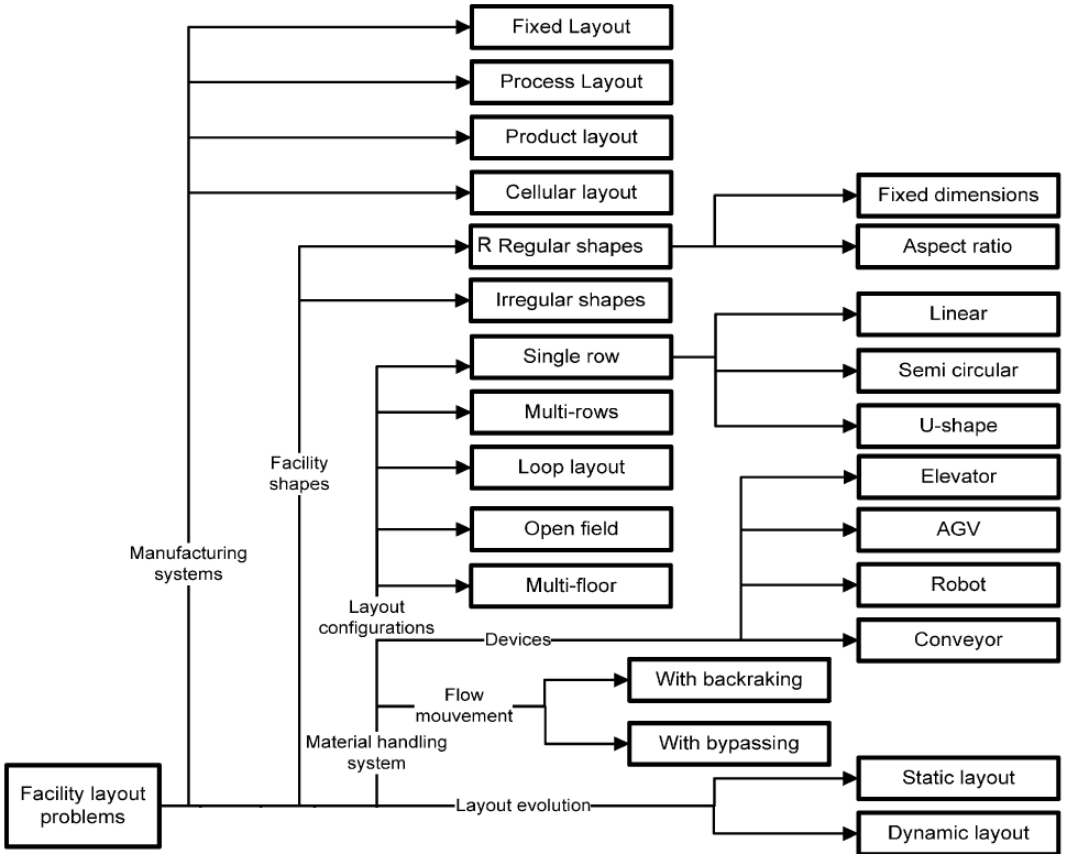


Figura 1 – Caraterísticas de layout Fonte: Drira 2007

Tendo em conta todas estas características para a criação de um *layout*, iremos mais à frente, explorar os diferentes tipos e caraterísticas de *layout* que as empresas podem utilizar.

2.2. Sistemas de Produção

Na contextualização do problema, serão analisados, de seguida, quatro projetos de sistemas de manufatura que caracterizam o panorama atual das indústrias que trabalham com lotes pequenos e médios de fabricação, tendo como finalidade conduzir a um modelo que vise justificar o porquê de procurar uma nova situação de *layout*.

Os sistemas identificados podem ser classificados em *layout* por processo (*process layout*), *layout* por produto (*product layout*), *layout* posicional (*fixed position layout*), e o *layout* em grupo (*cellular layout*).

2.2.1. *Layout* por processo

O *layout* por processo, também conhecido como *process layout*, tem como principal caraterística a agrupação das atividades em departamentos semelhantes ou secções de trabalho de acordo com o processo ou função que desempenham.

Por exemplo, numa empresa industrial, os tornos deverão ser posicionados numa secção, as fresadoras noutra secção, bem como as prensas e assim sucessivamente. Uma loja de vestuário também se encontra agrupada em secções, em que as roupas femininas, roupas masculinas, roupas infantis, cosméticos e calçados estão localizados em departamentos separados.

A figura 2 mostra um diagrama esquemático de um exemplo de *layout* por processo numa empresa de serviços.

Lingerie de Mulher	**Sapatos**	**Utensílios de Cozinha**
Vestidos de Mulher	**Cosméticos e Joias**	**Secção de Criança**
Roupa de Desporto	**Entrada / Saída da Loja**	**Secção de Homem**

Figura 2- Exemplo de layout de um processo na indústria de serviços, caso de uma loja. Fonte: Elaboração própria

Já na figura 3 mostra um diagrama esquemático de um exemplo de *layout* por processo numa empresa industrial, onde podemos observar que os tornos estão agrupados, assim como as fresas, as furadeiras e por último a linha de retificações.

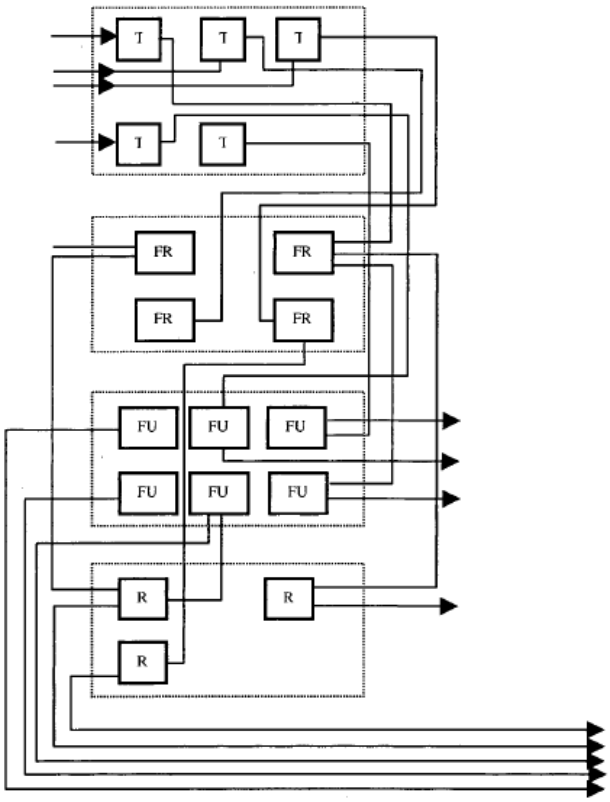


Figura 3- Process Layout manufacturing Fonte: Black, 1998

Neste tipo de *layout cada secção de trabalho é relativamente autónoma, e um determinado produto vai para qualquer secção de trabalho que seja necessária para a sua realização.* Este tipo de *layout* normalmente é usado em ambientes fabris que produzem pequenas séries de produtos, que por sua vez são produtos realizados por encomenda e que vão de encontro às exigências de cada cliente.

Assim sendo cada trabalhador não flui através do sistema de uma forma ordenada, mas sim o produto é que viaja de secção em secção, o que pode levar a um retrocesso e o movimento de departamento para departamento pode levar uma quantidade considerável de tempo, o que leva a que as filas tendam a aumentar. Além disso, cada novo produto pode exigir que uma operação seja configurada de forma diferente para cumprir os requisitos específicos do produto.

O espaço de armazenamento de um *layout* por processo é grande para acomodar a grande quantidade de inventário. A própria empresa pode ter um aspeto de um armazém, uma vez que existem centros de trabalho espalhados entre corredores de armazenamento. O processo de inventário é alto porque o material tem que se mover de secção de trabalho para secção de trabalho, em lotes. O inventário destes produtos, por outro lado, é baixo porque os bens estão a ser fabricados para um determinado produto específico, o que leva a que estes sejam logo enviados ao cliente e não armazenados.

Os *layouts* por processo em empresas de produção flexíveis exigem equipamento de manuseamento de material (como empilhadoras), que podem seguir vários caminhos, e que se movimentam em qualquer direção e transportam grandes quantidades de materiais de secção em secção. Uma empilhadora que movimente paletes de material a partir de uma secção de trabalho para outras secções de trabalho precisa de corredores largos para movimentar a carga.

Um método de controlo deste tipo de empilhadoras é tipicamente controlado por aparelhos de rádio e varia de dia para dia e hora a hora. As rotas são determinadas conforme as prioridades e diferentes cargas, cumprindo o prazo de entrega.

A grande preocupação do *layout* é onde localizar os departamentos ou secções de máquinas em relação uns aos outros. Embora cada trabalho tenha potencialmente um caminho diferente através das secções, alguns caminhos serão mais comuns do que outros. O histórico de informação sobre os pedidos dos clientes e as projeções de pedidos de clientes podem ser usadas para desenvolver padrões de fluxo através da empresa.

Tabela 1 - Vantagens e desvantagens do layout por processo Fonte: Monks, 1987

Vantagens	Desvantagens
➢ Flexibilidade na distribuição de equipamentos e operadores dentro do mesmo grupo; ➢ Produção nas estações de trabalhos independentes do fluxo, não penaliza a linha de montagem; ➢ Menor influência de paragens de máquinas no processo produtivo; ➢ Sistemas flexíveis de trabalho quanto aos prazos de entrega ao cliente; ➢ Equipamento de uso geral a preço mais reduzido; ➢ Supervisão especializada por processo de fabrico.	➢ Fluxos produtivos complexos, vários pontos de inversão e de acumulação; ➢ Elevado nível de produtos em curso de fabrico; ➢ Dificuldade na coordenação e planeamento da produção; ➢ Custos elevados com manuseio de materiais; ➢ Mão-de-obra especializada de alto custo; ➢ Elevado custo de supervisão; ➢ Longos lead time; ➢ Dificuldade na identificação das causas de defeitos; ➢ Dificuldade na implementação de melhorias devido ao fluxo irregular do processo

2.2.2. *Layout* por produto

O *layout* por produto, mais conhecido como *product layout*, organiza as atividades numa linha de acordo com a sequência de operações que precisam de ser executadas para montar um determinado produto específico. Cada produto tem a sua própria "linha" especificamente concebida para satisfazer as suas exigências.

O fluxo de trabalho é ordenado de forma eficiente, no qual se desloca a partir de uma secção de trabalho para outras secções da linha de montagem, até ao produto final sair no fim da “linha”.

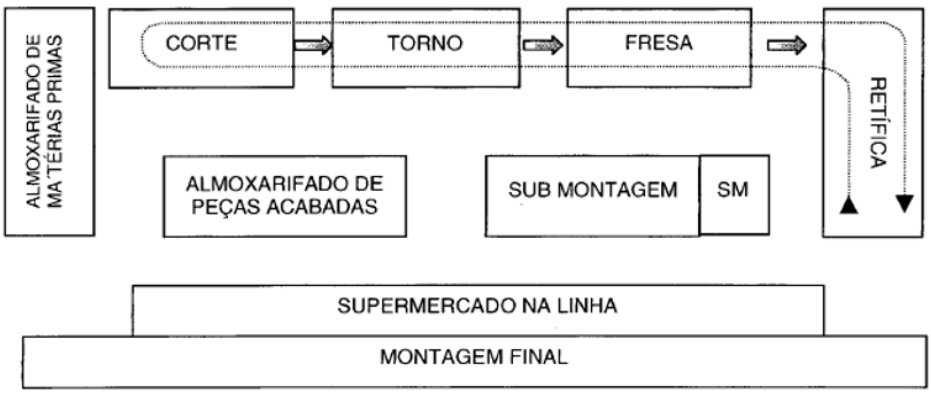


Figura 4 - Representação de Layout em Linha. Fonte: Moura, 1989

Uma vez que a “linha” é configurada para um tipo de produto ou serviço, as máquinas necessárias podem ser compradas para atender às necessidades

específicas de processamento do produto. O *layout* por produto é adequado para a produção em massa ou operações repetitivas em que a fabricação é estável e o volume é alto.

O produto ou serviço é um modelo feito para um mercado em geral, não para um cliente particular. Por causa do alto nível de exigência, o *layout* por produto é mais automatizado de que o *layout* por processo, bem como o papel do trabalhador é diferente. Cada trabalhador executa estritamente cada tarefa definida na "linha" de montagem, o que leva a que cada trabalhador não seja tão versátil como no *layout* por processo.

A grande preocupação em um *layout* por produto é equilibrar a linha de montagem, para que nenhum trabalhador na estação de trabalho se torne sobrecarregado e que o fluxo de trabalho se mantenha constante através da "linha".

O *layout* por produto necessita que o material se mova numa única direção ao longo da linha de montagem e sempre com o mesmo padrão. Os tapetes de rolamentos são os equipamentos de manuseamento de materiais mais comuns para este tipo de *layout*. Os mesmos podem ser passeados (automaticamente ajustado para controlar a velocidade de trabalho) ou num ritmo irregular (parado e iniciado pelos trabalhadores de acordo com seu ritmo). O trabalho de montagem pode ser realizado em linha (isto é, sobre os rolamentos) ou fora de linha (em uma estação de trabalho servida pelo tapetes).

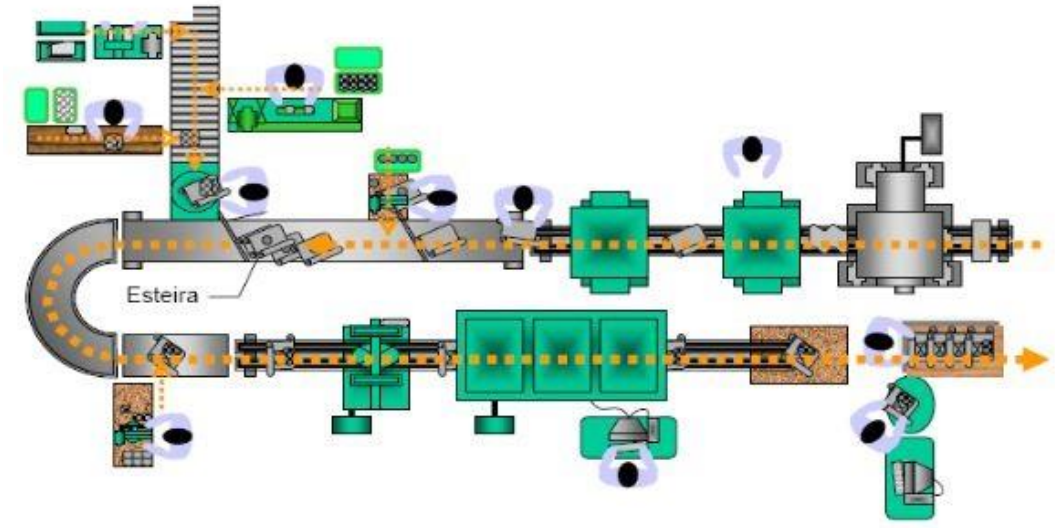


Figura 5 - Trabalho de montagem em linha Fonte: Doblas, 2010.

Os corredores são estreitos porque o material é movido apenas de uma maneira e o material não é movimentado para muito longe. O transportador é uma parte integrante do processo de montagem, geralmente com estações de trabalho em ambos os lados. O planeamento dos transportadores, uma vez que eles estão instalados, funciona

unicamente com uma variável, essa variável depende do quanto rápido os trabalhadores devem operar.

Tabela 2 - Vantagens e desvantagens de layout por produto Fonte: Tompkins et al. 1996 e Monks,1987

Vantagens	Desvantagens
➢ As sequências das operações resultam em linhas de fluxo planas e lógicas; ➢ A produção de uma máquina alimenta a próxima, tendo como resultado pequenos *stock* intermédios; ➢ O tempo de produção total é reduzido; ➢ A disposição de máquinas minimiza distâncias entre operações; ➢ Reduzido manuseio de material; ➢ O planeamento da produção é simples e os sistemas de controlo de qualidade são facilitados; ➢ Necessidade de mão-de-obra pouco especializada.	➢ O processo produtivo é inflexível; ➢ Equipamento específico com custos de aquisição elevado; ➢ Operações interdependentes; ➢ Paragem de uma máquina provoca a paragem da linha produtiva; ➢ Necessidade de supervisão constante; ➢ Elevado sincronismo entre tempos de produção de máquinas.

2.2.3. *Layout* posicional

O *layout* posicional, mais conhecido por *fixed position layout* é caracterizado por um número relativamente baixo de unidades de produção, em comparação com o *process* e *product layout,* e no qual o produto produzido é muito frágil, volumoso ou pesado para ser movimentado. Este tipo de *layout* é geralmente encontrado em empresas que fabricam produtos de grandes dimensões, como navios, casas, e aviões (Muther, 1978). Neste tipo de *layout*, o produto permanece imóvel durante todo o ciclo de produção.

Equipamentos, trabalhadores, materiais e outros recursos são trazidos para o local de produção. Assim, por exemplo, na construção de aviões, os rebites que são usados durante a construção seriam colocados perto da fuselagem e por sua vez, os reatores pesados, que só serão instalados no fim, devem ser colocados num local mais distante, para no fim, serem colocados de uma vez só.

A utilização do equipamento é baixa porque muitas vezes é menos dispendioso deixar os equipamentos de utilização ocasional em um local onde será necessário novamente em poucos dias, do que movê-lo para trás e para frente.

Frequentemente, o equipamento é alugado ou subcontratado, porque ele é usado por períodos limitados de tempo. Os trabalhadores chamados para o local de trabalho

são altamente qualificados na realização das tarefas especiais nos quais são solicitados a fazer. A figura que se segue demonstra como funciona o *layout.*

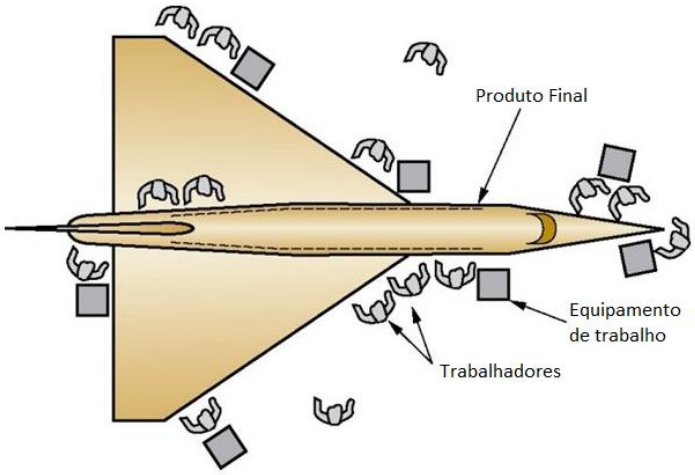


Figura 6- Fixed-Position Layout Fonte: Elaboração própria

Sendo assim o *layout* posicional tem como vantagem a economização de tempo e custos envolvidos no movimento de trabalhadores a partir de uma secção para outra. É flexível com a mudança no projeto e a sequência de operações podem ser facilmente incorporado. O *layout* é mais económico quando várias ordens em diferentes estágios de progresso estão a ser executadas em simultâneo e os ajustes podem ser feitos para atender a escassez de materiais ou ausência de trabalhadores, alterando a sequência de operações.

Mas como todos os métodos não são perfeitos, existem vantagens e desvantagens como podemos ver na seguinte tabela:

Tabela 3 - Vantagens e desvantagens do layout posicional. Fonte: Tompkins et al.,1996 e Slack, 2002

Vantagens	Desvantagens
➢ Fluxo de operações e responsabilidade pelos resultados; ➢ Flexibilidade no processo, rapidamente adaptável a mudanças do produto e volume de produção; ➢ Movimentação do material reduzido.	➢ Custo de movimentação de operadores e materiais elevado; ➢ Necessidade de áreas de implantação com dimensões elevadas; ➢ Requer investimento em duplicação de equipamentos; ➢ Especialização de mão-de-obra elevada; ➢ Necessidade de supervisão constante; ➢ Requer elevado planeamento e produção sincronizada.

2.2.4. *Layout* em grupo ou por células

Fundamentalmente, o que se pretende obter com o *layout* em grupo é dividir um sistema de manufatura em subsistemas, com o objetivo de maximizar a produção de uma grande variedade de produtos fabricados em pequenos lotes. A estrutura física do arranjo fabril visualizado na figura 7 caracteriza uma típica empresa que adota o sistema de produção por *layout* por processo, onde os departamentos são divididos de acordo com o tipo de processo executado em cada fase da elaboração dos produtos.

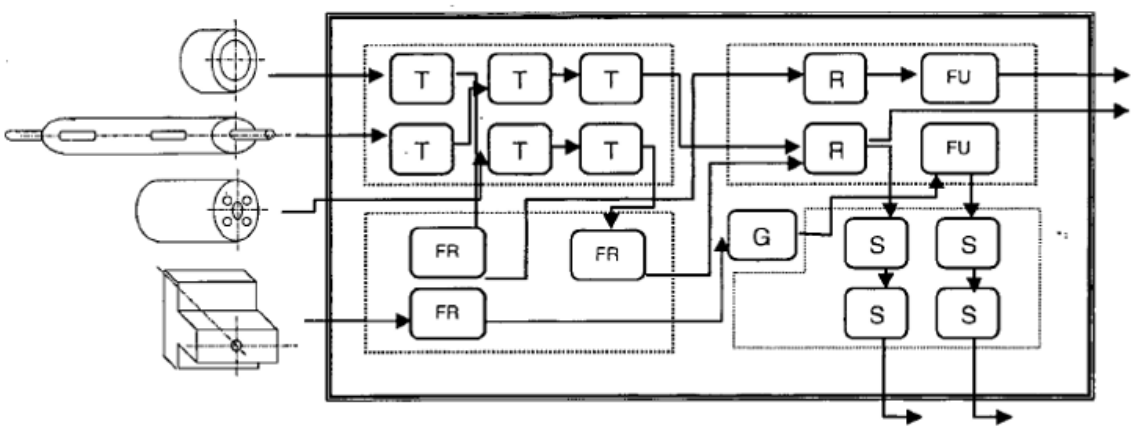


Figura 7 - Layout organizado por processo. Fonte: Montevechi, 1989

Agrupando-se peças similares em famílias é possível formar grupos de máquinas capazes de processá-las. Estes grupos de máquinas são denominados “sectores ou células de produção”, as quais são indicadas na figura 8 pelas duas áreas isoladas. Nestas áreas, um determinado conjunto de máquinas é responsável pela execução das peças indicadas na figura em cima. Este *layout* em grupo visa melhorar a produtividade em indústrias de médios e pequenos lotes. Tenta-se, assim, aliar a produtividade dos sistemas em linha com sistemas de *layout* por processo.

Depois de definidos os fluxos de produção para os principais produtos da empresa, podemos assim elaborar o *layout* final para o setor produtivo. Assim sendo, é elaborado um *layout* tendo em vista sempre aspetos operacionais que possam ser postos em prática ao longo da implantação, como viabilidade de movimentação de máquinas, minimização de alterações na construção e mínimo investimento em recursos para adaptação.

Este tipo de *layout* é uma opção bastante atrativa relativamente aos meios convencionais de produção, devido principalmente à sua flexibilidade na adaptação às exigências do mercado, à qualidade dos bens produzidos e, naturalmente por ser economicamente atrativo devido à sua melhor produtividade em relação ao sistema com *layout* por processo.

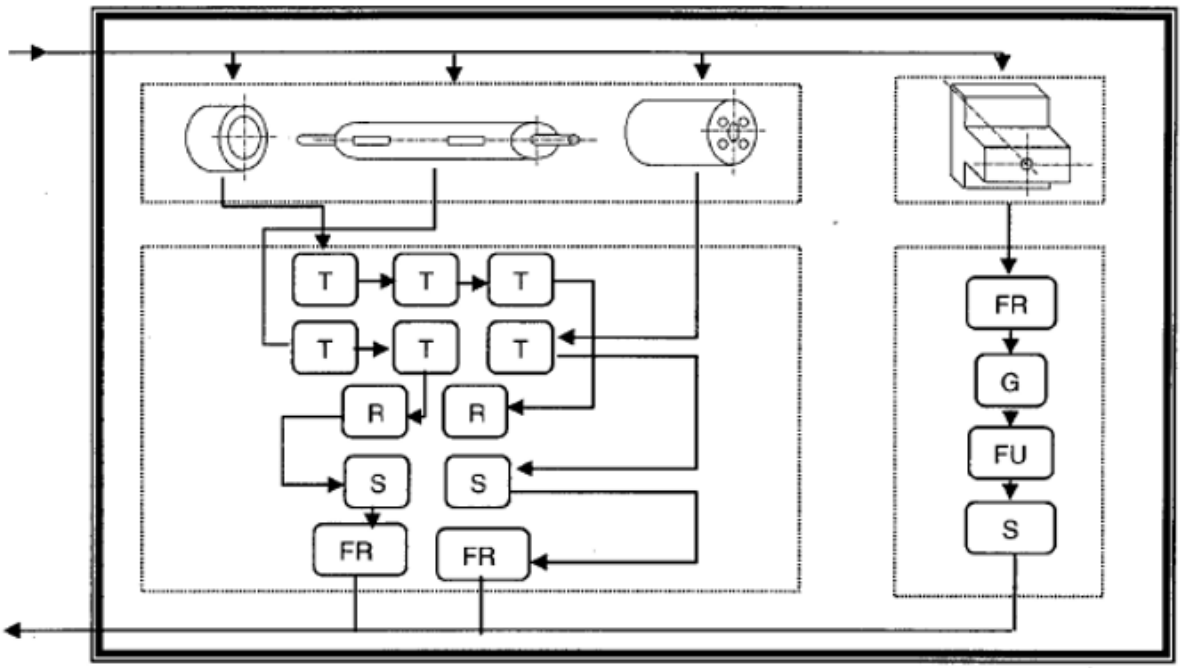


Figura 8 - Conversão de um sistema tradicional em layout por grupo utilizando o conceito da Tecnologia de Grupo. Fonte: Montevechi, 1989

Com as células de produção, conseguimos aliar a flexibilidade à produtividade (Montevechi, 1989) tal como conseguimos visualizar no esquema da Figura 9:

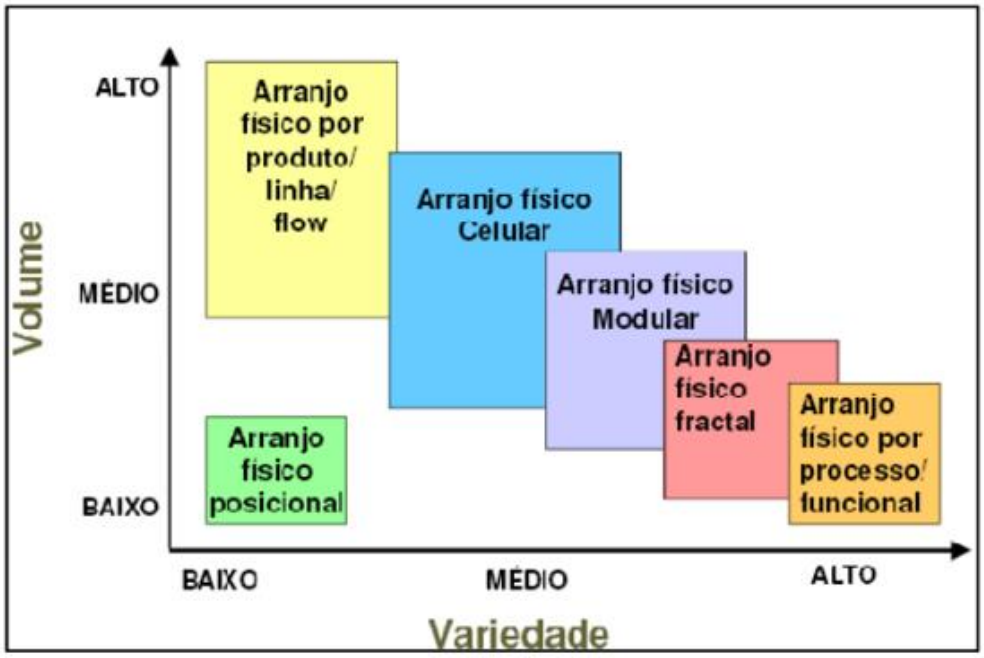


Figura 9 - Tipos de Layout (Volume e Variedade). Fonte: Montevechi, 1989

De acordo com Montevechi (1989), a adoção deste tipo de *layout* pode originar três tipos de células:

1. Máquinas isoladas;
2. Grupos de máquinas;
3. Grupos de máquinas obedecendo “Flow Shop”.

A célula de máquinas isoladas é definido quando uma estação de trabalho onde determinada máquina ou operador executa uma atividade exclusiva para uma determinada peça, ou por qualquer tipo de condições adversas, que a impeça de ser colocada junto com outras máquinas em um sector fabril. Isto pode acontecer, por exemplo, numa estação de tratamento térmico de peças metálicas, onde os equipamentos de grandes dimensões delimitam um espaço específico para os mesmos.

