Universidade do Minho
Escola de Engenharia

Mário Manuel Silva Leite

Abordagens de otimização para o planeamento e escalonamento integrado de operações

Dissertação de Mestrado
Mestrado em Engenharia de Sistemas

Trabalho efetuado sob a orientação do
Professor Doutor Cláudio Manuel Martins Alves

Janeiro de 2017

AGRADECIMENTOS

Não poderia deixar passar este momento importante da minha vida académica sem agradecer às pessoas que tiveram um papel fundamental ao longo do meu percurso e, em especial, neste projeto.

Queria agradecer de uma forma particular ao Professor Doutor Cláudio Manuel Martins Alves pela sua orientação na minha dissertação. Por toda a compreensão, paciência, apoio e dedicação. Estou certo que não poderia ter um melhor suporte académico. A sua exigência e crítica promoveram o meu máximo empenho. Na fase inicial desta etapa, num momento conturbado para mim, não me esqueço que se mostrou 100% disponível para que esta dissertação pudesse efetivamente avançar.

Também nessa altura pude contar com o apoio da Diretora do Mestrado em Engenharia de Sistemas, a Professora Doutora Maria Teresa Torres Monteiro, pelo que também lhe agradeço.

Um agradecimento especial à minha namorada Mónica por estar sempre presente. Nos bons e nos maus momentos, esteve sempre disponível para me ouvir e me ajudar. Foi o meu pilar apoiando-me nas minhas decisões mais difíceis.

A toda a minha família, agradeço pela oportunidade que me deram em poder prosseguir os estudos e pelo interesse sempre demonstrado na evolução dos meus projetos.

De forma generalizada, agradeço a todos os meus amigos e professores de mestrado que contribuíram para o meu crescimento pessoal e académico.

RESUMO

Neste projeto consideramos o problema integrado de planeamento e escalonamento de operações em máquinas paralelas e idênticas, descrito em Kis e Kovács (2012). O problema é composto por duas partes que são resolvidas, simultaneamente, de uma forma integrada. A primeira parte consiste em determinar as tarefas que serão processadas em cada período de tempo. Esta é a parte de planeamento do problema que tem de ser resolvido para o horizonte de tempo determinado. A segunda parte consiste em atribuir as tarefas às máquinas disponíveis em cada período de tempo, de acordo com as suas datas de lançamento correspondentes e de modo a que todas as tarefas sejam processadas até ao final do horizonte de planeamento. Sempre que uma tarefa é realizada antes ou após a sua data esperada, incorre numa penalização. O objetivo global do problema consiste em determinar o planeamento e o escalonamento associados que minimizem os custos totais dessas penalizações.

Tendo em conta as caraterísticas particulares do problema em estudo, foi feita uma breve análise a problemas de escalonamento de operações existentes na literatura e onde se percebe a forte ligação entre os problemas de máquinas paralelas idênticas e os problemas de corte e empacotamento de 1-dimensão, mais especificamente, de *bin-packing*. Pelo que foram seguidas técnicas associadas a este tipo de problemas.

Esta dissertação apresenta novas abordagens de otimização com base em métodos heurísticos de pesquisa local e meta-heurísticos, baseados na pesquisa de vizinhança variável (VNS – *Variable Neighborhood Search*) usando duas estruturas de vizinhança. Foram implementados dois algoritmos diferentes na construção das soluções inicias conjugados com quinze variantes da sequência inicial das tarefas, alcançando, naturalmente, resultados distintos.

Para a obtenção de resultados e de modo a avaliar o desempenho dos mesmos, foram realizadas experiências computacionais com instâncias de referência descritas na literatura. Assim, para além da comparação entre os diferentes resultados obtidos neste projeto foi também possível comparar os resultados conseguidos com outros já existentes para as mesmas instâncias.

Palavras-Chave: Planeamento, Escalonamento, Problemas integrados de otimização, Heurísticas, Meta-heurísticas

ABSTRACT

In this project, we consider the integrated planning and scheduling problem on parallel identical machines as in Kis and Kovács (2012). The problem is composed by two parts that are solved simultaneous in an integrated form. The first part consists in assessing which should be processed in each period of time. This is the part of the problem planning that has to be solved in a certain planning horizon. The second part consists in allocating jobs to the available machines in each time period according to their corresponding release date in a way that every job is processed until the end of the time space. Every time a job is allocated before or after its due date it incurs in a penalty. The overall objective of the problem consists in establishing the integrated planning and scheduling that minimizes the total costs of these penalties.

Taking into consideration the specifics of the problem under study a brief analysis to the scheduling operation problems in literature was made in which a strong connection can be correlated between the problems of identical parallel machine and the problems of cutting and packing 1-dimension, more specifically bin-packing. Therefore, techniques associated with this type of problem were followed.

This dissertation introduces new approaches of optimization based in heuristics methods of local search and metaheuristics based on Variable Neighborhood Search (VNS) using two neighborhood structures. Two different algorithms were implemented in the construction of initial solutions combining with fifteen different initial sequence of jobs, reaching distinct results.

To achieve results and in order to evaluate their performance, computational experiences were performed with reference instances described in the literature. Thus, in addition to the comparison between different obtained results in this project was also possible to compare with the results achieved in other already existing projects for the same instances.

Keywords: Planning, Scheduling, Integrated optimization problems, Heuristics, Metaheuristics

ÍNDICE

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE GRÁFICOS

LISTA DE ALGORITMOS

LISTA DE ABREVIATURAS, SIGLAS E ACRÓNIMOS

BPP – Problemas de empacotamento em caixas (*Bin Packing Problem*)

BVNS – Método de pesquisa por vizinhanças variáveis (*Basic Variable Neighborhood Search*)

C&P – Corte e Empacotamento (*Cutting & Packing*)

CSP – Problema de corte de *stock* (*Cutting Stock Problem*)

IIPP – Problema de empacotamento de itens idênticos (*Identical Item Packing Problem*)

IPT – Problema do tipo intermédio (*Intermediate Problem Types*).

KP – Problema de mochila (*Knapsack Problem*)

MBSBPP – Problema de empacotamento em caixas de tamanho variável (*Multiple Bin-Size Bin Packing Problem*)

MHKP – Problema de várias mochilas heterogéneas (*Multiple Heterogeneous Knapsack Problem*)

MHLOPP – Problemas de alocação em vários objetos grandes heterogéneos (*Multiple Heterogeneous Large Object Placement Problem*)

MIKP – Problema de várias mochilas idênticas (*Multiple Identical Knapsack Problem*)

MILOPP –Problemas de alocação em vários objetos grandes idênticos (*Multiple Identical Large Object Placement Problem)*

MSSCSP – Problema de corte com *stock* de tamanho variado (*Multiple Stock-Size Cutting Stock Problem*)

NP – *Nondeterministic Polynomial-bounded*

ODP – Problema com dimensão variável (*Open Dimension Problem*)

PP – Problema de alocação (*Placement Problem*)

RBPP – Problema de empacotamento de caixas residual (*Residual Bin Packing Problem*)

RCSP – Problema de corte residual (*Residual Cutting Stock Problem*)

SBSBPP – Problema de empacotamento em caixas de tamanho único (*Single Bin-Size Bin Packing Problem*)

SKP – Problema de mochila única (*Single Knapsack Problem*)

SLOPP – Problemas de alocação num único objeto grande *(Single Large Object Placement Problem*)

SSSCSP – Problema de corte com *stock* de tamanho único (*Single Stock-Size Cutting Stock Problem*)

VNS – Pesquisa por vizinhança variável (*Variable Neighborhood Search*)

1. INTRODUÇÃO

A importância e o potencial benefício que advêm da otimização integrada de operações nas cadeias de abastecimento é bem conhecida. Otimizar as operações ao longo das diferentes áreas funcionais das empresas contribui para a redução dos custos e para a melhoria do desempenho das mesmas. Em linha com essas observações, muitos autores têm apoiado a ideia de uma coordenação mais abrangente entre planeamento e escalonamento desde os processos de aquisição de materiais até ao planeamento das entregas de produtos acabados (Grossmann, 2009). De um ponto de vista computacional, a resolução de problemas reais de otimização integrada apresenta-se como um verdadeiro desafio.

Os problemas de planeamento e escalonamento incluem aspetos estratégicos de longo prazo, decisões táticas de médio prazo e questões operacionais de curto prazo, como acontece, por exemplo, quando se trata de decidir se determinadas instalações produtivas devem ser, ou não, construídas em função da quantidade de artigos que se prevê vir a produzir nos recursos existentes. Uma vez que estes problemas são profundamente interdependentes, eles devem ser resolvidos de forma integrada. Essa integração levanta várias dificuldades tanto ao nível organizacional como ao nível do comportamento humano, tal como foi descrito em Shobrys e White (2000), e que não são exclusivos ao domínio da gestão de cadeias de abastecimento.

Neste projeto foi considerado o problema integrado de planeamento e escalonamento de operações em máquinas paralelas e idênticas, analisado por Kis e Kovács (2012). Este é composto por duas partes que são resolvidas, simultaneamente, de uma forma integrada. A primeira parte, que é relativa ao planeamento que tem de ser resolvido para um horizonte de tempo determinado, consiste em determinar as tarefas que serão processadas em cada período de tempo. A segunda parte, escalonamento, consiste em atribuir as tarefas às máquinas disponíveis em cada período de tempo, tendo em conta as datas em que as tarefas estão disponíveis para serem executadas, de modo a que todas as tarefas sejam processadas até ao final do horizonte de planeamento. Sempre que uma tarefa é feita antes ou depois da sua data esperada esta incorre numa penalização. O objetivo do problema consiste em determinar o planeamento tático e escalonamento que minimizem os custos totais associados a essas penalizações.

Na abordagem descrita em Kis e Kovács (2012), o horizonte de planeamento é dividido em períodos de tempo. Para cada período de tempo existe um conjunto de máquinas idênticas com disponibilidade limitada. Além disso, existe um conjunto de tarefas com datas

de disponibilidade, prazos de entrega, tempos de processamento e penalizações que são aplicadas sempre que a tarefa é concluída antes ou depois do seu prazo de entrega. Cada tarefa tem de ser atribuída a um período de tempo e a uma máquina, de tal modo que as penalizações totais incorridas sejam minimizadas. Uma vez que as tarefas não podem ser divididas entre períodos e entre máquinas, o problema pode ser entendido como um problema de empacotamento a 1-dimensão que é conhecido por ser NP-difícil (Garey e Johnson, 1978). Neste tipo de problemas o tempo computacional para encontrar uma solução ótima aumenta exponencialmente com o tamanho do problema, pelo que se torna relevante conseguir encontrar a solução através de métodos heurísticos e de meta-heurísticas que se apresentam com uma exigência computacional substancialmente menor a uma outra que considere todas as possibilidades.

1.1 Objetivo da Dissertação

Tendo em conta as caraterísticas específicas deste problema em estudo, com esta dissertação pretende-se desenvolver novas abordagens de otimização com base em métodos heurísticos de pesquisa local e meta-heurísticas, tais como a pesquisa por vizinhanças variáveis (VNS – *Variable Neighborhood Search*). O problema integrado de planeamento e escalonamento de operações em máquinas paralelas e idênticas começou a ser estudado muito recentemente, e o número de abordagens descritas na literatura é ainda muito reduzido. O estudo dos algoritmos heurísticos existentes nesta área é importante para perceber as suas limitações e poder desenvolver novos métodos alternativos com melhores desempenhos.

O desenvolvimento destas novas abordagens contemplará, numa primeira fase, a descrição de todos os elementos que caraterizam abordagens desta natureza. Já na segunda fase, fazer a implementação dos algoritmos propostos numa linguagem de programação e no teste funcional dos mesmos. A sua implementação e execução com um conjunto abrangente de experiências computacionais permitirão avaliar o desempenho dos algoritmos desenvolvidos. Conjunto esse que será constituído por instâncias de referência descritas na literatura para esse efeito, o que permitirá fazer uma melhor comparação dos resultados entre os novos algoritmos com os já existentes, favorecendo a análise computacional dos métodos implementados.

1.2 Metodologia de Investigação

A filosofia de investigação usada está relacionada com a forma como o mundo é observado. O posicionamento seguido é importante para melhor entender os pressupostos da investigação e guiar o caminho científico. Sendo que este vai influenciar as estratégias e métodos de investigação utilizados no decorrer da mesma.

Assim, recorrendo a várias pesquisas na literatura e estudos focados nas áreas de intervenção deste projeto, consideramos o conhecimento como aceitável e válido (visão epistemológica, também conhecida como a Teoria do Conhecimento). Esta visão relaciona-se com uma outra, sobretudo no caso de pesquisa científica, centrada na natureza do objeto de estudo (ontologia). O objetivo de encontrar abordagens alternativas de otimização para problemas de planeamento e escalonamento vistos de uma forma integrada, com base em métodos e modelos, anteriormente estudados, e poder fazer novas experiências, vai ao encontro de uma metodologia de investigação típica das ciências exatas, o positivismo.

A abordagem de investigação é dedutiva, uma vez que a teoria existente pode ser usada para comprovar a eficácia das hipóteses avançadas através da análise dos dados recolhidos. Quanto à estratégia de investigação aplicada é a experimentação, assente num estudo exploratório.

As escolhas ou métodos usados para a análise dos resultados foram com base em valores quantitativos, portanto, com métodos únicos de avaliação.

As técnicas e procedimentos para recolha e análise de dados foram através de experiências computacionais, no sentido de avaliar o desempenho dos algoritmos desenvolvidos. Mais uma vez reiteramos que foram usadas instâncias de referência descritas na literatura contemplando um conjunto muito diversificado de situações. Os dados obtidos foram dispostos em folhas de *Excel* para permitir uma visão estatística através de uma componente gráfica de fácil visibilidade. De modo a comprovar a fiabilidade e validade dos resultados encontrados, todas as instâncias testadas tiveram um conjunto de documentação que certifica e sustenta a credibilidade dos mesmos.

1.3 Estrutura da Dissertação

Capítulo 1: Introdução

Neste primeiro capítulo fazemos uma apresentação e enquadramento do tema da dissertação, do objetivo de estudo e as metodologias de investigação seguidas.

Capítulo 2: Revisão de literatura

Neste capítulo, fazemos uma revisão geral da literatura mais recente associada à resolução de problemas de escalonamento que nos permite distinguir as grandes classes de problemas nesta área. O objetivo deste capítulo consiste em identificar as principais características que distinguem os problemas de escalonamento e perceber a ligação que estes têm com problemas de corte e empacotamento. É feita também uma breve análise à vantagem que é o uso de heurísticas para resolver os problemas mais complexos.

Capítulo 3: Problema de planeamento e escalonamento

Aqui debruçamo-nos especificamente sobre o problema descrito, em 2012, por Kis e Kovács, onde consideram a integração do planeamento e do escalonamento de operações em máquinas paralelas e idênticas. É exposta a definição do problema bem como a notação deste, recorrendo a exemplos.

Capítulo 4: Algoritmos de otimização

Neste capítulo, desenvolvemos novas abordagens de otimização, numa linguagem de programação, para o problema em estudo, baseadas em métodos de pesquisa local e meta-heurísticas. Concebemos abordagens alternativas de resolução baseadas em métodos heurísticos. Começamos por métodos de pesquisa local, definindo todos os elementos que caracterizam as abordagens desta natureza. A seguir definimos abordagens meta-heurísticas, em particular, abordagens de pesquisa baseadas em várias vizinhanças (*variable neighborhood search*).

Capítulo 5: Experiências computacionais

Realizamos um conjunto abrangente de experiências computacionais para avaliar o desempenho dos algoritmos desenvolvidos. Usamos instâncias de referência descritas na literatura para esse efeito. Descrevemo-las e apresentamos os resultados. Por fim, fazemos uma análise crítica destes.

Capítulo 6: Conclusões

Neste capítulo apresentamos as principais conclusões que foram retiradas do vasto conjunto de resultados obtidos nas experiências computacionais.

Bibliografia

Apresentamos uma lista de referências de todo o material que fundamentou esta dissertação de mestrado.

Anexos

Os anexos contemplam um problema (exemplo) para demonstrar passo a passo a construção de uma solução inicial, bem como a exemplificar as trocas possíveis entre tarefas. Têm também uma pequena parte do conjunto de informações que acompanham cada um dos resultados de cada instância.

2. REVISÃO DE LITERATURA

Neste capítulo é feita uma revisão bibliográfica associada aos principais conceitos existentes no nosso problema em análise. Entre os quais destacam-se as noções de planeamento e escalonamento, as classificações dos problemas de escalonamento, a associação de problemas de escalonamento a problemas de corte e empacotamento, a complexidade do problema e uma revisão às heurísticas com base em pesquisa local.

2.1 Introdução

O mercado atual está cada vez mais competitivo, exigindo às empresas que procurem sucessivas alternativas para que possam enfrentar, em vantagem, a concorrência na comercialização dos seus produtos. Aquelas que ficam satisfeitas com a sua situação atual, e se acomodam, arriscam-se a regredir num futuro bem próximo.

Essa necessidade não é de agora, já no passado encontrámos situações em que a adaptação das empresas às circunstâncias da altura foi determinante para o sucesso. A procura em reduzir os custos através da eliminação de desperdícios e, simultaneamente, pensar em manter ou melhorar a qualidade apresentada aos seus potenciais clientes é sempre uma luta diária. Os clientes querem os produtos na hora certa, na quantidade certa, com qualidade e ao preço mais baixo possível. Tal como refere Sule (2007), as "organizações industriais existem para produzir e fornecer produtos que os clientes precisam, a um preço que eles estão dispostos a pagar.".

Percebe-se, desde logo, a importância e o potencial benefício que advêm da otimização integrada de operações nas suas cadeias de abastecimento.

A cadeia de abastecimento revela todos os estágios em que é acrescentado valor ao produto fabricado, desde os fornecedores, fabricantes, distribuidores até aos clientes (Chang e Lee, 2004). Atualmente, com a ideia de produzir só o que é estritamente necessário, *just-in-time*, as firmas procuram fazer o mesmo, ou mais, com os mesmos, ou menos, recursos. Ficando apenas com o que é essencial à produção, excluindo o excedente que apenas aumenta o custo de fabrico do produto. A produção só é realizada se for necessária, sendo processada no momento em que faz falta, minimizando os custos de inventário (Sugimori *et al.*, 1977).

Neste momento, o facto de o mercado possuir um consumidor com uma procura muito variada, produtos com um tempo de vida curto e a necessidade das empresas em reduzir os custos, leva a que estas adiram a uma política de sistemas com "inventário zero". Mas, para

conseguirem satisfazer da melhor forma os seus clientes, torna-se necessário ter produtos em *stock* de modo a darem uma resposta rápida aos pedidos. Em muitos casos a espera por parte do cliente pode ditar a perda deste.

> As tendências atuais do mercado, como a procura por variedade, ciclos de vida mais curtos e a pressão competitiva para reduzir custos, resultaram na necessidade de sistemas de inventário zero. No entanto, para manter a quota de mercado, o sistema deve ser de resposta rápida o que implica que tem de ser mantido mais *stock*. (Jain e Meeran, 1999)

Os gestores das empresas, recorrentemente, têm várias questões às quais têm de dar respostas de forma rápida, por exemplo: *O que comprar?*, *Quando comprar?*, *Onde guardar ou colocar?*, *O que vender?*, *O que produzir?, Quando produzir?*, *Onde produzir?*, entre outras. Não existem respostas fixas. Estas estão constantemente a mudar conforme a procura do cliente, as variações na quantidade de matéria-prima e subjacentemente com o seu preço (Shobrys e White, 2000).

Estas condições de mercado, mais exigentes e dinâmicas, têm conduzido a uma interação mais estreita entre cada uma das fases e despertou, especialmente nas últimas décadas, esforços em investigações de modelos integrados que resultam em melhorias nas tomadas de decisão, ajudando a encontrar as melhores soluções globais possíveis (Baldea e Harjunkoski, 2014).

A configuração organizacional de uma empresa, e certamente do seu sistema produtivo, é altamente dependente do seu posicionamento e objetivos, que se deverão alcançar por diretivas da gestão empresarial, incluindo a gestão da produção. Otimizar as operações, ao longo das diferentes áreas funcionais das empresas, contribui para a redução dos custos e para a melhoria do desempenho das mesmas. As abordagens tradicionais de problemas de produção e roteamento viam de forma separada a fase de produção dos artigos e a fase de distribuição ou entrega. Assim sendo, as duas fases vistas sem uma coordenação poderão dar origem a uma solução que não é globalmente ótima (Chang e Lee, 2004). Muitas indústrias têm a integração de problemas de produção e de transporte como um dos seus principais impulsionadores de negócio. Isto é, a sua eficácia é determinante para o sucesso, ou não, com os seus clientes. No entanto, para resolver simultaneamente problemas de produção e roteamento, mesmo que pequenos, requerem um tempo computacional elevado, sobretudo se os recursos de transporte forem reduzidos (Geismar *et al.*, 2008).

Nesta investigação, consideramos o problema integrado de planeamento e escalonamento de operações em máquinas paralelas e idênticas, descrito por Kis e Kovács

(2012), pelo que se torna importante fazer uma revisão dos conceitos específicos associados a este problema.

2.2 Planeamento e Escalonamento

O fabrico, ou produção, de uma gama de produtos é, geralmente, determinado com base em dois processos: o planeamento e o escalonamento de produção. Estes processos são duas atividades separadas, com funções diferentes, apesar de terem o mesmo fim (Shen *et al.*, 2006).

Maravelias e Sung (2009) distinguem o planeamento em três tipos distintos, o estratégico, o tático e o operacional, sendo que este último ao nível da produção é designado por escalonamento (figura 1).

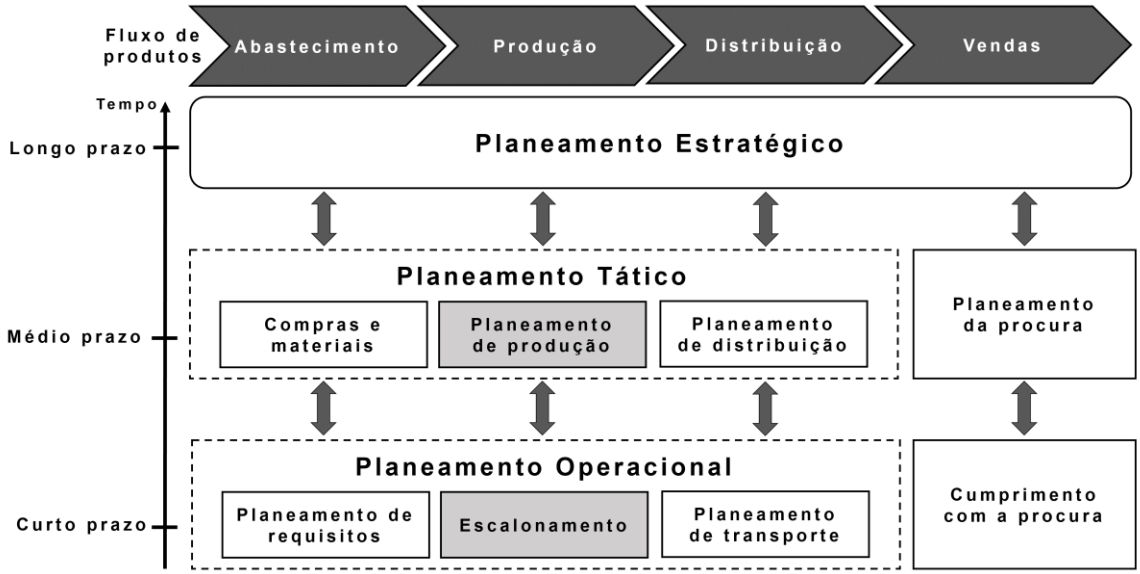


Figura 1 – Cadeia de abastecimento (Adaptada de Maravelias e Sung, 2009)

O planeamento estratégico tem uma escala de tempo de longa duração, na maior parte dos casos, meses ou anos. No entanto, a escala de tempo varia de acordo com a indústria em causa e o tipo de intervenções associadas às decisões. Este tipo de planeamento passa por definir novas estratégias para preservar a capacidade competitiva através de processos com vista a mudanças no negócio atual como, por exemplo, o lançamento de novos produtos, a construção de novas instalações, entre outras. Responde às perguntas: *O que produzir?*, *Como produzir?*, *Produzir com o quê?*. Isto é, define o objeto ou produto, os métodos e os meios. Já o planeamento tático é de médio prazo e, portanto, assume a função de intermediar os

objetivos estratégicos, que são mais amplos, em objetivos menos genéricos e mais detalhados. A sua principal preocupação é definir metas ou datas para a produção, respondendo à pergunta: *Para quando?*, definindo e controlando prazos.

O planeamento estratégico centra-se essencialmente nos requisitos geométricos e tecnológicos das tarefas. Pode até ser considerado como um processo distante do que acontece, na realidade, no local de operações. Como é referido por Shen *et al.* (2006), o planeamento pode partir de pressupostos irreais, como supor a disponibilidade de mais recursos e ter em conta uma visão do sistema produtivo mais calmo do que efetivamente se verifica. Não tem por base a carga de trabalho existente numa máquina em tempo real, nem a própria dinâmica presente no espaço fabril.

