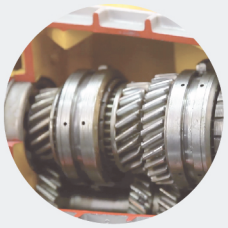
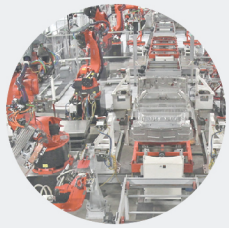
INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO
MESTRADO EM ENGENHARIA E GESTÃO INDUSTRIAL



DESENVOLVIMENTO DE UM ALGORITMO GREEDY RANDOMIZED ADAPTIVE SEARCH PROCEDURE PARA A PROGRAMAÇÃO DE MÁQUINAS PARALELAS DEDICADAS COM SETUPS E RECURSOS A

NUNO DIOGO GONÇALVES
novembro de 2024

POLITÉCNICO
DO PORTO

ISEP INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO

DESENVOLVIMENTO DE UM ALGORITMO GREEDY RANDOMIZED ADAPTIVE SEARCH PROCEDURE PARA A PROGRAMAÇÃO DE MÁQUINAS PARALELAS DEDICADAS COM MÚLTIPLAS ALTERNATIVAS, SETUPS, RECURSOS ADICIONAIS E DATAS DE ENTREGA

Nuno Diogo Gonçalves

Dissertação para obtenção do Grau de Mestre em Engenharia e Gestão Industrial

Orientador: Professor Doutor Manuel Joaquim Pereira Lopes

Porto, novembro 2024

Resumo

A INPLAS, empresa pertencente à divisão de Plásticos do Grupo Simoldes e especializada na injeção de componentes termoplásticos, identificou a necessidade de aprimorar o Sistema Inteligente de Apoio à Decisão (SIAD) utilizado pelo seu Departamento de Planeamento de Produção para dar resposta ao problema de programação de máquinas paralelas dedicadas com múltiplas alternativas, *setups* e recursos adicionais e datas de entrega. Com esse objetivo, desenvolveu-se um novo SIAD que incorpora a consideração de máquinas alternativas e a aplicação de uma metaheurística *Greedy Randomized Adaptive Search Procedure (GRASP)*, visando aumentar a flexibilidade e a eficiência do planeamento e responder melhor às necessidades de produção.

O processo de planeamento foi inicialmente estudado em profundidade, abrangendo tanto as etapas a montante quanto as etapas a jusante, com o objetivo de identificar as variáveis-chave, os *inputs* e os *outputs* essenciais para a resolução deste problema. Neste contexto, foram analisadas minuciosamente as características e restrições do processo, especialmente aquelas associadas às limitações do sistema de produção. Com uma compreensão detalhada do problema em mãos, identificaram-se potenciais oportunidades de melhoria que poderiam aumentar tanto a adaptabilidade do sistema ao ambiente de produção da INPLAS quanto a eficácia dos resultados obtidos pelo SIAD. Assim, decidiu-se que a melhoria a implementar, com o intuito de potencializar o desempenho do SIAD, seria a inclusão de máquinas alternativas no sistema, enquanto a resolução do problema de escalonamento seria abordada com a aplicação de uma metaheurística GRASP.

Considerando a complexidade do problema de escalonamento em máquinas paralelas dedicadas com múltiplas alternativas, *setups* e recursos adicionais, classificado como NP-Hard, este trabalho desenvolve uma metaheurística GRASP adaptada, fundamentada nas melhores práticas recomendadas na literatura. A abordagem proposta oferece cinco contribuições específicas. Em primeiro lugar, adapta os modelos de referência ao introduzir uma separação inovadora entre os processos de alocação e sequenciação, permitindo que estas etapas sejam tratadas de forma independente, o que é raro em abordagens tradicionais. Em segundo lugar, integra datas de entrega comuns, o que exige a introdução de uma variável de atraso para cada trabalho e de uma função multiobjetivo, com foco na minimização do número de trabalhos em atraso, assegurando o cumprimento dos prazos estabelecidos. Adicionalmente, a metaheurística GRASP foi adaptada para um sistema de produção contínua, em que os trabalhos inacabados do dia anterior são sequenciados na primeira posição do dia seguinte, refletindo de forma mais realista o ambiente produtivo. Por fim, ajustou-se uma heurística construtiva original, que utiliza um coeficiente exclusivo para classificar as posições de inserção dos trabalhos nas soluções parciais, visando não apenas a minimização dos atrasos, mas também a redução do número de *setups* e do *makespan* total.

Os resultados obtidos demonstram a eficácia da metaheurística GRASP no apoio ao Sistema Inteligente de Apoio à Decisão (SIAD), com uma taxa média de cumprimento de prazos de 96,9%

ao escalonar 396 dos 409 trabalhos planeados dentro do prazo. Nas cinco melhores instâncias, a taxa de cumprimento atinge 98%, evidenciando a robustez do GRASP em minimizar atrasos. A alocação inicial assegura a inclusão de 98,6% dos trabalhos solicitados, o que reflete uma elevada capacidade de resposta às necessidades dos clientes. Em termos computacionais, o tempo médio de processamento é de 22 minutos por instância, confirmando a viabilidade do SIAD para aplicações em ambientes de produção real.

À data de conclusão do projeto, o SIAD está pronto para ser integrado na interface do sistema atual. Inicialmente, será implementado no sistema de produção da INPLAS, com a possibilidade de, após adaptações mínimas, ser expandido para outras fábricas do Grupo Simoldes.

Palavras-chave: GRASP, Máquinas Paralelas Dedicadas, Múltiplas Alternativas, *Setups*, Recursos Adicionais, Máquinas Alternativas.

Abstract

INPLAS, a company in the Plastics division of the Simoldes Group and specialized in the injection of thermoplastic components, identified the need to enhance the Decision Support Intelligent System (SIAD) used by its Production Planning Department to address the scheduling problem for dedicated parallel machines with multiple alternatives, setups, and additional resources. To this end, a new SIAD was developed, incorporating the consideration of alternative machines and the application of a Greedy Randomized Adaptive Search Procedure (GRASP) metaheuristic, aiming to increase the flexibility and efficiency of planning and better meet production needs.

The planning process was initially studied in depth, covering both upstream and downstream stages, to identify key variables, the inputs, and the essential outputs for solving this problem. In this context, the characteristics and constraints of the process were thoroughly analysed, especially those associated with the limitations of the production system. With a detailed understanding of the problem, potential improvement opportunities were identified to increase both the adaptability of the system to the INPLAS production environment and the effectiveness of the results obtained by the SIAD. Thus, it was decided that the improvement to implement, aiming to enhance the SIAD's performance, would be the inclusion of alternative machines in the system, while the resolution of the scheduling problem would be tackled with a GRASP metaheuristic application.

Given the complexity of the scheduling problem for dedicated parallel machines with multiple alternatives, setups, and additional resources, classified as NP-Hard, this work develops an adapted GRASP metaheuristic, grounded in best practices recommended in the literature. The proposed approach offers five specific contributions. First, it adapts the reference models by introducing an innovative separation between allocation and sequencing processes, allowing these steps to be handled independently, which is rare in traditional approaches. Secondly, it incorporates common due dates, requiring the introduction of a delay variable for each job and a multi-objective function focused on minimizing the number of delayed jobs, ensuring compliance with established deadlines. Additionally, the GRASP metaheuristic was adapted to a continuous production system, where unfinished jobs from the previous day are sequenced in the first position of the next day, realistically reflecting the production environment. Finally, an original constructive heuristic was adjusted, using a unique coefficient to rank insertion positions for jobs in partial solutions, aiming not only to minimize delays but also to reduce the number of setups and the total makespan.

The results demonstrate the effectiveness of the GRASP metaheuristic in supporting the Decision Support System (SIAD), with an average deadline compliance rate of 96.2%, scheduling 393 out of 409 planned jobs on time. In the top five instances, the compliance rate reaches 97.2%, highlighting GRASP's robustness in minimizing delays. The initial allocation ensures the inclusion of 98.6% of requested jobs, reflecting a high response capacity to customers' needs. In terms of computational time, the average processing time is 22 minutes per instance, confirming the SIAD's feasibility for real production environments.

As of the project's completion, the SIAD is ready for integration with the current system interface. Initially, it will be implemented in INPLAS's production system, with the possibility of expansion to other Simoldes Group factories after minor adaptations.

Keywords: GRASP, Dedicated Parallel Machines, Multiple Alternatives, Setups, Additional Resources, Alternative Machines.

Agradecimentos

A execução deste projeto só foi possível devido a todos aqueles que, de forma direta ou indireta, deram o seu contributo e, por isso, gostaria de lhes expressar a minha profunda gratidão.

Em primeiro lugar, um sincero agradecimento ao meu orientador, o Professor Doutor Manuel Pereira Lopes, pela sua inestimável orientação e disponibilidade ao longo de todo este percurso. A sua experiência e conselhos foram essenciais para o desenvolvimento deste projeto.

Aos meus pais e família, devo um especial reconhecimento pelo apoio incondicional, carinho e por todas as oportunidades que me proporcionaram durante toda esta jornada. Sem a vossa constante motivação e paciência, este objetivo não teria sido possível. Obrigado por sempre acreditarem e exigirem o melhor de mim.

Aos meus amigos, agradeço pela amizade, pelas palavras de incentivo e por estarem sempre presentes nos momentos de desafio e conquista. A vossa companhia tornou este e todos os caminhos mais leves e gratificantes.

A todos, o meu sincero obrigado.

Índice

Lista de Figuras

Lista de Tabelas

Acrónimos e Símbolos

Lista de Acrónimos

CSH	Ciências Sociais e Humanas
UPMS	*Unrelated Parallel Machine Scheduling*
GRASP	*Greedy Randomized Adaptive Search Procedure*
VND	*Variable Neighbourhood Descent*
ACO	*Ant Colony Optimization*
MetaRaPS	*Metaheuristic for Randomized Priority Search*
SA	*Simulated Annealing*
RSA	*Reactive Simulated Annealing*
MILP	*Mixed-Integer Linear Programming*
IP	*Integer Programming*
CP	*Constraint Programming*
UPMSR	*Unrelated Parallel Machine Scheduling with Setup and Resource Constraints*
UPMSR-S	*Unrelated Parallel Machine Scheduling with Setup and Resource Constraints – Setup Resources*
UPMSR-P	*Unrelated Parallel Machine Scheduling with Setup and Resource Constraints – Processing Resources*
UPMSR-PS	*Unrelated Parallel Machine Scheduling with Setup and Resource Constraints – Processing and Setup Resources*
TPhA	*Two-Phase Algorithm*
ATCS	*Apparent Tardiness Cost with Setups*
GA	*Genetic Algorithm*
AIS	*Artificial Immune System*
CL	*Candidate List*
RCL	*Restricted Candidate List*
RPD	*Relative Percentage Deviation*
JIT	*Just in time*
DMI	*Dynamic Multiple Insertion*

1 Introdução

Neste capítulo será introduzido o projeto desenvolvido, contextualizando-o adequadamente nos desafios da indústria. Inicialmente, é apresentado o problema tratado no presente projeto, incluindo a sua contextualização e relevância. Em seguida, são delineados os objetivos a alcançar e a questão de investigação que o projeto pretende abordar. Por fim, no terceiro subcapítulo, são descritas as opções metodológicas consideradas para o desenvolvimento do projeto.

1.1 Problema de Investigação, Enquadramento e Pertinência

O Grupo Simoldes é um grupo empresarial português líder na produção de moldes e componentes plásticos, principalmente para a indústria automóvel, com presença global e grande foco em inovação e sustentabilidade. Fundado em 1959, o grupo é composto pela *Simoldes Tools*, especializada em moldes, e pela *Simoldes Plastics*, dedicada à produção de peças plásticas. A INPLAS é uma empresa do Grupo Simoldes divisão Plásticos, que se destaca na injeção de componentes termoplásticos, servindo para além de outros setores, o mercado automóvel (Inplas, 2024).

O compromisso do Grupo Simoldes com a inovação é evidenciado pelo contínuo investimento em investigação e desenvolvimento, procurando sempre soluções tecnológicas avançadas e sustentáveis que respondam às necessidades dos seus clientes globais. Neste contexto, embora o sistema de produção da INPLAS esteja dotado com um sistema inteligente de apoio à decisão que possibilita um planeamento ágil e eficiente em máquinas paralelas, este projeto surge com o objetivo de otimizar esse sistema, identificando oportunidades de melhoria e desenvolvendo novas funcionalidades no sistema atual.

O sistema de produção da INPLAS é constituído por trinta e três máquinas de injeção paralelas, que se diferenciam principalmente pela força de fecho que utilizam. O departamento de planeamento da produção da INPLAS tem como objetivo dar resposta aos pedidos diários dos

clientes, definindo a alocação dos moldes às máquinas disponíveis, garantindo o cumprimento das restrições de capacidade e elegibilidade de cada máquina. Além disso, é fundamental considerar as limitações associadas aos recursos adicionais, nomeadamente os operadores de máquina (necessários durante o processamento das peças), as equipas de *setup* (responsáveis pelas trocas de molde nas máquinas) e os moldes disponíveis.

O problema em questão inclui várias particularidades quando comparado com a maioria dos problemas de máquinas paralelas existentes na literatura, que ao serem consideradas, permitem uma maior conformidade com a realidade industrial em que o problema se insere. Em particular, considera-se a imposição de datas de entrega, o estado inicial de cada máquina e a existência de máquinas dedicadas, implicando que cada trabalho só possa ser alocado a um conjunto específico e restrito de máquinas.

Assim, o problema considerado neste estudo enquadra-se nos problemas de escalonamento em máquinas paralelas dedicadas com *setups* independentes da sequência, recursos adicionais de *setup* e de processamento e datas de entrega. A resolução da globalidade destes problemas divide-se em três subproblemas: alocação (consiste na afetação de cada trabalho à máquina que o vai processar), sequenciação (definição da ordem na qual os trabalhos serão processados em cada máquina), e tempo (definição do momento inicial e final de processamento de cada trabalho). Apesar dos problemas de escalonamento em máquinas de paralelas dedicadas não necessitarem, geralmente, do subproblema de alocação, o presente problema apresenta a particularidade de cada referência poder ser produzida na máquina principal, numa primeira máquina alternativa ou numa segunda máquina alternativa, exigindo, por isso, a inclusão deste subproblema.

Este problema representa um desafio real e complexo. Segundo (Garey et al., 1979), o problema de escalonamento em duas máquinas paralelas, considerando apenas *setups* com o objetivo de minimizar o *makespan*, é classificado como NP-hard. Com base nesta classificação, podemos também considerar o problema em questão como NP-hard. Dado que não existe um algoritmo que consiga resolver eficientemente todas as instâncias deste problema em tempo útil, optou-se por utilizar a metaheurística GRASP para a sua resolução.

A pertinência deste estudo alinha-se com as necessidades do Grupo Simoldes em introduzir inovações que conduzam a uma sustentabilidade global, aumentem a competitividade e promovam novos conhecimentos, nomeadamente através da criação de ferramentas inteligentes baseadas em técnicas de otimização que apoiem o processo de tomada de decisão.

1.2 Questão e Objetivos de Investigação

Considerando a problemática anteriormente descrita, este estudo tem como objetivo responder às seguintes questões:

- De que forma a inclusão do subproblema de alocação dos trabalhos a uma das máquinas alternativas disponíveis pode melhorar o sistema inteligente de apoio à decisão atualmente em vigor e otimizar a sua adaptabilidade às necessidades do departamento de planeamento de produção da INPLAS?

- Qual o grau de adequabilidade da aplicação da metaheurística GRASP a um problema de escalonamento em máquinas paralelas dedicadas com *setups*, recursos adicionais e datas de entrega?

Em resposta às questões de investigação, estabelece-se como objetivo geral o desenvolvimento de uma metaheurística GRASP para o problema de escalonamento em máquinas paralelas dedicadas, levando em consideração *setups*, recursos adicionais e datas de entrega. Para alcançar este objetivo geral, são definidos os seguintes objetivos específicos:

- Identificação de oportunidades de melhoria existentes na do sistema inteligente de apoio à decisão atualmente em vigor que impactem positivamente a adaptabilidade do mesmo às necessidades da INPLAS e a qualidade das soluções obtidas;

- Desenvolvimento e implementação, em *Python*, de um modelo de alocação de trabalhos a máquinas elegíveis, utilizando a biblioteca *Pulp*;

- Adaptação e extensão das metaheurísticas GRASP de (Yepes-Borrero et al., 2020) e (Lopez-Esteve et al., 2023) para o caso particular do escalonamento em máquinas paralelas dedicadas com *setups*, recursos adicionais e datas de entrega;

- Implementação da metaheurística GRASP em *Python*;

- Estudar o desempenho da metaheurística GRASP;

1.3 Opções Metodológicas

"Como nos restantes campos e áreas do saber, a investigação em CSH carateriza-se quer pela multiplicidade quer pela dependência contextual." (Pereira Coutinho, 2014). Neste contexto, e com o objetivo de caracterizar este estudo em termos de multiplicidade, são apresentados neste capítulo os métodos e abordagens utilizados para explorar de forma adequada as questões de pesquisa formuladas.

As opções metodológicas consideradas foram organizadas de acordo com a hierarquia proposta por (Pereira Coutinho, 2014), que descreve as técnicas, os métodos e as metodologias, como "três conceitos cujas fronteiras se tocam, com níveis de generalidade crescente". Deste modo,

no lugar mais baixo da hierarquia, definem-se as técnicas como procedimentos práticos concretos utilizados na investigação. Em seguida, os métodos são concebidos como conjuntos estruturados dessas técnicas, orientados por um objetivo científico específico. Por fim, a metodologia é entendida como a análise crítica dos métodos aplicados, refletindo sobre os seus fundamentos teóricos e a sua capacidade para gerar conhecimento.

A maioria dos autores, incluindo (Pereira Coutinho, 2014) e (Stockemer, 2018) concorda com a existência de duas principais perspetivas metodológicas: quantitativa e qualitativa. Por um lado, a perspetiva quantitativa caracteriza-se pela utilização de métodos estatísticos e matemáticos para medir variáveis, testar hipóteses, identificar padrões e estabelecer relações de causa e efeito de forma a "gerar a construção de contributos teóricos explicativos do fenômeno em questão" (Traqueia et al., 2021). Por outro lado, a perspetiva qualitativa, em vez de quantificar dados, foca-se na análise descritiva e interpretativa, procurando compreender a complexidade e a riqueza dos fenómenos a partir de perspetivas subjetivas. Conclui-se então, que presente estudo utiliza a metodologia quantitativa, uma vez que os objetivos propostos passam pela medição precisa das variáveis envolvidas e pela aplicação de métodos estatísticos para testar hipóteses, identificar padrões e estabelecer relações de causa e efeito. Esta escolha justifica-se ainda pela necessidade de quantificar fenómenos específicos e de generalizar os resultados para uma população mais ampla, assegurando, assim, a replicabilidade e a fiabilidade dos resultados obtidos.