A célula tipo grupo de máquinas é aquela em que algumas máquinas com funções semelhantes são agrupadas em “famílias” de máquinas ou sectores, mas não há previsão da sequência de movimentação das peças dentro do sector. Isto é, as peças podem não ter um fluxo direcionado no *layout* de máquinas que formam o sector. Podem ser processadas numa máquina posterior e voltar para uma máquina anterior.

A célula de produção obedecendo a um “Flow Shop” é a reunião de um grupo de máquinas destinadas à produção onde as peças passam pelas máquinas obedecendo a uma sequência. Certamente algumas operações podem ser omissas, mas o fluxo de trabalho precisa de obedecer sempre à mesma direção.

A adoção do *layout* em grupo tem as seguintes vantagens e desvantagens como podemos ver na Tabela 4.

Tabela 4 - Vantagens e desvantagens de layout em grupo. Fonte: Tompkins et al. 1996

Vantagens	Desvantagens
➢ Criação de grupos multifuncionais e visão de produto; ➢ Elevada taxa de ocupação de equipamentos; ➢ Controlo do sistema e confiabilidade dos prazos de entrega; ➢ Flexibilidade no processo; ➢ Baixo nível de *stocks*; ➢ Fluidez do processo produtivo; ➢ Controlo de custos; ➢ Melhoria na qualidade.	➢ Elevada supervisão do processo produtivo; ➢ Nível elevado de mão-de-obra especializada; ➢ Fluxo produtivo total, condicionando pela dependência das células individuais de trabalho; ➢ Reduzida possibilidade de utilizar os equipamentos para rápidas produções especiais; ➢ Requer elevada supervisão.

2.3. Formas e dimensões de *layout*

Existem diferentes formas de instalações que podem condicionar o tipo de *layout* a abordar, conforme é apresentado na figura 10. Assim sendo *podemos* salientar as formar regulares, geralmente apresentadas como retangulares (Kim & Kim, 2000) e as formas irregulares, no qual geralmente são representadas como polígonos que incluam pelo menos um ângulo interno de 270º (Lee & Kim, 2000).

Segundo Chwif, Pereira e Moscato (1998), uma instalação pode ter várias dimensões regulares e irregulares, definidas por um comprimento e uma largura fixa. Neste caso, as instalações são denominadas por blocos fixos ou rígidos (Meller et al., 1999).

Figura 10 - Forma regular e irregular de instalações. Fonte: Drira, 2007

No planeamento de um *layout*, é comum a área da empresa ser dividida em blocos retangulares com a mesma área e forma, onde a cada bloco é atribuído uma estação de trabalho (Fruggiero, Lambiase e Negri, 2006). No entanto, Wang, Hu e Ku (2005) referem que, da mesma forma que existem blocos de estações de trabalho com áreas diferentes, também as instalações com áreas irregulares devem ser planeadas atendendo a esse facto, de maneira a rentabilizar a área disponível para implantação.

2.4. Sistema de manuseamento

Segundo El-Baz (2004), em Francis e White (1974), afirma-se que 20% a 50% dos custos totais das despesas de produção são atribuídos à movimentação de materiais, pelo que o planeamento de *layout,* atendendo à circulação de materiais, poderá reduzir entre 10% a 30% dos custos totais de produção. Assim sendo, existem diferentes sistemas de manuseio de material, que podem ser ordenados em *layouts* de uma linha única, em múltiplas linhas, ou circuito e em campo aberto, no qual podemos ver na figura 11.

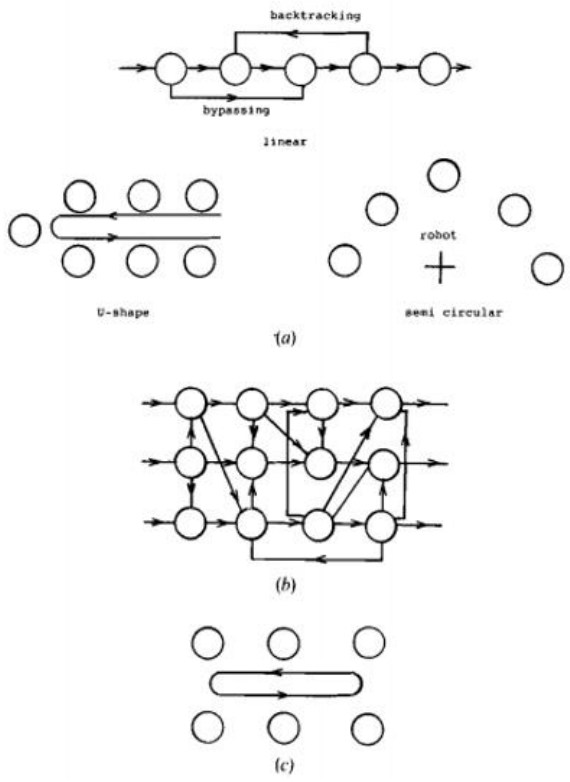


Figura 11 - Tipos de layout de máquinas. (a) Single row; (b) multi rows; (c) loop. Fonte: Hassan, 1991

O *layout* linha única (*single row*) pode assumir várias formas, tais como, linear, semicircular, ou em forma de U. As máquinas neste tipo de *layout* são organizadas o mais próximo possível umas das outras de forma a criar uma sequência de operações, no qual as peças são processadas, a fim de obter os benefícios de uma sequência em linha única.

Esses benefícios incluem custo pequeno de manuseio de materiais e tempo, fluxo unidirecional, menos atrasos, melhor controle das operações e a capacidade de usar transportadores. Um *layout* de linha única pode ser usado dentro das células GT (*group technology*), em instalações que implementam JIT (*just-in-time*), e às vezes com FMS (*flexible manufacturing systems*).

O *layout* de múltiplas linhas é geralmente em linha reta com as máquinas em série, interagindo umas com as outras assim como com as máquinas nas outras linhas. Quando o número de linhas do *layout* é igual ou superior a dois, mas não há nenhuma troca de material entre elas, o *layout* não é um layout de múltiplas linhas.

Em algumas empresas é possível ver dois *layout* de linha única colocados um ao lado do outro, que por alguma razão têm equipamentos de manuseamento de material em comum, mas que não é considerado *layout* de múltiplas linhas. Este tipo de *layout* é adequado para empresas de FMS.

O *layout* tipo circuito é também o *layout* usado em empresas de FMS. As máquinas são organizadas em torno de um caminho oval e o movimento das peças é geralmente

unidirecional. A principal vantagem da disposição em ciclo é o manuseamento de materiais que providencia a flexibilidade. Afentakis (1989) e Kouvelis e Kim (1992) apresentaram uma descrição detalhada deste tipo de disposição.

O *layout* em campo aberto caracteriza-se pela flexibilidade do fluxo produtivo, com a qual os materiais podem seguir com maior facilidade vários caminhos de uma estação de trabalho para outra. Este tipo de disposição corresponde aos casos em que as instalações podem ser colocadas sem as restrições ou limitações que estão associadas ao *layout* de linha única ou l*ayout* em circuito (Yang et al., 2005).

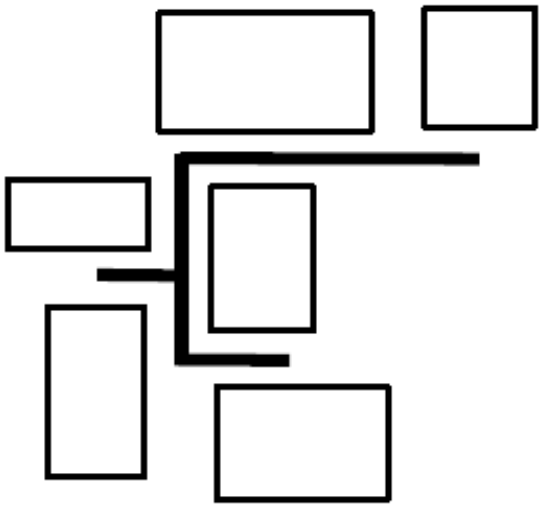


Figura 12- Open Field Layout Fonte: Yang et al, 2005

2.5.*Layout* em instalações por andares

Essencialmente em áreas urbanas ou parques industriais desenvolvidos, encontra-se de uma forma geral uma escassez de oferta de terrenos, o que provoca a subida de preços praticados. Assim sendo, a limitação do espaço horizontal disponível nas empresas cria a necessidade de utilizar um plano vertical da instalação, induzindo a necessidade de planear um *layout* por andares, conforme podemos ver na figura 13.

Esta organização permite que o fluxo na instalação possa deslocar-se não só horizontalmente, assim como de um andar para outro, localizados em diferentes níveis. A movimentação vertical dos materiais requer um dispositivo de transporte vertical, que por norma é um elevador. Em tais situações, tanto na posição horizontal, como na posição vertical, os níveis de *layout* têm que ser determinados para cada instalação, de modo que os problemas relacionados sejam tratados como problemas de *layout* por andares (Kochhar e Heragu, 1998).

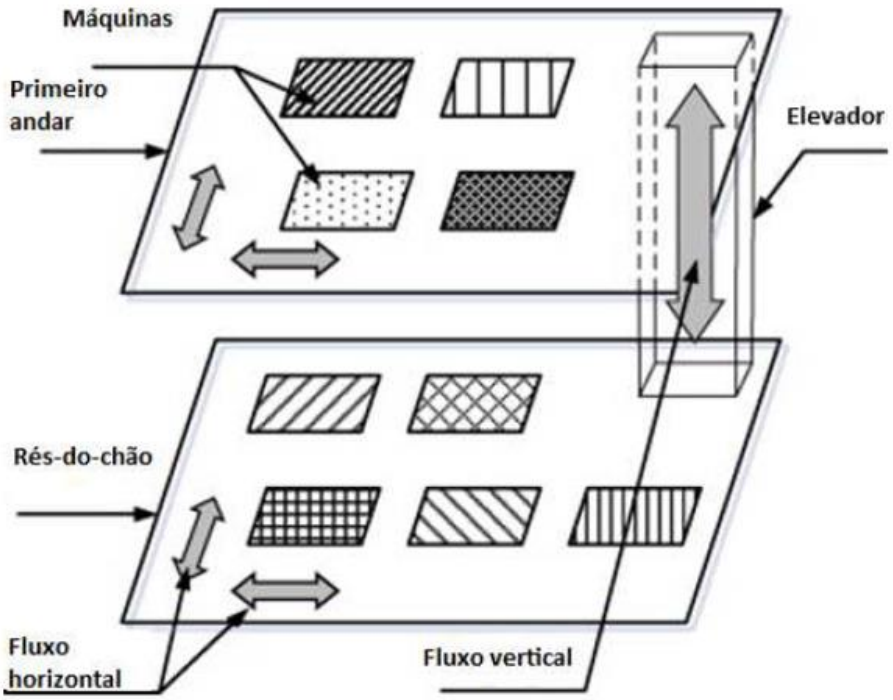


Figura 13 - Representação de layout por andares. Fonte: Drira, 2007

Johnson (1982) apresentou o problema de posições relativas das instalações em múltiplos andares. Como já referido, os elevadores são o sistema de manuseamento de material mais adotado no caso de instalações com múltiplos andares (Lee, Roh e Jeong, 2005). Segundo Matsuzaki et al. (1999), a capacidade de cada elevador pode ser uma limitação no processo da empresa.

Desta forma, Patsiatzis e Papageorgiou, (2002), Lee et al., (2005) referem a necessidade de determinar a capacidade e o número de elevadores necessários para o processo, atendendo à área de cada piso, bem como ao fluxo de transporte estimado entre cada piso.

2.6. Retroceder e ultrapassar

Em processos produtivos, retroceder (*backtracking*) e ultrapassar (*bypassing*), são dois movimentos específicos que podem ocorrer no fluxo da linha de produção.

O processo de retroceder caracteriza-se pela necessidade de determinado produto inverter o sentido do fluxo produtivo, regressando a um processo anterior para terminar uma tarefa produtiva (Braglia, 1996; Kouvelis e Chiang, 1992; Zhou, 1998).

Zhou (1998) acrescenta que estes tipos de deslocações devem ser minimizados, visto que podem vir a introduzir atrasos na produção, com deslocações que não traduzem um valor acrescentado ao produto final. Este problema é denominado de (PLFP) *Production line formation problem*, e consiste em determinar a disposição das máquinas (parcial ou total), de modo a minimizar a soma ponderada das deslocações

cuja direção é contrária ao fluxo produtivo comum, tendo em conta os constrangimentos sobre os postos das máquinas.

Foram desenvolvidas algumas formulações discretas por Braglia, (1996); Kouvelis e Chiang, (1992), de forma a minimizar o trajeto contrário ao fluxo produtivo.

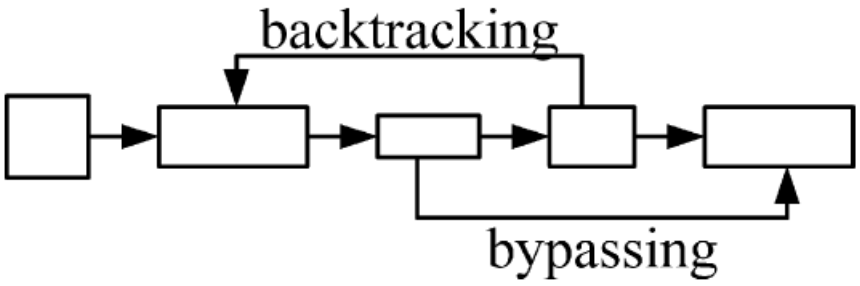


Figura 14 - Backtracking e Bypassing. Fonte: Drira, 2007

O movimento intitulado ultrapassar ocorre quando, no processo produtivo, determinado produto "salta" um ou mais processos durante o seu movimento no sentido do fluxo da linha produtiva, em virtude de não ser necessário utilizar determinada máquina na transformação do produto final (Chen et al., 2001).

2.7. Locais de pick-up e drop-off

Outra das condicionantes ao planeamento de um *layout,* é conhecer o local de entrada e saída de materiais numa determinada secção de trabalho, mais conhecidos por pontos de *pick-up* e *drop-off*. Este conceito é considerado como a distância percorrida pelo material desde a saída de uma máquina até à entrada na máquina seguinte, dentro do processo produtivo. Na tentativa de minimizar essa distância percorrida entre máquinas, é necessário considerar a área de recursos necessários para operar essa estação, denominada por área envolvente de um posto de trabalho (Lacksonen, 1997).

Embora estes pontos possam ser localizados em vários lugares (Kim e Kim, 2000), vários investigadores restringiram a suas posições possíveis para reduzir a sua complexidade (Das, 1993; Rajasekharan, Peters e Yang, 1998; Welgama e Gibson, 1993). Na figura 15 é apresentado um modelo de pontos de pick-up e drop-off de uma máquina com forma regular.

Figura 15 - Pontos de pick-up e drop-off de uma máquina com forma regular. Fonte: Drira, 2007

Assim conforme foi dito os pontos de pick-up e drop-off podem gerar algumas restrições na formulação do problema de *layout* (Kim e Kim, 2000; Welgama e Gibson, 1993; Yang et al., 2005).

2.8. Flexibilidade de *Layout*

Os sistemas flexíveis de produção (FMS), desempenham um papel fundamental nas complexas e modernas linhas de produção. Estes sistemas são geralmente constituídos por um grupo de máquinas, capazes de realizar um número de diferentes operações, interligadas por meio de um mecanismo automático, no qual o material é transportado e manuseado entre elas.

Os sistemas de produção flexíveis são muitas vezes utilizados em ambientes de empresas dinâmicas, capazes de se adaptarem a incertezas do mercado, tal como a variação no volume de encomendas.

Um sistema de *layout* flexível permite responder de forma eficiente a requisitos dinâmicos e incertos, o que se torna importante para conseguir um sistema adaptado, capaz de tornar proveitosa a relação custo-benefício. Além disso, um sistema de produção flexível é geralmente equipado com máquinas, ferramentas e sistemas de manuseio de materiais automatizados, que têm um elevado custo de investimento.

Este tipo de *layout* exige que envolva a programação de um projeto no qual o seu planeamento consiga ter a possibilidade de redistribuição, caso seja necessário. Os *layouts* desejados apresentam flexibilidade de duas maneiras: através da robustez para mudanças nos requisitos de produção e através da capacidade de adaptação do *layout* para estas novas exigências.

2.8.1. *Layout* Robusto

Um *layout* robusto é aquele que consegue responder eficazmente a um grande número de cenários de produção, onde a lógica de conceção favorece um *layout* satisfatório a fim da procura incessante por um *layout* ótimo. Neste tipo de *layout* a sua implementação tende a escolher a opção que parece ficar mais frequentemente perto da solução ótima, quando a solução ótima não é atingível, (Kouvelis et al., 1992).

Os autores Braglia et al., (2003) referem que um *layout* robusto é obtido ao se escolher por uma alternativa que se comporte eficazmente bem na maioria das vezes, mesmo que ocorram variações no volume e tipo de produtos. Segundo Benjaafar e Sheikhzadeh, (2000); Lahmar e Benjaafar, (2005), ter um *layout* robusto é ter um layout flexível.

Shore e Tompkins, (1980) e Balakrishnan e Cheng (2003) utilizaram o critério de custo para a penalização de cada tipo de *layouts* robustos. Segundo os autores, dado

um conjunto de cenários e a sua probabilidade de ocorrência, podem ser calculados os custos de penalização para cada tipo de *layou*t. Desta forma é considerado mais robusto o *layout* com menor número esperado de penalizações para o processo produtivo.

Por sua vez, Rosenblatt e Lee (1968) estabelecem como critério para adotar um *layout* robusto, o ambiente de incerteza no qual o valor exato da probabilidade de ocorrência de diferentes cenários é desconhecido. Segundo os autores, mediante o cenário de incerteza é melhor escolher uma alternativa de *layout* que se comporte bem em todos os cenários, em detrimento de uma alternativa ótima para um cenário (que pode eventualmente não ocorrer) e se revele mau, nos cenários possíveis de ocorrência.

2.8.2. *Layout* Dinâmico

O *layout* dinâmico é aquele que apresenta um bom desempenho ao longo do tempo, conseguindo promover alterações, adequando-se eficazmente a oscilações no fluxo de materiais no qual pode mudar ao longo de vários períodos.

No *layou*t dinâmico, têm-se em conta as previsões a médio/longo prazo tentando alcançar o *layout* mais eficaz em cada período, prevendo a frequência com que este deve mudar. Assim, é realizada uma avaliação de custo-benefício em múltiplos períodos, entre o aumento do fluxo de materiais no processo produtivo e os custos do aplicar um *re-layout*.

O horizonte do planeamento é geralmente dividido em períodos que podem ser definidos em semanas, meses ou anos, sendo que para cada período, o fluxo de dados estimados deverá permanecer constante. A figura 16 apresenta uma instalação com seis departamentos de tamanho igual a serem dispostos em cada um dos quatro períodos no futuro de planeamento, com o objetivo de determinar o melhor *layout* para cada período nesse futuro de planeamento.

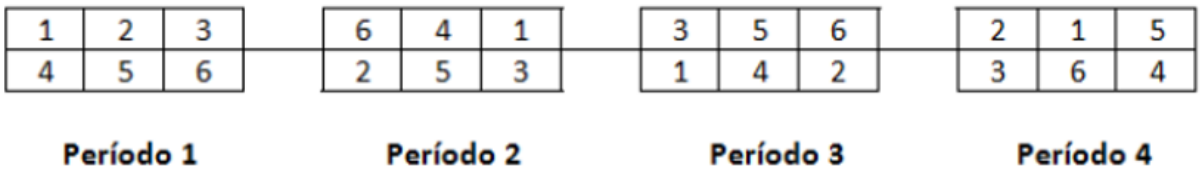


Figura 16 - Evolução de layout em quatro períodos. Fonte: Drira, 2007

Para tal, deve ser feita a ponderação entre a soma de custos de manuseamento de material, com a soma dos custos de *re-layout* entre os vários períodos de tempo, por forma a determinar qual o melhor layout para um determinado cenário (Balakrishnan, Cheng, Conway et al., 2003; Baykasoglu, Dereli e Sabuncu, 2006).

A análise de *layouts* dinâmicos utiliza uma correspondência “de-para” e um vetor de custos associados à mudança nos diferentes períodos, (Rosenblatt, 1986; Montreuil e Venkatadri, 1991; Lacksonen, 1997; Balakrishnan e Cheng, 1998).

Capítulo 3 – Formulação de Modelos Matemáticos

3. Formulação de Modelos Matemáticos

Existem várias maneiras de formular matematicamente os *"Facility layout problem"* de modo a que eles possam ser solucionados. Os modelos permitem que as relações complexas entre os diferentes elementos que intervêm na disposição um problema possam ser evidentes. Tais modelos podem basear-se em princípios diferentes, que incluem teoria de grafos (Kim & Kim, 1995; Leung, 1992; Proth, 1992) ou uma rede neuronal (Tsuchiya, Bharitkar, & Takefuji, 1996).

Estes modelos são geralmente usados para sugerir soluções para os problemas de projeto de *layout* de instalações, que na maioria das vezes são considerados como problemas de otimização, com uma única função objetivo. No entanto, em primeiro lugar temos que efetuar o enquadramento deste problema em dois tipos de sub-problemas:

1. Problemas de atribuição de espaço de uma dimensão, em que as instalações são dispostas ao longo de uma linha;
2. Problemas de atribuição de espaço de duas dimensões. Nesta situação as instalações estão organizadas ao longo de duas ou mais linhas.

Muitos dos planeamentos são considerados que as instalações são retangulares e que as suas formas são conhecidas logo à partida (é conhecida a relação entre o seu comprimento e a largura), e não existem nenhumas restrições à topologia do edifício onde vão ser posicionadas as instalações.

Embora a hipótese de que as instalações são de forma retangular não seja propriamente uma consideração realista, constata-se que as instalações que não são retangulares podem ser aproximadas a um retângulo (Heragu, 1997). A Figura 17 mostra como esta aproximação pode ser realizada. No entanto, se a forma das instalações difere bastante de um retângulo, então a qualidade da solução apresentará alguns senãos, embora se deva referir que esta situação é pouco frequente. Esta aproximação permite simplificar bastante o problema bem como a sua resolução.

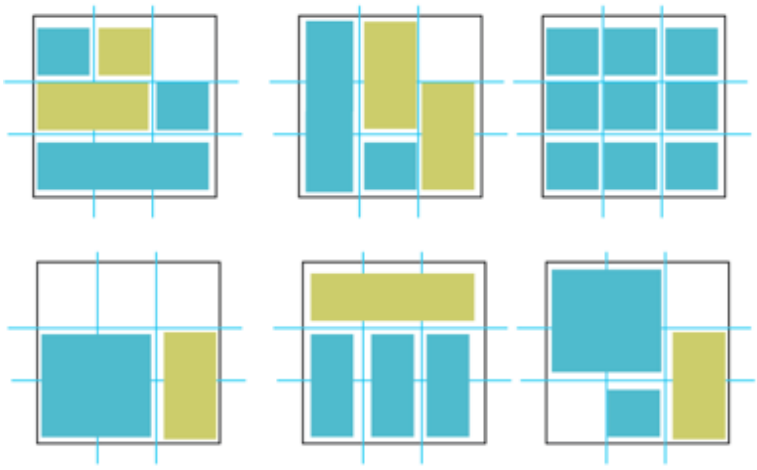


Figura 17 – Representação de instalações de que não são de forma retangular. Fonte: Elaboração própria

3.1. Problemas de atribuição de espaço de uma dimensão

Na formulação de problemas de *layout* em que temos a atribuição de espaços de uma dimensão, tem-se em consideração que as instalações estão dispostas ao longo de uma linha reta e no qual também as suas orientações são conhecidas logo à partida (Heragu e Kusiak, 1987).

Neste tipo de cenário, para efeitos de formulação, a orientação indica quais dos lados, se a largura ou se o comprimento, fica paralelo com a reta. Em muitos *layouts* de produção, os pontos de carga e descarga das instalações devem ficar alinhadas com os pontos de cargas e descargas dos equipamentos de manuseamento e dos transportadores. Uma vez que estes pontos são fixos, a orientação das máquinas é conhecida, logo à partida, para a maioria dos casos.

Assim sendo para este tipo de problemas temos a seguinte formulação, (Heragu e Kusiak, 1987).

