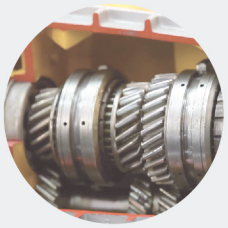
INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO
MESTRADO EM ENGENHARIA E GESTÃO INDUSTRIAL






ESCALONAMENTO DE MÁQUINAS PARALELAS DEDICADAS COM FAMÍLIAS DE SETUP E RECURSOS ADICIONAIS

ÂNGELO EMANUEL VIEIRA SOARES
julho de 2023

POLITÉCNICO
DO PORTO

Instituto Superior de Engenharia do Porto

PROGRAMAÇÃO DE MÁQUINAS PARALELAS DEDICADAS COM SETUPS DEPENDENTES DA SEQUÊNCIA DE FAMÍLIAS E RECURSOS ADICIONAIS

Ângelo Emanuel Vieira Soares

2023

Instituto Superior de Engenharia do Porto

Departamento de Engenharia Mecânica

P.PORTO


P.PORTO

Instituto Superior de Engenharia do Porto

PROGRAMAÇÃO DE MÁQUINAS PARALELAS DEDICADAS COM SETUPS DEPENDENTES DA SEQUÊNCIA DE FAMÍLIAS E RECURSOS ADICIONAIS

Ângelo Emanuel Vieira Soares

Estudante n.º 1180758

Dissertação apresentada ao Instituto Superior de Engenharia do Porto para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Engenharia e Gestão Industrial, realizada sob a orientação do Professor Doutor Manuel Joaquim Pereira Lopes.

2023

Instituto Superior de Engenharia do Porto

Departamento de Engenharia Mecânica

P.PORTO


P.PORTO

AGRADECIMENTOS

Para o desenvolvimento deste projeto foi crucial o apoio e a disponibilização de várias pessoas, as quais merecem que esta secção seja dedicada a elas.

Em primeiro lugar, ao Professor Doutor Manuel Joaquim Pereira Lopes pela orientação e acompanhamento em todos os processos de desenvolvimentos do projeto, fornecendo os conhecimentos necessários para cada etapa. As reuniões semanais foram sem dúvida uma mais-valia para o sucesso do projeto.

À minha família por serem o meu maior apoio emocional e exemplo de superação, que sempre investiram em mim e no meu futuro, tornando possível a minha formação em Engenharia e Gestão Industrial. Estiveram sempre presentes, fazendo-me acreditar que é possível atingir os meus sonhos.

Por último, mas não menos importante, aos meus amigos pela amizade, motivação e ajuda, o que me deu alento na realização do projeto.

página propositadamente em branco

RESUMO

Face ao desafio lançado pela empresa INPLAS para melhorar o planeamento da produção das suas fábricas, foi estabelecido um objetivo geral que consiste no desenvolvimento de um sistema de apoio à decisão para o problema da empresa. Neste sentido, este trabalho foca-se num problema real de programação de máquinas paralelas dedicadas com *setups* dependentes da sequência de famílias e recursos adicionais (PMSR).

Para resolver este problema foram adaptados dois modelos matemáticos. Um modelo segue a abordagem *strip-packing* e o outro é indexado ao tempo. Para refletir as condições reais do problema, foi proposta uma nova função objetivo que consiste na minimização da soma dos *makespan* de todas as máquinas. As soluções obtidas mostraram que a nova função objetivo fornece um cronograma da produção compacto que permite a minimização dos tempos ociosos das máquinas e dos tempos de *setup*, em simultâneo. Tendo em conta que os sistemas de produção são contínuos, introduziu-se uma configuração inicial das máquinas. Além disso, os modelos matemáticos de referência foram generalizados para serem capazes de resolver problemas com *setups* dependentes da sequência de famílias.

Os testes computacionais mostraram que o modelo *strip-packing* tem um desempenho superior comparativamente ao modelo indexado ao tempo. Após esta conclusão, o estudo prosseguiu apenas no modelo *strip-packing*. Como este modelo matemático só se mostrou capaz de resolver instâncias de pequena e média dimensões, foi desenvolvida uma heurística matemática. Uma das estratégias utilizadas na heurística matemática foi o mecanismo de *warm-start* em que se desenvolveu a metaheurística TS para gerar soluções iniciais válidas e que funcionam como *upper bound*. Para reduzir o espaço de soluções inferiormente, desenvolveu-se uma heurística construtiva (SCTUR) que gera soluções que funcionam como *lower bound*.

Em relação à heurística SCTUR, a sua utilização como *lower bound* revelou-se muito positiva. De facto, as soluções obtidas são bastante mais próximas da solução ótima do que os valores da relaxação linear, com desvios médios para a solução ótima de 3,61% e 5,09% para as instâncias pequenas e médias, respetivamente. Quanto ao desempenho do TS, este obteve soluções ótimas em 80% e 50% das instâncias pequenas e médias, respetivamente. Nas instâncias grandes, o *gap* médio para a solução da heurística SCTUR (*lower bound*) foi de 8,88%. Além disso, o TS apresentou resposta computacional razoável até ao conjunto de problemas extra grande II (32 máquinas e 100 tarefas). Comparativamente ao TS do estudo de Bektur & Saraç (2019), o TS proposto neste projeto apresentou um desempenho superior, o que se reflete nos tempos computacionais.

As diferentes abordagens da heurística matemática apresentaram resultados promissores na resolução deste problema complexo, destacando-se a estratégia que combina o TS e a heurística SCTUR. Esta estratégia representou um ganho de 79,61% comparativamente ao desempenho do modelo matemático, obtendo um *gap* médio de 4,24% para as instâncias grandes. Face ao exposto, é possível concluir que as estratégias da heurística matemática têm impacto nas instâncias de maiores dimensões em que o modelo matemático apresenta dificuldades.

PALAVRAS-CHAVE

Escalonamento, Máquinas paralelas, *Setups*, Recursos, *Tabu Search*, Heurística matemática.

página propositadamente em branco

ABSTRACT

Given the challenge presented by the company INPLAS to improve the production planning of their factories, a general objective was established, which consists of developing a decision support system for the company's problem. This work focused on a real world dedicated parallel machine scheduling problem with sequence-dependent setup families and additional resources (PMSR).

To solve this problem, two previously proposed models have been adapted. One model follows the strip-packing approach and the other is time-indexed. A novel objective function, the minimisation of the sum of the makespans of all machines, is proposed to reflect the real conditions of the manufacturing environment that motivates this work. The solutions obtained show that the new objective function provides a compact production schedule that allows the simultaneous minimisation of machine idle times and setup times. Considering that production systems are continuous, an initial configuration of the machines was introduced. Additionally, the reference mathematical models were generalized to solve problems with sequence-dependent setup families.

The computational tests showed that the strip-packing model outperformed the time-indexed model. Based on this conclusion, the study continued solely focusing on the strip-packing model. Since this mathematical model was only able to solve small and medium-sized instances, a matheuristic was developed. The warm-start mechanism was one of the strategies employed in the matheuristic, in which the TS metaheuristic was developed to generate valid initial solutions that work as an upper bound. To reduce the solution space inferiorly, a constructive heuristic (SCTUR) was developed to generate solutions that work as lower bound.

The utilization of the SCTUR heuristic as a lower bound proved to be very positive. Indeed, the achieved solutions are considerably closer to the optimal solution compared to the linear relaxation values, with mean deviations from the optimal solution of 3.61% and 5.09% for the small and medium instances, respectively. TS algorithm achieved optimal solutions in 80% and 50% of small and medium instances, respectively. For large instances, the average gap for the SCTUR heuristic solution (lower bound) was 8.88%. Additionally, the TS presented reasonable computational performance in problems with up to 32 machines and 100 jobs. In comparison to the TS algorithm proposed by Bektur & Saraç (2019), the TS algorithm developed in this project showed superior performance, as evidenced by the computational times.

Matheuristic approaches showed promising results in solving this complex problem, highlighting the strategy that combines TS (warm-start) and SCTUR heuristic. This combined approach achieves an average gap of 4.24% for large instances. Therefore, this strategy represents a gain of 79.61% compared to the performance of the mathematical model. Based on these findings, it can be concluded that the use of matheuristics has a significant impact on larger instances where the mathematical model encounters difficulties.

KEYWORDS

Scheduling, Parallel machines, Setups, Resources, Tabu Search, Matheuristic.

página propositadamente em branco

ÍNDICE

página propositadamente em branco

ÍNDICE DE FIGURAS

página propositadamente em branco

ÍNDICE DE TABELAS

página propositadamente em branco

LISTAS DE SIGLAS

Lista de Siglas

CP	*Constraint Programming*
CPLOT	*Center for Production and Logistics Optimization Technologies*
ISEP	Instituto Superior de Engenharia do Porto
MILP	*Mixed Integer Linear Programming*
PM	*Parallel Machine scheduling problem*
PMFRS	*Parallel Machine Flexible Resource Scheduling*
PMR	*Parallel Machine scheduling problem with Resources*
PMS	*Parallel Machine scheduling problem with sequence dependent Setup times*
PMSR	*Parallel Machine scheduling problem with Setups and Resources*
P. Porto	Instituto Politécnico do Porto
TS	*Tabu Search*
UPMFRS	*Unspecified Parallel Machine Flexible Resource Scheduling*

página propositadamente em branco

1. INTRODUÇÃO

Neste capítulo efetua-se um enquadramento ao projeto no que diz respeito à evolução e atualidade do estudo do problema de máquinas paralelas, expondo-se também a situação da empresa e a necessidade da solução proposta. De seguida, enunciam-se as questões de investigação que levaram ao estabelecimento do objetivo global e dos objetivos específicos que se pretendiam alcançar ao longo do projeto. O capítulo termina com a apresentação das opções metodológicas, onde se especifica a metodologia e estratégia de investigação.

1.1. Problema de investigação, enquadramento e pertinência

O Grupo Simoldes é um grupo empresarial que se dedica à produção de moldes de injeção para a indústria de plásticos, situando-se na cidade de Oliveira de Azeméis (*Simoldes Plastics*, sem data). Uma das suas empresas é a INPLAS que se distingue pela atividade de injeção de termoplásticos tendo como principal foco a indústria automóvel (*Inplas - Simoldes*, sem data).

Este projeto surge a partir do desafio lançado pela INPLAS ao centro de investigação *Center for Production and Logistics Optimization Technologies* (CPLOT) com o objetivo de melhorar o planeamento da produção das suas fábricas.

A INPLAS, sendo uma empresa produtora de peças de plástico, tem como objetivo responder à procura diária de peças tendo em conta restrições de capacidade, elegibilidade da máquina e disponibilidade do operador. Além das máquinas, existe ainda três tipos de recursos adicionais: moldes, operadores de máquinas e as equipas de *setup*. Cada peça tem um molde associado e o seu processamento é realizado numa única máquina. Peças que partilhem o mesmo molde, pertencem à mesma família e não necessitam de *setup* entre elas. Diariamente é necessário proceder à ordenação dos moldes para a produção em cada máquina, sendo que a montagem dos moldes nas máquinas ocorre de acordo com as suas compatibilidades. Aquando da mudança de molde é essencial recorrer a uma equipa de *setup*. Por motivos técnicos, a atribuição dos moldes às máquinas é previamente definida e, por isso, estamos perante um problema de máquinas paralelas dedicadas. Os operadores de máquinas são necessários durante o processamento das peças. As equipas de *setup* são compostas por operadores responsáveis pelas trocas de molde.

Esta realidade da empresa apresenta algumas condicionantes que não são consideradas nos problemas gerais de máquinas paralelas. Primeiramente, é necessário ter em conta o estado inicial dos equipamentos no início de cada dia (identificar que molde se encontra na máquina e se a máquina está em produção ou parada). Como as tarefas já estão alocadas às máquinas, considerou-se como indicador de desempenho a minimização da soma dos *makespan* de todas as máquinas.

Assim sendo, a realidade descrita consiste num problema de máquinas paralelas dedicadas com *setups* dependentes da sequência de famílias e recursos adicionais (PMSR). A versão mais geral deste problema tem uma complexidade muito elevada, uma vez que além da limitação de recursos, considera três níveis de decisão: atribuição das tarefas às máquinas (alocação), a ordenação das tarefas dentro de cada máquina (sequenciação) e a definição dos momentos em que cada tarefa será processada e o seu *setup* será realizado (tempo). No caso particular do PMSR com máquinas paralelas dedicadas, as tarefas já estão pré-alocadas às máquinas, não sendo necessário resolver o subproblema de alocação.

Face ao exposto, a situação estudada trata um problema real, complexo e muito difícil de resolver, podendo a sua resolução conferir uma vantagem competitiva para a empresa. Tendo em conta a complexidade da resolução do problema, os modelos matemáticos podem ter dificuldade em encontrar as soluções ótimas em termos computacionais adequados. Neste sentido, pode ser vantajosa a utilização de mecanismos que acelerem e facilitem o processo de procura da solução ótima, como as heurísticas matemáticas (que combinam a utilização em simultâneo de modelos matemáticos com métodos heurísticos).

Após este enquadramento ao problema e face à competitividade que se vive no mercado, é de extrema importância que as empresas possuam sistemas inteligentes que ajudem no seu processo de tomada de decisão, realçando assim a pertinência da solução proposta neste projeto. Esta solução partirá da adaptação de modelo gerais do problema, estudando-se o impacto das heurísticas matemáticas no desempenho dos algoritmos. Assim sendo, o sistema de apoio à decisão proposto pretende minimizar as ineficiências existentes ao nível do chão de fábrica, possibilitando a investigação do impacto desta solução no desempenho global das empresas.

1.2. Questão e objetivos de investigação

Tendo em conta o problema de investigação relatado definiram-se as seguintes questões de investigação: No caso particular de máquinas paralelas dedicadas o modelo baseado em *strip-packing* supera o modelo indexado ao tempo? As heurísticas matemáticas terão impacto na eficiência dos algoritmos utilizados?

Face ao problema apresentado e em resposta às questões de investigação, foi estabelecido um objetivo geral que consiste no desenvolvimento de um sistema de apoio à decisão para o planeamento de máquinas paralelas dedicas com *setups* dependentes da sequência de famílias e recursos adicionais. Além disso, também foram delineados os seguintes objetivos específicos:

- Adaptação do modelo *strip-packing* e do modelo indexado ao tempo ao caso particular de máquinas paralelas dedicadas com *setups* dependentes da sequência de famílias e recursos adicionais;
- Desenvolvimento de heurísticas para obter soluções iniciais;
- Implementação dos modelos matemáticos e da metaheurística em linguagem *Python*;
- Estudo do desempenho da heurística matemática.

1.3. Opções metodológicas

Em termos de caracterização das opções metodológicas, segue-se a abordagem sugerida por (Caixeta & Fabricio, 2018; Coutinho, 2011) que é composta por três níveis hierárquicos. Num primeiro nível surgem as técnicas como ferramentas para a aplicação prática, seguindo-se os métodos (conjunto de técnicas que especificam a abordagem particular) num patamar intermédio. Na base encontram-se as metodologias que consistem num sistema de métodos.

Relativamente aos tipos de metodologia, (Coutinho, 2011) indica que a maioria dos autores defende duas perspetivas: quantitativa e qualitativa. O projeto em questão enquadra-se na perspetiva quantitativa por se tratar de uma análise objetiva baseada em métodos científicos.

Efetivamente, é feito um estudo aos vários métodos de resolução do problema de programação de máquinas paralelas, procedendo-se a uma análise em termos de desempenho.

Em termos de métodos de investigação, (Coutinho, 2011) defende que um plano de investigação pode ser constituído por diversas estratégias, como investigação experimental, inquéritos, estudo de caso, investigação-ação, teoria fundamentada, etnografia e investigação histórica. Neste caso específico o método utilizado é a investigação-ação, uma vez que se pretende resolver um problema prático e real, partindo do enquadramento teórico do problema e com o objetivo de validar ou refutar a solução proposta através de um processo cíclico que alterna entre a ação e a análise crítica. Concretizando, a heurística matemática é implementada e testada computacionalmente, passando por um processo rigoroso de avaliação antes da sua validação final.

1.4. Estrutura do trabalho

Para além da Introdução, este trabalho contém os capítulos da Revisão Bibliográfica, dos Métodos e Aplicação, dos Resultados Computacionais e da Conclusão.

Na Revisão Bibliográfica efetua-se um enquadramento ao problema de máquinas paralelas com *setups* dependentes da sequência de famílias e recursos adicionais, relatando-se a evolução desta variante do problema, com foco na demonstração dos principais métodos de resolução.

Nos Métodos e Aplicação é feita a descrição do problema estudado de máquinas paralelas dedicadas. De seguida, são apresentados os dois modelos matemáticos implementados que foram adaptados à realidade do problema, destacando-se as principais contribuições deste estudo. No último subcapítulo são descritas as diferentes abordagens que compõem a heurística matemática.

Nos Resultados Computacionais demonstram-se os desempenhos das abordagens propostas. Concretamente, comparam-se os dois modelos matemáticos, analisa-se o desempenho do *Tabu Search* e avalia-se a eficiência das estratégias da heurística matemática desenvolvida.

Na Conclusão apresentam-se as principais conclusões do trabalho, bem como as perspetivas de trabalhos futuros.

Complementam estes capítulos, o Resumo, os Índices, a Lista de siglas, as Referências Bibliográficas e os Anexos.

2. REVISÃO BIBLIOGRÁFICA

Com o propósito de enquadrar o trabalho desenvolvido no projeto, neste capítulo apresenta-se uma revisão bibliográfica ao tema. Inicialmente é feita uma pequena contextualização ao problema de programação de máquinas paralelas, abordando os diferentes ambientes existentes. Posteriormente, nos subcapítulos 2.2 e 2.3 realça-se a importância de considerar os *setups* e a existência de recursos limitados no problema de programação. No subcapítulo 2.4 apresenta-se o problema de programação de máquinas paralelas com *setups* e recursos adicionais, relatando-se a evolução desta variante do problema, com foco na demonstração dos principais métodos de resolução. Neste sentido, uma abordagem para a resolução do problema consiste na modelação matemática de solução exata, que pode ter por base programação inteira, mas também *constraint programming* (CP). Além disso, também é possível recorrer a abordagens de solução aproximada, como os métodos heurísticos e metaheurísticos. No subcapítulo 2.5 é descrita a relevância das heurísticas matemáticas que consistem em abordagens híbridas para acelerar e facilitar o processo de resolução. No subcapítulo 2.6 são feitas avaliações de desempenho aos *solvers* e indicadores de desempenho utilizados nos estudos mais recentes. Por fim, no último subcapítulo apresentam-se os modelos matemáticos de referência deste estudo.

2.1. Programação de máquinas paralelas

Neste subcapítulo é feita uma introdução à programação de máquinas paralelas, apresentando-as principais características que distinguem os diferentes tipos de máquinas (idênticas, uniformes, não relacionadas e dedicadas). Além disso, são expostas algumas nuances relevantes associadas ao tema, tal como as restrições de elegibilidade da máquina.

Este trabalho aborda um caso de máquinas paralelas, sendo oportuno começar pela sua definição. Em termos globais, o problema de máquinas paralelas trata da atribuição de um conjunto de n tarefas a um conjunto de m máquinas, em que cada tarefa j deve ser processada em apenas uma máquina i e cada máquina i não processa mais que uma tarefa j em simultâneo. Com base nas características das máquinas estudadas, o sistema pode ser classificado de diferentes formas. Segundo Afzalirad & Rezaeian (2016); Cheng & Sin (1990); Fanjul-Peyro & Ruiz (2010); Potts (1985):

- Máquinas paralelas idênticas - uma tarefa pode ser processada por qualquer uma das máquinas livres não existindo alteração do tempo de processamento ($p_{ij} = p_j$);
- Máquinas paralelas uniformes - as máquinas apresentam velocidades diferentes (q_i) e, por isso, o tempo de processamento depende da velocidade da máquina alocada à tarefa: $p_{ij} = \frac{p_j}{q_i}$;
- Máquinas paralelas não relacionadas - representa o caso mais geral, em que uma máquina pode processar diferentes tarefas a diferentes velocidades, sendo que o tempo de processamento depende da combinação tarefa-máquina.

Segundo Kellerer & Strusevich (2003), o sistema pode ainda ser classificado como máquinas paralelas dedicadas. Nesta realidade do prolema, considera-se um conjunto de *n* tarefas ($N = \{1,2,\dots,n\}$) e *m* máquinas paralelas dedicadas ($M = \{1,2,\dots,m\}$). O conjunto de *N* tarefas é previemente dividido em *m* subconjuntos ($N_1, N_2, \dots N_m$) de modo a que as tarefas do conjunto N_i serão processadas apenas na máquina M_i (Glass et al., 2000; Kellerer & Strusevich, 2003). Neste caso considera-se a atribuição das tarefas às máquinas como fixa e o problema torna-se mais

particular. A flexibilidade do sistema diminui porque não é necessário resolver o subproblema de alocação das tarefas.

Em bastantes realidades de escalonamento que envolvem a produção de diferentes tipos de produtos, as máquinas estão associadas à produção de um determinado produto e, por vezes, as tarefas não podem ser processadas em nenhuma das máquinas disponíveis. Esta limitação é designada por restrições de elegibilidade da máquina, onde cada tarefa *j* só pode ser processada num subconjunto de máquinas m_j do total de *m* máquinas (Pinedo, 1995). Estas realidades estão presentes em sistemas de produção, como nas indústrias de plásticos, siderúrgica, semicondutores, mas também em serviços, como o caso da gestão de salas cirúrgicas hospitalares (Afzalirad & Rezaeian, 2016; Edis & Ozkarahan, 2010).

Neste sentido, é possível indicar que as máquinas dedicadas são um caso específico de restrições de elegibilidade em que cada tarefa pode ser processada em apenas uma máquina.

2.2. *Setups*

Neste subcapítulo demonstra-se a importância de considerar os *setups* na programação de máquinas paralelas, bem como as diferentes formas de o fazer. Posteriormente, apresentam-se alguns casos da literatura onde se explorou o problema tendo em conta a existência de *setups*.

As atividades de *setup* estão associadas à preparação das máquinas para o processamento de tarefas, envolvendo reconfigurações, ajustamentos ou limpeza de equipamentos, enquanto a produção é interrompida. Os *setups* são operações inevitáveis em muitas realidades industriais, tal como comprovado por Panwalkar et al. (1973) que apuraram que cerca de 75% das indústrias possuem tarefas que necessitam de atividades de *setup*. Neste sentido, é importante tentar minimizar o tempo gasto nesta operação e otimizar a sequência de processamento para que o sistema seja o mais eficiente possível. Quando a existência de *setups* é considerada nas decisões do escalonamento de realidades industriais registam-se reduções de custos e aumento da eficiência dos processos, o que salienta a importância de incorporar esta componente no estudo do problema (Allahverdi et al., 2008).

Como mencionado anteriormente, a existência de *setups* é muitas vezes inevitável, existindo diferentes formas de lidar com esta questão. Quando se considera que os *setups* não são dependentes da sequência significa que a duração depende apenas da tarefa a ser processada. Nestas situações, o tempo de *setup* é incluído no tempo de processamento da tarefa ou ignorado (Allahverdi et al., 1999). Neste caso, a ordenação das tarefas não é tão relevante, tratando-se apenas de um problema de alocação das tarefas às máquinas.

Contudo, um cenário mais próximo da realidade é quando se assume que o tempo de *setup* depende da sequência das tarefas e da máquina. Nesta situação, o tempo de *setup* é considerado de forma separada do tempo de processamento e é influenciado pela tarefa processada anteriormente. Esta questão é retratada no problema de máquinas paralelas com tempos de *setup* dependentes da sequência (PMS). A dificuldade de resolução do problema aumenta porque passam a existir dois níveis de decisão: alocação e sequenciação.

2.2.1. Programação de máquinas paralelas com *setups*

No que diz respeito ao PMS com máquinas não relacionadas serão apresentados alguns casos de aplicação desta variante do problema. Vallada & Ruiz (2011) aplicaram várias versões do algoritmo genético (GA) alcançando um RPD médio de 12,07% para a versão com melhor comportamento. As soluções do GA são razoáveis para instâncias com 250 tarefas e 30 máquinas. Além disso, o algoritmo proposto foi comparado com uma versão adaptada do modelo MIP de Guinet (1993), o que valida o seu desempenho porque as soluções obtidas pelo GA são próximas das soluções ótimas do modelo.

Mais tarde, Avalos-Rosales et al. (2015) apresentaram um MILP mais eficiente capaz de resolver instâncias até 60 tarefas e 5 máquinas. Os autores também implementaram duas metaheurísticas (*multi-start algorithm* e *variable neighbourhood descent*), obtendo soluções para instâncias com até 250 tarefas e 30 máquinas, com tempos de computacionais competitivos. Tran et al. (2016) implementaram duas abordagens de decomposição exata (*Logic-Based Benders Decomposition* e *Branch and Check*), partindo de uma base híbrida que conjuga um problema mestre de programação inteira mista e o *Travelling Salesman Problem* (TSP). Estas abordagens demonstraram-se eficientes na resolução de instâncias com até 120 tarefas e 8 máquinas, com tempos computacionais quatro ordens de grandeza inferiores aos modelos matemáticos anteriores.

Mais recentemente, Fanjul-Peyro et al. (2019) adaptaram e introduziram melhorarias nos dois modelos mencionados anteriormente, propondo um algoritmo exato para a resolução deste problema. Primeiramente, comprovou-se que o *solver* utilizado pode ter influência na eficiência do modelo, uma vez que para o mesmo modelo de Avalos-Rosales et al. (2015), este estudo consegue resolver instâncias de maiores dimensões (200 tarefas e 4 máquinas). Além disso, estes autores obtiveram melhores resultados do que os estudos que serviram de base ao seu trabalho, uma vez que o modelo proposto proporciona soluções ótimas para instâncias com dimensões consideráveis (até 400 tarefas). Ainda de mencionar que o método apresenta soluções próximas do ótimo (desvio relativo do LB inferior a 0,8%) para exemplos com 1000 tarefas e 8 máquinas.

Para terminar, numa perspetiva *constraint programming*, Gedik et al. (2018) desenvolveram um novo modelo CP. Para instâncias com até 120 tarefas e 10 máquinas, obtiveram um desvio médio em relação às melhores soluções heurísticas inferior a 3%.

2.3. Recursos

Neste subcapítulo enfatiza-se a relevância de considerar a existência de recursos limitados nos problemas de escalonamento, sendo expostos os principais pontos que caracterizam os recursos alocados aos sistemas. Posteriormente, apresentam-se as variantes do problema que consideram a existência de recursos adicionais.

Na maioria dos estudos sobre programação de máquinas paralelas, o único recurso considerado é a máquina, no entanto, em muitos ambientes industriais também se consideram outros recursos adicionais como veículos automáticos, operadores, ferramentas, paletes e robots industriais, tornando o estudo do problema de máquinas paralelas com recursos adicionais uma área de interesse científico (J. Blazewicz et al., 1983; Słowiński, 1980; Ventura & Kim, 2000).

Como a realidade deste problema tem associada a limitação de recursos, o estudo das suas características torna-se relevante. Segundo Błażewicz et al. (2004), se o recurso estiver associado a uma máquina será designado como um recurso adicional de processamento, enquanto caso seja necessário antes ou após uma fase de processamento, a sua designação será recurso *input-output*. Os recursos adicionais são ainda classificados em relação à sua renovabilidade e divisibilidade. Na perspetiva de renovabilidade (Jacek Blazewicz et al., 2019; Słowiński, 1980):

- Um recurso é considerado renovável, se o seu uso em cada momento for limitado (pode ser usado em tarefas consecutivas);
- Um recurso é não renovável se o seu consumo total for limitado (sendo utilizado numa tarefa, fica indisponível para a seguinte);
- Um recurso é considerado duplamente limitado caso seja simultaneamente renovável e não renovável.

Na perspetiva de divisibilidade (Jacek Blazewicz et al., 2019):

- Os recursos discretos podem ser alocados a tarefas presentes em unidades discretas de um determinado conjunto finito de alocações possíveis;
- Os recursos contínuos podem ser alocados a tarefas em quantidades arbitrárias dentro de um determinado intervalo.

Na eventualidade dos recursos não serem fixos ao longo do tempo, são definidos como dinâmicos.

2.3.1. Programação de máquinas paralelas com recursos

Quando se considera a limitação de recursos na realidade de máquinas paralelas, surge outra vertente do problema, a programação de máquinas paralelas com recursos adicionais (PMR). A dificuldade de resolução do problema aumenta porque passam a existir três níveis de decisão: alocação, sequenciação e tempo. O tempo diz respeito aos momentos em que as operações vão ocorrer, sendo influenciado pela disponibilidade dos recursos.