Relativamente à estratégia de investigação, de acordo com (Saunders et al., 2023), de entre as principais estratégias de investigação destacam-se, entre outros: Investigação experimental, inquéritos, estudo de caso, teoria fundamentada (*grounded theory*), etnografia, investigação arquivística, investigação narrativa e investigação -ação. Neste estudo, foi adotada a estratégia de investigação-ação, que envolve uma investigação prática realizada em organizações com o objetivo de resolver problemas concretos e reais. Nesta estratégia, o investigador participa ativamente no processo, colaborando com os membros da organização na implementação e avaliação de medidas para solucionar problemas identificados, promovendo um ciclo contínuo de pesquisa e ação.

2 Revisão bibliográfica

2.1 Problema de Escalonamento em Máquinas Paralelas

O principal desafio da teoria do escalonamento em máquinas múltiplas é providenciar a combinação perfeita, ou praticamente perfeita, entre máquinas e trabalhos e consequentemente determinar a sequência de trabalhos em cada máquina de forma a atingir os objetivos propostos (Cheng & Sin, 1990). Apesar de existir uma grande variedade de tipologias de problemas de escalonamento, os problemas de escalonamento em máquinas paralelas são uma das tipologias mais estudadas na literatura (Edis et al., 2013).

De acordo com (Mokotoff, 2001) no problema clássico de escalonamento em máquinas paralelas existem n trabalhos e m máquinas, sendo que cada trabalho tem de ser executado por uma das máquinas durante um tempo fixo de processamento. O objetivo é encontrar o escalonamento que otimiza um determinado indicador de desempenho.

Adicionalmente, (Błażewicz et al., 2001) referem-se aos problemas de escalonamento em máquinas paralelas como relações entre três entidades: os trabalhos a ser processados, as máquinas disponíveis para o processamento dos trabalhos e os eventuais recursos adicionais necessários ao seu processamento. Neste sentido, o escalonamento, consiste na alocação das máquinas e, possivelmente, dos recursos adicionais aos trabalhos de forma a completar a produção o seu processamento respeitando, em simultâneo, as restrições existentes.

As abordagens aos problemas de escalonamento em máquinas paralelas variam de acordo com a classificação da tipologia de máquinas que o compõe. Neste sentido, (Błażewicz et al., 2001) classificam as máquinas como podendo ser paralelas, desempenhando as mesmas funções ou dedicadas, quando são especializadas na execução de determinados trabalhos. As máquinas paralelas podem ainda ser classificadas de acordo com a sua velocidade:

- Idênticas: Se todas as máquinas do conjunto têm a mesma velocidade de processamento para cada trabalho;

- Uniformes: Se as máquinas apresentam velocidades diferentes, mas a velocidade de cada máquina é constante e não depende do trabalho a processar (nestes casos o tempo de processamento do trabalho é diretamente proporcional à velocidade da máquina e é possível calculá-lo caso se conheça a velocidade da máquina);

- Não-relacionadas: Se a velocidade de processamento de um trabalho depende exclusivamente do trabalho e da máquina onde é processada, ou seja, cada trabalho tem uma velocidade de processamento pré-definida para cada máquina;

- Dedicadas: cada trabalho só pode ser processada num subconjunto de máquinas predefinido, extinguindo-se o subproblema de alocação.

2.2 Setups

"As máquinas têm frequentemente de ser reconfiguradas ou limpas entre trabalhos. Estas tarefas designam-se por mudanças ou *setups*" (Pinedo, 1999) . A competitividade da economia global atual força as empresas a serem competitivas para garantirem a sua sobrevivência, tendo de aumentar a sua produtividade, eliminar o desperdício e as atividades que não geram valor, e aumentar a eficiência da utilização dos recursos (Allahverdi, 2015).

Com efeito, pode-se considerar que eliminar ou procurar reduzir o número de *setups*, deve estar presente nos objetivos de qualquer empresa que procure ser competitiva, uma vez que estas operações, para além de não gerarem valor consomem tempo de processamento precioso, e ainda acarretam custos associados à mão de obra e desperdício de matérias-primas (Pinedo, 1999). No entanto, apesar do seu impacto na produtividade das empresas, de acordo com (Allahverdi et al., 2008), a maioria dos artigos de planeamento e escalonamento de operações, assume os tempos e custos de *setup* como negligenciáveis ou como parte dos tempos e custos de processamento.

De forma a compreender a importância e a influência da inclusão de *setups* no estudo de problemas de escalonamento em máquinas paralelas pode-se analisar a exaustiva revisão bibliográfica deste tipo de problemas em (Allahverdi et al., 1999, Allahverdi et al., 2008 e Allahverdi, 2015) onde os autores analisam os estudos publicados sobre o tema entre 1960 (data da análise do primeiro artigo publicado sobre o escalonamento em máquinas paralelas com *setups*) e 2014, analisando, ao todo, cerca de 1000 publicações.

De acordo com os mesmos autores, os *setups*, podem ser abordados de duas formas distintas nos problemas UPMS (*Unrelated Parallel Machine Scheduling*):

- Dependentes da sequência: o tempo de *setup* entre dois trabalhos varia de acordo com o trabalho que foi realizado anteriormente e do trabalho que será realizado após a operação de *setup*;

- Independentes da sequência: o tempo de *setup* é fixo para cada trabalho, independentemente do trabalho anterior.

Desta forma, *setups* dependentes da sequência tornam o problema de escalonamento mais complexo, pois a ordem de execução dos trabalhos tem um impacto direto no tempo total de *setup*. Isso significa que é necessário considerar a melhor sequência de trabalhos para minimizar o tempo total de *setup* e, consequentemente, o tempo total de processamento.

Por fim, na Figura 1 podemos encontrar, de forma resumida a classificação dos problemas de escalonamento de máquinas paralelas com *setups* com base na notação de três fatores α/β/γ. Nesta classificação introduzida por (Graham et al., 1979), o primeiro fator, α, descreve a configuração das máquinas, o segundo fator, β, descreve a informação dos trabalhos (neste caso em relação aos *setups*), outras condições do ambiente de produção e características do processo, e fator γ define o indicador de desempenho considerado para avaliar as soluções do problema.

α		β		γ	
Notation	Description	Notation	Description	Notation	Description
1	Single machine	ST_{si}	Sequence-independent setup time	C_{max}	Makespan
P	Parallel machines (identical)	SC_{sd}	Sequence-dependent setup cost	E_{max}	Maximum earliness
Q	Parallel machines (uniform)	ST_{sd}	Sequence-dependent setup time	L_{max}	Maximum lateness
R	Parallel machines (unrelated)	$ST_{si,f}$	Sequence-independent family setup time	T_{max}	Maximum tardiness
Fm	m-stage flowshop	$ST_{sd,f}$	Sequence-dependent family setup time	D_{max}	Maximum delivery time
FFm	m-stage flexible (hybrid) flowshop	$SC_{sd,f}$	Sequence-dependent family setup cost	TSC	Total setup/changeover cost
AFm	m-stage assembly flowshop	ST_{psd}	Past-sequence-dependent setup time	TST	Total setup/changeover time
J	Job shop	$Prec$	Precedence constraints	TNS	Total number of setups
FJ	Flexible job shop	r_j	Non-zero release date	ΣF_j	Total flowtime
O	Open shop			ΣC_j	Total completion time
				ΣE_j	Total earliness
				ΣT_j	Total tardiness
				ΣW_j	Total waiting time
				ΣU_j	Number of tardy (late) jobs
				$\Sigma w_j C_j$	Total weighted completion time
				$\Sigma w_j F_j$	Total weighted flowtime
				$\Sigma w_j U_j$	Weighted number of tardy jobs
				$\Sigma w_j E_j$	Total weighted earliness
				$\Sigma w_j T_j$	Total weighted tardiness
				$\Sigma w_j W_j$	Total weighted waiting time
				$\Sigma h(E_j)$	Total earliness penalties
				$\Sigma h(T_j)$	Total tardiness penalties
				$TADC$	Total absolute differences in completion times

Figura 1 – Classificação de três fatores α/β/γ dos problemas UPMS

2.2.1 Problema de Escalonamento em Máquinas Paralelas com *Setups*

Os autores do artigo (Diana et al., 2015) propõem estudar o problema de escalonamento em máquinas paralelas não relacionadas, com tempos de *setup* dependentes da sequência de fabrico, com o objetivo de minimização do *makespan*. Com efeito, apresentam um algoritmo inspirado em sistemas imunológicos, baseado na seleção clonal. O algoritmo desenvolvido utiliza a fase de construção da metaheurística GRASP para gerar a população inicial, uma função de avaliação para evitar ótimos locais, a pesquisa local VND como operador de hipermutação somática e um operador de reseleção para manter a qualidade e diversidade das soluções, todas elas particularidades que o distinguem de outros existentes na literatura. Os autores concluem que o algoritmo é eficaz na geração de soluções de alta qualidade para o problema de minimização do *makespan* e ainda que os resultados dos testes realizados indicam que o algoritmo proposto supera outros métodos existentes, especialmente em instâncias com um maior número de máquinas: o *Genetic Algorithm* desenvolvido em (Vallada & Ruiz, 2011), a metaheurística *Ant Colony Optimization* apresentada em (Arnaout et al., 2014) e *Simulated Annealing* representado em (Ying et al., 2012).

Em (Vallada & Ruiz, 2011), os autores abordam também o problema de escalonamento em máquinas paralelas não relacionadas com tempos de preparação dependentes da sequência, visando minimizar o *makespan*. Este estudo, envolve o desenvolvimento de um algoritmo genético, que utiliza operadores de cruzamento e mutação específicos para lidar com a complexidade dos tempos de preparação. Neste sentido, os parâmetros utilizados no algoritmo são considerados como os principais responsáveis para a eficácia dos mesmos e foram calibrados através de técnicas de experimentação fatorial para otimizar o seu desempenho. Os autores concluem que o algoritmo é uma abordagem robusta e eficiente para este tipo de problema de escalonamento, apesar, de não ser a metodologia que apresenta melhores resultados.

Outra abordagem para a resolução deste problema é apresentada em (Arnaout et al., 2014), onde os autores desenvolvem um *Ant Colony Optimization Algorithm* (ACO) em duas fases, denominado ACO II, melhorando o algoritmo anterior (ACO I) introduzido pelos mesmos autores em . O algoritmo ACO II é projetado para resolver problemas de escalonamento complexos dividindo-o em subproblemas: a atribuição de trabalhos às máquinas e o sequenciamento dos trabalhos atribuídos. O estudo desenvolvido conclui que "os testes extensivos e expandidos realizados provam a superioridade do ACO II aprimorado em comparação com algoritmos existentes, incluindo ACO I, MetaRaPS e SA".

Por fim, no artigo de (Ying, Lee & Lin, 2012), os autores estudam o mesmo problema e desenvolvem um algoritmo *Restricted Simulated Annealing* (RSA) que incorpora uma estratégia de procura limitada para otimizar a procura por soluções vizinhas, eliminando movimentos ineficazes no espaço de soluções. Neste sentido, algoritmo RSA é projetado com o objetivo de reduzir o esforço computacional e melhorar a de forma eficaz a qualidade das soluções, especialmente em problemas de grande escala. Os resultados computacionais obtidos neste estudo, indicam que o algoritmo RSA proposto apresenta resultados semelhantes aos obtidos através das metaheurísticas existentes na literatura para instâncias de pequena dimensão, mas apresenta resultados significativamente melhores do que os modelos SA e outras metaheurísticas existentes para instâncias de grande dimensão.

2.3 Recursos

“A maioria dos estudos relacionados com escalonamento em máquinas paralelas considera as máquinas como os únicos recursos. No entanto, na maioria dos ambientes industriais reais, os trabalhos podem necessitar de recursos adicionais, como operadores de máquinas, ferramentas, paletes, moldes, ou robôs industriais, para o seu processamento” (Edis et al., 2013). Isto é, embora, conforme indicado em (Yepes-Borrero et al., 2020), haja consideravelmente menos estudos relacionados com problemas de escalonamento que considerem recursos adicionais, estes representam uma área de estudo com muito significado.

“Os recursos adicionais, de acordo com a sua natureza e utilização, podem ser classificados em classes, categorias e tipos” (Blazewicz et al., 2004). Em relação à divisão em classes, de acordo com os autores deste artigo, os recursos podem ser classificados como:

- Recursos de processamento: quando os recursos são utilizados em conjunto com a máquina, como as ferramentas, por exemplo;

- Recursos input-output: quando os recursos não são utilizados em conjunto com a máquina, mas sim antes ou depois do processamento de um trabalho.

Adicionalmente, a classificação em categorias engloba dois pontos de vista. Por um lado, diferenciam-se quanto à sua renovabilidade (Blazewicz et al., 2007; Slowinski, 1980):

- Recurso renovável: a sua utilização total está apenas limitada em cada instante de tempo, ou seja, se este recurso for utilizado num instante de tempo, pode voltar a ser utilizado em instantes seguintes, caso já tenha sido libertado das tarefas pelas quais estava a ser usado. As máquinas, os operadores e ferramentas, são exemplos destes recursos.

- Recurso não renovável: a utilização total deste recurso é limitada no horizonte temporal do problema, ou seja, de cada vez que é utilizado por uma tarefa não pode ser associado a nenhuma tarefa seguinte, uma vez que foi consumido. Dinheiro e matéria-prima, são exemplos deste recurso.

- Recurso duplamente limitado: a utilização deste recurso é limitada em cada instante de tempo e consumível, sendo a sua utilização também limitada no horizonte temporal do problema.

Por outro lado, diferenciam-se da seguinte forma quanto à sua divisibilidade (Blazewicz et al., 2007; Slowinski, 1980):

- Recurso discreto: apenas pode ser alocado em quantidades discretas;

- Recurso contínuo: pode ser alocado em quantidades arbitrárias.

Por fim, a classificação dos recursos quanto ao tipo apenas diz respeito às funções que os recursos cumprem, sendo que recursos do mesmo tipo são supostos de realizar as mesmas funções (por exemplo, os recursos de *setup* são supostos de serem utilizados em operações de *setup*)

2.3.1 Problema de Escalonamento em Máquinas Paralelas com Recursos

A consideração de recursos adicionais nos problemas de escalonamento acrescentam complexidade ao problema. Para além do objetivo de se obter uma alocação e sequenciamento eficientes, é ainda necessário realizar conjugações de trabalhos possíveis que garantam que em cada instante de tempo, os trabalhos em processamento não excedem o limite de recursos adicionais disponíveis (Pinto & Grossmann, 1997). A Figura 2 representa de forma visual a influência que a consideração de recursos adicionais pode ter nos problemas de escalonamento, sendo que muitas das vezes, o impacto da inclusão de restrições de recursos adicionais resulta em tempos inativos por falta de recursos num determinado instante de tempo.

Conclui-se então que problemas de escalonamento em máquinas paralelas com recursos englobam a resolução de três tipos de subproblemas (Ferreira, 2023):

- Alocação: afeta-se os trabalhos/recursos às máquinas;
- Sequenciação: estabelece-se a ordem de execução dos trabalhos/recursos;
- Tempo: determina-se especificamente o instante inicial e final de cada trabalho/recurso.

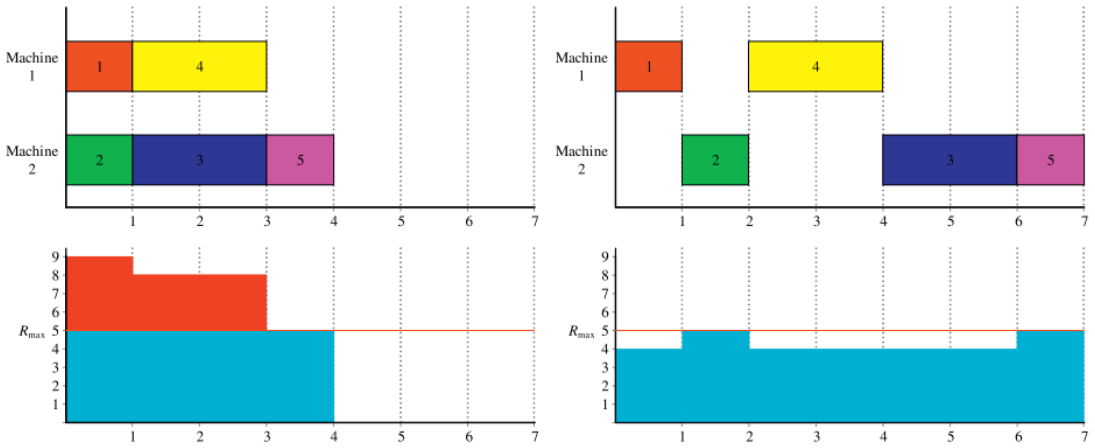


Figura 2 – Impacto da consideração de recursos adicionais nos problemas de escalonamento

O problema de escalonamento em máquinas paralelas sem considerar recursos adicionais pode ser uma limitação na aproximação a problemas reais. Neste sentido, este tipo de problemas tem sido amplamente estudados nos últimos anos.

No artigo de (Fanjul-Peyro et al., 2017) os autores analisam um problema de escalonamento em máquinas paralelas onde o processamento de trabalhos requer a utilização de um recurso escasso renovável. A quantidade utilizada deste recurso depende do trabalho e da máquina. Os

autores abordam o problema através de dois modelos matemáticos MILP (*Mixed-Integer Linear Programming*): Um modelo baseado num já existente na literatura e outro original desenvolvido pelos autores, tendo por base um modelo de *strip-packing*. Uma vez que para instâncias de tamanho médio os modelos matemáticos não encontram soluções ótimas, os autores propõe três heurísticas matemáticas (combinação entre metaheurísticas e modelos de programação matemáticas) para a resolução dos dois modelos: *machine-assignment fixing*, *job-machine reduction* e *greedy-based fixing.* Este estudo conclui que que os modelos desenvolvidos com base no *strip-packing*, apresentam gradualmente melhores resultados do que o modelo baseado na literatura à medida que o tamanho dos problemas aumenta. Uma vez que o problema abordado é classificado como NP-Hard, as heurísticas matemáticas superam modelos MILP em instâncias de média dimensão e ainda apresentam resultados não muito diferentes dos mesmos para instâncias de pequena dimensão. Para instâncias de grande dimensão (50 trabalhos e 15 máquinas), os autores concluem que é necessário mais investigação, uma vez que as heurísticas matemáticas propostas nem sempre apresentam soluções satisfatórias.

Um outro artigo de (Villa et al., 2018) analisa a performance de duas abordagens distintas aplicadas ao problema de escalonamento em máquinas paralelas não-relacionadas com um recurso adicional escasso, com o objetivo de minimização de *makespan*. Para as duas abordagens são testadas várias heurísticas, sendo que a primeira abordagem considera as restrições de recursos durante toda a fase construtiva, enquanto a segunda abordagem define a solução construtiva sem considerar as restrições de recursos , reparando-a, posteriormente, nos momentos que não respeitam as limitações existentes de forma a obter-se uma solução viável. São criadas duas heurísticas que seguem a primeira abordagem e cinco heurísticas que seguem a segunda, sendo que as heurísticas utilizadas divergem em relação à ordenação aplicada aos trabalhos da lista de trabalhos pendentes antes de se iniciar a fase construtiva. Neste sentido, a ordem dos trabalhos varia quanto ao tipo de ordenação dos trabalhos (ordem crescente ou ordem decrescente) e no critério de ordenação escolhido (maior valor de processamento, menor valor de processamento, maior valor de recursos consumidos ou menor valor de recursos consumidos). Os autores concluem que duas das heurísticas (*M4* e *M5*) que seguem a segunda apresentam o melhor desempenho, uma vez que os resultados obtidos são iguais ou melhores do que os obtidos pelas restantes heurísticas e, em termos computacionais, utilizam muito menos tempo computacional para a obtenção de soluções.