Quanto ao escalonamento, é um processo de curto prazo, que se preocupa mais com o imediato. É também conhecido como agendamento ou calendarização. Este processo procura responder a perguntas como: *Quanto?*, *Quando?*, *Onde?*, *Quem?*, procurando saber a quantidade, o momento, o local e as pessoas envolvidas. É o procedimento que atribui os trabalhos às máquinas ou recursos (afetação) e/ou define o sequenciamento das mesmas ao longo do tempo, procurando encontrar a forma mais otimizada possível. Determina a altura mais adequada para executar cada tarefa, tendo em conta as limitações das capacidades dos recursos existentes, bem como, as relações temporais entre os processos. Os diferentes modos com que se podem fazer estas atribuições podem afetar a otimização do escalonamento atendendo a diferentes critérios ou objetivos. Tais como, os custos associados, os atrasos e/ou adiantamentos, utilização de mão de obra, entre outros. Os problemas de escalonamento não são exclusivos dos espaços fabris. Eles existem um pouco por todas as organizações. Por exemplo, em hospitais na alocação de operações planeadas, em instituições de ensino na formulação dos horários, aeroportos e empresas de transporte (Shen *et al.*, 2006).

O problema abordado nesta dissertação está relacionado com a gestão da produção e é composto por duas fases, o planeamento e o escalonamento. Estas são resolvidas simultaneamente de uma forma integrada em que o *output* da primeira fase é o *input* da segunda (Rietz *et al.* 2016). No sentido de poder vir a ter cenários mais próximos da realidade e, consequentemente, mais eficazes, deve ser considerada esta integração. Estas fases sendo, inicialmente, abordadas sem qualquer correlação podem resultar em duas soluções ótimas e, depois do seu cruzamento, podem resultar em algo abaixo do esperado (solução não ótima). No entanto, se deixarmos de ver o problema como sendo composto por duas fases separadas e individuais poderemos chegar a uma solução globalmente ótima. Assim, mesmo num caso em que haja alguma perturbação na procura, a ligação entre os diferentes níveis é fundamental

para uma reorganização. Os sistemas que têm em vista a otimização destes processos são cada vez mais procurados e importantes para as empresas, já que lhes permitirá aumentar a produtividade e rentabilidade, podendo assim competir melhor com o mercado atual.

Em suma, para o nosso projeto, o planeamento consiste em determinar as tarefas que serão processadas em cada período de tempo e este terá de ser resolvido para o horizonte de tempo definido. O escalonamento consiste em atribuir as tarefas às máquinas disponíveis em cada período de tempo de acordo com suas datas de disponibilidade correspondentes da forma mais eficiente com base no critério definido. Michael Pinedo (2008) menciona que o *"*escalonamento é um processo de tomada de decisão que é usado regularmente em muitas indústrias de manufatura e serviços. Trata-se da alocação de recursos para tarefas em determinados períodos de tempo e o seu objetivo é otimizar um ou mais objetivos".

Esta abordagem integra o planeamento de médio prazo (tático) com o planeamento de curto prazo (componente operacional - escalonamento).

2.3 Classificação de Problemas de Escalonamento

Uma das primeiras classificações de problemas de escalonamento surge no livro "*Theory of Scheduling*" de Conway, Maxwell e Miller, em 1967. Esta aponta que existem quatro tipos de informação que determinam a classificação deste género de problemas:

A. As tarefas e operações que são sujeitas a processamento;
B. O número e tipos de máquinas que compõem o espaço para o processamento;
C. As restrições existentes para as atribuições;
D. Os critérios pelos quais é avaliado o escalonamento.

O conjunto da informação destes quatro parâmetros é apresentado através de uma quádrupla A/B/C/D, em que A, B, C e D representam as informações enumeradas acima, usando uma notação própria de problemas de escalonamento também referida pelos mesmos autores.

Ao longo dos tempos várias redefinições foram feitas como, por exemplo, em Graham *et al.* (1979). Neste a classificação dos problemas aparece através da combinação de três campos principais: caraterísticas das máquinas, caraterísticas dos trabalhos e a função objetivo a ser otimizada, representadas por $\alpha \mid \beta \mid \gamma$, respetivamente.

O primeiro campo ou conjunto (α) aparece representado com uma notação de entrada única, isto é, apenas pode contemplar um valor. No entanto, pode ser visto como uma

combinação de dois sub-conjuntos $\alpha = \alpha_1\ \alpha_2$ que vão caraterizar a disposição do espaço fabril (*machine environment*) e a quantidade de recursos (máquinas).

O primeiro sub-conjunto (α_1) diz respeito às diferentes configurações dos problemas existentes. De uma forma geral, podemos dividir em dois casos, máquina única (a) e múltiplas máquinas (b).

O segundo sub-conjunto, o número de máquinas (α_2), é um número inteiro positivo *m*, $\alpha_2 = m$, que representa o número de máquinas quando constante; $\alpha_2 = Ø$, caso o valor de *m* seja variável.

(a) Máquina Única (*Single Machine*)

- **$\alpha_1 = Ø$**. O valor de α_2 nesta situação é, necessária e evidentemente, igual a 1.
  - Trata-se da configuração mais simples e elementar, pelo que tem a importância de ser o caso de base para ambientes mais complexos. Neste apenas se coloca a questão de quando se deve executar cada tarefa – problema de sequenciamento.

Para os problemas de máquina única o valor de $\alpha \in \{1\}$.

(b) Múltiplas Máquinas

Nestas configurações para além de se equacionar o problema de sequenciamento tem, também, a questão: *Onde serão os trabalhos executados?* – Problema de afetação.

- **Máquinas Idênticas em Paralelo (*Identical Parallel Machines*): $\alpha_1 = P$**
  - As máquinas disponíveis são equivalentes e podem executar as mesmas funções de transformação. Qualquer tarefa pode ser processada por qualquer uma das máquinas, salvaguardando não haver qualquer restrição neste sentido. A existirem restrições são declaradas no campo β.
- **Máquinas Uniformes em Paralelo (*Uniform Parallel Machines*): $\alpha_1 = Q$**
  - Categoria muito parecida com a anterior. Difere no facto de as máquinas não serem equivalentes, apresentam velocidades de execução diferentes entre si.

- **Máquinas Não Relacionadas em Paralelo (*Unrelated Parallel Machines*): α_1 = R**
  - Máquinas em paralelo que apresentam velocidades de execução diferentes para cada tarefa. Caso a velocidade de processamento seja independente da tarefa, então estamos perante o caso anterior.
- **Linha Aberta (*Open Shop*): α_1 = O**
  - Neste esquema não há restrições ao fluxo ou trajeto das tarefas ao longo das máquinas. Cada trabalho é independente dos outros e pode seguir o seu próprio percurso.
- **Linha Geral (*Flow Shop*): α_1 = F**
  - Linha de produção na qual as máquinas estão dispostas em série, onde o trabalho pode ser processado em fases sucessivas sem a inversão do fluxo.
- **Oficina (*Job Shop*): α_1 = J**
  - Configuração que não apresenta qualquer tipo de restrição relativamente ao sentido do fluxo no espaço fabril, nem existe nenhum padrão comum nos movimentos dos materiais. No entanto, cada trabalho tem de obrigatoriamente passar por cada máquina pelo menos uma vez.

Michael Pinedo (2002), refere mais dois ambientes. Um consiste na utilização do ambiente de máquinas paralelas em conjunto com o ambiente Linha Geral e outro com a Oficina. Todas as outras considerações são idênticas, a grande diferença para os dois últimos ambientes expressos é que nestas novas situações, a que o autor designa por **Linha Geral Flexível** (*Flexible Flow Shop*) e **Oficina Flexível** (*Flexible Job Shop*), o número de máquinas *m* não é considerado. Ao invés disso, é tido em conta um valor, inteiro e positivo, *c*, que representa o número de centros de trabalho. Estes têm disponível um número de máquinas paralelas para os vários processos envolvidos. A notação para estes dois tipos de problemas são α = FFc para o primeiro caso e α = FJc para o segundo.

Assim sendo, para o primeiro campo de classificação em múltiplas máquinas temos que $\alpha \in$ {P, Pm, Q, Qm, R, Rm, O, Om, F, Fm, FFc, J, Jm, FJc}.

O segundo campo (β) pode ter mais do que um valor em simultâneo e representam as caraterísticas que devem ser consideradas nos trabalhos sujeitos a processamento (restrições e

condicionamentos). Caso não exista qualquer restrição $\beta \in \{ \emptyset \}$. Graham *et al.* (1979) apenas consideram seis caraterísticas relativas aos trabalhos. Entretanto, Pinedo (2002) apresenta um conjunto mais alargado para este campo β (em que j e k são exemplos de trabalhos):

- Datas de lançamento (*release time*) – r_j
- Tempo de configuração dependente da sequência (*sequence dependent setup time*) – s_{jk}
- Preempções (*preemptions*) – prmp
- Avarias das máquinas (*breakdowns*) – brkdwn
- Restrições de elegibilidade da máquina (*machine eligibility restrictions*) – M_j
- Permutação (*permutation*) – prmu
- Bloqueio (*blocking*) – block
- Sem espera (*no-wait*) – nwt
- Recirculação (*recirculation*) – recrc

Caso existam mais restrições associadas aos trabalhos podem ser aqui consideradas. As datas esperadas para o término dos trabalhos geralmente não são consideradas aqui porque, na maioria dos casos, a função objetivo é suficiente para entender esse aspeto. Tal como refere Pinedo (2002): "Qualquer outra entrada que possa aparecer no campo β é autoexplicativa" e as "datas esperadas, em contraste com as datas de lançamento, geralmente não são explicitamente especificadas neste campo; o tipo de função objetivo fornece as indicações suficientes se os trabalhos tiverem datas esperadas".

O terceiro campo (γ) representa a função objetivo. É o fator que determina a avaliação de cada solução ou alternativa, por isso, acaba por ter um papel fundamental e decisivo. Existem medidas regulares, cuja função é do tipo não decrescente, como minimizar o *makespan* ($\gamma = C_{max}$) medida bastante utilizada e, segundo Jain e Meeran (1999), foi das primeiras medidas a ser aplicada nos inícios de 1950. As medidas com base nos atrasos, como minimizar o atraso máximo dos trabalhos ($\gamma = L_{max}$), são também medidas regulares. Com a introdução de novas formas na gestão dos inventários, seguindo políticas como o *Just-In-Time*, levou a que medidas não regulares começassem a ter uma maior atenção por parte dos investigadores (Gordon *et al.*, 2002). A penalidade que conjuga o adiantamento ou a antecipação da realização dos trabalhos com os atrasos é um exemplo de uma medida não regular, visto que a primeira parte relativa à penalização por antecipação não é crescente ao longo do horizonte temporal considerado para o problema. Neste tipo de função há

penalização associada se os trabalhos acabarem mais cedo do que a data prevista ou depois desta. Como podemos ver em Panwalkar *et al.* (1982) e Baker e Scudder (1990), os trabalhos têm uma data limite e o objetivo é de encontrar a melhor solução que minimize a função de penalidade total. Esta é baseada na data em que o processo é concluído tendo em conta a data limite, ora como atrasado ora como adiantado. Esta última função de penalidade é também seguida por Kis e Kovács (2012) como a sua função objetivo e, consequentemente, também por nós considerada. Panwalkar *et al.* (1982) referem que os custos variam em função da data limite para a realização do trabalho, sendo que o custo de haver um atraso está mais relacionado com os objetivos do cliente, enquanto o custo de existir um adiantamento está relacionado com os objetivos de produção e, consequentemente, da empresa. Um trabalho vai ficar em espera se for realizado bastante antes da data pedida pelo cliente o que vai ter como consequência um valor acrescentado do produto bastante mais baixo (Abdulmalek e Rajgopal, 2007).

De acordo com a classificação de problemas de escalonamento definida por Graham *et al.* (1979) e também por Pinedo (2002), podemos definir o nosso problema em estudo como:

1. **Caraterísticas das máquinas (α)**

   Problema em que podemos ter *m* máquinas paralelas idênticas a poderem executar qualquer uma das tarefas de cada vez. Temos, portanto, α = Pm.

2. **Caraterísticas dos trabalhos (β)**

   Este campo contempla as restrições impostas ao problema por parte das tarefas a serem executadas. No nosso caso, apenas temos de garantir que os trabalhos só são executados depois de disponíveis, isto é, depois da sua data de lançamento. Pelo que, $\beta = \{ r_j \}$

3. **Objetivo (γ)**

   O critério para o qual a solução do nosso problema se tem de mostrar eficiente é a minimização da função penalidade descrita, em função de t_j (período de tempo em que é executada a tarefa *j*), como:

$$f(t_j) = \sum_{j=1}^{N} (e_j \times E + l_j \times T)$$

Onde e_j equivale à penalidade de adiantamento associada ao trabalho *j*, d_j é a data esperada para a conclusão do trabalho *j*, *E* é o *earliness* ($E = \max\{0, d_j - t_j\}$), l_j

é a penalidade de atraso associada ao trabalho *j*, e *T* é o *tardiness* ($T = \max\{0, t_j - d_j\}$)

Assim sendo, o nosso problema é classificado como:

$$Pm \mid r_j \mid f(t)$$

Mais modelos e classificações surgem na tentativa de conseguir abranger a totalidade dos problemas de escalonamento existentes. Os problemas de escalonamento passaram a ser vistos de perspetivas diferentes, cada caso tem as suas *nuances* e devem ser vistos por aquilo que mantêm em comum uns com os outros. Não só pelo modo como se encontram as soluções, mas pelos tipos de modelos considerados na literatura e também pelos critérios mais procurados pelos investigadores (Nagar *et al.*, 1995).

Assim, Nagar *et al.* (1995) desenvolveram um sistema que considera a natureza dos problemas, a configuração do espaço, a metodologia da solução, a medida de performance e o critério ou critérios como as caraterísticas que definem a classificação do problema (figura 2).

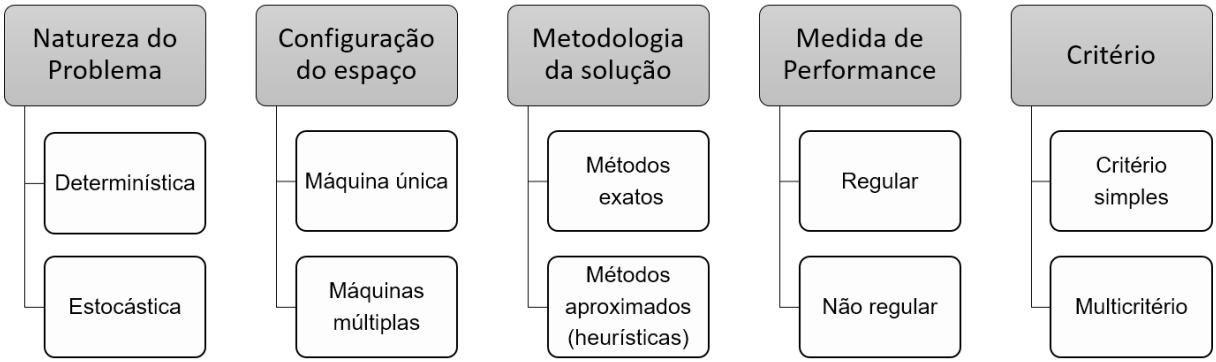


Figura 2 – Classificação de problemas de escalonamento proposto por Nagar *et al.* (1995)

Relativamente à **natureza dos problemas**, esta pode ser determinística ou estocástica (Nagar *et al.*, 1995). Se os tempos de processamento dos trabalhos forem fixos, ou se forem determinados com certeza, então o problema é de natureza determinística. Ao invés disso, se os tempos variarem, estamos perante problemas estocásticos. Essa variação pode acontecer por diversos motivos, entre os quais, o efeito da aprendizagem na elaboração das tarefas pelo operador que ao ganhar alguma experiência no processamento poderá reduzir o tempo de execução, *learning effects* (Biskup, 1999), ou algo que cause o inverso, como o cansaço do operador que o levará a demorar mais tempo do que o esperado. É compreensível que os modelos estocásticos conseguem fazer um retrato mais correto e fiável do que realmente

acontece nos processamentos. Para garantir estas variabilidades os modelos usam algumas variáveis com distribuições de probabilidades (Nagar *et al.*, 1995).

A **configuração do espaço** corresponde, praticamente, ao *machine environment* que vimos anteriormente.

Os **métodos usados** para chegar às soluções são também um fator para categorizar os problemas desta perspetiva de Nagar *et al.* (1995). As metodologias podem ser exatas ou aproximadas. Métodos exatos, como o próprio nome indica, encontram sempre o valor ótimo para o problema, mas para problemas de elevada dimensão essa solução poderá demorar imenso tempo a ser encontrada e, portanto, há a necessidade de recorrer a métodos que não equacionem todas as hipóteses de solução. No sentido de poderem reduzir significativamente o tempo computacional dos algoritmos, várias heurísticas e meta-heurísticas foram desenvolvidas. Nagar *et al.* (1995) dizem mesmo que estas heurísticas foram projetadas como uma solução intermédia para dar resposta ao conflito entre a qualidade da solução e o tempo computacional envolvido.

A **medida de performance** pode ser vista como o equivalente do campo γ da primeira classificação apresentada por Graham *et al.* (1979). Trata-se da função objetivo usada como medida.

Nagar *et al.* (1995) referem que os problemas e indefinições diárias nas empresas levam-nas a terem de decidir de forma rápida e acertada. Para isso, devem ter bem definidos os seus objetivos, tendo em conta as restrições que lhes são impostas. Os agentes de decisão, isto é as pessoas com a responsabilidade e a competência para tomarem as decisões, devem considerar convenientemente os critérios antes de decidirem por qualquer uma das alternativas. Uma solução que é ótima para um dado critério, pode ser uma alternativa fraca para um outro. Estas relações de compromisso ou relações onde há conflitos de escolhas (*trade-off*) são extremamente úteis para o agente decisor. Os problemas que têm em conta mais do que um fator de decisão, ou seja, mais do que um critério (**multicritério**), conseguirão aproximar-se mais daquilo que é a realidade. Nagar *et al.* (1995) consideram surpreendente que a investigação realizada sobre problemas de escalonamento com apenas um critério é substancialmente superior a uma que possa retratar muito melhor os factos.

Nas investigações elaboradas até aos finais de 1980, era prática comum a função objetivo só ter em conta um critério. No entanto, um sistema produtivo assenta em vários critérios que deveriam ser considerados. Tais como: o inventário, o cumprimento das datas de entrega, a carga existente nos centros de trabalho, entre outros (Hoogeveen, 2005).

Por exemplo, uma empresa que siga uma política com base no critério de inventário zero (conseguindo reduzir os custos inerentes ao *stock*), se apenas considerar esse critério, irá ter níveis baixos de *stock*, possivelmente, através de um sistema *pull* que só produz quando existir uma ordem nesse sentido. No entanto, a resposta aos clientes poderia vir a não ser a melhor, não os satisfazendo muitas das vezes com a rapidez necessária. Para conseguir baixar o tempo de resposta poderá ser necessário ter produtos disponíveis mesmo que não haja um pedido (sistema *push*).

Os gráficos a) e b) da figura 3 mostram a relação da quantidade dos produtos em *stock* com o custo e o tempo de resposta, respetivamente. O gráfico c) diz respeito à combinação dos dois critérios.

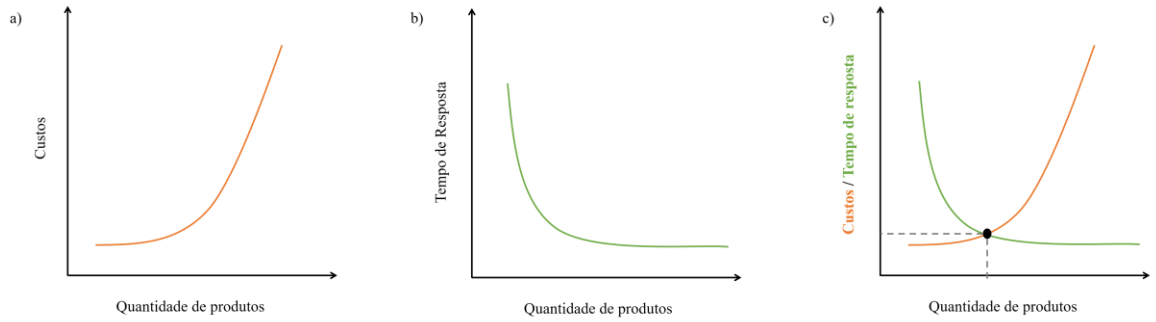


Figura 3 – Situação de conflito de escolha

a) Relação entre o custo e a quantidade de produto; b) Relação entre o tempo de espera do cliente e a quantidade de produto; c) Gráfico *trade-off* entre o custo/tempo de espera e a quantidade de produto.

Hoogeveen (2005) conclui que se se tiver em consideração apenas um critério, então o resultado é provável que seja desequilibrado, independentemente do critério considerado. Afirma ainda que é necessário chegar a um compromisso aceitável entre todos os critérios mais importantes. Estas conclusões levaram a uma maior investigação do escalonamento multicritério.

Tendo em conta as caraterísticas específicas do nosso problema, este envolve o problema de escalonamento em máquinas paralelas idênticas. Com o objetivo de o resolver vamos seguir técnicas associadas a problemas de corte e empacotamento que se comprovaram ser eficientes para este tipo de problemas, em Coffman *et al.* (1978). Em ambos há o objetivo de distribuir as tarefas pelos recursos existentes. Assim, torna-se pertinente fazer uma análise aos problemas de empacotamento existentes na literatura.

2.4 Problemas de Corte e Empacotamento

Os problemas de corte e empacotamento (C&P) são de grande interesse para muitas aplicações práticas e imensas pesquisas. O que leva as empresas a dedicarem grandes incentivos a esta área no sentido de chegar a soluções mais eficazes nos seus processos. Problemas de corte e empacotamento não são de agora e aparecem numa ampla variedade de indústrias. A primeira formulação de problemas de corte surgiu pelo economista russo Kantorovich (1960). Mais tarde, Gilmore e Gomory (1961) introduziram novos conceitos para este tipo de problemas de otimização combinatória, utilizando uma nova técnica de programação linear com o objetivo de tornar os problemas computacionalmente mais viáveis e contornar o facto de se tratarem de problemas NP-Difíceis (Haessler e Sweeney, 1991) (Delorme *et al.*, 2016).

Os problemas de corte e empacotamento têm caraterísticas em comum na sua estrutura. Ambos têm o propósito de dividir algum recurso em bocados mais pequenos de maneira a minimizar o volume ocupado (casos de empacotamento) ou minimizar a quantidade do recurso necessário (casos de corte). Em problemas de corte, o recurso é normalmente algum tipo de material físico como, por exemplo, uma chapa de alumínio ou vidro, tecido ou um rolo de papel. O espaço vazio é considerado o recurso dos problemas de empacotamento, como no caso do espaço disponível num meio de transporte (camião, avião ou barco), espaços de armazéns ou de paletes (Sweeney e Paternoster, 1992).

Num problema de corte, o objetivo é conseguir o corte do máximo de pedaços com o mínimo de recurso possível. Já num problema de empacotamento, pretende-se guardar o maior número de material no menor espaço possível. Ou seja, podemos observar a existência de uma forte ligação entre os dois tipos de problemas, pelo que devem ser vistos em conjunto (Dyckhoff e Finke, 1992). A figura 4 mostra-nos a dualidade que há entre o corte e empacotamento com o material e o espaço, comprovando que podem ser vistos como problemas análogos.

Dependendo da abordagem ao problema é que conseguimos distingui-los. No entanto, do ponto de vista de formulação são idênticos.

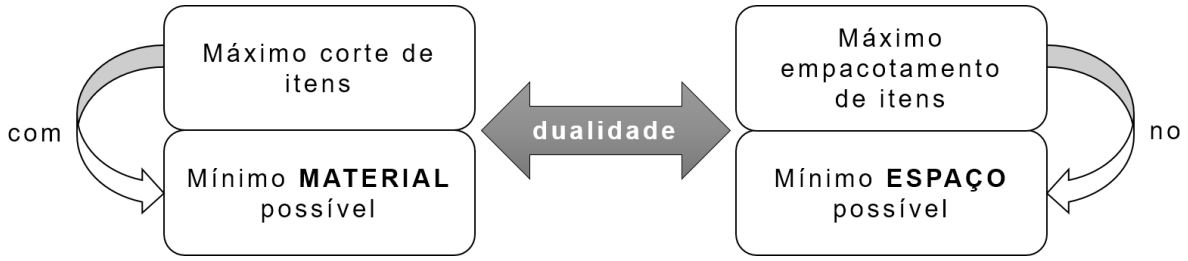


Figura 4 – Dualidade entre o corte e empacotamento com o material e o espaço

Wascher *et al.* (2007) consideram que neste tipo de problemas temos dois conjuntos de elementos, o conjunto dos grandes objetos e o dos pequenos objetos. Podem ser definidos em uma, ou duas, ou três, ou até num número maior (*n*) de dimensões geométricas.

Para estes problemas, segundo os autores, de uma forma geral e simplista, tem-se como objetivo selecionar pequenos objetos (todos, ou alguns) de modo a formarem um novo subconjunto capaz de ser atribuído a um, ou mais, conjuntos dos grandes objetos. Para isso, os pequenos objetos não se podem sobrepor e têm de ficar completamente dentro do objeto grande, preservando assim a condição geométrica dos conjuntos.