$n = Total\ de\ Instalações;$

$f_{ij} = Fluxo\ de\ viagens\ do\ equipamento\ i\ para\ equipamento\ j;$

$c_{ij} = Custo\ de\ transporte\ de\ viagens\ do\ equipamento\ i\ para\ equipamento\ j;$

$l_i = Comprimento\ da\ instalação\ i;$

$l_j = Comprimento\ da\ instalação\ j;$

$d_{ij} = Distância\ de\ folga\ entre\ as\ instalações\ i\ e\ j;$

$x_i = Distância\ entre\ o\ centro\ da\ instalação\ i\ e\ o\ referencial\ utilizado;$

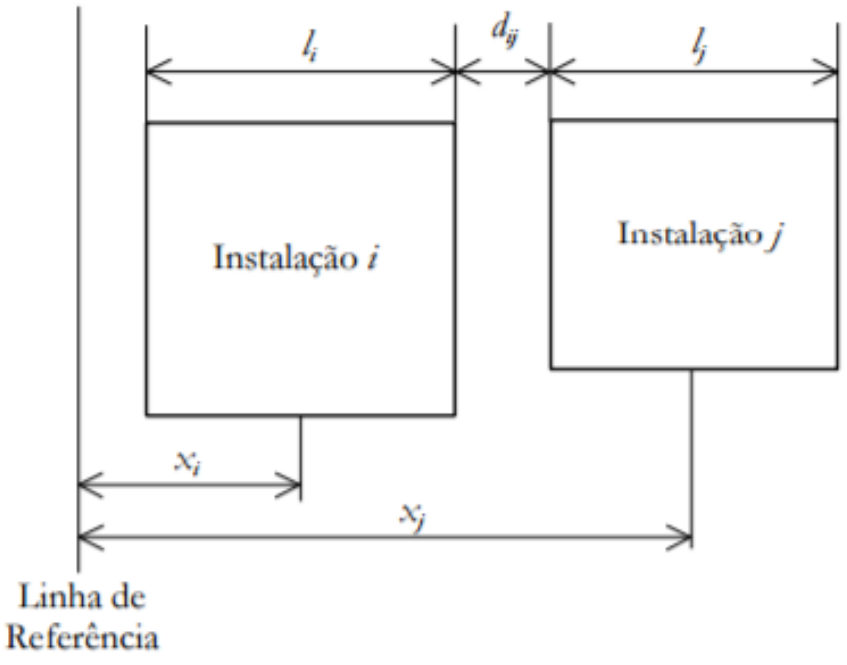


Figura 18 - Representação de instalações de forma diferente. Fonte: Tavares, 2000

Formulação da função objetivo:

$$Min \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} c_{ij} f_{ij} \left| x_i - x_j \right|$$

Formulação das restrições:

$$\left| x_i - x_j \right| \geq \frac{1}{2} \left(l_i + l_j \right) + d_{ij}, \ \forall i, j; i < j$$

$$x_i \geq 0 \, , \forall i$$

$$H - \frac{1}{2} \, l_i \geq x_i \geq \frac{1}{2} \, l_i, \forall i$$

Nesta formulação mostramos como se relacionam os parâmetros e as variáveis de decisão deste tipo de problemas. A função objetivo minimiza os custos totais relacionados com o fluxo de produtos entre as instalações. A primeira restrição certifica que as instalações não se sobreponham. Na segunda restrição podemos encontrar um requisito que embora não seja fundamental ele impõe que os valores das variáveis de decisão sejam sempre positivos. Já a terceira restrição é facultativa uma vez que só serve caso nós conheçamos as dimensões do edifício, em que H corresponde ao comprimento do edifício. Contudo como a função objetivo minimiza a distância entre instalações, implicitamente o *layout* obtido vai satisfazer as restrições pretendidas.

3.2. Problemas de atribuição de espaço de duas dimensões

Têm sido propostos vários tipos de modelos, para a realização do problema de *layout* com espaço a duas dimensões. Uma das características que distingue esses modelos está diretamente relacionada com a área das instalações. Alguns desses modelos consideram apenas instalações de áreas iguais, enquanto, que outros modelos tratam também instalações de áreas diferentes. Com isso, alguns destes modelos são modelos de Programação Inteira Mista (MIP), enquanto outros envolverão termos não lineares (Koopmans e Beckman, 1957; Heragu e Kusiak, 1990; Montreuil et al, 1993).

3.2.1. Problemas de Instalações de Áreas Iguais

Este é um problema clássico em que o objetivo a atingir tem a ver com a modelação do processo de localização de instalações inter-atuantes, com áreas iguais, tendo sido abordado pela primeira vez por Koopmans e Beckman (1957). Este é um típico problema do tipo MIP, que considera a existência de n instalações a serem atribuídas a n locais predefinidos. O termo atribuição aqui usado significa que se faz corresponder cada instalação a um local específico e vice-versa.

A formulação deste tipo de *Quadratic Assignment Problem* (QAP) requer que o número de instalações seja igual ao número de locais. Se existirem n locais e m instalações, sendo $n > m$, então devem-se considerar $n - m$ instalações fictícias, e considerar que os fluxos entre si bem como as demais instalações do problema é nulo. Por outro lado, se $n < m$ então o problema não tem solução.

O local da fábrica é dividido em blocos retangulares com a mesma área e forma, e cada bloco é atribuído para uma secção (Fruggiero, Lambiase, & Negri, 2006). As secções que têm áreas desiguais, podem ocupar diferentes blocos (Wang, Hu, e Ku, 2005).

Figura 19- Representação de layout bloco. Fonte: Drira, 2007

A resolução do QAP tem em consideração o fluxo de materiais entre pares de instalações, e a distância entre si. Normalmente, esta distância é medida entre o centro de cada par de instalações, utilizando-se uma métrica retilínea. Outro facto a ter em conta é o custo do transporte por unidade de distância e por peça.

Neste caso, a formulação típica deste tipo de *layout* consiste em determinar as posições relativas das localizações dos sectores, de forma a minimizar os custos de manuseamento de materiais e é representada da seguinte maneira (Drira 2007):

Formulação típica deste tipo de *layout:*

$n = Total\ de\ Equipamentos\ e\ Locais$;

$f_{ik} = Custo\ de\ fluxo\ do\ equipamento\ i\ para\ equipamento\ k$;

$d_{jl} = Distância\ do\ local\ j\ para\ o\ local\ l$;

$x_{ij} = Equipamento\ i\ em\ local\ j$

$x_{kl} = Equipamento\ k\ em\ local\ l$

Formulação da função objetivo:

$$Min \sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{k=1}^{n} \sum_{l=1}^{n} f_{ik} d_{jl} x_{ij} x_{kl}$$

Formulação das restrições:

$$\sum_{j}^{n} x_{ij} = 1 \qquad \forall i$$

$$\sum_{i}^{n} x_{ij} = 1 \qquad \forall j$$

$$x_{ij} = 0 \text{ ou } 1 \; \forall i, j$$

A função objetivo representa a soma dos custos de fluxo de cada par de instalações no qual tem que ser minimizada. Na primeira restrição garantimos que cada equipamento é colocado em apenas um local e na segunda restrição garantimos que cada local contém apenas um equipamento.

Por sua vez são sugeridas outros tipos de formulações por Kouvelis e Chiang, (1992) e Braglia (1996) de modo a minimizar o recuar nos *layouts* de (single row) linha única. O mesmo tipo de abordagem é também utilizada por (Afentakis, 1989), para criar um *layout* de círculo, de modo a minimizar o congestionamento do tráfego, isto é, o número de vezes que uma peça atravessa o circuito antes de todas as suas operações serem completadas.

Este tipo de representações de *layout* é comum no uso para problemas de *layout* dinâmico. Os problemas abordados estão relacionados com instalações de igual tamanho (Baykasoglu & Gindy, 2001; Lacksonen & Enscore, 1993) e devem respeitar as restrições que garantam que cada local é atribuído uma única instalação em cada fase, e que cada instalação é atribuída unicamente a cada localização (Baykasoglu & Gindy, 2001; McKendall, Shang, e Kuppusamy, 2006).

3.2.2. Problemas de Instalações de Áreas Diferentes

Neste tipo de *layout* é tratado em muitos problemas de *layout* reais como um *Mixed Integer Programming Problem* (Das, 1993). Todas as instalações são colocadas em qualquer lugar dentro do espaço e não devem sobrepor-se umas às outras (Das, 1993; Dunker et ai, 2005; Meller et al., 1999) como podemos ver na figura a seguir.

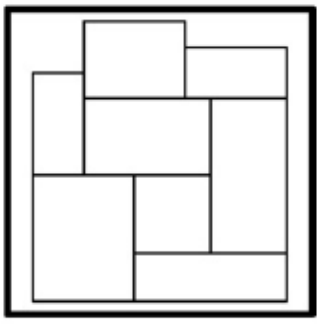


Figura 20 – Representação de layout contínuo. Fonte: Drira, 2007

As instalações são localizadas no local da fábrica pelas coordenadas do seu centróide (x_i, y_i). O centróide corresponde a metade do comprimento "L_i" e a metade da largura "W_i". Em alternativa, também podem ser representadas pelas coordenadas do canto inferior esquerdo, comprimento "L_i" e largura "W_i" da instalação. A distância entre duas instalações pode ser, por exemplo, expressa através da forma retilínea (Chow et al., 1998):

$$d_{ij}\left((x_i, y_i), (x_j, y_j)\right) = |x_i - x_j| + |y_i - y_j|$$

Os pontos de *pick-up* e *drop-off*, podem gerar limitações na formulação do problema de *layout* (Kim & Kim, 2000; Welgama & Gibson, 1993; Yang et al., 2005). Neste caso, a distância percorrida desde o *drop-off* da instalação "i" para o *pick-up* da instalação "j", pode, por exemplo, ser dada pela seguinte equação (Kim & Kim, 2000).

$$d_{ij} = |x_i^o - x_j^i| + |y_i^o - y_j^i|$$

na qual (x_i^o, y_i^o) designa as coordenadas do ponto de *drop-off* da instalação "i", e (x_j^i, y_j^i) as coordenadas do ponto de pick-up da instalação "j".

Como é de esperar, existem limitações relativamente à área da planta da empresa, o que requer que a superfície total disponível tenha que ser superior ou igual à soma de todas as áreas a serem ocupadas pelas instalações. A área alocada a cada instalação na planta deve também ter em conta o espaço de outros recursos (distâncias de folga) ou *buffers*, que são necessários para operar a máquina (Lacksonen, 1997).

Outra restrição que é muito importante é que as instalações não se devem sobrepor (Welgama e Gibson 1993). Assim sendo, estabeleceu-se duas condições para que não exista sobreposição das instalações em ordem a "x" e a ordem a "y":

A Condição de que a coordenada "x" do equipamento não se sobreponha é:

$$(x_{jt} - x_{ib})(x_{jb} - x_{it}) \geq 0$$

Onde (x_{it}, y_{it}) e (x_{ib}, y_{ib}) são as coordenadas do canto superior esquerdo e inferior direito da instalação "i" e (x_{jt}, y_{jt}) e (x_{jb}, y_{jb}) o canto superior esquerdo e o canto inferior direito da instalação "j".

Na imagem seguintes conseguimos visualizar a sobreposição dos equipamentos em ordem à coordenada "x" em que, $x_{jt} < x_{ib}$ e que $x_{jb} > x_{it}$, o que revela que o resultado vai ser < 0 logo que existe sobreposição.

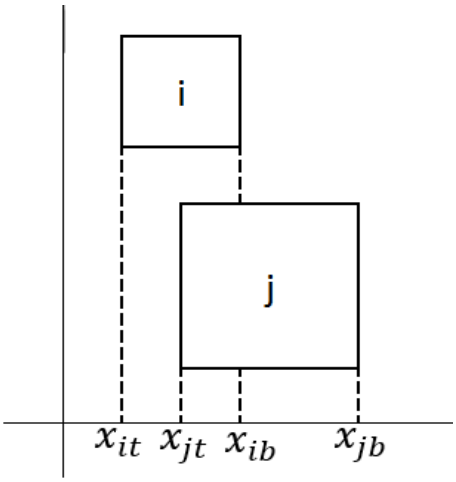


Figura 21 – Sobreposição em ordem a x (sobreposto). Fonte: Elaboração própria

O mesmo não ocorre na figura seguinte, uma vez que as coordenadas "x" não se sobrepõem.

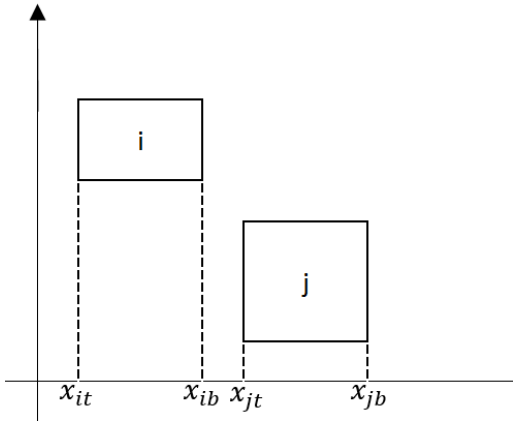


Figura 22 – Sobreposição em ordem a x (afastado). Fonte: Elaboração própria

Concluímos que, $(x_{jt} - x_{ib})(x_{jb} - x_{it}) \geq 0$ e que os dois equipamentos não estão sobrepostos em ordem a "x".

Analogamente, a condição para que dois equipamentos i e j não se sobreponham nas suas coordenadas y é:

$$(y_{jt} - y_{ib})(y_{jb} - y_{it}) \geq 0$$

A Figura 23 mostra a sobreposição das coordenadas "y entre o equipamento "i" e "j".

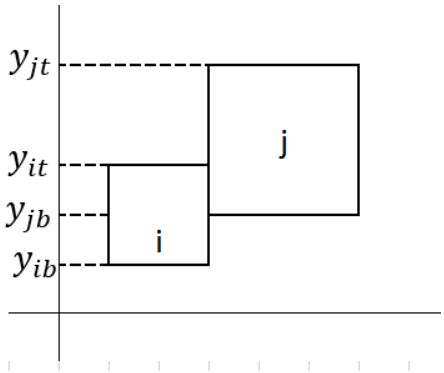


Figura 23 – Sobreposição em ordem a y (sobreposto). Fonte: Elaboração própria

Os autores Mir e Imam (2001) defendem que a área de sobreposição A_{ij} entre as duas instalações seja formulada pelo seguinte modelo.

Restrições:

$$A_{ij} = \gamma_{ij}(\Delta x_{ij})(\Delta y_{ij})$$

$$A_{ij} \leq 0$$

$$\Delta x_{ij} = \gamma_{ij}\left(\frac{L_i + L_j}{2}\right) - |x_i - x_j|$$

$$\Delta y_{ij} = \gamma_{ij}\left(\frac{W_i + W_j}{2}\right) - |y_i - y_j|$$

$$\gamma_{ij} = \begin{cases} -1 & se\ \Delta x_{ij} \leq 0\ e\ \Delta y_{ij} \leq 0 \\ 1 & caso\ contrário \end{cases}$$

Neste modelo, (L_i, W_i) são o comprimento e a largura da instalação i, e (x_i, y_i) são as coordenadas do centro da instalação i.

Assim sendo o Δx_{ij} vai nos verificar se as instalações estão ou não estão sobrepostas em relação às coordenadas de "x", o mesmo se aplica para Δy_{ij} em que nos indica a sobreposição em relação à coordenada "y". Já o γ_{ij}vai nos restringir a relações das sobreposições coordenadas "x" e "y".

Dado que o modelo usa variáveis inteiras (γ_{ij}) e variáveis contínuas (x_i, y_i), designa-se por um modelo de programação inteira misto (MIP).

Existem outras dificuldades que também se devem considerar na formulação do *layout*, tal como uma orientação pré-definida de determinadas instalações (Dunker et al., 2005). Dadas estas dificuldades, uma formulação típica alternativa para o problema de otimização pode ser dada por:

$$Min \sum_{i=1}^{n} \sum_{j=1}^{n} f_{ij}(|x_j^I - x_i^o| + |y_j^I - y_i^o|)$$

Nesta formulação, tem-se que n é o número de secções, f_{ij} é a quantidade / custo de fluxo de material do ponto de drop-off da secção i para o ponto de pick-up da secção "j", (x_i^o, y_i^o) são as coordenadas do ponto de drop-off da instalação "i", e (x_j^I, y_j^I) são as coordenadas do ponto de pick-up da instalação "j".

3.3. Formulação de problemas de *layout* multiobjectivo

Na maioria dos artigos sobre problemas de *layout*, o principal objetivo é minimizar uma função relacionada com a viagem de peças (o custo total do manuseio de materiais, o tempo de viagem, a distância de viagem, etc.).

Para ser mais realista, alguns autores têm considerado mais do que um único objetivo. Por exemplo, Dweiri e Meier, (1996), visa minimizar simultaneamente o fluxo de manuseio de materiais, o fluxo de equipamentos e o fluxo de informações.

A maioria, dos autores combinam os diferentes objetivos em uma única função ou por meio da metodologia "*Analytic Hierarchy Process*" (AHP) (Harmonosky & Tothero, 1992; Yang & Kuo, 2003) ou então utilizam uma combinação linear dos diferentes objetivos (Chen & Sha, 2005).

Poucos autores usaram uma abordagem de "Pareto" para gerar um conjunto de soluções não-dominadas. Aiello, Enea e Galante, (2006) lidaram com um problema de *layout* relacionado com a minimização do custo de manuseio de materiais e a maximização de uma função de adjacência (avaliação dos pedidos de proximidade entre dois departamentos). O conjunto de soluções não-dominadas é então encontrado e uma solução "melhor" é então selecionada a partir deste conjunto usando o "Método Electre".

Segundo o autor Roy (1968) o "Metodo Electre" tem como objetivo obter um subconjunto de alternativas, no qual as alternativas que fazem parte desse subconjunto sobre classificam as que não fazem, em outras palavras, busca-se reduzir o tamanho do conjunto de alternativas, explorando o conceito de dominância.

Para isso, são utilizados dois índices: o índice de concordância, que mede a vantagem relativa de cada alternativa sobre as outras, e o índice de discordância, que mede a relativa desvantagem.

3.4. Resolução simultânea de diferentes problemas

É comum que outros problemas devam ser resolvidos em conjunto com o projeto de *layout*. Por exemplo, isto ocorre quando a conceção de sistemas de *layout* em grupo tem problemas de formação de grupo e complicação em determinar a posição de cada máquina dentro do grupo.

A posição de cada grupo da planta no chão de fábrica tem também de ser determinada. Em vez de formular e resolver estes problemas sequencialmente, por vezes é possível resolver estes dois problemas como um mesmo problema (Gupta, Gupta, Kumar, e Sundaram, 1996).

Capítulo 4 – Métodos de Otimização de Layout

4. Métodos de otimização de *layout*

Os métodos para o desenvolvimento de *layouts* são classificados de acordo com seu ponto de partida (Tompkins et al., 2003): construção ou melhoria. A construção consiste no desenvolvimento do *layout* novo, ao passo que a melhoria, como o próprio nome indica pretende melhorar um *layout* existente e visa aumentar ou rearranjar a disposição de equipamentos ou o fluxo dos produtos.

Assim sendo os métodos utilizados para resolver problemas de otimização combinatória são divididos em dois tipos: os métodos exatos e os métodos heurísticos.

Os métodos exatos procuram obter a solução ótima para o problema a partir da construção de modelos matemáticos de otimização e da implementação de algoritmos específicos para a sua resolução. Contudo, devido à complexidade combinatória de alguns problemas, o tempo computacional necessário para a sua resolução torna-se muito elevado, sendo, geralmente, um tempo não-polinomial.

Já os métodos heurísticos, são capazes de encontrar soluções viáveis em tempo de execução polinomial, mas não garantem a qualidade da solução encontrada. Estes métodos tentam adotar uma estratégia que equilibre a qualidade da solução obtida com o tempo total de processamento. Assim, o algoritmo, a cada iteração, aproxima-se da solução ótima. A qualidade da solução alcançada está diretamente relacionada ao tempo de execução.

Em suma, a princípio, quando resolvemos problemas de otimização através de métodos exatos, temos a garantia que a solução encontrada é a solução ótima e quando se utiliza os métodos aproximados, podemos garantir que a solução encontrada está próxima de uma solução ótima. Já quando, são utilizados os métodos heurísticos, não é possível garantir da otimização da solução encontrada.

A utilização de métodos exatos justifica-se em casos em que as instâncias consideradas para um determinado problema são relativamente pequenas ou o tempo de processamento disponível é adequado o bastante para a aplicação considerada.

Em resolução de problemas práticos, os métodos heurísticos têm-se mostrado bastante eficazes uma vez que, permitem obter boas soluções, não ótimas, mas com uma maior rapidez que os métodos exatos.

4.1. Métodos exatos

Entre alguns artigos que tratam de métodos exatos, (Kouvelis e Kim, 1992) desenvolveram um algoritmo de *branch-and-bound*, para o problema de *layout* circuito unidirecional. Meller et al., (1999), também utilizou esta mesma abordagem para resolver o problema de submeter “n” instalações retangulares dentro de uma dada área retangular disponível. (Kim e Kim, 1999) abordou o problema de encontrar locais de *pick-up* e *drop-off (*P/D) em instalações de tamanho fixo para um determinado *layout*.

O objetivo do problema é o de minimizar a distância total de material que flui entre os pontos (P/D). Outros autores sugeriram o método de *branch-and-bound* para encontrar a localização ideal dos pontos de *pick-up* e *drop-off* de cada instalação.

Segundo Hillier e Lieberman, (2005) o *branch-and-bound*, é um método muito aplicado, com bastante ocorrência e no qual tem tido sucesso a uma larga variedade de problemas de programação inteira. Para lidar com este tipo de problemas, esta técnica baseia-se na ideia de "dividir e conquistar".

Neste caso o problema é dividido em vários sub-problemas suficientemente pequenos, ao ponto de ser possível executar a sua resolução. A fase em que se divide o problema original é a ação de *branch* que significa "ramificar". Analisa-se cada ramo procurando a solução ótima dentro desse ramo e usa-se essa informação para a ação de *bound* que significa "construir um limite" e que é feita à medida que se descobre uma solução melhor que a anterior. Este método de branch-and-bound, ao ser realizado a todo o problema, garante que a melhor solução encontrada seja a solução ótima.

Como exemplo prático considere-se a seguinte situação: uma empresa tem dois produtos na linha de produção e necessita de saber as quantidade x e y destes produtos a serem produzidas para obter o máximo lucro.

Suponha ainda que a formulação deste problema em programação linear é a seguinte:

Função Objetivo:

$\text{Max} = 3x + 7y$

Restrições:

$x \leq 3.5$

$5x - 4y \leq 10$

$\frac{4}{7}x + 2y \leq 9$

$x, y \geq 0$ *e inteiras*

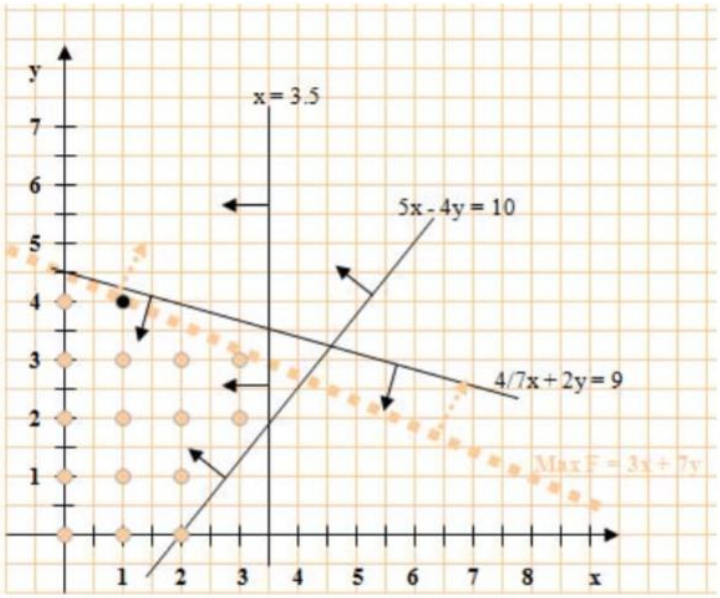


Figura 24 – Programação Linear fonte: Carravilla e Oliveira (2001)

Este problema tem como solução ótima inteira x=1 e y=4. Vamos usá-lo para ilustrar a aplicação do método de Branch-and-bound e perceber como se pode obter esta solução ótima inteira.

O primeiro passo do método consiste em considerar, não o problema original, mas sim a sua Relaxação Linear, que é quase o mesmo problema mas eliminando as condições de integralidade das variáveis. As variáveis x e y deixam de ser consideradas obrigatoriamente números inteiros e passam a ser apenas números reais não negativos:

Função Objetivo:

$\text{Max} = 3x + 7y$

Restrições:

$x \leq 3.5$

$5x - 4y \leq 10$

$\frac{4}{7}x + 2y \leq 9$

$x, y \geq 0$

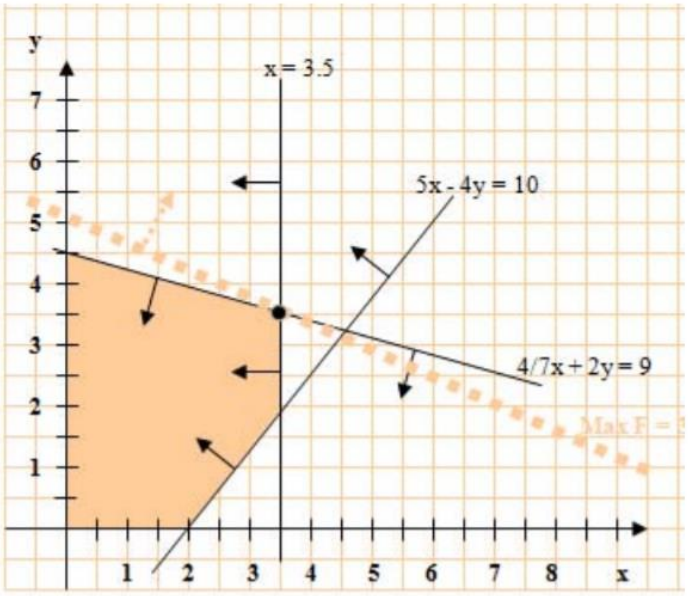


Figura 25 – Nó LP0 da relaxação linear Fonte: Carravilla e Oliveira (2001)

Nesta primeira etapa a solução ótima encontrada para a relaxação linear é $F = 35$ e que $x = 3{,}5$ e $y = 3{,}5$. Isto significa que 35 é um limite superior para a solução inteira do problema. Neste caso como o x e y não são números inteiros teremos que começar a ramificar o problema, eliminando a possibilidade de voltar a obter esta solução fracionária. Para isso escolhe-se uma variável cuja solução não seja inteira e usa-se para ramificar em dois subproblemas, acrescentando a cada um a restrição de que esta variável deve ser inferior ou igual ao valor da solução arredondado para o inteiro imediatamente inferior, ou superior ou igual ao arredondamento para o inteiro imediatamente superior.

Sendo assim, aqui pode-se optar por ramificar através da variável $x = 3{,}5$, que arredondada seria 3 ou 4. Num ramo acrescentamos a restrição $x \leq 3$ e no outro a restrição $x \geq 4$. Desta forma a solução anterior $x = 3{,}5$ fica impossibilitada em ambos os ramos mas todas as soluções inteiras continuam possíveis em algum dos ramos LP01 e LP02.