Por fim, (Edis & Ozkarahan, 2012), aborda o problema de escalonamento em máquinas paralelas com recursos adicionais e elegibilidade de máquinas, aplicado a um cenário real da indústria de injeção de moldes. Os autores começam por testar um modelo matemático de programação inteira com o objetivo de minimização do *makespan*. No entanto, tendo em conta o elevado número de variáveis incluídas no modelo, este não apresenta resultados satisfatórios. Neste sentido, testam-se duas novas abordagens que consistem na partição do problema original em duas fases distintas: a fase de alocação e a fase de sequenciação. As duas abordagens utilizam modelos de programação inteira na fase de alocação, procurando minimizar a carga máxima de máquinas e operadores, sendo que na fase de sequenciação, a primeira abordagem volta a utilizar um modelo de *Integer Programming* (IP), enquanto a segunda abordagem usa um

modelo de *Constraint Programming* (CP). Concluiu-se que, globalmente, apesar dos resultados obtidos pelas duas abordagens não serem ótimos, produzem resultados eficientes num espaço de tempo razoável e apresentando valores de GAP relativamente baixos. Por fim, uma análise mais detalhada concluiu que a abordagem IP/IP produz melhores resultados para situações com maior número de operadores, enquanto a abordagem IP/CP produz resultados com menor tempo computacional e devolve soluções mais eficientes nos casos em que as restrições de recursos são mais "apertadas".

2.4 Problemas de Escalonamento em Máquinas Paralelas com Setups e Recursos Adicionais

Um cenário mais complexo do que os problemas que consideram independentemente *setups* ou recursos adicionais é aquele que reflete as realidades industriais onde ambos são considerados em conjunto. Neste tipo de problemas, se as máquinas paralelas envolvidas utilizarem recursos, quer durante a produção, quer durante as operações de *setup*, tanto os trabalhos como as operações de *setup* terão de aguardar até que os recursos necessários estejam disponíveis para serem iniciados.

De acordo com (Fanjul-Peyro, 2020), os problemas de escalonamento em máquinas paralelas não-relacionadas com *setups* e recursos adicionais podem ser denominados por UPMSR e embora este tipo de problema seja raramente estudado, existem diversos exemplos reais em que estes dois cenários são considerados simultaneamente, justificando a sua pertinência. Para além disso, os autores dividem ainda os problemas UPMSR em três outros problemas, de acordo o tipo de recursos considerados, podendo ser problemas UPMSR-S (só consideram recursos de *setup*), UPMSR-P (só consideram recursos de processamento) ou UPMSR-PS (consideram recursos de processamento e *setup*, em simultâneo).

2.4.1 Casos Práticos

Analisando o estudo conduzido por (Fanjul-Peyro, 2020) , o autor apresenta dois modelos MILP, um baseado em problemas de *strip-packing* e outro baseado em índices de tempo (*time index*), para resolver um problema de escalonamento em máquinas paralelas com *setups* e recursos (UPMSR), com o objetivo de minimizar o *makespan*. De acordo com o autor, nas versões mais completas dos problemas UPMSR, é necessário considerar seis fatores: os tempos de processamento, os tempos de *setup*, os recursos utilizados no processamento (*P-Resources*), os recursos utilizados nas operações de *setup* (*S-Resources*), os recursos partilhados utilizados no processamento (*HP-Resources*) e os recursos partilhados utilizados nas operações de *setup* (*HS-Resources*). O MILP baseado em *strip-packing*, apresenta melhores resultados

comparativamente ao MILP baseado em *time index*, apesar de os dois só devolverem soluções para instâncias pequenas. Nesse sentido, o autor propõe um algoritmo exato de três fases (*Three Phase Exact Algorithm*, TPhA) que, como o próprio nome indica, resolve o problema em três fases distintas: a alocação de cada trabalho a uma máquina, a sequenciação do conjunto de trabalhos em cada máquina e a definição do instante temporal em que cada trabalho e operação de *setup* se iniciam e terminam. O TPhA obtém soluções de qualidade idêntica às obtidas pelos MILP para instâncias pequenas (com um GAP de 6,94%) e, à medida que o número de trabalhos aumenta, a qualidade das soluções melhora, apresentando um *GAP* médio de 6,94% para instâncias médias e 5,09% para instâncias grandes.

Em (Bektur & Saraç, 2019), os autores consideram um problema de escalonamento em máquinas paralelas com um *common server*. Os problemas com *common servers* envolvem a utilização de servidores para realizar operações de *setup* e, neste caso específico, é considerado apenas um servidor (uma equipa única de *setup* que trabalha em conjunto para realizar as operações de *setup*), o que implica que apenas uma operação de *setup* pode ser realizada em cada instante. Além disso, o problema considera *setups* dependentes da sequência e restrições de elegibilidade de máquinas. O indicador de desempenho utilizado é a minimização do atraso ponderado total. Para resolver o problema, os autores desenvolvem um modelo MILP, que, devido à complexidade *NP-hard* do problema, apenas consegue apresentar soluções ótimas num intervalo de tempo de 18.000 segundos para instâncias com duas máquinas e dez trabalhos, e soluções viáveis para instâncias com duas máquinas e vinte trabalhos, três máquinas e quinze trabalhos, ou cinco máquinas e vinte e cinco trabalhos. Para combater a ineficiência do modelo MILP, os autores propõem o uso de metaheurísticas *Tabu Search* e *Simulated Annealing* para obter soluções para instâncias de maiores dimensões. O estudo conclui que de acordo com as comparações experimentais, o algoritmo *Tabu Search*, com memória de longo prazo, e um mecanismo de procura de solução inicial baseado na regra ATCS modificada (regra para obtenção de soluções iniciais com base nos custos dos atrasos) produz melhores resultados do que os outros métodos heurísticos.

Em (Ozer et al., 2019), os autores abordam o problema de escalonamento em máquinas paralelas com *setups* dependentes da sequência, restrições de elegibilidade e cópias múltiplas de recursos partilhados (apenas são considerados recursos de processamento). De acordo com os autores, se o recurso partilhado for único, apenas pode ser utilizado por um trabalho em simultâneo, no entanto, neste caso, podem ser processados em simultâneo um número de trabalhos igual ao número de cópias de recursos partilhados existente. Neste estudo são apresentados dois modelos MILP (M1 e M2) que apresentam soluções ótimas para as instâncias de pequena dimensão, destacando-se o modelo M2 na obtenção da solução ótima num menor tempo computacional. Para instâncias de média dimensão, com 40 trabalhos, o modelo M2 resolve, em 7200 segundos, apenas 30% das instâncias, apresentando, no entanto, soluções de boa qualidade (GAP de 0.72). Por outro lado, o modelo M1 resolve 80% das instâncias de média dimensão apesar de devolver soluções de menor qualidade (GAP de 0.99). Nenhum dos dois modelos MILP tem capacidade de devolver soluções para instâncias grandes. Para uma resolução mais eficaz das instâncias de média e grande dimensão os autores propõe uma

heurística matemática baseada num *Genetic Algortihm* (GA) que apresenta resultados notoriamente melhores para problemas com 40 e 100 trabalhos.

Em (Yunusoglu & Topaloglu Yildiz, 2022), os autores estudam um problema de escalonamento em máquinas paralelas não-relacionadas, com restrições de recursos. A função objetivo é de minimização do *makespan* e de forma a refletir os ambientes de produção de forma real, são incluídos *setups* dependentes da sequência, relações de precedência, restrições de elegibilidade de máquinas e datas de lançamento. Para a resolução deste problema, os autores apresentam um modelo de solução exata baseado no *Constraint Programming* (CP). Ao CP apresentado são adicionadas restrições de *lower bound* e restrições redundantes para o enriquecer e duas estratégias de ramificação com o objetivo de reduzir o seu tempo computacional: ordenação de variáveis e ordenação de valores. Os autores concluem que o CP proposto supera as alternativas de *Integer Programming* (IP) e as metaheurísticas *Genetic Algorithm* (GA) e os algoritmos *Artificial Immune System* (AIS) existentes na literatura, com um GAP médio de 15.52% quando comparado com os algoritmos AIS para instâncias de grande dimensão.

2.5 Greedy Randomized Adaptive Search Procedure

O *Greedy Randomized Adaptive Search Procedure* (GRASP), introduzido por (Feo & Resende, 1989), é um algoritmo metaheurístico amplamente utilizado na resolução de problemas de otimização combinatória. A aplicação do GRASP é particularmente útil em situações nas quais é computacionalmente difícil (ou inviável) encontrar uma solução exata, devido ao tamanho ou complexidade do espaço de soluções. Para além disso, a versatilidade do GRASP é extensamente reconhecida na literatura, sendo possível encontrar documentadas diversas áreas de aplicação. Por exemplo, em (Alvarez-Valdes et al., 2008) aplica-se o GRASP ao escalonamento de projetos com recursos parcialmente renováveis, enquanto em (Parreño et al., 2010) utiliza-se o algoritmo na resolução de problemas de *bin-packing* com duas e três dimensões. O último exemplo encontra-se em (Anticona, 2006), onde se aplica o GRASP ao contexto industrial, para resolver problemas de escalonamento de tarefas dependentes das máquinas em que são processadas e máquinas paralelas não-relacionadas, com o objetivo de minimizar o *makespan*. Estes exemplos evidenciam, de forma clara, a robustez e adaptabilidade do GRASP em diferentes contextos e problemas de otimização.

Analisando mais aprofundadamente o algoritmo GRASP, de acordo com (Resende & Ribeiro, 2008), este é caracterizado como uma metaheurística *multi-start*, na qual são executados múltiplos algoritmos de pesquisa local, a partir de diferentes soluções iniciais, para resolver problemas de otimização combinatória. De acordo com os autores, cada iteração do GRASP é composta por duas fases principais:

- Fase construtiva: nesta fase, o algoritmo constrói iterativamente uma solução viável, elemento a elemento, de modo a aproximar-se de uma solução inicial que possa ser melhorada posteriormente;

- Fase de pesquisa local: após a construção da solução inicial, esta fase explora a vizinhança da solução obtida, realizando ajustes até encontrar um mínimo local (em problemas de minimização) ou um máximo local (em problemas de maximização).

No entanto, em determinados casos, o algoritmo GRASP pode incluir uma terceira fase, nomeadamente quando a solução gerada na fase construtiva não cumpre todas as restrições do problema, tornando-se, assim, uma solução inviável. Nestes cenários, é necessário aplicar um procedimento de reparação, que vai ajustar as soluções obtidas de forma a troná-las válidas. Após a fase de pesquisa local, a solução obtida, que representa o melhor resultado local (mínimo ou máximo, conforme o tipo de problema), é guardada, e o processo recomeça com a geração de uma nova solução inicial. Este ciclo repete-se por um número predefinido de iterações, e, no final, a solução final para o problema é a melhor solução encontrada ao longo de todas as iterações. A Figura 3 ilustra o algoritmo GRASP na sua forma mais básica, conforme descrito por (Feo & Resende, 1995).

```
procedure grasp()
1    InputInstance();
2    for GRASP stopping criterion not satisfied →
3        ConstructGreedyRandomizedSolution(Solution);
4        LocalSearch(Solution);
5        UpdateSolution(Solution,BestSolutionFound);
6    rof;
7    return(BestSolutionFound)
end grasp;
```

Figura 3 – Pseudocódigo geral do algoritmo GRASP

Na fase construtiva (Figura 4), constrói-se uma solução de forma iterativa, adicionando a uma solução parcial elemento a elemento. A forma como cada elemento é adicionado à solução em cada iteração construtiva é determinada pela ordenação de todos os elementos que ainda não fazem parte da solução numa lista de candidatos (*Candidate List*), segundo uma função gananciosa (*greedy*) predefinida.

Pode-se afirmar que a fase construtiva do GRASP é adaptativa, pois, após cada iteração, a função que determina o benefício de adicionar cada elemento *s* à solução parcial é recalculada, ajustando-se ao novo contexto da solução parcial. Após a formação da Lista de Candidatos, é criada uma Lista Restrita de Candidatos (*Restricted Candidate List*), composta por uma quantidade fixa ou uma percentagem dos elementos da CL que apresentam os melhores valores de adequabilidade à solução.

A característica probabilística do GRASP surge na seleção do candidato a ser incluído na solução a cada iteração, que não é necessariamente o melhor candidato da RCL, mas sim um candidato escolhido aleatoriamente entre os presentes na RCL. Esta abordagem torna o GRASP um método

multi-start, uma vez que a seleção aleatória dos elementos a incluir na solução reduz a probabilidade de gerar duas soluções iguais. Os parâmetros do GRASP incluem o tamanho da RCL, a função que define a adequabilidade de cada elemento à solução em cada iteração, e a distribuição de probabilidade usada para selecionar os elementos da RCL.

```
procedure ConstructGreedyRandomizedSolution(Solution)
1    Solution = {};
2    for Solution construction not done →
3        MakeRCL(RCL);
4        s = SelectElementAtRandom(RCL);
5        Solution = Solution ∪ {s};
6        AdaptGreedyFunction(s);
7    rof;
end ConstructGreedyRandomizedSolution;
```

Figura 4 – Pseudocódigo da fase construtiva do algoritmo GRASP

As soluções geradas pela fase construtiva nem sempre constituem ótimos locais. Neste sentido, quase sempre vantajoso aplicar um procedimento de pesquisa local para explorar a vizinhança da solução inicial, tentando melhorar a solução obtida na fase construtiva e, se possível, encontrar um ótimo local.

O procedimento de pesquisa local funciona introduzindo pequenas alterações na solução inicial, seja de forma aleatória ou mais direcionada, com o objetivo de convergir mais rapidamente para um ótimo local. Cada solução ligeiramente alterada é avaliada de acordo com critérios de desempenho predefinidos e, se apresentar melhor desempenho do que a melhor solução encontrada até então, é adotada como a nova solução atual. Este processo de pesquisa local, representado na Figura 5, termina quando não são encontradas soluções melhores na vizinhança ou quando é atingido o critério de paragem.

```
procedure local(P,N(P),s)
1    for s not locally optimal →
2        Find a better solution t ∈ N(s);
3        Let s = t;
4    rof;
5    return(s as local optimal for P)
end local;
```

Figura 5 – Pseudocódigo da fase de pesquisa local do algoritmo GRASP

2.6 Modelos de Referência

O presente trabalho insere-se no contexto da utilização de uma metaheurística GRASP aplicada à resolução de problemas de escalonamento em máquinas paralelas com tempos de *setup* e recursos adicionais. Nos estudos de (Yepes-Borrero et al., 2020) e (Lopez-Esteve et al., 2023), os autores desenvolvem e analisam o desempenho de diferentes metaheurísticas GRASP para a resolução de problemas UPMS e UPMSR, respetivamente. As diferentes metaheurísticas GRASP

são obtidas através da combinação de diferentes estratégias apresentadas para cada uma das três fases do GRASP (fase construtiva, fase de reparação e fase de pesquisa local). Ambos os estudos consideram como função objetivo a minimização do *makespan*, sendo que estes diferem no número de recursos adicionais considerados. Enquanto no estudo de (Yepes-Borrero et al., 2020), são considerados apenas os recursos de *setup*, em (Lopez-Esteve et al., 2023) são incluídos recursos adicionais de processamento e de *setup*.

Os testes realizados nestes dois artigos permitem identificar boas práticas e os valores dos vários parâmetros utilizados pelas metaheurísticas GRASP estudadas que conduzem ao seu melhor desempenho, servindo, por isso, de base para o desenvolvimento da metaheurística GRASP apresentada no presente trabalho.

2.6.1 Modelo de Referência 1

2.6.1.1 Fase Construtiva

Em (Yepes-Borrero et al., 2020) os autores apresentam três algoritmos para a fase construtiva seguindo duas abordagens distintas. Para a primeira abordagem (que inclui os algoritmos C1 e C2), foram adaptados dois algoritmos existentes na literatura para a construção da solução inicial sem considerar qualquer informação acerca dos recursos adicionais. A segunda abordagem (que inclui o algoritmo C3) considera a utilização de recursos adicionais durante a construção da solução inicial e é um algoritmo original desenvolvido neste estudo.

O algoritmo *Constructive 1* (C1) é baseado no algoritmo proposto em (Diana et al., 2015) que considera a seguinte ideologia: para cada trabalho j não inserido na solução (denominado por trabalho pendente) avalia-se o aumento no *makespan* provocado pela inserção do mesmo em cada posição da solução parcial e insere-se o trabalho na posição que provoca o menor aumento de *makespan* (sendo esta considerada a melhor posição). Seguindo a mesma abordagem do algoritmo anterior, o algoritmo *Constructive 2* (C2) tem por base o algoritmo proposto por (Avalos-Rosales et al., 2015). Em C2 começa por se ordenar os trabalhos pendentes de forma decrescente do tempo médio de processamento em cada máquina. Posteriormente, seleciona-se o primeiro trabalho da lista de trabalhos pendentes e calcula-se o aumento de *makespan* provocado pela possível inserção desse trabalho em cada posição da solução parcial. Por fim o trabalho selecionado é alocado à posição que gera o menor aumento de *makespan* (considerada como a melhor posição) e passa-se a analisar o trabalho seguinte da lista. Os trabalhos em C1 e C2 são inseridos um a um na solução parcial, até não existirem mais trabalhos pendentes.

O algoritmo *Constructive 3* (C3), segue uma abordagem original, diferente da primeira e, por isso, considera para além do tempo de processamento nas máquinas, o número de recursos necessários para inserir um trabalho numa dada posição. Conforme apresentado pelos autores, é importante realçar que dada uma sequência $(j_1, j_2, \dots, j_{k-1}, j_k, \dots, j_l)$, ao inserir um trabalho j na posição k, a sequência passa a ser $(j_1, j_2, \dots, j_{k-1}, j, j_k, \dots, j_l)$. Ou seja, deixa de ser

necessária a operação de *setup* entre j_{k-1} e j_k e, no seu lugar, passam a ser necessárias realização de duas outras operações de *setup*: uma entre j_{k-1} e j e outra entre j e j_k.