Os artigos mais antigos debruçam-se sobre máquinas paralelas idênticas, como é o caso de Błażewicz et al. (1987) que estudou o problema com o objetivo de minimizar o tempo médio de fluxo. Relativamente ao PMR com máquinas não relacionadas, o estudo mais específico desta variante é mais recente. Edis & Oguz (2011) desenvolveram uma versão simplificada do problema considerando os operadores das máquinas como recurso adicional. Os autores seguem uma abordagem de *constraint programming* (CP) baseada num *Lagrangian Relaxation Problem* que apresentou um *gap* médio de 2,28% para instâncias com 30 tarefas e 5 máquinas. Edis & Ozkarahan (2012) estudaram o problema considerando restrições de recursos e de elegibilidade de máquina para um caso real de um sistema de produção de injeção de moldes. Comparando duas abordagens: uma tendo por base CP e outra que combina programação inteira (IP) e CP, fornecem soluções relativamente eficientes quando o número de recursos é escasso. Para instâncias com 100 tarefas e 36 máquinas, as abordagens de CP e CP+IP obtêm um *gap* médio de 4,31% e 1,49%, respetivamente, destacando-se a abordagem híbrida.

Edis et al. (2013) apresentaram uma revisão abrangente dos estudos sobre o problema, estabelecendo um mecanismo de classificação de cinco categorias principais: ambiente de máquina, recurso adicional, função objetivo, complexidade e métodos de solução.

Para resolver uma versão dinâmica e mais complexa do PMR com máquinas não relacionadas, Fanjul-Peyro et al. (2017) apresentaram três heurísticas matemáticas, *Machine-Assignment Fixing* (MAF), *Job-Machine Reduction* (JMR) e *Greedy-Based Fixing* (GBF). Estas estratégias melhoraram o desempenho de dois modelos matemáticos MILP (um baseado na ideologia *strip-packing* e o outro com indexação ao tempo). As abordagens híbridas utilizadas têm foco na fase de atribuição das tarefas às máquinas. Na estratégia MAF, é resolvido o problema de alocação em que as tarefas são atribuídas às máquinas sem especificar o momento em que serão processadas. De seguida, assumindo esta atribuição como fixa, resolve-se o problema de alocação dos recursos. A heurística matemática JMR reduz o conjunto de possíveis alocações tarefa-máquina, descartando para cada tarefa as máquinas que são responsáveis por maiores tempos de processamento. Por fim, a GBF é uma estratégia gulosa, na qual o problema é resolvido sequencialmente para pequenos subconjuntos de tarefas, mantendo fixas as alocações encontradas nas iterações anteriores. O processo termina quando todas as tarefas estiverem atribuídas às máquinas e com horário de início agendado. Após os resultados obtidos, apurou-se que nas instâncias de menor dimensão (até 12 tarefas), as heurísticas matemáticas MAF e GBF associadas ao modelo de *strip-packing* são responsáveis pelos melhores desempenhos. Nas instâncias de dimensão média (até 30 tarefas), as heurísticas matemáticas superam o desempenho dos modelos matemáticos. Ainda de mencionar que o modelo baseado em *strip-packing* apresenta melhor desempenho comparativamente ao modelo indexado ao tempo, o que se intensifica com o aumento da dimensão das instâncias e com a utilização das heurísticas matemáticas. Em termos de reflexão, Fanjul-Peyro et al. (2017) sugerem o desenvolvimento de novas heurísticas matemáticas, uma vez que as estudadas nem sempre apresentam boas soluções nas instâncias de maiores dimensões (50 tarefas e 15 máquinas).

Villa et al. (2018) recorreram a duas heurísticas para resolver o problema. A primeira considera a restrição de recursos durante o processo e a segunda baseia-se na atribuição das tarefas às máquinas sem considerar a limitação de recursos. Esta última possui um mecanismo para reparar possíveis soluções inviáveis. Concretizando, a primeira estratégia é composta por três fases: ordenação das tarefas, construção da solução e melhoria da solução. Na fase de construção utiliza-se à heurística NEH e na fase de melhoria da solução recorre-se ao *Local Search*. No que diz respeito à segunda estratégia, são testadas heurísticas *Multi-Pass* compostas por duas etapas: construção da solução (através de regras de alocação) e aplicação de diferentes Local Search. Em termos de resultados, a segunda abordagem obteve melhor desempenho, destacando-se esta diferença nas instâncias de maior dimensão (350 tarefas e 30 máquinas). Em Fleszar & Hindi (2018) foi implementado um novo modelo MILP que obteve um *gap* médio de 0,13% para as instâncias de igual dimensão do estudo de Fanjul-Peyro et al. (2017). Assim, este modelo é capaz de resolver instâncias até 1000 tarefas, mas apenas com 2 máquinas. Para dar resposta a problemas com mais de 2 máquinas, foi proposta uma heurística de duas etapas. Na fase inicial, um modelo MILP é usado para calcular o *lower bound* e alocar as tarefas às máquinas e, em seguida, recorre-se a um modelo CP para sequenciar as tarefas nas respetivas máquinas. Esta estratégia obteve um *gap* médio para todas as instâncias de 1,77%. No caso específico das instâncias com 500 e 1000 tarefas, os *gaps* médios obtidos foram de 2,04% e 1,98%, respetivamente. Segundo os autores, esta estratégia supera todos os métodos encontrados na literatura para o mesmo problema.

2.3.2. Programação de máquinas paralelas com recursos flexíveis

Na área das máquinas paralelas com recursos adicionais, Daniels et al. (1996) introduziram uma nova classe de problemas, a programação de máquinas paralelas com recursos flexíveis (PMFRS). Nesta realidade, a atribuição das tarefas às máquinas é estabelecida à priori. Tendo em conta que cada máquina tem as suas tarefas alocadas previamente, trata-se de um problema de máquinas paralelas dedicadas. Os tempos de processamento não assumem um valor fixo, dependendo da quantidade e do tipo de recursos alocados à tarefa. Nesta realidade, a incorporação de recursos flexíveis assume que existe um recurso limitado que é partilhado pelas várias máquinas/tarefas. A mão de obra é um exemplo de recursos flexíveis que são capazes de serem alocados a vários estágios do sistema. Esta capacidade dos recursos cria uma alavancagem poderosa para melhorar o desempenho operacional (Daniels et al., 1996).

Segundo Daniels et al. (1996), este problema pode ser encarado de duas perspetivas diferentes no que diz respeito à alocação dos recursos flexíveis:

- Estática: a decisão da alocação dos recursos permanece fixa ao longo do horizonte temporal, não sendo possível os recursos alternarem entre as máquinas durante o processo. Este problema pode ser resolvido em tempo polinomial;
- Dinâmica: representam ambientes mais gerais, onde a decisão da alocação dos recursos pode variar ao longo do horizonte temporal e, por isso, os recursos podem alternar entre as máquinas durante o processo. O problema torna-se NP-difícil.

Caso se considere que a atribuição das tarefas às máquinas não é previamente especificada, sendo necessário resolver o subproblema de alocação, segundo Daniels et al. (1999) surge uma vertente mais geral do problema com a designação de programação de máquinas paralelas não específicas com recursos flexíveis (UPMFRS).

As principais diferenças do PMFRS em relação ao PMR com máquinas não relacionadas são os tempos de processamento dependerem da quantidade de recursos alocados e as máquinas serem dedicadas. Segundo (Edis et al., 2013), o PMR com máquinas não relacionadas é um problema mais geral por incorporar o subproblema de alocação das tarefas às máquinas.

Em termos de estudo científico, o problema começou por ser abordado por Daniels et al. (1996) que desenvolveram versões estática e dinâmica do problema para instâncias com até 80 tarefas, 4 máquinas e 6 recursos. Neste estudo demonstrou-se que a perspetiva dinâmica produz melhorias no desempenho do sistema. Os autores também comprovaram que a flexibilidade dos recursos é benéfica para problemas de grandes dimensões, com recursos escassos e tempos de processamento dependentes dos recursos alocados. Como continuação do estudo anterior, Daniels et al. (1999) estudaram o UPMFRS e implementaram um modelo para a versão estática do problema. Tendo em conta a complexidade enfrentada, os autores compararam o desempenho entre dois métodos heurísticos: *Tabu Search e Decomposition*. Os métodos heurísticos mostraram-se capazes de resolver instâncias com até 50 tarefas, 5 máquinas e com capacidade máxima de 7 recursos. A heurística *Tabu Search* revelou melhor aptidão para encontrar soluções mais próximas do ótimo. Por outro lado, Grigoriev et al. (2007) abordaram uma versão dinâmica do problema através de um modelo de programação inteira.

Su & Lien (2009) desenvolveram duas estratégias heurísticas para o UPMRFS estático. Na primeira estratégia, as tarefas são atribuídas às máquinas, seguindo-se a alocação dos recursos. Na segunda

estratégia a ordem inverte-se. As instâncias utilizadas possuem até 10 máquinas, 52 tarefas e com recursos a variar entre 10 e 20 unidades. As duas estratégias foram avaliadas tendo em conta a qualidade das soluções, comparando com limite inferior alcançado em Lee & Massey (1988). Ambas as abordagens apresentam valores médios de qualidade de solução superior a 97%, destacando-se a segunda abordagem com melhor desempenho em 63% dos testes.

Edis & Oguz (2012) abordaram o problema dinâmico PMFRS, comparando uma versão melhorada do modelo de Daniels et al. (1996) com uma abordagem híbrida IP-CP. Esta abordagem híbrida resulta da combinação do modelo IP para a alocação dos recursos, com um modelo CP para a ordenação das tarefas. Num conjunto de instâncias com até 20 tarefas, 5 máquinas e 7 recursos, o modelo PMFRS apresentou um *gap* médio de 0,41% e soluções ótimas para 78% dos testes. A abordagem IP-CP alcançou um *gap* médio de 0,82%, tempos computacionais inferiores a 10 segundos e soluções ótimas em 66% dos testes. Posteriormente, a estratégia híbrida é aplicada a instâncias com 100 tarefas, o que desencadeou um tempo computacional inferior a 60 segundos, soluções ótimas em 50% das instâncias e *gaps* inferiores a 0,67%. Numa segunda fase, os autores estudaram a capacidade de resposta da abordagem IP-CP para o problema UPMFRS. Nas instâncias maiores (100 tarefas) obteve-se um tempo computacional médio de 75 segundos, soluções ótimas em 20% dos testes e *gaps* inferiores a 1,43%.

Mais recentemente, Fanjul-Peyro (2020) considerou a existência de recursos flexíveis que são partilhados entre as operações de processamento e de *setup*, no entanto, neste caso não se pressupõem que os tempos de processamento dependem da quantidade de recursos alocados. Este estudo será explorado no subcapítulo 2.4.

2.4. Programação de máquinas paralelas com *setups* e recursos adicionais

Neste capítulo apresenta-se o problema mais próximo da realidade estudada, o PMSR. Posteriormente, é feito um levantamento à evolução do estudo deste problema, com principal foco no desenvolvimento atual da ciência.

No sentido de se aproximar da realidade dos problemas de máquinas paralelas, é importante considerar em simultâneo as operações de *setup* e a existência de recursos limitados. Esta realidade torna o problema mais geral e pode ser definida como programação de máquinas paralelas com *setups* e recursos adicionais (PMSR).

Este problema pode ser definido como um conjunto de n tarefas a serem processadas num conjunto de *m* máquinas paralelas com um tempo de processamento da tarefa j na máquina i (p_{ij}). Os tempos de processamento e os recursos necessários são fornecidos para cada tarefa e o problema permite a consideração de diferentes tipos de recursos adicionais. As tarefas durante o seu processamento podem necessitar de um conjunto de tipos de recursos $R = \{R_1, R_2, \dots, R_u\}$ que estão disponíveis nas quantidades de $b_1, b_2, \dots, b_u$unidades, respetivamente (Edis et al., 2013).

Esta realidade do PMSR é complexa, uma vez que tratando-se de um problema de escalonamento com recursos limitados, envolve três níveis de decisão: a atribuição das tarefas às máquinas (alocação), a ordenação da execução das tarefas em cada máquina (sequenciação) e a definição dos momentos de início e término das operações das tarefas e da alocação de recursos (tempo) (Reklaitis, 1996). De facto, além de se definir a máquina onde uma tarefa vai ser processada e a ordem nessa mesma máquina, é fundamental determinar o momento temporal em que se reúnem

condições para o processamento de determinada tarefa. Além disso, é necessário avaliar o agrupamento que se faz em cada máquina tendo em conta o número de recursos disponíveis em cada momento (Pinto & Grossmann, 1997). O PMSR com máquinas paralelas dedicadas corresponde a uma realidade em que o conjunto de tarefas que serão processadas em cada máquina é pré-determinado. Esta suposição simplifica o problema eliminando o subproblema de alocação das tarefas às máquinas.

Casos práticos

Recentemente, alguns trabalhos tratam o caso mais geral do problema de programação de máquinas paralelas com *setups* e recursos. Afzalirad & Rezaeian (2016) inspirados no problema de programação de montagem de blocos num estaleiro, abordaram o PM com máquinas não relacionadas considerando restrições de recursos, tempos de *setup* dependentes da sequência, datas de entrega, elegibilidade da máquina e restrições de precedência. O objetivo era minimizar o *makespan*. Os resultados indicam que, no caso de problemas de pequena escala, ambos os métodos são eficientes e eficazes, enquanto em problemas de grande escala, a abordagem AIS supera estatisticamente a abordagem GA.

Ozer & Sarac (2019) estudaram um problema de máquinas paralelas idênticas com recursos de processamento, tempos de *setup* dependentes da sequência e restrições de elegibilidade da máquina. Com o objetivo de minimizar o tempo total ponderado de conclusão, foi implementada uma heurística matemática composta por um modelo MIP e a metaheurística GA. Neste caso, ao contrário da versão original do GA, o cromossoma contém apenas os valores das variáveis de decisão referentes aos subproblemas de alocação e sequenciação. Neste sentido, o GA procede à alocação das tarefas às máquinas e à ordenação das tarefas em cada máquina. Os valores destas variáveis são passados como parâmetros ao modelo matemático que determina os tempos de início, de espera e de conclusão das tarefas. Por fim, o *fitness* do cromossoma assume o valor da função objetivo do modelo. Para as instâncias de pequena dimensão (8 tarefas, 3 máquinas e 5 recursos), o modelo matemático chegou à solução ótima em quase todos os testes. Nas instâncias de dimensões média (40 tarefas, 6 máquinas e 20 recursos) e grande (100 tarefas, 8 máquinas e 50 recursos) as melhores soluções foram obtidas utilizando a heurística matemática.

Bektur & Saraç (2019) estudaram um problema de máquinas paralelas não relacionadas com tempos de *setup* dependentes da sequência e restrições de elegibilidade da máquina. Considerou-se o caso particular de uma equipa como recurso de *setup*, o que implica que em cada momento só poderá existir um *setup* em execução. Tendo como objetivo a minimização do atraso ponderado total, foi proposto um modelo matemático MILP e as metaheurísticas *Tabu Search* e *Simulated Annealing*. Com um tempo limite de 18000 segundos, o modelo apenas alcançou soluções ótimas nas instâncias de 2 máquinas e 10 tarefas. Relativamente às metaheurísticas, o TS apresentou melhores resultados nas instâncias com até 10 máquinas e 100 tarefas. Os autores ainda recorreram a uma regra ATCS que minimiza o atraso total ponderado. Esta heurística é utilizada para fornecer soluções iniciais às metaheurísticas. A utilização desta regra permite obter melhores resultados, o que se comprova pela redução do desvio percentual, no caso do TS de 13,6% para 0,28%, enquanto no SA de 9,20% para 5,58%. Face ao exposto, a estratégia que desencadeou um melhor comportamento foi o TS com a regra ATCS.

Pinheiro et al. (2020) analisaram o problema de máquinas paralelas não relacionadas com famílias de *setup* e restrição de recursos, com o objetivo de minimizar o atraso total. Neste caso, as tarefas

são agrupadas em famílias e a operação de *setup* só ocorre entre tarefas que pertençam a diferentes famílias. Numa fase inicial foi desenvolvido um MILP que apresentou bom desempenho para instâncias com até 20 tarefas e 6 máquinas, com um tempo limite computacional de 3600 segundos. Para dar resposta a problemas de maior dimensão, os autores desenvolveram duas metaheurísticas, *Simulated Annealing* (SA) e *Iterated Greedy* (IG). Estes algoritmos partem de uma solução inicial fornecida por uma heurística construtiva que sequencia as tarefas em cada máquina tendo em conta a proximidade da data de entrega. Além disso, o momento de processamento das tarefas é definido de forma a garantir que a restrição de recursos é respeitada. Para instâncias com até 50 tarefas, os algoritmos IG e SA apresentam desvios relativos médios de 8,84% e 39,87%, respetivamente. Assim, para esta realidade do problema o algoritmo IG superou o SA.

Fanjul-Peyro (2020) abordou o problema PMSR com máquinas não relacionadas, sendo o objetivo a minimização do *makespan*. Neste caso, assumiu-se que os tempos de processamento e de *setups* dependem da máquina e das tarefas, sendo os recursos necessários de três tipos: específicos para processamento, específicos para *setups* e compartilhados (podem ser usados quer no processamento quer durante os *setups*). Neste sentido, foram considerados seis fatores: tempos de processamento, tempos de *setup*, recursos de processamento, recursos de *setup*, recursos partilhados usados no processamento e recursos partilhados usados nos *setups*. Para dar resposta ao problema, os autores apresentaram dois modelos matemáticos MILP (um baseado em *strip-packing* e o outro indexado ao tempo). Para melhorar a performance dos modelos, é proposto um algoritmo exato com três fases (TPhA): alocação, sequenciação e tempo. Os modelos matemáticos conseguem resolver instâncias com até 50 tarefas, 8 máquinas e 40 recursos com um *gap* médio de 8,94%, no entanto para problemas de maiores dimensões apresentam dificuldades. O TPhA demonstrou um comportamento similar ao resolver instâncias com 50 e 140 tarefas, com um *gap* médio de 6,94% em ambas as situações. Além disso, este algoritmo apresentou na maioria dos casos um *gap* inferior a 2,7% para instâncias de 400 tarefas e o mesmo número de máquinas. De referir ainda que também se comprovou que o modelo mais geral permite resolver um caso mais particular (limitação de recursos sem considerar a existência de *setups*) de uma forma mais eficiente.

Yepes-Borrero et al. (2020) recorrem ao algoritmo GRASP para o PMSR com máquinas não relacionadas com o objetivo de minimizar o *makespan*. Os autores seguiram 2 abordagens para a fase construtiva: a primeira sem considerar os recursos adicionais de *setup* e a segunda integrando a restrição dos trabalhadores. De seguida, foi sugerido um mecanismo de reparação para as situações em que a restrição dos recursos de *setup* não seja respeitada, adiando-se o *setup* que inicia mais tarde. Os testes foram realizados em instâncias com até 250 tarefas e 30 máquinas. A estratégia que considera os recursos como limitados na fase construtiva permitiu obter melhores resultados, com um desvio médio da melhor solução encontrada de 0,19%.

O estudo de Yepes-Borrero et al. (2021) consiste numa extensão dos trabalhos de Yepes-Borrero et al. (2020) e Fanjul-Peyro (2020) focando-se numa estratégia multiobjetivo com a minimização do *makespan* e do número de recursos de *setup*. Para tal, os autores propuseram o algoritmo *Truncated Restarted Iterated Pareto Greedy* (T-RIPG), composto por seis fases. Na fase de inicial, recorre-se ao GRASP do Yepes-Borrero et al. (2020) para gerar um conjunto de soluções iniciais. De seguida, na fase *greedy* ocorre a reconstrução de uma solução para gerar novas. Caso seja necessário, existe uma etapa de reparação em que se adia o início dos *setups* para reduzir o consumo de recursos. Na etapa de seleção, do conjunto de soluções iniciais determina-se que

solução será processada na fase seguinte do algoritmo. Posteriormente, a solução selecionada sofre uma fase de *local search*. Na última fase, o algoritmo reinicia e volta à fase inicial para gerar um novo conjunto de soluções iniciais. No estudo computacional apurou-se que o algoritmo desenvolvido fornece melhores soluções comparativamente às restantes metaheurísticas em instâncias com até 250 tarefas e 30 máquinas.

Em Zhang et al. (2021) considerou-se a limitação dos recursos de *setup* e a capacidade de aprendizagem dos trabalhadores para o PMSR. Efetivamente, foi desenvolvido um CEA (*Combinatorial Evolutionary Algorithm*) composto por 3 métodos: heurística LS, regra SST e a regra ECT, com uma abordagem multiobjetivo para minimizar o *makespan* e o consumo de energia. O algoritmo proposto apresentou melhor desempenho que os algoritmos comparados para instâncias com dimensão de 400 tarefas, 22 máquinas e 10 recursos.

Lopez-Esteve et al. (2022) também implementaram uma metaheurística GRASP partindo de heurísticas construtivas que sofrem um processo de randomização e uma fase de *Local Search* (LS). As estratégias heurísticas utilizadas são compostas por duas etapas, uma construtiva e outra de reparação. Os recursos adicionais considerados foram de processamento e de *setup*. Na fase de testes, utilizaram-se as mesmas instâncias do estudo de Fanjul-Peyro (2020), demonstrando-se que esta estratégia encontra soluções com um *gap* de 1,80% em relação às soluções do algoritmo de Fanjul-Peyro (2020) em apenas 120 segundos. Para as instâncias de dimensão média (140 tarefas) obteve-se um RDP médio de 2,26%. Por fim, realça-se que as fases de randomização e LS ajudam a melhor a qualidade das soluções, permitindo a libertação de ótimos locais.

Numa perspetiva diferente, como o *constraint programming* revela bastante eficiência em problemas de escalonamento, Yunusoglu & Topaloglu Yildiz (2022) propuseram um modelo exato baseado em CP para o problema PMSR com máquinas não relacionadas considerando restrições de elegibilidade da máquina e datas de entrega. Em termos de resultados, nos problemas de maior dimensão (60 tarefas e 8 máquinas), esta abordagem supera as melhores soluções de trabalhos anteriores com um *gap* médio de 15,52%.

2.5. Heurísticas matemáticas

Tendo em conta a complexidade do problema, nesta secção é introduzida a relevância da utilização das heurísticas matemáticas, expondo-se as diversas formas de aplicar esta estratégia. Na fase final deste capítulo expõe-se com maior detalhe uma das abordagens das heurísticas matemáticas, o *warm-start*.

O problema de programação de máquinas paralelas com *setups* dependentes da sequência por si só já é um problema complexo por necessitar de dois tipos de decisão: alocação e sequenciação. Quando se considera a existência de recursos limitados, a complexidade do problema aumenta porque além da restrição de recursos, passa a existir mais um nível de decisão, o tempo. Tendo em conta esta complexidade muita elevada do problema, é fundamental encontrar estratégias que permitam atingir as soluções ótimas num tempo computacional razoável. As heurísticas matemáticas são uma estratégia possível onde se conjugam em simultâneo modelos matemáticos com métodos heurísticos, usufruindo dos benefícios de ambos os métodos (Boschetti et al., 2009).

Esta abordagem é muitas vezes utilizada para dar resposta a problemas de maiores dimensões, com o objetivo de simplificar o modelo matemático e reduzir o tempo computacional necessário para

resolver o problema, através da diminuição do espaço de pesquisa (Moussavi et al., 2019). As heurísticas matemáticas podem decompor o problema original em subproblemas de menor dimensão e complexidade, com o objetivo de considerar menos variáveis e parâmetros em simultâneo.

Puchinger & Raidl (2005) classificaram as combinações de algoritmos exatos e metaheurísticos em duas categorias principais:

- Combinações colaborativas - os métodos exatos e heurísticos podem ser executados sequencialmente, entrelaçados ou em paralelo. Apesar dos algoritmos trocarem informações, não fazem parte um do outro.
- Combinações integrativas - nesta técnica, um dos algoritmos está integrado no outro, existindo duas possibilidades: o método heurístico integrado no método exato ou o método exato integrado no método heurístico.

Nas combinações colaborativas sequenciais, tal como em Applegate et al. (1998), um método heurístico fornece uma solução inicial válida ao método exato que, posteriormente, resolve o problema encontrando uma solução ótima. Também é possível a sequência contrária em que o modelo matemático reduz o espaço de pesquisa para a posterior atuação do método heurístico, tal como em Vasquez & Hao (2001). As combinações colaborativas também podem ocorrer em paralelo ou entrelaçadas, contudo estas são menos frequentes (Puchinger & Raidl, 2005).

Nas combinações integrativas, uma possível estratégia, tal como em Della Croce et al. (2014), consiste na incorporação de um método exato numa abordagem heurística. Neste sentido, o método heurístico determina variáveis de decisão parciais para cada iteração. De seguida, aplicando o modelo matemático aos subproblemas com variáveis de decisão parciais pré-determinadas, determinam-se as restantes variáveis de decisão (Woo & Kim, 2018). Na perspetiva contrária, os métodos heurísticos também podem ser integrados em modelos matemáticos efetuando, por exemplo, operações de *bounding*, *cutting* ou *pricing*. Na primeira situação (*bounding*), as metaheurísticas completam uma solução parcial, fornecendo um limite superior ao modelo matemático *Branch and Bound*. No caso do *cutting*, as metaheurísticas podem ser usadas no procedimento de corte da solução no algoritmo *Branch and Cut*. No algoritmo *Branch and Price*, o *princing* das colunas pode ser realizada por uma metaheurística (Puchinger & Raidl, 2005).

Em termos práticos, as heurísticas matemáticas começaram por ser utilizadas no problema de roteamento de veículos, no entanto, atualmente possuem uma aplicação ampla (Karlsson et al., 2021). Ao longo dos subcapítulos anteriores já foram apresentados alguns casos práticos onde os autores recorreram a heurísticas matemáticas para alcançar melhores resultados. No subcapítulo seguinte explora-se com mais detalhe uma das possíveis abordagens, o *warm-start*.

2.5.1. Mecanismo de solução Inicial – *warm-start*

Em situações em que os tempos computacionais sejam bastante elevados ultrapassando o limiar do razoável, é benéfico tentar acelerar o processo de obtenção da solução ótima. Uma forma de o fazer é encurtar o espaço de soluções viáveis que se pretende analisar através de mecanismos de aquecimento (*warm-start*), possibilitando a redução do tempo computacional. Em termos concretos, com recurso a algoritmos eficientes, este mecanismo fornece ao modelo uma solução

inicial viável e de qualidade razoável, a partir da qual a procura da solução ótima se vai iniciar (Abreu & Fuchigami, 2022; Heinz et al., 2022).

Esta estratégia apresenta um potencial enorme, tal como demonstrado em Crainic et al. (2021), onde se apurou que ao ao utilizar métodos heurísticos para fornecer uma solução inicial ao modelo matemático, o desempenho computacional do modelo melhou em cerca de 60% dos testes. Ainda de referir que esta estratégia é diversificada, uma vez que a solução incial a fornecer ao modelo matemático pode ser gerada por um método heurístico, um MILP ou até mesmo um modelo CP (Vahedi-Nouri et al., 2022).

Tendo em conta as vantagens descritas anteriormente, o mecanismo de *warm-start* é uma ferramenta atual com enorme potencial sendo possível de ser aplicado em diferentes áreas e através de diversas formas como será exposto de seguida.

Casos práticos

Heinz et al. (2022), no seu estudo do problema de máquinas paralelas idênticas com recursos de *setup* limitados, recorreram a um *warm-start* em que uma solução inicial é fornecida ao modelo CP. Esta solução inicial consiste na atribuição das tarefas às máquinas e na ordenação das tarefas em cada uma das máquinas (subproblemas de alocação e sequenciação). Assim, foram desenvolvidos dois métodos heurísticos LOSOS e ROSOL em que a principal diferença consiste no momento que lidam com a alocação dos recursos. Na heurística LOSOS, na construção da sequência em cada máquina, a seleção da próxima tarefa tem em conta uma medida de avaliação e a disponibilidade do recurso. Enquanto ROSOL efetua a sequenciação sem avaliar a disponibilidade do recurso, aplicando posteriormente um mecanismo de reparação caso as soluções sejam inválidas. Ambos os algoritmos utilizam 3 funções específicas: gerar tarefas iniciais, selecionar próxima tarefa e otimizar o escalonamento final de cada máquina.

O desempenho das heurísticas não foi muito diferente, sendo ROSOL ligeiramente melhor, o que se intensifica quando o número de recursos é mais escasso. Através da utilização do mecanismo *warm-start* produziram-se resultados bastante melhores comparativamente à utilização do modelo matemático sem *warm-start*. De facto, obtiveram-se soluções viáveis para instâncias com dezenas de máquinas e centenas de tarefas, com um desvio relativo médio de 5% em relação ao LB e com tempos computacionais na ordem dos minutos. Também se demonstrou que o modelo era bem dimensionado e eficaz quando uma solução inicial razoável lhe era fornecida. Ainda de referir que os autores efetuaram uma comparação com algumas abordagens anteriores da literatura, onde os resultados mostraram que a metodologia *warm-start* fornece melhores soluções que a heurística híbrida de Kim & Lee (2012) e o GA de Huang et al. (2010), intensificando-se esta diferença com o aumento do tamanho das instâncias.