Então temos a seguinte situação no ramo LP01:

Função Objetivo:

$\text{Max} = 3x + 7y$

Restrições:

$x \leq 3.5$

$5x - 4y \leq 10$

$\frac{4}{7}x + 2y \leq 9$

$x, y \geq 0$

$x \leq 3$

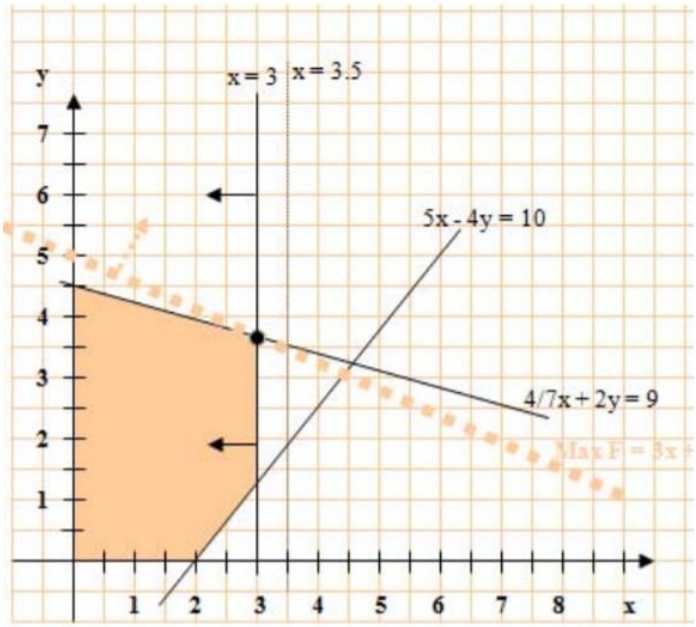


Figura 26 - Nó LP01 da relaxação linear Fonte: Carravilla e Oliveira (2001)

A solução encontrada é uma solução não inteira em que $F = 34{,}5$ e que $x = 3$ e $y = 3{,}6$. De seguida aplicamos a mesma resolução em que $x \geq 4$ que fica no nó LP02 da árvore de pesquisa:

Função Objetivo:

$\text{Max} = 3x + 7y$

Restrições:

$x \leq 3.5$

$5x - 4y \leq 10$

$\frac{4}{7}x + 2y \leq 9$

$x, y \geq 0$

$x \geq 4$

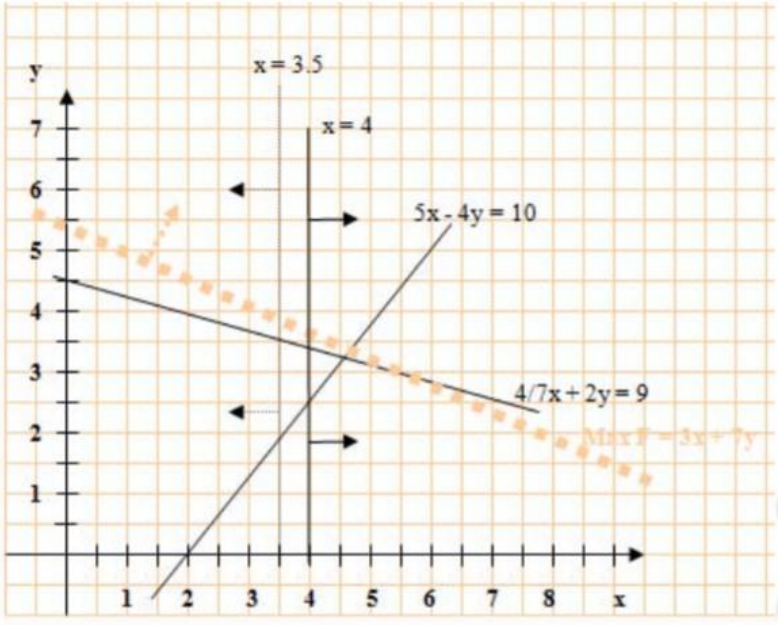


Figura 27 - Nó LP02 da relaxação linear Fonte: Carravilla e Oliveira (2001)

Nesta solução conseguimos verificar que a solução não é admissível, porque a primeira e última restrições contradizem-se. Assim teremos que dividir mais uma vez a solução anteriormente encontrada. Neste caso mantemos a restrição $x \leq 3$ do nó LP01, uma vez que foi uma solução admissível e iremos procurar a solução de modo a que $y \leq 3$ (nó LP011) ou então $y \geq 4$ (nó LP012). Assim o nó LP011 contém o seguinte problema:

Função Objetivo:

$\text{Max} = 3x + 7y$

Restrições:

$x \leq 3.5$

$5x - 4y \leq 10$

$\frac{4}{7}x + 2y \leq 9$

$x, y \geq 0$

$x \leq 3$

$y \leq 3$

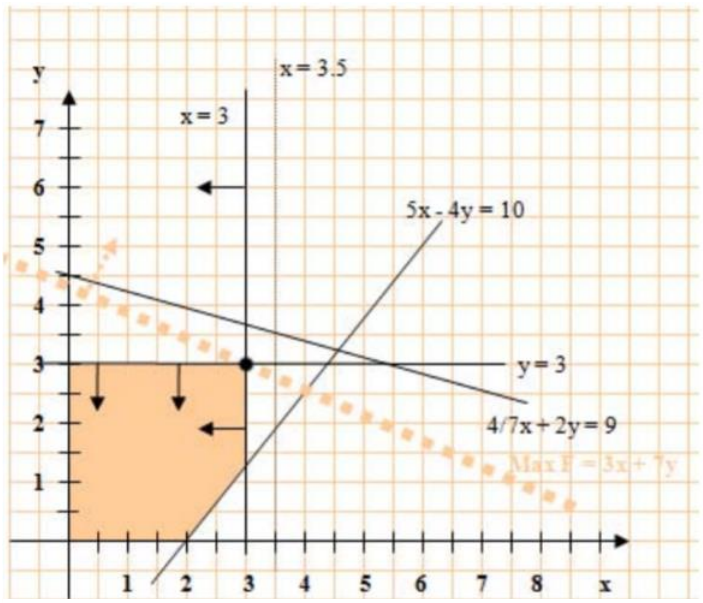


Figura 28 - Nó LP011 da relaxação linear Fonte: Carravilla e Oliveira (2001)

A solução encontrada é uma solução inteira em que $x = 3$ e $y = 3$ e é a melhor solução encontrada até ao momento. Apesar de ser uma solução para o problema inteiro como se pretendia, apenas garante que $F = 30$ seja um limite inferior para a solução ótima inteira, por isso temos que explorar a solução em que $y \geq 4$, para nos certificarmos se a solução é a melhor.

No nó LP012 o problema a resolver é o seguinte:

Função Objetivo:

$\text{Max} = 3x + 7y$

Restrições:

$x \leq 3.5$

$5x - 4y \leq 10$

$\frac{4}{7}x + 2y \leq 9$

$x, y \geq 0$

$x \leq 3$

$y \geq 4$

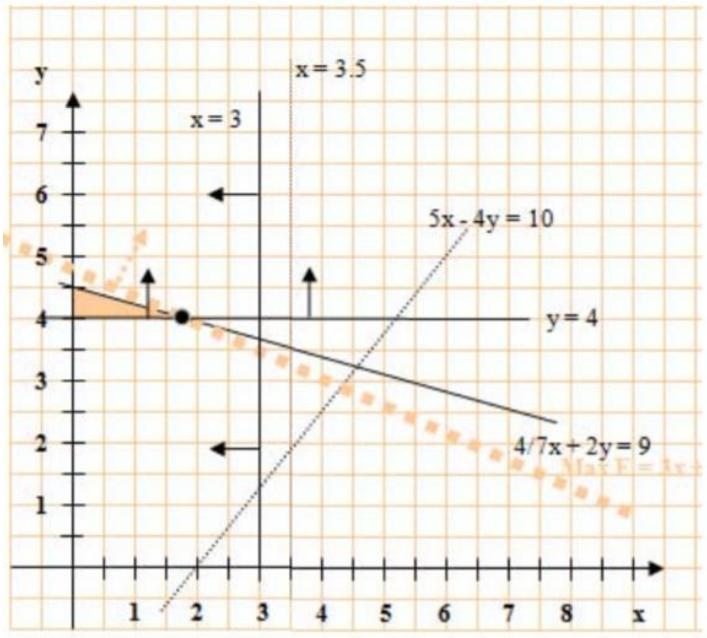


Figura 29 - Nó LP012 da relaxação linear Fonte: Carravilla e Oliveira (2001)

Encontrada a solução não inteira em que $F = 33{,}2$, e que $x = 1{,}7$ e $y = 4$, temos que novamente dividir. Assim, arredondando $x = 1{,}7$ para os inteiros mais próximos temos as restrições $x \leq 1$ ou $x \geq 2$.

O primeiro caso origina o seguinte problema no nó LP0121:

Função Objetivo:

$\text{Max} = 3x + 7y$

Restrições:

$x \leq 3.5$

$5x - 4y \leq 10$

$\frac{4}{7}x + 2y \leq 9$

$x, y \geq 0$

$x \leq 3$

$y \geq 4$

$x \leq 1$

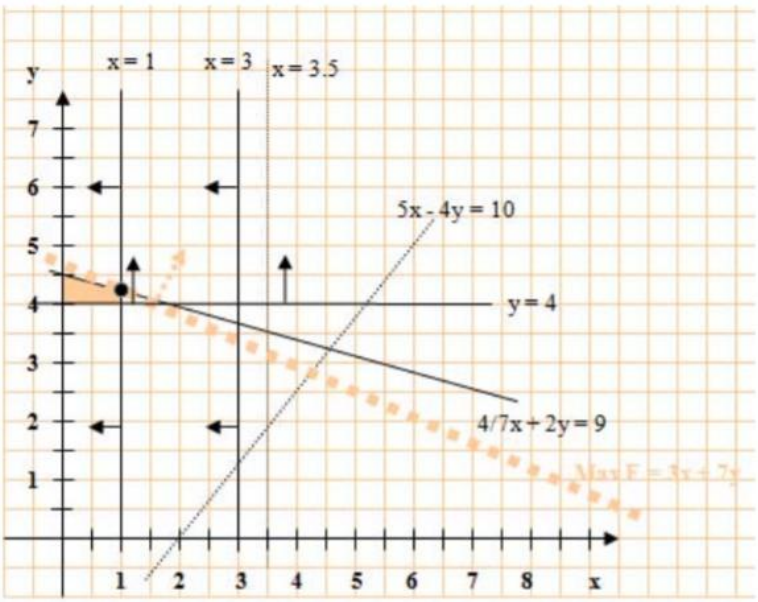


Figura 30 - Nó LP0121 da relaxação linear Fonte: Carravilla e Oliveira (2001)

Mais uma vez a solução não é inteira em que $F = 32{,}5$, e que $x = 1$ e $y = 4{,}2$, assim, iremos testar para $x \geq 2$ para encontrar uma solução que seja inteira.

Função Objetivo:

$\text{Max} = 3x + 7y$

Restrições:

$x \leq 3.5$

$5x - 4y \leq 10$

$\frac{4}{7}x + 2y \leq 9$

$x, y \geq 0$

$x \leq 3$

$y \geq 4$

$x \geq 2$

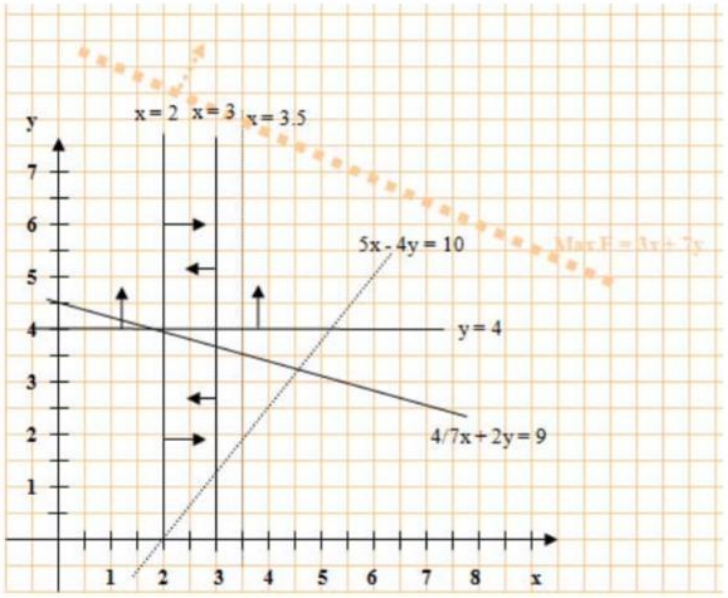


Figura 31 - Nó LP0122 da relaxação linear Fonte: Carravilla e Oliveira (2001)

O problema originado neste nó não tem qualquer solução admissível, por isso teremos que dividir mais uma vez algum dos nós deixados pendentes anteriormente. Neste caso vamos voltar ao nó LP0121, uma vez que foi uma solução admissível e iremos procurar a solução de modo a que $y = 4$ e $y = 5$.

Função Objetivo:

$\text{Max} = 3x + 7y$

Restrições:

$x \leq 3.5$

$5x - 4y \leq 10$

$\frac{4}{7}x + 2y \leq 9$

$x, y \geq 0$

$x \leq 3$

$y \geq 4$

$x \leq 1$

$y \leq 4$

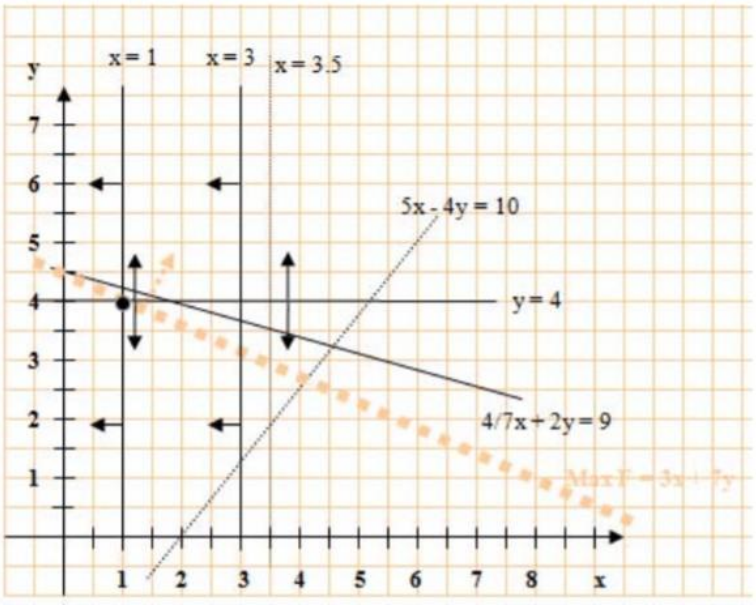


Figura 32 - Nó LP01211 da relaxação linear Fonte: Carravilla e Oliveira (2001)

Esta solução encontrada é uma solução inteira em que $x = 1$ e $y = 4$ e $F = 31$ é a melhor solução encontrada até ao momento. Apesar de solução inteira melhor, temos que explorar a situação em que $y \geq 5$, para nos certificarmos se a solução é a melhor.

Função Objetivo:

$\text{Max} = 3x + 7y$

Restrições:

$x \leq 3.5$

$5x - 4y \leq 10$

$\frac{4}{7}x + 2y \leq 9$

$x, y \geq 0$

$x \leq 3$

$y \geq 4$

$x \leq 1$

$y \geq 5$

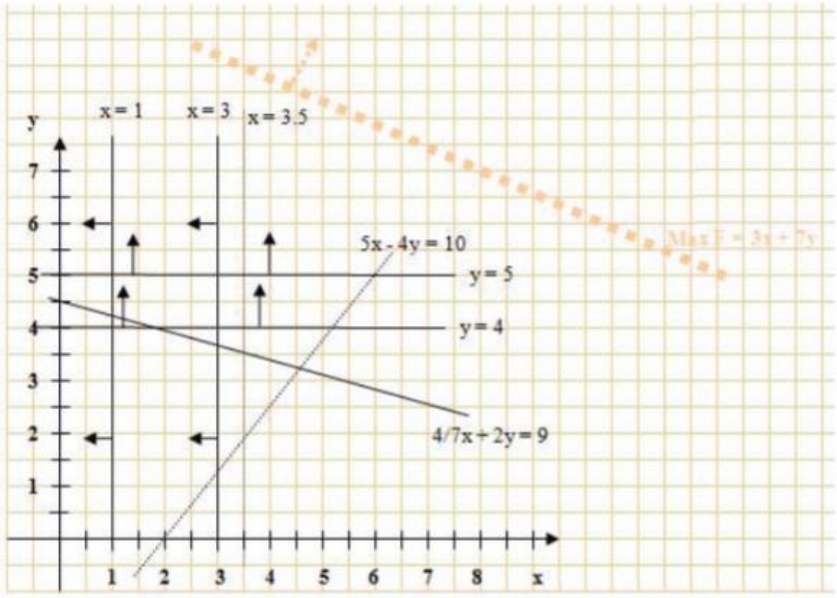


Figura 33 - Nó LP01212 da relaxação linear Fonte: Carravilla e Oliveira (2001)

Por fim o problema encontrado neste nó não tem soluções admissíveis, e portanto a melhor solução de todas foi a, obtida anteriormente em que $x = 1$ e $y = 4$ e $F = 31$. Neste caso a figura que se segue apresenta a árvore de pesquisa do método de *Branch-and-bound* e permite uma melhor visualização das divisões feitas ao longo deste processo.

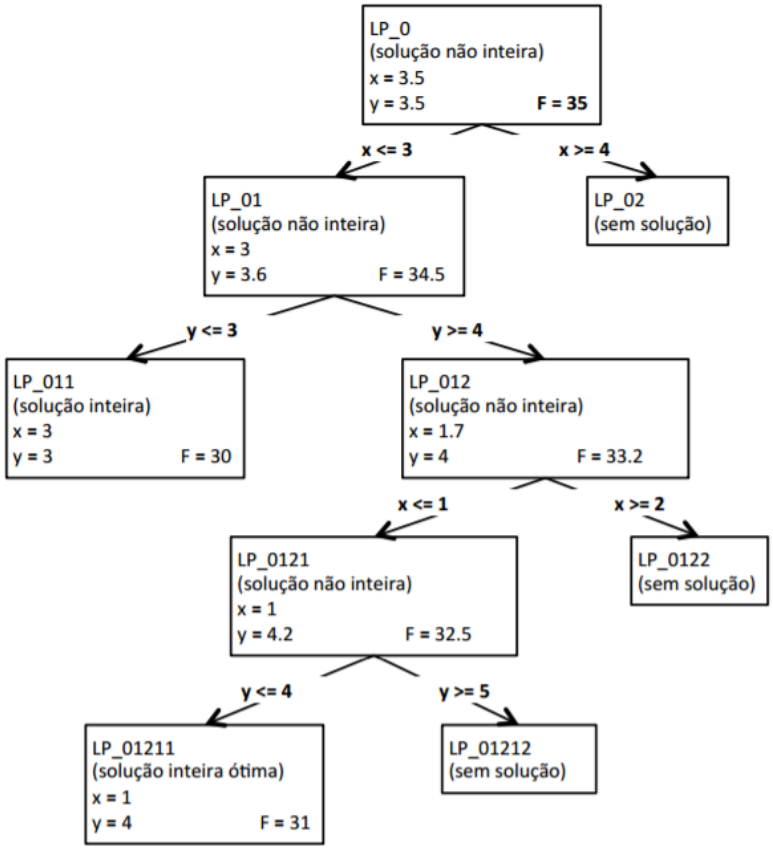


Figura 34 – Árvore de branch-and-bound Fonte: Carravilla e Oliveira (2001)

Este mesmo processo é usado para solucionar problemas de programação linear inteiros (MIP) tais como os modelos que apresentamos acima para planificação de *layouts* uma vez que nos permite restringir áreas, custos de fluxos, distâncias ente sectores, entre outros.

4.2. Métodos Heurísticos

Os métodos exatos são muito vantajosos para encontrar uma solução ótima em algumas utilizações práticas. Contudo, há dificuldade em resolver problemas com instâncias de maiores dimensões (Arroyo, 2002), por esse motivo, os algoritmos exatos deixam de ser eficientes na resolução, porque podem levar muito tempo computacional até encontrarem a melhor solução (Garey & Johson, 1979).

Assim sendo segundo (Drira, 2007) podemos classificar os métodos heurísticos em quatro grandes grupos, heurísticas construtivas, heurísticas de melhoramento, meta-heurísticas e meta-heurísticas hibridas.

4.2.1. Heurísticas Construtivas

Este tipo de heurísticas são responsáveis por construir uma solução viável no qual a sua solução é construída através de passo a passo. Em outras palavras, as instalações são colocadas uma a uma, pelo nível de importância e as suas respetivas adjacências, até todas as seções serem alocadas, completando assim o *layout*.

Estes métodos são bastante rápidos, mas a qualidade das soluções não é melhor do que a obtida por outras heurísticas (Hoos & Stützle, 2004). Por isso as heurísticas construtivas podem ser usadas como ponto de partida para a subsequente aplicação de outras.

Um estudo revelado por Moore (1974) afirma que existem duas vezes mais heurísticas construtivas que heurísticas de melhoramento. Alguns dos mais conhecidos algoritmos de construção são. CORELAP (Lee & Moore, 1967), ALDEP (Seehof & Evans, 1967) e COFAD (Tompkins & Reed, 1976), SHAPE (Hassan, Hogg, & Smith, 1986).

4.2.2. Heurísticas de Melhoramento

As heurísticas de melhoramento são um método usado para melhorar uma determinada solução, aplicando movimentos nos elementos da solução. Através de uma solução inicialmente alcançada, este método, aplica através de um procedimento iterativo, no qual por norma, esses procedimentos envolvem trocas de posições das tarefas na sequência do processamento das máquinas, buscando encontrar uma esboço de tarefas melhor que a atualmente solucionada.

Essa nova solução é designada de solução vizinha. As soluções vizinhas geradas a partir de um grupo de movimentos formam a vizinhança da solução.

Após cada iteração, a solução atual x é substituída por uma solução vizinha chamada por exemplo x' se a solução x' for melhor que a solução x. A busca termina quando todas as soluções vizinhas criadas forem piores que a solução atual, ou após um numero máximo de iterações estipulado à partida, representando que foi encontrada uma boa solução.

Exemplos de heurísticas de melhoria são: CRAFT (Armour & Buffa, 1963), FRAT (Khalil, 1973) e DISCON (Drezner, 1987).

4.3. Métodos Meta-heurísticos

Em casos em que não sejam conhecidos métodos exatos que resolvam o *facility layout problem* em tempo aceitável, o uso de meta-heurísticas é por norma uma solução geralmente recomendada. Uma das principais vantagens das meta-heurísticas é serem capazes de fornecer soluções de alta qualidade em um tempo razoavelmente pequeno e também para problemas de grande dimensão.

As meta-heurísticas podem ser definidas como procedimentos de propósito geral usados para resolver diferentes problemas de otimização combinatória.

A palavra “meta” é utilizada para descrever uma heurística que está sobreposta a uma outra heurística, constituindo um diferente “nível heurístico”. Segundo Osman e Laporte, (1996), afirmam que os métodos meta-heurísticos são um processo que guia uma heurística subordinada, com a combinação de diferentes conceitos para explorar e aproveitar o espaço de busca, utilizando estratégias de aprendizagem para organizar a informação a fim de encontrar de modo eficaz soluções próximas de ótimas ou mesmo soluções ótimas.

Algumas destas meta-heuristicas apresentadas serão, *Simulated Annealing, Tabu Search, Genetic Algorithm, Ant Colony*, entre outras. No caso da meta-heurística *Ant Colony*, a mesma vai ser substituída pelo *Particle Swarm Optimization* uma vez que o princípio de funcionamento é bastante idêntico.

- *Simulated* A*nnealing*: método que se baseia na procura aleatória da vizinhança, inspirado no comportamento termodinâmico da matéria;

- *Tabu Search:* método de procura local que utiliza uma lista de movimentos proibidos, aceitando um sujeito da vizinhança mesmo que ele deteriore o valor da função objetivo;

- *Genetic Algorithm*: método baseado no conceito de classe evolutiva, que privilegia os sujeitos mais adaptados, onde o procedimento de procura é baseado nos operadores de combinação e modificação;

- *Ant Colony:* método inspirado no comportamento de orientação das formigas para encontrar o melhor caminho, onde cada sujeito compartilha a informação com os outros sujeitos acerca do caminho experimentado aleatoriamente;

- *Particle Swarm:* método muito semelhante ao das colónias de formigas, herdando os modelos sociocognitivos aplicados à aprendizagem, tais como avaliação de estímulos, adaptação cultural e imitação.

4.3.1. Simulated Annealing

Este método é muito conhecido por “arrefecimento simulado” no qual teve as suas origens por A. H. Teller e E. Teller, (1953), quando foi usado pela primeira vez, para simular o processo de arrefecimento de cristais e foi posteriormente descrita por Kirkpatrick et al. 1983.

Como anteriormente foi introduzido, este método vem do comportamento termodinâmico no qual, o arrefecimento de alguns materiais consiste em submete-los numa fase inicial a altas temperaturas e reduzi-las gradualmente até atingirem o equilíbrio térmico. O equilíbrio térmico consiste em aumentos e reduções do estado de energia, tornando-os assim, consistentes e rígidos.

O método de *simulated annealing* é um algoritmo de forma probabilística, onde uma solução vizinha "s" é descoberta uniformemente de maneira aleatória no conjunto $N(s)$. Então, toda solução vizinha "s'" pertencente ao conjunto $N(s)$ tem a probabilidade $\frac{1}{|N(s)|}$ de ser escolhida, sendo que a primeira solução "s'" gerada tem um melhor custo que "s" é aceite, ou seja, a solução "s" é substituída por "s'".

Assim sendo esta técnica evita mínimos locais que, por vezes, aceita pontos que podem ter valores maiores para a função objetivo, fazendo que, em algumas iterações, o algoritmo tenda a maximizar a função objetivo em vez de minimizá-la.

4.3.2. Tabu Search

É uma meta-heurística que faz uma pesquisa na vizinhança explorando o espaço de soluções além da otimização local, de uma forma eficiente. O processo de procura é acelerado usando métodos aleatórios de seleção no qual a melhor solução vizinha é escolhida. A estratégia de escolher o melhor vizinho, em conjunto com uma estrutura de memória para armazenar as soluções geradas, tem como objetivo não deixar a procura presa em um ótimo local.

Assim sendo o algoritmo inicia com uma solução inicial "S_0", e a cada iteração um subconjunto "V" da vizinhança do conjunto $N(s)$ é explorado ($V \subset N(s)$). Se a solução "s'" pertencer a "V" e tem um melhor valor da função objetivo "f", é então selecionada como nova solução "s", mesmo que "s'" seja pior que "s", isto é, que $f(s') > f(s)$.

Este critério de escolha do melhor vizinho é utilizado para fugir de ótimos locais, no entanto, isto pode fazer com que o algoritmo retorne a soluções já analisadas, fazendo ciclos no sistema. Para que isso não ocorra, usa-se uma lista tabu "T", uma lista que contem soluções analisadas recentemente e que ficam proibidas de serem analisadas por um certo número de iterações, evitando dessa maneira, caminhos cíclicos no grafo de vizinhança.

A lista possui movimentos reversos aos últimos movimentos realizados e trabalha como uma lista de tamanho limitado, ou seja, quando um novo movimento é adicionado à lista e a mesma se encontra cheia, o movimento mais antigo sai. Dessa forma, quando o subconjunto $V \subset N(s)$ é explorado os "n" elementos que estão na lista tabu "T" são excluídos da busca, isto é, os vizinhos "s'" que são obtidos dos "n" movimentos em "T" são excluídos da procura.