Com efeito, os autores desenvolveram um coeficiente que considera todos os fatores que são afetados quando inserimos um trabalho j na posição k da máquina i, na solução parcial. O coeficiente $\lambda_{i,j,k}$ é definido como:

$$\lambda_{i,j,k} = C'_i + p_{ij} + \left(\theta_{s(i,k-1,k)} * \theta_{r(i,k-1,k)}\right) + \left(\theta_{s(i,k,k+1)} * \theta_{r(i,k,k+1)}\right) - \left(\gamma_{s(i,k)} * \gamma_{r(i,k)}\right) \quad (1)$$

Onde:

- C'_i é o tempo de processamento total, na solução parcial, da máquina i onde o trabalho j é inserido;
- $\theta_{s(i,k-1,k)}$ é o tempo necessário para a nova operação de *setup* entre as posições $k-1$ e k quando se insere o trabalho j na máquina i ($\theta_{s(i,k,k+1)}$ é definido analogamente);
- $\theta_{r(i,k-1,k)}$ é o número de recursos necessários para a nova operação de *setup* entre as posições $k-1$ e k quando se insere o trabalho j na máquina i ($\theta_{r(i,k,k+1)}$ é definido analogamente);
- $\gamma_{s(i,k)}$ é o tempo da operação de *setup* que deixa de existir entre $k-1$ e k, quando se insere o novo trabalho na posição k da máquina i;
- $\gamma_{r(i,k)}$é o número de recursos que deixam de ser necessários para executar a operação de *setup* entre $k-1$ e k, quando se insere o novo trabalho na posição k da máquina i;

A novidade introduzida por este algoritmo reside na consideração das restrições de recursos durante a construção das soluções, com o objetivo de gerar soluções que utilizem menos recursos. Consequentemente, de acordo com os autores, esta novidade torna a fase de reparação menos exigente, uma vez que se parte de uma solução mais próxima de uma solução viável.

2.6.1.2 Fase de Reparação

A fase de reparação segue um modelo amplamente utilizado na literatura e aplica um método de avaliação baseado no cálculo do número de recursos utilizados em cada instante temporal. Se as restrições de recursos forem cumpridas nesse instante, prossegue-se para a análise do instante temporal seguinte. Caso contrário, o mecanismo de reparação é ativado, e o *setup* que começa mais tarde é adiado para fora do horizonte temporal em que ocorre a sobreposição e o excesso de recursos, ou seja, é reagendado para após a conclusão do *setup* que termina mais

cedo. Em seguida, reavalia-se o consumo de recursos para esse instante e, se as restrições forem cumpridas, passa-se para o instante seguinte.

A junção dos três algoritmos construtivos com o algoritmo comum de reparação resulta nas seguintes três heurísticas:

- Heurística 1: C1 + algoritmo de reparação;
- Heurística 2: C2 + algoritmo de reparação;
- Heurística 3: C3 + algoritmo de reparação.

2.6.1.3 Randomização

De acordo com os autores, "a randomização da fase construtiva é amplamente utilizada nos problemas de otimização combinatória com o objetivo de evitar ótimos locais". Neste sentido, no lugar de escolher a melhor posição a inserir cada candidato, seleciona-se uma posição, de forma aleatória, a partir de uma Lista Restrita de Candidatos. A Lista Restrita de Candidatos considerada neste estudo depende de um valor de $\alpha \in [0,1]$, que é calibrado na fase experimental.

2.6.1.4 Fase de Pesquisa Local

Por fim, com o objetivo de melhorar o *makespan* das sequências obtidas na fase construtiva, aplica-se a fase de pesquisa local, que segue a mesma filosofia da segunda abordagem da fase construtiva, procurando melhorias na sequência que considerem não apenas os tempos de processamento nas máquinas, mas também o consumo de recursos. A fase de pesquisa local utilizada é composta por três etapas: *Internal Swap*, *External Swap* e *External Insertion*. Antes e depois de cada uma destas operações, a solução é reparada (se necessário) e avaliada, de modo a reter sempre a melhor solução.

2.6.1.5 Conclusões

De forma a ser possível avaliar o desempenho dos algoritmos propostos, os autores utilizam o indicador *Relative Percent Deviation* (RPD), para comparar os resultados obtidos através das três heurísticas consideradas com os melhores resultados conhecidos para cada instância:

$$RPD = \frac{C_{max}(alg) - C_{max}(best)}{C_{max}(best)} * 100 \quad (2)$$

Onde $C_{max}(alg)$ representa o *makespan* da solução obtida através do algoritmo testado e $C_{max}(best)$ representa o melhor *makespan* conhecido para a instância.

Comparando o desempenho das três heurísticas consideradas no estudo, os autores concluem que para instâncias de pequena dimensão a Heurística 2 apresenta um desempenho ligeiramente melhor (RPD: 9.67%, t=6.51 *ms*) do que a Heurística 1 (RPD: 11.01%, t=6.53 *ms*) e a Heurística 3 (RPD: 11.07%, t=7.49 *ms*). No entanto, para instâncias de grande dimensão a Heurística 3 (RPD: 0.19%, t=0.228 *s*) apresenta um desempenho largamente melhor do face à Heurística 1 (RPD: 39.97%, t=0.148 *s*) e à Heurística 2 (RPD: 69.66%, t=0.098 *s*). Com efeito, posteriormente, serão consideradas apenas as conclusões apresentadas com respeito à Heurística 3, uma vez que é a mais relevante para o presente estudo.

Relativamente às conclusões do estudo, os autores apresentam os seguintes pontos:

- Em instâncias de grande dimensão, para a Heurística 3, o valor de $\alpha = 0.25$ (que define o tamanho da RCL) apresenta as melhores soluções com $Av.RPD = 0.52\%$, enquanto os valores de $\alpha = 0.5$ e $\alpha = 0.75$ apresentam valores de RPD superiores a 40%;

- Após uma análise detalhada, não foram encontradas diferenças estatisticamente significativas entre as heurísticas que utilizam algoritmos construtivos enquadrados com a primeira abordagem e aquelas que utilizam algoritmos construtivos enquadrados com a segunda abordagem, para instâncias de pequena dimensão. No entanto, para instâncias de grande dimensão, a diferença entre as duas abordagens é evidente, sendo que os algoritmos que consideram as informações de recursos na fase construtiva apresentam um desempenho superior.

- Demonstra-se empiricamente que, em situações de escassez de recursos, incluir o conhecimento relativo aos recursos na fase construtiva melhora significativamente tanto a qualidade das soluções obtidas como o tempo computacional.

- Os algoritmos a que se aplicam a fase de pesquisa local encontram soluções melhores do que aqueles aos quais a mesma não é aplicada, tendo sido observadas diferenças estatisticamente significativas, de acordo com a análise ANOVA dos testes conduzidos.

2.6.2 Modelo de Referência 2

Em (Lopez-Esteve et al., 2023), os autores testam quatro estratégias heurísticas aplicadas à resolução de um problema UPMSR-PS. As quatros heurísticas abordadas neste estudo resultam da combinação de dois algoritmos da fase construtiva (C1 e C2) e dois algoritmos da fase de reparação (R1 e R1), obtendo-se: $\mathrm{H1} = \mathrm{C1} + \mathrm{R1}, \mathrm{H2} = \mathrm{C1} + \mathrm{R2}, \mathrm{H3} = \mathrm{C2} + \mathrm{R1}, \mathrm{H4} = \mathrm{C2} + \mathrm{R2}$.

2.6.2.1 Fase Construtiva

O algoritmo C1 desenvolvido para a Fase Construtiva começa com uma lista que inclui todos os trabalhos a serem inseridos na solução (lista de trabalhos pendentes) e uma solução parcial vazia com *makespan* zero. Posteriormente, aplica-se um *Initial Step* e um processo iterativo:

- *Initial step*: o trabalho da lista de trabalhos pendentes que consome o maior número de recursos em cada máquina, é associado a essa máquina;

- Processo iterativo: Em cada iteração subsequente, um trabalho da lista de trabalhos pendentes é alocado a uma das máquinas disponíveis, garantindo que essa alocação trabalho-máquina provoca o menor aumento possível no makespan em relação à iteração anterior. Após a alocação, o trabalho é removido da lista de trabalhos pendentes, e passa-se a avaliar a melhor alocação para o trabalho seguinte da lista. Esta fase termina quando já não existirem trabalhos na lista de trabalhos pendentes.

O algoritmo C2 é semelhante ao C1, mas não considera o *Initial Step* na alocação dos primeiros trabalhos às máquinas.

2.6.2.2 Fase de Reparação

A Fase de Reparação analisa a sequência obtida na Fase Construtiva e repara-a, sempre que o número de recursos disponíveis em algum instante seja excedido. A fase de reparação envolve um processo de análise que decorre do instante $t = 0$ até ao instante com maior valor de t existente na solução. Se o número disponível de recursos de processamento (ou de recursos de *setup*) for excedido em algum instante, é ativado um mecanismo de reparação aplicado a todas as máquinas que estão a processar trabalhos (ou a realizar operações de *setup*) nesse instante t. Os autores apresentam então duas estratégias distintas para reduzir o consumo de recursos no instante t identificado:

- Primeira estratégia (R1): O processamento (*setup*) de um dos trabalhos (*setups*) que está a decorrer no instante t é adiado até que os outros trabalhos (*setups*) estejam concluídos. Esta operação é repetida para todos os trabalhos (*setups*) que se sobrepõem no instante t, e mantém-se o adiamento que provoca o menor aumento no makespan. Se, após adiar um dos trabalhos (*setups*), ainda houver uma utilização excessiva de recursos, o processo é repetido com os restantes trabalhos (*setups*).

- Segunda estratégia (R2): Em vez de adiar os trabalhos (*setups*) até à conclusão dos outros, opta-se por adiá-los por uma unidade de tempo e verificar se o número de recursos continua a exceder a disponibilidade no instante t. Se o excesso de recursos no instante t for resolvido, passa-se para o instante $t + 1$. Caso contrário, o processo continua, adiando uma unidade de tempo de cada vez, até que a utilização de recursos

deixe de exceder a capacidade. Esta operação é repetida para todos os trabalhos (*setups*) que estão a decorrer no instante t, e mantém-se o adiamento que provoca o menor aumento no makespan. Se ainda forem utilizados mais recursos do que os disponíveis, a mesma operação é repetida com os restantes trabalhos.

2.6.2.3 Randomização

Após observarem que as soluções obtidas apresentavam um GAP considerável face aos MILP existentes na literatura, os autores aplicaram uma estratégia de randomização da Fase Construtiva, de forma a diminuírem o GAP e, simultaneamente, obterem uma maior variedade de soluções. A estratégia de randomização utilizada segue o modelo apresentado em (Feo & Resende, 1989). Para cada trabalho da lista de trabalhos pendentes, as possíveis alocações trabalho – máquina são ordenadas de forma crescente de acordo com o aumento do *makespan* provocado pela alocação. Os primeiros s elementos dessa lista ordenada vão constituir a *Restricted Candidate List* (RCL), ou seja, a RCL inclui as s alocações trabalho – máquina que provocam o menor aumento de makespan na solução.

A diferença reside no método de seleção da alocação trabalho – máquina, que passa a ser realizado de forma aleatória, em vez de se escolher sempre a melhor opção.

2.6.2.4 Fase de Pesquisa Local

Com o objetivo de evitar ótimos locais, os autores propõem uma fase de pesquisa local. A estratégia de pesquisa local utilizada inicia com a remoção aleatória de um trabalho da máquina gargalo (a máquina cuja conclusão do último trabalho ocorre mais tarde) e, posteriormente, testa-se a sua reinserção em todas as posições possíveis de todas as máquinas. Após cada reinserção avalia-se o novo *makespan* da solução e mantém-se o menor valor obtido. Assim, este processo é repetido até que se realizem 50 iterações seguidas sem que ocorra melhoria no valor do *makespan* obtido.

2.6.2.5 Avaliação de parâmetros considerados

Os autores consideram que a eficiência e a qualidade dos resultados obtidos pelos algoritmos desenvolvidos no estudo dependem não apenas da estrutura utilizada, mas também dos valores dos parâmetros e outros fatores considerados. Nesse sentido, este estudo explora de forma exaustiva o impacto dos seguintes parâmetros e fatores:

- O tamanho da RCL: São testados três valores diferentes para o tamanho da RCL (2, 3 e 4 elementos), correspondendo a um fator com três níveis $RCL \in \{2, 3, 4\}$;

- O tempo máximo de computação: O tempo máximo de computação permitido é definido de acordo com o tamanho da instância (n trabalhos), de acordo com a fórmula $t_{max} = n * w\ (segundos)$. São realizados testes para os valores de $w\ \in \{1, 2, 3, 4, 5\}$;

- A distribuição de probabilidade de cada elemento da RCL ser escolhido: São testadas duas distribuições de probabilidade. Por um lado, a distribuição uniforme atribui uma probabilidade de escolha igual a todos os elementos que compõe a RCL. Por outro lado, a distribuição decrescente atribui uma probabilidade de escolha diferente para cada elemento da RCL, dada pela fórmula $(s - k + 1)\frac{2}{s(s+1)}$. A distribuição de probabilidade corresponde a um teste com 2 níveis $P \in \{U, D\}$.

- A consideração de recursos na fase construtiva: Após ter sido provado de forma empírica em (Yepes-Borrero et al., 2020), que considerar a utilização de recursos na fase construtiva conduz a um melhor desempenho do GRASP, neste estudo os autores testam o impacto de considerar o número de recursos necessários no momento de definir a melhor alocação trabalho – máquina. Para isso utilizam a fórmula $\alpha * makespan + (1 - \alpha) * resources, \alpha \in [0,1]$. Neste sentido, a ordenação dos elementos da RCL é feita de acordo com uma combinação do aumento provocado no makespan e do número de recursos (de *setup* e processamento) utilizados pela alocação. São então considerados dois níveis para este fator $R \in \{Y, N\}$.

- A eficiência da Fase de Pesquisa Local: Este fator considera as opções de considerar ou não considerar o Local Search sendo criados dois níveis $LS \in \{Y, N\}$.

2.6.2.6 Conclusões

Para avaliar os resultados obtidos com as diferentes estruturas de algoritmos, parâmetros e outros fatores utilizados, os autores comparam a qualidade das soluções obtidas neste estudo com as soluções obtidas em (Fanjul-Peyro, 2020):

- GAP de 6.14% para instâncias de pequena dimensão;

- GAP de 4.47% para instâncias de média dimensão;

- GAP de 2.26% para instâncias de grande dimensão.

A medida de qualidade utilizada para comparar os resultados obtidos no presente trabalho com os resultados obtidos no estudo de referência é a *Relative Percent Deviation* (RPD), onde z^M representa o makespan obtido através da metaheurística desenvolvida no estudo e z^F representa o makespan obtido pelo modelo desenvolvido em (Fanjul-Peyro, 2020).

$$RPD = 100\frac{z^M - z^F}{z^M} \tag{3}$$

Analisando o desempenho das quatro heurísticas consideradas no estudo (H1, H2, H3 e H4), apresentado na Tabela 1, conclui-se que todas apresentam um valor elevado de RPD. No entanto, observa-se que a heurística H4 é a que apresenta o menor valor médio de RPD, permitindo concluir que a combinação do algoritmo construtivo C2 com o algoritmo de reparação R2 apresenta os melhores resultados, sendo por isso a combinação considerada ao longo das experiências consideradas no estudo. Importa ainda salientar que, à medida que o número de trabalhos nas instâncias aumenta, o RPD tende a diminuir, dado que o modelo MILP de (Fanjul-Peyro, 2020)devolve soluções mais distantes da solução ótima.

Tabela 1 – Valores de RPD para as 4 heurísticas estudadas para instâncias com n trabalhos

n	H1	H2	H3	H4
10	21.21	11.30	9.06	8.45
20	18.15	12.06	9.95	9.86
30	17.62	11.68	8.94	9.44
40	12.80	9.43	6.56	6.55
50	8.46	5.67	3.08	3.02
Média	15.65	10.03	7.52	7.46

De forma resumida, os autores apresentam as seguintes conclusões em relação aos fatores que afetam o desempenho da metaheurística após a aplicação dos testes estatísticos:

- A randomização da Fase Construtiva melhora significativamente a qualidade das soluções obtidas;

- Permitir um tempo computacional maior também melhora as soluções, como seria de esperar. No entanto, para instâncias de média e grande dimensão, após um determinado tempo computacional, as melhorias nas soluções deixam de ser significativas. Com efeito, recomenda-se a execução do GRASP durante $3n$ segundos, uma vez que as soluções obtidas apresentam qualidade estatisticamente idêntica às obtidas com um tempo de execução de $5n$ segundos;

- Utilizar uma distribuição de probabilidade decrescente, atribuindo maior probabilidade de escolha aos primeiros elementos da RCL, melhora o desempenho do algoritmo;

- Considerar a informação de recursos juntamente com o makespan na avaliação das alocações trabalho-máquina na RCL melhora o desempenho do algoritmo para instâncias de pequena dimensão, uma vez que as soluções criadas são de maior qualidade. Contudo, esta melhoria não se verifica em instâncias de média e grande dimensão, pois, ao tornar o algoritmo mais lento, gera-se um número demasiado reduzido de soluções iniciais;

- Adicionar um algoritmo de pesquisa local à metaheurística melhora consideravelmente o desempenho da mesma.

3 Métodos e aplicação

3.1 Descrição do Problema

Este estudo desenvolve-se no departamento de planeamento de produção da INPLAS, uma empresa do Grupo Simoldes especializada na fabricação de peças plásticas por injeção, destinadas, sobretudo, à indústria automóvel. Embora este trabalho se concentre num cenário específico, a solução desenvolvida pode ser adaptada a outras situações que partilhem as mesmas características globais do problema identificado, mediante a introdução de pequenas adaptações.

Tendo em conta a exigência do mercado no qual a INPLAS opera, a empresa adota um sistema de laboração contínua, operando de segunda a sexta-feira (5 dias por semana) em três turnos diários de 8 horas cada. As encomendas são colocadas diariamente pelos clientes no mínimo 24h antes do prazo de entrega definido, exigindo que o departamento de produção e, consequentemente, o departamento de planeamento de produção possuam uma capacidade adicional para agir e tomar decisões de forma ágil. As datas de entrega no problema correspondem a valores diários dentro do horizonte temporal do planeamento e definem o momento limite até ao qual a produção de cada trabalho deve estar concluída. Uma vez que vários trabalhos partilham a mesma data de entrega, considera-se que o problema se enquadra na tipologia de problemas com datas de entrega comuns.

Neste trabalho considera-se, então, um problema de escalonamento em máquinas paralelas, no qual é necessário produzir n trabalhos em m máquinas com o objetivo de cumprir os prazos e quantidades de entrega dos pedidos dos clientes. Para cada dia, é necessário definir o escalonamento dos trabalhos que representam as necessidades dos clientes para esse dia e para os seis dias seguintes, ou seja, em cada iteração, o horizonte de planeamento considerado é de sete dias. Cada trabalho tem associado uma referência, que identifica o tipo de produto a produzir, e uma quantidade de produção dessa referência. Pedidos de cliente das mesmas referências que tenham datas de entrega diferentes, originam trabalhos diferentes. As referências de maior rotação operam em *picking* (nome utilizado pela empresa para descrever um sistema que funciona de forma semelhante ao JIT) e podem ser produzidas na quantidade exata pedida pelo cliente, sem restrições. Por outro lado, as referências de menor rotação seguem a estratégia *Make to Stock*, e a quantidade de produção dos trabalhos que incluem estas referências nunca pode ser inferior à quantidade mínima de encomenda. As referências de menor rotação devem também respeitar um limite máximo de *stock*. Todos estes valores limite, assim como a indicação de se uma referência pertence ou não ao sistema de *picking*, são informações técnicas fornecidas pela empresa.