Abreu & Fuchigami (2022), para o problema de escalonamento *flowshop*, recorreram a programação inteira para implementar vários modelos matemáticos, selecionando o que apresentou melhor comportamento. De seguida, foram aplicados três métodos heurísticos (SPT, LPT e uma adaptação da heurística NEH) que fornececem uma solução inicial ao modelo selecionado. As abordagens referidas foram aplicadas a 240 instâncias com diferentes dimensões até 500 tarefas e 20 máquinas. Na abordagem sem *warm-start*, para 2% dos testes não foi obtida uma solução viável. Em contrapartida, os resultados do método com *warm-start* foram mais robustos, obtendo soluções viáveis em todas as instâncias. Numa análise comparativa entre os três

métodos de solução inicial, o LPT funcionou melhor para encontrar soluções ótimas (43% das instâncias), seguindo-se pelo SPT e depois pela heurística NEH. Por outro lado, com a utilização da heurística NEH o modelo apresentou um maior número de soluções viáveis, seguindo-se SPT e LPT. Em termos de tempos computacionais e *gap* de otimização, os métodos foram avalidados tendo em conta a sua eficiência (*P(f)* - percentagem das instâncias em que o método encontra o melhor valor) e robustez (*f* - rapidez com que o procedimento atinge 100% dos melhores valores). O LPT-*warm-start* foi o mais robusto e, por isso, o mais rápido a atingir 100% dos melhores valores para tempos computacionais (*f* = 23.96) e *gap* de otimização (*f* = 1.44). O SPT-*warm-start* foi o mais eficiente em termos de *gap* (*P(f)* = 0,64). Por fim, a NEH-*warm-start* de uma forma geral apresenta piores eficiência e robustez do que o modelo matemático.

Vahedi-Nouri et al. (2022) debruçaram-se sobre o problema de planeamento da mão de obra e programação da produção em sistemas RMS (*Reconfigurable Manufacturing System*), implementando modelos matemáticos CP e MILP para a resolução deste problema. Os autores desenvolveram versões melhoradas dos modelos, as quais nomearam como I-CP e I-MILP. Posteriormente, propuseram uma abordagem híbrida baseada em *warm-start* com as versões melhoradas dos 2 modelos (I-CP-WS), onde o modelo I-MILP fornece uma solução inicial ao modelo I-CP. Concretizando, o I-MILP resolve o problema, definindo as atribuições máquina-*setup* e as alocações dos trabalhadores às máquinas. Esta solução parcial é o ponto de partida para o modelo I-CP. Para a análise pretendida foram preparadas 33 instâncias com diferentes combinações de tarefas (entre 15 e 110), máquinas (entre 3 e 16) e *setups* por máquina (entre 3 e 7).

Os modelos propostos conseguiram reduções significativas de RPD comparativamente às versões originais (cerca de 22% no I-CP e 95% no I-MILP), salientando o RPD médio de 0,59 do I-MILP. Como I-MILP apresentou melhor desempenho que I-CP, a abordagem híbrida de *warm-start* (I-CP-WS) é comparada com o modelo I-MILP. Foi possível constatar que o I-CP-WS obteve o melhor RPD médio, proporcionando uma redução de 4,44% (I-MILP) para 0,91%, com tempos computacionais bastante inferiores. O método *warm-start* apresenta um valor médio de tempo computacional de 264 segundos, o que contrasta com o valor médio de 413 segundos do I-MILP. Para terminar, a técnica de *warm-start* relevou ser a mais robusta, proporcionando os melhores resultados em quase todas as instâncias e uma menor variação do RPD nos vários testes.

Numa análise do problema de programação de equipas de manutenção preventiva nos sistemas ferroviários, M. Pour et al. (2018) desenvolveram uma estrutura híbrida, em que numa primeira fase um modelo CP gera soluções iniciais viáveis para alimentar um modelo MIP que, numa segunda fase, melhora a solução recebida. Para efetuar o estudo foram dimensionadas 4 instâncias reais (D2, D4, D6 e D8 que diferem no número de semanas). Nos testes computacionais, o modelo MIP apenas resolveu o problema de 2 semanas (D2) dentro do limite temporal de 3 horas. Esta dificuldade foi contornada com a abordagem *warm-start*, onde o CP reduz muito o tempo requerido pelo MIP para produzir uma solução. De facto, as soluções geradas pela abordagem híbrida possuem um *gap* relativo ao LB que varia entre 3% e 23%. Além disso, esta abordagem encontrou soluções viáveis para problemas com até 8 semanas. Para os planos de 4, 6 e 8 semanas (D4, D6 e D8), a abordagem híbrida CP/MIP obtém soluções viáveis geradas na fase de construção. Na fase de melhoria, o CP/MIP destaca-se bastante com um *gap* relativo que varia entre 16% e 22%.

Fleszar & Hindi (2018) para o problema de máquinas não relacionadas com restrição de recursos, propuseram uma abordagem de duas etapas. Numa fase inicial usaram um modelo MILP para

calcular um LB e alocar as tarefas às máquinas. Em seguida, recorreram a um modelo CP para sequenciar as tarefas nas respetivas máquinas. Caso o algoritmo mencionado anteriormente não encontre a solução ótima, esta solução é fornecida como *warm-start* a um modelo CP completo para o problema PMR. Com um tempo limite de 60 segundos, este mecanismo permitiu obter soluções ótimas em todas as instâncias com 8 e 12 tarefas, apresentando melhor comportamento que a abordagem de duas etapas sozinha. Também permite melhorias ao nível do UB e LB para instâncias com até 16 tarefas. Mantendo este tempo limite de computação, os autores compararam a sua estratégia com os algoritmos de Fanjul-Peyro et al. (2017), alcançando melhores soluções com menos tempo computacional. Para instâncias com até 30 tarefas, a estratégia de *warm-start* obteve um *gap* médio de 0,10% e um *gap* máximo de 4,14%, enquanto os algoritmos de Fanjul-Peyro et al. (2017) obtêm um *gap* médio de 1,35% e *gap* máximo de 22,03%. Além disso, com um tempo limite de 300 segundos, a estratégia *warm-start* encontra soluções ótimas em todas as instâncias com até 16 tarefas, sendo o *gap* médio de todos os testes de 0,46%. No caso específico de 500 e 1000 tarefas, obtiveram-se *gaps* médio de 1,13% e 1,22%, respetivamente.

Face ao exposto, ficou claro que uma abordagem *warm-start* pode ser uma excelente decisão para acelerar o processo de obtenção de solução ótima, principalmente quando se estudam instâncias de grandes dimensões, independentemente do problema a ser tratado. De facto, esta abordagem consiste num método de solução eficiente, especialmente quando é necessária uma solução de alta qualidade num tempo computacional adequado e limitado.

2.6. Indicadores de desempenho e *solvers*

Nesta secção é feito um levantamento relativamente aos indicadores de desempenho e *solvers* mais utilizados, de forma a contextualizar as escolhas efetuadas no presente projeto.

Indicadores de desempenho

Na programação matemática, associada aos objetivos do problema são considerados indicadores de desempenho que podem variar consoante a especificidade de cada caso. Para o problema de programação de máquinas paralelas foi efetuado um levantamento dos principais indicadores de desempenho considerados, apurando-se que geralmente o objetivo está associado à minimização do *makespan*. Contudo, a minimização dos recursos necessários, dos tempos de conclusão das tarefas e do atraso total das tarefas são também objetivos que muitas vezes são considerados em problemas reais. Em alguns casos são estabelecidos indicadores mais específicos para determinadas realidades, como são os casos da quantidade produzida dentro de um horizonte temporal, do número de movimentações de recursos e do consumo energético das máquinas. Quando apenas um objetivo não é suficiente para garantir a viabilidade do problema são seguidas abordagens multiobjetivo, onde se considera mais que um indicador de desempenho.

A Tabela 1 apresenta os indicadores de desempenho de alguns casos de estudo de máquinas paralelas, onde podemos ver os exemplos mencionados anteriormente.

Tabela 1 - Indicadores de desempenho utilizados em estudos de máquinas paralelas

Caso de Estudo	Problema	Estratégia	Recursos Adicionais	Indicador de Desempenho
Soares & Carvalho (2022)	Máquinas idênticas	Metaheurística	Sim	*Makespan*
Gedik et al. (2018)	Máquinas não relacionadas	CP	Não	*Makespan*
Heinz et al. (2022)	Máquinas idênticas	CP e heurística matemática	Sim	*Makespan*
Yunusoglu & Yildiz (2022)	Máquinas não relacionadas	CP	Sim	*Makespan*
Fleszar & Hindi (2018)	Máquinas não relacionadas	MILP, CP e heurística matemática	Sim	*Makespan*
Fanjul-Peyro et al. (2019)	Máquinas não relacionadas	MILP	Não	*Makespan*
Lopez-Esteve et al. (2022)	Máquinas não relacionadas	Metaheurística	Sim	*Makespan*
Fanjul-Peyro et al. (2017)	Máquinas não relacionadas	MILP e heurística matemática	Sim	*Makespan*
Fanjul-Peyro (2020)	Máquinas não relacionadas	MILP	Sim	*Makespan*
Yepes-Borrero et al. (2020)	Máquinas não relacionadas	Metaheurística	Sim	*Makespan*
Bektur & Saraç (2019)	Máquinas não relacionadas	MILP e Metaheurística	Sim	Atraso das tarefas
Moser et al. (2022)	Máquinas não relacionadas	MILP e metaheurística	Não	Atraso das tarefas
Yepes-Borrero et al. (2021)	Máquinas não relacionadas	Metaheurística	Sim	Multiobjetivo[1]
Bitar et al. (2021)	Máquinas não relacionadas	MILP	Sim	Multiobjetivo[2]
Zhang et al. (2021)	Máquinas não relacionadas	CEA[3]	Sim	Multiobjetivo[4]

Solvers

Em termos computacionais, para a resolução de problemas através de modelos matemáticos são necessários *solvers*. Na atualidade da literatura do problema de escalonamento, as opções comerciais mais usadas, por se destacarem em relação às demais, são o CPLEX e o GUROBI. Este último apresenta melhor desempenho na generalidade dos estudos analisados.

De referir que muitas vezes os *solvers* comerciais podem não ser opções viáveis para as empresas, pois exigem uma licença de utilização, assim sendo, surgem alternativas de código aberto com desempenho consideravelmente competitivo, como os casos do *COIN-OR* (*CBC*), *SCIP* e *GLPK*.

[1] Makespan e número de recursos

[2] Número de produtos concluídos antes do final de um determinado horizonte temporal, soma ponderada dos tempos de conclusão e número de movimentações de recursos auxiliares.

[3] *Combinatorial Evolutionary Algorithm*

[4] Atraso ponderado das tarefas e consumo energético das máquinas.

Na Tabela 2 são apresentados alguns casos de estudo onde é possível efetuar a comparação entre diferentes *solvers* para diferentes problemas, refletindo o que foi descrito até então.

Tabela 2 – Comparação do desempenho entre diferentes *solvers*

Caso de Estudo	Problema	*Solvers*	Melhor *solver*
Fanjul-Peyro et al. (2019)	Máquinas paralelas	CPLEX, GUROBI	GUROBI
Fanjul-Peyro (2020)	Máquinas paralelas	CPLEX, GUROBI	GUROBI
Vázquez-Serrano et al. (2021)	Máquinas paralelas	CPLEX, GUROBI, COIN-OR (CBC)	GUROBI
Nguyen et al. (2013)	Máquina única	GLPK, GUROBI, COIN-OR (CBC)	GUROBI
Ku & Beck (2016)	*Job-shop*	CPLEX, GUROBI, SCIP	CPLEX
Hatami et al. (2013)	*Flowshop*	CPLEX, GUROBI	GUROBI
Wang et al. (2020)	Máquinas paralelas	CPLEX, GUROBI	GUROBI
Hatami et al. (2015)	Máquinas paralelas	CPLEX, GUROBI	CPLEX

2.7. Modelos matemáticos de referência para o projeto

Este subcapítulo é composto pela formulação matemática dos dois modelos de referência para este projeto. Os modelos foram desenvolvidos por Fanjul-Peyro (2020). Inicialmente é apresentada a formulação comum aos dois modelos e de seguida as componentes específicas de cada um deles.

Os parâmetros comuns necessários são:

- Um conjunto de *m* máquinas que podem processar as tarefas, denotado como $M = \{1, \dots, m\}$, indexado por i.
- Um conjunto de n tarefas a serem processadas, $N = \{1, \dots, n\}$, indexado por j, k.
- p_{ij} representa o tempo necessário para a máquina i processar a tarefa j.
- s_{ijk} representa o tempo necessário para efetuar o *setup* na máquina i entre as tarefas j e k.
- r_{max}^{P} é a quantidade máxima de recursos de processamento. Estes recursos são necessários especificamente para o processamento das tarefas e não podem ser usados nos *setups*, sendo denotados como recursos-P.
- r_{max}^{S} é a quantidade máxima de recursos de recursos de *setup*. Estes recursos são necessários especificamente para as operações de *setup* e não podem ser usados no processamento das tarefas, sendo denotados como recursos-S.
- r_{max}^{H} é a quantidade máxima de recursos não específicos que podem ser usados no processamento e nos *setups*. Estes recursos são denotados como recursos-H.
- r_{ij}^{P} são os recursos-P necessários para o processamento da tarefa j na máquina i.
- r_{ijk}^{S} são os recursos-S necessários para o *setup* entre a tarefa j e a tarefa k na máquina i.
- r_{ij}^{HP} são os recursos-H necessários para o processamento da tarefa j na máquina i. O processamento da tarefa j na máquina i pode necessitar r_{ij}^{P} recursos-P e r_{ij}^{HP} recursos-H.
- r_{ijk}^{HS} são os recursos-H necessários para o *setup* entre as tarefas j e k na máquina i. O *setup* entre as tarefas j e k na máquina i pode necessitar r_{ijk}^{S} recursos-P e r_{ijk}^{HS} recursos-H.
- *M* é uma constante suficientemente grande.

As variáveis de decisão comuns aos dois modelos são:

- $X_{ijk} = 1$ se a tarefa k é processada imediatamente a seguir à tarefa j na máquina i, $X_{ijk} = 0$ caso contrário.
- $Y_{ik} = 1$ se a tarefa k é processada na máquina i, $Y_{ik} = 0$ caso contrário.
- C_j é o tempo de conclusão da tarefa *j* (coordenada onde o processamento da tarefa *j* termina na dimensão temporal) (minutos).
- B_j é o tempo de conclusão do *setup* antes do processamento da tarefa *j* na máquina correspondente (coordenada onde o *setup* da tarefa *j* termina na dimensão temporal) (minutos).

A formulação matemática comum a ambos os modelos é a seguinte:

$$Min\ Z = C_{max} \tag{1}$$

$$\sum_{j\in N_0, k\in N, j\neq k} s_{ijk}X_{ijk} + \sum_{k\in N} p_{ik}Y_{ik} \leq C_{max}, i \in M \tag{2}$$

$$\sum_{k\in N} X_{i0k} \leq 1, i \in M \tag{3}$$

$$\sum_{i\in M} Y_{ik} = 1, k \in N \tag{4}$$

$$Y_{ik} = \sum_{j\in N_0, j\neq k} X_{ikj}, i \in M, k \in N \tag{5}$$

$$Y_{ik} = \sum_{j\in N_0, j\neq k} X_{ijk}, i \in M, k \in N \tag{6}$$

$$C_k - C_j + M(1 - X_{ijk}) \geq s_{ijk} + p_{ik}, i \in M, j \in N_0, k \in N, j \neq k \tag{7}$$

$$C_k \leq C_{max}, k \in N \tag{8}$$

$$X_{ijk} \in \{0,1\}, Y_{ij} \geq 0, C_k \geq 0, C_0 = 0, j, k \in N, i \in M \tag{9}$$

$$B_k \leq C_k - \sum_{i\in M} Y_{ik}p_{ik}, k \in N \tag{10}$$

$$B_k - C_j + M\left(1 - \sum_{i\in M} X_{ijk}\right) \geq \sum_{i\in M} s_{ijk}X_{ijk}, j \in N_0, k \in N, j \neq k \tag{11}$$

$$C_k \geq \sum_{i\in M, j\in N_0} s_{ijk}X_{ijk} + \sum_{i\in M} Y_{ik}p_{ik}, k \in N \tag{12}$$

As variáveis de decisão específicas para o MILP *strip-packing* são:

- $R_j^P \in [0, r_{max}^P]$ é a coordenada do processamento da tarefa *j* na dimensão recursos-P (recursos específicos necessários para o processamento da tarefa *j*).

- $R_j^S \in [0, r_{max}^S]$ é a coordenada do *setup* da tarefa j na dimensão recursos-S (recursos específicos necessários para o *setup* que ocorre antes do processamento da tarefa j).
- $D_{jk}^P, E_{jk}^P, F_{jk}^S\ G_{jk}^S, U_{jk}^{HP}, U_{jk}^{HS}, V_{jk}^{HP}, U_{jk}^{HS}$ são variáveis binárias auxiliares com o valor = 1 se o valor da variável estudada nas restrições (tempo ou recursos) da tarefa k é maior que o valor da variável estudada (tempo ou recursos) da tarefa j. As letras S, P, HP e HS são usadas para indicar a dimensão em que as variáveis são usadas.

Neste MILP utilizam-se as equações (1) – (12) e as restantes restrições são:

$$r_{max}^P \geq R_k^P \geq \sum_{i \in M} Y_{ik} r_{ik}^P, k \in N \tag{13}$$

$$r_{max}^S \geq R_k^S \geq \sum_{i \in M, l \in N} X_{ilk} r_{ilk}^S, k \in N \tag{14}$$

$$r_{max}^H \geq R_k^{HP} \geq \sum_{i \in M} Y_{ik} r_{ik}^{HP} \geq R_k^{HS} \geq \sum_{i \in M, l \in N} X_{ilk} r_{ilk}^{HS}, k \in N \tag{15}$$

$$C_k - C_j + M\left(1 - D_{jk}^P\right) \geq \sum_{i \in M} Y_{ik} p_{ik}, j, k \in N, j \neq k \tag{16}$$

$$R_k^P - R_j^P + M\left(1 - E_{jk}^P\right) \geq \sum_{i \in M} Y_{ik} r_{ik}^P, j, k \in N, j \neq k \tag{17}$$

$$D_{jk}^P + D_{kj}^P + E_{jk}^P + E_{kj}^P \geq 1, j, k \in N, j \neq k \tag{18}$$

$$B_k - B_j + M\left(1 - F_{jk}^S\right) \geq \sum_{i \in M, l \in N_0} X_{ilk} s_{ilk}, j, k \in N, j \neq k \tag{19}$$

$$R_k^S - R_j^S + M\left(1 - G_{jk}^S\right) \geq \sum_{i \in M, l \in N_0} X_{ilk} r_{ilk}^S, j, k \in N, j \neq k \tag{20}$$

$$F_{jk}^S + F_{kj}^S + G_{jk}^S + G_{kj}^S \geq 1, j, k \in N, j \neq k \tag{21}$$

$$C_k - B_j + M\left(1 - U_{jk}^{HP}\right) \geq \sum_{i \in M} Y_{ik} p_{ik}, j, k \in N, j \neq k \tag{22}$$

$$B_j - C_k + M\left(1 - U_{kj}^{HS}\right) \geq \sum_{i \in M, l \in N_0} X_{ilj} r_{ilj}^S, j, k \in N, j \neq k \tag{23}$$

$$R_k^{HP} - R_j^{HS} + M\left(1 - V_{jk}^{HP}\right) \geq \sum_{i \in M} Y_{ik} r_{ik}^{HP}, j, k \in N, j \neq k \tag{24}$$

$$R_j^{HS} - R_k^{HP} + M\left(1 - V_{jk}^{HS}\right) \geq \sum_{i \in M, l \in N_0} X_{ilj} r_{ilj}^{HS}, j, k \in N, j \neq k \tag{25}$$

$$U_{jk}^{HP} + U_{kj}^{HS} + V_{jk}^{HP} + V_{kj}^{HS} \geq 1, j, k \in N, j \neq k \tag{26}$$

A formulação matemática do modelo MILP indexado ao tempo é a seguinte:

- $E_{ikt} = 1$ se k estiver em processamento na máquina i no instante t, $E_{ikt} = 0$ caso contrário.
- $F_{ijkt} = 1$ se k estiver em *setup* depois da tarefa j na máquina i no instante t, $F_{ijkt} = 0$ caso contrário.
- t_{max} é o horizonte do planeamento.

E_{ikt} e F_{ijkt} são variáveis de decisão e t_{max} é um parâmetro. Neste MILP utilizam-se as equações (1) – (12). As restantes restrições são:

$$\sum_{t \le t_{max}} E_{ikt} \ge p_{ik} Y_{ik}\,, i \in M, k \ \in N \tag{27}$$

$$C_k \ge \sum_{i \in M} t E_{ikt}\,, k \ \in N, t \le t_{max} \tag{28}$$

$$C_k + 1 - \sum_{i \in M} p_{ik} Y_{ik} \le \sum_{i \in M} t E_{ikt} + M\left(1 - \sum_{i \in M} E_{ikt}\right), i \in M, k \ \in N, t \le t_{max} \tag{29}$$

$$\sum_{t \le t_{max}} F_{ijkt} \ge s_{ijk} X_{ijk}\,, i \in M, j \in N_0, k \ \in N, j \ne k \tag{30}$$

$$B_k \ge \sum_{i \in M, j \in N_0} t F_{ijkt}\,,, k \ \in N, t \le t_{max} \tag{31}$$

$$B_k + 1 - \sum_{i \in M, j \in N_0} s_{ijk} X_{ijk} \le \sum_{i \in M, j \in N_0} t F_{ijkt} + M\left(1 - \sum_{i \in M, j \in N_0} F_{ijkt}\right), k \in N, t \le t_{max} \tag{32}$$

$$r_{max}^{P} \ge \sum_{i \in M, k \in N} r_{ik}^{P} E_{ikt}\,, t \le t_{max} \tag{33}$$

$$r_{max}^{S} \ge \sum_{i \in M, j,k \in N, j \ne k} r_{ijk}^{S} F_{ijkt}\,, t \le t_{max} \tag{34}$$

$$r_{max}^{H} \ge \sum_{i \in M, j,k \in N} r_{ik}^{HP} E_{ikt} + \sum_{i \in M, j,k \in N, j \ne k} r_{ijk}^{HS} F_{ijkt}\,, t \le t_{max} \tag{35}$$

3. MÉTODOS E APLICAÇÃO

Neste capítulo inicialmente é feita a descrição do problema de máquinas paralelas dedicadas. De seguida, são apresentados os dois modelos matemáticos implementados que foram adaptados à realidade do problema, destacando-se as principais contribuições deste estudo. No último subcapítulo são descritas as diferentes abordagens que compõem a heurística matemática, onde métodos heurísticos trocam informações com o modelo matemático através de técnicas de *warm-start* e *lower bound*.

3.1. Descrição do problema

Este estudo é motivado pelo departamento de produção da empresa INPLAS que produz diversas peças de plástico por injeção para a indústria automóvel. A empresa responde a encomendas curtas e pequenas com um prazo de entrega de 48 horas. A produção é feita por turnos e o escalonamento é fundamental para garantir o nível de serviço neste mercado exigente. O número de operadores é conhecido no início de cada turno, o que exige que o sistema seja rápido o suficiente para fornecer soluções no início do turno.

Formalmente, um conjunto de n tarefas é processado num conjunto de m máquinas paralelas, onde a alocação das tarefas às máquinas é conhecida previamente, resultando num problema de máquinas dedicadas. Para processar as tarefas é necessária uma quantidade discreta de recursos específicos limitados. Cada tarefa pertence a uma família, tem um molde associado e é processada numa única máquina sem interrupção. O mesmo molde pode produzir vários tipos de peças introduzindo diferentes composições de materiais. Os moldes são montados nas máquinas de acordo com a sua compatibilidade.

As peças são organizadas em famílias que compartilham o molde de produção, o que oferece uma vantagem ao eliminar a necessidade de trocar o molde entre essas peças. Nesse sentido, vários lotes da mesma família podem ser programados. Um *setup* dependente da sequência deve ser realizado entre tarefas que pertençam a famílias diferentes e a máquina não pode executar nenhuma tarefa durante a operação de *setup*. É importante realçar que as operações de *setup* devem começar e terminar no mesmo turno.

Os operadores são recursos de processamento, sendo apenas necessários enquanto as tarefas são processadas. As equipas de *setup* são recursos necessários quando uma máquina termina de processar uma tarefa e necessita de ser reconfigurada porque a família da próxima tarefa é diferente. O número de *setups* simultâneos é limitado pela disponibilidade de pontes rolantes. Os recursos são renováveis porque podem ser reaproveitados por diferentes máquinas, estando sujeitos a uma restrição geral de consumo. Os recursos também são dinâmicos, o que significa que a quantidade alocada a uma máquina pode variar ao longo do horizonte de tempo.

Cada máquina só pode processar uma tarefa de cada vez e possui uma configuração inicial que resulta do facto de se tratar de uma operação contínua. Considerando o número de operadores disponíveis para o turno, a configuração inicial das máquinas pode variar. De facto, pode ser necessário que uma máquina esteja parada no início do programa de produção. Nesse caso, o tempo de conclusão da tarefa inicial será maior que o seu tempo de processamento. Se uma determinada tarefa não for concluída no turno anterior, esta deve ser agendada como a tarefa

inicial no turno atual para terminar o seu processamento. O tempo de processamento da tarefa inicial no turno atual será igual ao tempo restante para concluir o processamento da tarefa. Nesta transição de turnos não se exige nenhum tempo de *setup*.

O objetivo do departamento de produção é responder à procura diária de peças, respeitando os limites de capacidade e a disponibilidade de operadores e de equipas de *setup*. Face ao exposto, o sequenciamento das tarefas em cada máquina é limitado pela quantidade de recursos existentes, sendo a principal distinção desta realidade relativamente aos problemas clássicos de escalonamento.

3.2. Principais contribuições

Neste subcapítulo descrevem-se as principais contribuições deste estudo que incluem a definição de uma nova função objetivo, a consideração do estado inicial das máquinas, a adaptação dos modelos de referência ao problema de máquinas paralelas dedicadas e a generalização dos modelos para lidar com a realidade de *setups* dependentes da sequência de famílias.

3.2.1. Nova função objetivo

Tendo em conta o conhecimento do autor, nenhum trabalho utiliza como função objetivo a minimização da soma dos *makespan* de todas as máquinas, o que confere uma componente inovadora a este projeto. Na literatura do PMSR, a minimização do *makespan* é uma medida de desempenho muito comum, no entanto, esse objetivo não é adequado para a realidade do problema deste estudo. Como as máquinas são dedicadas, minimizar o *makespan* terá impacto apenas na máquina *bottleneck*, originando tempos ociosos nas restantes máquinas, tal como se observa na Figura 1.

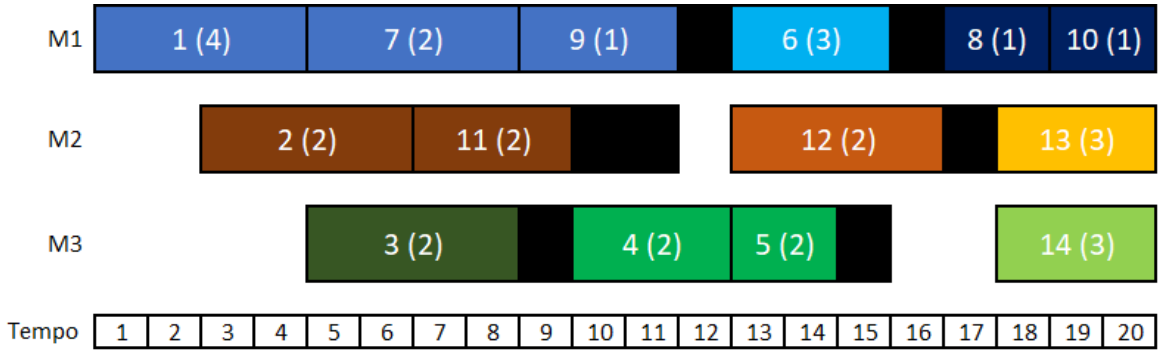


Figura 1 - Solução ótima para a minimização do *makespan*

Por outro lado, a função objetivo proposta apresenta virtudes na resolução do problema estudado. Na verdade, obtém-se um cronograma da produção compacto, o que permite a minimização dos tempos ociosos de todas as máquinas e a maximização da utilização das mesmas. Além disso, com o objetivo de processar as tarefas em cada máquina o mais rápido possível, o sequenciamento das tarefas será executado no sentido de minimizar o número de *setups* necessários. A eficiência da função objetivo proposta é expressa na Figura 2. De realçar que cada retângulo corresponde a uma tarefa diferente, sendo identificada com o seu número. O número de operadores necessários para o processamento de cada tarefa encontra-se entre parênteses. As famílias são identificadas com os

diferentes tons de cor e a operação de *setup* é representada pelos retângulos pretos. No exemplo considerou-se um número de operadores disponíveis de 8 e 1 equipa de *setup*.