Assim sendo, podemos diferenciar cinco sub-problemas no problema geral (Wascher *et al.*, 2007):

1. Seleção dos grandes objetos;
2. Seleção dos pequenos objetos;
3. Definição dos subconjuntos a efetuar nos pequenos objetos;
4. Definir a atribuição dos agrupamentos dos pequenos objetos aos grandes objetos;
5. Definir o esquema de disposição dos objetos relativamente à sua condição geométrica.

Alguns casos, mais particulares, não consideravam todos estes sub-problemas ou consideravam ainda mais.

2.4.1 Tipologia de Dyckhoff

Classificações variadas surgiram na literatura, entre as quais a tipologia desenvolvida por Dyckhoff (1990). Foi vista como o primeiro instrumento para organizar e categorizar os problemas de corte e empacotamento.

Dyckhoff (1990) propôs uma tipologia para os problemas de corte e empacotamento com base nas suas principais caraterísticas. A tipologia é semelhante a uma classificação. No entanto, uma tipologia por si só pode não ser suficiente para caraterizar um problema

específico. Como o próprio nome indica, é a ciência que estuda os tipos e serve para definir diferentes categorias, mas pode não ser uma categorização completa e pode ser distorcida. Apesar destas particularidades negativas, contribuiu para unificar algumas definições e notações, o que se apresenta como um elemento simplificador para a investigação nesta área. Ao existirem critérios para designar os diferentes tipos de problema evitam-se possíveis más interpretações (Wascher *et al.*, 2007).

Os problemas de corte e empacotamento foram assim classificados, por Dyckhoff (1990), em quatro critérios. O primeiro critério é a dimensionalidade dos objetos, o segundo é o tipo de atribuição dos pequenos objetos em relação aos grandes, o terceiro critério diz respeito à variedade dos objetos grandes e, por último, o quarto é a variedade dos objetos pequenos, como mostra a figura 5.

1. Dimensionalidade
(1) Unidimensional
(2) Bidimensional
(3) Tridimensional
(N) N- dimensional com N > 3

2. Tipo de atribuição
(B) Todos os objetos e uma parte dos itens
(V) Uma parte dos Objetos e Todos os itens.

3. Variedade de objetos grande
(O) Um objeto
(I) Objetos de figuras idênticas
(D) Objetos de figuras diferentes

4. Variedade de pequenos objetos
(F) Poucos objetos (de figuras diferentes);
(M) Muitos objetos de muitas figuras diferentes;
(R) Muitos objetos com relativamente poucas figuras diferentes;
(C) Figuras congruentes.

Figura 5 – Tipologia de Dyckhoff (1990)

(1) Dimensionalidade

A dimensionalidade é o número real mínimo de dimensões necessárias para descrever o problema. Dyckhoff (1990) considera-a a característica mais importante. Cada problema pode ser: unidimensional (1), bidimensional (2), tridimensional (3) e, no caso de mais do que três dimensões, um problema multidimensional (n).

(2) Tipo de atribuição

Dyckhoff (1990) faz distinção entre dois tipos de atribuição, o tipo B e V (a denominação por estas letras surge por serem as letras inicias de duas palavras alemãs, *Beladeproblem* e *Verladeproblem*, respetivamente. O tipo de atribuição pode-se definir como sendo a relação existente entre os pequenos objetos e os grandes objetos.

O tipo B define que todos os objetos grandes têm de ser utilizados, isto é, um conjunto de pequenos objetos deve ser selecionado para preencher o máximo dos objetos grandes. Já o tipo V considera a utilização de todos os pequenos objetos com uma seleção adequada de objetos grandes.

Este conceito e esta diferenciação devem ser claros para caraterizar o problema, tal como nos sugere Dyckhoff (1990): "Para ambos, os grandes objetos e os pequenos objetos, tem de ser claro se somente uma seleção ou todos eles têm de ser atribuídos aos padrões (não triviais) correspondentes.".

(3) Variedade dos objetos grandes

Relativamente à variedade dos objetos grandes são considerados três tipos. Um tipo em que só existe um objeto grande (O); outro que é composto por vários objetos grandes, todos eles idênticos (I); e um também composto por vários objetos grandes, mas com figuras diferentes (D).

(4) Variedade dos objetos pequenos

Os pequenos objetos podem ser distribuídos por 4 grupos. Um grupo onde existem, relativamente, poucos objetos pequenos com figuras diferentes (F); o grupo com muitos objetos pequenos em que as suas formas e figuras são bastante diferentes (M); um grupo em que estão muitos objetos pequenos, mas alguns objetos são diferentes (R); e, por último, o grupo em que todos os objetos são idênticos, com figuras congruentes (C).

Segundo Dyckhoff (1990), através destes quatro critérios seria possível classificar de forma consistente os diversos problemas. Com os amplos desenvolvimentos na área, esta forma de organizar os problemas mostrou-se insuficiente.

Insuficiência da tipologia de Dickhoff

Apesar dos contributos que a tipologia de Dyckhoff (1990) trouxe no sentido de unificar os diferentes tipos de problemas na área de corte e empacotamento, esta não foi completamente aceite internacionalmente (Wascher *et al.*, 2007).

Aparece como sendo uma boa primeira estrutura básica comum, no entanto, ao longo dos anos, foram-lhe apontados alguns problemas ou limitações por vários investigadores.

Wascher *et al.* (2007) foram uns dos que enumeraram algumas desvantagens. Entre as quais:

↳ **A classificação pode não ser necessariamente única,** isto é, não ser exclusiva a um tipo de problema. Segundo Dyckhoff (1990), o problema de carregamento de um veículo pode ser classificado como 1/V/I/F e 1/V/I/M. Percebe-se que o modo para diferenciar o tipo F e o tipo M não é claro e é preferível ter uma classificação bem definida para este tipo de problemas. A classificação relativa à variedade dos pequenos objetos do tipo F surge apenas uma vez nos exemplos de problemas combinados de Dyckhoff (1990). O que prova que ao invés desta diferenciação deveria existir um único tipo que englobasse os tipos F e M (Wascher *et al.*, 2007).

↳ **Tipologia parcialmente inconsistente**, pois pode ter resultados confusos explicados com algumas considerações pessoais de Dyckhoff (1990).

↳ A aplicação da tipologia de Dyckhoff (1990) não resulta, necessariamente, em categorias de problemas homogéneos. **Tipologia confusa** que leva a classificar de igual forma situações diferentes. Ou seja, temos problemas diferentes, com métodos de solução diferentes, mas com uma classificação igual (Wascher *et al.*, 2007). Gradišar *et al.* (2002) consideram que utilizar a mesma classificação para duas situações não é adequado. “Isto significa que nos casos em que o material é de comprimentos diferentes existem duas situações diferentes que requerem duas abordagens de solução diferentes.” (Gradišar *et al.*, 2002)

↳ **O facto de existirem termos de origem alemã,** nomeadamente, os casos do tipo B (*Beladeproblem*) e V (*Verladeproblem*). O que para a comunidade científica se apresentou como um problema já que esta usa, por norma, a língua inglesa. Como veremos mais à frente, foi uma lacuna ultrapassada noutras classificações através da introdução de novas denominações mais apropriadas (em inglês) (Wascher *et al.*, 2007).

No sentido de poder fazer uma completa classificação de todos os problemas de C&P que foram sendo desenvolvidos ao longo dos anos, mesmo em áreas menos investigadas, Wascher *et al.* (2007) referem que a sua classificação "ajuda a identificar "manchas em branco", ou seja, áreas, em que nenhuma ou apenas muito pouca pesquisa tem sido feita. Finalmente, a tipologia sugerida também está aberta a novos tipos de problemas a serem introduzidos no futuro". Ou seja, de modo a colmatar algumas lacunas de outras classificações e estar também preparada para investigações futuras, foi sugerida uma nova tipologia por estes autores em 2007.

2.4.2 Tipologia de Wäscher *et al.*

Os problemas de corte e empacotamento são distinguidos, por Wascher *et al.* (2007), em dois tipos, elementares ou combinados. Estes podem servir de "base para o desenvolvimento de modelos, algoritmos e geradores de problemas, e para a categorização da literatura devem ser relativamente homogéneos".

Os problemas combinados são definidos com base em 5 critérios: dimensionalidade, tipo de atribuição, variedade dos objetos grandes, variedade dos pequenos objetos e a sua forma (Wascher *et al.*, 2007).

1. **Dimensionalidade**

   Neste critério, tal como na tipologia proposta por Dyckhoff (1990), são consideradas 1, 2 ou 3 dimensões para os problemas. Problemas com mais de 3 dimensões são vistos como variantes.

2. **Tipo de atribuição**

   São duas as situações consideradas neste critério, tal como acontecia na tipologia defendida por Dyckhoff (1990). No entanto, Wascher *et al.* (2007) propõem outras designações para o tipo de atribuição evitando os anteriores conceitos de origem alemã. Denominam assim os problemas, de um modo geral, como de maximização de saída (*output value maximisation*) e de minimização de entrada (*input value minimisation*).

   O primeiro é o caso em que um conjunto de pequenos itens é atribuído a um conjunto de grandes objetos. O objetivo é conseguir maximizar a produção de uma seleção de itens (pequenos objetos) que usem os grandes objetos. Sendo que não

existe uma seleção dos objetos maiores, mas sim o propósito de encontrar o melhor subconjunto de itens para um uso otimizado dos objetos grandes que são de quantidade limitada.

O segundo caso, de minimização de entrada, de forma similar ao primeiro também se trata de uma atribuição de pequenos itens a um conjunto de objetos grandes. No entanto, neste, o objetivo é acomodar todos os pequenos itens no objeto maior de forma a selecionar o subconjunto mínimo do objeto grande.

3. Variedade dos pequenos objetos

Os pequenos objetos ou itens podem ser classificados em três grupos. Temos o grupo de itens idênticos, o grupo de itens em que a sua variedade é pouco heterogénea e, por último, o conjunto de itens fortemente heterogéneos.

Os pequenos objetos que tenham a mesma forma e tamanho são considerados itens idênticos. Na classificação de Dyckhoff (1990) seria o equivalente ao tipo C.

Os itens com uma variedade fracamente heterogénea são aqueles que apresentam poucas diferenças entre si. Dentro destes podem-se agrupar em classes de itens aqueles que são mais ou menos semelhantes, de acordo com a sua forma e o tamanho. Dyckhoff (1990) considerava para este tipo de objetos o tipo R.

O conjunto de pequenos itens com uma variedade fortemente heterógena são aqueles em que na sua grande maioria são diferentes em relação à forma e tamanho. Pelo que haverá poucos itens idênticos, sendo estes analisados à parte. O tipo M e F da classificação de Dyckhoff (1990) são englobados neste conjunto de elementos.

4. Variedade dos objetos grandes

Wascher *et al.* (2007) consideram duas vertentes relativamente aos objetos grandes.

O caso em que o conjunto de objetos grandes tem apenas um elemento, com as suas dimensões a serem fixas ou variáveis. Este é equivalente ao tipo O da classificação idealizada por Dyckhoff (1990).

E um outro caso que considera vários objetos. Os objetos grandes podem ser idênticos, pouco heterogéneos e muito heterogéneos, tal como acontecia com os objetos pequenos. Este tipo de caraterização é mais aprofundada comparativamente à tipologia de Dyckhoff (1990) que apenas refere o tipo I e D, objetos grandes idênticos e diferentes, respetivamente.

5. **Forma dos pequenos objetos**

A forma dos pequenos objetos é uma caraterização adicional dos problemas. Este critério não era mencionado por Dyckhoff (1990). A forma distingue os pequenos itens em regulares e irregulares, em problemas bidimensionais e tridimensionais. São considerados objetos regulares os triângulos, retângulos, caixas, círculos, bolas, cilindros, entre outros.

Problemas em que existam pequenos objetos simultaneamente regulares e irregulares são vistos como variantes.

Problemas básicos, intermédios e refinados

O critério dimensionalidade está presente em todos os problemas do tipo básico, do tipo intermédio e do tipo refinado.

A) Tipos de problemas básicos

Para definir os tipos básicos de problemas de C&P, é utilizada a combinação de mais dois critérios da tipologia de Wascher *et al.* (2007): o tipo de atribuição e a variedade dos pequenos objetos. A existência dos tipos de problemas básicos permite uma nova terminologia para cada problema de acordo com as suas caraterísticas específicas, sendo estes nomes mais facilmente adotados, já que houve um rigoroso cuidado para não existirem ambiguidades. Para os autores, sempre que possível, "as categorias de problema são nomeadas de acordo com as notações existentes. Ao fazê-lo, os autores não pretendem apenas evitar interpretações desnecessárias, nem mesmo confusão, mas também melhorar o grau de aceitação entre os investigadores da área".

B) Tipos de problema intermédio (*Intermediate Problem Types*)

Para além dos problemas básicos e com o objetivo de incorporar a variedade dos objetos grandes, Wascher *et al.* (2007) reestrutura-os dando origem aos tipos de problemas intermédios.

Assim, os problemas básicos, acima referidos, são combinados com as diferentes caraterísticas dos objetos grandes, entre as quais: objeto grande único, vários objetos grandes pouco heterogéneos e vários objetos grandes muito heterogéneos.

Tipo de atribuição – maximização de saída

Problemas do tipo de atribuição de maximização de saída são caraterizados pela quantidade limitada dos objetos maiores. Pelo que se deve otimizar o seu uso de forma a produzir o máximo de itens de menores dimensões.

Como problemas básicos de maximização de saída temos:

> **Problema de empacotamento de itens idênticos (IIPP – *Identical Item Packing Problem*)**

Como o próprio nome indica, trata-se de um problema de atribuição de pequenos itens idênticos para um dado conjunto, limitado, de objetos grandes. Isto é, pretende encontrar a melhor disposição dos pequenos itens de forma a se conseguir colocar o maior número possível.

> **Problema de alocação (PP – *Placement Problem*)**

Wascher *et al.* (2007), no sentido de generalizarem os vários nomes existentes na literatura para os problemas deste tipo, usam um termo mais neutro. Designam por problema de alocação aos problemas de atribuição de pequenos itens pouco heterogéneos ao conjunto de grandes objetos. Tratando-se de um problema de maximização de saída, pretende acomodar da melhor forma os pequenos objetos, minimizando o desperdício dos objetos maiores.

Dentro desta categoria podemos distinguir três tipos de problemas intermédios:

- Problemas de alocação num único objeto grande
  (SLOPP – *Single Large Object Placement Problem*)
- Problemas de alocação em vários objetos grandes idênticos
  (MILOPP – *Multiple Identical Large Object Placement Problem)*
- Problemas de alocação em vários objetos grandes heterogéneos
  (MHLOPP – *Multiple Heterogeneous Large Object Placement Problem*)

> **Problema de mochila (KP – *Knapsack Problem*)**

Neste tipo de problemas os itens ou pequenos objetos são muito heterogéneos e são atribuídos aos objetos grandes (nesta caso particular o objeto grande é visto como uma espécie de mochila).

Estes são atribuídos ao conjunto de objetos grandes nas suas variantes (problemas do tipo intermédio). Tal como nos problemas de alocação, há a distinção dos objetos grandes em três tipos:

- Problema de mochila única
  (SKP – *Single Knapsack Problem*)
- Problema de várias mochilas idênticas
  (MIKP – *Multiple Identical Knapsack Problem*)
- Problema de várias mochilas heterogéneas
  (MHKP – *Multiple Heterogeneous Knapsack Problem*)

Tipo de atribuição – minimização de entrada

Nos problemas do tipo de atribuição de minimização de entrada, a quantidade dos grandes objetos é suficiente para todos os pequenos itens. O objetivo passa por acomodar todos os pequenos itens usando o mínimo possível do objeto maior.

Problemas básicos de minimização de entrada temos:

> **Problema com dimensão variável (ODP – *Open Dimension Problem*)**

Problemas em que, pelo menos, uma das dimensões dos grandes objetos é variável. Na literatura, geralmente, estes problemas são apenas discutidos para um único objeto grande e só é possível para problemas com duas ou mais dimensões.

> **Problema de corte de *stock* (CSP – *Cutting Stock Problem*)**

Neste tipo de problemas, os pequenos objetos são pouco heterogéneos. São todos atribuídos a uma seleção de grandes objetos (a menor possível) com o objetivo de os rentabilizar. Ao contrário dos problemas do tipo anterior, este tem os objetos grandes com dimensões fixas. Os problemas intermédios podem ter três vertentes, tendo em conta a variedade dos objetos grandes: objetos idênticos, objetos pouco heterogéneos e objetos fortemente heterogéneos.

- Problema de corte com *stock* de tamanho único
  (SSSCSP – *Single Stock-Size Cutting Stock Problem*)
- Problema de corte com *stock* de tamanho variado
  (MSSCSP – *Multiple Stock-Size Cutting Stock Problem*)
- Problema de corte residual
  (RCSP – *Residual Cutting Stock Problem*)

> **Problemas de empacotamento em caixas (BPP – *Bin Packing Problem*)**

Esta categoria de problemas caracteriza-se pela forte heterogeneidade dos pequenos objetos que devem ser alocados a um conjunto de objetos grandes tendo em vista minimizar a quantidade de objetos grandes utilizados. Os problemas de empacotamento de caixas é extremamente útil na prática e tem uma grande variedade de aplicações em várias áreas.

No que diz respeito aos problemas intermédios, os grandes objetos podem ser de apenas um elemento (objeto grande único), um conjunto de objetos grandes fracamente heterogéneo ou fortemente heterogéneo.

- Problema de empacotamento em caixas de tamanho único
  (SBSBPP – *Single Bin-Size Bin Packing Problem*)
- Problema de empacotamento em caixas de tamanho variável
  (MBSBPP – *Multiple Bin-Size Bin Packing Problem*)
- Problema de empacotamento de caixas residual
  (RBPP – *Residual Bin Packing Problem*)

C) Tipos de problemas refinados

Os tipos de problemas refinados consistem no uso de mais um critério conjugado com os problemas do tipo intermédio. O critério é a forma dos objetos pequenos (caso a dimensionalidade seja de duas ou três dimensões).

A estrutura de um problema refinado passa por definir:

- Dimensão ⇒ 1, 2, ou 3-dimensional;
- Forma ⇒ Ø, retangular, circular, …, irregular;
- IPT ⇒ Problema do tipo intermédio (***I**ntermediate **P**roblem **T**ypes*).

A figura 6 mostra em síntese a classificação dos problemas de corte e empacotamento segundo os critérios definidos por Wascher *et al.* (2007). Esta agrega as diferentes combinações que caracterizam os problemas do tipo básico, do tipo intermédio e do tipo refinado.

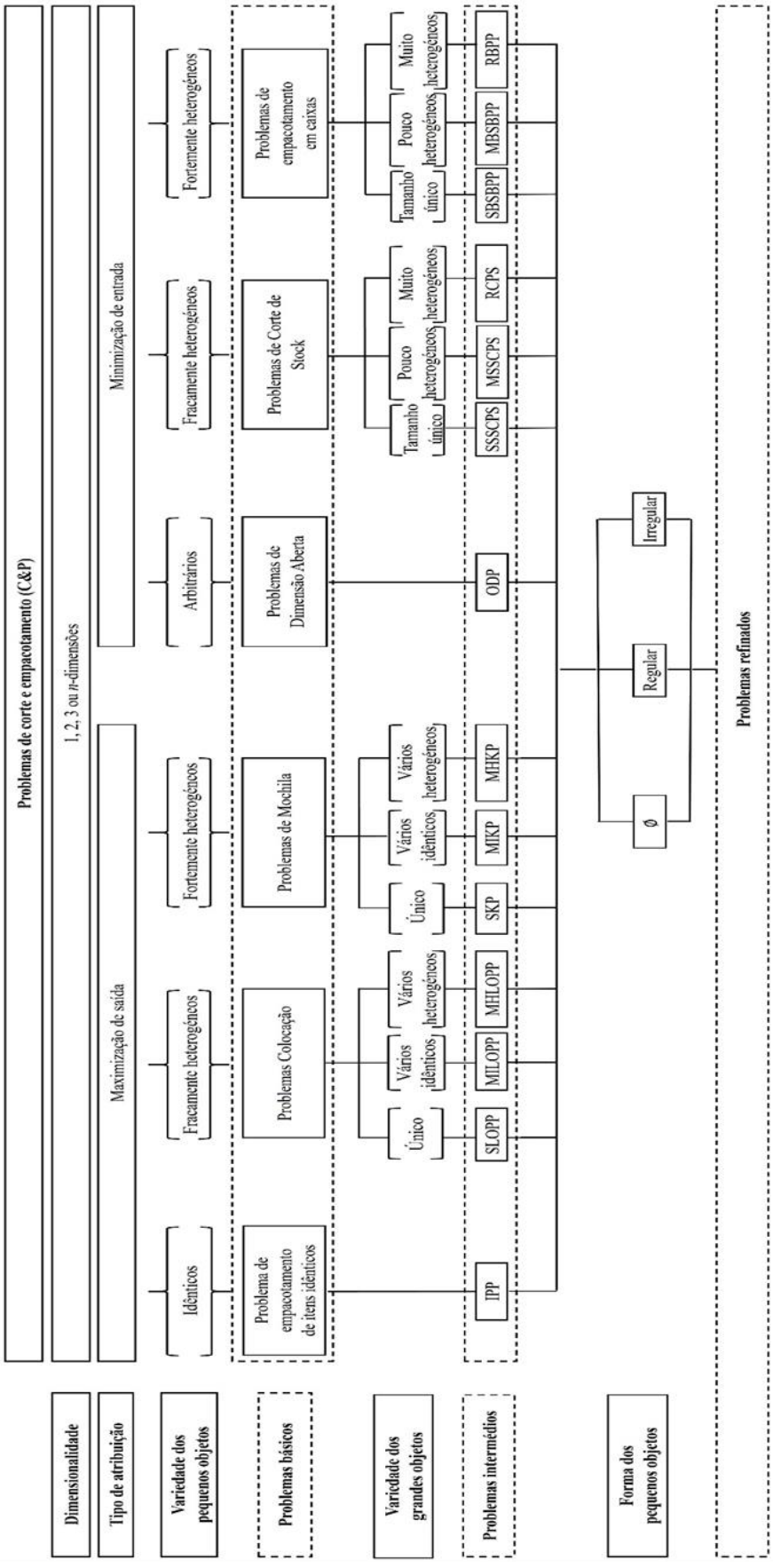


Figura 6 – Tipologia de Wäscher *et al.* (2007)

A associação de problemas de escalonamento com problemas de empacotamento é evidente. No nosso caso, podemos definir os pequenos objetos como as tarefas a serem processadas e os grandes objetos como os períodos de tempo disponíveis nas várias máquinas existentes.

Assim, definindo o problema de acordo com a tipologia de Wascher *et al.* (2007) temos um problema de 1-dimensão no que diz respeito à dimensionalidade. O tipo de atribuição é de minimização de entrada onde se pretende atribuir as tarefas a um subconjunto dos períodos de tempo (figura 7 – a). A variedade das tarefas, neste contexto, é definido pelo tempo de processamento das mesmas e os tempos de processamento podem ser muito heterogéneos (figura 7 – b).

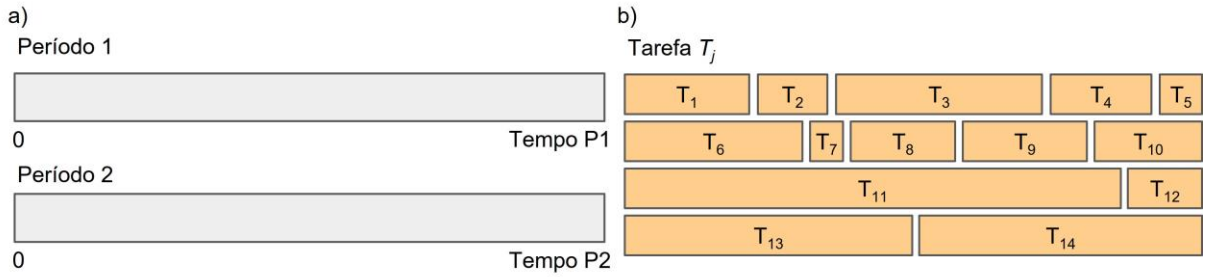


Figura 7 - a) Grandes objetos – períodos; b) Pequenos objetos – tarefas

Com os critérios já mencionados, podemos definir o tipo de problema básico como um problema de *bin-packing* (figura 8).