Cada referência só pode ser produzida utilizando um molde específico, mas um molde pode produzir várias referências quando introduzidas diferentes configurações de materiais ou nos casos em que é composto por mais do que um tipo de cavidade. Cada referência pode ainda ser produzida, no máximo, em três máquinas distintas (Máquina Principal, Máquina Alternativa 1 e Máquina Alternativa 2, por esta ordem de prioridade). Esta questão exige a criação de restrições que aumentam a complexidade do problema: a necessidade de assegurar que cada molde não é utilizado simultaneamente em mais de uma máquina e a necessidade de garantir que um molde só é transferido para uma máquina, após ter sido retirado da máquina na qual foi utilizado anteriormente.

Apenas pode ser realizado um trabalho de cada vez em cada máquina, sendo que o processamento dos mesmos pode ser interrompido em situações de diminuição de recursos disponíveis ou quando as prioridades de produção são alteradas. No entanto, os trabalhos interrompidos devem ser retomados mais tarde sem que o molde em utilização seja removido da máquina. Ou seja, se um trabalho A for interrompido devido à necessidade de utilizar os operadores que estavam alocados ao seu processamento para o processamento, numa outra máquina, de um trabalho B mais prioritário, a máquina onde o processamento do trabalho A foi interrompido, não pode processar outro trabalho até terminar o processamento do trabalho A.

Este problema considera a necessidade de realizar operações de *setup* apenas entre o processamento de trabalhos que não pertencem à mesma família (ou seja, que não utilizam o mesmo molde). Para além disso, o tempo de *setup* é independente da sequência, apresentando um valor fixo de 60 minutos. Neste contexto, as trocas de molde envolvem a utilização de equipas de *setup* especializadas, cujo número está limitado e varia em cada turno. As equipas de *setup* têm também capacidade limitada e cada equipa só pode realizar uma troca de molde de cada vez. Pode, então, afirmar-se que as características das operações de *setup* do presente problema, permitem o seu enquadramento com a tipologia de problemas de escalonamento com *setups* entre famílias de produtos e independentes da sequência.

O processamento de cada trabalho envolve a utilização de um número de recursos previamente definido pela empresa para cada referência. Com efeito, os recursos são considerados dinâmicos, uma vez que a quantidade de recursos alocados a cada máquina varia ao longo do tempo.

Por fim, é essencial verificar se, no início de um novo dia, existem trabalhos inacabados pendentes do dia anterior, uma vez que o tempo remanescente necessário para a sua conclusão será contabilizado como o tempo de processamento do primeiro trabalho da máquina para o novo dia. Os trabalhos processados no início de cada dia que transitam, incompletos, do dia anterior são denominados de "trabalhos de início de dia". Embora se considere que esses trabalhos devem estar concluídos antes do primeiro dia de planeamento, ao longo deste estudo utiliza-se, por vezes, o valor 0 para a sua data de entrega, com o único propósito de os identificar de forma clara.

Devido às variações de recursos adicionais disponíveis em cada turno, nomeadamente do número de operadores disponíveis, não se pode garantir que a máquina consiga retomar esses

trabalhos no início de um novo dia, podendo ser necessário interromper a produção de algumas das máquinas. Seguindo a mesma lógica, devido às variações do número de equipas de *setup* disponíveis em cada turno, não é permitido que uma operação de *setup* inicie e termine em turnos diferentes. Assim, impede-se que qualquer operação de *setup* seja iniciada menos de 60 minutos antes do final de cada turno. Na ausência de trabalhos de início de dia, o conhecimento do último molde utilizado nas máquinas é desconsiderado, assumindo-se que qualquer trabalho pode ser produzido no início do dia sem necessidade de realizar uma operação de *setup*.

Com isto, o objetivo do departamento de produção passa por responder à procura diária de peças, respeitando os limites de capacidade, prazos de entrega, a disponibilidade de operadores e de equipas de *setup*.

3.2 Contribuições e Adaptações aos Modelos GRASP de referência

3.2.1 Caso Particular das Máquinas Paralelas Dedicadas com Múltiplas Alternativas

A metaheurística GRASP apresentada no capítulo *3.4. - GRASP para UPMSR-PS*, baseia-se nos algoritmos desenvolvidos por (Yepes-Borrero et al., 2020) e (Lopez-Esteve et al., 2023). Ambos os estudos de referência analisam e avaliam diferentes abordagens para as três fases que compõem o GRASP. Além disso, os autores investigam as variações de desempenho de cada abordagem mediante a utilização de diferentes parâmetros e outros fatores que influenciam a sua eficácia. No final de cada estudo, os autores apresentam as abordagens, parâmetros e fatores que permitem obter os melhores resultados para os problemas UPMS (Yepes-Borrero et al., 2020) e UPMSR-PS (Lopez-Esteve et al., 2023).

Os dois artigos abordam problemas com máquinas paralelas não relacionadas, onde qualquer trabalho pode ser processado em qualquer máquina. Nos dois cenários, o tempo de *setup* depende tanto da máquina em que ocorre, quanto dos trabalhos entre os quais a operação é realizada (é dependente da máquina e da sequência). O tempo de processamento de cada trabalho também varia de acordo com a máquina na qual é processado (p_{ij}). A complexidade provocada por estas dependências obriga a determinar a alocação mais eficiente dos trabalhos às máquinas durante a fase construtiva do processamento metaheurístico, uma vez que a alocação de cada trabalho a uma máquina depende diretamente das alocações anteriores e influência diretamente as alocações seguintes.

No entanto, o problema abordado neste estudo considera máquinas paralelas dedicadas com múltiplas alternativas, onde quer os tempos de processamento (p_j), quer os tempos de *setup*

(s_j) dependem exclusivamente dos trabalhos e são fixos para todas as máquinas alternativas em que o trabalho pode ser processado. Neste tipo de problemas, também existe a necessidade de alocar trabalhos às máquinas, contudo, considera-se que essa alocação está limitada ao subconjunto de máquinas elegíveis para cada trabalho e pode ser realizada antes da fase construtiva, uma vez que cada alocação não é influenciada pelas alocações anteriores nem influência as alocações seguintes.

Neste contexto, a pré-alocação dos trabalhos às máquinas segue uma abordagem heurística que visa minimizar o número de trabalhos não alocados à sua máquina preferencial, respeitando as restrições de capacidade e a hierarquia das alternativas disponíveis. O subcapítulo *3.3. – Heurística para alocação trabalho-máquina* apresenta o procedimento heurístico desenvolvido, no qual a alocação prioriza, em primeiro lugar, a Máquina Principal, seguida da Máquina Alternativa 1 e, só depois, da Máquina Alternativa 2. Caso nenhuma das alternativas seja viável, os trabalhos são atribuídos a uma Máquina Fictícia.

3.2.2 Resolução de Problemas com Datas de Entrega

Os dois modelos de referência considerados apresentam uma função objetivo de minimização de *makespan*. Contudo, uma vez que no presente trabalho se incorporam datas de entrega associadas a cada trabalho, logicamente, o objetivo passará a ser garantir a entrega do número máximo de pedidos dos clientes antes da data de entrega definida pelos mesmos, ou seja, garantir a conclusão do número máximo de trabalhos antes da data de entrega que lhes está associada.

Com efeito, considerando que quando um trabalho é concluído até à data de entrega, o seu atraso é nulo, e que quando é concluído após a data de entrega, o seu atraso assume o valor da diferença entre a data de conclusão e a data de entrega ($C_j - d_j$), a função objetivo considerada passa a ser conforme apresentada equação 4:

$$Min\ Z = \sum_{j \in N} max(0, C_j - d_j) \tag{4}$$

Assim, a função objetivo utilizada assume o valor da diferença entre o tempo de conclusão do trabalho j e a respetiva data de entrega ($C_j - dj$), quando este valor é maior do que 0, e o valor 0 quando o trabalho é concluído dentro do prazo ($C_j \leq dj$).

Tal como referido anteriormente, o problema em estudo apresenta, ainda, a particularidade de os trabalhos estarem organizados em subconjuntos com datas de entrega comuns, o que permite enquadrá-lo na categoria de problemas com datas de entrega comuns.

3.2.3 Configuração Inicial das Máquinas

Tal como mencionado no subcapítulo *3.1. - Descrição do Problema*, para tornar o planeamento mais eficiente, é crucial conhecer a configuração das máquinas no início de cada dia. Este conhecimento é particularmente relevante em sistemas de laboração contínua, pois permite obter, no início de cada dia, informações cruciais sobre o último trabalho processado em cada máquina, como o tempo de processamento remanescente (caso o trabalho não tenha sido concluído no dia anterior) e o molde que se encontra inserido em cada máquina naquele momento.

Com efeito, ao contrário dos modelos de referência, nos quais a fase construtiva do GRASP inicia com soluções parciais vazias e a sequenciação dos primeiros trabalhos assume que as máquinas estão paradas e sem moldes inseridos, para o problema em estudo são feitas as seguintes considerações relativamente à situação da solução parcial antes do início da fase construtiva:

- Garante-se que o primeiro trabalho de cada máquina corresponde ao trabalho que transitou incompleto do dia anterior, considerando o tempo de processamento remanescente do mesmo;

- Caso a máquina não tenha nenhum trabalho incompleto em transição do dia anterior, assume-se que não há nenhum molde em carga nem existe a necessidade de realizar uma operação de *setup*, em conformidade com procedimento estabelecido pela empresa onde este estudo foi realizado.

3.2.4 Coeficiente de Classificação

Um dos fatores que influencia a qualidade do GRASP é a forma como, na fase construtiva, se quantifica o impacto de inserir um trabalho em cada posição da solução parcial. Para definir este fator, recorre-se a uma adaptação do método utilizado em (Yepes-Borrero et al., 2020), que consiste em considerar não apenas o tempo de processamento do trabalho na máquina, mas também o número de recursos necessários.

Conforme apresentado pelos autores do artigo mencionado, é importante realçar que dada uma sequência $(j_1, j_2, \dots, j_{k-1}, j_k, \dots, j_l)$, ao inserir um trabalho j na posição k, a sequência passa a ser $(j_1, j_2, \dots, j_{k-1}, j, j_k, \dots, j_l)$. Ou seja, deixa de ser necessária a operação de *setup* entre j_{k-1} e j_k e, no seu lugar, passa a ser necessário realizar duas outras operações de *setup*: uma entre j_{k-1} e j e outra entre j e j_k.

Nº setups antes	2
Nº setups depois	3

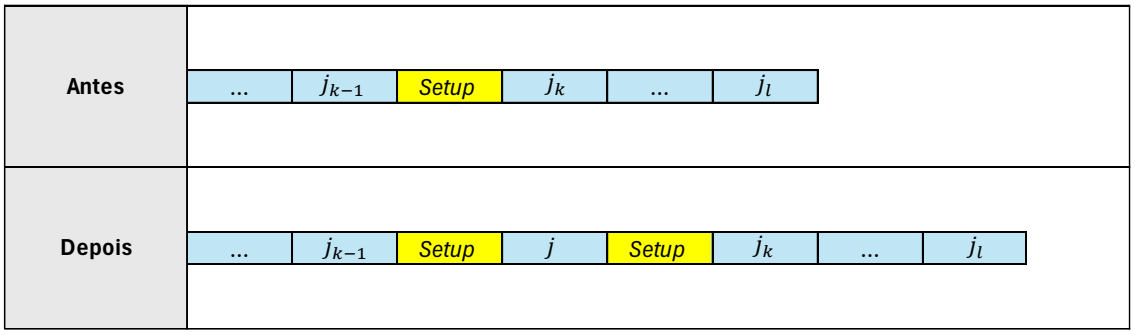


Figura 6 – Impacto no número de *setups* ao inserir o trabalho j na solução

Com efeito, os autores desenvolveram um coeficiente que considera múltiplos fatores que são afetados quando inserimos um trabalho j na posição k da máquina i, na solução parcial. O coeficiente $\lambda_{i,j,k}$ é definido como:

$$\lambda_{i,j,k} = C'_i + p_{ij} + \left(\theta_{s(i,k-1,k)} * \theta_{r(i,k-1,k)}\right) + \left(\theta_{s(i,k,k+1)} * \theta_{r(i,k,k+1)}\right) - \left(\gamma_{s(i,k)} * \gamma_{r(i,k)}\right) \quad (5)$$

Onde:

- C'_i é o tempo de processamento total da máquina i onde o trabalho é inserido, na solução parcial;
- $\theta_{s(i,k-1,k)}$ é o tempo necessário para a nova operação de *setup* entre os trabalhos das posições $k-1$ e k quando inserimos o trabalho j na posição k da máquina i ($\theta_{s(i,k,k+1)}$ é definido analogamente);
- $\theta_{r(i,k-1,k)}$ é o número de recursos necessários para a nova operação de *setup* entre os trabalhos das posições $k-1$ e k quando inserimos o trabalho j na posição k da máquina i ($\theta_{r(i,k,k+1)}$ é definido analogamente);
- $\gamma_{s(i,k)}$ é o tempo da operação de *setup* que deixa de existir entre $k-1$ e k, quando inserimos o novo trabalho na posição k da máquina i;
- $\gamma_{r(i,k)}$é o número de recursos que deixam de ser necessários para executar a operação de *setup* entre $k-1$ e k, quando inserimos o novo trabalho na posição k da máquina i;

A adaptação do coeficiente λ apresentado em (Yepes-Borrero et al., 2020) ao problema considerado no presente estudo, exige a remoção de algumas das variáveis consideradas e a inserção de novas variáveis. Tendo em conta a pré-alocação dos trabalhos às máquinas, deixa de fazer sentido utilizar a variável C'_i, uma vez que todas as posições analisadas para cada trabalho apresentarão o mesmo valor para esta variável. Da mesma forma, não é necessário

considerar o impacto das variáveis relacionadas com o consumo de recursos de *setup* ($\theta_{r(i,k-1,k)}$, $\theta_{r(i,k,k+1)}$ e $\gamma_{r(i,k)}$), dado que, no problema em estudo, cada operação de *setup* consome apenas um recurso, não influenciando, assim, o cálculo do coeficiente. Por fim, sendo o tempo de processamento exclusivamente dependente do trabalho, e não da máquina como acontece no modelo de referência, deixa também de fazer sentido considerar a variável p_{ij}.

Por outro lado, a adaptação do coeficiente λ que classifica a adequabilidade de cada posição na qual o trabalho j pode ser inserido na máquina i, à qual está alocado, exige a inclusão de dois novos fatores: $\delta_{(i,k,k+1)}$, β_i e α_k. O coeficiente $\lambda_{i,j,k}$, considerado no presente estudo encontra-se representado pela equação 6.

$$\lambda_{i,j,k} = \delta_{(i,k,k+1)} + (\beta_i * 100) + \alpha_k + (\theta_{s(i,k-1,k)} + \theta_{s(i,k-1,k)} - \gamma_{s(i,k)}), \qquad i = máq.(j) \quad (6)$$

Onde:

- $\delta_{(k,k+1)}$ representa a penalidade imposta ao trabalho j de acordo com a diferença entre a sua data de entrega e a data de entrega do trabalho da posição seguinte (d_{k+1}).

$$\delta_{(k,k+1)} = 50 * d_j - d_{k+1}, \qquad se\ d_j > d_{k+1} \quad (7)$$

Tal como será apresentado na subsecção *3.4.2. - Fase Construtiva*, os trabalhos são inseridos na solução parcial de forma sequencial, por ordem crescente da respetiva data de entrega. Este fator justifica a aplicação de uma penalização apenas quando se insere um trabalho j numa posição anterior a um trabalho cuja data de entrega seja inferior ($d_j > d_{k+1}$). Por outro lado, não existe penalização associada à inserção de um trabalho j com data de entrega inferior à data de entrega do trabalho na posição anterior ($d_j < d_{k-1}$) porque não existe em qualquer momento, na solução parcial, um trabalho já inserido com uma data de entrega superior à de j.

- β_k corresponde à soma do número de trabalhos em atraso na máquina i ao inserir o trabalho j na posição k.

- α_k representa a penalização aplicada a um trabalho j quando a posição k corresponde a uma posição igual ou anterior à última posição na solução parcial ocupada por um trabalho que utiliza o mesmo molde que j. Esta penalização tem como objetivo organizar, por ordem crescente da data de entrega, os trabalhos que partilham o mesmo molde e que estão sequenciados consecutivamente na solução.

O desenvolvimento do novo parâmetro $\lambda_{i,j,k}$ assegura que os fatores considerados contribuem para uma lógica ordenação da solução parcial, otimizando, a utilização dos moldes, minimizando as operações de *setups* e promovendo uma sequência mais eficiente na linha de produção, garantindo, acima de tudo, o menor número de trabalhos em atraso.

3.2.5 Função Multiobjetivo

Tal como referido no subcapítulo, *3.2.2. - Resolução de Problemas com Datas de Entrega*, a consideração das datas de entrega dos trabalhos altera os objetivos do problema. Se nos modelos exemplo o objetivo principal é maximizar a eficiência do sistema de produção através da minimização do makespan, no presente trabalho a prioridade é garantir a conclusão do número máximo de trabalhos antes da respetiva data de entrega. Esta prioridade assegura à empresa uma maior taxa de serviço junto dos seus clientes. Assim, a função objetivo é alterada para a minimização do número de trabalhos com *tardiness* ($C_j > d_j$).

No entanto, existe uma panóplia de soluções ótimas que apresentam o mesmo valor mínimo do número de trabalhos com atraso, sendo, por isso, necessário definir um método que permita identificar a solução ótima mais adequada ao problema. Nesse sentido, optou-se por alterar a função objetivo simples para uma função multiobjetivo que, além de minimizar o número de trabalhos com atraso, visa também reduzir o número total de tempos improdutivos. No contexto do problema, consideram-se como tempos improdutivos os tempos de *setup* e os tempos inativos.

Em suma, a função objetivo proposta pretende criar o cronograma de produção mais compacto possível, minimizando simultaneamente o número de trabalhos com atraso, o número total de operações de *setup* realizadas e o makespan máximo da solução.

A otimização dos três objetivos presentes na função multiobjetivo segue uma otimização lexicográfica, conforme a classificação apresentada por (T'kindt & Billaut, 2006), onde os objetivos são priorizados de forma hierárquica. Assim, o primeiro objetivo consiste em minimizar o número de trabalhos em atraso. Em seguida, o segundo objetivo, visa minimizar o número de *setups*, e é otimizado dentro do conjunto de soluções que já minimizam o atraso. Por fim, o último objetivo, que consiste na minimização do makespan máximo, é otimizado entre as soluções que já minimizam tanto os atrasos quanto o número de *setups*.

3.3 Heurística para Alocação Trabalho-Máquina

Tendo em conta a hierarquia de máquinas estabelecida, o planeamento ideal contemplaria a produção de todos os trabalhos nas suas máquinas principais. No entanto, devido às restrições

de capacidade de produção diária de cada máquina, torna-se inviável para algumas máquinas acomodar todos os trabalhos que as utilizam como máquina principal e, ao mesmo tempo, assegurar a conclusão de todos os trabalhos antes da sua data de entrega. Essa limitação exige ajustes no planeamento para redistribuir a carga de trabalho de forma a potencializar o cumprimento dos prazos estabelecidos.