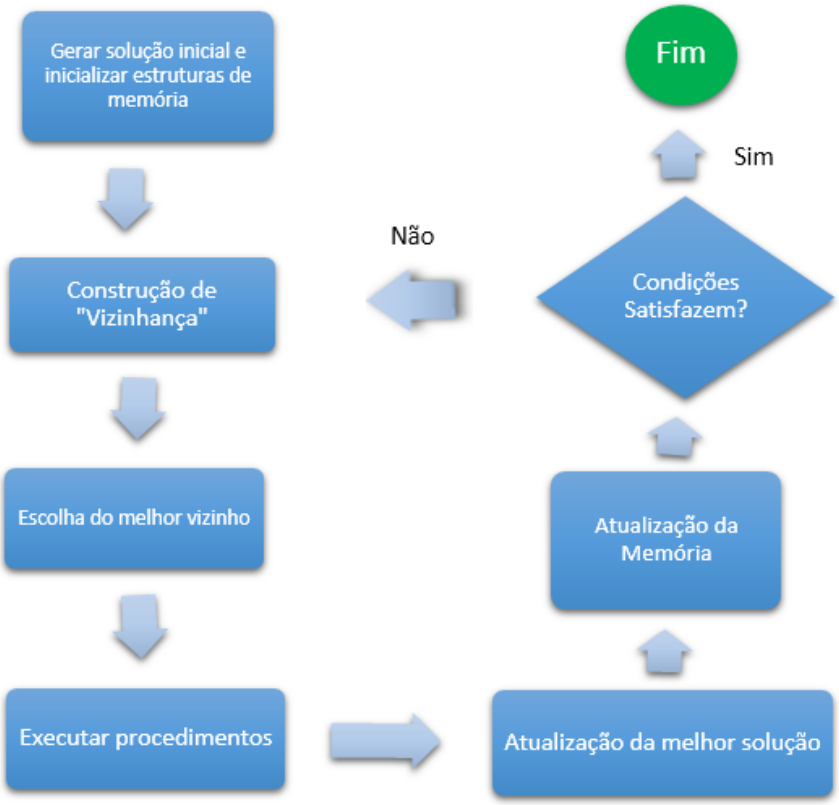


Figura 35 – Diagrama de Tabu Search. Fonte: Elaboração própria

4.3.3. Algoritmos Genéticos

O método de *genetic algorithm*, é baseado numa população de representações abstratas, baseadas em processos de seleção natural de evolução para que melhore a solução do problema. Os indivíduos com caraterísticas melhores têm maior chance de sobrevivência e de produzir filhos cada vez mais aptos, e os indivíduos menos aptos tendem a desaparecer (Goldberg,1989 e Golden, 1977).

Esta técnica de programação que reproduz a evolução biológica tem quatro elementos essenciais para a sua criação. A **população** é um conjunto de **indivíduos** ou também chamados de **cromossomas**. Cada indivíduo equivale a uma solução do problema, dessa forma, uma população é um conjunto de soluções. Cada indivíduo é constituído por componentes que são chamados de **alelos**. Cada alelo, são possíveis valores que cada componente da solução pode atingir. O valor que cada alelo possui é chamado de **gene**. A figura 36 representa os elementos do *genetic algorithm*.

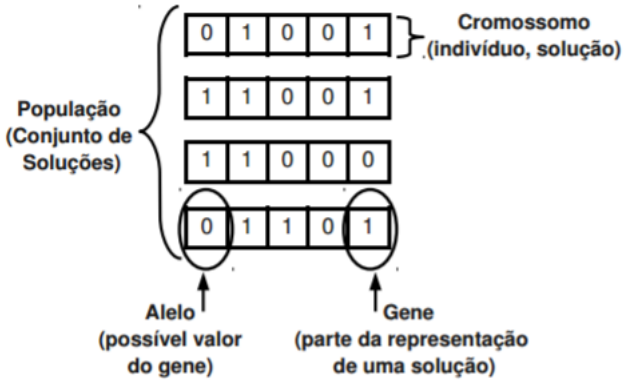


Figura 36 - Elementos de um Algoritmo Genético Fonte: Costa 2011

No algoritmo, existem quatro operações básicas que são: cálculo de aptidão (*fitness evaluation*), seleção (*selection*), cruzamento (*crossover*) e mutação (*mutation*).

Na primeira fase de aptidão são escolhidos dois elementos mais fortes da solução, que vão ser chamados de pais, seguida passam por uma fase de reprodução através da seleção de um ponto de corte. Numa terceira fase são gerados os filhos com base no ponto de corte e por fim é aplicado um processo de mutação, onde um determinado gene pode ser modificado. A figura seguinte ilustra o procedimento principal de um *genetic algorithm.*

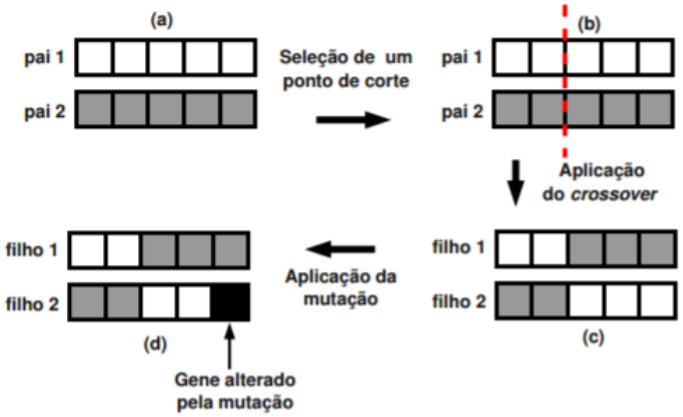


Figura 37 - Procedimento Básico de um Algoritmo Genético Fonte: Costa 2011

4.3.4. Ant Colony

A técnica *Ant Colony* foi criada por Dorigo, 1996 e é inspirada no comportamento das colónias de formigas. Como muitas espécies de formigas são quase cegas, a informação entre elas é realizada através de uma substância química chamada "feromona". Algumas espécies de formigas usam a feromona para criar os caminhos, que as guiam. As formigas que saem aleatoriamente da colónia à procura de alimentos, quando o encontram, elas depositam a feromona no chão, fazendo um caminho. As formigas sentem o odor da feromona, e selecionam com maior probabilidade o caminho

que tem o cheiro mais intenso, ou seja, com maior proporção de feromona. Estes caminhos são usados para encontrar o alimento e achar o caminho de volta à colónia.

As formigas artificiais constroem de forma incremental soluções através da agregação de elementos ou conjuntos de elementos do grafo. O processo de construção das soluções é estocástico e guiado pelo modelo de feromonas, que é um conjunto de valores associados com as componentes do grafo, sejam vértices ou arestas, e que sofrem alteração durante o trabalho.

Esta técnica é aplicada em programação de layout, uma vez que cada equipamento é alocado a cada divisão conforme a sua proximidade e relação com os outros equipamentos. A ligação entre equipamentos gera assim vários caminhos interligados, sendo criadas várias rotas que os materiais têm que percorrer conforme esquematizado na imagem seguinte.

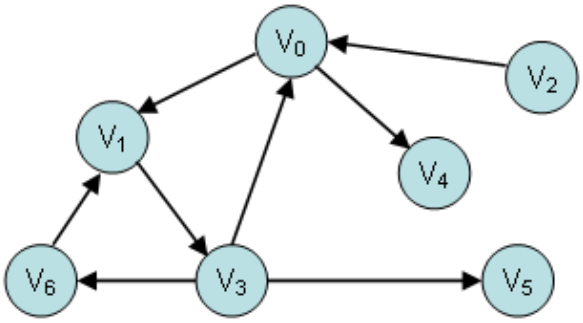


Figura 38 – Diagrama Ant Colony Fonte: https://www.cpp.edu/~ftang/courses/CS241/notes/graph.htm

Em cada vértice (i, j) do grafo, é definida uma variável τ_{ij}, conhecida como caminho artificial de feromonas. No início, τ_{ij} é igual para todos os vértices da rede. Cada formiga "k" produz uma solução a partir de um dos vértices do grafo de forma aleatória. Assim, a variável τ_{ij} é incrementada conforme o passar das formigas e decrementada a cada ciclo. A intensidade das feromonas no caminho esclarecerá a utilidade deste para as formigas, isto é, quanto maior a quantidade de feromonas, melhor é o caminho e as formigas tendem a utilizar esse percurso em vez de outros.

A cada nó, a formiga artificial executa uma função estocástica para calcular a probabilidade de utilização dos vértices.

Obtemos assim uma fórmula probabilística, que veremos de seguida, onde cada formiga "k" constrói o seu caminho movendo-se através de uma sequência de locais vizinhos. O termo τ_{ij} controla a deposição do feromonas nos vértices, enquanto o η_{ij} é um valor heurístico relacionado à natureza do problema e que permite uma maior exploração.

Já os parâmetros "α" e "β" controlam a intensidade de feromonas τ_{ij}, e a qualidade da aresta η_{ij}, respetivamente.

$$p_{ij}^{k} = \frac{\left[\tau_{ij}\right]^{\alpha} * \left[\eta_{ij}\right]^{\beta}}{\sum_{l \in J_i^k}\left[\tau_{ij}\right]^{\alpha} * \left[\eta_{ij}\right]^{\beta}}$$

No qual podemos definir os seguintes campos:

p_{ij}^{k} é a probabilidade da formiga "k", que se encontra na cidade "i", escolher o vértice "j" como próximo vértice a ser visitado;

τ_{ij} é a quantidade de feromonas existente no caminho (i, j). Inicialmente, define-se um mesmo valor "T_0" para todos os vértices da rede;

η_{ij} é a função heurística que representa a atratividade do vértice (i, j);

$l \in J_i^k$ é o conjunto de pontos ainda não foram visitados pela formiga "k", que se encontra atualmente no ponto "i";

"α" é um parâmetro que pondera a importância relativa do caminho de feromonas τ_{ij} na decisão de movimentação da formiga;

"β" é valor heuristicamente escolhido, que pondera a influência relativa da distância η_{ij} entre os nós "i" e "j" no processo de decisão;

Sendo assim observar que se $\alpha = 0$, as formigas selecionarão a heurística de vizinho mais próximo, enquanto que se $\beta = 0$, as formigas selecionarão o caminho com maior nível de feromonas e pode ocorrer uma estagnação com o passar das iterações, levando o algoritmo a pontos sub-ótimos.

Na equação anterior é possível verificar que a preferência da formiga por um determinado caminho é maior para os caminhos com maior nível de feromonas e com menor distância. Para que esta situação não ocorresse utiliza-se uma atualização global.

Nesta atualização é pretendido recompensar as arestas que pertencem a rotas mais curtas. Uma vez que as formigas artificiais terminam seus caminhos, a melhor formiga deposita feromonas em arestas visitadas que pertencem a seu caminho e as outras permanecem inalteradas.

Assim, a cada vértice (i, j) da rede, adiciona-se uma quantidade de feromonas proporcional ao tamanho da rota alcançada, ou seja, quanto mais curta a rota, maior será a quantidade de feromonas depositadas. Podemos ver essa atualização na seguinte equação:

$$\tau_{ij} = (1 - \rho) * \tau_{ij} + \rho . \Delta\tau_{ij} \;\; \rho \in [0,1]$$

Onde:

$$\Delta\tau_{ij} = \begin{cases} \frac{1}{L_n}, & se\ (i,j) usado \\ 0 & c.c \end{cases}$$

No primeiro termo das equações temos a responsabilidade pela evaporação das feromonas. O parâmetro *"ρ"* é utilizado para que os caminhos menos frequentados sejam esquecidos com o passar das iterações. Já o segundo termo da equação é responsável pelo aumento da concentração de feromonas apenas nos arcos visitados pela melhor formiga, onde "L_n" é a distância total percorrida na rota construída pela melhor formiga da iteração. Quanto menor a rota, maior a quantidade de feromonas depositadas. Esse procedimento repete-se até atingir o número máximo de iterações ou caso não se verifique mais melhorias nas soluções encontradas.

4.3.5. Particle Swarm

Este método é muito idêntico ao anterior mas funciona como um enxame de partículas, no qual foi desenvolvido por Kennedy e Eberhart, em 1995, partir de uma análise do comportamento de um grupo de pássaros à procura de alimento ou de um lugar para criar um ninho.

O *Particle Swarm* cria uma simulação passeada no "comportamento social" dos pássaros, ou seja, quando um pássaro encontra alimento, todo o bando passa a encontrar também o alimento, de uma forma mais rápida. O que ocorre é uma aprendizagem por parte do bando no momento em que um dos pássaros adquire determinado conhecimento.

O enxame de partículas é muito similar aos métodos de computação evolucionária em que, uma população (enxame), formada por indivíduos (partículas) procuram no espaço de busca a solução apropriada para uma determinado problema. No caso da otimização, cada indivíduo tem uma velocidade, responsável pela exploração do espaço (evolução) e uma memória, para guardar a melhor posição já visitada (Eberhart et al, 1996). O algoritmo também considera a melhor posição encontrada pela população.

Cada partícula é tratada como um ponto dentro do espaço de busca, que ajusta a sua própria pesquisa de acordo com a sua própria experiência, bem como a experiência de pesquisa de outras partículas (Parsopoulos e Vrahatis, 2002).

No algoritmo de *Particle Swarm*, primeiro gera-se uma população inicial (bando), onde cada indivíduo (partícula) é candidato a uma possível solução para o problema.

Por exemplo, considerando um espaço bidimensional, cada partícula possui uma posição no espaço de soluções (x,y) e uma velocidade que permite a mesma percorrer esse espaço. A cada iteração e para cada partícula, a velocidade (v_i) é atualizada, de

acordo com a velocidade anterior (v_i) somando a parte cognitiva da fórmula, que representa o conhecimento e mais a parte social, que representa a colaboração entre as partículas.

Parte cognitiva: $[c_1 * rand(\quad) * (pbest_i - x_i)]$

Parte social: $[c_2 * rand(\quad) * (gbest_i - x_i)]$

Assim sendo podemos dizer que a velocidade é atualizada pela seguinte fórmula:

$$v_i = v_i + [c_1 * rand(\quad) * (pbest_i - x_i)] + [c_2 * rand(\quad) * (gbest_i - x_i)]$$

Uma vez alcançada a velocidade podemos calcular a posição com a seguinte fórmula:

$$x_i = x_i + v_i$$

Com as duas fórmulas criadas podemos definir os seguintes campos:

v_i é a velocidade atual da partícula "i";

c_1e c_2 são parâmetros de confiança;

$rand(\quad)$ é a função aleatória;

$pbest_i$ é a melhor posição que a partícula "i" já alcançou durante a procura;

$gbest_i$ é a melhor posição encontrada pelas partículas do bando;

x_i é a posição atual da partícula "i".

Segundo Shi e Eberhart, (1998), os parâmetros de confiança indicam quanto uma particulada confia em si "c_1" e no bando "c_2". Estes parâmetros têm valores variáveis dependendo muito do problema em questão e são previamente conhecidos, podendo ser fixos ou variar em cada iteração.

Shi e Eberhart,(1998) revelaram nos seus estudos que o valor de confiança para "c_1" e "c_2" é de 2, no entanto Kennedy (1998) constatou que com o valor de 0,5, se obtinha melhor resultados.

A função do parâmetro $rand(\quad)$ é criar números aleatórios entre 0 e 1 para manter a diversidade do bando.

A equação que atualiza a velocidade da partícula é constituída por dois parâmetros de aceleração por distância: $(pbest_i - x_i)$ e $(gbest_i - x_i)$ no qual o primeiro parâmetro representa a distancia entre a melhor posição já encontrada pela partícula "i" e a sua posição atual. O segundo parâmetro representa a distância entre a posição encontrada pelo bando e a posição atual da partícula "i".

É possível visualizar o princípio de funcionamento deste método no diagrama a baixo no qual nos dá uma melhor precessão da sua implementação.

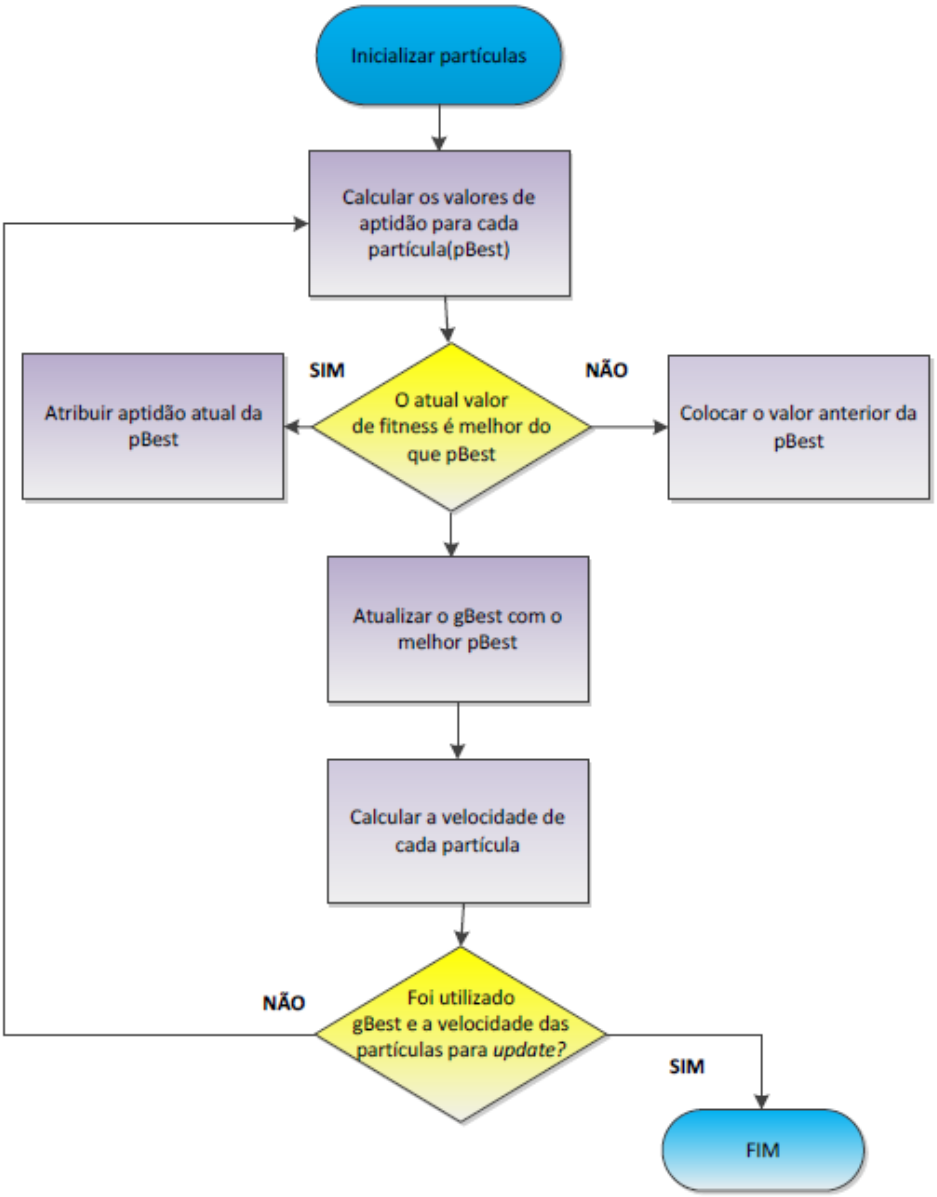


Figura 39 – Diagrama PSO. Fonte: Seixas (2013)

Na figura que se segue podemos perceber o que acontece na realidade à movimentação das partículas e os seus respetivos lugares.

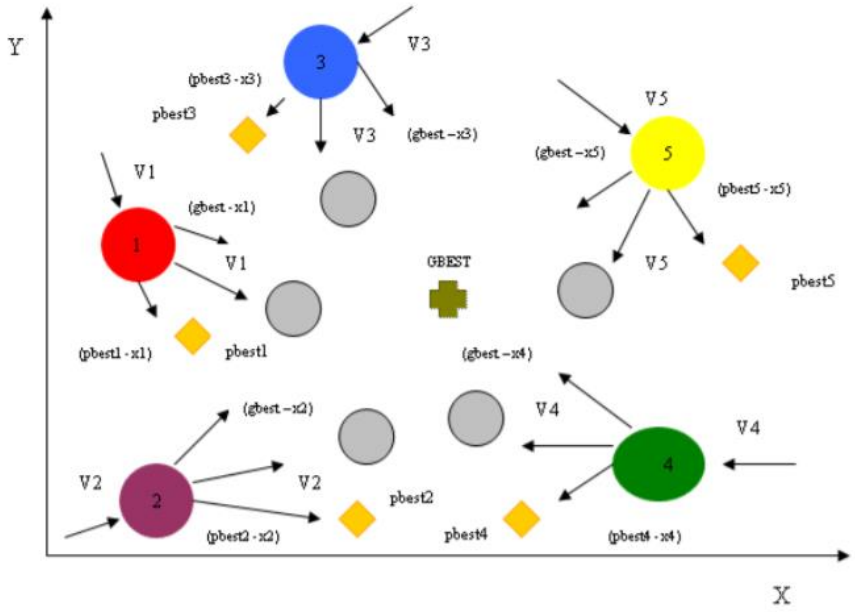


Figura 40 – Movimentação das partículas no espaço de busca (x,y). Fonte: Maia 2009

- $pbest_i$ = melhor posição da partícula;
- $gbest$= melhor posição encontrada pelo bando;

($pbest_i - x_i$) = distância entre a melhor posição da partícula e a sua posição atual;

($gbest_i - x_i$) = distância entre a melhor posição das partículas no bando e a sua posição atual;

= posições atuais das partículas;

= próxima posição de cada partícula.

Existem também meta-heurística hibridas, quando utiliza a combinação de duas ou mais meta-heurísticas, desta forma, combinando as diferentes características de dois ou mais métodos para se formar uma meta-heurística mais eficiente. Neste tipo de heurística não existe nenhuma restrição para se combinar as meta-heurísticas.

As combinações mais comuns encontradas na literatura são:

• Busca Tabu + Simulated Annealing;

• Algoritmo Genético + Busca Tabu;

Assim, é possível combinar as diferentes características dos métodos, de modo a criar uma meta-heurística mais eficiente.

Capítulo 5 – Experiências computacionais

5. Experiências computacionais

Neste capítulo serão apresentados alguns métodos exatos, neste caso vamos usar uma formulação discreta (QAP) e uma formulação continua (MIP) e também métodos aproximados, nomeadamente a heurística PSO, para solucionar problemas de *layout*. Uma vez que pare se solucionar este tipo de formulação exige muito tempo, vamos usar o programa "*A Mathematical Programming Language*" (AMPL), que vai ajudar a encontrar a solução muito mais rapidamente usando também o Solver comercial "G*urobi*" para solucionar o nosso problema. A implementação da heurística será feita em MatLab.

5.1. "*A Mathematical Programming Language*"

O AMPL ("Algebraic Modeling Programming Language ") é um "software" comercial, sendo possível obter uma versão de demonstração limitada (máximo de 300 variáveis e 300 restrições). A partir da sua página na "Internet", cujo endereço é http://www.ampl.com, pode ser obtida a versão limitada ("Student Edition").

Este *software* consiste numa linguagem de modelação algébrica, própria para escrever os modelos de problemas de otimização de larga escala e é usado em conjunto com um solver. O AMPL apenas permite escrever a formulação matemática do problema; o que permite solucionar o problema é o solver. O AMPL pode ser usado com vários *solvers*, tais como o *Gurobi*, *Cplex*, *Xpress*, *Knitro*, *Minos*, etc., a escolher consoante o tipo de problema de otimização que se pretende resolver.

Assim, uma vez que o nosso modelo é quadrático, vamos usar o solucionador *Gurobi* que serve para solucionar modelos inteiros mistos, lineares e quadráticos.

5.2. Problemas com Instalações de Áreas Iguais

A formulação QAP tem em consideração o fluxo de materiais entre pares de instalações, e a distância entre si. Normalmente, essa distância é medida entre o centro de cada par de instalações, utilizando-se uma métrica retilínea. Outro fator a ter em conta é o custo do transporte por unidade de distância e por peça. A resolução do problema passa por se atribuir a cada instalação um e um só local de modo a minimizar o custo do transporte.

Parâmetros:

$$n = Total\ de\ Equipamentos\ e\ Locais;$$

$$f_{ik} = Custo\ de\ fluxo\ do\ equipamento\ i\ para\ equipamento\ k;$$

$$d_{jl} = Distância\ do\ local\ j\ para\ o\ local\ l;$$

Variáveis:

$$x_{ij}=\begin{cases}1 & se\ o\ equipamento\ i\ está\ alocado\ ao\ local\ j\\ 0 & Outras\end{cases}$$

Função Objetivo:

$$min\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n}\sum_{l=1}^{n}f_{ik}d_{jl}x_{ij}x_{kl}$$

Restrições:

$$\sum_{j}^{n}x_{ij}=1 \qquad \forall_i$$

$$\sum_{i}^{n}x_{ij}=1 \qquad \forall_j$$

$$x_{ij}=\text{é } binario$$

Vamos exemplificar como se resolveria uma pequena instância deste problema com esta formulação. Consideremos uma situação em que temos três equipamentos para alocar em três possíveis locais A,B e C. Considere-se que o problema tem como parâmetros conhecidos de fluxo e distâncias as matrizes seguintes.

Matriz de fluxo entre equipamentos "i" e equipamento "k":

	k			
i		1	2	3
	1	0	5	8
	2	5	0	3
	3	8	3	0

Matriz de distância entre o local "j" e o local "l":

	l			
j		A	B	C
	A	0	4	7
	B	4	0	9
	C	7	9	0

Como podemos ver na figura que se segue, temos os respetivos locais (A,B,C) e as respetivas distâncias entre os mesmos.