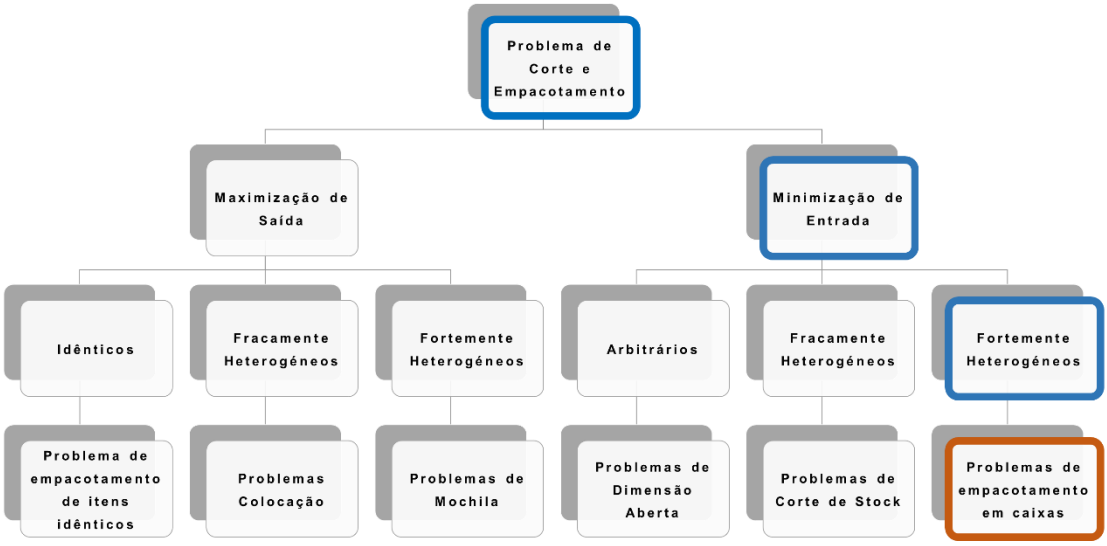


Figura 8 - Problemas Básicos

Relativamente ao critério da variedade dos grandes objetos (para nós, em específico, a variedade do tamanho dos períodos) trata-se de um problema com os grandes objetos de tamanho único. Isto é, e usando a figura 7 – a) como exemplo, o tempo de P1 = P2. Desse modo, como problema intermédio temos um problema de empacotamento em caixas de tamanho único (SBSBPP).

A forma dos pequenos objetos que se apresenta como uma das caraterísticas dos problemas refinados não existe, visto estarmos perante um problema de 1-dimensão.

Este problema de planeamento e escalonamento pode ser visto como um problema de corte e empacotamento de 1-dimensão em que a sua solução não exceda as *m* máquinas disponíveis. O escalonamento em máquinas paralelas idênticas e os problemas de corte e empacotamento são conhecidos por serem problemas de uma complexidade elevada.

As soluções encontradas, quer no planeamento, quer no escalonamento e, portanto, também na sua visão integrada, são uma tarefa extremamente desafiadora quanto mais pesada for a carga do sistema e mais restrições complexas existirem. No sentido de poder chegar a uma solução, fazem-se simplificações do que acontece na realidade ou do modelo (com agregações ou decomposições), utilizando-se métodos matemáticos ou métodos heurísticos.

2.5 Heurísticas

Para se conseguir encontrar uma solução para os problemas mais complexos temos duas formas de encará-los, através de métodos exatos ou métodos aproximados (heurísticas e meta-heurísticas).

Recorrendo aos primeiros, apesar da natureza dos problemas ser NP-Difícil, é possível em algumas situações especiais conseguir obter uma solução com alguma facilidade. Tal como referem Garey e Johnson (1978) depende da sua aplicação em cada caso específico.

Para problemas que não são de fácil resolução, em tempo considerado útil, as heurísticas podem apresentar-se como uma boa alternativa. As soluções encontradas através de algoritmos heurísticos não são necessariamente soluções ótimas, mas do ponto de vista do tempo de resposta é manifestamente mais rápido e podem ser adaptadas a uma grande variedade de aplicações (Pinedo, 2008).

No nosso projeto vamos abordar um problema que é caraterizado por ser complexo, através de métodos heurísticos de pesquisa local e de meta-heurísticas que recorrem a esses métodos, como é o caso da pesquisa por vizinhanças variáveis (VNS). Rietz *et al.* (2015),

também abordam a utilização de processos heurísticos com vista à resolução completa do problema em estudo (descrito por Kis e Kovács (2012)). Os autores descrevem e analisam heurísticas rápidas, de baixa complexidade, com abordagens construtivas e o impacto que diferentes regras na construção das soluções iniciais tem na solução final.

Problemas heurísticos baseados em pesquisa local são dos mais utilizados devido à simplicidade de implementação e apresentam resultados satisfatórios para muitas das situações em que são aplicados. Um problema de pesquisa local carateriza-se por ser um processo iterativo, que nos dá um ótimo local, com três ideias principais: gerar uma solução inicial, gerar soluções vizinhas e definir a medida de avaliação entre soluções. No entanto, um dos aspetos negativos destes métodos é que podem encontrar uma solução que é um ótimo local, mas ser ainda muito distante da solução ótima global, tal como mostra a figura 9.

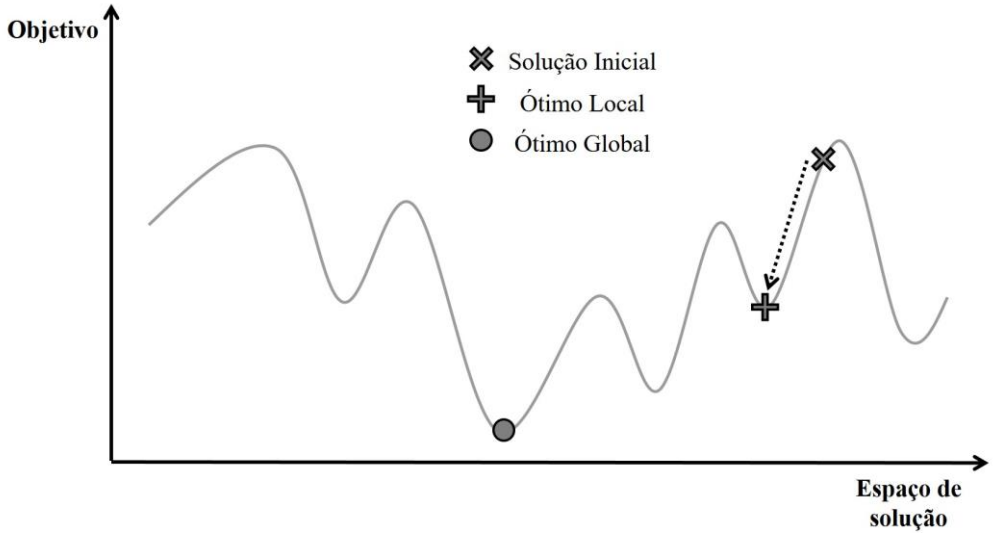


Figura 9 – Comportamento da pesquisa local

Em Aarts e Lenstra (2003), os autores referem que os algoritmos de pesquisa local podem ficar presos em ótimos locais pobres, principalmente, se a geração da vizinhança for simples. Contudo, a geração de vizinhanças mais complexas, por exemplo, envolvendo um maior número de trocas, podem aumentar muito significativamente o tempo de execução de um algoritmo que se espera rápido.

Facilmente se percebe a importância que a qualidade da solução inicial vai ter no espaço de solução analisado. Assim, na tentativa de verificar o maior espaço de soluções possível existem estratégias que ao invés de iterarem apenas uma vez com uma única solução inicial, são iteradas várias vezes com várias soluções iniciais. Às abordagens baseadas nesta

técnica dá-se o nome de *multistart* (Michiels *et al.*, 2007). O *multistart* é entendido como uma meta-heurística que complementa a heurística de pesquisa local.

O termo meta-heurística foi introduzido em 1986 por Glover, onde refere que a "pesquisa tabu pode ser vista como uma "meta-heurística" sobreposta a outra heurística". A etimologia da palavra "heurística" é da antiga palavra grega *heuriskein* que significava a arte de descobrir novas estratégias. A palavra "meta", que precede a palavra heurística, também é de origem grega sendo o seu significado "metodologia de nível superior" (Talbi, 2009).

Como referido anteriormente, a principal abordagem ao problema considerado nesta dissertação vai contemplar o recurso à meta-heurística VNS associada, obviamente, à pesquisa local. Esta meta-heurística surge pela primeira vez referida na literatura por Mladenović e Hansen (1997). Os autores deste "novo" conceito mencionam que: "Contrariamente à maioria dos outros métodos de pesquisa local, o VNS não segue uma trajetória, mas explora vizinhanças cada vez mais distantes da solução atual e salta de lá para uma nova se, e somente se, uma melhoria for feita". Nos últimos anos diversas aplicações do VNS têm sido estudadas, analisando a sua combinação com outros métodos.

A metodologia VNS, como o próprio nome indica, carateriza-se por mudanças de vizinhanças que procuram soluções melhores até encontrar o ótimo local, bem como tentar conseguir "saltar" dos vales de soluções onde o algoritmo poderia vir a ficar preso. Hansen *et al.* (2010) apontam três observações que se verificam no VNS:

I. Um mínimo local em relação a uma estrutura de vizinhança não é necessariamente um mínimo local de uma outra;
II. Um mínimo global é um mínimo local em relação a todas as estruturas de vizinhança possíveis;
III. Para muitos problemas os mínimos locais em relação a uma ou várias vizinhanças são relativamente próximos uns dos outros.

Várias são as abordagens possíveis com a meta-heurística VNS. Aquela que seguimos foi o método VNS básico (BVNS). Na figura 10 está representado um exemplo gráfico genérico de possíveis iterações de um método VNS básico com pelo menos três tipos diferentes de estruturas de vizinhança para uma mesma solução. A implementação e os passos do VNS básico será abordada com um maior detalhe no capítulo 4.

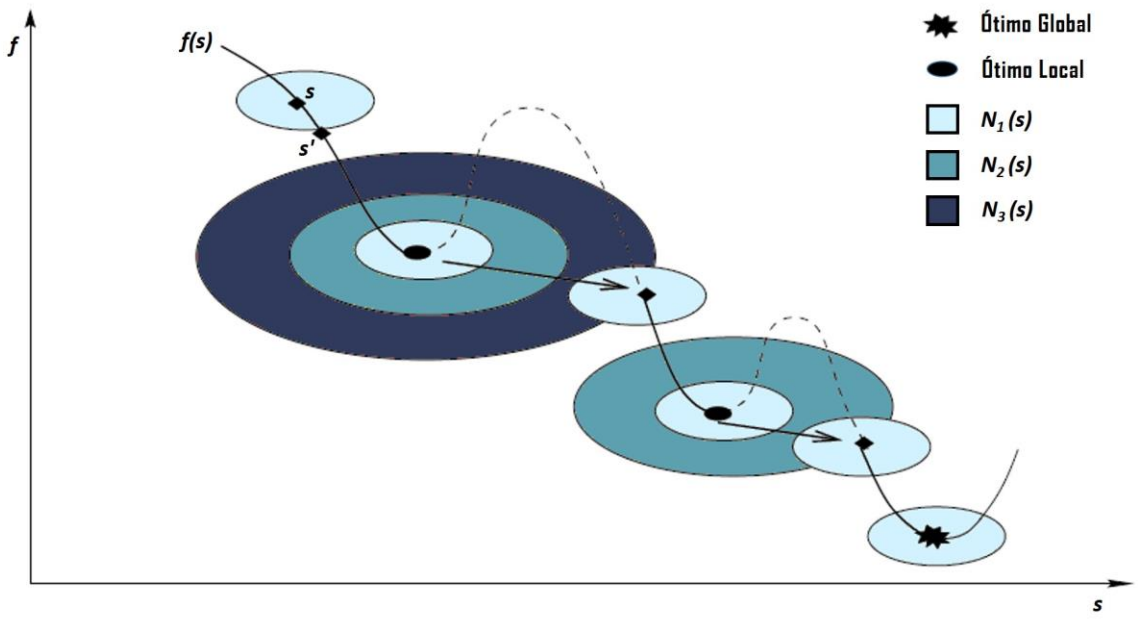


Figura 10 – Iterações de BVNS (Adaptada de Hansen *et al.* (2010)

Hansen *et al.* (2010) expõem, de forma discriminada, uma série de procedimentos comuns para qualquer implementação de métodos meta-heurísticos. Sintetizando, os passos a ter em conta são:

1) Familiarizar-se com o problema em questão;
2) Ler sobre o problema e os métodos de solução existentes na literatura;
3) Utilizar instâncias de teste;
4) Pensar na forma como a solução do problema será representada (estrutura de dados);
5) Gerar uma solução inicial;
6) Determinar a função objetivo do problema.

3. PROBLEMA DE PLANEAMENTO E ESCALONAMENTO

Nem sempre é fácil fazer um verdadeiro retrato das situações do mundo real e, tal como vimos, fazer simplificações dos problemas é muito comum. Os casos dos problemas de planeamento e escalonamento não são exceção, bem como os que incluem a sua integração.

3.1 Definição

O projeto realizado nesta dissertação, como referido anteriormente, tem por base um estudo recente de Kis e Kovács (2012) que considera o planeamento e escalonamento de forma combinada em máquinas paralelas e idênticas. O planeamento define os períodos em que as tarefas vão ser processadas e o escalonamento preocupa-se com a atribuição destas às máquinas disponíveis, tendo em conta as datas de lançamento (datas em que as tarefas estão prontas a ser executadas). Tudo com o objetivo de minimizar as penalidades incorridas pelas tarefas que acabem de ser processadas cedo ou tarde demais, de acordo com as suas respetivas datas esperadas.

3.2 Notação

O problema é formulado da seguinte forma. Todo o horizonte de tempo do problema (T) é dividido em r períodos de tempo de comprimento igual a $P \in \mathbb{N}$. Obtemos então o conjunto $T = \{1,\ldots,r\}$. Relativamente às caraterísticas específicas de cada problema, temos M máquinas paralelas e idênticas, onde será possível processar o conjunto N de trabalhos. Cada trabalho $j \in N$ tem a si associado 5 atributos. São eles:

1) tempo de processamento – $p_j \in \mathbb{N} \setminus \{0\}$;
2) data de lançamento (*release date*) – $r_j \in T$;
3) data esperada (*due date*) – $d_j \in T$;
4) e 5) fatores de penalização associados a um processo executado antes ($e_j \in \mathbb{N}$ - *early*) ou depois ($l_j \in \mathbb{N}$ - *late*) da data esperada.

O primeiro atributo, o tempo de processamento p_j, corresponde às unidades de tempo necessárias para que o trabalho $j \in N$ seja concluído. A seguir, a data de lançamento r_j trata do período de tempo mínimo, pertencente a T, em que o trabalho $j \in N$ está disponível para ser processado, isto é, o trabalho j só poderá ser processado num período de tempo $t \in T$ se $rj \leq t$. A data esperada d_j é a data estipulada para a realização do trabalho sem sanções associadas,

geralmente é a data acordada com o cliente, obedecendo obrigatoriamente à seguinte condição $dj \geq rj$. Já os fatores de penalização aplicam-se sempre que um trabalho $j \in N$ seja processado num período de tempo antes ou depois da sua data esperada, função *earliness* ou *tardiness* respetivamente (figura 11). Caso um trabalho $j \in N$ seja realizado no período de tempo $t \in T$ implica uma pena $w_t{}^j$ segundo a seguinte fórmula:

$$w_t{}^j := e_j \times \underbrace{max\{0, d_j - t\}}_{earliness} + \ell_j \times \underbrace{max\{0, t - d_j\}}_{tardiness}$$

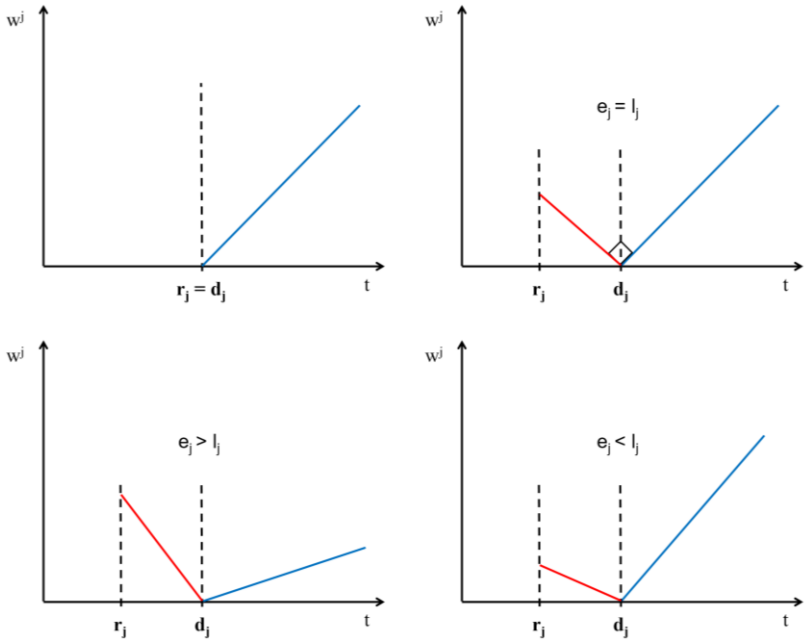


Figura 11 – Variação da função penalidade

Mais considerações são tidas em conta na formulação do problema. Cada trabalho só é realizado numa máquina e esta só pode executar um trabalho de cada vez. Não é permitida a interrupção de trabalhos e, associado a esta condição, também não é permitido um trabalho realizar-se em dois períodos distintos (mesmo que consecutivos). E por último, a soma dos tempos de processamento a serem executados num determinado período de tempo $t \in T$ não pode ser superior a P. Daqui concluímos que $P \geq max\{ p_j : j \in N \}$.

O principal objetivo é minimizar a soma das penalidades associadas ao adiantamento ou atraso na atribuição dos trabalhos aos períodos de tempo do problema cumprindo as restrições anteriormente referidas.

3.3 Exemplo

Do ponto de vista da formulação de um problema e considerando, a título de exemplo, um problema com um horizonte temporal de um dia de trabalho, onde se pretendem processar 10 tarefas em 2 máquinas paralelas idênticas, nos tradicionais 3 turnos de 8 horas existentes num dia de trabalho, temos assim, N=10, M=2, P=8 e T=3. Isto é, as 24 horas do dia de trabalho dividido em 3 turnos/períodos de trabalho. Temos que o conjunto T={1, 2, 3}.

tarefa	turno	turno	horas	penalidades	
j	r_j	d_j	p_j	e_j	l_j
1	1	1	2	0	24
2	2	2	4	0	10
3	1	2	1	22	20
4	1	3	2	30	20
5	3	3	5	0	10
6	2	2	1	0	25
7	1	2	7	15	15
8	1	1	2	0	5
9	2	3	2	45	20
10	1	1	3	0	12

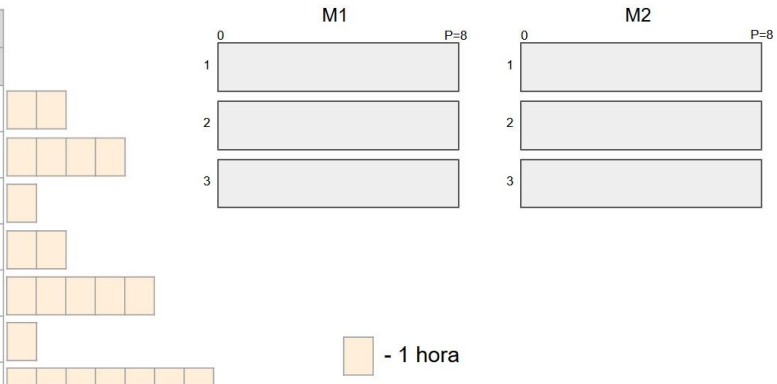


Figura 12 – Exemplo gráfico de um problema

Assumindo que o turno 1, 2 e 3 indicam o turno da manhã, tarde e noite, respetivamente. Segundo a figura 12, temos a tarefa j=1 com um tempo de processamento de 2 horas e está disponível para ser processada no turno da manhã e deve ficar o trabalho completo também nesse mesmo turno ($r_j = d_j = 1$). Ou seja, esta tarefa ou é realizada no turno esperado ou é atrasada para os próximos. Caso seja realizada no turno da tarde ou da noite, terá uma penalização associada. Na primeira situação, onde tem um turno de atraso, tem uma penalização $w^1 = 24\times1 = 24$ e na segunda, como é realizada ainda mais tarde (2 turnos), a penalização é $w^1 = 24\times2 = 48$.

A tarefa j=3 tem um tempo de processamento de 1 hora e encontra-se disponível para processamento no turno da manhã. No entanto, o turno esperado para que esta tarefa esteja pronta é apenas o turno da tarde ($r_j \neq d_j$). Assim, há a possibilidade de a tarefa ser executada exatamente no turno esperado, no turno anterior (antes) ou no turno seguinte (depois). Sendo que nas últimas duas hipóteses tem penalizações associadas de acordo com a sua situação. Deste modo, temos que a tarefa j=3 vai ter uma penalização $w^3 = 22\times1 = 22$ no caso de ser

realizada no turno da manhã, ou uma penalização $w^3 = 0$ ao ser realizada no turno esperado que era no da tarde, ou uma penalização $w^3 = 20 \times 1 = 20$ para o caso se ser executada apenas no turno da noite.

Neste exemplo foi considerado o horizonte temporal de apenas 1 dia (24 horas) dividido em três. De acordo com cada situação, em específico, pode ser facilmente alterado. Como é o caso de um horizonte com base em semanas, muito usado em problemas de planeamento e escalonamento.

4. ALGORITMOS DE OTIMIZAÇÃO

Os algoritmos de otimização experimentados têm por base a pesquisa local. Por isso, no início deste capítulo vamos clarificar os principais aspetos desta abordagem, entre os quais: a construção das soluções iniciais e os diferentes procedimentos para gerar soluções vizinhas. Por último, descrevemos a implementação do VNS e as suas estruturas de vizinhança associadas.

Todas as soluções geradas são soluções válidas ou viáveis. Na construção, quer da primeira solução, quer de soluções vizinhas, são garantidas as restrições que o problema impõe não sendo sequer geradas soluções inválidas (exclusão de soluções inviáveis).

4.1 Pesquisa Local

A pesquisa local é um processo iterativo em que, a cada iteração, são geradas soluções vizinhas da atual. Se o valor da função objetivo do problema for melhorado, que no nosso caso é encontrar um valor de penalidade menor, este processo continua com a nova solução melhor até não existirem soluções vizinhas melhores. A seguir mostramos o algoritmo de melhoria iterativa baseado na pesquisa local e os seus três princípios básicos.

Algoritmo 1 – Pesquisa local iterativa

		Princípios
	Função pesquisaLocal()	
1	*s*:= alguma solução inicial;	Solução Inicial
2	**Repete**	
3	gerar um vizinho $s' \in N(s)$;	Gerar soluções vizinhas
4	**se** $f(s') < f(s)$ **então** $s := s'$;	Medida de avaliação entre soluções (F.O.)
	até $f(s') \geq f(s)$ para todos $s' \in N(s)$;	

4.1.1 Construção de Soluções Iniciais

Como referido na secção 2.6 desta dissertação, as soluções iniciais têm um papel fundamental em todo o processo de pesquisa local subsequente e, portanto, foram tidas em conta diferentes maneiras de abordar o problema. Realizaram-se experiências para encontrar soluções iniciais através de quatro combinações obtidas entre o tipo de construção e a ordem das tarefas. Temos dois tipos de construção de soluções possíveis: construção a partir da data de lançamento e a partir da data esperada de cada tarefa. Quanto à ordem das tarefas, estas

podem ser ordenadas segundo regras de prioridade ou sem regras de prioridade, seguindo a ordem natural em que surgem no problema.

Tabela 1 – Construção das soluções inicias do problema

Construção da Solução Inicial	Tipo de construção	Ordem das tarefas
CSI1	A partir da data de lançamento	Sem regras de prioridade
CSI2	A partir da data de lançamento	Com regras de prioridade[1]
CSI3	A partir da data esperada	Sem regras de prioridade
CSI4	A partir da data esperada	Com regras de prioridade[1]

O tipo de construção a partir da data de lançamento aloca, se possível, as tarefas no período correspondente à sua data de lançamento. Se as máquinas do problema não têm capacidade no período correspondente à data de lançamento da tarefa, isto é, não tenham tempo suficiente para processar a tarefa nesse período, esta é alocada no período seguinte salvaguardando também a existência de tempo para processar a tarefa nesse próximo período. As tarefas assim que disponíveis (a partir de r_j) são alocadas no primeiro período t disponível em que $t \geq r_j$. Este tipo de construção não se preocupa com as possíveis penalizações a que as tarefas ficam sujeitas.

As soluções iniciais conseguidas através do tipo de construção a partir da data esperada (CSI3 e CSI4) procuram alocar cada uma das tarefas aos períodos e às máquinas disponíveis que resultem na menor penalização associada a cada tarefa. No momento em que é feita a sua alocação, esta não interfere com alocações anteriores. Assim as tarefas são colocadas, se possível, nas suas datas esperadas procurando cumprir os seus prazos. Se não existirem períodos correspondentes à data esperada da tarefa com capacidade para a processar, esta terá de ser adiantada ou atrasada. Mas numa primeira fase, são colocadas todas as tarefas que podem ser alocadas nos seus períodos esperados e só depois se adiantam ou atrasam as tarefas de acordo com seguintes princípios. Se $r_j = d_j$, a tarefa j terá de ser adiada e, portanto, colocada num dos períodos seguintes (com capacidade para a processar) que resulte na menor penalização para a tarefa j; se $r_j \neq d_j$, a tarefa j é colocada num período anterior ou posterior à sua data esperada, em ambos os casos a colocação da tarefa j procurará também o período que resultar na menor penalização para a tarefa. Este tipo de construção da solução inicial pode ser visto como sendo mais ganancioso (*greedy constructive*) do que o tipo

[1] Regras apresentadas na Tabela 2, p. 43

anterior, visto que este se preocupa sempre com a penalidade tida pelas tarefas em cada alocação.