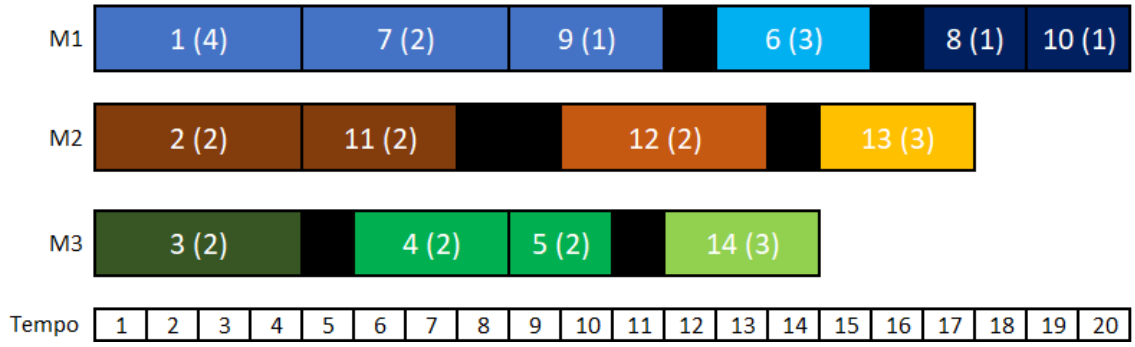


Figura 2 - Solução ótima para a minimização da soma dos *makespan* de todas as máquinas

3.2.2. Configuração inicial das máquinas

Outro aspeto significativo abordado neste projeto é a inclusão da configuração inicial das máquinas. Os modelos gerais publicados até ao momento não suportam esse tipo de problema prático real, pois não são projetados para considerar o estado inicial do sistema de produção. Contudo, na realidade, os sistemas de produção são contínuos e possuem sempre uma configuração inicial, tornando-se essencial clarificar o estado das máquinas durante a transição de turnos. Com a inclusão desta componente, através da introdução das restrições (37) garantiu-se que as primeiras tarefas processadas em cada máquina são as tarefas que transitam do turno anterior, denominadas como tarefas iniciais. Além disso, com a adaptação das restrições (42) define-se que a conclusão do processamento das tarefas iniciais será maior ou igual ao seu tempo de processamento. De referir que as restrições referidas serão apresentadas no subcapítulo 3.3.

3.2.3. Realidade de máquinas paralelas dedicadas

No artigo de referência deste projeto, os modelos matemáticos desenvolvidos por Fanjul-Peyro (2020) tratam o problema de máquinas paralelas não relacionadas. Nesse caso, as tarefas podem ser processadas em qualquer uma das máquinas, sendo necessário definir a alocação das tarefas às máquinas. Os tempos de processamento e de *setup* dependam dessa alocação (p_{ij} e s_{ijk}).

No problema estudado neste trabalho, como as máquinas são dedicadas, os tempos de processamento e de *setup* não dependem da máquina a que as tarefas estão alocadas, o que faz com que estes dois parâmetros não necessitem do índice associado à máquina (p_j e s_{jk}).

Além disso, foi efetuada uma simplificação do modelo em termos de variáveis devido a se conhecer previamente a atribuição das tarefas às máquinas. De facto, a variável Y_{ik} (do modelo de referência presente no subcapítulo 2.7), que indica se a tarefa k está alocada à máquina i, não é necessária no problema de máquinas paralelas dedicadas. A alocação das tarefas às máquinas é guardada através dos m subconjuntos N_i que agrupam as tarefas a serem processadas em cada máquina i. Como se poderá ver através da equação (43), essas restrições são garantidas para cada conjunto de tarefas que pertence à máquina i. Para definir uma condição entre tarefas de máquinas diferentes,

foi necessário distinguir o índice da máquina (N_i e N_q) tal como é visível nas restrições (51)-(56). As restrições referidas serão apresentadas no subcapítulo 3.3.

Face ao exposto, foi necessário adaptar os modelos matemáticos à realidade do problema de máquinas paralelas dedicadas.

3.2.4. Generalização do modelo para *setups* dependentes da sequência de famílias

O modelo matemático geral não permite a resolução de problemas com *setups* dependentes da sequência de famílias, uma vez que obriga à existência de *setup* sempre que ocorre o término de uma tarefa. Na Figura 3 é representada esta situação, onde o *setup* da máquina M2 não se realiza imediatamente a seguir ao término do processamento da tarefa 2 porque o modelo de referência considera que nesse momento está a correr um *setup* na máquina M1, o que não é verdade. Neste sentido, foi necessário adaptar o modelo de referência à realidade de *setups* dependentes da sequência de famílias. Esta generalização é garantida através das restrições (44), onde a condição só se aplica nas situações em que ocorra *setup* ($s_{jk} > 0$). Isto significa que apenas as tarefas k que possuam $s_{jk} > 0$ terão um $B_k > 0$. Assim, o modelo proposto neste estudo já permite a realização do *setup* da máquina M2 imediatamente a seguir ao término do processamento da tarefa 2, tal como se vê na Figura 4.

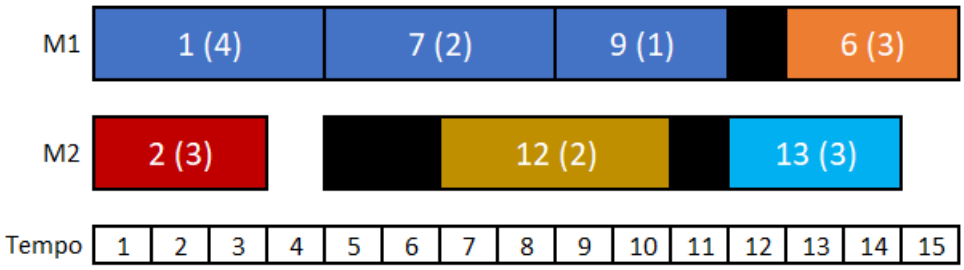


Figura 3 - Modelo matemático de referência não resolve problemas com famílias de *setup*

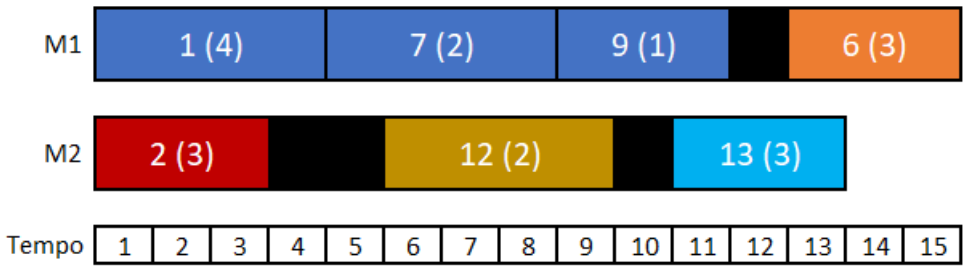


Figura 4 – Modelo matemático proposto resolve o problema com famílias de *setup*

3.3. Modelos matemáticos

Nesta secção apresenta-se formalmente o problema de máquinas paralelas dedicadas com *setups* dependentes da sequência de famílias e recursos adicionais. Neste projeto, foram implementados dois modelos matemáticos introduzidos por Fanjul-Peyro (2020), um baseado na ideologia *strip-packing* e o outro indexado ao tempo. Estes modelos foram adaptados à realidade do problema de

máquinas paralelas dedicadas e à existência de *setups* dependentes da sequência de famílias. As máquinas possuem uma configuração inicial considerando-se um sistema de produção contínuo. Além disso, é proposta uma nova função objetivo para minimizar a soma dos *makespan* de todas as máquinas.

A formulação matemática comum aos dois modelos é apresentada de seguida. De salientar que se utilizam as letras minúsculas para definir os parâmetros e letras maiúsculas para definir as variáveis. Os conjuntos de parâmetros são:

- Um conjunto de m máquinas que podem processar as tarefas, denotado como $M = \{1, \dots, m\}$, indexado por i.
- Um conjunto de $m + n$ tarefas a serem processadas, $N = \{1, \dots, m + n\}$, indexado por j, onde as primeiras m tarefas representam as tarefas iniciais.
- p_j representa o tempo necessário para processar a tarefa j.
- s_{jk} representa o tempo necessário para efetuar o *setup* entre as tarefas j e k.
- r_{max}^P é a quantidade máxima de recursos de processamento. Estes recursos são necessários especificamente para o processamento das tarefas e não podem ser usados nos *setups*, sendo denotados como recursos-P.
- r_{max}^S é a quantidade máxima de recursos de *setup*. Estes recursos são necessários especificamente para as operações de *setup* e não podem ser usados no processamento das tarefas, sendo denotados como recursos-S.
- r_j^P são os recursos-P necessários para o processamento da tarefa j.
- r_j^S são os recursos-S necessários para o *setup* da tarefa j.
- *M* é uma constante suficientemente grande.

Como as máquinas são dedicadas, o conjunto de N tarefas é dividido em m subconjuntos $(N_1, \dots, N_m)$ de modo que as tarefas do conjunto N_i são apenas processadas na máquina i. Cada máquina tem uma tarefa fictícia (0) representando o início e o fim do sequenciamento da máquina. Portanto, $N_{i0} = N_i \cup \{0\}$ representa o conjunto de tarefas que são processadas na máquina i adicionando a tarefa fictícia. Além disso, os tempos de processamento e de *setup* (p_j and s_{jk}), bem como os recursos necessários para cada operação (r_j^P and r_j^S) são determinísticos.

As variáveis de decisão da formulação comum são as seguintes:

- $X_{ijk} = 1$ se a tarefa k é processada imediatamente a seguir à tarefa j na máquina i, $X_{ijk} = 0$ caso contrário.
- $Cmax_i$ é o tempo de conclusão da máquina i (minutos).
- C_j é o tempo de conclusão da tarefa j (coordenada onde o processamento da tarefa j termina na dimensão temporal) (minutos).
- B_j é o tempo de conclusão do *setup* antes do processamento da tarefa j na máquina correspondente (coordenada onde o *setup* da tarefa j termina na dimensão temporal) (minutos).

A formulação matemática comum a ambos os modelos é a seguinte:

$$Min\ Z = \sum_{i \in M} Cmax_i \tag{36}$$

$$X_{i0i} = 1, i \in M \tag{37}$$

$$\sum_{j \in N_i, j \neq k} X_{ijk} = 1, i \in M, k \in N_i \backslash \{i\} \tag{38}$$

$$\sum_{j \in N_i \backslash \{i\}, j \neq k} X_{ikj} = 1, i \in M, k \in N_i \tag{39}$$

$$C_k - C_j + M\left(1 - X_{ijk}\right) \geq s_{jk} + p_k, i \in M,\ j \in N_i, k \in N_i \backslash \{i\}, j \neq k \tag{40}$$

$$C_k \geq \sum_{j \in N_i, j \neq k, s_{jk} > 0} X_{ijk}\, s_{jk} + p_k, k \in N_i \backslash \{i\}, i \in M \tag{41}$$

$$C_i \geq p_i, i \in M \tag{42}$$

$$Cmax_i \geq C_k, i \in M, k \in N_i \tag{43}$$

$$B_k - C_j + M\left(1 - X_{ijk}\right) \geq s_{jk},\ i\ \in M, j \in N_i, k\ \in N_i \backslash \{i\}, j \neq k, s_{jk} > 0 \tag{44}$$

$$B_k \leq C_k - p_k, k\ \in N \tag{45}$$

$$X_{ijk} \in \{0,1\}, C_k \geq 0, B_k \geq 0\ j, k \in N_i,\ i \in M \tag{46}$$

A função objetivo (36) minimiza a soma dos *makespan* de todas as máquinas. As restrições (37) garantem que as tarefas iniciais são as primeiras a serem processadas em cada máquina imediatamente a seguir à tarefa fictícia. As restrições (38) e (39) garantem que apenas uma tarefa é processada imediatamente antes e apenas uma tarefa é processada imediatamente depois de cada tarefa. As restrições (40) impedem sub-rotas e fornecem a ordem correta do escalonamento, porque se k é processada depois de j na máquina i, então a conclusão do processamento da tarefa k não ocorre mais cedo que $s_{jk} + p_k$. As restrições (41) definem o limite inferior do tempo de conclusão da tarefa k (C_k) como sendo maior ou igual à soma do seu *setup* mais o seu tempo de processamento, enquanto as restrições (42) fazem o mesmo para as tarefas iniciais. As restrições (43) estabelecem a relação entre o $Cmax_i$ da máquina i e o tempo de conclusão das tarefas nessa máquina. As restrições (44) impõem que, se a tarefa j for processada antes da tarefa k na máquina i, a conclusão do tempo de *setup* k deve ser posterior à conclusão do processamento da tarefa j mais o tempo de *setup* da tarefa k. As restrições (45) garantem que o tempo de conclusão do *setup* da tarefa k ocorre mais cedo que o início do processamento da tarefa k. Finalmente, as restrições (46) definem os limites das variáveis. Neste modelo, além de várias simplificações do modelo original, a função objetivo (36) e as restrições (37) são novas para refletir as condições específicas do problema real.

3.3.1. Modelo *strip-packing*

Neste modelo matemático, a ideologia *bin-packing 2D* é adaptada ao problema de escalonamento, sendo que as tarefas são representadas através de *items* e as máquinas consistem em *bins*. De referir que para tarefas de máquinas diferentes serem processadas em paralelo são necessários mais recursos em simultâneo. Tendo uma das dimensões fixas, o objetivo da metodologia consiste em colocar um conjunto de tarefas dentro de uma caixa retangular de forma a minimizar a outra dimensão. Cada tarefa é representada por um retângulo em que a sua largura corresponde ao seu tempo de processamento (p_j) e a sua altura corresponde aos recursos necessários no seu processamento (r_j). A posição de cada retângulo é definida pelas coordenadas do seu canto superior direito (C_j, R_j).

De seguida, na Figura 5 e na Figura 6 é apresentado um exemplo da aplicação do *strip-packing* considerando recursos de processamento (eixo y na Figura 5) e tempos de conclusão do processamento das tarefas (eixo x na Figura 5). Neste caso, a dimensão fixa consiste nos recursos, definida pelo $r_{max}^P = 8$ e pretende-se minimizar os tempos de conclusão. A tarefa 1 está alocada à máquina M1 e as tarefas 2 e 3 alocadas à máquina M2. A tarefa 1 é a primeira a ser alocada, existindo 2 opções para o sequenciamento das tarefas da máquina M2. A primeira opção consiste no processamento da tarefa 2 antes da tarefa 3. Tendo em conta a restrição de recursos, a tarefa 2 pode ser processada em simultâneo com a tarefa 1 da máquina M1, tal como se pode ver na Figura 5. De salientar que nesta situação, a coordenada R_2^P resulta da soma dos recursos necessários para o processamento das tarefas 1 e 2.

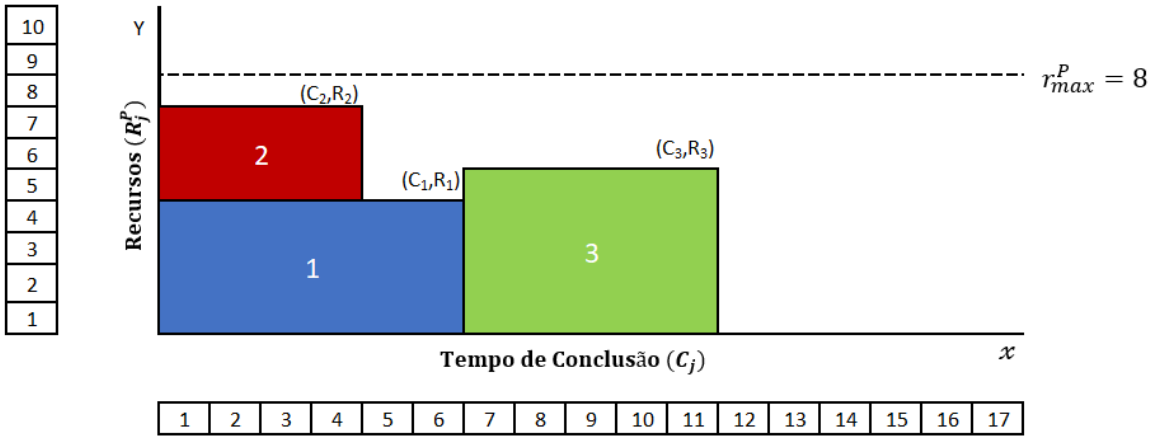


Figura 5 - Exemplo da metodologia strip-packing (opção 1)

A segunda opção consiste no processamento da tarefa 3 antes da tarefa 2 (Figura 6). Neste caso, as tarefas 1 e 3 não podem ser processadas em simultâneo, uma vez que isso não respeita a restrição de recursos. Devido a esta situação, o processamento da tarefa 3 apenas inicia após o término do processamento da tarefa 1, o que aumenta a soma dos tempos de conclusão das máquinas comparativamente à opção anterior. Para os recursos de *setup* a ideologia é a mesma.

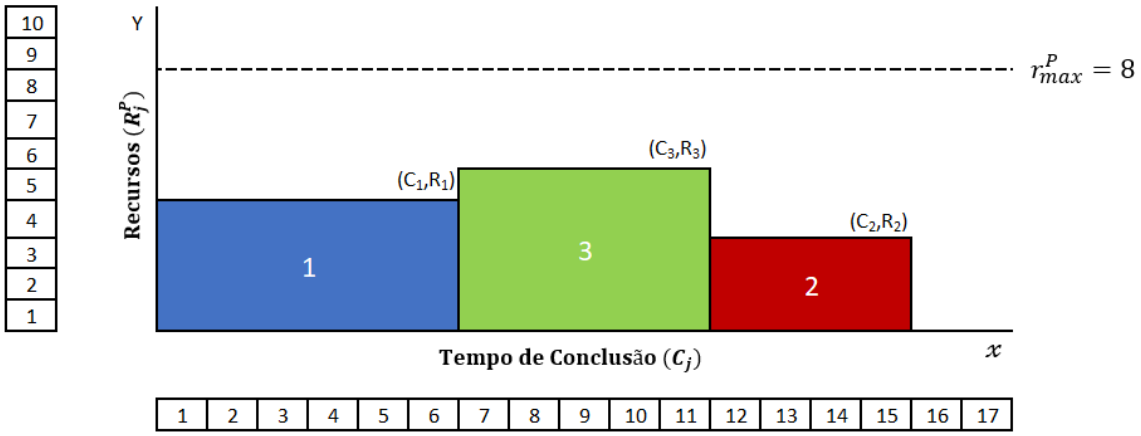


Figura 6 - Exemplo da metodologia *strip-packing* (opção 2)

Após a contextualização da metodologia do *strip-packing*, de seguida são apresentadas as variáveis de decisão específicas deste modelo matemático. As letras S e P indicam a dimensão em que as variáveis são usadas.

- $R_j^P \in [0, r_{max}^P]$ é a coordenada do processamento da tarefa j na dimensão recursos-P (recursos específicos necessários para o processamento da tarefa j).
- $R_j^S \in [0, r_{max}^S]$ é a coordenada do *setup* da tarefa j na dimensão recursos-S (recursos específicos necessários para o *setup* que ocorre antes do processamento da tarefa j).
- $D_{jk}^P, E_{jk}^P, F_{jk}^S\ G_{jk}^S$ são variáveis binárias auxiliares com o valor = 1 se o valor da variável estudada nas restrições (tempo ou recursos) da tarefa k for maior que o valor da variável estudada (tempo ou recursos) da tarefa j.

Neste MILP utilizam-se as equações (36) – (46) e as restantes restrições são:

$$r_{max}^P \geq R_k^P, k \in N \quad (47)$$

$$R_k^P \geq r_k^p, k \in N \quad (48)$$

$$r_{max}^S \geq R_k^S, k \in N \quad (49)$$

$$R_k^S \geq r_k^S, k \in N \quad (50)$$

$$C_k - C_j + M\left(1 - D_{jk}^P\right) \geq p_k, i, q \in M, q \neq i, j \in N_i, k \in N_q \quad (51)$$

$$R_k^P - R_j^P + M\left(1 - E_{jk}^P\right) \geq r_k^P, i, q \in M, q \neq i, j \in N_i, k \in N_q \quad (52)$$

$$D_{jk}^P + D_{kj}^P + E_{jk}^P + E_{kj}^P \geq 1, i, q \in M, q \neq i, j \in N_i, k \in N_q, j \neq k \quad (53)$$

$$B_k - B_j + M\left(1 - F_{jk}^S\right) \geq \sum_{l \in N_{q0}, l \neq k, s_{lk} > 0} X_{qlk}\, s_{lk}, i, q \in M, q \neq i,\ j \in N_i, k \in N_q \quad (54)$$

$$R_k^S - R_j^S + M\left(1 - G_{jk}^S\right) \geq r_k^S, i, q \in M, q \neq i,\ j \in N_i, k \in N_q \quad (55)$$

$$F_{jk}^S + F_{kj}^S + G_{jk}^S + G_{kj}^S \geq 1, i, q \in M, q \neq i\ , j \in N_i, k \in N_q \quad (56)$$

As restrições (47) – (50) definem os limites das coordenadas dos recursos específicos usados no processamento (R_k^P) e recursos específicos usados nos *setups* (R_k^S). As restrições (51) mostram que, se C_k for posterior a C_j, a diferença entre esses tempos de conclusão é maior ou igual ao tempo de processamento da tarefa k. Em (52) se R_k^P for maior que R_j^P, a diferença entre as coordenadas desses pontos de recursos é maior ou igual aos recursos utilizados no processamento da tarefa k. As restrições (53) forçam pelo menos uma das variáveis binárias a ter o valor 1. As restrições (54) mostram que, se B_k for posterior a B_j, a diferença entre esses tempos de conclusão de *setup* é maior ou igual ao tempo de *setup* da tarefa k. Em (55) se R_k^S for maior que R_j^S, a diferença entre as coordenadas desses pontos de recursos é maior ou igual aos recursos utilizados no *setup* da tarefa k. As restrições (56) forçam pelo menos uma das variáveis binárias a ter o valor 1.

3.3.2. Modelo indexado ao tempo

No modelo indexado ao tempo, as variáveis possuem um índice relacionado com o tempo e cada unidade de tempo é controlada individualmente, sendo o tempo tratado como uma dimensão discreta. A formulação matemática do modelo MILP indexado ao tempo é a seguinte:

- $E_{ikt} = 1$ se k estiver em processamento na máquina i no instante t, $E_{ikt} = 0$ caso contrário.
- $F_{ijkt} = 1$ se k estiver em *setup* depois da tarefa j na máquina i no instante t, $F_{ijkt} = 0$ caso contrário.
- t_{max} é o horizonte do planeamento.

E_{ikt} e F_{ijkt} são variáveis de decisão e t_{max} é um parâmetro. Neste MILP utilizam-se as equações (36) – (46). As restantes restrições são:

$$\sum_{t \le t_{max}} E_{ikt} \ge p_k, i \in M, k \in N_i \tag{57}$$

$$C_k \ge tE_{ikt}, i \in M, k \in N_i, t \le t_{max} \tag{58}$$

$$C_k + 1 - p_k \le tE_{ikt} + M(1 - E_{ikt}), i \in M, k \in N_i, t \le t_{max} \tag{59}$$

$$\sum_{t \le t_{max}} F_{ijkt} \ge s_{jk} X_{ijk}, i \in M, j \in N_i, k \in N_i, j \ne k, s_{jk} > 0, k \ge m \tag{60}$$

$$B_k \ge \sum_{j \in N_i, j \ne k} tF_{ijkt}, i \in M, k \in N_i, t \le t_{max}, s_{jk} > 0, k \ge m \tag{61}$$

$$B_k + 1 - \sum_{j \in N_i, j \ne k, s_{jk} > 0} s_{jk} X_{ijk} \le \sum_{j \in N_i, j \ne k, s_{jk} > 0} tF_{ijkt} + M\left(1 - \sum_{j \in N_i, j \ne k, s_{jk} > 0} F_{ijkt}\right), i \in M, k \in N_i, t \le t_{max} \tag{62}$$

$$R_{max}^P \ge \sum_{i \in M, k \in N_i} r_k^P E_{ikt}, t \le t_{max} \tag{63}$$

$$R_{max}^{S} \geq \sum_{i \in M, j, k \in N_i, j \neq k, s_{jk} > 0} r_k^S F_{ijkt}, t \leq t_{max} \quad (64)$$

As restrições (57) e (58), em conjunto, garantem que a variável E_{ikt} apenas assume o valor de 1 durante o tempo exato em que a tarefa k está a ser processada na máquina i. Isto evita tempo inativo desnecessário da máquina e garante que cada tarefa seja processada pelo tempo mínimo necessário para concluí-la. As restrições (59) complementam as restrições anteriores exigindo que quando E_{ikt} for igual a 1, o tempo atual t deve ser maior ou igual ao tempo de conclusão da tarefa k menos o seu tempo de processamento mais 1. As restrições (60) – (62) representam o mesmo, mas para os *setups*, onde o tempo considerado é o tempo de *setup*. As restrições (63) garantem que a soma de todos os recursos utilizados no processamento em cada instante t seja menor ou igual ao número máximo de recursos disponíveis para processamento. As restrições (64) garantem o mesmo para os *setups*.

3.4. Estratégias da heurística matemática

Uma vez que se suspeitava que o modelo matemático teria dificuldade na resolução de problemas de grande dimensão, definiu-se que seria vantajoso desenvolver heurísticas matemáticas em que o modelo matemático troca informações com métodos heurísticos. O objetivo passou por implementar técnicas para facilitar o esforço do modelo matemático, encurtando o espaço de soluções a analisar. Assim, foram seguidas estratégias de *warm-start*, onde métodos heurísticos fornecem uma solução inicial válida e um limite superior (*upper bound*) ao modelo matemático. Numa primeira fase foi desenvolvida uma heurística construtiva, sendo posteriormente, implementado um método mais composto e robusto, a metaheurística *Tabu Search*. Além das técnicas de *warm-start*, foi construída uma heurística construtiva para gerar soluções que possam funcionar como *lower bound*. Estas estratégias são descritas de seguida.

3.4.1. Heurística CR

A heurística construtiva desenvolvida é composta por duas fases: construção e reparação, tendo por base o método seguido por Yepes-Borrero et al. (2020). Numa primeira etapa é construída uma solução considerando a existência de recursos ilimitados. Na segunda etapa, a solução obtida pela fase construtiva é analisada de forma a verificar se as restrições de recursos são satisfeitas. Esta heurística gera soluções válidas e assume a designação de CR: “Construção e Reparação”.

Fase Construtiva

Comparativamente à abordagem seguida por Yepes-Borrero et al. (2020), a fase de construção foi adaptada à realidade do problema estudado. De facto, foi necessário ajustar a forma de construir a solução devido a se tratar um problema com *setups* dependentes da sequência de famílias e em que as máquinas são dedicadas.

Concretizando, na primeira fase do algoritmo, as tarefas são agrupadas por famílias, sendo importante definir a ordem entre as famílias. A primeira família a ser sequenciada corresponde à família da tarefa inicial. Na transição de famílias, a próxima a ser selecionada é a responsável por um menor tempo de *setup*. O procedimento repete-se até que todas as famílias da máquina sejam

sequenciadas. Posteriormente, com exceção da tarefa inicial, as posições das tarefas dentro de cada família são definidas aleatoriamente. Este processo repete-se em todas as máquinas. Para terminar esta fase, a solução é construída partindo do pressuposto que existem recursos ilimitados.

Para clarificar o funcionamento da fase construtiva da heurística apresenta-se um exemplo. Considere-se um problema com 2 máquinas e 10 tarefas (2 iniciais e 8 normais) e a alocação das tarefas às máquinas encontra-se na Tabela 3. Os tempos de processamento e a família de cada tarefa encontram-se na Tabela 4. Os tempos de *setup* dependentes da sequência de famílias estão registados na Tabela 5, sendo que a sequência entre famílias é dada pela ordem linha-coluna, ou seja, caso a tarefa 6 (família 2) seja processada imediatamente a seguir à tarefa 9 (família 1), o valor do *setup* é 1 ($s_{96} = 1$).