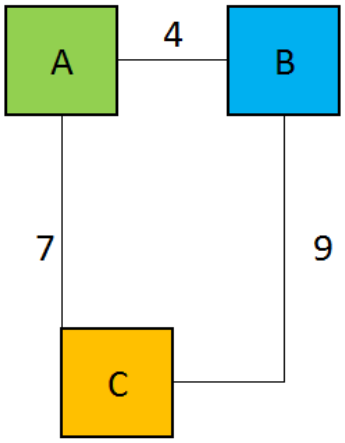


Figura 41 - Deslocação entre os locais (A,B,C) Fonte: Elaboração própria

Após termos todos os parâmetros, podemos escrever a função objetivo e as restrições do problema em concreto. Assim sendo a função objetivo representa-se da seguinte maneira.

$$Min = 0 * 4\, x_{A1}x_{B1} + 5 * 4x_{A1}x_{B2} + 8 * 4x_{A1}x_{B3} + 0 * 7x_{A1}x_{C1} + 5 * 7x_{A1}x_{C2} + 8 * 7x_{A1}x_{C3} + 5 * 4x_{A2}x_{B1} + 0 * 4x_{A2}x_{B2} + 3 * 4x_{A2}x_{B3} + 5 * 7x_{A2}x_{C1} + 0 * 7x_{A2}x_{C2} + 3 * 7x_{A2}x_{C3} + 8 * 4x_{A3}x_{B1} + 3 * 4x_{A3}x_{B2} + 0 * 4x_{A3}x_{B3} + 8 * 7x_{A3}x_{C1} + 3 * 7x_{A3}x_{C2} + 0 * 7x_{A3}x_{C3} + 0 * 4x_{B1}x_{A1} + 5 * 4x_{B1}x_{A2} + 8 * 4x_{B1}x_{A3} + 0 * 9x_{B1}x_{C1} + 5 * 9x_{B1}x_{C2} + 8 * 9x_{B1}x_{C3} + 5 * 4x_{B2}x_{A1} + 0 * 4x_{B2}x_{A2} + 3 * 4x_{B2}x_{A3} + 5 * 9x_{B2}x_{C1} + 0 * 9x_{B2}x_{C2} + 3 * 9x_{B2}x_{C3} + 8 * 4x_{B3}x_{A1} + 3 + 4x_{B3}x_{A2} + 0 * 4x_{B3}x_{A3} + 8 * 9x_{B3}x_{C1} + 3 * 9x_{B3}x_{C2} + 0 * 9x_{B3}x_{C3} + 0 * 7x_{C1}x_{A1} + 5 * 7x_{C1}x_{A2} + 8 * 7x_{C1}x_{A3} + 0 * 9x_{C1}x_{B1} + 5 * 9x_{C1}x_{B2} + 8 * 9x_{C1}x_{B3} + 5 * 7x_{C2}x_{A1} + 0 * 7x_{C2}x_{A2} + 3 * 7x_{C2}x_{A3} + 5 * 9x_{C2}x_{B1} + 0 * 9x_{C2}x_{B2} + 3 * 9x_{C2}x_{B3} + 8 * 7x_{C3}x_{A1} + 3 * 7x_{C3}x_{A2} + 0 * 7x_{C3}x_{A3} + 8 * 9x_{C3}x_{B1} + 3 * 9x_{C3}x_{B2} + 0 * 9x_{C3}x_{B3}$$

Sujeito a:

$$\sum_{j}^{n} x_{ij} = 1 \qquad \forall_i = 1,2,3$$

$$\sum_{i}^{n} x_{ij} = 1 \qquad \forall_j = A, B, C$$

$$x_{ij} = 0 \text{ ou } 1 \qquad \forall_{i,j}$$

Para a utilização do solucionador "*gurobi*", construímos um ficheiro de texto com o código do modelo implementado em AMPL, para que o problema possa ser resolvido. Como podemos verificar no código temos os seguintes atributos:

Lista de Equipamentos: set Equipamentos;

Lista de Locais Possíveis: set Locais;

Fluxo de Equipamentos de i para k*x: param Fluxo{i in Equipamentos, k in Equipamentos};

Distância de Locais j para l: param Distancia {j in Locais, l in Locais};

Após a definição de cada parâmetro de utilização, definimos a variável de decisão, a função objetivo e por último as restrições.

Variável de decisão:

```
var  x{j in Locais, i in Equipamentos} binary;
```

Função Objetivo:

```
minimize f_obj:
(sum {i in Equipamentos}
  (sum {j in Locais}
     (sum {k in Equipamentos: i<>k}
        (sum {l in Locais: l<>j}
  Fluxo[i,k]*Distancia[j,l]*x[j,i]*x[l,k]
          )
        )
   )
);
```

Restrições:

Restriçao1: em cada local tem que ser instalado só 1 equipamento

```
subject to  equip {j in Locais} :
(sum {i in Equipamentos}((x[j,i])))== 1 ;
```

Restrição2: cada equipamento tem que ser instalado em só 1 local

```
subject to  local {i in Equipamentos} :
(sum {j in Locais}((x[j,i])))== 1 ;
```

Uma vez que já temos a criação da estrutura do modelo para o problema, vamos de seguida dar os valores dos respetivos parâmetros para as instância deste problema..

```
data;
#lista de Equipamentos
set Equipamentos:= 1 2 3;

# Lista de Locais
set Locais:= A B C;

#Fluxo de Equipamentos de i para k*x
param Fluxo:     1  2  3 :=
                 1  .  5  8
                 2  5  .  3
                 3  8  3  .;

#Distancia de Locais j para l
param Distancia:     A  B  C :=
                     A  .  4  7
                     B  4  .  9
                     C  7  9  .;

# Numero de Equipamentos
#param N:=3;
```

Por último utilizamos o “gurobi” para encontrar a solução deste problema e solicitamos ao programa que mostre o valor da função objetivo e variáveis “x” na solução final

```
option solver gurobi;
solve;
display f_obj;
display x;
```

No seguimento do programa, é criada uma distribuição aleatória e a partir desse passo encontrar soluções melhores com respeito à minimização da função objetivo (FO). O processo inicia-se com uma solução inicial gerada aleatoriamente, que neste exemplo correspondeu a um valor da FO de196. Após esta iteração o programa gera novas soluções que compara com as anteriores, no sentido de caminhar na direção da solução ótima, usando o método de *Branch-and-Bound* que foi descrito no capítulo 4.1.

Uma vez que o programa detetou 3 vértices, ele vai de vértice a vértice até obter um valor inferior, no qual detetou o valor mínimo de 188.

O relatório de execução do solver *Gurobi* ao resolver esta instância encontra-se no anexo 2.

Apos 46 iterações que decorreram em 0,02 segundos conseguimos obter um valor mínimo de 188 para a função objetivo e onde conseguimos alocar no Local “A” o equipamento “1”, no Local “B” o Equipamento “3” e no Local “C” o Equipamento “2”. Podemos visualizar os seguintes resultados num ficheiro “txt” no qual tem a seguinte solução:

```
f_obj = 188
x :=
A 1   1
A 2   0
A 3   0
B 1   0
B 2   0
B 3   1
C 1   0
C 2   1
C 3   0
```

De salientar que este programa foi corrido por um computador com as seguintes características:

- ✓ Processador: Intel® Core™ i7 6700HQ Processor;
- ✓ Sistema Operativo: Windows 10 Home;
- ✓ Chipset: Intel® HM170 Chipset e Intel® HD Graphic 530;
- ✓ Memória RAM: DDR4 2133 MHz SDRAM, 16 GB;
- ✓ Gráfica: NVIDIA® GeForce® GTX 960M com 4GB GDDR5 VRAM;
- ✓ Armazenamento: 1TB HDD 7200 RPM e 256GB SSD

O resultado desta formulação é possível ser visualizado na figura que se segue, onde temos, o quadrado de verde é o local “A” onde foi alocado o equipamento “1” e que de seguida desloca para o local “B” onde tem uma distância de 4m e um fluxo de 8, no seguimento vai para o local “B” onde se encontra o quadrado a azul e onde foi colocado o equipamento “3”, depois desloca-se mais uma vez e segue para o local “C” onde foi colocado o equipamento “2” e onde tem que percorrer uma distância de 9m e com uma intensidade de “3”. Por último voltamos para o local “A” onde tem que se percorrer uma distância de 7m e com um fluxo de 5.

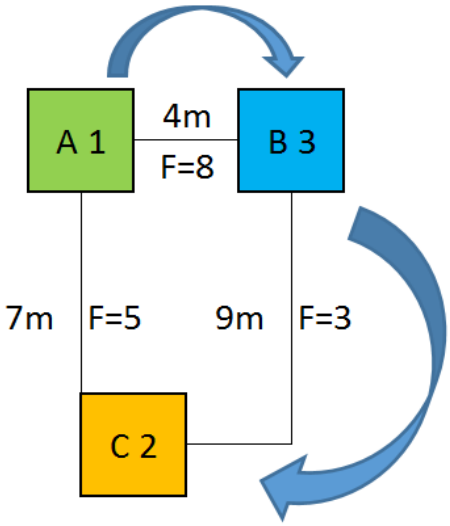


Figura 42 - Resolução de instalações de áreas iguais. Fonte: Elaboração Própria

Uma vez que obtivemos as soluções de acordo com o previsto, tentamos implementar num caso real de uma empresa de têxtil.

Tal processo não foi possível uma vez que a empresa apresenta muitos equipamentos e que a versão do (AMPL) utilizada possuía restrições a nível de memória e de número máximo de iterações, o que tornou impossível a resolução do problema. Desse modo decidimos aplicar uma meta-heurística, essa meta-heurística vai ser tratada mais à frente no capítulo 5.4.

5.3. Problemas com Instalações de Áreas Diferentes

Os equipamentos de áreas diferentes são representados na figura a seguir, no qual também apresenta o centroide de cada uma deles.

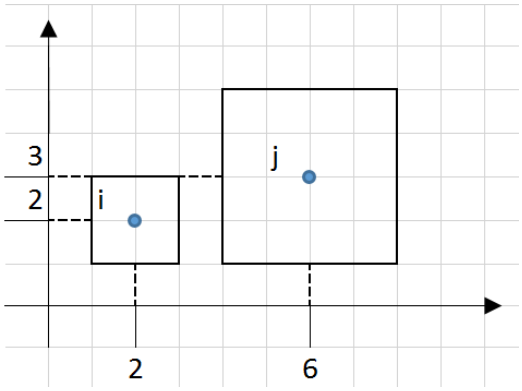


Figura 43 – Exercício prático. Fonte: Elaboração própria

Já o fluxo de equipamento é dado pela seguinte matriz.

	j		
i		1	2
	1	0	5
	2	6	0

Na resolução deste exemplo, colocamos que a matéria de saída do equipamento "i" seja no centro da máquina e que a matéria de entrada da máquina "j" seja na mesma no centro da máquina, assim podemos dizer que o centro de cada equipamento é o ponto de entrada e saída de matéria.

Assim sendo a função objetivo deste exemplo é apresentada em seguida.

$$Min \sum_{i=1}^{n} \sum_{j=1}^{n} f_{ij}(\left|x_j^I - x_i^o\right| + \left|y_j^I - y_i^o\right|)$$

Assim sendo na equação que se segue ilustramos uma resolução para este exercício, resolução essa, que nos dá o mínimo de custo que é o que nos é pretendido. Este exercício uma vez que tem só duas entidades a sua resolução é muito simples, contudo ao aumentarmos o número de equipamentos teremos sempre que ver os pontos de entrada e saída de matérias de todos eles assim como fazer todos as iterações entre eles e as suas relações.

$$Min \sum_{i=1}^{n} \sum_{j=1}^{n} f_{ij}(\left|x_j^I - x_i^o\right| + \left|y_j^I - y_i^o\right|) = 5(|6-2| + |3-2|) + 6(|2-6| + |2-3|) = 55$$

Este modelo não foi possível ser implementado, uma vez que a versão do (AMPL) utilizada possuía restrições a nível de memória e de número máximo de iterações, o que tornou impossível a resolução do problema.

5.4. Meta-heurística PSO para o problema de *layout*

Nesta secção, vamos utilizar uma heurística do tipo *Particle Swarm* aplicada ao *Facility Layout Design Problem.* Esta heurística, já descrita na seção 4.3.5, foi implementada em *Matlab*. O código foi baseado na versão de (S. Mostapha Kalami Heris). Na implementação deste modelo, foi necessário a definição de vários parâmetros do processo, como por exemplo, alturas, larguras, espaço de transporte e localização de entrada e saída de matérias.

No seguinte código temos os seguintes parâmetros:

w – Largura de cada equipamento;

h – Comprimento de cada equipamento;

$delta$ – Espaço para transporte;

rin – Entrada de material no equipamento;

$rout$ – Saída de material no equipamento;

a – Matriz de troca de matéria;

W – Largura total disponível do local de implementação;

H – Comprimento total disponível do local de implementação.

As duas coordenadas "rin" e "$rout$", vão nos dar a localização de onde a matérias entra nos equipamentos e de onde ela sai, com isso conseguimos que a distância de percurso entre equipamentos seja reduzida ao máximo e que exista menos custos de transporte.

Assim sendo temos o seguinte código de *Matlab - CreateModel*.

```
    % Tamanho do Blocos
w=[23 24 12 24 20 11 14 18];    % Largura
h=[25 25 12 25 25 17 22 12];    % Altura

delta=[2 1 4 1 3 1 4 1];    % Intervalo

rin= [0.17 0.70 0.73 0.27 0.04 0.09 0.42 0.69];%
Local de Entrada de Materia
rout= [0.31 0.95 0.03 0.43 0.38 0.76 0.79 0.18];% Local de Saída de Materia
```

Todos estes campos são inseridos pelo utilizador de forma dar a conhecer as características da empresa e do seu *layout*. De seguida são também necessários os seguintes parâmetros.

A matriz "A" no qual é o parâmetro que nos define a troca de matérias entre as respetivas máquinas. Neste exemplo utilizamos a seguinte matriz.

```
A=[     0   50   45   20    0   19   46   15
       28    0   13   15   24   27   25   48
       13   28    0    0   31   12    0   49
        0   14   20    0   26   47   41   33
       47   49   42   33    0   48   25   12
       16   10   27   32   19    0   19    0
       43   41   47   15   15   30    0   14
       32    0   17   44   17   23   13    0 ];
```

A área de instalação é dada por “W” que tem o valor de 100m de comprimento e por “H” com um valor de 80m de largura.

A função objetivo será dada pela soma de três parcelas

$$\text{Min z= } \alpha + \beta + \gamma$$

Onde α representa o espaço percorrido para executar a sequência de tarefas, β o custo da área não utilizada do espaço disponível, e γ é uma grandeza artificial que tem como objetivo impedir a sobreposição de diferentes máquinas.

Analisemos então cada uma das diferentes parcelas utilizadas nesta implementação.

$$\alpha = \sum_{i,j} a_{ij} * d_{ij}$$

onde a_{ij} é o fluxo de matéria do local “i” para “j” e d_{ij} é a distância entre o local “i” e “j”.

$$\beta = \varphi \left(1 - \frac{M_a}{C_a}\right)$$

onde M_a representa a soma das áreas ocupada por todas as máquinas (incluindo a margem para transporte) que é necessário colocar no *layout*, C_a representa a área do menor retângulo que inscreve todas as máquinas do *layout* na solução calculada, e φ é um fator de escala, que poderá representar o peso/custo que o desaproveitamento de espaço tem para a empresa em questão.

Por fim, $\gamma = \tau * v$, onde τ é uma constante a ser definida pelo utilizador que tem diversas ordens de grandeza acima das ordens de grandeza de α e β, e τ toma o valor 0, se nenhuma das máquinas se sobrepõem no *layout* e toma um valor positivo caso contrário.

Outro dos parâmetros que temos de delimitar no código, é o número máximo de iterações e o tamanho da população do *Swarm*. No exemplo em análise, utilizamos 500 iterações e uma população de 50 elementos o que permite um compromisso entre rapidez e qualidade da solução, já que o aumento quer do número de iterações quer da dimensão da população não conduziu a melhorias significativas. Obviamente, novos problemas deverão ser testados com novas parametrizações para se aferir das possíveis melhorias conseguidas.

No caso do código aplicado, consideramos os seguintes parâmetros.

```
w=1.0;          % Peso da Inercia
wdamp=0.99;  % Relação de amortecimento com o peso da inercia
c1=0.7;         % Coeficiente individual de aprendizagem
c2=1.5;         % Coeficiente global de aprendizagem
```

A primeira iteração parte de uma solução aleatória, que neste problema gerou um custo inicial de 8881086212 e que, no fim das 500 iterações, foi reduzido para 64396 valor correspondente ao “Melhor Custo” encontrado.

Esta implementação, devolve também outros parâmetros considerados importantes como a área de trabalho, área das máquinas, área não usada, o rácio de área não usada, as coordenadas totais da área de trabalho e as coordenadas de cada departamento. Neste exemplo, obtiveram-se os seguintes resultados:

Área total de trabalho – 5029m^2;

Área só das máquinas – 3130m^2;

Área não usada – 1899m^2;

Rácio de Área não usada com Área usada – 37,8%

Dada a primeira iteração ser determinada de forma aleatória, diferentes execuções poderão resultar em diferentes resultados, dando assim opções de escolha ao decisor para escolher o melhor layout possível.

Dessa forma, corremos 50 resoluções e cada resolução teve como limite as 500 iterações. Todos os resultados encontram-se no anexo 4.

Observamos que nessas 50 resoluções, os parâmetros podem não ser simultaneamente óptimos. Por exemplo, podemos ter o melhor parâmetro “Melhor Custo” e um péssimo “Rácio de Área não Usada”. Assim sendo, registaram-se os valores mínimos, máximos, médias e desvios padrões para todas estas relações. conforme informação da tabela que se segue.

Tabela 5 - Tratamento de resultados. Fonte: Elaboração própria

	Função Objetivo	Área Envolvente	Área não Usada	Rácio de Área não Usada
Mínimo	54220	4235	1105	26,10%
Máximo	74152	6233	3103	49,80%
Média	65694,24	5078,24	1948,42	37,88%
Desvio Padrão	4496,05	449,03	449,26	5,38%

Podemos verificar que o resultado obtido anteriormente tem o “Melhor Custo” abaixo da média, o que podemos dizer que tem um bom resultado, mas não o ótimo.

Neste caso o melhor resultado foi a resolução Nº18 no qual podemos verificar o seu *layout* de figura que se segue como os respetivos parâmetros.

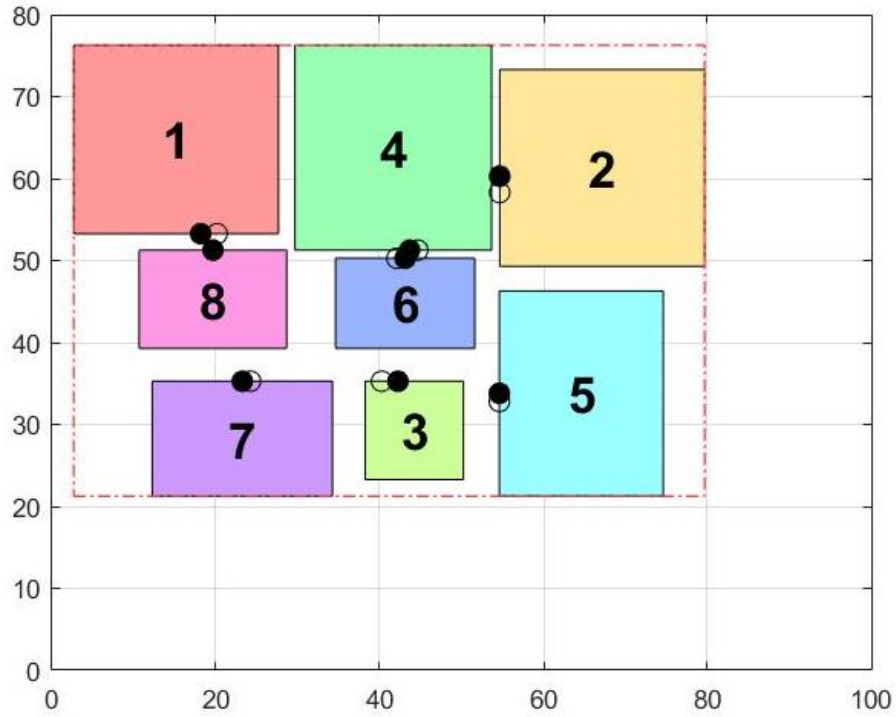


Figura 44 – Melhor solução do layout do exercício. Fonte: Elaboração própria

Tabela 6 – Resultados da resolução Nº18 (Anexo 4). Elaboração própria

Melhor Custo	XMIN	YMIN	XMAX	YMAX	Área envolvente	Área das Máquinas	Área não Usada	Rácio de Área não Usada
54220	**2,64**	**21,26**	**79,64**	**76,26**	**4235**	**3130**	**1105**	**26,1%**

5.4.1. Implementação num caso prático

Neste exemplo foi criada uma empresa fictícia com 8 equipamentos e que foram colocados aleatoriamente dentro da empresa com uma determinada distância de comprimento e largura e entrada e saída de matéria do equipamento.

Apos a resolução do problema de entrada e saída de matérias, apresentamos mais abaixo os respetivos equipamentos que foram colocados na empresa. Todos estes equipamentos são fictícios uma vez que as medidas dadas são aleatórias.

- ✓ 1º Equipamento:
    - Largura = 10m;
    - Altura = 7m;
    - Entrada de Matéria:
        - Xin= -5;
        - Yin= 0;
    - Saída de Matéria:
        - Xout = 5;
        - Yout = 0.

Figura 45 - Representação gráfica do 1ºEquipamento. Fonte: Elaboração própria

- ✓ 2º Equipamento:
    - Largura = 7m;
    - Altura = 12m;
    - Entrada de Matéria:
        - Xin= 0;
        - Yin= 6;
    - Saída de Matéria:
        - Xout = 0;
        - Yout = -6.

Figura 46 - Representação gráfica do 2ºEquipamento. Fonte: Elaboração própria

✓ 3º Equipamento:

- Largura = 10m;
- Altura = 10m;
- Entrada de Matéria:
  - Xin= -5;
  - Yin= -5;
- Saída de Matéria:
  - Xout = 5;
  - Yout = 5.

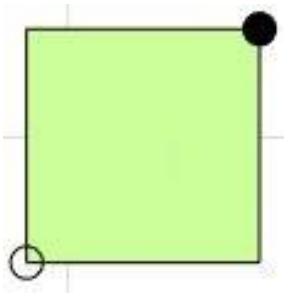


Figura 47 - Representação gráfica do 3ºEquipamento. Fonte: Elaboração própria

✓ 4º Equipamento:

- Largura = 20m;
- Altura = 10m;
- Entrada de Matéria:
  - Xin= 10;
  - Yin= -5;
- Saída de Matéria:
  - Xout = -10;
  - Yout = 5.

Figura 48 - Representação gráfica do 4ºEquipamento. Fonte: Elaboração própria

✓ 5º Equipamento:

- Largura = 10m;
- Altura = 10m;
- Entrada de Matéria:
  - Xin= 5;
  - Yin= 5;
- Saída de Matéria:
  - Xout = -5;
  - Yout = -5.

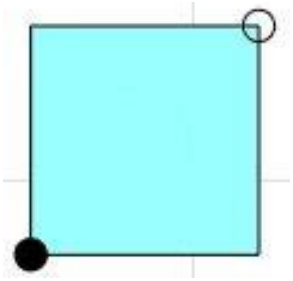


Figura 49 - Representação gráfica do 5ºEquipamento. Fonte: Elaboração própria

✓ 6º Equipamento:

- Largura = 15m;
- Altura = 10m;
- Entrada de Matéria:
  - Xin= -7,5;
  - Yin= 0;
- Saída de Matéria:
  - Xout = 7,5;
  - Yout = 0.

Figura 50 - Representação gráfica do 6ºEquipamento. Fonte: Elaboração própria

✓ 7º Equipamento:

- Largura = 15m;
- Altura = 15m;
- Entrada de Matéria:
  - Xin= 0;
  - Yin= -7,5;
- Saída de Matéria:
  - Xout = 0;
  - Yout = 7,5.

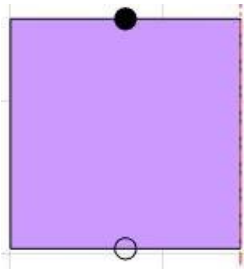


Figura 51 - Representação gráfica do 7ºEquipamento. Fonte: Elaboração própria

✓ 8º Equipamento:

- Largura = 40m;
- Altura = 20m;
- Entrada de Matéria:
  - Xin= -20;
  - Yin= 0;
- Saída de Matéria:
  - Xout = 20;
  - Yout = 0.

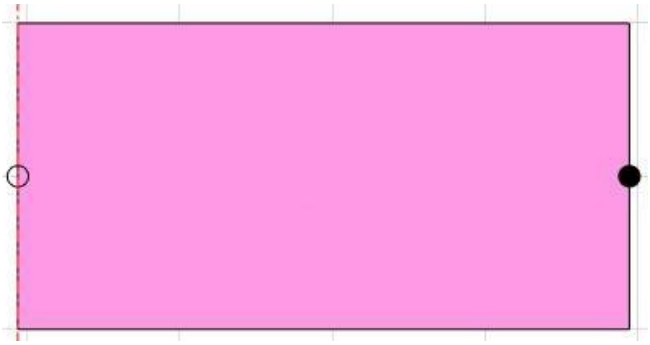


Figura 52 - Representação gráfica do 8ºEquipamento. Fonte: Elaboração própria

Com todas as especificações dos equipamentos já escolhidas, é necessário refletir essas e outras opções no nosso modelo. Criamos então a função/*script* “CreateModel”, que iniciará as variáveis necessárias para aplicar a rotina de PSO. Uma breve explicação é apresentada nas linhas seguintes.

```
function model=CreateModel()

% Definição das Máquinas
w=[10 07 10 20 10 15 15 40]; % Largura
h=[07 12 10 10 10 10 15 20]; % Altura
```

Para cada uma das máquinas consideramos o valor de origem do “delta”, uma vez que o valor do “delta” é o espaço necessário para trabalhar à volta da máquina, a seguinte margem de segurança.

```
delta=[2 1 4 1 3 1 4 1];    % Intervalo
```

Nesta rotina, cada local de carga de cada máquina é definido por apenas um número real (coordenada r_in), acontecendo o mesmo para o local de descarga com a coordenada r_out. Esta codificação obedece à necessidade de se colocar os locais de entrada e saída da mercadoria na aresta mais próxima com respeito ao retângulo que inscreve a máquina.

A função codificar_ins.m apresentada em anexo, efetua essa codificação, transformando as coordenadas retangulares usuais (com respeito ao referencial cuja origem é o ponto médio da máquina), na codificação utilizada por esta implementação do PSO. Conforme já visto anteriormente, as coordenadas de carga e descarga de cada uma das máquinas foram arbitradas em:

```
xin=[-5,0,-5,10,5,-7.5,0,-20] %Local de entrada (x)
yin=[0,6,-5,-5,5,0,-7.5,0] %Local de entrada (y)
xout=[5,0,5,-10,-5,7.5,0,20] %Local de saída (x)
yout=[0,-6,5,5,-5,0,7.5,0] %Local de saída (y)
```

Definimos ainda a matriz “A”, cujas entradas A(i,j) contêm os custos do fluxo entre quaisquer duas máquinas i,j.

```
A=[  0   0   5   0  10   0   0   0
     0   0   0   0   0   0   0   0
     0   0   0   5   0   0   0   0
     0   0   0   0   0   0   0  10
     0   0   0   0   0   5   0   0
     0   0   0   0   0   0   0  20
     0   0   0   0   0   0   0   0
     0  17   0   0   0   0  40   0 ];
```

Conforme pode ser visto na figura seguinte, e confirmado na matriz supra, assumimos que não existe retrocesso entre máquinas.