Nas soluções inicias sem regras de prioridade associadas (CSI1 e CSI3) a ordem da alocação das tarefas é determinada pela ordem do *input* do problema.

As regras de prioridade consideram critérios para reordenar a lista das tarefas e determinam a ordem pela qual cada uma das tarefas vai ser alocada aos períodos e às máquinas. Ou seja, a tarefa que for considerada mais prioritária (a primeira tarefa da lista) será a primeira a ser colocada e assim sucessivamente até alocar as *N* tarefas.

Foram consideradas 14 variantes de ordenação que denominamos por *heu1*, *heu2*, *heu3*, *heu4*, *heu5*, *heu6*, *heu7*, *heu8*, *heu9*, *heu10*, *heu11*, *heu12*, *heu13* e *heu14* (tabela 2). As sequências são definidas em cada variante pelo critério 1 na ordem 1 e, em caso de empate, pelo critério 2 na ordem 2.

Tabela 2 – Variantes de ordenação dos trabalhos

Variante	Critério 1	Ordem 1	Critério 2	Ordem 2
heu1	$(e_j + l_j) / p_j$	decrescente	p_j	decrescente
heu2	$(e_j + l_j) / p_j$	decrescente	p_j	crescente
heu3	l_j / p_j	decrescente	p_j	decrescente
heu4	l_j / p_j	decrescente	p_j	crescente
heu5	p_j	decrescente		
heu6	p_j	crescente		
heu7	$e_j + l_j$	decrescente	p_j	decrescente
heu8	$e_j + l_j$	decrescente	p_j	crescente
heu9	l_j	decrescente	p_j	decrescente
heu10	l_j	decrescente	p_j	crescente
heu11	$p_j / (e_j + l_j)$	crescente	p_j	decrescente
heu12	$p_j / (e_j + l_j)$	crescente	p_j	crescente
heu13	p_j / l_j	crescente	p_j	decrescente
heu14	p_j / l_j	crescente	p_j	crescente

Nas variantes *heu1*, *heu2*, *heu3* e *heu4*, no critério 1, é considerado o quociente da divisão inteira entre as penalidades e o tempo de processamento, trata-se da penalidade associada por unidade de tempo utilizada pela tarefa. (Ver anexo I – a))

Todas as variantes ponderam o critério tempo de processamento. Esta análise torna-se importante porque ambos os casos (ordem crescente e decrescente) têm as suas vantagens e desvantagens dependendo das caraterísticas específicas de cada *input*. Considerando a ordem crescente, primeiro as tarefas de menor duração, poderá levar a uma menor quantidade de tarefas sujeitas a penalizações (ideal para problemas em que o objetivo seja reduzir o número de tarefas atrasadas), mas ao provocar o atraso das tarefas maiores pode levar a que estas sejam "empurradas" para períodos muito distantes das suas datas esperadas (subindo o valor da função objetivo). Isto acontece devido à sua duração poder levar à existência de grandes folgas em períodos distantes (figura 13 – a)). Por outro lado, se a ordem das tarefas for por tempo de processamento decrescente, ou seja, as tarefas de maior duração serem alocadas primeiro, poderá haver a colocação de menos tarefas nas suas datas esperadas, mas as restantes poderão ter penalizações menores já que estas podem ficar mais perto da sua data pretendida (figura 13 – b)).

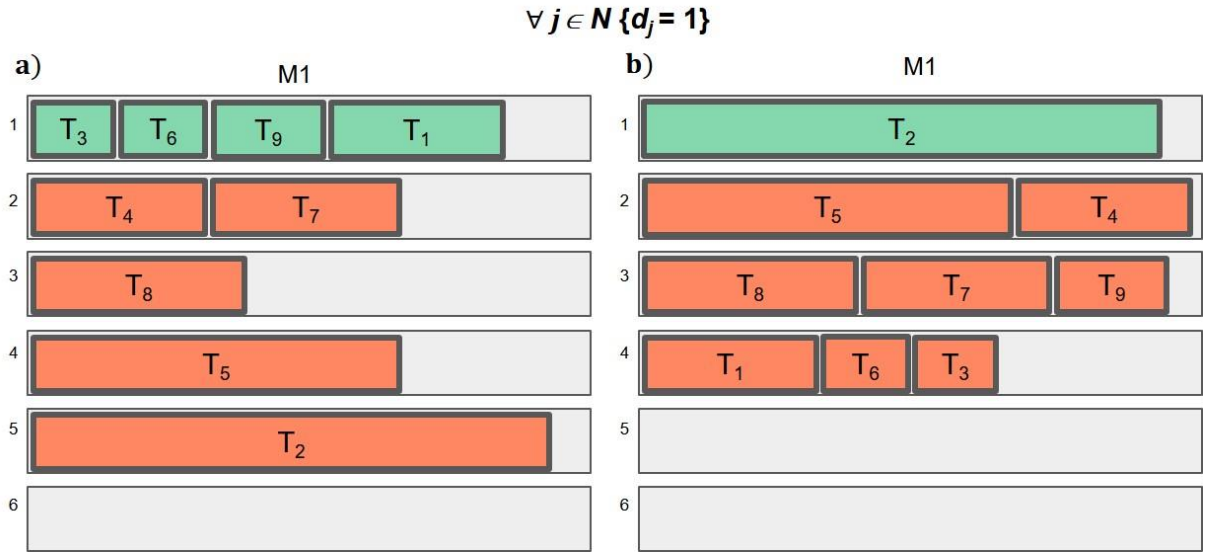


Figura 13 – Alocação de tarefas a períodos

As variantes *heu7* a *heu10* consideram apenas as penalidades sem que haja uma divisão pelo tempo de processamento. Em casos em que (muitas) tarefas apresentem um tempo de processamento maior que as penalidades, ao efetuar essa divisão (inteira) o valor iria ser sempre igual a 0. Ou seja, as variantes heu1, heu3 e heu5 apresentariam a lista das tarefas exatamente na mesma ordem (apenas com base na ordem decrescente do tempo de

processamento) e, heu2, heu4 e heu6, também com a mesma lista entre elas, mas com o tempo de processamento em ordem crescente. Deste modo, estas variantes tentam dar prioridade às tarefas mais penalizadoras mesmo com tempos de processamento elevados, o que acabava por não acontecer nas outras variantes. Assim sendo, e para contemplar também a relação entre o tempo de processamento e os fatores de penalização para problemas em que as tarefas tenham o tempo de processamento bastante elevado, consideramos ainda as variantes *heu11*, *heu12*, *heu13* e *heu14*, em ordem crescente.

Com todas estas variantes pretendemos alcançar um grande espaço de soluções que nos garantem mais hipóteses de sucesso.

No sentido de conseguir a melhor solução inicial, isto é, aquela que apresenta o melhor valor para a nossa função objetivo na construção da primeira solução, fizemos também uma outra abordagem (CSI5) em que verificávamos qual o melhor começo para a nossa solução tendo em conta as várias possibilidades já aqui enumeradas e explicadas (CSI1, CSI2, CSI3 e CSI4).

Esta abordagem assemelha-se à meta-heurística designada por *multistart*. O múltiplo começo procura através das várias soluções iniciais conseguir fugir a ótimos locais e alcançar o melhor valor final possível. No entanto, na nossa abordagem o *multistart* serve apenas para conseguirmos encontrar aquela que se apresenta como a melhor alternativa para a solução inicial e a partir dessa aplicar métodos de pesquisa local e VNS. De referir que encontrar a solução inicial em todas as abordagens ocorria em poucos milissegundos. Analisaremos, posteriormente, se a melhor solução inicial é efetivamente aquela que nos garante o melhor valor da solução final.

4.1.2 Soluções Vizinhas

Após obter a solução inicial é necessário provocar trocas na solução com o objetivo de encontrar soluções vizinhas com resultados melhores. Nesta subsecção descrevemos o processo de obtenção de soluções vizinhas $N(s)$ a partir de uma solução s.

Dada uma solução s, numa primeira fase, as tarefas são ordenadas numa lista por ordem decrescente de penalidade e é escolhida a primeira tarefa (denominada tarefa j) para a mudar de posição, por outras palavras, é selecionada a tarefa mais penalizadora. É verificado se esta tarefa pode ser alocada a um período onde incorra numa penalidade menor ou até em nenhuma penalidade. Começa por tentar trocá-la com alguma outra tarefa (tarefa k) que esteja num período d_j e a troca compense. Para a troca compensar a soma das penalidades da tarefa j

e da tarefa *k* nas suas novas localizações tem de ser inferior à soma das suas penalidades iniciais. A tarefa *j* só poderá trocar com a tarefa *k* se o tempo restante do período atual da tarefa *k* (t_{rest_k}) menos o tempo de processamento de *k* (p_k) mais o de *j* (p_j) for igual ou superior a 0 ($t_{rest\_k} - p_k + p_j \geq 0$), garantido assim não ultrapassar o limite de tempo do problema.

A troca efetuada pode não ser, necessariamente, uma troca direta entre as duas tarefas. Recorrendo a um exemplo, a tarefa *j* que estava num período X é colocada no período Y anteriormente ocupada pela tarefa *k,* e a tarefa *k* vai para um outro período possível onde incorra na sua menor penalidade (sem efetuar troca com mais nenhuma tarefa, isto é, ocupando possivelmente um espaço disponível num período mais vantajoso). Só no caso do melhor sítio possível para a tarefa *k* ser o período X (primeira localização da tarefa *j*) é que estamos perante uma situação de troca direta. (Ver anexo I – i) e l))

Caso nos períodos d_j não seja encontrada nenhuma tarefa em que a troca pela tarefa *j* seja praticável ou compense, é testado de igual forma no período seguinte ou no período anterior, se possível, de acordo com as penalizações e_j e l_j. Isto é, se for menos penalizador adiantar do que adiar a tarefa *j* ($e_j < l_j$) é testado o período anterior ($d_j - 1$), senão é preferível adiar para o período seguinte ($d_j + 1$). Salvaguardando, como é óbvio, o facto de estarmos já no limite inferior ou superior dos períodos disponíveis do problema.

Na hipótese de no período escolhido as trocas testadas serem, novamente, desfavoráveis é analisado se compensa adiantar ou adiar tendo em conta a nova situação em que o valor de adiantar ou atrasar já não é igual ao inicial. Se no início a penalidade de adiantar e atrasar eram e_j e l_j, respetivamente, agora os valores das penalidades são diferentes.

Considerando que na primeira situação compensou adiantar e no período $d_j - 1$ não existiam trocas favoráveis é preciso definir um novo período para a pesquisa. Assim sendo, os novos valores das penalidades associadas à tarefa *j* para adiantar (dois períodos, visto que adiantar 1 já foi testado) ou atrasar (um período) são: $e_j \times 2$ e l_j, pelo que se $e_j \times 2 < l_j$ continuará a compensar adiantar. No sentido contrário, se na primeira situação temos $l_j < e_j$, ou seja, uma situação em que compensa adiar, é testado o período $d_j + 1$. Caso se verifique que não há trocas favoráveis nesse período é analisada a seguinte inequação $e_j < l_j \times 2$. Se for uma inequação válida, compensa analisar o período antes ($d_j - 1$), senão compensa adiar e verificar o período $d_j + 2$.

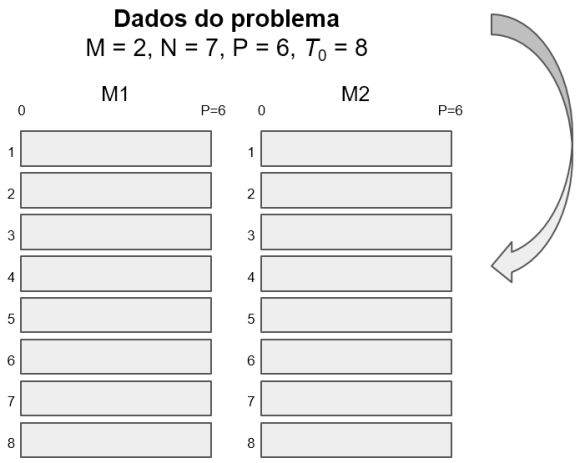
Se no fim de analisar todas as tarefas possíveis com quem a tarefa *j* pode trocar não existirem trocas que sejam vantajosas para o resultado final é verificado se é possível colocar essa tarefa *j* em algum outro período menos penalizador, mas sem estabelecer uma troca.

Apenas há a mudança de período. As construções das soluções iniciais seguiam de forma semelhante este último princípio, sendo que se procura a melhor alocação para uma certa tarefa na solução.

Neste processo iterativo de pesquisa local, a seleção do vizinho é feita assim que haja uma melhoria no valor da solução (*first improvement*). Inicialmente, ainda foi experimentada a seleção com base na melhor melhoria (*best improvement*), mas a demora que esta alternativa exige para vizinhanças de tamanho considerável acabou por nos fazer debruçar apenas na primeira melhoria. Sempre que exista uma troca ou uma mudança de período por alguma das tarefas (situação em que há uma melhoria), a lista dos trabalhos ordenados por penalidade é reordenada e selecionado, novamente, o primeiro da lista. Caso não hajam mudanças, porque não compensa, é selecionado o segundo trabalho da lista efetuando-se o mesmo processo sucessivamente até não haverem mais melhorias possíveis (avaliação completa da vizinhança).

4.1.3 Exemplo

Para melhor entender a construção de uma solução inicial, bem como as trocas promovidas para que se encontrem melhores soluções, encontra-se no anexo I uma descrição passo a passo para o exemplo simples descrito na figura 14, onde são consideradas 2 máquinas para executar 7 tarefas.



Tarefas ($j \in N$):

j	rj	dj	pj	ej	lj
1	5	5	6	0	100
2	5	5	5	0	100
3	5	5	5	0	25
4	3	5	4	10	30
5	6	6	4	0	50
6	2	5	4	20	50
7	2	5	4	5	8

Figura 14 – Dados de um problema (exemplo)

4.2 Pesquisa por Vizinhanças Variáveis (VNS)

O VNS é uma meta-heurística que consiste em mudanças sistemáticas de vizinhanças combinada com a pesquisa local e evidencia-se por não parar no primeiro ótimo local. Vários são os métodos VNS, sendo que o abordado neste projeto foi o VNS básico (BVNS – *Basic Variable Neighborhood Search*) (Mladenović e Hansen, 1997) que concilia mudanças determinísticas de vizinhanças e mudanças estocásticas. Estas últimas conferem ao método alguma aleatoriedade importante a fim de evitar possíveis ciclos e permitir aumentar o espaço de soluções abrangidas.

4.2.1 Implementação do Algoritmo

O VNS básico tem três parâmetros de entrada: uma solução inicial (*s*), o k_{max} que determina o número máximo de estruturas de vizinhança e o t_{max} que é o tempo máximo de execução do algoritmo.

O método segue os seguintes passos:

Algoritmo 2 – Passos do BVNS

	Função *basicVNS*(*s*, k_{max}, t_{max})
1	**repete**
2	k:=1;
3	**repete**
4	*s'*:= *shake*(*s*, *k*);
5	*s''*:= *firstImprovement*(*s'*);
6	**se** *f(s'') < f(s)* **então**
7	*s*:=*s''*;
8	*k*:=1;
9	**senão** *k*:=*k*+1;
	até $k>k_{max}$;
10	*t*:=*CPUTime()*;
	até $t > t_{max}$;
11	*s*:=*pesquisaLocal*();

No passo 4 é aplicada a função *shake(s,k)* que, como o próprio nome indica, provoca uma agitação à solução *s* de acordo com a estrutura de vizinhança definida para determinado valor de *k*.

De seguida, passo 5, é selecionado um vizinho de *s'*, através do processo iterativo de pesquisa local com uma seleção *first improvement*, onde há apenas uma melhoria da solução dada no passo anterior, resultando numa solução *s''*.

Se a nova solução encontrada *s''* é melhor que a solução de início da iteração *s*, então a solução *s* passa a ter o valor de *s''* e a estrutura de vizinhança a considerar na próxima iteração volta a ser a primeira *k:=1*. Senão, a estrutura de vizinhança passa a ser a próxima *k:=k+1*.

Caso nenhuma solução vizinha resulte num melhor resultado (atinge o k_{max}) o processo é repetido se o tempo de execução do algoritmo t_{max} ainda não foi excedido. A função *shake* é composta por uma componente aleatória que possibilita que mesmo executando o idêntico processo os resultados possam ser diferentes.

No passo 11 são realizadas sucessivas iterações de pesquisa local até se atingir o mínimo local já que podem ter acontecido trocas que tenham disponibilizado espaços para algumas tarefas poderem conseguir um sítio melhor. Ao contrário do que acontece no passo 5, que para encontrar a primeira melhoria, este faz um "varrimento" completo até não existirem mais melhorias (algoritmo 1).

As sucessivas mudanças de vizinhança que caraterizam esta abordagem são parte fundamental e, portanto, estão descritas ao detalhe na subsecção seguinte, bem como a função *shake* que faz uso dessas mesmas estruturas variáveis.

4.2.2 Estruturas de Vizinhança

Nesta investigação consideramos duas estruturas de vizinhança distintas para provocarem a variedade das soluções vizinhas. São elas:

k=1 – Troca entre duas tarefas não necessariamente de períodos adjacentes;
k=2 – Troca entre duas tarefas de períodos adjacentes.

Em ambas as estruturas, a aleatoriedade está presente na escolha da primeira e da segunda tarefa. No entanto, para se cumprirem as restrições do problema, a escolha da segunda tarefa depende da primeira tarefa escolhida. A segunda tarefa é escolhida aleatoriamente entre as tarefas possíveis para a troca com a primeira. Isto é, a data de lançamento da segunda tarefa tem de ser igual ou inferior ao período atual da primeira e, de forma análoga, a data de lançamento da primeira tarefa tem de ser igual ou inferior ao período atual da segunda tarefa. A troca entre tarefas que estejam no mesmo período não são

consideradas porque é sabido à partida que não alterará o valor da penalidade. A juntar a estas condicionantes tem de ser garantido que o tempo de processamento das tarefas não ultrapassa o limite do período em conjunto com os tempos das tarefas lá alocadas.

Assumindo um exemplo de 30 tarefas com o mesmo tempo de processamento, representadas por T_j para $j \in N$ na figura 15, temos em a) representadas a verde as tarefas elegíveis para troca com a tarefa T_7 de acordo com a estrutura de vizinhança k=1 e as restrições impostas. As tarefas de 21 a 30 como só estão disponíveis no período 3 e a tarefa 7 está no período 2, não podem fazer parte das tarefas elegíveis para troca com T_7. Para a estrutura de vizinhança k=2, em b), são consideradas tarefas elegíveis aquelas que se encontrem em períodos adjacentes e cumpram também com as restrições do problema. Este é, obviamente, um exemplo simplista já que ao admitir o mesmo tempo de processamento para todas as tarefas evita desde logo as restrições associadas à capacidade do período para efetuar a troca.

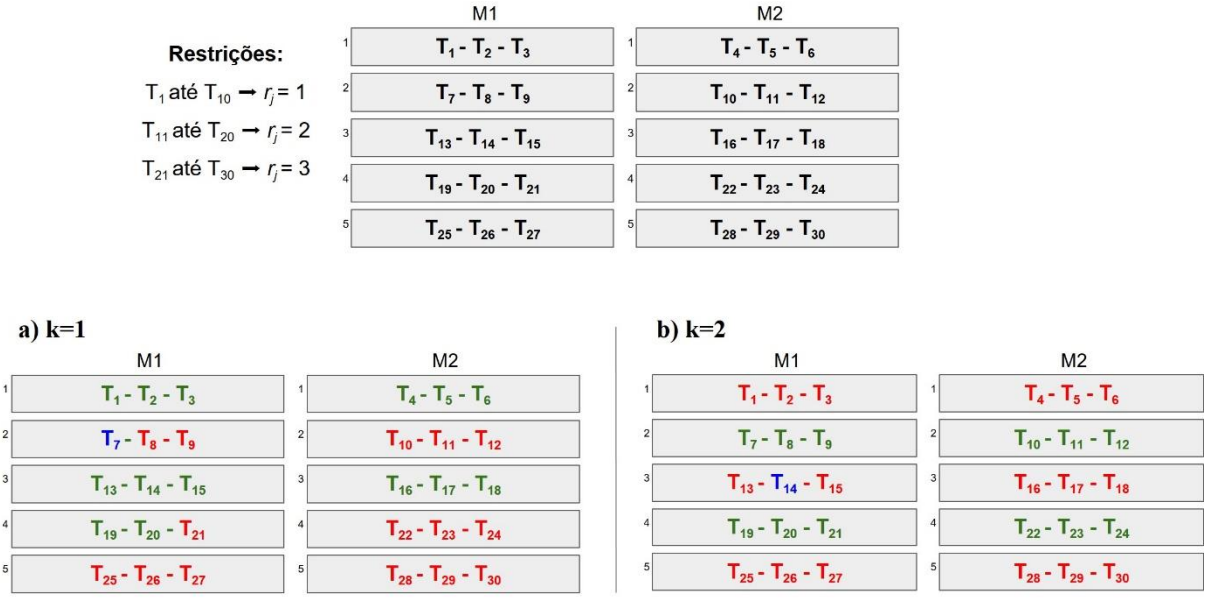


Figura 15 – Tarefas elegíveis de acordo com a estrutura de vizinhança usada

A nossa função *shake* segue os passos seguintes:

Algoritmo 3 – Passos da função shake utilizada na BVNS

Função *shake*(*k*)

1	encontrouTroca:=0;		
2	**Repete**		
3		tarefa1:= *random*(entre todas as tarefas);	// seleciona uma tarefa aleatória
4		**se** (k=1) **então**	//se é a primeira estrutura de vizinhança
5		trabalhosElegiveis := *daTarefasElegiveis*(tarefa1);	// lista das possíveis tarefas para troca com a tarefa1 de todos os períodos do problema
6		**Senão**	//se é a segunda estrutura de vizinhança
7		trabalhosElegiveis := *daTarefasAdjacentesElegiveis*(tarefa1);	// lista das possíveis tarefas para troca com a tarefa1 exclusivamente dos períodos adjacentes
8		**se** (trabalhosElegiveis > 0) **então**	// se tem trabalhos elegíveis
9		tarefa2 := *random*(entre os trabalhos elegíveis);	// seleciona uma segunda tarefa aleatória
10		*swapJobs*(tarefa1, tarefa2);	//troca de posições entre a tarefa1 e a tarefa2
11		encontrouTroca:=1;	
	até encontrouTroca=1;		

5. EXPERIÊNCIAS COMPUTACIONAIS

Para as diferentes abordagens (CSI1, CSI2, CSI3, CSI4 e CSI5, em conjunto com as suas possíveis variantes) foram realizadas experiências computacionais para avaliar e comparar os seus desempenhos. Todas as estruturas e algoritmos foram implementados na linguagem de programação *Java* e executados num computador com um processador *Intel Core* i7-2620M de 2.70 GHz e 8 GB de RAM no sistema operativo *Windows* 10.

Como os algoritmos testados têm uma componente com alguma aleatoriedade (existente na função *shake*) os resultados obtidos numa só execução podem não ser um bom exemplo do desempenho e eficácia do algoritmo. Por isso, para colmatar essa situação, foram executadas 5 vezes cada uma das experiências, conseguindo assim ter uma amostra maior e confirmar, ou não, a consistência dos resultados. O algoritmo com base no método VNS básico tem, geralmente, um limite de tempo ou de iterações para efetuar a procura. No nosso caso foi considerado o limite de tempo de 5 segundos (t_{max}=5).

As instâncias utilizadas para as experiências são instâncias de referência utilizadas em Kis e Kovács (2012) para esse efeito. Estabelecemos comparações de desempenho com estudos feitos precisamente para o mesmo problema com abordagens exatas de Rietz *et al.* (2016) e heurísticas de Rietz *et al.* (2015). As diferentes caraterísticas das instâncias consideradas permitiu-nos apurar quais as situações em que os nossos algoritmos têm um comportamento mais eficaz.

5.1 Descrição das Instâncias

As instâncias são compostas por problemas que contêm tarefas de pequena dimensão, problemas com tarefas de pequena dimensão e de grande dimensão, e outros unicamente de tarefas de grande dimensão. De acordo com estas caraterísticas específicas relativas ao tempo de processamento das tarefas, as instâncias foram distinguidas em três grandes conjuntos A, B e C, respetivamente.

Em todas as instâncias o tamanho definido para cada período de tempo foi 100, pelo que P=100. Kis e Kovács (2012) consideram que as tarefas de pequena dimensão (conjunto A) são as que têm um tempo de processamento inferior a um terço de P ($p_j \leq 33$). Enquanto as de grande dimensão (conjunto C) são as que têm um tempo de processamento superior a 33 até ao valor de P ($34 \leq p_j \leq 100$). O conjunto B contempla metade das tarefas de pequena dimensão e a outra metade de grande dimensão.