Neste sentido, tal como mencionado anteriormente, foi desenvolvida, no presente estudo, uma heurística com o objetivo de otimizar a alocação dos trabalhos às múltiplas máquinas alternativas onde podem ser produzidos. Este procedimento heurístico, visa acima de tudo, minimizar o número de trabalhos excluídos do planeamento por falta de capacidade do sistema de produção (equivalente a minimizar o número de trabalhos alocados na Máquina Fictícia), respeitando, para os trabalhos planeados, a hierarquia de máquinas definida para cada trabalho. O output obtido através desta heurística serve como ponto de partida para a aplicação do modelo GRASP, apresentado no capítulo seguinte. Ao longo deste capítulo, descreve-se e analisa-se a heurística construtiva de alocação desenvolvida, evidenciando a sua contribuição para a eficiência do planeamento.

3.3.1 Restrições de Capacidade

Para garantir que o processo de alocação dos trabalhos às máquinas reflete corretamente as características do problema, é essencial, primeiramente, definir os limites de capacidade do sistema de produção. Neste sentido, sabendo que a produção da INPLAS opera em três turnos de oito horas cada (480 minutos), tem-se que cada dia de trabalho corresponde a 1440 minutos. Assim, na ausência de tempos inativos, cada máquina pode processar um número de trabalhos e realizar um número de operações de *setup* cuja soma dos tempos de processamento e de *setup* não exceda os 1440 minutos por dia.

Desta forma, depois de se adicionar os trabalhos de início de dia (ou seja, os trabalhos cujo processamento transita, incompleto, do último turno do dia anterior) às necessidades de produção existentes para os sete dias planeamento, os trabalhos são agrupados por máquinas e por data (dia) de entrega. Como mencionado anteriormente, os trabalhos de início de dia têm data de entrega no dia 1, embora, em alguns exemplos, lhes seja atribuída a data de entrega 0 apenas para fins de identificação.

Após se definir a lista de necessidades agrupada por subconjuntos de datas de entrega, para cada um dos sete dias de planeamento e para cada uma das trinta e três máquinas do sistema de produção, soma-se o tempo de produção dos trabalhos de cada subgrupo e o tempo mínimo de *setup* entre esses trabalhos. O tempo de *setup* mínimo é dado pela multiplicação do tempo de *setup* pelo número de moldes utilizados pelos trabalhos menos um, incluindo, caso exista, o molde que se encontra na máquina no início de dia.

Para cada dia de produção, em cada máquina, é necessário assegurar que o tempo total de produção dos subgrupos de trabalhos com datas de entrega iguais ou inferiores a esse dia não excede a capacidade da máquina até essa data, conforme apresentado no exemplo a seguir.

A Figura 7 apresenta um exemplo da lista de necessidades de produção, agrupada por máquina e por subconjuntos de datas de entrega, para quatro máquinas e três dias de planeamento. Note-se que a lista exibida já segue as orientações previamente mencionadas, incluindo os trabalhos que transitam, inacabados, do dia anterior ao primeiro dia de planeamento (destacados a cor laranja).

Necessidades de Produção Agrupadas por Máquina e Data de Entrega

	Dia 1			Dia 2	Dia 3	
Máquina 1	A MO. 10	B MO. 10	C MO. 20	D MO. 30	E MO.10	F MO. 20
Tempo processamento (minutos)	250	300	700	450	300	670

	Dia 1		Dia 2			Dia 3
Máquina 2	G MO. 40	H MO. 50	I MO. 50	J MO. 40	K MO. 60	L MO. 70
Tempo processamento (minutos)	600	900	350	460	670	600

	Dia 1		Dia 3			
Máquina 3	M MO. 80	N MO. 90	O MO. 80	P MO. 80	Q MO. 80	R MO. 90
Tempo processamento (minutos)	900	550	300	890	700	540

	Dia 1		Dia 2		Dia 3	
Máquina 4	S MO. 100	T MO. 110	U MO. 100	V MO. 110	W MO. 120	X MO. 120
Tempo processamento (minutos)	100	200	400	200	150	260

Figura 7 – Necessidades de produção agrupadas por máquina e data de entrega

Seguindo o procedimento definido, depois de se ter a lista de necessidades de produção agrupada por máquina e por data de entrega, é necessário analisar a capacidade de cada máquina para produzir os trabalhos que lhe estão alocados, antes da sua data de entrega. Essa análise consiste, resumidamente, na comparação entre o tempo de processamento acumulado e a capacidade acumulada para cada dia de planeamento, tal como se encontra apresentado na Tabela 2.

A Tabela apresenta, para cada máquina e para cada subgrupo de data de entrega, os seguintes elementos:

- Σ Tempo de Processamento: soma do tempo de processamento dos trabalhos cuja data de entrega corresponde ao dia de cada subgrupo de datas de entrega;

- Tempo de Processamento Acumulado: soma dos tempos de processamento dos trabalhos com data de entrega igual ou anterior ao dia de cada subgrupo de datas de entrega;

- Tempo Mínimo de *Setup*: tempo mínimo necessário para realizar as operações de *setup* entre os trabalhos processados com data de entrega igual ou anterior ao dia de cada subgrupo de datas de entrega;

- Capacidade Acumulada: capacidade acumulada diária de cada máquina até o dia de cada subgrupo de datas de entrega;

- Folga: diferença entre a capacidade acumulada da máquina e a soma dos tempos de processamento dos trabalhos com data de entrega igual ou anterior a esse dia, indicando a folga disponível.

Esta Tabela permite uma visão detalhada da carga de trabalho em cada máquina ao longo dos dias de planeamento, identificando facilmente os dias com excesso de carga (valores negativos em vermelho) e a folga disponível (valores positivos em verde).

É importante notar que a soma do tempo de processamento diário não está diretamente relacionada com a folga da máquina. O que realmente importa é o tempo de processamento acumulado. Por exemplo, observe-se a máquina 3: apesar de a soma do tempo de processamento dos trabalhos com data de entrega para o dia 3 exceder, significativamente, a capacidade diária da máquina, o tempo de processamento acumulado dos trabalhos com data de entrega até ao dia 3 é consideravelmente inferior à capacidade acumulada da máquina. Por essa razão, a folga na máquina é positiva para esse dia, indicando que, acumulativamente, a máquina consegue atender à procura sem ultrapassar sua capacidade.

De forma resumida, a informação relevante da Tabela 2 concentra-se na análise da folga diária de cada máquina. Quando a folga de uma máquina é nula ou positiva para um determinado dia, isso indica que a máquina tem capacidade suficiente para produzir todos os trabalhos alocados com data de entrega igual ou anterior a esse dia. Por outro lado, se a folga for negativa em algum momento, isso significa que a máquina não possui capacidade suficiente para produzir os trabalhos alocados com data de entrega igual ou anterior a esse dia, exigindo assim ajustes no planeamento para redistribuir a carga de trabalho ou realocar trabalhos para outras máquinas disponíveis, conforme será apresentado no subcapítulo seguinte

Tabela 2 – Análise de utilização, capacidade e folga do sistema de produção

	∑ Tempo Processamento			Tempo Processamento Acumulado			Tempo Mínimo Setup			Capacidade Acumulada			Folga		
	Dia 1	Dia 2	Dia 3	Dia 1	Dia 2	Dia 3	Dia 1	Dia 2	Dia 3	Dia 1	Dia 2	Dia 3	Dia 1	Dia 2	Dia 3
Máquina 1	1250	450	970	1250	1700	2670	60	120	120	1440	2880	4320	130	1060	1530
Máquina 2	1500	1480	600	1500	2980	3580	60	120	180	1440	2880	4320	-120	-220	560
Máquina 3	1450	0	2430	1450	1450	3880	60	60	60	1440	2880	4320	-70	1370	380
Máquina 4	300	600	410	300	900	1310	60	60	120	1440	2880	4320	1080	1920	2890

3.3.2 Processo de Realocação

Nos dias em que o tempo de processamento acumulado de uma máquina (calculado pela soma dos tempos de processamento dos trabalhos com data de entrega igual ou anterior ao dia em análise) excede a sua capacidade de produção disponível até esse momento, fica claro que a máquina não tem capacidade para concluir a produção de todos os trabalhos até esse dia dentro dos prazos de entrega. Nesses casos, torna-se necessária a realocação de alguns trabalhos, até que o limite de capacidade seja respeitado e a máquina tenha condições de produzir todos os trabalhos que lhe estão alocados, com data de entrega até esse dia.

A realocação pode seguir uma de três opções disponíveis, sendo que as máquinas principal e alternativas associadas a cada trabalho dependem exclusivamente do próprio trabalho:

- Realocação de trabalhos para a sua Máquina Alternativa 1;
- Realocação de trabalhos para a sua Máquina Alternativa 2;
- Realocação de trabalhos para uma Máquina Fictícia;

O objetivo principal do problema é garantir o processamento do maior número possível de trabalhos antes das suas respetivas datas de entrega. Assim, o primeiro objetivo da alocação é minimizar o número de trabalhos atribuídos à Máquina Fictícia, uma vez que estes trabalhos não serão considerados posteriormente no problema de escalonamento. Para atingir este objetivo, procura-se corrigir os excessos de carga identificados priorizando a realocação dos trabalhos para as suas máquinas alternativas, desde que essa realocação também respeite os limites de capacidade dessas máquinas.

Com efeito, seguindo o objetivo do problema e a hierarquia de prioridades das máquinas alternativas estabelecida no mesmo, o procedimento heurístico divide-se em duas fases. Na primeira fase, tenta-se realocar trabalhos para a Máquina Alternativa 1, ou para a Máquina Alternativa 2 nos casos em que o excesso de capacidade não pode ser resolvido exclusivamente com a realocação para a Máquina Alternativa 1. Na segunda fase, se, após essas tentativas, os limites de capacidade ainda não forem respeitados, realoca-se o número mínimo de trabalhos para a Máquina Fictícia, de modo a garantir o cumprimento da capacidade.

No entanto, para cada situação em que se identifica a necessidade de realocação de trabalhos, a alocação dos mesmos para as Máquinas Alternativas 1 e 2 não é feita de forma aleatória, mas sim seguindo uma lógica específica alinhada com os objetivos do problema. Em primeiro lugar, priorizam-se os trabalhos cujo molde é utilizado por um menor número de trabalhos até ao dia em que ocorre o excesso de capacidade na máquina. Dentro dos grupos de trabalhos que compartilham a mesma frequência de utilização de molde, a ordenação é feita de forma decrescente com base no tempo de produção dos trabalhos. Esta abordagem visa assegurar uma

utilização mais eficiente dos moldes entre as máquinas que os utilizam e, simultaneamente, reduzir o número de trabalhos que não são produzidos na sua máquina principal ou em alternativas preferenciais.

Seguindo a mesma lógica, a alocação para a Máquina Fictícia também não é realizada de forma aleatória, mas obedece a um procedimento específico. Assim, consideram-se para alocação à Máquina Fictícia apenas os trabalhos que pertencem ao subgrupo com a data de entrega mais elevada dentro dos subgrupos de datas de entrega englobados no horizonte de planeamento em que o excesso de capacidade é identificado (considera-se que este é o conjunto de trabalhos responsável pelo excesso de utilização da máquina). Dentro desse subgrupo, dá-se prioridade aos trabalhos que utilizam o molde menos frequente nesse dia, bem como aos trabalhos com maior tempo de processamento, de forma a minimizar o número de trabalhos realocados para a Máquina Fictícia.

No exemplo apresentado anteriormente, a análise da Tabela 2 mostra que há três momentos em que a folga apresenta valores negativos, indicando que as máquinas não conseguem garantir o processamento de todos os trabalhos com data de entrega com valor igual ou inferior às datas a que correspondem esses excessos de capacidade:

- Nos dias 1 e 2, para a Máquina 2;
- No dia 1, para a Máquina 3.

Esses valores negativos de folga sugerem a necessidade de realocação de trabalhos para respeitar os limites de capacidade e garantir que os prazos de entrega sejam cumpridos.

A Tabela 3 resume as máquinas alternativas onde cada trabalho considerado no exemplo apresentado pode ser processado.

Tabela 3 – Máquinas alternativas dos trabalhos A a X

Trabalho	Máquina Principal	Máquina Alternativa 1	Máquina Alternativa 2
A	Máquina 1	-	-
B	Máquina 1	-	-
C	Máquina 1	-	-
D	Máquina 1	-	-
E	Máquina 1	-	-
F	Máquina 1	-	-
G	Máquina 2	-	-
H	Máquina 2	Máquina 1	Máquina 4
I	Máquina 2	-	-
J	Máquina 2	-	-
K	Máquina 2	-	-
L	Máquina 2	-	-
M	Máquina 3	-	Máquina 2
N	Máquina 3	Máquina 1	-
O	Máquina 3	-	-
P	Máquina 3	-	-
Q	Máquina 3	-	-
R	Máquina 3	-	-
S	Máquina 4	-	-
T	Máquina 4	-	-
U	Máquina 4	-	-
V	Máquina 4	-	-
W	Máquina 4	-	-
X	Máquina 4	-	-

A análise da capacidade e consequente correção dos excessos identificados, é realizada por dia e por máquina, seguindo esta ordem de prioridade. Assim, começa-se pela análise do primeiro dia de planeamento, que deve alocar, para cada máquina, todos os trabalhos com data de entrega até esse dia.

Na Máquina 2, conforme ilustrado na Figura 7, observa-se que a soma do tempo de processamento dos Trabalhos G e H, juntamente com o tempo mínimo de *setup* entre eles (60 minutos), resulta num tempo total de utilização da máquina de 1500 minutos. Como este tempo total de utilização excede a capacidade da Máquina 2 para o dia 1, é necessário proceder à realocação de um destes dois trabalhos para garantir que o limite de capacidade é respeitado e que todos os trabalhos com data de entrega para o dia 1 podem ser concluídos dentro do prazo.

Neste exemplo, dado que o Trabalho G é um trabalho de início de dia e, portanto, não pode ser realocado, o Trabalho H é definido como o trabalho a ser realocado. Ao analisar a Tabela 3, verifica-se que o Trabalho H pode ser produzido, além da sua máquina principal, na Máquina 1 e na Máquina 4, seguindo essa ordem de prioridade.

Contudo, a Tabela 3 mostra que a folga da Máquina 1 para o dia 1 é inferior ao tempo de processamento do Trabalho H (130 minutos < 900 minutos), indicando que não é possível alocar o Trabalho H para a sua Máquina Alternativa 1. Passa-se então para a análise da possibilidade de realocar o Trabalho H para a sua Máquina Alternativa 2, a Máquina 4. A folga da Máquina 4 para o dia 1 é de 1080 minutos, o que significa que, mesmo considerando o tempo da operação de *setup* extra necessária, a Máquina 4 tem capacidade suficiente para alocar o Trabalho H (1080 minutos > 900 + 60 minutos). Consequentemente, o Trabalho H é realocado para a Máquina 4, resolvendo o excesso de capacidade na Máquina 2 para o primeiro dia.

Na Máquina 3, o tempo de processamento dos trabalhos com data de entrega para o dia 1, somado ao tempo mínimo de *setup* entre eles, também excede a capacidade de produção disponível para esse dia, exigindo que um desses trabalhos seja realocado. Como nenhum dos trabalhos é de início de dia, ambos são elegíveis para realocação.

Considerando a prioridade de máquinas alternativas, a realocação do Trabalho N para a sua Máquina Alternativa 1 tem prioridade sobre a realocação do Trabalho M para a sua Máquina Alternativa 2. No entanto, conforme indicado na Tabela 2, a Máquina Alternativa 1 do Trabalho N é a Máquina 1, que não possui folga suficiente no dia 1 para acomodar o trabalho (130 < 550 minutos). Passa-se então para a tentativa de realocação do Trabalho M na sua Máquina Alternativa 2, a Máquina 2. Embora a Máquina 2 agora tenha uma folga de 780 minutos para o dia 1, essa folga ainda é insuficiente para alocar o Trabalho M.

Diante deste cenário, a única solução é alocar um dos trabalhos para a Máquina Fictícia. Seguindo os critérios de alocação para essa máquina, começa-se por analisar o número de vezes que cada molde utilizado pelos dois trabalhos é empregue até ao dia 1. Como ambos os moldes são utilizados de forma exclusiva (apenas uma vez), passa-se para o segundo critério, que considera o tempo de processamento. Assim, procede-se à alocação do trabalho com o maior tempo de processamento, o Trabalho M, para a Máquina Fictícia, resolvendo o excesso de capacidade da Máquina 3 para o dia 1.

Assim, o esquema da distribuição final obtida após as realocações é representado na Figura 8, sendo que, agora, conforme indicado pela análise da Tabela 4, o mesmo respeita os limites de capacidade de cada máquina. Esta distribuição ajustada assegura que todas as máquinas operam dentro da sua capacidade diária, garantindo o cumprimento dos prazos de entrega do maior número de trabalhos possível.

Tabela 4 - Análise de utilização, capacidade e folga do sistema de produção após realocações

	∑ Tempo Processamento			Tempo Processamento Acumulado			Tempo Mínimo Setup			Capacidade Acumulada			Folga		
	Dia 1	Dia 2	Dia 3	Dia 1	Dia 2	Dia 3	Dia 1	Dia 2	Dia 3	Dia 1	Dia 2	Dia 3	Dia 1	Dia 2	Dia 3
Máquina 1	1250	450	970	1250	1700	2670	60	120	120	1440	2880	4320	130	1060	1530
Máquina 2	600	1480	600	600	2080	2680	60	120	180	1440	2880	4320	780	680	1460
Máquina 3	550	0	2430	550	550	2980	60	60	60	1440	2880	4320	830	2270	1280
Máquina 4	1200	600	410	1200	1800	2210	60	60	120	1440	2880	4320	180	1020	1990

Necessidades de Produção Agrupadas por Máquina e Data de Entrega

	Dia 1			Dia 2	Dia 3	
Máquina 1	A MO. 10	B MO. 10	C MO. 20	D MO. 30	E MO.10	F MO. 20
Tempo processamento (minutos)	250	300	700	450	300	670

	Dia 1	Dia 2			Dia 3
Máquina 2	G MO. 40	I MO. 50	J MO. 40	K MO. 60	L MO. 70
Tempo processamento (minutos)	600	350	460	670	600

	Dia 1	Dia 3			
Máquina 3	N MO. 90	O MO. 80	P MO. 80	Q MO. 80	R MO. 90
Tempo processamento (minutos)	550	300	890	700	540

	Dia 1			Dia 2		Dia 3	
Máquina 4	S MO. 100	T MO. 110	H MO. 50	U MO. 100	V MO. 110	W MO. 120	X MO. 120
Tempo processamento (minutos)	100	200	900	400	200	150	260

	Dia 1
Máquina Fictícia	M MO. 80
Tempo processamento (minutos)	900

Figura 8 – Distribuição final dos trabalhos após realocações

3.3.3 Pseudocódigo

A Figura 9 representa o pseudocódigo do procedimento heurístico explicado nas subsecções anteriores. A heurística avalia se todas as máquinas têm condições de processar os trabalhos que representam os pedidos dos clientes antes da sua data de entrega sem exceder os limites de capacidade. Caso os limites sejam excedidos, procede-se à realocação (de acordo com as prioridades de máquinas alternativas existentes) dos trabalhos até que a capacidade seja aceitável.

```
procedure Realocar_Trabalhos:

adicionar trabalhos de início de dia às necessidades de produção
dias ← dias únicos de data de entrega
máquinas ← máquinas únicas presentes nas necessidades de produção
for each dia in dias:
    for each máquina in máquinas:
        necessidades_anteriores_dia ← trabalhos da máquina atual com data de entrega ≤ dia
        min_tempo_setup ← (número de ferramentas em necessidades_anteriores_dia - 1) * 60
        while soma de tempos de produção + min_tempo_setup > tempo disponível até ao dia:
            alocacao_completa ← FALSO
            subconjunto_alt1 ← trabalhos com 'Alternativa 1' ≠ ∅, por ordem decrescente do tempo processamento
            for each trabalho in subconjunto_alt1:
                if tempo_folga_alt1 >= tempo de processamento trabalho + 60:
                    atribuir trabalho à 'Alternativa 1'
                    recalcular necessidades_anteriores_dia
                    alocacao_completa ← VERDADEIRO
                    interromper
                end if;
            end for;
            if alocacao_completa == FALSO:
                subconjunto_alt2 ← trabalhos com 'Alternativa 2' ≠ ∅, por ordem decrescente do tempo
                                   processamento
                for each trabalho in subconjunto_alt1:
                    if tempo_folga_alt2 >= tempo de processamento trabalho + 60:
                        atribuir trabalho à 'Alternativa 2'
                        recalcular necessidades_anteriores_dia
                        alocacao_completa ← VERDADEIRO
                        interromper
                    end if;
                end for;
            end if;
            if alocacao_completa == FALSO:
                trabalhos_candidatos ← trabalhos do dia, com menor uso de molde, por ordem decrescente do tempo
                                       de produção
                atribuir primeiro trabalho da lista à 'Máquina Fictícia'
                recalcular necessidades_anteriores_dia
            end if;
        end while;

end Realocar_Trabalhos.
```

Figura 9 – pseudocódigo da função *Realocar_Trabalhos*

3.4 GRASP para UPMSR-PS

As metaheurísticas são geralmente controladas por vários fatores e parâmetros que podem ser alterados de forma a melhorar a pesquisa de uma solução para o problema de otimização em questão (Dillen et al., 2021). Neste sentido, a construção da metaheurística desenvolvida neste estudo rege-se, sempre que possível, pela utilização dos melhores fatores e parâmetros apresentados nos exemplos mencionados no capítulo *2.6 - Modelos de Referência*.