Tabela 3 – Alocação das tarefas às máquinas

Máquina	**Tarefa Inicial**	**Tarefas normais**
M1	1	6, 7, 8, 9, 10
M2	2	11, 12, 13

Tabela 4 – Tempos de processamento e famílias de *setup* das tarefas

Tarefa *j*	**1**	**2**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**
Família	1	4	2	1	3	1	3	4	5	6
p_j (min)	4	4	3	4	2	3	2	3	4	3

Tabela 5 – Tempos de *setup* dependentes da sequência de famílias

setup (min)	1	2	3	4	5	6
1	-	1	2	3	2	1
2	1	-	2	2	1	3
3	3	1	-	3	2	2
4	2	2	2	-	2	2
5	1	1	3	2	-	1
6	3	3	2	1	3	-

Começando pela máquina M1, a primeira família a ser sequenciada é a família 1 porque a tarefa inicial pertence a esta família. Para definir a próxima família, sabendo que sucede a família 1, é necessário identificar qual proporciona o menor tempo de *setup*. As opções existentes são 1-2 (1 minuto) e 1-3 (2 minutos). A próxima família a ser sequenciada é a família 2 porque o menor *setup* é 1 minuto. Todas as famílias estão sequenciadas e a ordem entre elas é 1,2,3. Finalmente, as posições das tarefas em cada família são definidas aleatoriamente, sendo uma possível solução: 1-7-9-6-8-10. Na máquina M2, o procedimento repete-se seguindo a mesma ordem de ideias. Para terminar a primeira fase do algoritmo procede-se à construção da solução sem ter em conta a restrição de recursos, tal como é visível na Figura 7.

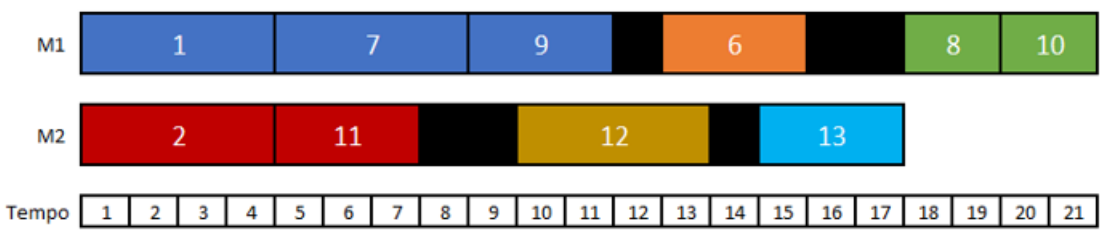


Figura 7 - Construção da solução sem considerar a restrição de recursos

O pseudocódigo da fase construtiva é apresentado na Figura 8 onde F_i representa as famílias da máquina i.

for $i \in M$ **do**
    Group jobs by families;
    The family of the initial job is placed in position 1; $allocated\ families \leftarrow 1$;
    while $allocated\ families < len(F_i)$ **do**
        Find the best family to place in the next position taking into account the shortest setup time;
    end
    for $f \in F_i$ **do**
        Randomly sequence the jobs of family *f*;
    end
    for $j \in N_i$ **do**
        Save values of C_j and B_j taking into account unlimited resources;
    end
end

Figura 8 - Pseudocódigo da fase construtiva da heurística

Mecanismo de Reparação

Na segunda fase do algoritmo, a solução fornecida pela fase construtiva é avaliada para verificar se as restrições de recursos são respeitadas. Este método de avaliação consiste em calcular o número total de recursos necessários em cada instante de tempo. Caso as restrições de recursos sejam satisfeitas num momento, o algoritmo avalia o próximo instante de tempo. Caso não sejam respeitadas as restrições de recursos, é aplicado o mecanismo de reparação que consiste em adiar o início da operação que inicia mais tarde para o momento de término da operação que termina mais cedo. De seguida, a quantidade de recursos necessários nesse momento é novamente avaliada e, caso a restrição de recursos seja respeitada, avançasse para o próximo momento. Este processo é repetido até que todos os instantes de tempo sejam validados.

O mecanismo de reparação desenvolvido por Yepes-Borrero et al. (2020) aplica-se a um problema em que apenas existem recursos de *setup*. Neste seguimento, uma das contribuições da heurística construtiva CR proposta neste estudo é a adaptação do mecanismo de reparação no sentido de corrigir soluções nos dois tipos de operações, processamentos e *setups*.

Na Figura 9 é possível visualizar o funcionamento do mecanismo de reparação. No exemplo ilustrado, o número máximo de operadores disponíveis (r_{max}^{P}) é 6 e existe 1 equipa de *setup*. Nesta situação, percebe-se que a restrição de operadores não é respeitada no momento inicial porque são necessários 7 operadores. Assim sendo, esta solução necessita de ser reparada. Como as tarefas 1 e 2 terminam ao mesmo tempo, a tarefa que sofrerá alterações será a que necessita de menos operadores. Assim, o processamento da tarefa 2 é adiado para o momento de tempo 4 (término do processamento da tarefa 1), o que corresponde à correção 1 da Figura 9. Após esta correção, as restrições de recursos são satisfeitas até ao instante de tempo 11. Nesse momento ocorrem 2 *setups*, o que não é possível devido a só existir 1 equipa de *setup*. Nesta situação adia-se o início do *setup* da máquina M2 para o fim do *setup* da máquina M1. A partir daqui as restrições de recursos são satisfeitas em todos os instantes de tempo e, por isso, a solução final é válida.

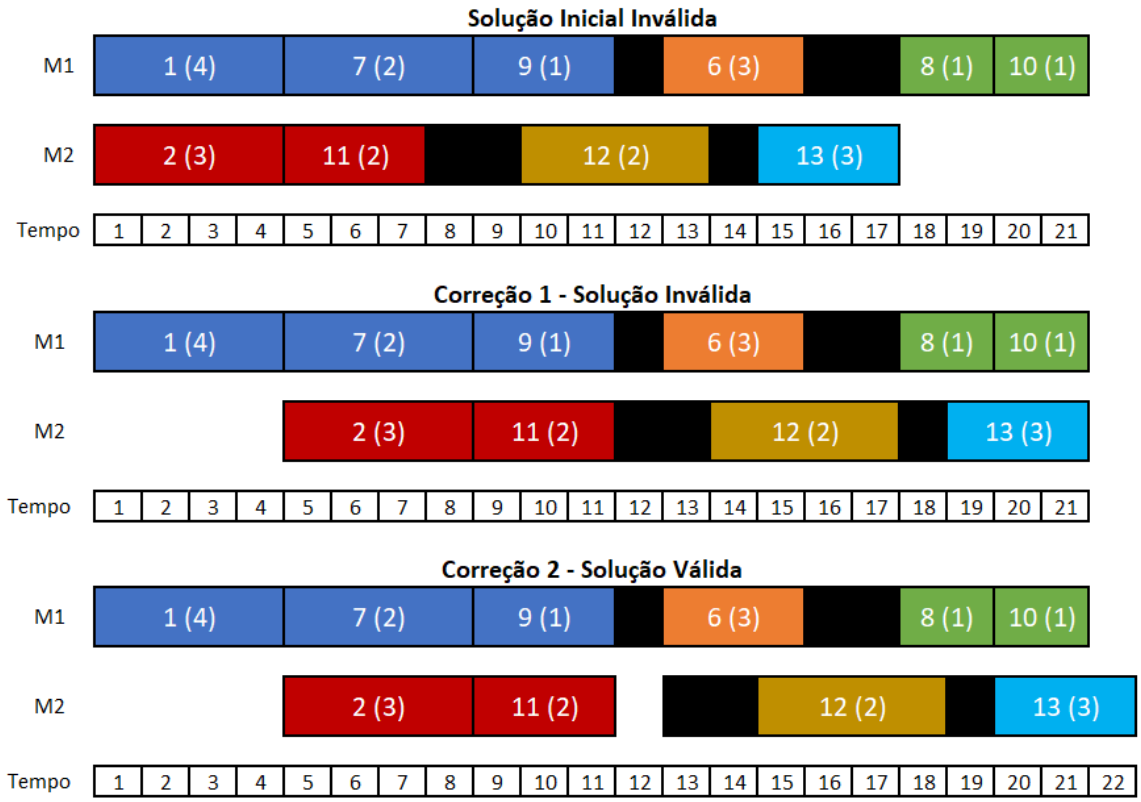


Figura 9 - Exemplo do mecanismo de reparação

O pseudocódigo do mecanismo de reparação é apresentado na Figura 10.

$timeP \leftarrow 0$; $timeS \leftarrow 0$;
while $timeP \leq max\ (C_j)$ or $timeS \leq max\ (B_j)$ **do**
    Evaluate the shortest completion time of job not validated ($timeP$);
    if $required\ operators > r^P_{max}$ **then**
        Postpone the begining of the processing in the machine that begins latest out of those that overlap at instante $timeP$;
        Correct the remaining jobs of the same machine if necessary;
    end
    Evaluate the shortest completion time of setup not validated ($timeS$);
    if $required\ setup\ teams > r^S_{max}$ **then**
        Postpone the begining of the setup in the machine that begins latest out of those that overlap at instante $timeS$;
        Correct the remaining jobs of the same machine if necessary;
    end
end

Figura 10 - Pseudocódigo do mecanismo de reparação

3.4.2. Tabu Search

O algoritmo *Tabu Search* foi introduzido por Glover (1986) e é utilizado com sucesso em muitos problemas de otimização, nomeadamente nos problemas de escalonamento. Bilge et al. (2004) desenvolveram um algoritmo TS para o escalonamento de máquinas paralelas uniformes, considerando *setups* dependentes da sequência e com o objetivo de minimizar o atraso total. Saidi-

Mehrabad & Fattahi (2007) implementaram o TS para estudar o problema *job shop* com o objetivo de minimizar o *makespan*. Na área de máquinas paralelas não relacionadas, Bozorgirad & Logendran (2012) propuseram um algoritmo TS para minimizar o atraso e os tempos de conclusão, considerando *setups* dependentes da sequência e datas de entrega. Mais recentemente, Bektur & Saraç (2019) estudaram um problema de máquinas paralelas não relacionadas com tempos de *setup* dependentes da sequência e restrições de elegibilidade da máquina. Para minimizar o atraso total ponderado, os autores propuseram um algoritmo TS.

Como se provará nos resultados computacionais, a heurística construtiva CR não consegue encontrar a solução ótima nos diferentes tipos de instâncias. Portanto, nesta secção é proposta uma metaheurística que consiste num método mais composto e robusto. A metaheurística desenvolvida é o *Tabu Search* (TS) tendo por base o trabalho de Bektur & Saraç (2019).

O algoritmo é composto por uma fase construtiva e uma fase de *local search*, utilizando também uma estrutura de memória para escapar a ótimos locais. Na fase construtiva é fornecida uma solução inicial pela heurística CR descrita na secção anterior. Para melhorar esta solução, é-lhe aplicado um processo de *local search*. Neste sentido, a fase construtiva está associada a uma estratégia de diversificação (memória de longo prazo) consistindo numa pesquisa de regiões não visitadas. A fase de *local search* é uma estratégia de intensificação (memória de curto prazo), onde um espaço de pesquisa é explorado. No final da fase de *local search* é gerada uma nova solução inicial e o processo de intensificação é reiniciado.

De referir que Bektur & Saraç (2019) abordam um problema de máquinas paralelas não relacionadas apenas com recursos de *setup* e, por isso, a heurística utilizada na sua fase construtiva tem em conta essa realidade. Assim sendo, a heurística construtiva CR desenvolvida neste projeto é diferente para refletir o problema estudado de máquinas dedicadas com a inclusão de *setups* dependentes da sequência de famílias e com dois tipos de recursos adicionais.

Relativamente à fase de *local search*, a geração de vizinhança é feita através do método *swap* em que duas tarefas na mesma máquina trocam de posição. Após a aplicação do movimento *swap*, a solução é novamente avaliada em termos de restrições de recursos através do mecanismo de reparação. Como o problema em causa trata máquinas paralelas dedicadas, a quantidade de trocas possíveis entre as tarefas da mesma máquina não é muito grande. Assim sendo, para cada solução em análise são gerados todos os movimentos *swap* possíveis. Cada uma destas soluções é avaliada, com o objetivo de perceber se a melhor solução será atualizada. Na Figura 11 encontra-se um exemplo dos 3 possíveis movimentos *swap* numa máquina partindo de uma solução inicial. O processo de *local search* através dos movimentos *swap* é retratado na Figura 12.

Esta metaheurística possui um parâmetro denominado *tabu list* que impede que um determinado movimento *swap* seja realizado caso esse movimento esteja na *tabu list*. A dimensão da *tabu list* é constante e quando esta se encontra totalmente preenchida, o movimento há mais tempo na lista é eliminado. Como não é possível a troca de tarefas entre máquinas diferentes, foi criado uma *tabu list* para cada máquina. Este parâmetro possui uma exceção, uma vez que se um determinado movimento estiver na *tabu list* e for responsável pela melhor solução encontrada até ao momento, essa solução é aceite. Além deste, existem ainda dois parâmetros que consistem nos números máximos de iterações efetuadas na fase construtiva (g) e na fase de *local search* (v).

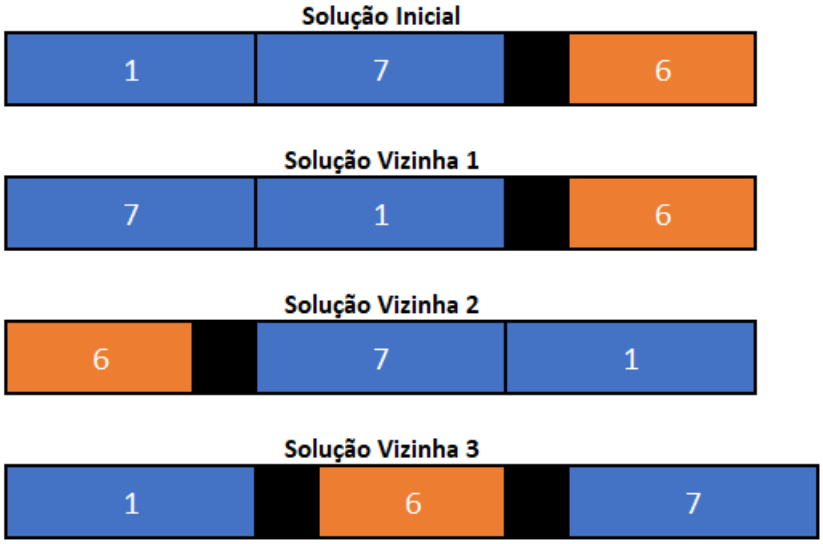


Figura 11 – Exemplo da estratégia de movimentos *swap*

for $i \in M$ **do**
  for $j = 1, \dots, len(N_i \backslash \{i\})$ **do**
    for $k = 1, \dots, j$ **do**
      if $j \neq k$ **then**
        Apply swap;
        Apply repairing mechanism;
        Save S_n^t and E_n^t;
      end
    end
  end
end

Figura 12 - Pseudocódigo da fase de local search

Os passos do algoritmo TS surgem na Figura 13, sendo importante definir alguns termos utilizados no mesmo.

- $MTLS$: dimensão máxima da *tabu list*.
- TLL_i: dimensão atual da *tabu list* da máquina *i*.
- g: número máximo de iterações da fase de diversificação (construção).
- v: número máximo de iterações da fase de intensificação (*local search*).
- $iterlong$: contador das iterações para a fase de diversificação (construção).
- $itershort$: contador das iterações para a fase de intensificação (*local search*).
- E_c: valor da função objetivo da solução atual.
- E_n: valor da função objetivo da solução vizinha.
- E_n^t: valor da função objetivo da solução vizinha nº t.
- E_{best}: valor da função objetivo da melhor solução.
- S_0: solução inicial.
- S_c: solução atual.
- S_n: solução vizinha.
- S_n^t: solução vizinha nº t.
- S_{best}: melhor solução.

$iterlong \leftarrow 0$;
while $iterlong < g$ **do**
    Obtain S_0 and E_0 using the constructive heuristic;
    $iterlong \leftarrow iterlong + 1$; $TLL \leftarrow 1$; $itershort \leftarrow 1$;
    $S_c \leftarrow S_0$; $E_c \leftarrow E_0$;
    if $E_0 < E_{best}$ **then**
        $S_{best} \leftarrow S_0$ and $E_{best} \leftarrow E_0$;
    end
    while $itershort < v$ **do**
        Generate neighbor solutions and sort ascending order according to objective function value (S_n^t); $t = 1$; $check = 0$;
        while $check == 0$ **do**
            if the movement of S_n^t not tabu or $E_n^t < E_{best}$ **then**
                $S_c \leftarrow S_n^t$; $E_c \leftarrow E_n^t$;
                Save the index of the machine i involved in the swap movement;
                Insert the movement of S_c at the bottom of tabu list i;
                $TLL_i \leftarrow TLL_i + 1$; $check \leftarrow 1$;
            else
                $t \leftarrow t + 1$;
            end
        end
        if $TLL_i == MTLS + 1$ **then**
            $Delete\ the\ first\ element\ in\ the\ tabu\ list$;
            $TLL_i \leftarrow TLL_i - 1$;
        end
        if $E_c < E_{best}$ **then**
            $S_{best} \leftarrow S_c$; $E_{best} \leftarrow E_c$;
        end
        $itershort \leftarrow itershort + 1$;
    end
end

Figura 13 - Pseudocódigo da metaheurística *Tabu Seach*

3.4.3. Heurística SCTUR

Para simplificar o esforço do modelo matemático, procuraram-se técnicas que reduzissem o espaço de soluções inferiormente. Assim, desenvolveu-se uma heurística construtiva que gera soluções com recursos infinitos e, por isso, inválidas. O objetivo era apurar se esta estratégia reduzia mais o espaço de soluções a analisar que o valor da relaxação linear. O resultado da heurística é garantidamente menor que a solução ótima, sendo esse valor fornecido ao modelo matemático como um *lower bound*. O nome da heurística é *Shortest Completion Time and Unlimited Resources* (SCTUR) que significa "menor tempo de conclusão e recursos infinitos". O procedimento dessa heurística é explorado de seguida.

O processo é efetuado em cada máquina. Inicialmente, somam-se os tempos de processamento das tarefas alocadas a essa máquina. De seguida, para cada tarefa, com exceção da inicial, seleciona-se o menor tempo de *setup* possível. Estes menores valores são adicionados à soma dos tempos de processamento das tarefas, resultando no tempo total dessa máquina. O procedimento repete-se para todas as máquinas e no fim somam-se os tempos totais de cada máquina e esse

valor corresponde ao *lower bound*. O pseudocódigo da heurística desenvolvida encontra-se na Figura 14.

for $i \in M$ **do**
    $time\_machine_i \leftarrow 0$;
end
for $i \in M$ **do**
    for $j \in N_i$ **do**
        $time\_machine_i \leftarrow time\_machine_i + p_j$;
    end
end
for $k \in N_i \backslash \{i\}$ **do**
    Find the shortest setup time of job k (*short*);
    $time\_machine_i \leftarrow time\_machine_i + short$;
end

Figura 14 - Pseudocódigo da heurística de recursos infinitos

Demonstração

Retomando o exemplo da secção anterior e considerando a máquina M1 com as tarefas 1,6,7,8,9 e 10 e a máquina M2 com as tarefas 2,11,12 e 13. Na fase inicial a heurística soma todos os tempos de processamento das máquinas, o que representa uma solução com recursos ilimitados, Figura 15.

$$time\_machine_1 = 4 + 3 + 4 + 2 + 3 + 2 \; = \; 18\, minutos \tag{65}$$

$$time_{machine_2} = 4 + 3 + 4 + 3 = 14\, minutos \tag{66}$$

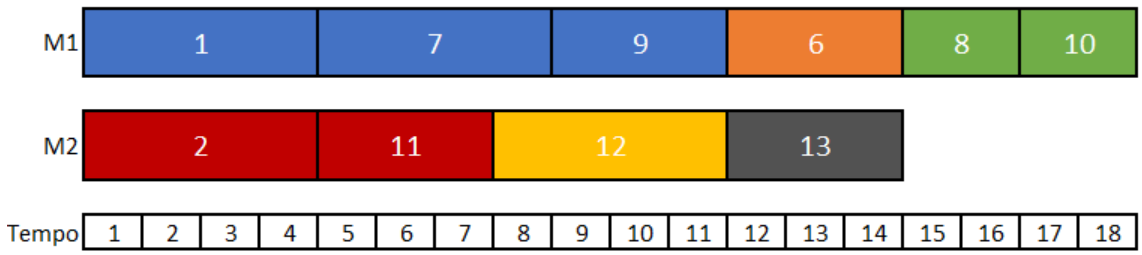


Figura 15 – Exemplo da primeira fase da heurística SCTUR

A próxima fase consiste em determinar o menor valor de *setup* para cada tarefa, excluindo a tarefa inicial. Retomando a Máquina M1, existem 2 opções para o *setup* da tarefa 6 (que pertence à família 2). De facto, antes desta tarefa pode surgir uma tarefa da família 1 ou uma tarefa da família 3. Os tempos de *setup* para a primeira opção (família 1 - família 2) e para a segunda opção (família 3 - família 2) são ambos de 1 minuto. Assim sendo, o menor valor de *setup* para a tarefa 6 seria 1 minuto. Para a tarefa 7 (que pertence à família 1), as opções para o *setup* são família 1 – família 1, família 2 - família 1 e família 3 - família 1. Neste caso como a tarefa 7 pertence à mesma família que outras tarefas nesta máquina (tarefas 1 e 9), o menor valor seria 0 minutos. O mesmo acontece para as tarefas 8,9 e 10. Neste sendo, é possível concluir que para as tarefas que pertençam à mesma família de outras tarefas na máquina, o menor valor será 0. No caso das famílias que possuam apenas uma tarefa na máquina, o menor valor resulta das possibilidades de transição entre essa família e as restantes. Assim sendo, os 5 menores valores de *setup* (1,0,0,0,0) são

adicionamos ao valor da soma dos tempos de processamento (18 minutos). O tempo total da máquina M1 obtém-se com a seguinte expressão:

$$time\_machine_1 = 18 + 1 + 0 + 0 + 0 + 0 = 19\ minutos \quad (67)$$

Assim, o tempo da máquina 1 seria 19 minutos. O procedimento repete-se para a máquina M2 e no fim soma-se os tempos totais de cada máquina e esse valor corresponde ao *lower bound*.

Como na máquina M1 existem 3 famílias, numa solução válida existirão pelo menos 2 *setups* nessa máquina e cada um desses tempos será pelo menos 1 minuto (Tabela 5). Acrescentando estes tempos de *setup* à soma dos tempos de processamento, dá garantidamente um valor superior aos 19 minutos. Face ao exposto, o valor calculado pela heurística SCTUR é garantidamente menor que o *makespan* da M1 na solução ótima. O mesmo se confirma para a máquina M2 e, por isso, para a soma das duas máquinas.

4. RESULTADOS COMPUTACIONAIS

Nesta secção são apresentados os resultados do estudo computacional. Primeiramente é explorada a arquitetura de dados e o processo de geração de instâncias. De seguida, os dois modelos matemáticos são comparados em termos de desempenho e qualidade das formulações. Finalmente, é realçada a eficiência da heurística matemática através dos resultados das diferentes abordagens que a compõem.

Relativamente à arquitetura de dados, as instâncias foram geradas aleatoriamente no Excel com base na realidade da empresa. Estes dados foram fornecidos ao ambiente de desenvolvimento *Microsoft Visual Studio Code 2023 IDE*, onde a programação foi executada em linguagem *Python* com recurso à biblioteca *Pulp* e utilizando o *solver CPLEX 22.1.1.0*. Apesar de na revisão bibliográfica se ter apurado que na generalidade o *GUROBI* apresenta melhor desempenho, através de estudos experimentais concluiu que nas versões mais recentes dos *solvers* o *CPLEX* superou o *GUROBI*.

Os testes computacionais foram executados num processador *Intel Core i7-85654 CPU1.80GHz*, com 8.00 GB de RAM. Para garantir tempos computacionais razoáveis foi estabelecido um limite de tempo de 1800 segundos para os modelos matemáticos. Ao longo do estudo computacional, os indicadores analisados foram o *gap* para a solução ótima, a percentagem de soluções ótimas obtidas e o tempo computacional. A arquitetura de dados pode ser consultada na Figura 16.

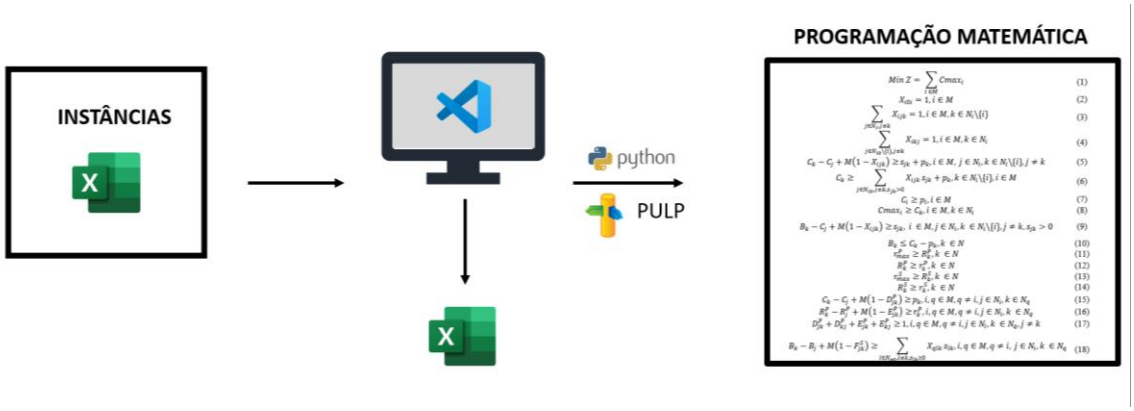


Figura 16 – Arquitetura de Dados

4.1. Geração de instâncias

Para a geração de instâncias, com base na realidade da empresa, foi necessário definir os números de tarefas, máquinas, equipas de *setup* e operadores disponíveis. Assim, foram criados cinco conjuntos de problemas com dimensões diferentes, tal como se pode ver na Tabela 6. Os conjuntos de maiores dimensões, denominados extra grande, foram criados exclusivamente para testar os limites computacionais da metaheurística TS.

Para cada um dos conjuntos foram geradas aleatoriamente 10 instâncias com tempos de processamento, tempos de *setup* e famílias por máquina gerados a partir de distribuições uniformes de $U(20,240)$, $U(30,50)$, e $U(1,3)$, respetivamente. O número de operadores para suportar o processamento das tarefas nas máquinas foi gerado segundo a distribuição uniforme de $U(1,4)$. A alocação das tarefas às máquinas também é um processo aleatório. Os tempos de *setup*

devem satisfazer a desigualdade triangular. Cada *setup* necessita de uma equipa de *setup*. Para o modelo indexado ao tempo, o t_{max} foi definido como 1000 minutos.

O número de operadores disponíveis (r_{max}^{P}) em cada conjunto de instâncias foi determinado com base na expressão usada por Fanjul-Peyro (2020), onde r_{max} representa o número máximo de operadores necessários para uma tarefa e r_{min} representa o número mínimo possível.

$$r_{max}^{P} = m\frac{r_{max} - r_{min}}{2} = m\frac{4-1}{2} = 2{,}5m \tag{68}$$

Tabela 6 – Características das instâncias de teste

Dimensão da Instância	Máquinas	Tarefas	Operadores	Equipas de *setup*
Pequena (P)	4	12	10	1
Média (M)	8	24	20	2
Grande (G)	16	40	40	2
Extra Grande I (EGI)	32	80	80	2
Extra Grande II (EGII)	32	100	80	2

Para identificar cada instância, utiliza-se a letra associada à sua dimensão e o número da respetiva instância. Por exemplo, a instância número 2 do conjunto de problemas pequeno, será identificada como P_2.

4.2. Parâmetros do *Tabu Search*

Na implementação de metaheurísticas, a parametrização é um aspeto muito importante que consiste na afinação dos valores dos parâmetros. Este processo foi efetuado por experimentação.

Neste sentido, relembrando que os parâmetros utilizados no *Tabu Search* são a dimensão da *tabu list*, o número de iterações da fase de diversificação (g) e o número de iterações da fase de intensificação (v). As combinações de parâmetros que apresentaram bons resultados encontram-se na Tabela 7.

Tabela 7 - Valores dos parâmetros usados no *Tabu Search*

tabu list	$\boldsymbol{g}$	$\boldsymbol{v}$
3	50	250

4.3. Comparação entre os modelos matemáticos

Antes de avaliar o desempenho de cada um dos modelos matemáticos, considerou-se oportuno analisar a dimensão de cada um dos modelos, bem como a qualidade da formulação dos mesmos.

Neste sentido, para comparar a dimensão dos dois modelos em termos de número de variáveis e restrições, recorreu-se a um exemplo de uma instância com 4 máquinas e 12 tarefas. Tal como é visível na Tabela 8, o número de variáveis e restrições no modelo indexado ao temo é muito superior ao modelo *strip-packing*. Assim, a resolução dos problemas de teste será mais difícil no modelo indexado ao tempo.