Cada conjunto (A, B, e C) tem 5 instâncias diferentes para a combinação do número de tarefas (N), o número de máquinas (M) e o número de períodos (τ_0) entre os quais as tarefas têm as suas datas de lançamento e as suas datas esperadas (estas variam de 1 até τ_0). O número de máquinas (M) e o número de períodos (τ_0) considerados para cada conjunto foram iguais sendo $M \in \{2, 6, 10\}$ e $\tau_0 \in \{2, 6, 10\}$. Os valores do número de tarefas é que variam para cada conjunto devido à diferente complexidade. Assim sendo, para o conjunto A o número de tarefas foi $N \in \{100, 150, 200, 250, 300\}$. No conjunto B temos $N \in \{50, 100, 150, 200, 250, 300\}$. E, por último, no conjunto C temos $N \in \{40, 60, 80, 100\}$.

Para garantir que todos os problemas eram exequíveis para as suas diferentes caraterísticas, Kis e Kovács (2012) definem o número de períodos (τ) existentes em cada máquina, como

$$\tau = \tau_0 + 2 \times \sum_{j \in N} p_j / (P \times M)$$

Tal como já haviam feito Rietz *et al.* (2015), consideramos também as instâncias maiores (mais tarefas) que Kis e Kovács (2012) não mencionam. Tendo em conta estas caraterísticas (5 instâncias diferentes e N, M e τ_0 a variarem como já vimos) temos:

- Para o conjunto A – 225 instâncias (5×5×3×3=225)
- Para o conjunto B – 270 instâncias (5×6×3×3=270)
- Para o conjunto C – 180 instâncias (5×4×3×3=180)

O que perfaz um total de 675 instâncias utilizadas para as experiências computacionais.

5.2 Resultados Computacionais e Análise Crítica

Os algoritmos executaram 5 vezes cada uma das instâncias para as diferentes abordagens de problema. Para as abordagens CSI1 e CSI3, que não consideram as regras de prioridade, temos 5×675+5×675=6750 resultados. Para as abordagens CSI2 e CSI4, que consideram as 14 regras de prioridade, temos 5×675×14+5×675×14=94500 resultados. A abordagem CSI5 ("*multistart*"), tal como as abordagens CSI1 e CSI3, não tem variantes já que a própria abordagem escolhe a melhor, pelo que temos 5×675=3375 resultados. No total temos 104625 resultados para as instâncias consideradas o que dá 155 resultados analisados por instância.

Cada resultado tem a si associado uma série de informações. Para além do valor final da solução, também possui a lista das tarefas pela ordem em que é tentada a alocação aos períodos, o valor da solução inicial, o tempo total de execução do algoritmo para aquela instância, o tempo em que foi encontrado o melhor valor da solução e, por último, a solução propriamente dita com todas as informações relativas às máquinas, períodos e tarefas (ver anexo II). Estas informações atestam e certificam os valores obtidos.

As nossas soluções encontradas foram comparadas com resultados já existentes na literatura com o objetivo de perceber a sua qualidade, entre os quais os resultados obtidos através de um método exato em Rietz *et al.* (2016) (executando cada uma das instâncias durante 1200 segundos) e os resultados obtidos por intermédio de heurísticas de Rietz *et al.* (2015). Estes determinam o limite inferior e superior, respetivamente.

Para cada um dos três conjuntos, A, B e C, foram recolhidos os resultados médios das cinco instâncias existentes para problemas com as mesmas caraterísticas, isto é, com o mesmo número de tarefas, *N*, mesmo número de máquinas, *M*, e o mesmo número de períodos em que a data de lançamento e a data esperada podem estar, τ_0 (*tau*). De seguida, apresentamos o significado das colunas das tabelas utilizadas para sintetizar os resultados obtidos:

- *best lb*: média dos melhores valores do limite inferior dado pelo método exato em Rietz *et al.* (2016);
- *best ub*: média dos melhores valores do limite superior obtidos em Rietz *et al.* (2015);
- *gap*: intervalo médio de otimização, em percentagem, entre o *best lb* e o *best ub* ((*best lb* - *best ub*)/(*best lb*)). No caso de estarmos perante uma diferença para o limite inferior e este limite for igual a 0, é colocado o símbolo "-" (expressão sem significado para números reais, divisão por 0);
- CSI1 a CSI5: média das melhores soluções obtidas através do algoritmo *basicVNS* em cada uma das abordagens de CSI1 a CSI5;
- *vbest*: média das melhores soluções obtidas em todas as abordagens consideradas;
- *gap'*: diferença entre o *gap* de *best lb* e *vbest* e o *gap* referido em cima, também em percentagem;
- CSI1 a CSI5 *optimum*: número de vezes em que a solução encontrada através do algoritmo *basicVNS* é igual ao limite inferior em cada uma das abordagens de CSI1 a CSI5.

Nas tabelas 3, 4, e 5 estão os resultados computacionais de forma agregada do conjunto A, B e C, respetivamente.

Tabela 3 – Resultados computacionais para as instâncias do conjunto A

|N|	|M|	\tau	best lb	best ub	gap	CSI1	CSI2	CSI3	CSI4	CSI5	vbest	gap'	CSI1 optimum	CSI2 optimum	CSI3 optimum	CSI4 optimum	CSI5 optimum
100	2	2	1015,0	1025,8	1,07	1136,0	1085,8	1123,6	1037,4	1047,4	1037,4	1,13	0,0	0,0	0,0	0,0	0,0
100	2	6	269,8	283,2	4,99	313,0	303,8	302,4	291,4	299,0	291,2	2,89	0,0	0,0	0,0	0,0	0,0
100	2	10	63,0	65,0	3,16	70,4	67,8	70,6	68,6	74,6	67,4	3,53	0,0	0,0	0,0	0,0	0,0
100	6	2	106,4	106,6	0,16	115,8	113,0	112,8	110,6	110,4	109,8	3,07	0,0	0,0	0,0	0,0	0,0
100	6	6	1,2	1,2	0,00	1,6	1,2	1,6	1,2	1,4	1,2	0,00	3,0	5,0	3,0	5,0	4,0
100	6	10	0,0	0,0	0,00	0,0	0,0	0,0	0,0	0,0	0,0	0,00	5,0	5,0	5,0	5,0	5,0
100	10	2	21,0	21,0	0,00	24,0	22,6	22,6	21,8	23,0	21,8	2,93	1,0	1,0	1,0	2,0	1,0
100	10	6	0,0	0,0	0,00	0,0	0,0	0,0	0,0	0,0	0,0	0,00	5,0	5,0	5,0	5,0	5,0
100	10	10	0,0	0,0	0,00	0,0	0,0	0,0	0,0	0,0	0,0	0,00	5,0	5,0	5,0	5,0	5,0
150	2	2	2711,8	2757,6	1,68	3009,0	2830,6	2999,6	2773,0	2800,0	2773,0	0,57	0,0	0,0	0,0	0,0	0,0
150	2	6	1145,0	1408,6	23,00	1328,8	1317,4	1287,2	1218,2	1222,2	1211,4	-17,22	0,0	0,0	0,0	0,0	0,0
150	2	10	440,4	686,8	55,55	516,8	506,6	519,6	485,0	503,2	485,0	-45,35	0,0	0,0	0,0	0,0	0,0
150	6	2	354,4	356,0	0,45	391,8	386,6	396,0	370,2	371,8	370,2	4,11	0,0	0,0	0,0	0,0	0,0
150	6	6	30,0	30,2	0,51	35,0	33,6	33,2	32,4	33,4	32,0	5,15	0,0	0,0	0,0	1,0	0,0
150	6	10	0,2	0,2	0,00	0,2	0,2	0,4	0,2	0,2	0,2	0,00	5,0	5,0	4,0	5,0	5,0
150	10	2	113,8	113,8	0,00	124,4	120,8	122,2	119,4	120,6	118,8	4,67	0,0	0,0	0,0	0,0	0,0
150	10	6	0,6	0,6	0,00	1,0	0,6	1,0	1,0	1,0	0,6	0,00	4,0	5,0	4,0	4,0	4,0
150	10	10	0,0	0,0	0,00	0,0	0,0	0,0	0,0	0,0	0,0	0,00	5,0	5,0	5,0	5,0	5,0
200	2	2	5203,6	5399,0	3,76	5784,6	5383,2	5820,4	5292,2	5339,4	5292,2	-2,06	0,0	0,0	0,0	0,0	0,0
200	2	6	2613,2	2967,8	13,65	3065,6	3028,0	2932,4	2735,4	2733,8	2732,0	-9,10	0,0	0,0	0,0	0,0	0,0
200	2	10	1110,4	1591,0	43,08	1355,0	1328,0	1306,6	1238,4	1242,6	1234,4	-31,98	0,0	0,0	0,0	0,0	0,0
200	6	2	918,4	947,8	3,05	1024,8	1039,0	1041,4	950,8	953,0	950,2	0,40	0,0	0,0	0,0	0,0	0,0
200	6	6	111,2	143,8	30,04	128,8	123,8	123,6	121,6	124,0	120,6	-21,27	0,0	0,0	0,0	0,0	0,0
200	6	10	6,0	6,0	0,00	7,4	6,8	7,6	6,6	7,2	6,2	10,00	0,0	2,0	2,0	3,0	1,0
200	10	2	289,6	293,4	1,40	331,2	316,4	327,4	306,6	307,4	306,0	4,28	0,0	0,0	0,0	0,0	0,0
200	10	6	0,0	0,0	0,00	0,0	0,0	0,0	0,0	0,0	0,0	0,00	5,0	5,0	5,0	5,0	5,0
200	10	10	0,0	0,0	0,00	0,0	0,0	0,0	0,0	0,0	0,0	0,00	5,0	5,0	5,0	5,0	5,0
250	2	2	7934,2	8144,0	2,64	9113,4	8185,2	9124,4	8061,0	8149,6	8061,0	-1,05	0,0	0,0	0,0	0,0	0,0
250	2	6	4884,6	5299,4	8,61	5579,0	5585,8	5495,8	5049,2	5095,6	5048,8	-5,23	0,0	0,0	0,0	0,0	0,0
250	2	10	497,0	3373,4	23,34	3243,0	3187,0	3126,0	2930,8	2951,4	2929,0	-15,41	0,0	0,0	0,0	0,0	0,0
250	6	2	1694,2	1763,2	4,06	1926,4	1875,4	1898,2	1738,0	1743,0	1737,8	-1,49	0,0	0,0	0,0	0,0	0,0
250	6	6	303,0	471,0	55,42	353,8	345,0	346,2	335,8	347,4	335,8	-44,56	0,0	0,0	0,0	0,0	0,0
250	6	10	38,8	40,8	5,61	48,2	44,0	44,2	42,6	44,8	42,6	4,80	0,0	0,0	0,0	0,0	0,0
250	10	2	592,2	625,6	5,35	662,6	660,0	664,4	623,2	626,8	623,2	-0,12	0,0	0,0	0,0	0,0	0,0
250	10	6	29,4	29,4	0,00	35,8	35,0	35,6	33,0	33,2	32,6	14,03	0,0	0,0	0,0	0,0	0,0
250	10	10	0,0	0,0	0,00	0,0	0,0	0,0	0,0	0,0	0,0	0,00	5,0	5,0	5,0	5,0	5,0
300	2	2	0	12360,8	-	13793,4	12370,2	13638,4	12192,8	12359,6	12192,8	-	0	0	0	0	0
300	2	6	0	8740	-	9203,2	9066	9018,8	8418,6	8492,6	8418,6	-	0	0	0	0	0
300	2	10	0	6137	-	6044,6	6087,8	5866,4	5587,8	5624,4	5582,6	-	0	0	0	0	0
300	6	2	2846,6	2960,6	4,009632	3259,8	3117,4	3219,4	2921	2933,6	2920,6	-1,40	0	0	0	0	0
300	6	6	579,6	876,6	51,59878	683,6	658	662,6	636,4	647,2	634,8	-41,93	0	0	0	0	0
300	6	10	128,8	243,4	88,71016	162,6	156,2	158,6	148	152	148	-73,67	0	0	0	0	0
300	10	2	1135,6	1178,8	3,575548	1273,4	1270,6	1286,8	1188,6	1189,2	1187,8	1,00	0	0	0	0	0
300	10	6	109,4	164,2	48,25619	130,8	127,4	124	121	123,2	120,8	-37,82	0	0	0	0	0
300	10	10	1,8	1,8	0	2	2	2,2	2	2,2	2	3,33	4	4	3	4	4

Tabela 4 – Resultados computacionais para as instâncias do conjunto B

|N|	|M|	\tau	best lb	best ub	gap	CSI1	CSI2	CSI3	CSI4	CSI5	vbest	gap'	CSI1 optimum	CSI2 optimum	CSI3 optimum	CSI4 optimum	CSI5 optimum
50	2	2	653,0	653,0	0,00	706,2	685,4	712,6	679,0	705,8	678,8	3,79	0,0	0,0	0,0	0,0	0,0
50	2	6	261,6	261,6	0,00	324,6	290,4	301,4	280,0	297,8	279,2	6,68	0,0	0,0	0,0	0,0	0,0
50	2	10	123,6	123,6	0,00	145,2	131,4	140,4	132,6	145,6	129,0	4,34	0,0	1,0	0,0	1,0	0,0
50	6	2	89,4	89,4	0,00	109,2	98,2	108,6	96,0	102,4	95,2	6,25	0,0	0,0	0,0	0,0	0,0
50	6	6	4,6	4,6	0,00	6,6	5,2	9,6	4,8	5,8	4,8	6,67	0,0	2,0	0,0	4,0	2,0
50	6	10	0,0	0,0	0,00	0,0	0,0	0,4	0,0	0,0	0,0	0,00	5,0	5,0	3,0	5,0	5,0
50	10	2	26,8	26,8	0,00	31,0	28,8	36,0	28,4	31,8	28,4	6,96	0,0	0,0	0,0	0,0	0,0
50	10	6	0,0	0,0	0,00	0,0	0,0	0,0	0,0	0,0	0,0	0,00	5,0	5,0	5,0	5,0	5,0
50	10	10	0,0	0,0	0,00	0,0	0,0	0,0	0,0	0,0	0,0	0,00	5,0	5,0	5,0	5,0	5,0
100	2	2	2874,0	2877,4	0,12	3071,8	3009,0	3049,4	2985,6	3008,2	2981,2	3,58	0,0	0,0	0,0	0,0	0,0
100	2	6	1829,2	1856,4	1,47	2088,0	2018,0	2009,0	1994,8	2026,2	1985,2	7,07	0,0	0,0	0,0	0,0	0,0
100	2	10	1167,6	1191,0	1,96	1468,8	1369,4	1356,8	1298,2	1330,4	1298,0	9,17	0,0	0,0	0,0	0,0	0,0
100	6	2	676,0	676,0	0,00	775,2	751,2	748,8	720,4	729,2	719,2	6,45	0,0	0,0	0,0	0,0	0,0
100	6	6	146,2	147,0	0,54	187,6	170,0	195,0	170,2	176,2	168,4	15,73	0,0	0,0	0,0	0,0	0,0
100	6	10	33,2	33,2	0,00	47,2	43,2	55,0	39,6	45,4	39,6	19,05	0,0	0,0	0,0	0,0	0,0
100	10	2	244,8	244,8	0,00	287,0	274,0	292,8	268,0	282,2	267,8	9,33	0,0	0,0	0,0	0,0	0,0
100	10	6	23,8	23,8	0,00	31,6	28,2	39,4	27,6	29,2	27,4	14,86	0,0	0,0	0,0	0,0	0,0
100	10	10	0,2	0,2	0,00	0,8	0,4	1,4	0,2	0,2	0,2	0,00	2,0	4,0	1,0	5,0	5,0
150	2	2	7279,0	7851,8	7,87	7725,8	7527,0	7747,0	7507,4	7505,2	7489,0	-4,99	0,0	0,0	0,0	0,0	0,0
150	2	6	5036,0	5673,4	12,69	5480,8	5451,4	5525,6	5365,6	5406,8	5351,2	-6,46	0,0	0,0	0,0	0,0	0,0
150	2	10	4181,8	5179,0	23,84	4878,2	4731,4	4729,4	4597,6	4624,8	4586,0	-14,29	0,0	0,0	0,0	0,0	0,0
150	6	2	1798,0	1799,2	0,06	1952,0	1924,8	1928,2	1893,6	1899,6	1893,6	5,23	0,0	0,0	0,0	0,0	0,0
150	6	6	630,4	638,6	1,34	822,6	752,2	748,2	719,8	737,8	719,8	12,86	0,0	0,0	0,0	0,0	0,0
150	6	10	197,4	199,2	0,76	253,0	241,4	274,6	237,0	242,4	232,6	17,52	0,0	0,0	0,0	0,0	0,0
150	10	2	827,4	827,8	0,05	947,0	924,2	915,8	888,0	895,2	887,0	7,15	0,0	0,0	0,0	0,0	0,0
150	10	6	147,2	148,0	0,51	187,0	175,4	201,6	171,6	177,6	171,6	16,32	0,0	0,0	0,0	0,0	0,0
150	10	10	14,8	14,8	0,00	23,2	22,6	34,2	18,8	20,2	18,8	30,32	0,0	0,0	0,0	0,0	0,0
200	2	2	13023,0	13801,2	6,00	13823,8	13448,4	13805,0	13385,8	13403,4	13381,8	-3,23	0,0	0,0	0,0	0,0	0,0
200	2	6	10941,8	12137,4	11,02	11880,4	11701,6	11805,0	11547,6	11558,6	11539,6	-5,58	0,0	0,0	0,0	0,0	0,0
200	2	10	8461,6	10094,6	19,18	9577,8	9323,4	9310,0	9085,2	9163,6	9064,0	-12,01	0,0	0,0	0,0	0,0	0,0
200	6	2	3600,2	3610,6	0,28	3896,8	3794,2	3894,8	3767,8	3764,6	3761,6	4,21	0,0	0,0	0,0	0,0	0,0
200	6	6	1647,2	1668,6	1,30	2028,4	1928,2	1939,6	1863,8	1886,2	1857,2	11,47	0,0	0,0	0,0	0,0	0,0
200	6	10	745,4	768,2	3,02	975,6	911,2	931,2	890,2	919,6	887,2	16,11	0,0	0,0	0,0	0,0	0,0
200	10	2	1706,4	1706,6	0,01	1896,8	1870,8	1841,2	1815,4	1814,8	1811,0	6,11	0,0	0,0	0,0	0,0	0,0
200	10	6	358,2	363,2	1,38	459,2	435,2	473,2	428,8	449,0	425,2	17,34	0,0	0,0	0,0	0,0	0,0
200	10	10	68,6	69,2	0,68	106,6	96,0	121,0	87,2	95,0	87,2	29,56	0,0	0,0	0,0	0,0	0,0
250	2	2	8267	22585,8	5,730957	22655	21964,6	22622,4	21960,6	21944,2	21911	-2,91	0	0	0	0	0
250	2	6	0	19794,8	-	19410,2	19065	19461,2	18869,8	18882,8	18866	-	0	0	0	0	0
250	2	10	0	16916,6	-	16128,6	15883	16105,2	15702,4	15728,4	15668,6	-	0	0	0	0	0
250	6	2	5931	6129,8	3,360081	6326,6	6183,4	6365	6130	6130,6	6119,4	-0,20	0	0	0	0	0
250	6	6	3163,6	3602,4	13,85121	3848,8	3691,4	3560,6	3508,4	3515	3498,4	-3,25	0	0	0	0	0
250	6	10	1617,6	2006,6	23,97648	2108	1961,8	1998,8	1907	1952	1907	-6,09	0	0	0	0	0
250	10	2	2938,8	2949,4	0,383684	3202,6	3172,4	3161,6	3100,8	3097,6	3094,6	4,92	0	0	0	0	0
250	10	6	933,2	953	2,129374	1181,6	1105,4	1168,4	1091,4	1107,2	1081,2	13,73	0	0	0	0	0
250	10	10	331,2	339	2,423472	443,2	416,8	459,4	411,4	432,6	410,6	21,50	0	0	0	0	0
300	2	2	0	32978,2	-	33000	32203,6	33051,2	32116,8	32126,4	32097,4	-	0	0	0	0	0
300	2	6	0	28224,6	-	28059,8	27320,8	27911,6	27217,8	27179,6	27124,4	-	0	0	0	0	0
300	2	10	0	25157,4	-	24461,2	24351,6	24580,8	23819	23942	23819	-	0	0	0	0	0
300	6	2	8874,2	9460,6	6,592448	9488,2	9307,2	9556,2	9228,8	9220,6	9212	-2,79	0	0	0	0	0
300	6	6	5302,4	6135,6	15,81653	6117,6	5938,4	5886,4	5728,2	5734,4	5725,4	-7,84	0	0	0	0	0
300	6	10	3106,4	4134,8	33,27978	3876,2	3649,6	3691,6	3602,2	3638	3592,6	-17,62	0	0	0	0	0
300	10	2	4686,8	4956	5,862151	5048,2	4954,2	5036,6	4883,4	4888	4880	-1,74	0	0	0	0	0
300	10	6	1927,4	2261	17,65598	2444,6	2302	2271,2	2215,8	2244,2	2209,2	-2,90	0	0	0	0	0
300	10	10	724,2	788,6	8,827927	936,8	897,8	948,8	886,4	930,6	874,8	11,98	0	0	0	0	0

Tabela 5 – Resultados computacionais para as instâncias do conjunto C

|N|	|M|	\tau	best lb	best ub	gap	CSI1	CSI2	CSI3	CSI4	CSI5	vbest	gap'	CSI1 optimum	CSI2 optimum	CSI3 optimum	CSI4 optimum	CSI5 optimum
40	2	2	1208,2	1208,2	0,00	1231,4	1214,0	1235,8	1212,0	1242,4	1210,8	0,22	0,0	1,0	0,0	1,0	0,0
40	2	6	685,2	685,2	0,00	737,8	715,4	712,6	689,4	712,8	688,4	0,46	0,0	0,0	0,0	0,0	0,0
40	2	10	330,0	330,0	0,00	349,8	339,6	355,0	331,8	340,8	331,8	0,60	0,0	1,0	1,0	2,0	1,0
40	6	2	216,4	216,4	0,00	222,6	219,4	222,4	217,0	222,6	217,0	0,32	0,0	2,0	0,0	4,0	0,0
40	6	6	34,6	34,6	0,00	36,6	34,6	35,0	34,6	34,8	34,6	0,00	3,0	5,0	4,0	5,0	4,0
40	6	10	7,2	7,2	0,00	7,2	7,2	7,6	7,2	7,2	7,2	0,00	5,0	5,0	4,0	5,0	5,0
40	10	2	58,8	58,8	0,00	59,8	58,8	60,0	58,8	60,8	58,8	0,00	1,0	5,0	2,0	5,0	2,0
40	10	6	3,4	3,4	0,00	3,8	3,4	3,8	3,4	4,0	3,4	0,00	4,0	5,0	4,0	5,0	3,0
40	10	10	0,0	0,0	0,00	0,0	0,0	0,0	0,0	0,0	0,0	0,00	5,0	5,0	5,0	5,0	5,0
60	2	2	2736,6	2736,6	0,00	2796,6	2754,2	2790,8	2746,8	2807,2	2746,0	0,34	0,0	0,0	0,0	0,0	0,0
60	2	6	1792,6	1792,6	0,00	1894,8	1858,4	1860,6	1814,6	1880,2	1814,6	1,18	0,0	0,0	0,0	0,0	0,0
60	2	10	1076,0	1076,0	0,00	1134,6	1114,8	1109,6	1104,2	1129,8	1096,0	1,90	0,0	0,0	0,0	0,0	0,0
60	6	2	657,0	657,0	0,00	678,8	671,0	679,2	662,4	681,8	662,2	0,81	0,0	0,0	0,0	0,0	0,0
60	6	6	168,8	168,8	0,00	177,4	169,2	171,6	170,2	176,0	169,0	0,13	1,0	3,0	1,0	2,0	0,0
60	6	10	48,4	48,4	0,00	49,8	48,6	50,8	48,6	52,8	48,6	0,37	2,0	4,0	2,0	4,0	1,0
60	10	2	238,0	238,0	0,00	246,6	243,2	245,4	239,4	249,4	239,4	0,61	0,0	0,0	0,0	1,0	0,0
60	10	6	35,2	35,2	0,00	36,2	35,2	36,4	35,4	37,6	35,2	0,00	1,0	5,0	2,0	4,0	0,0
60	10	10	1,8	1,8	0,00	2,2	1,8	1,8	1,8	2,0	1,8	0,00	4,0	5,0	5,0	5,0	4,0
80	2	2	4907,6	4907,6	0,00	5126,2	4930,2	5109,6	4929,4	5090,6	4926,4	0,38	0,0	0,0	0,0	0,0	0,0
80	2	6	3906,6	3906,6	0,00	4135,0	3997,4	4102,8	3945,8	4080,8	3945,8	1,01	0,0	0,0	0,0	0,0	0,0
80	2	10	2523,6	2523,6	0,00	2662,0	2617,8	2612,8	2562,8	2623,2	2558,6	1,41	0,0	0,0	0,0	0,0	0,0
80	6	2	1334,0	1334,0	0,00	1373,2	1358,6	1378,2	1344,8	1380,6	1344,8	0,84	0,0	0,0	0,0	0,0	0,0
80	6	6	440,8	440,8	0,00	464,4	457,4	460,0	450,0	462,0	448,6	1,86	0,0	0,0	0,0	0,0	0,0
80	6	10	185,0	185,0	0,00	194,4	188,8	193,6	188,8	191,6	188,0	1,76	0,0	0,0	0,0	1,0	0,0
80	10	2	534,8	534,8	0,00	552,6	544,8	561,8	538,2	544,6	538,2	0,71	0,0	0,0	0,0	1,0	0,0
80	10	6	107,4	107,4	0,00	113,2	109,0	112,6	109,6	114,4	108,8	1,11	0,0	2,0	0,0	1,0	0,0
80	10	10	14,2	14,2	0,00	15,4	14,6	15,0	14,2	15,8	14,2	0,00	1,0	3,0	2,0	5,0	2,0
100	2	2	8035,0	8035,0	0,00	8355,0	8096,0	8357,8	8082,4	8199,0	8082,4	0,59	0,0	0,0	0,0	0,0	0,0
100	2	6	6533,4	6533,4	0,00	6754,2	6678,2	6754,6	6599,2	6730,0	6599,0	1,01	0,0	0,0	0,0	0,0	0,0
100	2	10	4852,8	4852,8	0,00	5084,8	5047,6	5070,8	4908,6	4955,4	4908,6	1,14	0,0	0,0	0,0	0,0	0,0
100	6	2	2055,6	2055,6	0,00	2141,0	2083,6	2144,2	2070,0	2116,0	2070,0	0,70	0,0	0,0	0,0	0,0	0,0
100	6	6	1005,0	1005,0	0,00	1067,4	1065,6	1059,4	1027,6	1084,6	1026,8	2,21	0,0	0,0	0,0	0,0	0,0
100	6	10	361,8	361,8	0,00	394,8	381,0	382,2	372,2	393,0	369,8	2,25	0,0	0,0	0,0	0,0	0,0
100	10	2	1003,8	1003,8	0,00	1046,0	1038,6	1046,6	1017,0	1044,8	1017,0	1,31	0,0	0,0	0,0	0,0	0,0
100	10	6	212,0	212,0	0,00	223,0	218,8	223,4	218,6	233,6	216,4	2,37	0,0	0,0	0,0	0,0	0,0
100	10	10	46,6	46,6	0,00	48,6	48,2	50,6	47,2	50,2	47,2	1,54	1,0	2,0	1,0	3,0	1,0

Para o conjunto A, 27,11% das nossas soluções apresentam valores exatamente iguais aos do *best lb*, enquanto os resultados do conjunto B têm apenas 9,26% e do conjunto C 35,56%. Estabelecendo comparações com o limite superior (*best ub*), dado pelos valores de Rietz *et al.* (2015), apresentamos no gráfico 1 a percentagem de instâncias em que os nossos resultados foram melhores, iguais ou piores, para cada um dos conjuntos.