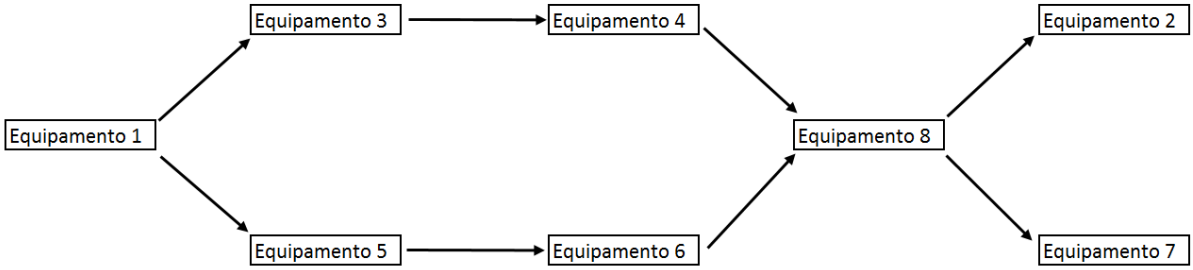


Figura 53 - Representação fictícia dos fluxos de matéria dentro da empresa. Fonte: Elaboração própria

Por fim, outros parâmetros a ter em conta dizem respeito à área de implementação disponível, definida através do comprimento W e da largura H, que neste caso foram definidos como 60m e 80m, respetivamente. Definimos também o parâmetro "phi", correspondente à penalização pela percentagem de área livre (i.e, não ocupada pelas máquinas), com respeito ao rectângulo que inscreve a área utilizada numa dada solução. A este valor, que podemos pensar como sendo o factor de penalização associado a um índice de dispersão de uma solução, atribui-se o peso "semi-aleatório" de 8160.

Mantivemos os parâmetros que delimitam o nosso programa, como o número máximo de iterações e a população do Particle Swarm. Neste exemplo, modificamos porém os parâmetros (constantes) "c1" e "c2", com respeito ao exemplo inicial, para o valor de 2 uma vez que Eberhart e Kennedy (1995) referem que as constantes c1/c2 tem que estar compreendidas entre [0;2].

```
MaxIt=500;      % Numero Máximo de Iterações

nPop=50;        % Tamanho da População (Swarm Size)

w=1.0;          % Peso da Inercia
wdamp=0.99;     % Relação de amortecimento com o peso da inercia
c1=2;           % Coeficiente individual de aprendizagem
c2=2;           % Coeficiente global de aprendizagem
```

Com todos os parâmetros inseridos fomos correr o programa num computador com as seguintes características e ver os seu comportamento em relação ao tempo de realização e revelar os respetivos dados.

Características do computador:

- ✓ Processador: Intel® Core™ i7 6700HQ Processor;
- ✓ Sistema Operativo: Windows 10 Home;
- ✓ Chipset: Intel® HM170 Chipset e Intel® HD Graphic 530;
- ✓ Memória RAM: DDR4 2133 MHz SDRAM, 16 GB;
- ✓ Gráfica: NVIDIA® GeForce® GTX 960M com 4GB GDDR5 VRAM;
- ✓ Armazenamento: 1TB HDD 7200 RPM e 256GB SSD

Uma vez mais, o programa usa sempre a primeira iterações de forma "*rand"*, o que faz com que muitas das vezes não apresente a melhor solução. Assim da mesma forma que fizemos anteriormente, fomos fazer 50 resoluções e cada resolução teve 500 iterações. Todos os resultados encontram-se no anexo 5.

Voltamos a verificar que nessas 50 resoluções os parâmetros não estão relacionados. Por exemplo podemos ter o melhor a parâmetro no "Melhor Custo" e ter um péssimo "Rácio de Área não Usada", assim sendo fomos verificar os valores mínimos, máximos, médias e desvios padrões para todas estas relações. Isso tudo é possível ser visto na tabela que se segue.

Tabela 7 - Tratamento de resultados (Empresa Fictícia). Fonte: Elaboração própria

	Função Objetivo	Área Envolvente	Área não Usada	Rácio de Área não Usada
Mínimo	4453	2420	691	28,5%
Máximo	7413	3380	1651	48,8%
Média	5600,88	2830,76	1101,68	38,62%
Desvio Padrão	626,82	194,29	194,36	4,05%

Na resolução onde o "Melhor Custo", que é a função objetivo, é a melhor, é no caso da resolução Nº30, que por sua vez, não é onde a área envolvente e a área não usada são as melhores. Já onde essas áreas são melhores é na resolução Nº48 onde o melhor custo tem o valor de 4978.

Assim sendo verificamos mais uma vez que este código não encontra a melhor solução de todas mas sim oferece um leque muito grandes de soluções viáveis no qual o decisor pode escolher a que melhor se adequa à empresa.

Neste caso se o decisor escolhe o *layout* pela sua forma de apresentação no chão de fábrica, ele poderia escolher a resolução Nº9 onde podemos verificar na figura, que o *layout* aparentemente se comporta melhor.

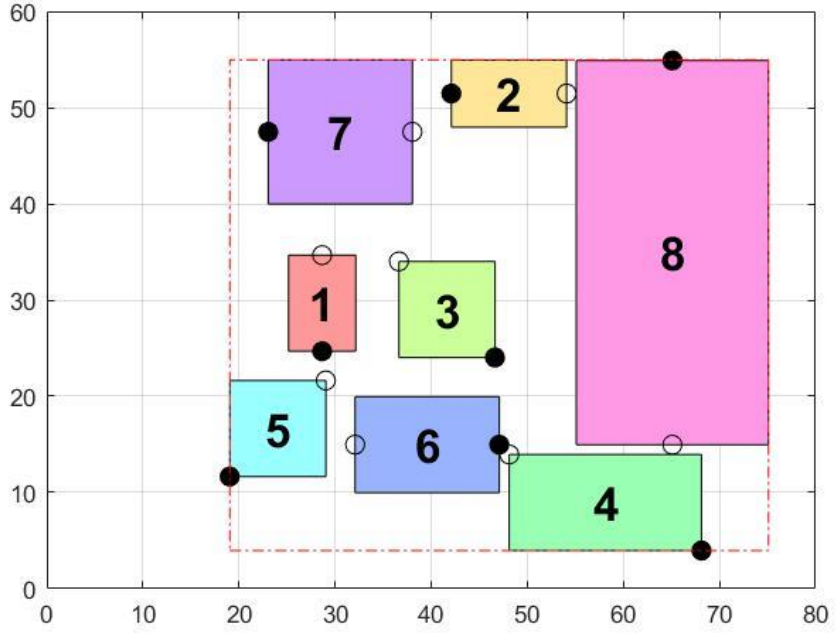


Figura 54 – Melhor solução a nível de layout do exercício. Fonte: Elaboração própria

Se o decisor escolher o *layout* onde o "Melhor Custo" é o melhor (4453), temos então a resolução Nº 30 com o seguinte *layout*.

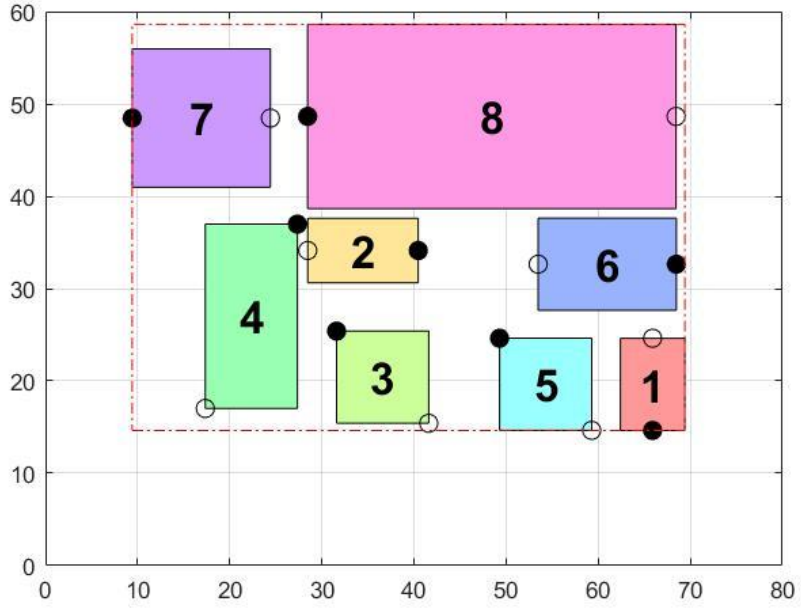


Figura 55 - Melhor solução a nível de Bestcost do exercício. Fonte: Elaboração própria

Neste *layout* temos os equipamentos distribuídos de forma diferente o que leva com que o "Melhor Custo" seja o melhor, mas contudo como foi referido anteriormente a área envolvente e a área não usada não é das melhores.

Mais uma vez se o decisor optar por ter a área de trabalho com melhor aproveitamento pode considerar o *layout* que se seja, no qual corresponde à resolução Nº 48.

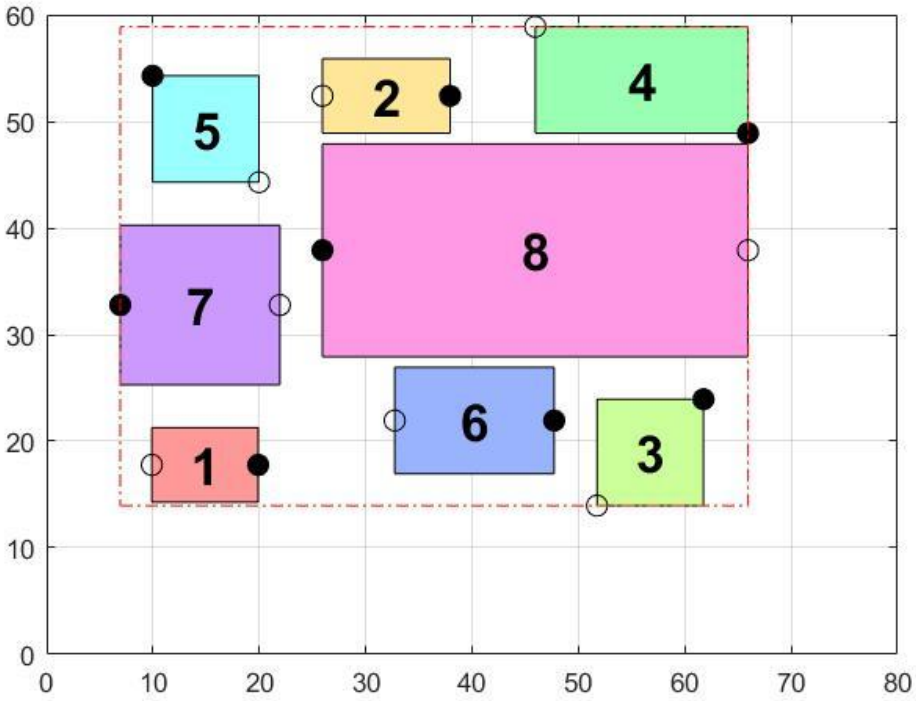


Figura 56 - Melhor solução a nível de área de trabalho. Fonte: Elaboração própria

Todas estas soluções são fiáveis e resolvem o problema de diminuir o custo, mas nenhuma delas é a perfeita. A escolha destes *layouts* vai depender de todos os parâmetros inerentes ao mesmo.

Capítulo 6 – Conclusões e Trabalho Futuro

6. Conclusões e Trabalho Futuro

Com o passar do tempo as empresas têm sido confrontadas com diversas dificuldades, no qual tentam dar resposta para que se possam manter e até mesmo ter um crescimento no mercado onde atuam.

A gestão industrial tem tido uma tarefa muito importante para a criação de sistemas eficientes e flexíveis, na tentativa de atingir a produtividade, satisfação dos funcionários e a atividade segura nas condições adversas que caracterizam o cenário competitivo atual.

A rivalidade entre as organizações exige a diminuição de custos operacionais para que as mesmas mantenham o seu ranking no mercado, sendo que isto, apenas se torna possível com estruturas bem planificadas capazes de rentabilizar recursos, espaços e eliminando ao máximo os desperdícios.

O planeamento de um *layout* tende a tornar mais claro e mais eficiente a ligação entre os equipamentos, mão-de-obra, materiais, áreas de movimentação, áreas de stock, áreas administrativas, resumindo, todos os componentes que contribuem para o bom funcionamento de uma organização.

Desta forma, parece-me interessante revelar que existem muitas formulações, formulações essas, que podem ser discretas, continuas, *fuzzy,* entre outras, e métodos, como por exemplo abordagens exatas e abordagens aproximadas.

Nesta tese foram estudadas, e aplicadas a casos concretos e fictícios de pequena dimensão, um modelo de afetação quadrático (QAP) para uma formulação discreta do problema de *layout*, um modelo de programação inteira misto (MIP) para uma formulação contínua deste problema, e uma meta-heurística de bando de partículas (PSO). Foi utilizado *software* específico da área da Investigação Operacional e Engenharia, nomeadamente a linguagem de modelação de problemas de otimização *AMPL*, o solver comercial de problemas de programação linear e quadrática *Gurobi*, e a linguagem de computação matemática para resolução de problemas de engenharia *MatLab*.

Assim, uma vez que o nosso tipo de problema é quadrático usamos o solucionador "*Gurobi*" para solucionar problemas contínuos e de inteiros mistos.

A formulação QAP tem em consideração o fluxo de materiais entre pares de instalações, e a distância entre si, outro fator a ter em conta é o custo do transporte por unidade de distância e por peça. A resolução do problema passa por se atribuir a cada instalação um e um só local de modo a minimizar o custo do transporte.

Após termos obtido soluções para pequenas instâncias do problema de *layout* com os modelos exatos QAP, tentamos implementá-los num caso real de uma empresa de têxtil.

Tal processo não foi possível uma vez que a empresa apresenta muitos equipamentos e que a versão do (AMPL) utilizada possuía restrições a nível de memória e de número máximo de iterações, o que tornou impossível a resolução do problema. Desse modo decidimos aplicar uma meta-heurística.

Assim, optámos por utilizar uma heurística do tipo *Particle Swarm Optimization* aplicada ao *Facility Layout Design Problem.* Esta heurística, foi implementada em *Matlab*. O código foi baseado na versão de (S. Mostapha Kalami Heris). Na implementação deste modelo, foi necessário a definição de vários parâmetros do processo, como por exemplo, alturas, larguras, espaço de transporte e localização de entrada e saída de materiais.

Uma vez que este programa determina primeira iteração de forma aleatória, o que faz com que muitas das vezes não apresente a melhor solução, dando assim, opções de escolha ao decisor para escolher o melhor *layout* possível.

Verificamos que nessas 50 resoluções observamos que os parâmetros não estão relacionados. Por exemplo podemos ter o melhor parâmetro “Melhor Custo” e um péssimo “Rácio de Área não Usada” assim sendo, fomos verificar os valores mínimos, máximos, médias e desvios padrões para todas estas relações.

Todos os métodos e abordagens utilizadas nesta tese revelam que os resultados obtidos não são sempre os perfeitos mas sim permitem informar melhor o decisor sobre as várias possibilidades para a planificação do *layout*.

As maiores dificuldades e condicionantes que encontramos neste desafio, foram a programação de métodos em que a versão de estudante nos dava bastantes limitações a nível de restrições, o que nos levou a procurar outras resoluções fiáveis para a continuidade deste projeto.

Como primeira proposta de trabalho futuro, será a procura de um modelo mais leve que nos dê a solução ótima, ou de uma outra heurística que nos dê uma solução melhor, uma vez que a heurística usada nos dá uma boa solução viável, mas que não tem garantia de ser a solução ótima. A escolha da solução a adotar será sempre uma opção do decisor. Desse modo, mesmo uma aproximação de *layout* ótimo irá melhorar a sua organização e reforçar a sua posição competitiva em mercados internos e externos.

REFERÊNCIAS BIBLIOGRÁFICAS

Afentakis, P., (1986). A model for layout design in FMS. In Flexible Manufacturinq ystems: Methods and Studies, edited by A. Kusiak (Amsterdam: Elsevier).

Aiello, G., & Enea, M. (2001). Fuzzy approach to the robust facility layout in uncertain production environments. International Journal of Production Research, 39 (18), 4089 – 4101.

Al-Hakim, L. (2000). On solving facility layout problems using genetic algorithms. International Journal of Production Research, 38 (11), 2573 – 2582.

Al-Hakim, L. A., (1991). Two Graph-Theoretic Procedures for an Improved Solution to the Facilities Layout Problem. International Journal of Production Research, 29 (8), pp 1701-1718.

Al-Hakim, L. A., (1992). A Modified Procedure for Converting a Dual Graph to a Block Layout. International Journal of Production Research, 30 (10), pp 2467-2476.

Arroyo, J. E. (2002). Heurísticas e Metaheurísticas para Optimização Combinatória Multiobjectivo. Tese de Doutoramento, Faculdade de Engenharia Eléctrica e de Computação, Universidade Estadual de Campinas, Campinas, São Paulo.

Armour e Buffa, (1963). A heuristic algorithm and simulation approach to relative location of facilities. Management Science, 9, pp 294-309.

Azadivar, F., & Wang, J. (2000). Facility layout optimization using simulation and genetic algorithms. International Journal of Production Research, 38 (17), 4369 – 4383.

A. H. Teller, and E. Teller, (1953). N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, ,“Equation of State Calculations by Fast Computing Machines,” The Journal of Chemical Physics, vol. 21, no. 6, p. 1087.

Balakrishnan, J., & Cheng, C. H. (1998). Dynamic layout algorithms: A stateof- the-art survey. Omega, 26 (4), 507 – 521.

Balakrishnan, J., Cheng, C. H., Conway, D. G., & Lau, C. M. (2003). A hybrid genetic algorithm for the dynamic plant layout problem. International Journal of Production Economics, 86 (2), 107 – 120.

Balakrishnan, Jaydeep; Cheng, Chun Hung, Multi-period planning and uncertainty issues in cellular manufacturing: A review and future directions. European Journal of Operational Research, 2007, Vol.177 (1), p.281-309.

Baykasoglu, A., & Gindy, N. N. Z. (2001). A simulated annealing algorithm for dynamic layout problem. Computers&Operations Research, 28 (14), 1403 – 1426.

Baykasoglu, A., Dereli, T., & Sabuncu, I. (2006). An ant colony algorithm for solving budget constrained and unconstrained dynamic facility layout problems. Omega, 34 (4), 385 – 396.

Benjaafar, S., Heragu, S. S., & Irani, S. A. (2002). Next generation factory layouts: Research challenges and recent progress. Interface, 32(6), 58–76.

Black, J. T. O projeto da fábrica com futuro. Bookman, 1998.

Braglia, M. (1996). Optimization of a simulated-annealing-based heuristic for single row machine layout problem by genetic algorithm. International Transactions in Operational Research, 3 (1), 37 – 49.

Braglia, M., Zanoni, S., & Zavanella, L. (2003). Layout design in dynamic environments: Strategies and quantitative indices. International Journal of Production Research, 41 (5), 995 – 1016.

Carravilla M. A. e Oliveira J. F. (2001), Programação Inteira Transparências de apoio à leccionação de aulas teóricas FEUP

Costa, Carine Rodrigues (2011). Condução de Experimentos Computacionais com Métodos Heurísticos. Tese de mestrado na Universidade Federal de Goiás.

Chen, C. W., & Sha, D. Y. (2005). Heuristic approach for solving the multiobjective facility layout problem. International Journal of Production Research, 43 (21), 4493 – 4507.

Chen, D. S., Wang, Q., & Chen, H. C. (2001). Linear sequencing for machine layouts by a modified simulated annealing. International Journal of Production Research, 39 (8), 1721 – 1732.

Cheng, R., & Gen, M. (1998). Loop layout design problem in flexible manufacturing systems using genetic algorithms. Computers & Industrial Engineering, 34 (1), 53 – 61.

Cheng, R., Gen, M., & Tosawa, T. (1996). Genetic algorithms for designing loop layout manufacturing systems. Computers & Industrial Engineering, 31 (3 – 4), 587 – 591.

Chiang, W. C., & Kouvelis, P. (1996). An improved tabu search heuristic for solving facility layout design problems. International Journal of Production Research, 34 (9), 2565 – 2585.

Chwif, L., Pereira Barretto, M. R., & Moscato, L. A. (1998). A solution to the facility layout problem using simulated annealing. Computers in Industry, 36 (1 – 2), 125 – 132.

Das, S. K. (1993). A facility layout method for flexible manufacturing systems. International Journal of Production Research, 31 (2), 279 – 297.

Deb, S. K., & Bhattacharyya, B. (2005). Fuzzy decision support systems for manufacturing facilities layout planning. Decision Support Systems, 40, 305 – 314.

Doblas, D. (2010) “Arranjo físico e o planejamento estratégico” [Consultado em 23/10/2016]

Dorigo, M.; Maniezzo, V.; Colorni, A, (1996). Ant System: Optimization by a Colony of Cooperating Agents. IEEE Transactions on Systems, Man and Cybernetics, Part B (Cybernetics), 26(1):29–41.

Drezner, Z. (1987). A heuristic procedure for the layout of a large number of facilities. International Journal of Management Science, 33 (7), 907 – 915.
Drira A, Pierreval H, Hajri-Gabouj S (2007) Facility layout problems: a survey. Annu Rev Control 31(2):255–267

Dunker, T., Radonsb, G., & Westkampera, E. (2005). Combining evolutionary computation and dynamic programming for solving a dynamic facility layout problem. European Journal of Operational Research, 165 (1), 55 – 69.

Dweiri, F. (1999). Fuzzy development of crisp activity relationship charts for facilities layout. Computers & Industrial Engineering, vol. 36, p. 1–16.

Dweiri, F., & Meier, F. A. (1996). Application of fuzzy decision-making in facilities layout planning. International Journal of Production Research, 34 (11), 3207 – 3225.

Eberhart, R. C. and Kennedy, J, 1995. A new optimizer using particle swarm theory. Proceedings of the sixth international symposium on micro machine and human science pp. 39-43. IEEE service center, Piscataway, NJ, Nagoya, Japan.

Eberhart CG, Maines JZ, Wasserman SA, 1996. Meiotic cell cycle requirement for a fly homologue of human Deleted in Azoospermia.

El-Baz, M. A. (2004). A genetic algorithm for facility layout problems of different manufacturing environments. Computers & Industrial Engineering, 47 (2 – 3), 233 – 246.

Evans, G. W., Wilhlem, M. R., & Karwowsky, W. (1987). A layout design heuristic employing the theory of fuzzy sets. International Journal of Production Research, 25, 1431 – 1450.

Francis, R. L.; White, J. A. (1974). Facility Layout and Location – An Analytical Approach. New Jersey: Prentice-Hall.

Francis, Richard L., McGinnis, Leon F. e White, John A., (1992). Facility layout and location: an analytical approach. 2nd edn, Englewood Cliffs, NJ: Prentice Hall, pp 27-171.

Fruggiero, F., Lambiase, A., & Negri, F. (2006). Design and optimization of a facility layout problem in virtual environment.. In Proceeding of ICAD 2006 (pp. 2206–).
Garey, M. R., & Johson, D. S. (1979). Computers and intractability - a guide to NP-completeness. San Francisco: W.H. Freeman and Company.

Gen, M., Ida, K., & Cheng, C. (1995). Multi row machine layout problem in fuzzy environment using genetic algorithms. Computers & Industrial Engineering, 29 (1 – 4), 519 – 523.

Goldberg, D. E, 1989. Genetics Algorithms in Search, Optimization, and Machine Learning. Addison-Wesley, Berkeley.

Golden, B. L, 1977. A Statistical Approach to the TSP. Networks, 7:209–225.

Grobelny, J. (1987a). The fuzzy approach to facility layout problems. Fuzzy Sets and Systems, 23, 175 – 190.

Grobelny, J. (1987b). On one possible 'fuzzy' approach to facility layout problems. International Journal of Production Research, 25, 1123 – 1141.

Gupta, Y. P., Gupta, M. C., Kumar, A., & Sundaram, C. (1996). A genetic algorithm-based approach to cell composition and layout design problems. International Journal of Production Research, 34 (2), 447 – 482.

Hamamoto, S., Yih, Y., & Salvendy, G. (1999). Development and validation of genetic algorithm-based facility layout-a case study in the pharmaceutical industry. International Journal of Production Research, 37, 749 – 768.

Harmonosky, C. M., & Tothero, G. K. (1992). A multi-factor plant layout methodology. International Journal of Production Research, 30 (8), 1773 – 1789.
Hassan, M. M. D., e Hogg, G. L., (1989). On Converting a Dual Graph into a Block Layout. International Journal of Production Research, 27 (7), pp 1149-1160.

Hassan, M. M. D., e Hogg, G. L., (1991). On Constructing a Block Layout by Graph Theory. International Journal of Production Research, 29 (6), pp 1263-1278.

Hassan, M. M. D., Hogg, G. L., e Smith, D. R., (1986). SHAPE: A Construction Algorithm for Area Placement Evaluation. International Journal of Production Research, 24, pp 1283-1295.

Heragu, S. S. e Kusiak, A., (1987). The Facility Layout Problem, European Journal of Operational Research, 53, pp 1-13.

Heragu, S. S., & Kusiak, A. (1990). Machine layout: An optimization and knowledge-based approach. International Journal of Production Research, 28, 615 – 635.

Heragu, S. S. (1997). Facilities design. Boston: BWS.

Hiller. F. S., Lieberman, G. J. (2005) Introduction to Operations Research. Eighth Edition. McGraw-Hill International Edition.

Seixas, I. C. V. (2013). Aplicação de Métodos Emergentes em Problemas de Escalonamento do Tipo Flexible Job Shop, tese de Mestrado no Instituto Politécnico de Bragança.

Islier, A. A. (1998). A genetic algorithm approach for multiple criteria facility layout design. International Journal of Production Research, 36 (6), 1549 – 1569.

Johnson, R. V. (1982). SPACECRAFT for multi-floor layout planning. Management Sciences, 28 (4), 407 – 417.

Jones, Gareth R.; GEORGE, Jennifer M. (2008). Administração Contemporânea. 4ª edição. São Paulo: McGraw-Hill.

Khalil, T. M., (1973). Facilities relative allocation technique (FRAT). International Journal of Production Research, 11 (2), pp 183-194.

Kennedy, J. (1998). The behavior of particles. In V. W. Porto, N. Saravanan, D. Waagen & A. E. Eiben (Eds.), Lecture notes in computer science. Evolutionary programming VII: proceedings of the 7-th annual conference on evolutionary programming (pp. 581–589). San Diego, CA. Berlin: Springer

Kim, C. B., Kim, S. S., & Bobbie, L. F. (1996). Assignment problems in singlerow and double-row machine layouts during slow and peak periods. Computers & Industrial Engineering, 30 (3), 411 – 422.

Kim, J. G., & Kim, Y. D. (1999). A branch and bound algorithm for locating input and output points of departments on the block layout. Journal of the operational research society, 50 (5), 517 – 525.

Kim, J. G., & Kim, Y. D. (2000). Layout planning for facilities with fixed shapes and input and output points. International Journal of Production Research, 38 (18), 4635 – 4653.

Kim, J. Y., & Kim, Y. D. (1995). Graph theoretic heuristics for unequal-sized facility layout problems. Omega, 23 (4), 391 – 401.

Kochhar, J. S. e Heragu, S. S., (1998). Facility layout design in a changing enviroment, International Journal of Production Research, 37 (11), pp 2429-2446.

Kochhar, J. S. e Heragu, S. S., (1998). MULTI-HOPE: a tool for multiple floor layout problems, International Journal of Production Research, 36 (12), pp 3421-3435.

Kochhar, J. S., Foster, B. L. e Heragu, S. S., (1996). Genetic Algorithm for Unequal area Facility Layout Problem. Rensselaer Polytechnic Institute, Troy, New York, Technical Report.