Tabela 8 – Exemplo da comparação dos modelos matemáticos em termos de dimensão

	Modelo indexado ao tempo	**Modelo *strip-packing***
Número de variáveis	18771	834
Número de restrições	24141	1265

No sentido de apurar a qualidade da formulação dos modelos, comparou-se os valores obtidos através da relaxação linear com as soluções ótimas, como é possível consultar na Tabela 9 e na Tabela 10. De salientar que nas instâncias de dimensão média só são apresentadas as instâncias que se conhece o valor da solução ótima.

Tabela 9 – Resultados da Relaxação Linear – para as instâncias de pequena dimensão

Instância	**Modelo indexado ao tempo**	**Modelo *strip-packing***	**Desvio (%)**	**Solução ótima**
P_0	790,00	790,00	49,84	1575
P_1	1152,00	1152,00	39,50	1904
P_2	1373,67	1373,67	46,47	2566
P_3	1034,20	1034,20	39,84	1719
P_4	1035,86	1035,86	44,90	1880
P_5	1179,44	1179,44	42,77	2061
P_6	1227,70	1227,70	45,27	2243
P_7	1289,14	1289,14	34,36	1964
P_8	1421,28	1421,28	40,23	2378
P_9	981,87	981,87	48,67	1913
Média			43,18	

Tabela 10 - Resultados da Relaxação Linear – para as instâncias de média dimensão

Instância	**Modelo indexado ao tempo**	**Modelo *strip-packing***	**Desvio (%)**	**Solução ótima**
M_0	2352,14	2352,14	43,32	4150
M_2	2653,85	2653,85	38,79	4336
M_3	2190,79	2190,79	45,56	4024
M_4	2367,02	2367,02	42,31	4103
M_5	2402,24	2402,24	43,89	4281
M_6	2282,00	2282,00	45,94	4221
M_7	2595,17	2595,17	43,25	4573
Média			43,29	

Após a análise das Tabela 9 e Tabela 10, apurou-se que a qualidade dos modelos é semelhante, uma vez que o resultado da relaxação linear é igual em ambos os modelos para todas as instâncias. Neste sentido, concluiu-se também que as formulações dos dois modelos são fracas, pois os resultados da relaxação linear não se aproximam das soluções ótimas, sendo cerca de 43% o desvio médio da solução ótima em ambas as dimensões de teste.

Como a qualidade das formulações dos modelos é semelhante, é importante compará-los em termos de capacidade para encontrar soluções ótimas. Esta comparação foi efetuada para as instâncias de teste de pequena dimensão, com um tempo limite de 3600 segundos. Na Tabela 11 é possível observar os resultados em termos de *gap* para a solução ótima, tempo computacional e função objetivo.

Tabela 11 – Desempenho dos modelos matemáticos para as instâncias de pequena dimensão

	Modelo *strip-packing*			**Modelo indexado ao tempo**		
Instância	*Gap* (%)	Tempo (s)	Solução (min)	*Gap* (%)	Tempo (s)	Solução (min)
P_0	0,00	3,17	1575	26,35	3600,00	1575
P_1	0,00	6,77	1904	8,77	3600,00	1904
P_2	0,00	0,98	2566	0,00	183,17	2566
P_3	0,00	1,01	1719	0,00	2200,28	1719
P_4	0,00	2,66	1880	0,00	1111,03	1880
P_5	0,00	3,73	2061	2,06	3600,00	2061
P_6	0,00	9,17	2243	18,94	3600,00	2297
P_7	0,00	2,28	1964	0,00	593,44	1964
P_8	0,00	4,75	2378	0,00	1526,42	2378
P_9	0,00	6,03	1913	6,45	3600,00	1936
Média	0,00	4,06		6,26	2361,43	

Com a análise dos resultados constatou-se que o modelo *strip-packing* apresenta um desempenho muito superior comparativamente ao modelo indexado ao tempo. Para os problemas de pequena dimensão, o modelo *strip-packing* foi capaz de obter a solução ótima em todas as instâncias com um tempo médio de computação de 4,06 segundos. Enquanto o modelo indexado ao tempo, para o mesmo conjunto de instâncias, encontra a solução ótima em 50% das instâncias com um tempo médio computacional de 2361,43 segundos.

Para os conjuntos de problemas de maiores dimensões, o modelo indexado ao tempo não é capaz de encontrar soluções ao final de 1 hora. Tendo em conta que a qualidade da formulação dos modelos é semelhante e que o modelo *strip-packing* supera bastante o desempenho do modelo indexado ao tempo, o estudo do modelo indexado ao tempo foi abandonado, centrando-se exclusivamente no modelo baseado em *strip-packing*.

4.4. Desempenho do modelo *strip-packing* nas instâncias médias e grandes

Nas Tabela 12 e Tabela 13 são apresentados os resultados do modelo *strip-packing* para as instâncias de dimensões média e grande.

Tabela 12 – Desempenho do modelo *strip-packing* nas instâncias de dimensão média

Instância	***Gap* (%)**	**Tempo (s)**	**Solução (min)**
M_0	0,00	1203,97	4150
M_1	16,76	1800,00	3758
M_2	0,00	1005,14	4336
M_3	0,00	665,30	4024
M_4	0,38	1800,00	4103
M_5	0,72	1800,00	4281
M_6	22,03	1800,00	4294
M_7	8,41	1800,00	4629
M_8	12,85	1800,00	4575
M_9	15,59	1800,00	4481
Média	7,67	1547,44	

Tabela 13 - Desempenho do modelo *strip-packing* nas instâncias de dimensão grande

Instância	*Gap* (%)	Tempo (s)	Solução (min)
G_0	18,07	1800	8054
G_1	22,73	1800	8518
G_2	23,15	1800	7636
G_3	20,73	1800	6850
G_4	21,78	1800	8378
G_5	20,52	1800	8842
G_6	18,86	1800	8123
G_7	23,27	1800	8259
G_8	13,50	1800	8167
G_9	25,35	1800	7349
Média	20,80	1800	

No conjunto de instâncias médias, o modelo matemático foi capaz de atingir a solução ótima em 30% das instâncias. O *gap* médio para a solução ótima foi de 7,67% e em 50% das instâncias o *gap* foi inferior a 0,72%. No conjunto de instâncias de dimensão grande, o modelo matemático não foi capaz de atingir a solução ótima ao final do tempo limite de 1800 segundos. O *gap* médio para a solução ótima foi de 20,80%.

Como o modelo matemático apresenta dificuldades na resolução dos problemas de dimensão média e grande, desenvolveram-se outras estratégias para melhorar o desempenho do modelo matemático.

4.5. Desempenho da heurística matemática

Nesta secção apresentam-se os resultados das diferentes abordagens da heurística matemática com o objetivo de analisar o impacto de cada estratégia na eficiência dos algoritmos utilizados. As nomenclaturas para cada abordagem serão as seguintes:

- WS-CR: mecanismo de *warm-start* com a heurística CR
- WS-TS: mecanismo de *warm-start* com a metaheurística TS
- LB-SCTUR: soluções da heurística SCTUR como *lower bound*
- WS-CR + LB-SCTUR: mecanismo de *warm-start* com a heurística CR + soluções da heurística SCTUR como *lower bound*
- WS-TS + LB-SCTUR: mecanismo de *warm-start* com a metaheurística TS + soluções da heurística SCTUR como *lower bound*

4.5.1. *Warm-start* com heurística CR

A heurística CR desenvolvida foi utilizada para fornecer uma solução válida ao modelo matemático para acelerar o seu processo de atingir a solução ótima. Os resultados são expostos nas Tabela 14, Tabela 15 e Tabela 16.

Tabela 14 – Desempenho do WS-CR nas instâncias de dimensão pequena

Instância	Solução CR (min)	*Gap* (%)	Tempo (s)	Solução Final (min)
P_0	1575	0,00	2,73	1575
P_1	2059	0,00	2,19	1904
P_2	2639	0,00	0,86	2566
P_3	1719	0,00	0,59	1719
P_4	1942	0,00	1,31	1880
P_5	2214	0,00	3,15	2061
P_6	2256	0,00	4,64	2243
P_7	2091	0,00	1,24	1964
P_8	2403	0,00	1,33	2378
P_9	1931	0,00	6,11	1913
Média		0,00	2,42	

Tabela 15 – Desempenho do WS-CR nas instâncias de dimensão média

Instância	Solução CR (min)	*Gap* (%)	Tempo (s)	Solução Final (min)
M_0	4150	0,00	1018,75	4150
M_1	3820	19,18	1800,00	3741
M_2	4527	0,00	250,12	4336
M_3	4050	0,00	555,00	4024
M_4	4169	0,00	1189,00	4103
M_5	4333	0,72	1800,00	4281
M_6	4351	12,82	1800,00	4251
M_7	4650	3,16	1800,00	4578
M_8	4757	14,15	1800,00	4581
M_9	4524	10,19	1800,00	4453
Média		6,02	1381,29	

Tabela 16 - Desempenho do WS-CR nas instâncias de dimensão grande

Instância	Solução CR (min)	*Gap* (%)	Tempo (s)	Solução Final (min)
G_0	7606	10,00	1800	7603
G_1	8417	20,13	1800	8417
G_2	7355	17,02	1800	7251
G_3	6795	17,38	1800	6686
G_4	8418	20,76	1800	8418
G_5	8854	19,41	1800	8818
G_6	7870	16,11	1800	7870
G_7	7924	20,87	1800	7924
G_8	8045	12,10	1800	8045
G_9	6995	17,19	1800	6934
Média		17,10	1800	

Para as instâncias de pequena dimensão, o *warm-start* com a heurística CR obtém a solução ótima em todas as instâncias, com um tempo médio de 2,42 segundos. Para as instâncias de dimensão média, esta estratégia encontra a solução ótima em 40% dos testes. Em 60% dos testes o *gap* foi inferior a 3,17%. O *gap* médio foi de 6,02% com um tempo médio de 1381,29 segundos. Por fim,

para as instâncias de dimensão grande, esta abordagem não consegue encontrar a solução ótima em nenhuma das instâncias ao final de 1800 segundos, sendo o *gap* médio de 17,10%.

4.5.2. *Warm-start* com *Tabu Search*

Para melhorar as soluções obtidas pela heurística CR, foi desenvolvida a metaheurística *Tabu Search*. As soluções obtidas pelo *Tabu Search* foram fornecidas ao modelo matemático como solução inicial. Os resultados para as instâncias de pequena, média e grande dimensão encontram-se nas Tabela 17, Tabela 18 e Tabela 19.

Tabela 17 – Desempenho do WS-TS nas instâncias de dimensão pequena

Instância	Solução TS (min)	*Gap* (%)	Tempo (s)	Solução Final (min)
P_0	1575	0,00	2,73	1575
P_1	1904	0,00	3,47	1904
P_2	2566	0,00	1,78	2566
P_3	1719	0,00	0,56	1719
P_4	1880	0,00	1,29	1880
P_5	2061	0,00	2,39	2061
P_6	2256	0,00	4,64	2243
P_7	1964	0,00	1,23	1964
P_8	2379	0,00	0,94	2378
P_9	1913	0,00	5,69	1913
Média		0,00	2,47	

Tabela 18 – Desempenho do WS-TS nas instâncias de dimensão média

Instância	Solução TS (min)	*Gap* (%)	Tempo (s)	Solução Final (min)
M_0	4150	0,00	1018,75	4150
M_1	3757	13,62	1800,00	3739
M_2	4336	0,00	247,21	4336
M_3	4050	0,00	553,45	4024
M_4	4103	0,00	941,23	4103
M_5	4281	0,00	1714,3	4281
M_6	4221	11,79	1800,00	4221
M_7	4595	1,14	1800,00	4573
M_8	4561	10,78	1800,00	4561
M_9	4457	9,44	1800,00	4453
Média		4,68	1347,49	

Tabela 19 - Desempenho do WS-TS nas instâncias de dimensão grande

Instância	Solução TS (min)	*Gap* (%)	Tempo (s)	Solução Final (min)
G_0	7606	10,00	1800	7603
G_1	8417	20,13	1800	8417
G_2	7249	15,12	1800	7249
G_3	6686	17,38	1800	6686
G_4	8260	18,00	1800	8255
G_5	8762	19,28	1800	8762
G_6	7870	16,11	1800	7870
G_7	7839	19,43	1800	7839
G_8	8020	10,65	1800	8020
G_9	6995	17,19	1800	6934
Média		16,33	1800	

Para as instâncias de pequena dimensão, o WS-TS obtém a solução ótima em todas as instâncias com um tempo médio de 2,47 segundos. Para as instâncias de dimensão média, esta estratégia *warm-start* permite encontrar a solução ótima em 50% dos testes. O *gap* médio foi de 4,68% com um tempo médio de 1347,49 segundos. Em 60% dos testes o *gap* é inferior a 1,15%. Por fim, para as instâncias de dimensão grande, esta abordagem não consegue encontrar a solução ótima em nenhuma das instâncias ao final de 1800 segundos, sendo o *gap* médio de 16,33%.

4.5.3. Lower bound

Nas Tabela 20, Tabela 21 e Tabela 22 apresenta-se a comparação entre os valores da relaxação linear e os resultados obtidos pela heurística SCTUR. De salientar que o desvio para a solução ótima apenas surge nas instâncias de pequena dimensão e nas instâncias de média dimensão em que se obteve a solução ótima. Nas restantes instâncias essa comparação não é efetuada por não se conhecer o valor da solução ótima.

Tabela 20 - Resultados da relaxação linear e da heurística SCTUR nas instâncias pequenas

Instância	Relaxação Linear		Heurística SCTUR		Solução ótima
	Solução	Desvio (%)	Solução	Desvio (%)	
P_0	790,00	49,84	1538	2,35	1575
P_1	1152,00	39,50	1865	2,05	1904
P_2	1373,67	46,47	2566	0,00	2566
P_3	1034,20	39,84	1601	6,86	1719
P_4	1035,86	44,90	1827	2,82	1880
P_5	1179,44	42,77	1935	6,11	2061
P_6	1227,70	45,27	2117	5,62	2243
P_7	1289,14	34,36	1962	0,10	1964
P_8	1421,28	40,23	2256	5,13	2378
P_9	981,87	48,67	1816	5,07	1913
Média		43,18		3,61	

Tabela 21 - Resultados da relaxação linear e da heurística SCTUR nas instâncias médias

Instância	Relaxação Linear		Heurística SCTUR		Solução ótima
	Solução	Desvio (%)	Solução	Desvio (%)	
M_0	2352,14	43,32	3960	4,58	4150
M_1	1932,72	-	3599	-	-
M_2	2653,85	38,79	4198	3,18	4336
M_3	2190,79	45,56	3782	6,01	4024
M_4	2367,02	42,31	3886	5,29	4103
M_5	2402,24	43,89	3986	6,89	4281
M_6	2282,00	45,94	4083	3,27	4221
M_7	2595,17	43,25	4279	6,43	4573
M_8	2613,94	-	4340	-	-
M_9	2516,93	-	4278	-	-
Média		43,29		5,09	

Tabela 22 - Resultados da relaxação linear e da heurística SCTUR nas instâncias grandes

Instância	Relaxação Linear		Heurística SCTUR		Solução ótima
	Solução	Desvio (%)	Solução	Desvio (%)	
G_0	4941,04	-	7323	-	-
G_1	4938,05	-	7672	-	-
G_2	4538,58	-	6592	-	-
G_3	4234,02	-	6032	-	-
G_4	5184,76	-	7612	-	-
G_5	5213,01	-	7991	-	-
G_6	4779,00	-	7530	-	-
G_7	4817,95	-	6900	-	-
G_8	5036,47	-	7645	-	-
G_9	4124,17	-	6179	-	-

Após a análise das tabelas anteriores é claro que a heurística SCTUR desenvolvida fornece bons resultados, sendo positiva a sua utilização como *lower bound*. Nas instâncias pequenas, entre os valores da relaxação linear e os resultados da heurística SCTUR, verifica-se uma redução do desvio médio para a solução ótima de 43,18% para 3,61%. Enquanto nas instâncias médias, a redução do desvio médio foi de 43,29% para 5,09%. Nos problemas de grande dimensão, não é possível apurar o desvio em relação à solução ótima, no entanto, os resultados da heurística são bastante melhores que os valores da relaxação linear. De referir também que os tempos computacionais da heurística SCTUR são inferiores a 1 segundo.

Face ao bom desempenho da heurística SCTUR, tornou-se importante avaliar o impacto da estratégia LB-SCTUR (soluções da heurística SCTUR como *lower bound*) na eficiência do modelo matemático. Os testes efetuados encontram-se nas Tabela 23, Tabela 24 e Tabela 25.

Tabela 23 - Desempenho da estratégia LB-SCTUR nas instâncias de dimensão pequena

Instância	Solução SCTUR	*Gap* (%)	Tempo (s)	Solução Final
P_0	1538	0,00	5,22	1575
P_1	1865	0,00	2,89	1904
P_2	2566	0,00	2,19	2566
P_3	1601	0,00	3,39	1719
P_4	1827	0,00	6,30	1880
P_5	1935	0,00	5,38	2061
P_6	2117	0,00	8,12	2243
P_7	1962	0,00	2,22	1964
P_8	2256	0,00	2,69	2378
P_9	1816	0,00	6,66	1913
Média		0,00	4,51	

Tabela 24 – Desempenho da estratégia LB-SCTUR nas instâncias de dimensão média

Instância	Solução SCTUR	*Gap* (%)	Tempo (s)	Solução Final
M_0	3960	4,58	1800,00	4150
M_1	3599	4,21	1800,00	3757
M_2	4198	3,18	1800,00	4336
M_3	3782	0,00	715,64	4024
M_4	3886	5,43	1800,00	4109
M_5	3986	0,72	1800,00	4281
M_6	4083	4,31	1800,00	4267
M_7	4279	4,04	1800,00	4634
M_8	4340	4,64	1800,00	4551
M_9	4278	4,59	1800,00	4484
Média		3,57	1691,56	

Tabela 25 - Desempenho da estratégia LB-SCTUR nas instâncias de dimensão grande

Instância	Solução SCTUR	*Gap* (%)	Tempo (s)	Solução Final
G_0	7323	6,32	1800	7818
G_1	7672	14,16	1800	8938
G_2	6592	12,57	1800	7540
G_3	6032	10,49	1800	6739
G_4	7612	10,48	1800	8503
G_5	7991	10,40	1800	8919
G_6	7530	4,70	1800	7901
G_7	6900	14,90	1800	8108
G_8	7645	6,25	1800	8155
G_9	6179	11,03	1800	6945
Média		10,13	1800	

Tal como previsto, utilizar as soluções da heurística SCTUR como *lower bound* tem um impacto positivo. De facto, nas instâncias de pequena dimensão atinge-se a solução ótima em todos os testes, com um tempo computacional médio de 4,51 segundos. No conjunto de instâncias de dimensão média, obteve-se a solução ótima apenas num teste. O *gap* médio foi de 3,57% com um

tempo computacional médio de 1691,56 segundos. Nas instâncias de dimensão grande, ainda não se obtém nenhuma solução ótima, sendo o *gap* médio de 10,13%.

4.5.4. Warm-start e *lower bound*

Neste subcapítulo são apresentados os resultados de duas abordagens da heurística matemática que combinam o método *warm-start* com a estratégia LB-SCTUR. Na primeira abordagem usou-se a heurística CR (WS-CR) e na segunda abordagem recorreu-se ao *Tabu Search* (WS-TS).

***Warm-start* com a heurística CR e *lower bound* com a heurística SCTUR**

Nas Tabela 26, Tabela 27 e Tabela 28 é possível ver os resultados da abordagem composta por WS-CR e LB-SCTUR.

Tabela 26 – Desempenho da abordagem WS-CR + LB-SCTUR nas instâncias pequenas

Instância	*Gap* (%)	Tempo (s)	Solução (min)
P_0	0,00	5,08	1575
P_1	0,00	1,88	1904
P_2	0,00	0,52	2566
P_3	0,00	1,73	1719
P_4	0,00	1,02	1880
P_5	0,00	3,01	2061
P_6	0,00	4,56	2243
P_7	0,00	1,31	1964
P_8	0,00	1,33	2378
P_9	0,00	6,10	1913
Média	0,00	2,65	

Tabela 27 – Desempenho da abordagem WS-CR + LB-SCTUR nas instâncias médias

Instância	*Gap* (%)	Tempo (s)	Solução (min)
M_0	0,00	480,12	4150
M_1	4,13	1800,00	3741
M_2	0,00	329,56	4336
M_3	0,00	504,97	4024
M_4	0,00	1492,61	4103
M_5	0,00	1541,14	4281
M_6	3,73	1800,00	4241
M_7	1,25	1800,00	4578
M_8	4,64	1800,00	4551
M_9	3,97	1800,00	4455
Média	1,77	1334,84	

Tabela 28 - Desempenho da abordagem WS-CR + LB-SCTUR nas instâncias grandes

Instância	*Gap* (%)	Tempo (s)	Solução (min)
G_0	3,59	1800	7596
G_1	8,85	1800	8417
G_2	9,06	1800	7249
G_3	9,78	1800	6686
G_4	9,57	1800	8418
G_5	8,43	1800	8727
G_6	4,32	1800	7870
G_7	12,37	1800	7874
G_8	4,97	1800	8045
G_9	10,71	1800	6920
Média	8,17	1800	

Para as instâncias de pequena dimensão, esta abordagem da heurística matemática obtém a solução ótima em todas as instâncias com um tempo médio de 2,65 segundos. Para as instâncias de dimensão média, esta estratégia permite encontrar a solução ótima em 50% dos testes. O *gap* médio foi de 1,77% com um tempo médio de 1334,84 segundos. Por fim, nas instâncias de dimensão grande não se obtém solução ótima em nenhum dos testes ao final de 1800 segundos, sendo o *gap* médio de 8,17%.

Warm-start **com a metaheurística TS e** ***lower bound*** **com a heurística SCTUR**

Os resultados da abordagem composta por WS-TS e LB-SCTUR surgem nas Tabela 29, Tabela 30 e Tabela 31.

Tabela 29 – Desempenho da abordagem WS-TS + LB-SCTUR nas instâncias pequenas

Instância	*Gap* (%)	Tempo (s)	Solução (min)
P_0	0,00	5,41	1575
P_1	0,00	2,84	1904
P_2	0,00	0,06	2566
P_3	0,00	1,66	1719
P_4	0,00	0,98	1880
P_5	0,00	2,29	2061
P_6	0,00	4,12	2243
P_7	0,00	1,53	1964
P_8	0,00	0,84	2378
P_9	0,00	5,45	1913
Média	0,00	2,52	

Tabela 30 – Desempenho da abordagem WS-TS + LB-SCTUR nas instâncias médias

Instância	*Gap* (%)	Tempo (s)	Solução (min)
M_0	0,00	480,12	4150
M_1	0,54	1800,00	3739
M_2	0,00	227,25	4336
M_3	0,00	501,09	4024
M_4	0,00	704,41	4103
M_5	0,00	945,24	4281
M_6	0,21	1800,00	4221
M_7	0,81	1800,00	4573
M_8	2,09	1800,00	4551
M_9	1,12	1800,00	4453
Média	0,48	1185,81	

Tabela 31 - Desempenho da abordagem WS-TS + LB-SCTUR nas instâncias grandes

Instância	*Gap* (%)	Tempo (s)	Solução (min)
G_0	0,80	1800	7596
G_1	4,72	1800	8417
G_2	3,61	1800	6830
G_3	6,25	1800	6686
G_4	4,57	1800	8255
G_5	5,13	1800	8727
G_6	0,98	1800	7870
G_7	8,12	1800	7839
G_8	2,03	1800	8020
G_9	6,20	1800	6920
Média	4,24	1800	

Para as instâncias de pequena dimensão, esta abordagem obtém a solução ótima em todas as instâncias, com um tempo médio de 2,52 segundos. Para as instâncias de dimensão média, permite encontrar a solução ótima em 50% dos testes. O *gap* médio foi de 0,48% com um tempo médio de 1334,84 segundos, sendo que em 80% destas instâncias o *gap* é inferior a 0,82%. Por fim, para as instâncias de dimensão grande, a estratégia WS-TS + LB-SCTUR não consegue encontrar a solução ótima em nenhuma das instâncias ao final de 1800 segundos, sendo o *gap* médio de 4,24%.

4.6. Comparação das diferentes abordagens

O objetivo desta secção consiste em comparar os resultados de todas as abordagens da heurística matemática. Nas Tabela 32, Tabela 33 e Tabela 34 são apresentados os tempos computacionais e os *gaps* médios para a solução ótima, bem como os ganhos de cada abordagem em relação ao desempenho do modelo matemático.

Tabela 32 - Comparação do desempenho das diferentes abordagens – instâncias pequenas

Abordagem	Tempo médio (s)	*Gap* médio (%)	Ganho médio (%)	Soluções ótimas (%)
Modelo matemático	4,06	0,00	-	100
WS-CR	2,42	0,00	40,44	100
WS-TS	2,47	0,00	39,04	100
LB-SCTUR	4,51	0,00	-11,12	100
WS-CR + LB-SCTUR	2,65	0,00	34,55	100
WS-TS + LB-SCTUR	2,52	0,00	37,90	100

Tabela 33 - Comparação do desempenho das diferentes abordagens – instâncias médias

Abordagem	Tempo médio (s)	*Gap* médio (%)	Ganho médio (%)		Soluções ótimas (%)
			Tempo	*Gap*	
Modelo matemático	1547,44	7,67	-	-	30
WS-CR	1381,29	6,02	10,74	21,53	40
WS-TS	1347,49	4,68	12,92	39,05	50
LB-SCTUR	1691,56	3,57	-9,31	53,48	10
WS-CR + LB-SCTUR	1334,84	1,77	13,74	76,91	50
WS-TS + LB-SCTUR	1185,81	0,48	23,37	93,78	50

Tabela 34 - Comparação do desempenho das diferentes abordagens – instâncias grandes

Abordagem	*Gap* médio (%)	Ganho médio (%)	Soluções ótimas (%)
Modelo matemático	20,80	-	0
WS-CR	17,10	17,79	0
WS-TS	16,33	21,48	0
LB-SCTUR	10,13	51,29	0
WS-CR + LB-SCTUR	8,17	60,74	0
WS-TS + LB-SCTUR	4,24	79,61	0

Nas instâncias de pequena dimensão obtém-se sempre a solução ótima, por isso, a percentagem de soluções ótimas não varia nem existe ganhos em termos de *gap*. Assim, regista-se impacto apenas no tempo computacional com uma redução em quase todas as abordagens da heurística matemática, com exceção da estratégia LB-SCTUR. O maior ganho em relação ao desempenho do modelo matemático é de 40,44% na estratégia WS-CR.

Nas instâncias de dimensão média faz sentido avaliar os ganhos em termos de tempo computacional e de *gap*. Ao nível do tempo computacional ocorre a diminuição do valor médio em quase todas as abordagens, com exceção da estratégia LB-SCTUR. A melhor abordagem foi o WS-TS com LB-SCTUR responsável por um ganho de 23,37% em relação ao modelo matemático. Em relação ao *gap* médio, ocorre melhoria em todas as abordagens, sendo a melhor a que combina WS-TS e LB-SCTUR com um ganho de 93,78%.

No conjunto de problemas de dimensão grande nunca se obtém a solução ótima e o tempo usado em cada teste foi de 1800 segundos. No que diz respeito ao *gap* para a solução ótima, obtiveram-se melhorias em todas as abordagens, destacando-se a que combina WS-TS e LB-SCTUR com um ganho de 79,61% comparativamente ao modelo matemático.

Face ao exposto, é possível tirar algumas conclusões gerais. A abordagem apenas com LB-SCTUR, em termos de tempo computacional, só apresenta um impacto positivo nas instâncias de dimensão grande. As estratégias com *warm-start* têm sempre um impacto positivo. Por fim, concluiu-se também que uma abordagem da heurística matemática mais composta (*warm-start* e LB-SCTUR em simultâneo) tem realmente impacto em instâncias de maiores dimensões, onde o modelo matemático apresenta dificuldades.