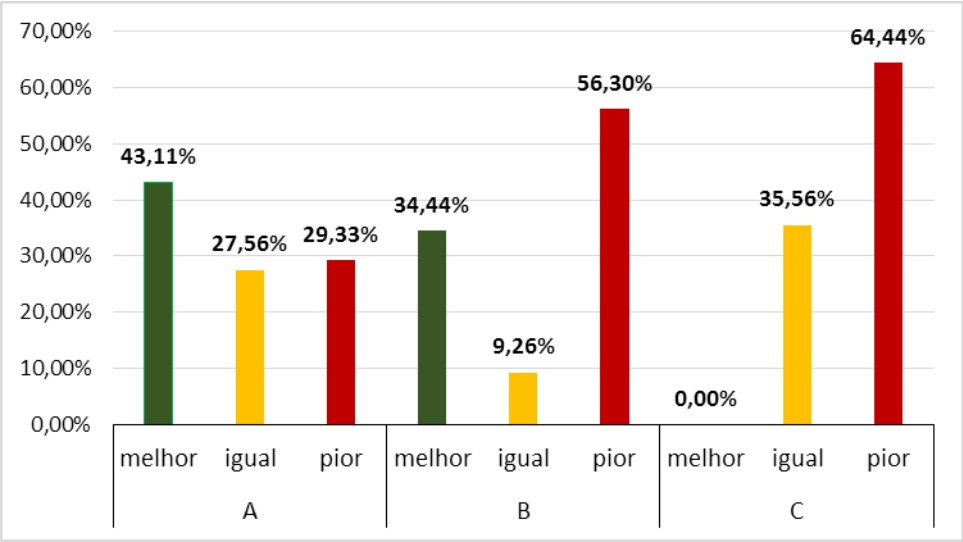


Gráfico 1 – Comparação dos melhores resultados computacionais obtidos com os *best ub*

O gráfico 2 apresenta-nos o valor do *gap* médio (%) com o limite inferior (*best lb*), nos diferentes conjuntos, das nossas melhores soluções (a azul) e das soluções de *best ub* (a vermelho).

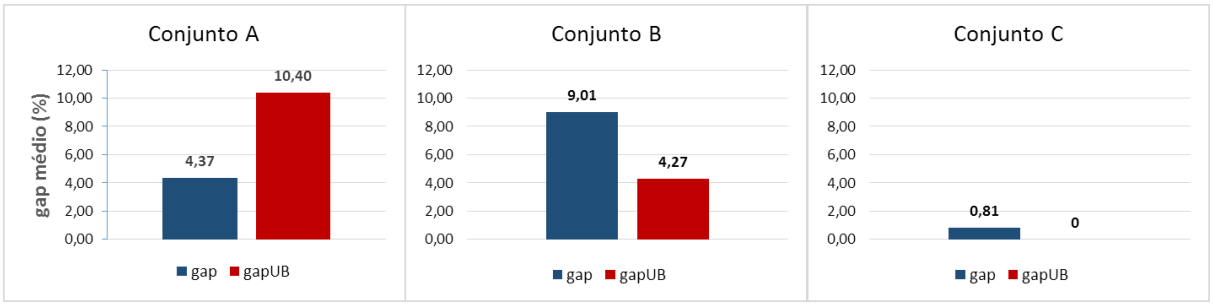


Gráfico 2 – Comparação do *gap* médio (%) para os conjuntos A, B e C

Para o conjunto A, como o valor de *gap* médio nos indica, temos os nossos valores mais próximos da solução ótima do que os obtidos por Rietz *et al.* (2015). A quantidade de soluções que são melhores neste conjunto também é maior (43,11% dos casos contra 29,33%). Enquanto nos conjuntos B e C verifica-se o contrário para ambas as situações (valores de *gap* médios e quantidade de soluções melhores). Assim, podemos considerar que os problemas com tarefas de menor tempo de processamento (conjunto A) são aqueles em que as nossas abordagens conseguem apresentar valores melhores.

O *gap* médio (%) de cada uma das nossas abordagens para os diferentes conjuntos está representado no gráfico 3.

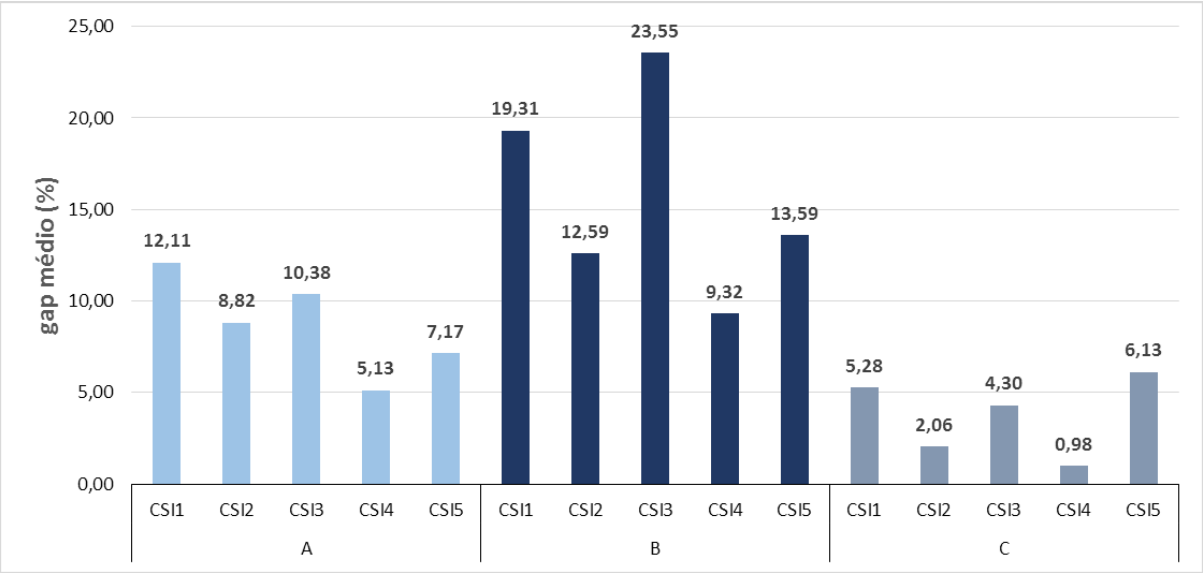


Gráfico 3 – *gap* médio (%) de cada abordagem nos três conjuntos

Para todos os conjuntos a abordagem CSI4 apresentou os melhores resultados. Podemos também constatar que as abordagens que contemplam regras de prioridade para a ordem das tarefas são as que conseguem as soluções mais favoráveis.

No gráfico seguinte (gráfico 4) temos os valores médios da primeira solução para as diferentes abordagens e para cada conjunto.

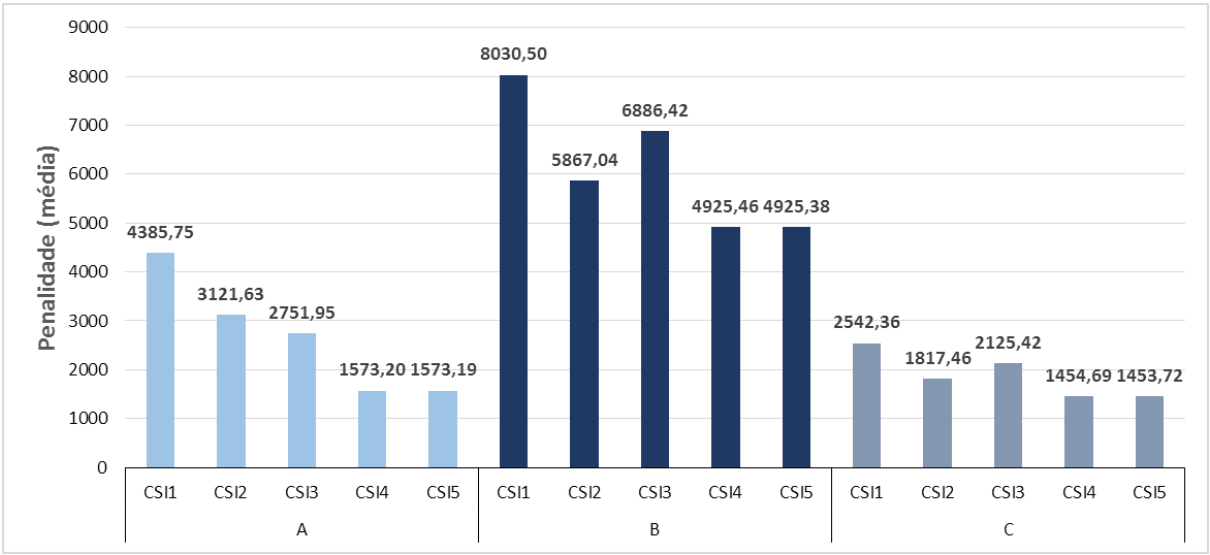


Gráfico 4 – Valor médio da primeira solução em cada abordagem nos conjuntos A, B e C

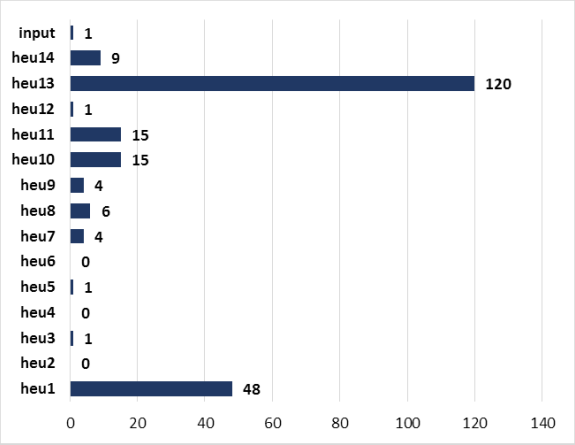
Como seria de esperar a abordagem CSI5 é aquela que apresenta sempre a média mais baixa das primeiras soluções. No entanto, como podemos verificar no gráfico 3, não é esta a abordagem que nos dá os melhores resultados. No caso do conjunto C, a abordagem CSI5 é mesmo a pior.

Nos resultados obtidos através da abordagem CSI5, apuramos que para o conjunto A foi sempre escolhida a construção de soluções iniciais a partir da data esperada das tarefas (CSI3 ou CSI4) como a que dava a melhor solução inicial. Relativamente à sequência ou ordem inicial das tarefas, foram escolhidas de acordo com a figura 16 – a). Nestas temos um caso em que é considerada a ordem do *input*, ou seja, CSI3. Todos os outros são CSI4 e a variante *heu13* foi a que obteve melhores resultados para a solução inicial. No entanto, como vemos na figura 16 – b), a variante que resulta mais vezes na melhor solução é a *heu14*.

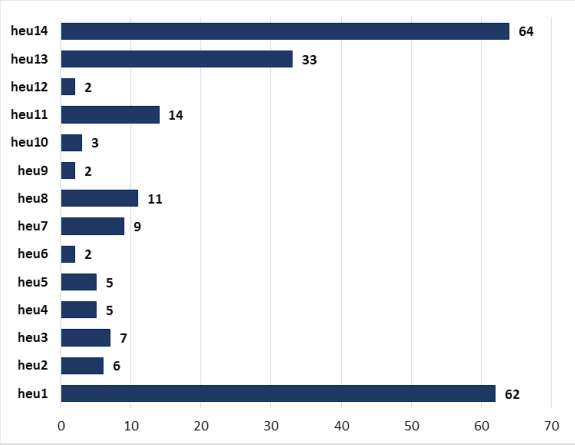
No conjunto B temos uma maior, mas não muita, correlação entre as melhores variantes para a primeira solução e as variantes que efetivamente deram origem à solução final figura 16 – c) e d). Mas os resultados obtidos neste conjunto B não foram os melhores, apresentando-se como a terceira melhor abordagem entre as consideradas neste projeto.

No conjunto C (figura 16 – e) e f)), a sequência que dá os melhores resultados para a primeira solução, curiosamente, nunca foi usada para alcançar a melhor solução. Os resultados de C com esta abordagem CSI5 apresentam-se como sendo os mais elevados (piores).

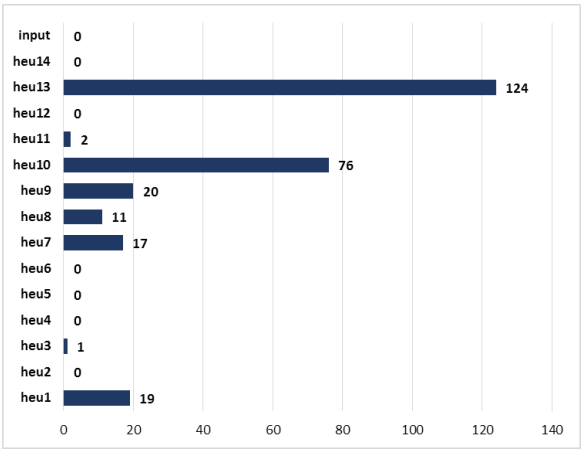
a) Sequências inicial usada em A



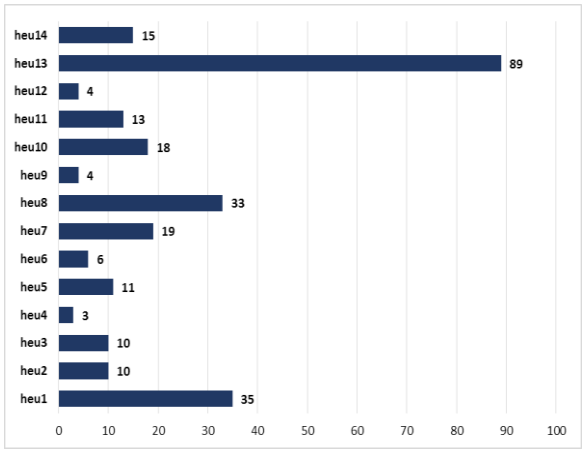
b) Variante que resulta no melhor resultado em A



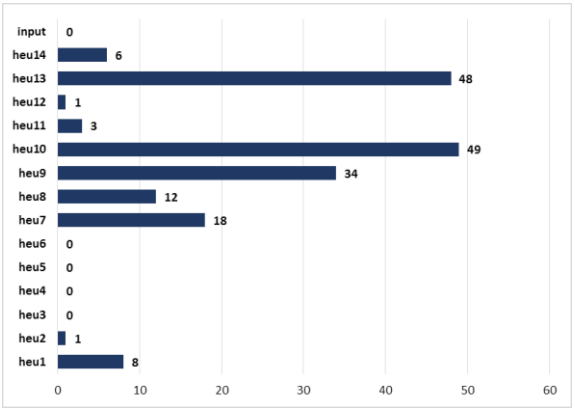
c) Sequências inicial usada em B



d) Variante que resulta no melhor resultado em B



e) Sequências inicial usada em C



f) Variante que resulta no melhor resultado em C

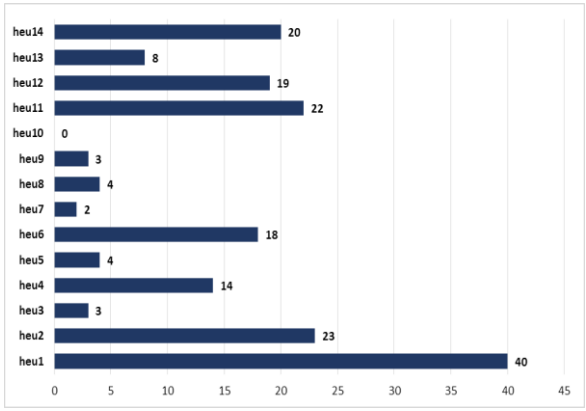
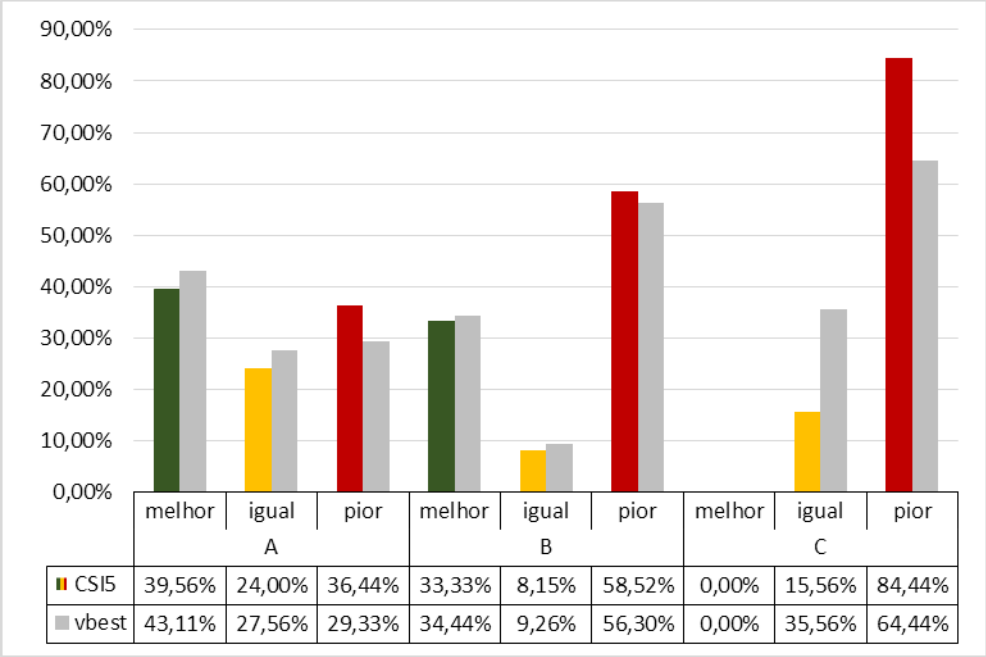


Figura 16 – Variantes da primeira solução e variantes dos melhores resultados obtidos

Torna-se também interessante comparar as soluções encontradas em CSI5 com o limite superior (*best ub*), tal como fizemos no gráfico 1 com os nossos melhores valores, comparando quando este é melhor, igual ou pior que os melhores resultados de Rietz *et al.* (2015). No gráfico 5 conjugamos as duas análises. Prova-se, mais uma vez, que os resultados da abordagem CSI5 não são os nossos melhores resultados (gráfico 6).



	melhor	igual	pior	melhor	igual	pior	melhor	igual	pior
	A			B			C		
CSI5	39,56%	24,00%	36,44%	33,33%	8,15%	58,52%	0,00%	15,56%	84,44%
vbest	43,11%	27,56%	29,33%	34,44%	9,26%	56,30%	0,00%	35,56%	64,44%

Gráfico 5 – Comparação dos resultados de CSI5 e *vbest* com os *best ub*

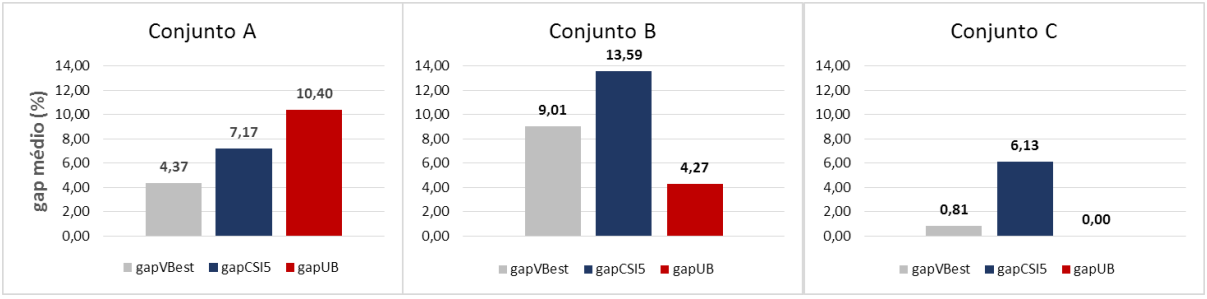


Gráfico 6 – Comparação do *gap* médio (%) de CSI5, *vbest* e *best ub* para os conjuntos A, B e C

Com esta análise podemos concluir que a melhor solução inicial não nos garante a melhor solução final. No entanto, e através da análise do gráfico 3 e 4, para a abordagem CSI4, percebemos que mesmo que não seja a melhor de todas as soluções inicias, encontrar uma boa solução inicial é fundamental para alcançar os melhores resultados (a média das soluções inicias de CSI4 não diferem muito das de CSI5).

No gráfico 7 temos o tempo médio para encontrar a solução final e o tempo médio de execução do algoritmo (ambos em segundos). A diferença entre os tempos ocorre porque as trocas entre tarefas que aconteceram depois do tempo em que é encontrada a solução, que se tornou a solução final, não resultaram numa melhor solução.

É sabido que para cada experiência computacional foi considerada uma execução de 5 segundos só para a componente do VNS, mas podemos ter médias inferiores a 5 segundos pois obtivemos resultados que através da construção da primeira solução já sejam ótimos e não executam a parte do VNS. De forma oposta, ter médias bem superiores a 5 segundos deve-se ao varrimento que acontece depois da execução do VNS. É possível verificar que das abordagens de CSI1 para CSI5 o tempo de execução vai diminuindo em todos os conjuntos.