Koopmans, T. C., & Beckmann, M. (1957). Assignment problems and the location of economic activities. Econometrica, 25(1), 53–76.

Kouvelis, P., & Chiang, W. C. (1992). A simulated annealing procedure for the single row layout problems in flexible manufacturing systems. International Journal of Production Research, 30, 717 – 732.

Kouvelis, P., and KIM, M. W., 1992, Unidirectional loop network layout problem in automated manufacturing systems. Operations Research, 40 (3),533-550.
Lacksonen, T. A. (1997). Preprocessing for static and dynamic facility layout problems. International Journal of Production Research, 35 (4), 1095 – 1106.

Lacksonen, T. A., &Enscore, E. E. (1993). Quadratic assignment algorithms for the dynamic layout problem. International Journal of Production Research, 31, 503 – 517.

Lahmar, M. and S. Benjaafar, “Design of Plant Layouts for Expansion,” Working Paper, Department of Mechanical Engineering, University of Minnesota, Minneapolis, 2005.

Lee, G. C., & Kim, Y. D. (2000). Algorithms for adjusting shapes of departments in block layouts on the gird-based plane. Omega, 28 (1), 111 – 122.

Lee, K. Y., Roh, M. I.,&Jeong, H. S. (2005). An improved genetic algorithm for multi-floor facility layout problems having inner structure walls and passages. Computers & Operations Research, 32 (4), 879 – 899.

Lee, R. e Moore, J. M., (1967). CORELAP-computerized relationship layout planning. Journal of Industrial Engineering, 18, pp195-200.

Lee, Y. H., & Lee, M. H. (2002). A shape-based block layout approach to facility layout problems using hybrid genetic algorithm. Computers & Industrial Engineering, 42, 237 – 248.

Leung, J. (1992). A graph-theoretic heuristic for flexible manufacturing systems. European Journal of Operational Research, 57 (2), 243 – 252.

Mahdi, A. H., Amet, H., & Portman, M. C. (1998). Physical layout with minimization of the transport cost (Research Internal Report). Nancy, France: LORIA.

Maia L. F. (2009). O uso do método do enxame de particulas na otimização da latência e consumo de energia de uma rede-em-chip., Universidade de Santa Cruz do Sul. http://hdl.handle.net/11624/424

Mason, E R. *Plant Layout requirements for the factory of the future.* AIPE Facilites Management, Operation and Engineering v. 16, n 1,p 32-35, 1989.

Matsuzaki, K., Takashi, I., & Yoshimoto, K. (1999). Heuristic algorithm to solve the multi-floor layout problem with the consideration of elevator utilization. Computers & Industrial Engineering, 36 (2), 487 – 502.

McKendall, A. R., Shang, J., & Kuppusamy, S. (2006). Simulated annealing heuristics for the dynamic facility layout problem. Computers& Operations Research, 33 (8), 2431 – 2444.

Mecklenburgh, J. C., (1985), Process Plant Layout. New York: Longman.

Meller, R. D., Narayanan, V., & Vance, P. H. (1999). Optimal facility layout design. Operations Research Letters, 23 (3 – 5), 117 – 127.

Meng, G., Heragu, S. S., & Zijm, H. (2004). Reconfigurable layout problem. International Journal of Production Research, 42 (22), 4709 – 4729.

Mir, M., & Imam, M. H. (2001). A hybrid optimization approach for layout design of unequal-area facilities. Computers & Industrial Engineering, 39 (1 – 2), 49 – 63.

Montevechi, J. A. B. (1989). Tecnologia de Grupo Aplicada ao Projeto de Células de Fabricação. Dissertação de Mestrado - UFSC. Florianópolis.

Montreuil, B. and Venkatadri, U. (1991), "Strategic Interpolative Design of Dynamic Manufacturing Systems Layout," Management Science, 37, 682-694.

Monks J.G., Theory and Problems of Operations Management. Schaum McGraw Hill, 1987.

Moore, J.M., (1974). Computer aided facilities design: an international survey, International Journal of Production Research. Vol. 12, No. 1, 21-44.

Moura, R. A. Kanban (1989)- A simplicidade do controle de produção. IM AN. São Paulo.

Muther, Richard. (1978), *Planejamento do layout: Sistema* SLP. São Paulo: Edgard Blucher.

Nearchou, A. C. (2006). Meta-heuristics from nature for the loop layout design problem. International Journal of Production Economics, 101 (2), 312 – 328.

Osman, I.; LAPORTE, G, (1996) Meta-heuristics: A Bibliography. Annals of Operations Research, 63(5):511–623.

Patsiatzis, D. I., & Papageorgiou, L. G. (2002). Optimal multi-floor process plant layout. Computers and Chemical Engineering, 26 (4 – 5), 575 – 583.

Proth, J. M. (1992). Conception et gestion des syste`mes de production. Presses Universitaires de France. pp. 68–77.

Rajasekharan, M., Peters, B. A., & Yang, T. (1998). A genetic algorithm for facility layout design in flexible manufacturing systems. International Journal of Production Research, 36 (1), 95 – 110.

Raoot, A. D., & Rakshit, A. (1991). A ‘fuzzy’ approach to facilities layout planning. International Journal of Production Research, 29, 835 – 857.

Rosenblatt, M. J. (1986). The dynamics of plant layout. Management Science, 32 (1), 76 – 86.

Rosenblatt, M. J. and Lee, H.L.(1968), “A Robustness Approach to Facilities Design,” International Journal on Production Research, 25, 479-486.

Seehof, J. M. e Evans, W. O., (1967). Automated layout design program. Journal of Industrial Engineering, 18 (12), pp 690-695.

Seixas, I. C. V. (2013). Aplicação de Métodos Emergentes em Problemas de Escalonamento do Tipo Flexible Job Shop, tese de Mestrado no Instituto Politécnico de Bragança.

Shayan, E., & Chittilappilly, A. (2004). Genetic algorithm for facilities layout problems based on slicing tree structure. International Journal of Production Research, 42 (19), 4055 – 4067.

Shi, Y.; Eberhart, R.C, 1998. A modified particle swarm optimizer. Proceedings of the IEEE International Conference on Evolutionary Computation. Piscataway, NJ: IEEE Press, p. 69-73.

Shore, R. H. and J. A. Tompkins, “Flexible Facilities Design,” AIIE Transactions, 12, 200-205, 1980.

Slack , Nigel et al., Administração de produção. Tradução A. B. Brandão et al. São Paulo, Editora Atlas, 1997

Solimanpur, M., Vrat, P., & Shankar, R. (2005). An ant algorithm for the single row layout problem in flexible manufacturing systems. Computers & Operations Research, 32 (3), 583 – 598.

S. Kirkpatrick, C. D. Gelatt, and M. P, (1983). Vecchi, “Optimization by Simulated Annealing,” Science, vol. 220, no. 4598, pp. 671–680.

S. M. Kalami Heris (2016) site: http://yarpiz.com/50/ypea102-particle-swarm-optimization#comment-399 (consultado em 2016).

Tanaka, K (1991). An Introduction to Fuzzy Logic for Practical Applications, Springer, 1997.

Tam, K. Y. (1992). Genetic algorithms, function optimization and facility layout design. European Journal of Operational Research, 63 (2), 322 – 346.

Tam, K. Y., & Chan, S. K. (1998). Solving facility layout problems with geometric constraints using parallel genetic algorithms: Experimentation and findings. International Journal of Production Research, 36 (12), 3253 – 3272.

Tavares J. A.R. (2000), Geração de Configurações de Sistemas Industriais com o Recurso à Tecnologia das Restrições e Computação Evolucionária, 2000

Tompkins, J. A. e Reed, R., (1976). An applied model for the facilities design problem. Journal of Production Research, 14 (5), pp 583-595.

Tompkins, J. A.,White, J. A., Bozer, Y. A., Frazelle, E.H., Tanchoco, J. M., & Trevino, J. (1996). Facilities planning. New York: Wiley

Tompkins, J. A. et al. Facilities Planning. 3 ed. New Jersey: John Wiley & Sons, 2003.

Tsuchiya, K., Bharitkar, S.,&Takefuji, Y. (1996). A neural network approach to facility layout problems. European Journal of Operational Research, 89 (3), 556 – 563.

Urban, T. L. (1993). A heuristic for the dynamic facility layout problem. IIE Transactions, 25 (4), 57 – 63.
URL:www.ebah.com.br/content/ABAAAA820AC/arranjo-fisico-planejamento-estrategico

Wang, M. J., Hu, M. H., & Ku, M. H. (2005). A solution to the unequal área facilities layout problem by genetic algorithm. Computers in Industry, 56 (2), 207 – 220.

Welgama, P. S., & Gibson, P. R. (1993). A construction algorithm for the machine layout problem with fixed pick-up and drop-off points. International Journal of Production Research, 31 (11), 2575 – 2590.

Wu, Y., & Appleton, E. (2002). The optimisation of block layout and aisle structure by a genetic algorithm. Computers & Industrial Engineering, 41 (4), 371 – 387.

Yang, T., & Kuo, C. (2003). A hierarchical AHP/DEA methodology for the facilities layout design problem. European Journal of Operational Research, 147, 128 – 136.

Yang, T., Peters, B. A., & Tu, M. (2005). Layout design for flexible manufacturing systems considering single-loop directional flow patterns. European Journal of Operational Research, 164 (2), 440 – 455.

Zadeh, L. A.; Fuzzy Sets. Information and Control, vol. 8, p. 338-353, 1965

Zhou, J. (1998). Algorithmes et outils pour l'analyse des flux de production a` l'aide du concept d'ordre. Ph.D. dissertation (in French). University of Strasbourg 1.

ANEXOS

Anexo 1 – [Modelo em AMPL para uma formulação discreta do problema de layout – instância da Figura 24]

```
#*****************************
#           SETS
#*****************************
#############################################################
#Lista de Equipamentos
set Equipamentos;
#Lista de Locais Possiveis
set Locais;
#Fluxo de Equipamentos de i para k*x
param Fluxo{i in Equipamentos, k in Equipamentos};
#Distancia de Locais j para l
param Distancia{j in Locais, l in Locais};
# Numero de Equipamentos
#param N;
#*********************************************
# * Decision variables
# *********************************************
# Xji = 1 quando no local j foi colocado o equipamento i
var  x{j in Locais, i in Equipamentos} binary;
/*********************************************
 * Objective function
 *********************************************/
minimize f_obj:
(sum {i in Equipamentos}
   (sum {j in Locais}
       (sum {k in Equipamentos: i<>k}
           (sum {l in Locais: l<>j}
   Fluxo[i,k]*Distancia[j,l]*x[j,i]*x[l,k]
             )
               )
    )
);
/*********************************************
 * Constraints
 *********************************************/
#restricao1 Em cada local tem que ser instadado só 1 equipamento
subject to  equip {j in Locais} :
(sum {i in Equipamentos}((x[j,i])))== 1 ;
#restricao1 Em cada equipamento tem que ser instadado só 1 local
subject to  local {i in Equipamentos} :
(sum {j in Locais}((x[j,i])))== 1 ;

/*********************************************
 * DATA  exemplo 3
 *********************************************/
data;
#lista de Equipamentos
set Equipamentos:= 1 2 3;
```

```
# Lista de Locais
set Locais:= A B C;

#Fluxo de Equipamentos de i para k*x

param Fluxo:      1 2 3 :=
                                1  . 5 8
                                2  5 . 3
                                3  8 3 .;


#Distancia de Locais j para l

param Distancia:      A B C :=
                                  A  . 4 7
                                  B  4 . 9
                                  C  7 9 .;

# Numero de Equipamentos
#param N:=3;

/*********************************************
 * SOLUTION AND VISUALIZATION
 *********************************************/

#option solver ipopt; #deu mas nao inteiros
#option solver gurobi; #nao acha o solver???
#option solver minos; #deu mas nao inteiros
#option solver cplex; #diz que nao tem licensa para o algoritmo barreira
solve;

display f_obj;
display x;
```

Anexo 2 – [Resultados do Gurobi para a instância da Figura 24 do problema de layout, resolvida pela formulação discreta]

```
Gurobi 6.5.0: outlev 1
logfreq 10
Logfile "Pasta/file_log.txt"
Optimize a model with 6 rows, 9 columns and 18 nonzeros
Model has 18 quadratic objective terms
Coefficient statistics:
  Matrix range     [1e+00, 1e+00]
  Objective range [0e+00, 0e+00]
  Bounds range     [1e+00, 1e+00]
  RHS range        [1e+00, 1e+00]
Found heuristic solution: objective 196
Presolve time: 0.00s
Presolved: 24 rows, 27 columns, 72 nonzeros
Variable types: 0 continuous, 27 integer (27 binary)

Root relaxation: objective 0.000000e+00, 7 iterations, 0.00 seconds

    Nodes    |    Current Node    |     Objective Bounds      |     Work
 Expl Unexpl |  Obj  Depth IntInf | Incumbent    BestBd   Gap | It/Node Time

     0     0    0.00000    0    6  196.00000    0.00000   100%     -    0s
     0     0    0.00000    0    6  196.00000    0.00000   100%     -    0s
H    0     0                     188.0000000    0.00000   100%     -    0s
     0     0   21.00000    0    7  188.00000   21.00000  88.8%     -    0s
     0     2   21.00000    0    7  188.00000   21.00000  88.8%     -    0s

Cutting planes:
  Gomory: 2
  MIR: 10
  Zero half: 4

Explored 3 nodes (46 simplex iterations) in 0.02 seconds
```

```
Thread count was 8 (of 8 available processors)

Optimal solution found (tolerance 1.00e-04)
Best objective 1.880000000000e+02, best bound 1.880000000000e+02, gap 0.0%
Optimize a model with 6 rows, 9 columns and 18 nonzeros
Model has 18 quadratic objective terms
Coefficient statistics:
  Matrix range    [1e+00, 1e+00]
  Objective range [0e+00, 0e+00]
  Bounds range    [1e+00, 1e+00]
  RHS range       [1e+00, 1e+00]
Iteration    Objective       Primal Inf.    Dual Inf.      Time
       0    1.8800000e+02   0.000000e+00   0.000000e+00      0s

Solved in 0 iterations and 0.00 seconds
Optimal objective  1.880000000e+02
Gurobi 6.5.0: optimal solution; objective 188
46 simplex iterations
3 branch-and-cut nodes
Gurobi 6.5.0: outlev 1
logfreq 10
Logfile "Pasta/file_log.txt"
Optimize a model with 6 rows, 9 columns and 18 nonzeros
Model has 18 quadratic objective terms
Coefficient statistics:
  Matrix range    [1e+00, 1e+00]
  Objective range [0e+00, 0e+00]
  Bounds range    [1e+00, 1e+00]
  RHS range       [1e+00, 1e+00]
Found heuristic solution: objective 196
Presolve time: 0.00s
Presolved: 24 rows, 27 columns, 72 nonzeros

Loaded MIP start with objective 188
```

Variable types: 0 continuous, 27 integer (27 binary)

Root relaxation: objective 0.000000e+00, 7 iterations, 0.00 seconds

Nodes		Current Node		Objective Bounds		Work	
Expl	Unexpl	Obj	Depth	IntInf	Incumbent	BestBd	Gap	It/Node	Time
0	0	0.00000	0	6	188.00000	0.00000	100%	-	0s
0	0	0.00000	0	6	188.00000	0.00000	100%	-	0s
0	0	21.00000	0	7	188.00000	21.00000	88.8%	-	0s
0	2	21.00000	0	7	188.00000	21.00000	88.8%	-	0s

Cutting planes:
  Gomory: 2
  MIR: 10
  Zero half: 4

Explored 3 nodes (46 simplex iterations) in 0.02 seconds
Thread count was 8 (of 8 available processors)

Optimal solution found (tolerance 1.00e-04)
Best objective 1.880000000000e+02, best bound 1.880000000000e+02, gap 0.0%
Optimize a model with 6 rows, 9 columns and 18 nonzeros
Model has 18 quadratic objective terms
Coefficient statistics:
  Matrix range   [1e+00, 1e+00]
  Objective range [0e+00, 0e+00]
  Bounds range   [1e+00, 1e+00]
  RHS range   [1e+00, 1e+00]

Iteration	Objective	Primal Inf.	Dual Inf.	Time
0	1.8800000e+02	0.000000e+00	0.000000e+00	0s

Solved in 0 iterations and 0.00 seconds

Optimal objective 1.880000000e+02

Gurobi 6.5.0: optimal solution; objective 188

46 simplex iterations

3 branch-and-cut nodes

Anexo 3 – [Função codificar_ins.m]

```
function [r]=codificar_ins(xin,yin,h,w)
[~,pos]=min(abs([xin-w/2,xin+w/2,yin-h/2,yin+h/2]))
switch pos(1)
    case 1
        r=(1.5+yin/h)/4;
    case 2
        r=(3.5-yin/h)/4;
    case 3
        r=(2.5-xin/w)/4;
    case 4
        r=(0.5+xin/w)/4;
end
```

Anexo 4 – [Resultados Matlab]

Resoluções	Duração	Best Cost	XMIN	YMIN	XMAX	YMAX	Container Area	Machines Area	Unused Are	Unused Area Ratio
1	00:01:00	65805	9,11	4,01	93,9	75,53	6064	3130	2934	48,4%
2	00:01:02	57686	15,2	19,54	96,22	72,77	4312	3130	1182	27,4%
3	00:01:00	60869	5,35	8,36	80,38	77,38	5177	3130	2047	39,5%
4	00:01:00	69518	17,61	5,85	91,13	75,92	5151	3130	2021	39,2%
5	00:01:00	72035	4,13	16,06	84,64	75,47	4783	3130	1654	34,6%
6	00:01:00	69573	13,77	1,3	93,77	64,33	5040	3130	1910	37,9%
7	00:01:00	68328	10,2	13,09	89,2	78,09	5135	3130	2005	39,0%
8	00:01:00	64132	14,38	4,16	89,06	70,16	4929	3130	1799	36,5%
9	00:01:00	71646	9,33	4,37	80,69	73,61	4940	3130	1810	36,6%
10	00:01:00	70873	16,43	1,08	87,5	76,09	5330	3130	2200	41,3%
11	00:01:00	64331	16,48	12,38	91,48	77,38	4875	3130	1745	35,8%
12	00:01:00	68056	25,58	1,29	96,32	76,59	5326	3130	2196	41,2%
13	00:01:00	67182	18,24	5,8	97,24	69,8	5056	3130	1926	38,0%
14	00:01:00	65843	20,08	7,95	93,08	78,28	5134	3130	2005	39,0%
15	00:01:00	65279	12,82	4,75	93,82	65,75	4941	3130	1811	36,6%
16	00:01:00	64959	10,2	10,68	84,73	75,68	4844	3130	1714	35,4%
17	00:01:00	66080	8,36	11,58	93,36	78,58	5695	3130	2565	45,0%
18	00:01:00	54220	2,64	21,26	79,64	76,26	4235	3130	1105	26,1%
19	00:01:00	61938	2,51	6,24	88,51	58,66	4508	3130	1378	30,5%
20	00:01:00	71823	8,48	8,38	86,61	75,12	5214	3130	2084	39,9%
21	00:01:00	56962	13,91	20,16	96,75	72,84	43647	3130	1234	28,3%
22	00:01:00	66552	12,8	408	93,69	68,92	5245	3130	2115	40,3%
23	00:01:00	61760	22,72	1,58	97,72	72,58	5325	3130	2195	41,2%
24	00:01:00	69297	23,16	2,34	97,88	76,74	5559	3130	2428	43,7%
25	00:01:00	60153	4,68	1,82	82,13	68,13	5135	3130	2005	39,0%
26	00:01:00	72060	15,13	3,21	88,51	77,21	5430	3130	2300	42,3%
27	00:01:00	69203	7,25	2,35	95,25	68,35	5800	3130	2678	46,1%
28	00:01:00	70982	15,82	7,94	92,82	74,94	5159	3130	2029	39,3%
29	00:01:00	62268	13,08	3,89	79,08	76,89	4818	3130	1688	35,0%
30	00:01:00	63754	2,85	5,54	81,85	75,19	5502	3130	2372	43,1%
31	00:01:00	59205	13,99	17,36	95,03	71,43	4381	3130	1251	28,6%
32	00:01:00	62603	1,4	3,13	81,4	71,13	5440	3130	2310	42,5%
33	00:01:00	69308	3,55	5,46	93,55	67,01	5539	3130	2409	43,5%
34	00:01:00	64941	6,01	12,24	85,51	71,24	4690	3130	1560	33,3%
35	00:01:00	61455	25,97	4,0997	92,29	74,59	4675	3130	1545	33,0%
36	00:01:00	59888	21,79	7,6	96,8	73,72	4959	3130	1829	36,8%
37	00:01:00	60907	6,47	7,58	78,47	70,7	4544	3130	1414	31,1%
38	00:01:00	67567	4,1	14,29	97,1	69,29	5115	3130	1985	38,8%
39	00:01:00	65635	9,067	19,27	97,06	74,39	4850	3130	1720	35,4%
40	00:01:00	74152	4,99	2,39	95,99	70,39	6188	3130	3058	49,4%
41	00:01:00	65582	15,95	9,65	93,26	67,67	4485	3130	1355	30,0%

42	00:01:00	70823	1,61	2,59	88,19	74,59	6233	3130	3103	49,8%
43	00:01:00	67085	21,18	3,26	89,06	78,27	5090	3130	1960	38,5%
44	00:01:00	59147	11,2	14,67	91,62	76,2	4948	3130	1,818	36,7%
45	00:01:00	65126	9,22	8,88	84,31	75,81	5025	3130	1895	37,7%
46	00:01:00	66288	8,61	4,08	88,63	68,53	5157	3130	2027	39,3%
47	00:01:00	71636	12,23	2,07	92,23	65,07	5040	3130	1910	37,9%
48	00:01:00	67189	23,72	7,42	95,72	72,47	4683	3130	1553	33,1%
49	00:01:00	68706	4,78	18,59	94,78	70,59	4680	3130	1550	33,1%
50	00:01:00	64302	20,44	3,31	97,44	70,38	5164	3130	2034	39,4%

Anexo 5 – [Resultados Matlab – Caso Pratico]

Resoluções	Tempo	Best Cost	XMIN	YMIN	XMAX	YMAX	Container Area	Machines Area	Unused Are	Unused Are Ratio
1	00:01:00	5229	1483	1	74,48	50,57	2957	1729	1228	41,5%
2	00:01:00	5345	2	2,14	78,98	44,26	3242	1729	1513	46,6%
3	00:01:00	5486	5,72	9,86	78,1	49,21	2848	1729	1119	39,3%
4	00:01:00	5728	8,67	1,02	69,47	50,25	2993	1729	1264	42,2%
5	00:01:00	5513	24,42	2,53	78,98	55,32	2880	1729	1151	39,9%
6	00:01:00	6498	9,05	6,08	62,49	56,99	2720	1729	991	36,5%
7	00:01:00	5515	3,48	19	73,53	59	2802	1729	1073	38,3%
8	00:01:00	5964	3,81	1,45	62,86	49,68	2847	1729	1118	39,3%
9	00:01:00	5620	19,03	3,9	75,09	55	2864	1729	1135	39,6%
10	00:01:00	6166	7,93	1,03	59,4	55,8	2818	1729	1089	38,6%
11	00:01:00	5353	25,79	3,1	78,99	55,12	2767	1729	1038	37,5%
12	00:01:00	5326	11,8	7,58	76,04	55,08	3051	1729	1322	43,3%
13	00:01:00	5685	2,54	15,76	75,61	56,11	2948	1729	1219	41,3%
14	00:01:00	5080	1	10,5	65,7	50,61	2595	1729	866	33,4%
15	00:01:00	6674	2,95	11,28	70,11	51,53	2703	1729	973	36,0%
16	00:01:00	5268	5,58	4,95	58,8	59	2876	1729	1147	39,8%
17	00:01:00	5840	4	3,64	57,22	54,65	2714	1729	985	36,3%
18	00:01:00	4892	12,15	10,81	77,3	49,91	2547	1729	818	32,1%
19	00:01:00	5127	5,55	11,71	61,9	59	2661	1729	931	35,0%
20	00:01:00	5364	3,79	20,36	77,07	56,64	2659	1729	930	34,9%
21	00:01:00	5819	8,42	16,76	74,66	58,05	2731	1729	1002	36,7%
22	00:01:00	6922	1	9,37	63,06	58,88	3073	1729	1344	43,7%
23	00:01:00	5324	16,67	9,62	75,7	55,35	2699	1729	970	35,9%
24	00:01:00	5999	23,81	4,49	75,9	58,5	2813	1729	1084	38,5%
25	00:01:00	5709	1,56	2,74	59,31	51,8	2833	1729	1104	38,9%
26	00:01:00	5644	16,61	7,74	76,95	55,76	2897	1729	1168	40,3%
27	00:01:00	5782	1,37	4,14	65,16	53,21	3129	1729	1400	44,7%
28	00:01:00	5620	2,82	10,89	73,06	56,78	3223	1729	1494	46,3%
29	00:01:00	5306	14,52	5,97	70,56	56	2803	1729	1074	38,3%
30	00:01:00	4453	9,41	14,61	69,4	58,67	2643	1729	914	34,6%
31	00:01:00	7202	2,18	11,91	74,41	55,29	3133	1729	1404	44,8%
32	00:01:00	5778	20,24	4,31	71,25	57,46	2711	1729	981	36,2%
33	00:01:00	4808	1	6,91	60,68	51,32	2651	1729	921	34,7%
34	00:01:00	4882	19,78	7,9	78,79	52,56	2634	1729	905	34,3%
35	00:01:00	6812	5,11	5,73	71,35	49,57	2904	1729	1175	40,4%
36	00:01:00	5137	3,38	12,03	73,66	51,07	2744	1729	1015	36,9%
37	00:01:00	5691	20,47	3,83	75,85	54,85	2825	1729	1096	38,8%
38	00:01:00	5026	11,46	12,02	71,4	57,11	2703	1729	974	36,0%
39	00:01:00	5229	6,18	6,32	59,29	55,33	2602	1729	873	33,5%
40	00:01:00	5065	1,08	11,9	63,45	56,52	2783	1729	1054	37,8%
41	00:01:00	5377	2,28	6,11	55,65	58,14	2776	1729	1047	37,7%
42	00:01:00	5295	5,76	1,5	64,77	49,45	2827	1729	1098	38,8%

43	00:01:00	6541	1,02	7,34	75,47	48,85	3090	1729	1361	44,0%
44	00:01:00	5140	17,96	1	77,29	54,07	3148	1729	1419	45,0%
45	00:01:00	4978	25,21	9,36	75,61	57,38	2420	1729	691	28,5%
46	00:01:00	7413	1	1,9	68,57	52	3380	1729	1651	48,8%
47	00:01:00	5838	16,96	1,3	64,96	58,81	2760	1729	1031	37,4%
48	00:01:00	5230	6,92	13,92	65,93	58,93	2656	1729	927	34,9%
49	00:01:00	5395	4,04	5,7	57,37	56,99	2733	1729	1004	36,7%
50	00:01:00	4956	4	11,8	73,76	50,82	2722	1729	993	36,4%