Este subcapítulo dedica-se à apresentação da metaheurística GRASP desenvolvida para resolver o problema de programação de máquinas paralelas dedicadas com múltiplas alternativas, *setups*, recursos adicionais e datas de entrega. Os subcapítulos seguintes apresentam em detalhe cada uma das fases do GRASP, a lógica subjacente e a explicação das opções utilizadas.

3.4.1 Procedimento Principal

O pseudocódigo apresentado na Figura 10 apresenta o código do procedimento principal do algoritmo GRASP desenvolvido no presente estudo.

O procedimento principal da metaheurística é composto por um ciclo que agrega três fases distintas: a fase construtiva, onde os trabalhos são inseridos sequencialmente numa solução parcial de acordo com diversos critérios até se alcançar a primeira solução completa; a fase de reparação, em que a solução obtida na fase anterior é ajustada face às restrições relacionadas com a utilização de recursos adicionais; e a fase de procura local, composta por duas estratégias distintas desenvolvidas para aperfeiçoar as soluções iniciais obtidas em cada iteração. De forma resumida, em cada iteração é construída uma solução inicial, que depois de reparada, é submetida a algoritmos de pesquisa local que exploram o espaço da solução inicial em busca de melhores soluções. No final da fase de pesquisa local, compara-se a melhor solução obtida nessa iteração com a melhor solução global do problema, que será a solução devolvida no final do mesmo.

```
procedure GRASP

Ler_Dados_Entrada():
tempo_inicial ← obter tempo atual
while tempo_decorrido < tempo_máximo do:
    solução, solução_setup ← Fase Construtiva ()
    solução, solução_setup ← Fase de Reparação (solução, solução_setup)
    melhor_solução_atual, melhor_solução_setup_atual ←  solução, solução_setup
    if melhor_solução_atual > solução_final:
        solução_final, solução_setup_final ← melhor_solução_atual, melhor_solução_setup_atual
        atualizar indicadores de performance
    end if;
    melhor_solução_atual, melhor_solução_setup_atual ← Pesquisa_Local 1 (melhor_solução_atual,
                                                       melhor_solução_setup_atual)
    melhor_solução_atual, melhor_solução_setup_atual ← Pesquisa_Local 2 (melhor_solução_atual,
                                                       melhor_solução_setup_atual)
    if melhor_solução_atual > solução_final:
        solução_final, solução_setup_final ← melhor_solução_atual, melhor_solução_setup_atual
        atualizar indicadores de performance
    end if;
    tempo_decorrido ← tempo_atual – tempo_inicial
end while;
return solução_final, solução_setup_final

end GRASP.
```

Figura 10 – Pseudocódigo da metaheurística GRASP desenvolvida

3.4.2 Fase Construtiva

Na fase construtiva, o objetivo é criar uma solução inicial sem considerar as restrições de recursos impostas ao problema, que servirá como ponto de partida para as fases subsequentes do algoritmo. Esta fase começa com a formação de uma lista (Lista N), composta pelos trabalhos alocados às 33 máquinas que integram o sistema de produção durante a fase de alocação. Os trabalhos alocados à Máquina Fictícia são excluídos e, portanto, não são incluídos no planeamento. Da mesma forma, os trabalhos de início de dia também não são considerados na Lista N, pois a sua posição já está previamente definida como a primeira posição na máquina à qual pertencem.

Essencialmente, a Lista N representa todas as necessidades de produção requeridas para satisfazer os pedidos dos clientes ao longo de um horizonte temporal de sete dias.

A lista N é ordenada por ordem crescente do prazo de entrega dos trabalhos e, para cada trabalho *j* da lista, aplica-se o seguinte procedimento baseado na estratégia denominada de *Dynamic Multiple Insertion* (DMI) apresentada em (Diana et al., 2015):

1. Define-se o número de posições nas quais o trabalho j pode ser inserido na solução parcial da máquina i à qual está alocado. Se existirem n trabalhos sequenciados na solução parcial da máquina i, o trabalho j poderá ser inserido em $n+1$ posições caso nenhum dos trabalhos seja de início de dia, ou em n posições caso contrário. A Figura 11 representa graficamente um cenário em que o trabalho j pode ser inserido em $3+1$ posições se não houver nenhum trabalho de início de dia na máquina i e um cenário oposto em que existe na máquina i um trabalho de início de dia e, por isso, o trabalho j só pode ser inserido em n posições.

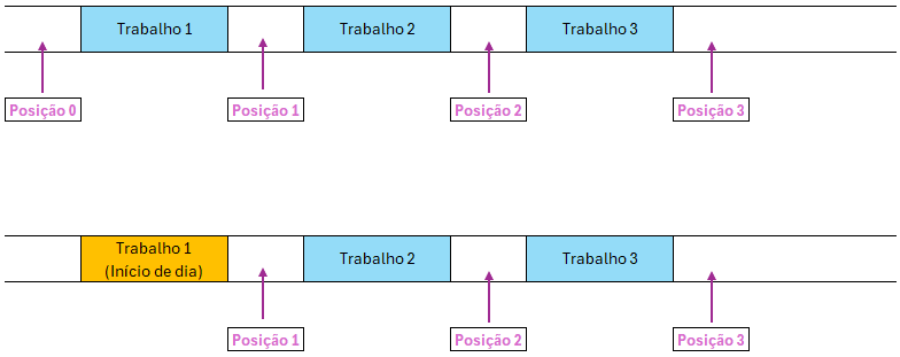


Figura 11 – Número de posições em que o trabalho j pode ser inserido numa solução parcial

2. Calcula-se o valor do coeficiente λ, apresentado na equação (6) da subsecção *3.2.4 – Coeficiente de Classificação*, para cada posição onde o trabalho j pode ser inserido.

3. Constrói-se a Lista de Candidatos (CL) que consiste numa lista das posições onde trabalho j pode ser inserido, ordenada por ordem crescente do valor de λ obtido para cada posição

4. Constrói-se a Lista Restrita de Candidatos (RCL) que inclui uma percentagem das primeiras posições que compõe a CL.

5. Seleciona-se aleatoriamente um elemento de RCL, correspondente à posição escolhida para inserir o trabalho j.

6. O trabalho j é inserido na posição selecionada da solução parcial, considerando as necessidades de *setup* com os trabalhos inseridos na posição anterior e posterior (caso aplicável). A solução de *setups* é editada em conformidade.

Após a conclusão do procedimento, o trabalho j é removido da Lista N e passa-se para a análise do próximo trabalho da lista de trabalhos pendentes. Este processo é repetido para todos os trabalhos da Lista N e, no final da fase construtiva, obtém-se uma solução completa, bem como a respetiva solução de *setups*.

```
procedure Fase Construtiva

N ← lista de trabalhos que representa as necessidades de produção excluindo os trabalhos da Máquina Fictícia
solução ← trabalhos de início de dia
solução_setup ← DataFrame()
for each trabalho in N:
    CL ← criar Lista de Candidatos de acordo com o valor λ atribuído a cada posição
    RCL ← construir um Lista Restrita de Candidatos correspondente a 25% das primeiras posições de CL
    posição_escolhida ← Escolher aleatoriamente uma posição presente em RCL
    solução, solução_setup ← inserir  o trabalho  j na posição escolhida e atualizar a solução de setup
end for;
return solução, solução_setup

end Fase Construtiva.
```

Figura 12 – Pseudocódigo da fase construtiva da metaheurística GRASP

A Tabela 5 apresenta a Lista N referente ao exemplo descrito no subcapítulo *3.3. – Heurística para Alocação Trabalho-Máquina*. É importante destacar que os trabalhos na Lista N estão ordenados por ordem crescente da sua data de entrega. Além disso, foram excluídos os trabalhos alocados à Máquina Fictícia (Trabalho M) e os trabalhos de início de dia (Trabalhos A, G e S), que já estão posicionados na solução parcial no início do procedimento construtivo, conforme ilustrado na Figura 13. Esta estrutura da Lista N facilita a construção de uma solução inicial organizada e alinhada com os objetivos de planeamento definidos.

Tabela 5 – Lista N para o exemplo apresentado na fase de alocação

Trabalho	Data de entrega	Máquina
B	Dia 1	Máquina 1
C	Dia 1	Máquina 1
N	Dia 1	Máquina 3
T	Dia 1	Máquina 4
H	Dia 1	Máquina 4
D	Dia 2	Máquina 1
I	Dia 2	Máquina 2
J	Dia 2	Máquina 2
K	Dia 2	Máquina 2
U	Dia 2	Máquina 4
V	Dia 2	Máquina 4
E	Dia 3	Máquina 1
F	Dia 3	Máquina 1
L	Dia 3	Máquina 2
O	Dia 3	Máquina 3
P	Dia 3	Máquina 3
Q	Dia 3	Máquina 3
R	Dia 3	Máquina 3
W	Dia 3	Máquina 4
X	Dia 3	Máquina 4

A Figura 14 apresenta o pseudocódigo utilizado para o cálculo do coeficiente λ, que é utilizado para avaliar a inserção do trabalho *j* em cada uma das posições possíveis na solução parcial. O coeficiente λ permite quantificar o impacto de inserir o trabalho *j* em diferentes posições, considerando critérios como datas de entrega, tempos de *setup* e outros fatores relevantes, sendo fundamental para orientar a fase construtiva, auxiliando na escolha da posição mais vantajosa para o trabalho *j* na solução parcial. Note-se que quando existe um trabalho de início de dia atribuído à máquina, a posição 0 é excluída da análise.

Para cada posição avaliada, o procedimento inicia-se pela identificação dos trabalhos que se encontram imediatamente antes e após a posição em análise, caso existam. Em seguida, calcula-se o *setup* original entre esses dois trabalhos e, posteriormente, os novos tempos de *setup* gerados pela inserção do trabalho *j* entre os trabalhos adjacentes identificados. Este processo permite avaliar os ganhos ou perdas em termos de tempos de *setup*, resultantes da inserção do trabalho *j* em cada posição.

Posteriormente, avalia-se a diferença entre a data de entrega do trabalho *j* e a data de entrega do trabalho que se encontra na posição seguinte à posição em análise, aplicando-se as penalidades correspondentes, com base na diferença identificada. Adicionalmente, é analisado o impacto da inserção do trabalho *j* na posição em análise, considerando tanto o atraso potencial do próprio trabalho *j* quanto os atrasos que possam ser induzidos nos trabalhos subsequentes.

Por fim, identifica-se o último trabalho da solução parcial que utiliza o mesmo molde que o trabalho *j* e aplica-se uma penalização se a posição em análise for igual ou anterior à posição desse trabalho. Assim, após a aplicação de todas as penalidades e ajustes, é possível calcular e retornar o valor do coeficiente λ para a inserção do trabalho *j* em cada posição.

Necessidades de Produção Agrupadas por Máquina e Data de Entrega

	Dia 1
Máquina 1	A MO. 10
Tempo processamento (minutos)	250

	Dia 1
Máquina 2	G MO. 40
Tempo processamento (minutos)	600

Máquina 3
Tempo processamento (minutos)

	Dia 1
Máquina 4	S MO. 100
Tempo processamento (minutos)	100

Figura 13 – Solução inicial antes do início da heurística construtiva

```
procedure Calcular_Lambda

if posição > 0:
    trabalho_anterior ← trabalho na posição – 1
end if;
if posição < última posição + 1:
    trabalho_seguinte ← trabalho na posição – 1
end if;
original_setup ← tempo de setup entre trabalho_anterior e trabalho_seguinte
novo_setup_anterior ← tempo de setup entre trabalho_anterior e trabalho j
novo_setup_seguinte ← tempo de setup entre trabalho j e trabalho_seguinte
tf_anterior ← tempo final de trabalho_anterior ou 0 se não existir trabalho_anterior
data_entrega ← data de entrega do trabalho j
tf_proposto_j ← tf_anterior + (novo_setup_anterior + 2 if novo_setup_anterior > 0, else 1) + trabalho j ['tempo_pr
trabalhos_atrasados ← 0
atraso ← max (0, tf_proposto_j – data_entrega)
if atraso > 0:
    trabalhos_atrasados ← trabalhos_atrasados + 1
end if;
diferença_dias ← trabalho j ['Dia'] – trabalho_seguinte['Dia'] (caso trabalho_seguinte exista)
penalidade_dias ← 0
if diferença_dias > 0:
    penalidade_dias ← penalidade_dias + diferença_dias * 50
end if;
trabalhos_atrasados ← trabalhos_atrasados + trabalhos atrasados após posição depois da inserção do trabalho j
molde_trabalho_j ← molde do trabalho j
última_posição_molde ← última posição na máquina do trabalho j com molde_trabalho_j
penalidade_molde ← 0
if última_posição_molde is not None:
    if posição <= última_posição_molde:
        penalidade_molde ← 500
    else:
        penalidade_molde ← 0
    end if;
end if;
valor_lambda ← penalidade_dias + trabalhos_atrasados*100 + penalidade_molde + ((novo_setup_anterior +
                    novo_setup_seguinte - original_setup))
return valor_lambda

end Calcular_Lambda.
```

Figura 14 - Pseudocódigo da função Fase de Reparação

Conforme mencionado anteriormente, a Lista Restrita de Candidatos (RCL) é composta por um subconjunto das posições com melhor classificação na Lista de Candidatos (CL). No entanto, é necessário definir quantos dos melhores elementos da CL irão compor a RCL. Embora existam diferentes abordagens, considerando que a CL é ordenada da melhor (menor valor de λ) para a pior posição (maior valor de λ) para a inserção do trabalho *j* na solução parcial, optou-se por definir a RCL como uma percentagem de 25% do tamanho da CL ($\alpha = 0.25$), conforme estabelecido no estudo de referência de (Yepes-Borrero et al., 2020).

Após a definição da RCL, e seguindo as conclusões apresentadas por (Yepes-Borrero et al., 2020) e (Lopez-Esteve et al., 2023), é selecionado aleatoriamente um dos elementos da RCL para

definir a posição onde o trabalho será inserido, em vez de se optar pela melhor posição disponível. Esta abordagem introduz uma randomização na construção da solução inicial em cada iteração, o que, de acordo com os estudos de referência, melhora significativamente o desempenho do GRASP.

Embora existam diversas opções para definir a distribuição de probabilidade associada à seleção de cada elemento da RCL, adota-se a distribuição que, conforme indicado por (Lopez-Esteve et al., 2023), contribui para um melhor desempenho do algoritmo. Neste caso, a probabilidade de seleção de cada elemento da RCL segue uma distribuição decrescente, dada por $(s-k+1)\frac{2}{s(s+1)}$, onde s representa o tamanho da Lista Restrita de Candidatos e k a posição do elemento na lista de candidatos.

Tabela 6 - Distribuição de probabilidade associada à escolha de cada elemento da RCL

Posição em RCL	1	2	…	s
Probabilidade	$s\frac{2}{s(s+1)}$	$(s-1)\frac{2}{s(s+1)}$	…	$\frac{2}{s(s+1)}$

Segue-se a aplicação do processo construtivo descrito anteriormente ao exemplo apresentado na subsecção *3.3. – Heurística para Alocação Trabalho-Máquina*. Uma vez que a sequenciação nas diferentes máquinas não influencia a sequenciação nas restantes, o processo construtivo será aplicado a cada máquina de forma independente, seguindo a ordem pela qual os trabalhos surgem na Lista N da Tabela 5.

Começando pela Máquina 1, onde já se encontra sequenciado na posição 0 o Trabalho A, de início de dia. O primeiro trabalho que surge na Lista N para esta máquina é o Trabalho B, que será inserido na posição 1, uma vez que é a única posição na qual pode ser inserido. Segue-se o Trabalho C que pode ser inserido na posição 1 ou na posição 2 na solução parcial da Máquina 1. Analisando a Tabela 7, que apresenta não só o valor λ, como também o valor de cada parâmetro incluído no cálculo do mesmo, percebe-se que a melhor posição para inserir o Trabalho C é a segunda posição, logo após o Trabalho B. Como a RCL é composta apenas por 25% da CL, a RCL contém apenas a posição 2 e o Trabalho C é então inserido na segunda posição.

Tabela 7 – Valor do coeficiente λ para cada posição onde o Trabalho C pode ser inserido

Posição	Penalidade Dias	Setup Original	Novo Setup (Antes)	Novo Setup (Depois)	Trabalhos Atrasados (x 100)	Penalidade Molde	Valor λ
1	0	0	60	60	0	0	120
2	0	0	60	0	0	0	60

Passando para o trabalho seguinte na Lista N, alocado à Máquina 1, surge o Trabalho D. Uma vez que a solução parcial da máquina 1 conta agora com três trabalhos na sua sequência (Trabalhos A, B e C), sendo um deles um trabalho de início de dia, o Trabalho D pode ser inserido em três posições distintas. A Tabela 8 representa, para cada uma das posições onde o Trabalho D pode ser inserido, o valor do coeficiente λ e dos parâmetros que o constituem. Através da análise da Tabela, é possível observar que as duas primeiras posições são fortemente penalizadas quer pela penalidade aplicada à inserção de um trabalho antes de outro com data de entrega inferior quer por provocarem um atraso num trabalho.