4.7. Desempenho do *Tabu Search*

Nas Tabela 35, Tabela 36 e Tabela 37 apuraram-se os ganhos que a metaheurística TS alcança comparativamente às soluções da heurística CR. Além disso, considerou-se oportuno perceber a capacidade do TS em chegar à solução ótima. De referir que para as instâncias em que não se obteve a solução ótima, apresenta-se o *gap* entre a solução do TS e a solução da heurística SCTUR (*lower bound*), calculado através da seguinte expressão.

$$gap\ (\%) = \frac{solução_{TS} - solução_{SCTUR}}{solução_{SCTUR}} \times 100 \tag{69}$$

Tabela 35 – Comparação das soluções do TS e da heurística CR nas instâncias pequenas

Instância	Solução CR	Solução TS	Ganho (%)	*Gap* (%) - TS	Solução ótima
P_0	1575	1575	0,00	0,00	1575
P_1	2059	1904	7,53	0,00	1904
P_2	2639	2566	2,77	0,00	2566
P_3	1719	1719	0,00	0,00	1719
P_4	1942	1880	3,19	0,00	1880
P_5	2214	2061	6,91	0,00	2061
P_6	2256	2256	0,00	0,58	2243
P_7	2091	1964	6,07	0,00	1964
P_8	2403	2379	1,00	0,04	2378
P_9	1931	1913	0,93	0,00	1913
Média			2,84	0,06	

Tabela 36 - Comparação das soluções do TS e da heurística CR nas instâncias médias

Instância	Solução CR	Solução TS	Ganho (%)	*Gap* (%) - TS	Solução ótima
M_0	4150	4150	0,00	0,00	4150
M_1	3820	3757	1,65	4,39	-
M_2	4527	4336	4,22	0,00	4336
M_3	4050	4050	0,00	0,65	4024
M_4	4169	4103	1,58	0,00	4103
M_5	4333	4281	1,20	0,00	4281
M_6	4351	4221	2,99	0,00	4221
M_7	4650	4595	1,18	0,48	4573
M_8	4757	4561	4,12	5,09	-
M_9	4524	4457	1,48	4,18	-
Média			1,84	1,48	

Tabela 37 - Comparação das soluções do TS e da heurística CR nas instâncias grandes

Instância	Solução CR	Solução TS	Ganho (%)	*Gap* (%) - TS	Solução ótima
G_0	7606	7606	0,00	3,86	-
G_1	8417	8417	0,00	9,71	-
G_2	7355	7249	1,44	9,97	-
G_3	6795	6686	1,60	10,84	-
G_4	8418	8260	1,88	8,51	-
G_5	8854	8762	1,04	9,65	-
G_6	7870	7870	0,00	4,52	-
G_7	7924	7839	1,07	13,61	-
G_8	8045	8020	0,31	4,91	-
G_9	6995	6995	0,00	13,21	-
Média			0,73	8,88	

Nas instâncias de pequena dimensão, o TS consegue melhorar as soluções da heurística CR em 70% das instâncias, com um ganho médio de 2,84%. Nas instâncias médias, ocorre a melhoria das soluções em 80% dos testes com um ganho médio de 1,84%. Nas instâncias grandes, o TS melhora as soluções em 60% das instâncias com um ganho médio de 0,73%.

Relativamente à capacidade do TS encontrar a solução ótima, nas instâncias pequenas isso acontece em 80% dos testes, sendo que nos restantes no pior dos casos o *gap* foi de 0,58%. Para os problemas de dimensão média, o TS obteve 50% de soluções ótimas e nos restantes casos, na pior das hipóteses um *gap* para solução da heurística SCTUR de 5,09%. Nas instâncias grandes não se obtém a solução ótima, no entanto, o *gap* médio para a solução da heurística SCTUR foi de 8,88%.

Após verificar o bom desempenho do *Tabu Search* nas instâncias de dimensões pequena, média e grande, considerou-se oportuno avaliar o mesmo em instâncias de dimensão maior. Neste sentido, aplicou-se o *Tabu Search* ao conjunto de problemas de dimensão extra grande. Os resultados da metaheurística em termos de tempo computacional e solução obtida para os vários conjuntos de instâncias são apresentados nas Tabela 38, Tabela 39, Tabela 40, Tabela 41 e Tabela 42.

Tabela 38 – Desempenho do TS nas instâncias de dimensão pequena

Instância	Tempo (s)	Solução (min)
P_0	9,27	1575
P_1	9,40	1904
P_2	8,48	2566
P_3	10,13	1719
P_4	14,24	1892
P_5	9,59	2214
P_6	9,51	2256
P_7	13,26	1964
P_8	11,97	2403
P_9	13,49	1915
Média	10,93	

Tabela 39 – Desempenho do TS nas instâncias de dimensão média

Instância	Tempo (s)	Solução (min)
M_0	24,64	4150
M_1	27,41	3791
M_2	42,71	4341
M_3	25,71	4050
M_4	27,09	4103
M_5	24,51	4281
M_6	25,34	4226
M_7	24,67	4607
M_8	26,86	4641
M_9	41,62	4464
Média	29,06	

Tabela 40 – Desempenho do TS nas instâncias de dimensão grande

Instância	Tempo (s)	Solução (min)
G_0	55,36	7606
G_1	62,37	8417
G_2	59,55	7249
G_3	60,20	6686
G_4	77,24	8260
G_5	63,42	8762
G_6	73,95	7870
G_7	63,41	7839
G_8	57,38	8020
G_9	71,51	6995
Média	64,44	7770,4

Tabela 41 – Desempenho do TS nas instâncias de dimensão extra grande I

Instância	Tempo (s)	Solução (min)
EGI_0	112,90	15492
EGI_1	112,16	14190
EGI_2	117,58	16381
EGI_3	128,83	15434
EGI_4	129,76	14842
EGI_5	117,73	15456
EGI_6	213,83	16427
EGI_7	417,31	15522
EGI_8	126,34	17000
EGI_9	111,98	14444
Média	158,84	

Tabela 42 – Desempenho do TS nas instâncias de dimensão extra grande II

Instância	Tempo (s)	Solução (min)
EGII_0	285,59	19768
EGII_1	520,95	24205
EGII_2	302,02	20258
EGII_3	286,64	20924
EGII_4	495,28	18467
EGII_5	500,48	19740
EGII_6	306,17	21798
EGII_7	287,59	20281
EGII_8	324,63	19761
EGII_9	267,09	20237
Média	357,64	

Para as instâncias de dimensão pequena, média e grande os tempos computacionais do TS são bastante razoáveis, sendo os valores médios de 10,93; 29,06 e 64,44 segundos, respetivamente. Para o conjunto de problemas extra grande I e II, os tempos médios são de 158,84 e 357,64 segundos, respetivamente. A dimensão extra grande II é identificada como o limite computacional do TS, isto é, o TS apresenta uma resposta computacional razoável em problemas com até 32 máquinas e 100 tarefas.

Na Tabela 43 é apresentada uma comparação entre o TS proposto neste projeto e o TS do artigo de referência Bektur & Saraç (2019), salientando que os problemas tratados são diferentes.

Tabela 43 - Comparação entre o TS proposto e o TS de Bektur & Saraç (2019)

	TS - Bektur & Saraç (2019)	TS proposto
Máquinas	Não relacionadas	Dedicadas
Recursos	*Setup*	*Setup* e processamento
Extra	Restrições de elegibilidade	-
Nº máquinas	10	32
Nº tarefas	100	100
Dimensão geral	10 x 100	32 x 100
Tempo médio (s)	651,07	357,64

O estudo de Bektur & Saraç (2019) aborda máquinas paralelas não relacionadas o que representa mais combinações possíveis (tarefa-máquina) comparativamente a máquinas dedicadas. No entanto, como existe restrições de elegibilidade da máquina, essas combinações diminuem. Nesse estudo só se considera recursos de *setup* e o algoritmo apresenta um tempo médio de computação de 651 segundos para instâncias com 10 máquinas e 100 tarefas. No estudo de Bektur & Saraç (2019) as instâncias têm mais tarefas por máquina, contudo, no problema do presente projeto as restrições de recursos são mais exigentes devido a se considerar mais um tipo de recursos e a um maior número de máquinas. Além disso, nos testes do TS proposto a dimensão geral do problema é superior (32x100). Tendo em conta este balanceamento, considera-se que o TS proposto neste projeto apresenta um desempenho superior ao TS de Bektur & Saraç (2019), o que se reflete nos tempos computacionais.

5. CONCLUSÃO

Tendo em conta a competitividade e exigência do mercado, os sistemas de apoio à decisão podem ter um papel muito importante no sucesso das empresas. Assim sendo, a solução proposta pretende minimizar as ineficiências existentes ao nível do chão de fábrica, possibilitando a investigação do impacto desta estratégia no desempenho global das empresas.

O presente projeto surge a partir do desafio lançado pela INPLAS ao centro de investigação CPLOT com o objetivo de melhorar o planeamento da produção das suas fábricas que consiste num problema de máquinas paralelas dedicadas com *setups* dependentes da sequência de famílias e recursos adicionais.

Neste sentido, foram definidas duas questões de investigação. Primeiramente, era necessário apurar se o modelo *strip-packing* superava o modelo indexado ao tempo. Além disso, pretendia-se investigar se as heurísticas matemáticas teriam impacto na eficiência dos algoritmos utilizados. Face ao problema apresentado e em resposta às questões de investigação, foi estabelecido um objetivo geral que consiste no desenvolvimento de um sistema de apoio à decisão para o problema da empresa.

O estudo desenvolvido possui várias contribuições ao nível dos modelos matemáticos e da metaheurística desenvolvida. No que diz respeito aos modelos, foi feita a adaptação de dois modelos matemáticos gerais ao problema de máquinas paralelas dedicadas. Nesta adaptação foi necessário efetuar simplificações ao nível das variáveis e dos parâmetros relacionados com a alocação das tarefas às máquinas. Tendo em conta que os sistemas de produção são contínuos, introduziu-se uma configuração inicial das máquinas. Com esta inclusão garantiu-se que as tarefas que se encontravam em processamento no final do turno anterior, seriam as primeiras a terminar o processamento no turno seguinte. Além disso, foi proposta uma nova função objetivo que pretende a minimização da soma dos *makespan* de todas as máquinas. Esta função objetivo conduz a soluções compactas, minimiza o tempo de conclusão de cada máquina e permite a minimização dos tempos ociosos e do número de *setups*. Por fim, os modelos matemáticos de referência não permitem a resolução de problemas com *setups* dependentes da sequência de famílias. Sendo assim, foi efetuada uma generalização destes modelos, o que já permite a resolução de problemas com *setups* dependentes da sequência de famílias.

Em relação à metaheurística TS, a fase construtiva do algoritmo de referência não estava preparada para o problema abordado neste estudo. Assim sendo, foi necessário adaptar o algoritmo à realidade de máquinas paralelas dedicadas e com *setups* dependentes da sequência de famílias. Além disso, o mecanismo de reparação original não incluiu a componente dos recursos de processamento, tendo sido necessária essa inclusão neste projeto.

Na fase experimental, através da comparação entre os dois modelos matemáticos, concluiu-se que a qualidade das formulações é idêntica, no entanto, o modelo *strip-packing* supera muito o desempenho do modelo indexado ao tempo na resolução deste tipo de problema. Após esta conclusão, o estudo prosseguiu apenas no modelo *strip-packing*. Este modelo fornece a solução ótima em apenas alguns segundos para todos os problemas de pequena dimensão, sendo também capaz de encontrar soluções ótimas em 30% das instâncias de dimensão média. Para o conjunto médio de instâncias, o *gap* médio obtido foi de 7,67%. Para o conjunto de instâncias de dimensão

grande, o modelo matemático apresenta dificuldades em encontrar a solução ótima, alcançando um *gap* médio de 20,80%.

Face à dificuldade do modelo matemático em instâncias de maiores dimensões, foi desenvolvida uma heurística matemática. A heurística matemática é composta por diferentes abordagens que permitem reduzir o espaço de soluções a analisar pelo modelo matemático. Numa primeira fase, implementou-se uma estratégia de *warm-start* com o objetivo de fornecer uma solução inicial válida e um *upper bound* ao modelo matemático. Na estratégia de *warm-start*, testaram-se dois métodos heurísticos a alimentar o modelo matemático: a heurística construtiva CR e a metaheurística *Tabu Search*. Para reduzir o espaço de soluções inferiormente, desenvolveu-se a heurística SCTUR que gera soluções que funcionam como *lower bound*.

A utilização da heurística SCTUR revelou-se muito vantajosa, uma vez que as soluções obtidas são bastante mais próximas da solução ótima do que os valores da relaxação linear. De facto, a heurística SCTUR gera soluções com um desvio médio da solução ótima de 3,61% e 5,09% para as instâncias pequenas e médias, respetivamente.

A metaheurística TS mostrou capacidade em melhorar as soluções fornecidas pela heurística construtiva CR em 70%, 80% e 60% dos testes nas instâncias pequenas, médias e grandes, respetivamente. Além disso, o TS obtém soluções ótimas em 80% das instâncias pequenas. Nas instâncias médias, o TS obtém a solução ótima em 50% dos testes e nos restantes casos, na pior das hipóteses um *gap* para solução da heurística SCTUR de 5,09%. Nas instâncias grandes não se obtém a solução ótima, no entanto, o *gap* médio para a solução da heurística SCTUR foi de 8,88%.

Também foram testados os limites computacionais da metaheurística TS, onde se concluiu que a sua resposta computacional é razoável até ao conjunto de problemas extra grande II (32 máquinas e 100 tarefas). Comparativamente ao TS do estudo de Bektur & Saraç (2019), o TS proposto neste projeto apresentou um desempenho superior, o que se reflete nos tempos computacionais.

As diferentes abordagens da heurística matemática apresentaram resultados promissores na resolução deste problema complexo, destacando-se a estratégia que combina WS-TS e LB-SCTUR. Esta estratégia permite chegar à solução ótima em 50% das instâncias de dimensão média, com um *gap* médio de 0,48%, o que representa um ganho de 93,78% comparativamente ao desempenho do modelo matemático. Além disso, para as instâncias de dimensão grande obtém um *gap* médio de 4,24%, o que representa um ganho de 79,61% comparativamente ao desempenho do modelo matemático. Face ao exposto, é possível concluir que as estratégias da heurística matemática têm realmente impacto nas instâncias de maiores dimensões em que o modelo matemático apresenta dificuldades.

Para terminar, é importante referir que este trabalho foi apresentação na 32ª edição da conferência internacional *Flexible Automation and Intelligent Manufacturing* e aceite para publicação em revista indexada. O artigo desenvolvido e publicado encontra-se no Anexo A.

5.1. Limitações e investigação futura

Como trabalhos futuros pode ser vantajoso apostar na melhoria da heurística SCTUR com o objetivo de reduzir ainda mais o espaço de soluções a analisar. Além disso, poder-se-á apostar numa metaheurística com uma fase construtiva mais sólida, o que poderá melhorar as soluções fornecidas ao modelo matemático. Por fim, também poderá ser enriquecedor desenvolver outro

tipo de abordagens da heurística matemática em que o modelo matemático forneça informações que melhorem as soluções devolvidas pela metaheurística. O que se está a propor é a troca de informação entre o modelo matemático e os métodos heurísticos nos dois sentidos.

No que a dificuldades diz respeito, o processo de investigação é bastante complexo, uma vez que é difícil prever os resultados obtidos por uma determinada abordagem. Na verdade, em investigação pode se desenvolver um caminho de experimentação e no final, tendo em conta os resultados, ser necessário abandonar esse caminho e recomeçar uma nova estratégia. De referir concretamente que o mais desafiante foi o desenvolvimento da heurística matemática para um problema tão complexo como a programação de máquinas paralelas dedicadas com *setups* dependentes da sequência de famílias e recursos adicionais.

REFERÊNCIAS BIBLIOGRÁFICAS

Abreu, A. P. de, & Fuchigami, H. Y. (2022). An efficiency and robustness analysis of warm-start mathematical models for idle and waiting times optimization in the flow shop. *Computers & Industrial Engineering*, *166*, 107976. https://doi.org/10.1016/j.cie.2022.107976

Afzalirad, M., & Rezaeian, J. (2016). Resource-constrained unrelated parallel machine scheduling problem with sequence dependent setup times, precedence constraints and machine eligibility restrictions. *Computers & Industrial Engineering*, *98*, 40–52. https://doi.org/10.1016/j.cie.2016.05.020

Allahverdi, A., Gupta, J. N. D., & Aldowaisan, T. (1999). A review of scheduling research involving setup considerations. *Omega*, *27*(2), 219–239. https://doi.org/10.1016/S0305-0483(98)00042-5

Allahverdi, A., Ng, C. T., Cheng, T. C. E., & Kovalyov, M. Y. (2008). A survey of scheduling problems with setup times or costs. *European Journal of Operational Research*, *187*(3), 985–1032. https://doi.org/10.1016/J.EJOR.2006.06.060

Applegate, D., Bixby, R., Sek, V., Atal, C., & Cook, W. (1998). On the Solution of Traveling Salesman Problems. *Documenta Mathematica*, *Extra Volume ICM III*, 645–656.

Avalos-Rosales, O., Angel-Bello, F., & Alvarez, A. (2015). Efficient metaheuristic algorithm and re-formulations for the unrelated parallel machine scheduling problem with sequence and machine-dependent setup times. *The International Journal of Advanced Manufacturing Technology*, *76*(9–12), 1705–1718. https://doi.org/10.1007/S00170-014-6390-6

Bektur, G., & Saraç, T. (2019). A mathematical model and heuristic algorithms for an unrelated parallel machine scheduling problem with sequence-dependent setup times, machine eligibility restrictions and a common server. *Computers & Operations Research*, *103*, 46–63. https://doi.org/10.1016/j.cor.2018.10.010

Bilge, Ü., Kiraç, F., Kurtulan, M., & Pekgün, P. (2004). A tabu search algorithm for parallel machine total tardiness problem. *Computers and Operations Research*, *31*(3), 397–414. https://doi.org/10.1016/S0305-0548(02)00198-3

Bitar, A., Dauzère-Pérès, S., & Yugma, C. (2021). Unrelated parallel machine scheduling with new criteria: Complexity and models. *Computers & Operations Research*, *132*, 105291. https://doi.org/10.1016/j.cor.2021.105291

Błażewicz, J., Brauner, N., & Finke, G. (2004). Scheduling with discrete resource constraints. *Handbook of Scheduling: Algorithms, Models, and Performance Analysis*.

Błażewicz, J., Kubiak, W., Röck, H., & Szwarcfiter, J. (1987). Minimizing mean flow-time with parallel processors and resource constraints. *Acta Informatica*, *24*(5), 513–524. https://doi.org/10.1007/BF00263292

Blazewicz, J., Lenstra, J. K., & Kan, A. H. G. R. (1983). Scheduling subject to resource constraints: classification and complexity. *Discrete Applied Mathematics*, *5*(1), 11–24. https://doi.org/10.1016/0166-218X(83)90012-4

Blazewicz, Jacek, Ecker, K. H., Pesch, E., Schmidt, G., Sterna, M., & Weglarz, J. (2019). *Handbook on Scheduling*. Springer International Publishing. https://doi.org/10.1007/978-3-319-99849-7

Boschetti, M. A., Maniezzo, V., Roffilli, M., & Bolufé Röhler, A. (2009). Matheuristics: Optimization, Simulation and Control. Em *Hybrid Metaheuristics: 6th International Workshop* (Vol. 5818, pp. 171–177). Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-04918-7_13

Bozorgirad, M. A., & Logendran, R. (2012). Sequence-dependent group scheduling problem on unrelated-parallel machines. *Expert Systems with Applications*, *39*(10), 9021–9030. https://doi.org/10.1016/j.eswa.2012.02.032

Caixeta, M. C. B. F., & Fabricio, M. M. (2018). Métodos e instrumentos de apoio ao codesign no processo de projeto de edifícios. *Ambiente Construído*, *18*(1), 111–131. https://doi.org/10.1590/s1678-86212018000100212

Cheng, T. C. E., & Sin, C. C. S. (1990). A state-of-the-art review of parallel-machine scheduling research. *European Journal of Operational Research*, *47*(3), 271–292. https://doi.org/10.1016/0377-2217(90)90215-W

Coutinho, C. P. (2011). *Metodologia de Investigação em Ciências Sociais e Humanas: Teoria e Prática*. Edições Almedina.

Crainic, T. G., Djeumou Fomeni, F., & Rei, W. (2021). Multi-period bin packing model and effective constructive heuristics for corridor-based logistics capacity planning. *Computers & Operations Research*, *132*, 105308. https://doi.org/10.1016/j.cor.2021.105308

Daniels, R. L., Hoopes, B. J., & Mazzola, J. B. (1996). Scheduling Parallel Manufacturing Cells with Resource Flexibility. *Management Science*, *42*(9), 1260–1276. https://doi.org/10.1287/mnsc.42.9.1260

Daniels, R. L., Hua, S. Y., & Webster, S. (1999). Heuristics for parallel-machine flexible-resource scheduling problems with unspecified job assignment. *Computers & Operations Research*, *26*(2), 143–155. https://doi.org/10.1016/S0305-0548(98)00054-9

Della Croce, F., Salassa, F., & T'kindt, V. (2014). A hybrid heuristic approach for single machine scheduling with release times. *Computers & Operations Research*, *45*, 7–11. https://doi.org/10.1016/j.cor.2013.11.016

Edis, E. B., & Oguz, C. (2011). Parallel Machine Scheduling with Additional Resources: A Lagrangian-Based Constraint Programming Approach. Em *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems: 8th International Conference* (Vol. 6697, pp. 92–98). Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-21311-3_10

Edis, E. B., & Oguz, C. (2012). Parallel machine scheduling with flexible resources. *Computers & Industrial Engineering*, *63*(2), 433–447. https://doi.org/10.1016/j.cie.2012.03.018

Edis, E. B., Oguz, C., & Ozkarahan, I. (2013). Parallel machine scheduling with additional resources: Notation, classification, models and solution methods. *European Journal of Operational Research*, *230*(3), 449–463. https://doi.org/10.1016/j.ejor.2013.02.042

Edis, E. B., & Ozkarahan, I. (2010). A combined integer/constraint programming approach to a resource-constrained parallel machine scheduling problem with machine eligibility

restrictions. *Engineering Optimization*, *43*(2), 135–157. https://doi.org/10.1080/03052151003759117

Edis, E. B., & Ozkarahan, I. (2012). Solution approaches for a real-life resource-constrained parallel machine scheduling problem. *The International Journal of Advanced Manufacturing Technology*, *58*(9–12), 1141–1153. https://doi.org/10.1007/s00170-011-3454-8

Fanjul-Peyro, L. (2020). Models and an exact method for the Unrelated Parallel Machine scheduling problem with setups and resources. *Expert Systems with Applications: X*, *5*, 100022. https://doi.org/10.1016/j.eswax.2020.100022

Fanjul-Peyro, L., Perea, F., & Ruiz, R. (2017). Models and matheuristics for the unrelated parallel machine scheduling problem with additional resources. *European Journal of Operational Research*, *260*(2), 482–493. https://doi.org/10.1016/j.ejor.2017.01.002

Fanjul-Peyro, L., & Ruiz, R. (2010). Iterated greedy local search methods for unrelated parallel machine scheduling. *European Journal of Operational Research*, *207*(1), 55–69. https://doi.org/10.1016/j.ejor.2010.03.030

Fanjul-Peyro, L., Ruiz, R., & Perea, F. (2019). Reformulations and an exact algorithm for unrelated parallel machine scheduling problems with setup times. *Computers & Operations Research*, *101*, 173–182. https://doi.org/10.1016/j.cor.2018.07.007

Fleszar, K., & Hindi, K. S. (2018). Algorithms for the unrelated parallel machine scheduling problem with a resource constraint. *European Journal of Operational Research*, *271*(3), 839–848. https://doi.org/10.1016/j.ejor.2018.05.056

Gedik, R., Kalathia, D., Egilmez, G., & Kirac, E. (2018). A constraint programming approach for solving unrelated parallel machine scheduling problem. *Computers & Industrial Engineering*, *121*, 139–149. https://doi.org/10.1016/j.cie.2018.05.014

Glass, C. A., Shafransky, Y. M., & Strusevich, V. A. (2000). Scheduling for parallel dedicated machines with a single server. *Naval Research Logistics*, *47*(4), 304–328. https://doi.org/10.1002/(SICI)1520-6750(200006)47:4<304::AID-NAV3>3.0.CO;2-1

Glover, F. (1986). Future paths for integer programming and links to artificial intelligence. *Computers & Operations Research*, *13*(5), 533–549. https://doi.org/10.1016/0305-0548(86)90048-1

Grigoriev, A., Sviridenko, M., & Uetz, M. (2007). Machine scheduling with resource dependent processing times. *Mathematical Programming*, *110*(1), 209–228. https://doi.org/10.1007/s10107-006-0059-3

Guinet, A. (1993). Scheduling sequence-dependent jobs on identical parallel machines to minimize completion time criteria. *International Journal of Production Research*, *31*(7), 1579–1594. https://doi.org/10.1080/00207549308956810

Hatami, S., Ruiz, R., & Andres-Romano, C. (2015). Heuristics for a Distributed Parallel Machine Assembly Scheduling Problem with eligibility constraints. *2015 International Conference on Industrial Engineering and Systems Management (IESM)*, 145–153. https://doi.org/10.1109/IESM.2015.7380149

Hatami, S., Ruiz, R., & Andrés-Romano, C. (2013). The Distributed Assembly Permutation Flowshop Scheduling Problem. *International Journal of Production Research*, *51*(17), 5292–5308. https://doi.org/10.1080/00207543.2013.807955

Heinz, V., Novák, A., Vlk, M., & Hanzálek, Z. (2022). Constraint Programming and constructive heuristics for parallel machine scheduling with sequence-dependent setups and common servers. *Computers & Industrial Engineering*, *172*, 108586. https://doi.org/10.1016/j.cie.2022.108586

Huang, S., Cai, L., & Zhang, X. (2010). Parallel dedicated machine scheduling problem with sequence-dependent setups and a single server. *Computers & Industrial Engineering*, *58*(1), 165–174. https://doi.org/10.1016/J.CIE.2009.10.003

Inplas - Simoldes. (sem data). Obtido 21 de Janeiro de 2023, de https://www.simoldes.com/plastics-companies/inplas/

Karlsson, E., Rönnberg, E., Stenberg, A., & Uppman, H. (2021). A matheuristic approach to large-scale avionic scheduling. *Annals of Operations Research*, *302*(2), 425–459. https://doi.org/10.1007/S10479-020-03608-6/TABLES/7

Kellerer, H., & Strusevich, V. A. (2003). Scheduling parallel dedicated machines under a single non-shared resource. *European Journal of Operational Research*, *147*(2), 345–364. https://doi.org/10.1016/S0377-2217(02)00246-1

Kim, M.-Y., & Lee, Y. H. (2012). MIP models and hybrid algorithm for minimizing the makespan of parallel machines scheduling problem with a single server. *Computers & Operations Research*, *39*(11), 2457–2468. https://doi.org/10.1016/j.cor.2011.12.011

Ku, W.-Y., & Beck, J. C. (2016). Mixed Integer Programming models for job shop scheduling: A computational analysis. *Computers & Operations Research*, *73*, 165–173. https://doi.org/10.1016/j.cor.2016.04.006

Lee, C.-Y., & Massey, J. D. (1988). Multiprocessor scheduling: combining LPT and MULTIFIT. *Discrete Applied Mathematics*, *20*(3), 233–242. https://doi.org/10.1016/0166-218X(88)90079-0

Lopez-Esteve, A., Perea, F., & Yepes-Borrero, J. C. (2022). GRASP algorithms for the unrelated parallel machines scheduling problem with additional resources during processing and setups. *International Journal of Production Research*, 1–17. https://doi.org/10.1080/00207543.2022.2121869

M. Pour, S., Drake, J. H., Ejlertsen, L. S., Rasmussen, K. M., & Burke, E. K. (2018). A hybrid Constraint Programming/Mixed Integer Programming framework for the preventive signaling maintenance crew scheduling problem. *European Journal of Operational Research*, *269*(1), 341–352. https://doi.org/10.1016/j.ejor.2017.08.033

Moser, M., Musliu, N., Schaerf, A., & Winter, F. (2022). Exact and metaheuristic approaches for unrelated parallel machine scheduling. *Journal of Scheduling*, *25*(5), 507–534. https://doi.org/10.1007/s10951-021-00714-6

Moussavi, S. E., Mahdjoub, M., & Grunder, O. (2019). A matheuristic approach to the integration of worker assignment and vehicle routing problems: Application to home healthcare

scheduling. *Expert Systems with Applications*, *125*, 317–332. https://doi.org/10.1016/j.eswa.2019.02.009

Nguyen, V. H., Tuong, N. H., Tran, V. H., & Thoai, N. (2013). An MILP-based makespan minimization model for single-machine scheduling problem with splitable jobs and availability constraints. *2013 International Conference on Computing, Management and Telecommunications (ComManTel)*, 397–400. https://doi.org/10.1109/ComManTel.2013.6482427

Ozer, E. A., & Sarac, T. (2019). MIP models and a matheuristic algorithm for an identical parallel machine scheduling problem under multiple copies of shared resources constraints. *TOP*, *27*(1), 94–124. https://doi.org/10.1007/s11750-018-00494-x

Panwalkar, S. S., Dudek, R. A., & Smith, M. L. (1973). Sequencing Research and the Industrial Scheduling Problem. *Symposium on the Theory of Scheduling and Its Applications*, 29–38. https://doi.org/10.1007/978-3-642-80784-8_2

Pinedo, M. L. (1995). *Scheduling theory, algorithms and systems* (Fifth Edit). Springer International Publishing. https://doi.org/10.1007/978-3-319-26580-3

Pinheiro, J. C. S. N., Arroyo, J. E. C., & Fialho, L. B. (2020). Scheduling unrelated parallel machines with family setups and resource constraints to minimize total tardiness. *Proceedings of the 2020 Genetic and Evolutionary Computation Conference Companion*, 1409–1417. https://doi.org/10.1145/3377929.3398150

Pinto, J. M., & Grossmann, I. E. (1997). A logic-based approach to scheduling problems with resource constraints. *Computers & Chemical Engineering*, *21*(8), 801–818. https://doi.org/10.1016/S0098-1354(96)00318-3

Potts, C. N. (1985). Analysis of a linear programming heuristic for scheduling unrelated parallel machines. *Discrete Applied Mathematics*, *10*(2), 155–164. https://doi.org/10.1016/0166-218X(85)90009-5

Puchinger, J., & Raidl, G. R. (2005). Combining Metaheuristics and Exact Algorithms in Combinatorial Optimization: A Survey and Classification. *First International Work-Conference on the Interplay Between Natural and Artificial Computation*, *3562*(PART II), 41–53. https://doi.org/10.1007/11499305_5

Reklaitis, G. V. (1996). Overview of Scheduling and Planning of Batch Process Operations. Em *Batch Processing Systems Engineering* (pp. 660–705). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-60972-5_27

Saidi-Mehrabad, M., & Fattahi, P. (2007). Flexible job shop scheduling with tabu search algorithms. *The International Journal of Advanced Manufacturing Technology*, *32*(5–6), 563–570. https://doi.org/10.1007/s00170-005-0375-4

Simoldes Plastics. (sem data). Obtido 21 de Janeiro de 2023, de https://www.simoldes.com/

Słowiński, R. (1980). Two Approaches to Problems of Resource Allocation Among Project Activities — A Comparative Study. *Journal of the Operational Research Society*, *31*(8), 711–723. https://doi.org/10.1057/jors.1980.134

Soares, L. C. R., & Carvalho, M. A. M. (2022). Application of a hybrid evolutionary algorithm to resource-constrained parallel machine scheduling with setup times. *Computers & Operations Research*, *139*, 105637. https://doi.org/10.1016/j.cor.2021.105637

Su, L.-H., & Lien, C.-Y. (2009). Scheduling parallel machines with resource-dependent processing times. *International Journal of Production Economics*, *117*(2), 256–266. https://doi.org/10.1016/j.ijpe.2008.10.014

Tran, T. T., Araujo, A., & Beck, J. C. (2016). Decomposition Methods for the Parallel Machine Scheduling Problem with Setups. *INFORMS Journal on Computing*, *28*(1), 83–95. https://doi.org/10.1287/ijoc.2015.0666

Vahedi-Nouri, B., Tavakkoli-Moghaddam, R., Hanzálek, Z., & Dolgui, A. (2022). Workforce planning and production scheduling in a reconfigurable manufacturing system facing the COVID-19 pandemic. *Journal of Manufacturing Systems*, *63*, 563–574. https://doi.org/10.1016/j.jmsy.2022.04.018

Vallada, E., & Ruiz, R. (2011). A genetic algorithm for the unrelated parallel machine scheduling problem with sequence dependent setup times. *European Journal of Operational Research*, *211*(3), 612–622. https://doi.org/10.1016/j.ejor.2011.01.011

Vasquez, M., & Hao, J.-K. (2001). A Hybrid Approach for the 0-1 Multidimensional Knapsack problem. *Proceedings of the International Joint Conference on Artificial Intelligence 2001*, 328–333.