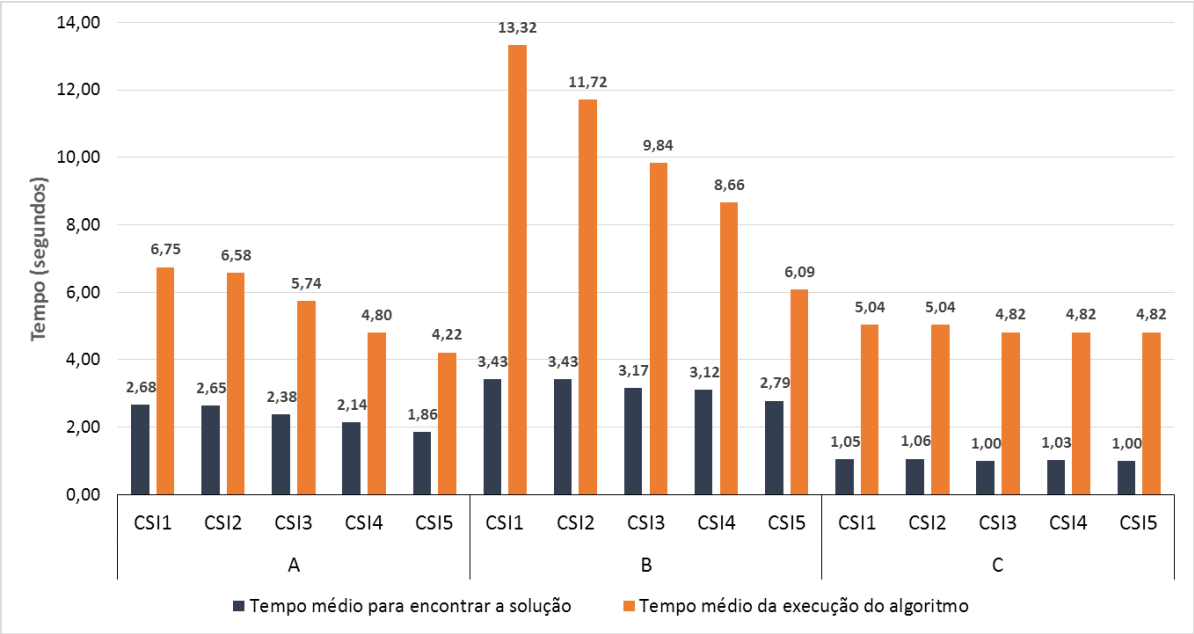


Gráfico 7 – Tempo médio para encontrar a solução e de execução do algoritmo

Tanto a semelhança dos tempos médios de execução encontrados como o baixo valor do tempo médio para encontrar a solução, entre todas as abordagens para o conjunto C, podem ser indicativos de que as estruturas de vizinhança consideradas possam não estar a conseguir provocar as variações necessárias para este tipo particular de problemas.

6. CONCLUSÕES

Esta dissertação centra-se na tentativa de chegar a novas abordagens de otimização para o problema integrado de planeamento e escalonamento de operações. Mais uma vez referimos que foi tido em conta o problema descrito por Kis e Kovács (2012), onde o horizonte de tempo do planeamento é dividido em períodos de tempo. Para cada período existem máquinas paralelas idênticas com disponibilidade limitada, nas quais são alocadas tarefas. As tarefas têm um conjunto de caraterísticas entre as quais: a data de lançamento, a data esperada/prazo de entrega, o tempo de processamento e penalizações em que cada tarefa incorre se não estiver pronta no seu prazo (período) de entrega, ora por ser realizada antes ou depois da data. O objetivo global é minimizar o valor ou custo das penalidades totais incorridas.

Feito um estudo aos problemas de escalonamento existentes na literatura, mais especificamente a problemas de máquinas paralelas e idênticas, percebeu-se a forte ligação destes com os problemas de C&P e, portanto, técnicas específicas deste tipo de problemas foram tidas em conta.

As novas abordagens consideradas neste projeto, tal como já vimos, têm por base métodos heurísticos de pesquisa local, a meta-heurística de pesquisa de vizinhança variável (VNS) recorrendo a duas estruturas de vizinhança e uma outra que pode ser vista como uma variante do *multistart*. Foram propostos dois algoritmos diferentes para a construção das soluções iniciais e quinze sequências que determinavam diferentes ordens na alocação das tarefas às máquinas, conseguindo assim alcançar um grande espaço de soluções.

Terminada a análise dos resultados podemos comparar os resultados das diferentes abordagens realizadas entre si, bem com, compará-los com outros resultados já existentes na literatura. A vantagem do uso de instâncias de referência da literatura permite-nos verificar se os nossos resultados são competitivos com outros estudos efetuados anteriormente. Comparando com os resultados de Rietz *et al.* (2015) apresentamos abordagens que se mostraram mais eficazes para problemas com tarefas cujo tempo de processamento é relativamente curto comparado com o tempo total do período (instâncias do conjunto A).

Tendo em conta os resultados obtidos temos como a nossa melhor abordagem a CSI4, ou seja, a solução inicial construída a partir da data esperada fazendo uso de regras de prioridade. Sendo as ordens de prioridade *heu13* e *heu14* as que mais vezes conseguiram alcançar melhores resultados (critério 1: p_j/l_j crescente e em caso de empate, critério 2: p_j decrescente (*heu13*) ou crescente (*heu14*)).

Constatamos também que as soluções alcançadas por abordagens sem ter em conta as regras de prioridade são as que apresentam mais vezes uma menor qualidade nas soluções. A abordagem CSI3 que tem uma construção da solução inicial a partir da data esperada, isto é, preocupada em garantir as menores penalidades associadas a cada tarefa alocada, tem valores piores (valores mais altos) comparativamente à abordagem CSI2 que não tem uma preocupação com o valor das penalidades na construção da solução inicial. Tendo assim, um maior relevo a componente que diz respeito à sequência das tarefas do que propriamente a parte construtiva da solução.

Relativamente a CSI5, tal como vários autores já relataram para abordagens envolvendo o *multistart*, não podemos concluir que a solução seja garantidamente melhor ou pior. A abordagem CSI5, se o melhor começo garantisse a melhor solução, deveria ser pelo menos tão boa como a abordagem CSI4, podendo existir ligeiras variações devido à componente aleatória proveniente do método VNS.

Como trabalhos futuros sem grandes adaptações ao modelo aqui apresentado podemos ver pequenas variantes do problema descrito. Por exemplo, considerar máquinas não idênticas e períodos de tamanho variável.

BIBLIOGRAFIA

Aarts, E. H., Lenstra, J. K., 2003, Local Search in Combinatorial Optimization, Princeton University Press

Abdulmalek, F. A., Rajgopal, J., 2007, Analyzing the benefits of lean manufacturing and value stream mapping via simulation: A process sector case study: International Journal of Production Economics, v. 107, p. 223-236.

Baker, K. R., Scudder, G. D., 1990, Sequencing with earliness and tardiness penalties - A review: Operations Research, v. 38, p. 22-36.

Baldea, M., Harjunkoski, I., 2014, Integrated production scheduling and process control: A systematic review: Computers & Chemical Engineering, v. 71, p. 377-390.

Biskup, D., 1999, Single-machine scheduling with learning considerations: European Journal of Operational Research, v. 115, p. 173-178.

Chang, Y. C., Lee, C. Y., 2004, Machine scheduling with job delivery coordination: European Journal of Operational Research, v. 158, p. 470-487.

Coffman, E. G., Garey, M. R., Johnson, D. S., 1978, Application of bin-packing to multiprocessor scheduling: Siam Journal on Computing, v. 7, p. 1-17.

Conway, R. W., Maxwell, W. L., and Miller, L. W., 1967, Theory of Scheduling, Addison-Wesley, Reading, Massachusetts

Delorme, M., Iori, M., Martello, S., 2016, Bin packing and cutting stock problems: Mathematical models and exact algorithms: European Journal of Operational Research, v. 255, p. 1-20.

Dyckhoff, H., 1990, A typology of cutting and packing problems: European Journal of Operational Research, v. 44, p. 145-159.

Dyckhoff, H., Finke, U., 1992, Cutting and packing in production and distribution: typology and bibliography, Springer-Verlag Berlin Heidelberg

Garey, M. R., Johnson, D. S., 1978, Strong NP-Completeness results - motivation, examples, and implications: Journal of the Acm, v. 25, p. 499-508.

Geismar, H. N., Laporte, G., Lei, L., Sriskandarajah, C., 2008, The integrated production and transportation scheduling problem for a product with a short lifespan: Informs Journal on Computing, v. 20, p. 21-33.

Gilmore, P. C., Gomory, R. E., 1961, A linear-programming approach to the cutting-stock problem: Operations Research, v. 9, p. 849-859.

Glover, F., 1986, Future paths for integer programming and links to artificial-intelligence: Computers & Operations Research, v. 13, p. 533-549.

Gordon, V., Proth, J. M., Chu, C. B., 2002, A survey of the state-of-the-art of common due date assignment and scheduling research: European Journal of Operational Research, v. 139, p. 1-25.

Gradisar, M., Resinovic, G., Kljajic, M., 2002, Evaluation of algorithms for one-dimensional cutting: Computers & Operations Research, v. 29, p. 1207-1220.

Graham, R. L., Lawler, E. L., Lenstra, J. K., Kan, A. H. 1979, Optimization and approximation in deterministic sequencing and scheduling: A survey, Annals of Discrete Mathematics 5, 287-326

Grossmann, I. E., 2009, Research Challenges in Planning and Scheduling for Enterprise-Wide Optimization of Process Industries: 10th International Symposium on Process Systems Engineering, v. 27, p. 15-21.

Haessler, R. W., Sweeney, P. E., 1991, Cutting stock problems and solution procedures: European Journal of Operational Research, v. 54, p. 141-150.

Hansen, P., Mladenović, N., Perez, J. A., 2010, Variable neighbourhood search: methods and applications: Annals of Operations Research, v. 175, p. 367-407.

Hoogeveen, H., 2005, Multicriteria scheduling: European Journal of Operational Research, v. 167, p. 592-623.

Jain, A. S., Meeran, S., 1999, Deterministic job-shop scheduling: Past, present and future: European Journal of Operational Research, v. 113, p. 390-434.

Kantorovich, L. V., 1960, Mathematical-methods of organizing and planning production: Management Science, v. 6, p. 366-422.

Kis, T., Kovács, A., 2012, A cutting plane approach for integrated planning and scheduling: Computers & Operations Research, v. 39, p. 320-327.

Maravelias, C. T., Sung, C., 2009, Integration of production planning and scheduling: Overview, challenges and opportunities: Computers & Chemical Engineering, v. 33, p. 1919-1930.

Michiels, W., Aarts, E. H., Korst, J., 2007, Theoretical Aspects of Local Search, Springer Science & Business Media

Mladenović, N., Hansen, P., 1997, Variable neighborhood search: Computers & Operations Research, v. 24, p. 1097-1100.

Nagar, A., Haddock, J., Heragu, S., 1995, Multiple and bicriteria scheduling - A literature survey: European Journal of Operational Research, v. 81, p. 88-104.

Panwalkar, S. S., Smith, M. L., Seidmann, A., 1982, Common due date assignment to minimize total penalty for the one machine scheduling problem: Operations Research, v. 30, p. 391-399.

Pinedo, M., 2002, Scheduling: Theory, Algorithms, and Systems, Prentice Hall

Pinedo, M., 2008, Scheduling: Theory, Algorithms, and Systems, Springer

Rietz, J., Alves, C., Braga, N., Carvalho, J. V., 2016, An exact approach based on a new pseudo-polynomial network flow model for integrated planning and scheduling: Computers & Operations Research, v. 76, p. 183-194.

Rietz, J., Alves, C., Carvalho, J. V., 2015, Fast Heuristics for Integrated Planning and Scheduling: Computational Science and Its Applications - Iccsa 2015, Pt Ii, v. 9156, p. 413-428.

Shen, W. M., Wang, L. H., Hao, Q., 2006, Agent-based distributed manufacturing process planning and scheduling: A state-of-the-art survey: Ieee Transactions on Systems Man and Cybernetics Part C-Applications and Reviews, v. 36, p. 563-577.

Shobrys, D. E., White, D. C., 2000, Planning, scheduling and control systems: why can they not work together: Computers & Chemical Engineering, v. 24, p. 163-173.

Sugimori, Y., Kusunoki, K., Cho, F., Uchikawa, S., 1977, Toyota production system and kanban system materialization of just-in-time and respect-for-human system: International Journal of Production Research, v. 15, p. 553-564.

Sule, D. R., 2007, Production Planning and Industrial Scheduling: Examples, Case Studies and Applications, Second Edition

Sweeney, P. E., Paternoster, E. R., 1992, Cutting and packing problems - A categorized, application-orientated research bibliography: Journal of the Operational Research Society, v. 43, p. 691-706.

Talbi, E., 2009, Metaheuristics: From Design to Implementation, John Wiley & Sons

Wascher, G., HauBner, H., Schumann, H., 2007, An improved typology of cutting and packing problems: European Journal of Operational Research, v. 183, p. 1109-1130.

ANEXOS

ANEXO I

CONSTRUÇÃO DA SOLUÇÃO INICIAL CSI4 E SOLUÇÕES VIZINHAS

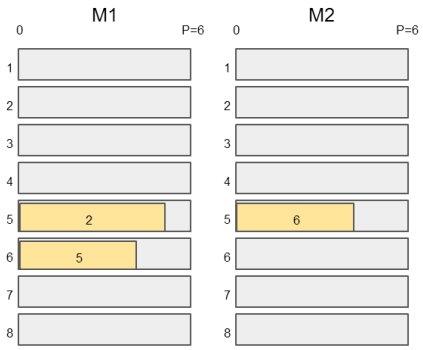
a)

Tarefas ($j \in N$): ***heu1***

j	*rj*	*dj*	*pj*	*ej*	*lj*		*(ej+lj)/ pj*
2	5	5	5	0	100		20
6	2	5	4	20	50		17
1	5	5	6	0	100		16
5	6	6	4	0	50		12
4	3	5	4	10	30		10
3	5	5	5	0	25		5
7	2	5	4	5	8		3

Nova ordem dos trabalhos:
2 - 6 - 1 - 5 - 4 - 3 - 7

b)

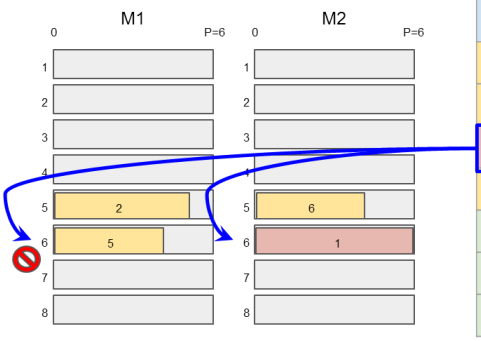


Tarefas ($j \in N$): ***heu1***

j	*rj*	*dj*	*pj*	*ej*	*lj*		*(ej+lj)/ pj*	*F.O.*
2	5	5	5	0	100		20	0
6	2	5	4	20	50		17	0
1	5	5	6	0	100		16	
5	6	6	4	0	50		12	0
4	3	5	4	10	30		10	
3	5	5	5	0	25		5	
7	2	5	4	5	8		3	

Colocação de todas as tarefas possíveis na sua data esperada pela ordem estabelecida

c)



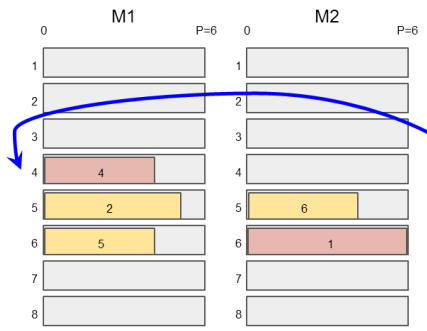
Tarefas ($j \in N$): ***heu1***

j	*rj*	*dj*	*pj*	*ej*	*lj*		*(ej+lj)/ pj*	*F.O.*
2	5	5	5	0	100		20	0
6	2	5	4	20	50		17	0
1	5	5	6	0	100		16	100
5	6	6	4	0	50		12	0
4	3	5	4	10	30		10	
3	5	5	5	0	25		5	
7	2	5	4	5	8		3	

Colocação das tarefas que não couberam nas suas datas esperadas vendo se compensa adiantar ou atrasar

tarefa 1 tem $r_j = d_j$ pelo que só pode ser adiada

d)

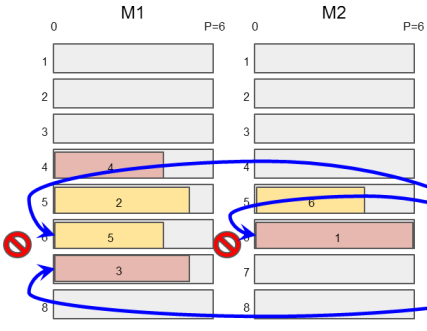


Tarefas ($j \in N$): *heu1*

j	*rj*	*dj*	*pj*	*ej*	*lj*		*(ej+lj)/ pj*	*F.O.*
2	5	5	5	0	100		20	0
6	2	5	4	20	50		17	0
1	5	5	6	0	100		16	100
5	6	6	4	0	50		12	0
4	3	5	4	10	30		10	10
3	5	5	5	0	25		5	
7	2	5	4	5	8		3	

tarefa 4 tem $r_4 \neq d_4$, ou seja, pode ser adiantada ou adiada
Como $e_4 < l_4$, então compensa adiantar.

e)

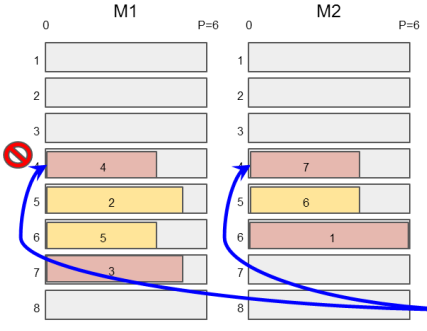


Tarefas ($j \in N$): *heu1*

j	*rj*	*dj*	*pj*	*ej*	*lj*		*(ej+lj)/ pj*	*F.O.*
2	5	5	5	0	100		20	0
6	2	5	4	20	50		17	0
1	5	5	6	0	100		16	100
5	6	6	4	0	50		12	0
4	3	5	4	10	30		10	10
3	5	5	5	0	25		5	50
7	2	5	4	5	8		3	

A tarefa 3 só pode ser adiada e tem de se afastar 2
períodos da sua data esperada

f)

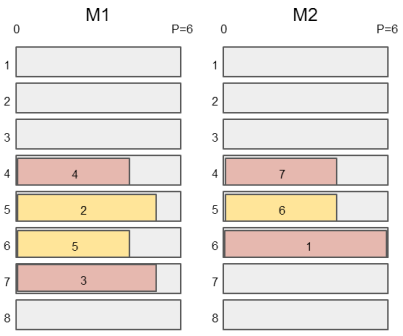


Tarefas ($j \in N$): *heu1*

j	*rj*	*dj*	*pj*	*ej*	*lj*		*(ej+lj)/ pj*	*F.O.*
2	5	5	5	0	100		20	0
6	2	5	4	20	50		17	0
1	5	5	6	0	100		16	100
5	6	6	4	0	50		12	0
4	3	5	4	10	30		10	10
3	5	5	5	0	25		5	50
7	2	5	4	5	8		3	5

tarefa 7 também compensa adiantar

g)

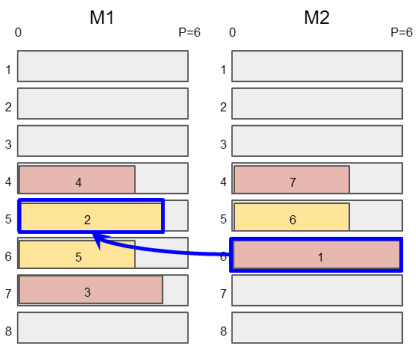


Tarefas ($j \in N$): *heu1*

j	*rj*	*dj*	*pj*	*ej*	*lj*		*(ej+lj)/ pj*	*F.O.*
2	5	5	5	0	100		20	0
6	2	5	4	20	50		17	0
1	5	5	6	0	100		16	100
5	6	6	4	0	50		12	0
4	3	5	4	10	30		10	10
3	5	5	5	0	25		5	50
7	2	5	4	5	8		3	5

Solução inicial do tipo CSI4 *heu1*, com uma penalização de 165.

h)

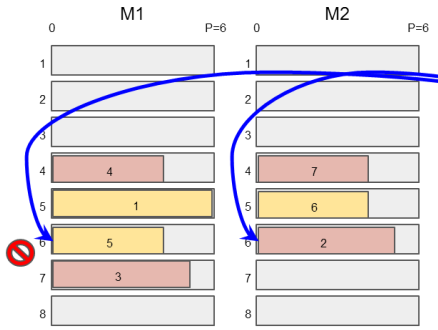


Tarefas ($j \in N$):

j	*rj*	*dj*	*pj*	*ej*	*lj*		*F.O.*
1	5	5	6	0	100		100
3	5	5	5	0	25		50
4	3	5	4	10	30		10
7	2	5	4	5	8		5
2	5	5	5	0	100		0
6	2	5	4	20	50		0
5	6	6	4	0	50		0

Ordena-se a lista das tarefas pela penalidade associada a cada tarefa de forma decrescente. E tenta-se trocar a primeira tarefa (a mais penalizadora, neste caso a tarefa 1) com as tarefas que estão em períodos correspondentes à sua data esperada (período 5). Primeiro tenta a troca com a tarefa 2

i)

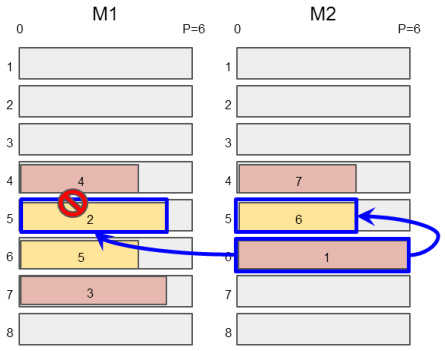


Tarefas ($j \in N$):

j	*rj*	*dj*	*pj*	*ej*	*lj*		*F.O.*
2	5	5	5	0	100		100

A tarefa 2 tem a rj = dj pelo que só pode ser adiada. O período onde resulta na menor penalidade para a tarefa 2 é na antiga posição da tarefa 1 (troca direta). No entanto, não resulta numa melhoria em relação à situação anterior. Neste caso igualam, mas não melhora. Assim a tarefa 1 terá de verificar as próximas tarefas.

j)

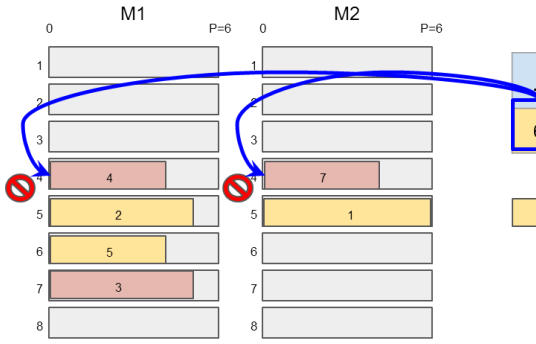


Tarefas ($j \in N$):

j	r_j	d_j	p_j	e_j	l_j		F.O.
1	5	5	6	0	100		100
3	5	5	5	0	25		50
4	3	5	4	10	30		10
7	2	5	4	5	8		5
2	5	5	5	0	100		0
6	2	5	4	20	50		0
5	6	6	4	0	50		0

Não compensa trocar a tarefa 1 com a tarefa 2.
A troca para o lugar da tarefa 6 como vemos vai resultar numa menor penalidade

k)

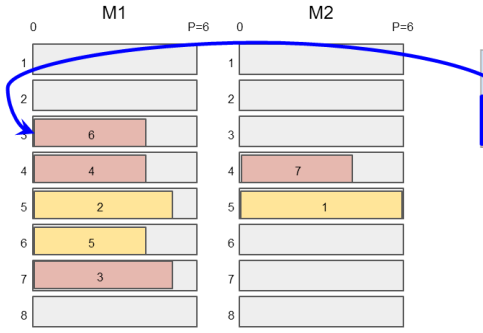


j	r_j	d_j	p_j	e_j	l_j		F.O.
6	2	5	4	20	50		0

6

A colocação da tarefa 6 para a sua melhor posição faz-se de forma idêntica à colocação das tarefas na construção da primeira solução.
Como $e_6 < l_6$, então compensa adiantar.

l)

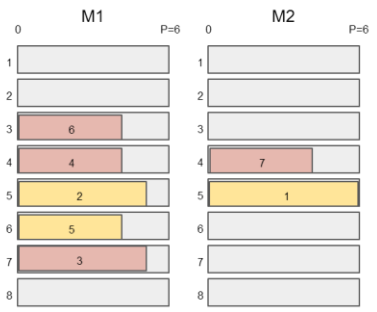


j	r_j	d_j	p_j	e_j	l_j		F.O.
6	2	5	4	20	50		40

A tarefa 6 também não vai caber no período anterior à sua data esperada em nenhuma máquina.
Adiantá-la mais aumentaria o seu custo para Aux_ej = 40 (2 períodos de adianto*20).
Como, 40 < 50 compensa, novamente, adiantar a tarefa 6.
A penalidade da solução baixa em 60 unidades.

Se a penalidade não fosse menor, seriam procuradas trocas melhores com a tarefa 1, o que neste caso não iria acontecer. Não existindo trocas para a tarefa 1 que fossem vantajosas passar-se-ia à avaliação de trocas com a segunda tarefa da lista (tarefa 3).

m)



j	*rj*	*dj*	*pj*	*ej*	*lj*		F.O.
3	5	5	5	0	25		50
6	2	5	4	20	50		40
4	3	5	4	10	30		10
7	2	5	4	5	8		5
2	5	5	5	0	100		0
1	5	5	6	0	100		0
5	6	6	4	0	50		0

Para este exemplo, este é o resultado após a primeira melhoria. Caso seja para fazer um "varrimento" completo à vizinhança há o reordenamento da lista das tarefas de acordo com a penalidade e fazem-se os mesmos procedimentos até não existirem mais melhorias na vizinhança considerada.

ANEXO II

INFORMAÇÕES DA SOLUÇÃO

Tarefas ordenadas segundo a *heu1*

Informações relativas à solução para uma determinada instância:

- Solução final;
- Penalidade da solução final;
- Penalidade da primeira solução;
- Tempo de execução;
- Tempo necessário para encontrar a solução final.