Vázquez-Serrano, J. I., Cárdenas-Barrón, L. E., Peimbert-García, R. E., Lin, Y.-K., Fiondella, L., Huang, C.-F., & Chang, P.-C. (2021). Agent Scheduling in Unrelated Parallel Machines with Sequence- and Agent–Machine–Dependent Setup Time Problem. *Mathematics 2021, Vol. 9, Page 2955*, *9*(22), 2955. https://doi.org/10.3390/MATH9222955

Ventura, J. A., & Kim, D. (2000). Parallel machine scheduling about an unrestricted due date and additional resource constraints. *IIE Transactions*, *32*(2), 147–153. https://doi.org/10.1023/A:1007662314880

Villa, F., Vallada, E., & Fanjul-Peyro, L. (2018). Heuristic algorithms for the unrelated parallel machine scheduling problem with one scarce additional resource. *Expert Systems with Applications*, *93*, 28–38. https://doi.org/10.1016/j.eswa.2017.09.054

Wang, S., Wu, R., Chu, F., & Yu, J. (2020). Identical parallel machine scheduling with assurance of maximum waiting time for an emergency job. *Computers & Operations Research*, *118*, 104918. https://doi.org/10.1016/j.cor.2020.104918

Woo, Y.-B., & Kim, B. S. (2018). Matheuristic approaches for parallel machine scheduling problem with time-dependent deterioration and multiple rate-modifying activities. *Computers & Operations Research*, *95*, 97–112. https://doi.org/10.1016/j.cor.2018.02.017

Yepes-Borrero, J. C., Perea, F., Ruiz, R., & Villa, F. (2021). Bi-objective parallel machine scheduling with additional resources during setups. *European Journal of Operational Research*, *292*(2), 443–455. https://doi.org/10.1016/j.ejor.2020.10.052

Yepes-Borrero, J. C., Villa, F., Perea, F., & Caballero-Villalobos, J. P. (2020). GRASP algorithm for the unrelated parallel machine scheduling problem with setup times and additional resources. *Expert Systems with Applications*, *141*, 112959. https://doi.org/10.1016/j.eswa.2019.112959

Yunusoglu, P., & Yildiz, S. T. (2022). Constraint programming approach for multi-resource-constrained unrelated parallel machine scheduling problem with sequence-dependent setup times. *International Journal of Production Research*, *60*(7), 2212–2229. https://doi.org/10.1080/00207543.2021.1885068

Zhang, L., Deng, Q., Lin, R., Gong, G., & Han, W. (2021). A combinatorial evolutionary algorithm for unrelated parallel machine scheduling problem with sequence and machine-dependent setup times, limited worker resources and learning effect. *Expert Systems with Applications*, *175*(2), 114843. https://doi.org/10.1016/j.eswa.2021.114843

ANEXO A

The identical parallel machine scheduling problem with setups and additional resources

Ângelo Soares[1[0000-0003-0783-7295]], Ana Rita Ferreira[1[0000-0001-9215-7192]] and Manuel Pereira Lopes[1,2*[0000-0002-1146-7226]]

[1]ISEP, Polytechnic of Porto, R. Dr. António Bernardino de Almeida, 4249-015 Porto, Portugal

[2] INESC TEC, R. Dr. Roberto Frias 400, 4200-465 Porto, Portugal

*mpl@isep.ipp.pt

Abstract. This paper studies a real world dedicated parallel machine scheduling problem with sequence dependent setups, different machine release dates and additional resources (PMSR). To solve this problem, two previously proposed models have been adapted and a novel objective function, the minimisation of the sum of the machine completion times, is proposed to reflect the real conditions of the manufacturing environment that motivates this work. One model follows the strip-packing approach and the other is time-indexed. The solutions obtained show that the new objective function provides a compact production schedule that allows the simultaneous minimisation of machine idle times and setup times. In conclusion, this study provides valuable insights into the effectiveness of different models for solving PMSR problems in real-world contexts and gives directions for future research in this area using complementary approaches such as matheuristics.

Keywords: Parallel machines, Scheduling, Setups, Additional Resources

1. Introduction

The study of Parallel Machine scheduling problems (PM) has mostly relied on machines as the primary resource, neglecting the fact that real manufacturing settings involve other critical resources such as automated guided vehicles, machine operators, tools, pallets, dies, and industrial robots [1-3]. The PM with sequence dependent setup times (PMS) where additional processing and setup resources are required (PMSR) is more complex. It involves sequencing and timing decisions and balancing multiple performance metrics. This study focuses on the injection moulding department of an automotive company that produces plastic injection moulded parts for final assembly plants with a maximum response time of 48 hours. To solve this real problem, two existing models are adapted to the PMSR involving identical PM with different machine release dates, and a new objective function that minimises the sum of the completion times for each machine is proposed. This objective function provides a compact production schedule that allows the simultaneous minimisation of machine idle times and setup times for the specific case of dedicated PM with sequence dependent setup times. To the authors' knowledge, no existing research has explored the minimization of the sum of completion times for each machine as an objective function. This study provides valuable insights into the effectiveness of different models for solving this type of PMSR problem in real-world contexts.

2. Literature review

In recent years, several papers have addressed the more general case of PMSP with setups and resources. Akyol Ozer, et al. [4] studied the identical PMSP with sequence-dependent setup times, machine eligibility restrictions and multiple copies of shared resources. In this paper, the objective function was the minimization of the total weighted completion time. For small instances (8 jobs, 3 machines and 5 resources), a mixed integer linear programming model (MILP) obtained optimal solutions in almost all tests. For larger problems, a matheuristic consisting of a MILP and a Genetic Algorithm (GA) has been implemented, which provides solutions for up to 100 jobs, 8 machines and 50 resources. Bektur, et al. [5] presented a MILP and two metaheuristics, Tabu Search (TS) and Simulated Annealing (SA), for the case of a team as a setup resource and the objective of minimizing the total weighted tardiness. MILP only gives optimal solutions for 2 machines and 10 jobs with a time limit of 7200 s. Regarding the metaheuristics, for up to 10 machines and 100 jobs, TS outperformed SA with an average relative deviation of 5.58%. Pinheiro, et al. [6] propose a solution approach to the unrelated PMSP with family setups and resource constraints, to minimize the total tardiness of the jobs. The authors develop a MILP and two metaheuristics, SA and Iterated Greedy (IG). The MILP showed good performance for instances with up to 20 jobs and 6 machines, with a computational time limit of 3600 s. To address larger problems, a constructive heuristic was used to provide an initial solution, which was then improved by the metaheuristics. For instances with up to 50 jobs, the IG algorithm outperformed the SA algorithm, exhibiting lower relative deviation values. In fact, IG and SA algorithms have average relative deviations of 8.84% and 39.87%, respectively. The computational time required for the IG algorithm to provide a solution was also reported to be lower than the MILP model. Fanjul-Peyro [7] addressed the Unrelated PM with Setups and Resources (UPMSR), with three types of resources: specific for processing, specific for setup and shared. The authors presented two MILP models and a three-phase exact algorithm (TPhA) to improve the model's performance. Yepes-Borrero, et al. [8] implemented a GRASP algorithm with the objective of minimizing makespan and in [9] have also proposed a multi-objective approach to minimise both the makespan and the number of setup resources, building on the previous work [7, 8]. An adaptation of the Restarted Iterated Pareto Greedy (RIPG) algorithm is proposed that gives solutions for up to 250 jobs and 30 machines. Zhang, et al. [10] tackled the unrelated PMSP, with limited worker resources and learning effect. They proposed a Combinatorial Evolutionary Algorithm to minimize makespan and energy consumption. The algorithm can handle up to 400 jobs, 22 machines, and 10 resources. Lopez-Esteve, et al. [11] also developed GRASP algorithms to solve the unrelated PMSP with additional resources during processing and setups which gives solutions with a gap of 1.80% in relation to the Fanjul-Peyro [7]. Yunusoglu, et al. [12] proposed an exact model based on constraint programming (CP) to solve the multi-resource-constrained unrelated PMSP with sequence-dependent setup times. The objective was to minimize the makespan and release dates were incorporated into the problem as operational constraints to reflect real-world manufacturing environments. The computational results indicate that the proposed CP model outperforms the best solutions of previous works with an average gap of 15.52%. Finally, Afzalirad, et al. [13] inspired on the block erection scheduling problem in a shipyard, studied the unrelated PMSP with resource constrains, sequence-dependent setup times, different release dates, machine eligibility and precedence constraints, to minimize the makespan. The results indicate that in the case of small-scale problems, both methods are efficient and effective, whereas in large-scale problems, the AIS approach statistically outperforms the GA approach.

3. Problem Formulation

3.1. Problem description

We consider a parallel dedicated machine scheduling problem, where *n* jobs are processed on *m* machines, each job belonging to a family and processed on a single machine without preemption. A set of parts which use the same die is a family and a sequence dependent setup must be performed between jobs belonging to different families. Operators are required to process jobs and setup teams are required to perform the machine setups. Operators and setup teams are renewable and dynamic resources. The number of operators for each shift is known in advance, and the system must provide scheduling solutions at the start of each shift. The objective of the Injection Moulding Department is to meet the daily demand for parts within the constraints of capacity, and the availability of operators and setup teams, minimizing the maximum completion time for each machine.

3.2. Model formulation

In this section we formally introduce the dedicated parallel machine with setups and resources. In this paper we use small letters to refer to parameters and capital letters to refer to variables. The sets of variables and parameters used are the following:

- A set of *m* machines that can process the jobs, denoted as $M = \{1, \dots, m\}$, indexed by *i*.
- A set of *m*+*n* jobs to be processed, $N = \{1, \dots, m + n\}$, indexed by *j*, where the first *m* jobs represent the initial jobs on the respective machine number.
- p_j denotes the time required to process job *j*.
- s_{jk} denotes the time required to do the setup between jobs *j* and *k*.
- r_{max}^P are the maximum amount of processing resources. These are required specifically for processing jobs and cannot be used for setups. We denote these resources as P-resources.
- r_{max}^S are the maximum amount of machine setup resources. These are required specifically for machine setup and cannot be used for job processing. We denote these resources as S-resources.
- r_j^P are the resources of P-resources required to do processing of job *j*.
- r_j^S are the resources of S-resources required to do setup of job *j*.
- *M* is a sufficiently large constant.

As the machines are dedicated, the set of *N* jobs is divided into *m* subsets ($N_1, \dots, N_m$) so that the jobs in set N_i are only processed on machine *i*. Each machine has a dummy job (0) representing its start and end. Therefore, N_{i0}=$N_i \cup \{0\}$ represents the set of jobs which are to be processed on machine *i* plus the dummy job. Moreover, processing and setup times (p_j and s_{jk}), as well as the resources required for each operation (r_j^P and r_j^S) are deterministic. In this paper the UPMSR models based on strip-packing and indexed to time introduced by [7] are adapted to a real problem where the parallel machines are dedicated and have different release dates (the production system has an initial state), and a new objective function is proposed to minimize the sum of the machines completion times. The decision variables used are the following:

- $X_{ijk} = 1$ if job *k* is processed immediately after job *j* on machine *i*, zero otherwise.
- $Cmax_i$ is the completion time (minutes) of machine *i*.

- C_j is the completion time of job *j* (coordinate where processing of job *j* finishes in the time dimension) (minutes).
- B_j is the completion time of the setup before the processing of job *j* on the corresponding machine (coordinate where setup of job *j* finishes in the time dimension).
- $R_j^P \in [0, r_{max}^P]$ is the coordinate of processing of job *j* in the P-resources dimension (the specific resources employed in the processing of job *j*).
- $R_j^S \in [0, r_{max}^S]$ is the coordinate of setup of job *j* in the S-resources dimension (the specific resources employed in the setup before processing of job *j*).
- D_{jk}^P, E_{jk}^P, F_{jk}^S G_{jk}^S are binary auxiliary variables with value = 1 when the variable value studied in constraints (time or resources) of job *k* is higher than the variable value studied (time or resources) of job *j*. The letters in superscript are used to indicate the dimension where they are used.

The MILP strip-packing model is:

$$Min\ Z = \sum_{i \in M} Cmax_i \tag{1}$$

$$X_{i0i} = 1, i \in M \tag{2}$$

$$\sum_{j \in N_i, j \neq k} X_{ijk} = 1, i \in M, k \in N_i \backslash \{i\} \tag{3}$$

$$\sum_{j \in N_{i0} \backslash \{i\}, j \neq k} X_{ikj} = 1, i \in M, k \in N_i \tag{4}$$

$$C_k - C_j + M\left(1 - X_{ijk}\right) \geq s_{jk} + p_k, i \in M,\ j \in N_i, k \in N_i \backslash \{i\}, j \neq k \tag{5}$$

$$C_k \geq \sum_{j \in N_{i0}, j \neq k, s_{jk} > 0} X_{ijk}\, s_{jk} + p_k, k \in N_i \backslash \{i\}, i \in M \tag{6}$$

$$C_k \geq \sum_{j \in N_{i0}, j \neq k, s_{jk} > 0} X_{ijk}\, (s_{jk} + p_j) + p_k, k \in N_i \backslash \{i\}, i \in M \tag{6b}$$

$$B_k \geq \sum_{j \in N_{i0}, j \neq k, s_{jk} > 0} X_{ijk}\, (s_{jk} + p_j) + p_k, k \in N_i \backslash \{i\}, i \in M \tag{6c}$$

$$B_k \geq \sum_{j \in N_{i0}, j \neq k, s_{jk} > 0} X_{ijk}\, (s_{jk} + p_j), k \in N_i \backslash \{i\}, i \in M \tag{6d}$$

$$C_i \geq p_i, i \in M \tag{7}$$

$$Cmax_i \geq C_k, i \in M, k \in N_i \tag{8}$$

$$B_k - C_j + M\left(1 - X_{ijk}\right) \geq s_{jk},\ i\ \in M, j \in N_i, k\ \in N_i \backslash \{i\}, j \neq k, s_{jk} > 0 \tag{9}$$

$$B_k \leq C_k - p_k, k\ \in N \tag{10}$$

$$r_{max}^{P} \geq R_k^P, k \in N \quad (11)$$

$$R_k^P \geq r_k^p, k \in N \quad (12)$$

$$r_{max}^{S} \geq R_k^S, k \in N \quad (13)$$

$$R_k^S \geq r_k^S, k \in N \quad (14)$$

$$C_k - C_j + M\left(1 - D_{jk}^P\right) \geq p_k, i, q \in M, q \neq i, j \in N_i, k \in N_q \quad (15)$$

$$R_k^P - R_j^P + M\left(1 - E_{jk}^P\right) \geq r_k^P, i, q \in M, q \neq i, j \in N_i, k \in N_q \quad (16)$$

$$D_{jk}^P + D_{kj}^P + E_{jk}^P + E_{kj}^P \geq 1, i, q \in M, q \neq i, j \in N_i, k \in N_q, j \neq k \quad (17)$$

$$B_k - B_j + M\left(1 - F_{jk}^S\right) \geq \sum_{l \in N_{q0}, l \neq k, s_{lk} \geq 0} X_{qlk}\, s_{lk}, i, q \in M, q \neq i, j \in N_i, k \in N_q \quad (18)$$

$$R_k^S - R_j^S + M\left(1 - G_{jk}^S\right) \geq r_k^S, i, q \in M, q \neq i, j \in N_i, k \in N_q \quad (19)$$

$$F_{jk}^S + F_{kj}^S + G_{jk}^S + G_{kj}^S \geq 1, i, q \in M, q \neq i, j \in N_i, k \in N_q \quad (20)$$

$$X_{ijk} \in \{0,1\}, Y \geq 0, C_k \geq 0, B_k \geq 0\ j, k \in N_i, i \in M \quad (21)$$

The objective function (1) minimizes the sum of maximum completion time for each machine. Constraints (2) ensure that the initial jobs are the first to be processed immediately after the dummy jobs on each machine. The constraints (3) and (4) make sure that only one job is processed immediately before and only one job is processed immediately after each job. Constraints (5) break subtours and give the right scheduling order, because if *k* is processed after *j* on machine *i*, it forces finishing the processing of job *k* not earlier than s_{jk} + p_k. Constraints (6) set the lower limit of completion time of job *k* (C_k) to be greater or equal than the sum of its setup plus its processing time while constraints (7) do the same for the initial jobs. Constraints (8) establish the relationship between $Cmax_i$ of machine *i* and the completion times of jobs on that machine. Constraints (9) force that, if job j is the previous job done before job *k* on machine *i*, the finishing time of setup *k* must be later than the finishing time of processing of job *j* plus setup time of job *k*. Constraints (10) ensure that setup finishing time of job *k* is earlier than processing starting time of job *k*. Constraints (11-14) set the coordinate limits of specific resources used in processing (R_k^P) and used specifically in setups (R_k^S). Constraints (15) show that, if C_k is later than C_j, the difference between these completion times is greater than or equal to the processing time of job *k*. In (16) if R_k^P is greater than R_j^P, the difference between coordinates of these resource points is greater than or equal to the resources used in the job *k* processing. Constraints (17) force at least one of the binary variables to have the value of 1. Constraints (18) show that, if B_k is later than B_j, the difference between these setup finishing times is greater than or equal to the setup time of job *k*. In (19) if R_k^S is greater than R_j^S, the difference between coordinates of these resource points is greater than or equal to the resources used in the job *k* setup. Constraints (20) force at least one of the binary variables to have the value of 1. Finally, constraints (21) set the limits of the variables. In this model, in addition

to several simplifications of the original model, the objective function (1) and the constraints (2) are new to reflect the specific conditions of the real problem. The MILP model indexed to time is:

- $E_{ikt} = 1$ if *k* is being processed on machine *i* in time *t*, zero otherwise.
- $F_{ijkt} = 1$ if *k* is in setup after job *j* on machine *i* in time *t*, zero otherwise.
- t_{max} is the planning horizon.

E_{ikt} and F_{ijkt} are decision variables and t_{max} is a parameter. For this MILP we use Eqs. (1) – (10) and (21). The other constraints are:

$$\sum_{t \le t_{max}} E_{ikt} \ge p_k, i \in M, k \in N_i \tag{22}$$

$$C_k \ge tE_{ikt}, i \in M, k \in N_i, t \le t_{max} \tag{23}$$

$$C_k + 1 - p_k \le tE_{ikt} + M(1 - E_{ikt}), i \in M, k \in N_i, t \le t_{max} \tag{24}$$

$$\sum_{t \le t_{max}} F_{ijkt} \ge s_{jk} X_{ijk}, i \in M, j \in N_i, k \in N_i, j \ne k, s_{jk} > 0, k \ge m \tag{25}$$

$$B_k \ge \sum_{j \in N_i, j \ne k} tF_{ijkt}, i \in M, k \in N_i, t \le t_{max}, s_{jk} > 0, k \ge m \tag{26}$$

$$B_k + 1 - \sum_{j \in N_i, j \ne k, s_{jk} > 0} s_{jk} X_{ijk} \le \sum_{j \in N_i, j \ne k, s_{jk} > 0} tF_{ijkt} + M\left(1 - \sum_{j \in N_i, j \ne k, s_{jk} > 0} F_{ijkt}\right), i \in M, k \in N_i, t \le t_{max} \tag{27}$$

$$R^P_{max} \ge \sum_{i \in M, k \in N_i} r^P_k E_{ikt}, t \le t_{max} \tag{28}$$

$$R^S_{max} \ge \sum_{i \in M, j, k \in N_i, j \ne k, s_{jk} > 0} r^S_k F_{ijkt}, t \le t_{max} \tag{29}$$

Constraints (21) and (22) work together to ensure that the E_{ikt} variable is only 1 for the exact duration of time that job *k* is being processed on machine *i*. This prevents unnecessary idle time on the machine and ensures that each job is processed for the minimum amount of time necessary to complete it. Constraints (23) complements these constraints by requiring that when E_{ikt} is equal to 1, the current time *t* must be greater than or equal to the completion time of job *k* minus its processing time plus 1. Constraints (24) - (26) represent the same for setups, where the time considered is the setup time. Constraints (27) ensures that the sum of all resources used in processing at each instant *t* is less than or equal to the maximum number of resources available for processing. Constraints (28) ensures the same for setups.

4. Computational experiments

To generate the instances, the number of jobs, number of machines, processing and setup times, number of setup teams, number of operators available, and their use by jobs were defined based on the real data from the company, and the time is in minutes. The sets of problems, small, medium, and large, are presented in Table 1. For each of the sets, 10 instances were randomly generated using processing and setup times generated from distributions of *U*(20,240), and *U*(30,50), respectively, and the number of operators to support the processing of jobs on the machines generated from the distribution of *U*(1,4). Setup times must satisfy the triangular inequality. The setups require a setup team. For the time indexed model, t_{max} was set to 480 minutes (one shift).

Table 1. Sets of test instances

Instance	Machines	Jobs	Operators	Setup Teams
Small	4	12	8	1
Medium	8	24	16	2
Large	16	40	30	2

The CPLEX 22.1.1.0 solver was used, with programming done in Python using the Pulp library on Microsoft Visual Studio Code 2023 IDE. The computational tests were executed on an Intel Core i7-85654 CPU1.80GHz, with 8.00 GB of RAM. To ensure reasonable runtimes, a time limit of 1800 seconds was set. Throughout the computational study, the indicators analysed were the gap to the optimum solution, the percentage of optimal solutions obtained, and the computation time. The performance of two models, strip-packing and time-indexed, was analysed and the average results for each set of instances are presented in Table 2. For medium and large cases, only the results for the strip-packing model are presented, as the time-index model could not find a solution within the time limit. The strip-packing model gives good results for the small and medium instances, and for the large instances it could always find a feasible solution with and average gap of 14.09%. In contrast, the time-indexed model can only solve small problems.

Table 2. Comparison between strip-packing and time-indexed models

	Strip-packing			Time-indexed		
Instance	%Gap	%Opt	Time (s)	%Gap	%Opt	Time (s)
Small	0.00	100	0.91	5.6	60	1800
Medium	2.15	40	984.05	-	-	-
Large	14.09	0	1800	-	-	-

5. Conclusions and future research

This paper studies the special case of PMSR. To solve this problem, two previously proposed models have been adapted and a novel objective function, the minimisation of the sum of the machine completion times, is proposed to reflect the real conditions of the manufacturing environment that motivates this work. Overall, the strip-packing model outperforms the time-indexed model in solving this type of PMSR problem and shows a good potential for solving real-world problems and for future research in this area. The solutions obtained show that the new objective function provides a compact production schedule that allows the simultaneous minimisation of machine idle times and setup times for the PMSR with identical parallel machines with different release dates. To further enhance the performance of the strip-packing model, complementary approaches such

as matheuristics can be explored. Matheuristics combine the strengths of mathematical programming and metaheuristics to provide a hybrid optimization approach that can efficiently solve complex problems.

References

1. Blazewicz, J., Lenstra, J.K., Kan, A.R.: Scheduling subject to resource constraints: classification and complexity. Discrete applied mathematics 5, 11-24 (1983).
2. Błażewicz, J., Kubiak, W., Röck, H., Szwarcfiter, J.: Minimizing mean flow-time with parallel processors and resource constraints. Acta informatica 24, 513-524 (1987).
3. Ventura, J.A., Kim, D.: Parallel machine scheduling about an unrestricted due date and additional resource constraints. Iie Transactions 32, 147-153 (2000).
4. Akyol Ozer, E., Sarac, T.: MIP models and a matheuristic algorithm for an identical parallel machine scheduling problem under multiple copies of shared resources constraints. Top 27, 94-124 (2019).
5. Bektur, G., Saraç, T.: A mathematical model and heuristic algorithms for an unrelated parallel machine scheduling problem with sequence-dependent setup times, machine eligibility restrictions and a common server. Computers & Operations Research 103, 46-63 (2019).
6. Pinheiro, J.C., Arroyo, J.E.C., Fialho, L.B.: Scheduling unrelated parallel machines with family setups and resource constraints to minimize total tardiness. In: Proceedings of the 2020 genetic and evolutionary computation conference companion, pp. 1409-1417. (2020).
7. Fanjul-Peyro, L.: Models and an exact method for the unrelated parallel machine scheduling problem with setups and resources. Expert Systems with Applications: X 5, 100022 (2020).
8. Yepes-Borrero, J.C., Villa, F., Perea, F., Caballero-Villalobos, J.P.: GRASP algorithm for the unrelated parallel machine scheduling problem with setup times and additional resources. Expert Systems with Applications 141, 112959 (2020).
9. Yepes-Borrero, J.C., Perea, F., Ruiz, R., Villa, F.: Bi-objective parallel machine scheduling with additional resources during setups. European Journal of Operational Research 292, 443-455 (2021).
10. Zhang, L., Deng, Q., Lin, R., Gong, G., Han, W.: A combinatorial evolutionary algorithm for unrelated parallel machine scheduling problem with sequence and machine-dependent setup times, limited worker resources and learning effect. Expert Systems with Applications 175, 114843 (2021).
11. Lopez-Esteve, A., Perea, F., Yepes-Borrero, J.C.: GRASP algorithms for the unrelated parallel machines scheduling problem with additional resources during processing and setups. International Journal of Production Research 1-17 (2022).
12. Yunusoglu, P., Topaloglu Yildiz, S.: Constraint programming approach for multi-resource-constrained unrelated parallel machine scheduling problem with sequence-dependent setup times. International Journal of Production Research 60, 2212-2229 (2022).
13. Afzalirad, M., Rezaeian, J.: Resource-constrained unrelated parallel machine scheduling problem with sequence dependent setup times, precedence constraints and machine eligibility restrictions. Computers & Industrial Engineering 98, 40-52 (2016).