Tabela 8 - Valor do coeficiente λ para cada posição onde o Trabalho D pode ser inserido

Posição	Penalidade Dias	Setup Original	Novo Setup (Antes)	Novo Setup (Depois)	Trabalhos Atrasados (x 100)	Penalidade Molde	Valor λ
1	50	0	60	60	1	0	270
2	50	60	60	60	1	0	210
3	0	0	60	0	0	0	60

O próximo trabalho da Lista N, alocado à Máquina 1, é o Trabalho E que utiliza o molde MO.10, à semelhança dos Trabalhos A e B na mesma máquina. O Trabalho E, com data de entrega de dia 3, pode ser agora inserido em 4 posições, sendo o valor do coeficiente λ para cada uma delas apresentado na Tabela 9. Apesar do Trabalho E utilizar partilhar o mesmo molde dos Trabalhos A e B, a sua data de entrega é bastante superior à data de entrega destes trabalhos, sendo, por esse motivo, as posições adjacentes a estes trabalhos (posições 1 e 2) fortemente penalizadas no que diz respeito ao parâmetro da penalidade de dias e de trabalhos atrasados. Neste sentido, uma vez que a CL é composta por apenas 4 trabalhos, a posição 4 compõe de forma exclusiva a RCL e o Trabalho E é inserido na quarta posição da sequencia da solução parcial da máquina 1.

Tabela 9 - Valor do coeficiente λ para cada posição onde o Trabalho E pode ser inserido

Posição	Penalidade Dias	Setup Original	Novo Setup (Antes)	Novo Setup (Depois)	Trabalhos Atrasados (x 100)	Penalidade Molde	Valor λ
1	100	0	0	0	1	500	700
2	100	60	0	60	1	0	200
3	50	60	60	60	0	0	110
4	0	0	60	0	0	0	60

Por fim, o último trabalho da Lista N atribuído à máquina 1 é o Trabalho F, com data de entrega de dia 3 e que utiliza o molde MO.20. Neste contexto, ao analisar a Tabela 10, que apresenta o valor do coeficiente λ para cada posição onde o Trabalho E pode ser inserido, conclui-se que as duas primeiras posições são fortemente penalizadas pelo parâmetro da Penalidade de Dias. Além disso, essas posições provocam atrasos nos trabalhos subsequentes e acarretam uma penalização adicional pelo facto de existir um trabalho sequenciado após essas posições que também utiliza o molde MO.20.

A inserção do Trabalho F na terceira posição implica apenas uma penalização por colocar um trabalho com data de entrega para o dia 3 antes de um trabalho com data de entrega para o dia 2, mas, ao mesmo tempo, otimiza a utilização do molde MO.20, uma vez que ocorre logo após

o último trabalho já sequenciado que também utiliza este molde, evitando assim quer a penalização de molde, quer a criação de uma operação de *setup* extra.

Por outro lado, a inserção do Trabalho F na quarta e quinta posições não acarreta nenhuma Penalidade de Dias, mas também não rentabiliza a utilização do molde MO.20, resultando num valor de λ de 60 para ambas as posições.

Neste exemplo, é possível notar que a escolha de um valor de penalidade de dias inferior ao tempo de *setup* não foi feita de forma arbitrária. Desta forma, caso a diferença de datas de entrega entre trabalhos consecutivos seja de apenas um dia, a inserção do trabalho numa posição que otimiza o número de *setups* torna-se preferencial, equilibrando o custo das penalidades e incentivando uma melhor sequência de produção.

Tabela 10 – Valor do coeficiente λ para cada posição onde o Trabalho F pode ser inserido

Posição	Penalidade Dias	Setup Original	Novo Setup (Antes)	Novo Setup (Depois)	Trabalhos Atrasados (x 100)	Penalidade Molde	Valor λ
1	100	0	60	60	1	500	820
2	100	60	60	0	1	500	700
3	50	60	0	60	0	0	50
4	0	60	60	60	0	0	60
5	0	0	60	0	0	0	60

A Lista de Candidatos obtida para a inserção do Trabalho F na sequência da Máquina 1 encontra-se apresentada na Tabela 11.

Tabela 11 - Lista de Candidatos para a inserção do Trabalho F

Posição	3	5	4	2	1
Valor λ	50	60	60	700	820

Como a CL é composta por 5 posições, desta vez a RCL passa a incluir não apenas a melhor posição da CL, mas as duas primeiras posições da Lista de Candidatos. A Tabela 12 ilustra a composição da RCL, bem como a distribuição de probabilidades associada a cada uma das posições que a compõem, indicando a probabilidade de cada posição ser selecionada para a inserção do trabalho.

Tabela 12 - Lista Restrita de Candidatos para a inserção do Trabalho F

Posição em RCL	1	2
Posição na solução	3	5
Probabilidade	0.667	0.333

Aplicando o processo realizado para a Máquina 1 às restantes 3 máquinas consideradas no exemplo, obtém-se a sequência de produção para cada máquina, conforme apresentado na Figura 15. É importante notar que esta sequência ainda não considera as restrições de utilização de recursos definidas no problema, sendo, portanto, necessário ajustá-la na fase seguinte: a Fase de Reparação. Essa fase será responsável por corrigir a sequência inicial para garantir o cumprimento das restrições de recursos, tornando a solução viável dentro do contexto do problema.

Necessidades de Produção Agrupadas por Máquina e Data de Entrega

	Dia 1		Z	Dia 1	Dia 3	Z	Dia 2	Z	Dia 3
Máquina 1	A MO. 10	B MO. 10		C MO. 20	F MO. 20		D MO. 30		E MO.10
Tempo processamento (minutos)	250	300	60	700	670	60	450	60	300

	Dia 1	Dia 2	Z	Dia 2	Z	Dia 2	Z	Dia 3
Máquina 2	G MO. 40	J MO. 40		I MO. 50		K MO. 60		L MO. 70
Tempo processamento (minutos)	600	460	60	350	60	670	60	600

	Dia 1	Dia 3	Z	Dia 3		
Máquina 3	N MO. 90	R MO. 90		P MO. 80	Q MO. 80	O MO. 80
Tempo processamento (minutos)	550	540	60	890	700	300

	Dia 1	Z	Dia 1	Z	Dia 1	Z	Dia 2	Dia 3	Z	Dia 2	Z	Dia 3
Máquina 4	S MO. 100		T MO. 110		H MO. 50		U MO. 100	X MO. 100		V MO. 110		W MO. 120
Tempo processamento (minutos)	100	60	200	60	900	60	400	260	60	200	60	150

	Dia 1
Máquina Fictícia	M MO. 80
Tempo processamento (minutos)	900

Z = *Setup*

Figura 15 – Sequência de produção por máquina

3.4.3 Fase de reparação

Após a alocação e sequenciação de todos os trabalhos, é necessário avaliar a solução obtida quanto ao cumprimento dos limites de utilização de recursos adicionais. Especificamente, é fundamental assegurar que:

- Para cada instante t da solução, o número de operadores e de equipas de *setup* utilizados não excede os valores disponíveis desses recursos para esse mesmo instante.

- Não existe nenhum instante t da solução em que um molde é utilizado em mais do que uma máquina;

- Para cada molde utilizado em mais do que uma máquina, a operação de *setup* que insere o molde numa máquina só pode iniciar após o término da operação de *setup* que o retira da última máquina na qual foi utilizado;

O pseudocódigo apresentado na Figura 16 ilustra o procedimento de reparação utilizado na metaheurística GRASP desenvolvida no presente estudo. Este procedimento consiste num ciclo composto por duas funções principais: a função *Fase_de_Reparação*, que assegura o cumprimento dos limites de recursos adicionais, e a função *Outros_Conflitos_Moldes*, que garante que as operações de *setup* que inserem um molde partilhado (molde utilizado em mais de uma máquina) numa máquina só são realizadas após esse molde ter sido removido da máquina na qual foi utilizado anteriormente. Como as intervenções realizadas por ambas as funções podem alterar as condições asseguradas pela outra, optou-se por estruturar um ciclo de forma que, após cada execução da função *Outros_Conflitos_Moldes*, a solução seja novamente corrigida em termos da utilização de recursos, garantindo assim o cumprimento de todas as restrições impostas pelo problema. O ciclo termina quando, após a execução da função *Fase_de_Reparação*, não são identificados conflitos associados aos moldes partilhados.

Este processo iterativo é essencial para assegurar que tanto a gestão dos recursos adicionais como a sincronização dos moldes partilhados entre as máquinas são resolvidas de maneira integrada, mantendo a coerência e viabilidade da solução final.

```
procedure Fase de Reparação e Outros Conflitos Moldes

solução, solução_setup ← Fase de Reparação (solução, solução_setup)
while Verdadeiro:
    solução, solução_setup, min_ti_adiado ← Outros_Conflitos_Moldes (solução, solução_setup)
    if min_ti_adiado is not None:
        break
    end if;
    solução, solução_setup ← Fase de Reparação (solução, solução_setup, tempo_início = min_ti_adiado)
end while;
return solução, solução_setup

end Fase de Reparação e Outros Conflitos Moldes.
```

Figura 16 - Pseudocódigo do ciclo de reparação e de resolução de conflitos de moldes

A função *Fase_de_Reparação* começa com a definição do primeiro instante para o qual é analisado o cumprimento das restrições de capacidade, definido como o valor mais baixo de t_i entre os presentes na solução e na solução de *setups*.

Nos casos em que a *Fase_de_Reparação* surge após a função *Outros_Conflitos_Moldes*, o primeiro valor de t analisado corresponde ao valor de t_i mais baixo de entre as operações de *setup* adiadas na função *Outros_Conflitos_Moldes*, uma vez que antes desse instante a solução não sofre alterações e, por isso, mantém o cumprimento das restrições previamente estabelecido. Desta forma, a *Fase_de_Reparação* concentra-se apenas nos pontos onde as

mudanças introduzidas pela função *Outros_Conflitos_Moldes* possam ter gerado novos conflitos ou necessidades de ajuste.

A função *Fase_de_Reparação* será aplicada entre o primeiro valor t definido e o valor de t_i mais alto (definido de forma dinâmica após cada iteração) presente na solução e na solução de *setup*. Para cada instante t analisado, a função *Corrigir_Sobrecargas* é executada, assegurando concretamente o cumprimento das restrições de utilização de moldes em diferentes máquinas e impondo os limites de utilização dos recursos adicionais, de acordo com o número de equipas de *setup* e operadores disponíveis no turno.

Após serem corrigidos os consumos de recursos para o instante t em análise, determina-se o instante temporal a ser analisado na iteração seguinte, que corresponde ao menor valor entre três opções: o valor mais baixo de t_i da solução após o valor atual em análise, o valor mais baixo de t_i da solução de *setup* após o valor atual em análise, e o instante temporal correspondente ao final do turno em análise. Esta estratégia é utilizada com base na premissa de que, enquanto não surgir um novo *setup*, um novo trabalho, ou um novo turno, não ocorrem alterações na necessidade de recursos. Os instantes finais de cada turno também são analisados, com o objetivo de garantir que não há nenhuma operação de *setup* em curso nesses momentos e de assegurar que os limites de recursos adicionais para o novo turno são respeitados. Adicionalmente, a análise dos instantes iniciais de cada turno permite ajustar o volume de produção aos novos valores de recursos disponíveis para esse turno, garantindo uma transição eficiente entre turnos e uma adaptação dinâmica da capacidade produtiva de acordo com os recursos disponíveis.

```
procedure Fase de Reparação

t ← valor mais baixo de t_i presente na solução e na solução_setup
while t <= valor mais alto de t_i presente na solução e na solução_setup:
    max_operadores ← número disponível de operadores no turno que inclui t
    max_equipas_setup ← número disponível de equipas de setup no turno que inclui t
    if operadores > max_operadores or equipas_setup > max_equipas_setup:
        solução, solução_setup ← Corrigir_Sobrecargas(solução, solução_setup)
    end if;
    próximo_t_ solução ← menor {t_i | t_i > t e t_i ∈ solução}
    próximo_t_ solução_setup ← menor {t_i | t_i > t e t_i ∈ solução_setup}
    fim_turno ← instante final do turno atual
    inicio_turno ← instante inicial do turno seguinte
    próximos_tempos ← [próximo_t_ solução, próximo_t_ solução_setup, fim_turno, inicio_turno]
    t ← mínimo {próximos_tempos}
end while;
return solução, solução_setup

end Fase de reparação.
```

Figura 17 - Pseudocódigo da função *Fase_de_Reparação*

Tal como referido anteriormente, é na função *Corrigir_Sobrecargas*, inserida na função *Fase_de_Reparação*, onde se garante o cumprimento das restrições de recursos e de utilização de moldes em mais do que uma máquina, impostas pelo problema.

No início desta função, caso o instante t em análise corresponda ao instante final de um turno, adiam-se os *setups* ativos nesse instante para começarem apenas no início do turno seguinte, garantindo que não existem *setups* em curso na transição de turnos.

Assim, exceto nas situações em que as limitações de recursos e as restrições de utilização de moldes são respeitadas, fazendo com que a função não realize nenhuma operação, existem três possíveis ações que são executadas de forma cíclica até que todas as restrições de recursos adicionais sejam cumpridas:

1. Verifica-se o excesso de utilização de equipas de *setup* utilizadas no instante t, acionando-se a função *Sobrecarga_Setup* para diminuir essa utilização;

2. Se são utilizados mais operadores do que os disponíveis no instante t, executa-se a função *Sobrecarga_Operadores*, que diminui essa utilização de operadores;

3. Se o mesmo molde é utilizado por mais do que uma máquina em simultâneo no instante t, aciona-se a função *Sobrecarga_Moldes* para garantir que apenas um dos trabalhos que utiliza esse molde é processado nesse instante.

```
procedure Corrigir_Sobrecargas

if t corresponde a um instante de final de turno:
    adiar todos os setups ativos em t  para o início do turno seguinte
end if;
while Verdadeiro:
    operadores ativos ← soma dos operadores dos trabalhos na solução onde t_i ≤ t and t_f ≥ t
    equipas_setup_ativas ← contar os setups na solução_setup onde t_i ≤ t and t_f ≥ t
    conflitos_moldes ← verificar se mais do que um dos trabalhos ativos utiliza o mesmo molde
    if operadores ativos ≤ max_operadores and equipas_setup_ativas ≤ max_equipas_setup:
        break
    end if;
    if equipas_setup_ativas > max_equipas_setup:
        solução, solução_setup ← Sobrecarga_Setup (solução, solução_setup)
    end if;
    if operadores ativos > max_operadores:
        solução, solução_setup ← Sobrecarga_Operadores (solução, solução_setup)
     end if;
    if conflitos_moldes == Verdadeiro:
        solução, solução_setup ← Sobrecarga_Moldes (solução, solução_setup)
    end if;
      return solução, solução_setup

end Corrigir_Sobrecargas.
```

Figura 18 - Pseudocódigo da função *Corrigir_Sobrecargas*

A função *Sobrecarga_Setup*, quando acionada, atua para reduzir a utilização de equipas de *setup* no instante t em análise. Como não é logicamente viável interromper uma operação de *setup* já em curso para iniciar outra, a função identifica, na solução de *setup*, os *setups* que estão programados para iniciar no instante t, designando-os como *novos_setups*. Apenas esses novos *setups* podem ser adiados, enquanto os *setups* que começaram antes do instante t permanecem inalterados.

No entanto, em alguns casos, é possível que apenas parte dos *novos_setups* sejam adiados, mantendo-se os restantes inalterados. Nesses cenários, é necessário decidir quais dos *novos_setups* devem ser mantidos e quais devem ser adiados.

Para isso, calcula-se o tempo de folga mínimo das máquinas que contêm *novos_setups*, analisando, para cada trabalho dessas máquinas que inicia após o instante t, a diferença entre o instante em que o trabalho termina e a sua data de entrega em minutos (obtida multiplicando o dia de entrega do trabalho pela duração de um dia de trabalho em minutos — 1440 minutos). Os *setups* que permanecem são aqueles associados às máquinas com os menores valores de folga mínima, pois entende-se que o adiamento nesses casos tem um impacto negativo mais significativo na qualidade da solução. De forma resumida, as máquinas cujos trabalhos têm menor folga são as que possuem menor capacidade para adiar trabalhos ou *setups* sem causar atrasos e comprometer os prazos de entrega.

Após definir os *setups* a serem adiados, é-lhes atribuído um novo valor de t_i. Este novo t_i será uma unidade superior ao instante final do *setup* em curso que termina mais cedo ou corresponderá ao instante inicial do turno seguinte, nos casos em que o instante final do *setup* em curso que termina mais cedo se encontra a menos de 61 minutos do final do turno atual.

Por fim, aplica-se a diferença temporal resultante do adiamento destes *setups* aos *setups* e trabalhos programados para serem realizados após o instante t.

procedure Sobrecarga_Setup

novos_setups ← setups na solução_setup onde $t_i = t$ e $t_f \geq t$
setups_ativos ← setups na solução_setup onde $t_i \leq t$ e $t_f \geq t$
tempos_folga_máquinas ← calcular tempos de folga para as máquinas com novos_setups
máquina_a_adiar ← máquina com maior folga positiva ou com a mínima folga absoluta
setup_a_adiar ← novo *setup* que pertence à máquina_a_adiar
ti_original ← t_i do setup_a_adiar
tf_primeiro_setup_a_terminar ← valor mínimo de t_f dos setups_ativos
novo_ti ← tf_primeiro_setup_a_terminar + 1
if fim_do_turno_atual - novo_ti < 60:
    novo_ti ← fim_do_turno_atual + 1
end if;
incremento_makespan ← new_ti - original_ti
atualizar setup_a_adiar com novos valores t_i e t_f na solução_setup
atualizar trabalhos e *setups* na solução e solução_setup para máquina_a_adiar com incremento_makespan
return solução, solução_setup

end Sobrecarga_Setup.

Figura 19 - Pseudocódigo da função *Sobrecarga_Setup*

A Figura 20 apresenta o gráfico de *Gantt* do exemplo utilizado nas fases de alocação e construtiva. Considerando que, para todo o horizonte temporal do exemplo, existe apenas uma equipa de *setup* disponível em cada instante, a análise do gráfico de *Gantt* revela um cenário de sobrecarga na utilização de equipas de *setup* a partir do instante $t = 1110$.

Essa sobrecarga é causada pelo início da operação de *setup* na Máquina 3, no minuto 1110, enquanto, simultaneamente, se encontra em processamento a operação de *setup* na Máquina 2, iniciada no instante $t = 1060$, entre os Trabalhos J e I.

Conforme mencionado anteriormente, como o *setup* da Máquina 2 já está em processamento no momento em que ocorre o conflito, o *setup* da Máquina 3 é identificado como novo *setup* e define-se este será o *setup* a ser adiado. Para realizar a prorrogação do *setup* da Máquina 3, que ocorre entre os Trabalhos R e P, estabelece-se que o seu início será uma unidade de tempo após o término do *setup* em processamento na Máquina 2. Como o *setup* da Máquina 2 termina no instante $t = 1130$, o *setup* da Máquina 3 passa a ser iniciado no instante $t = 1131$.

Esta reprogramação garante o cumprimento das restrições de recursos, evitando sobrecargas e assegurando que cada operação de *setup* respeita a disponibilidade limitada da equipa de *setup*.

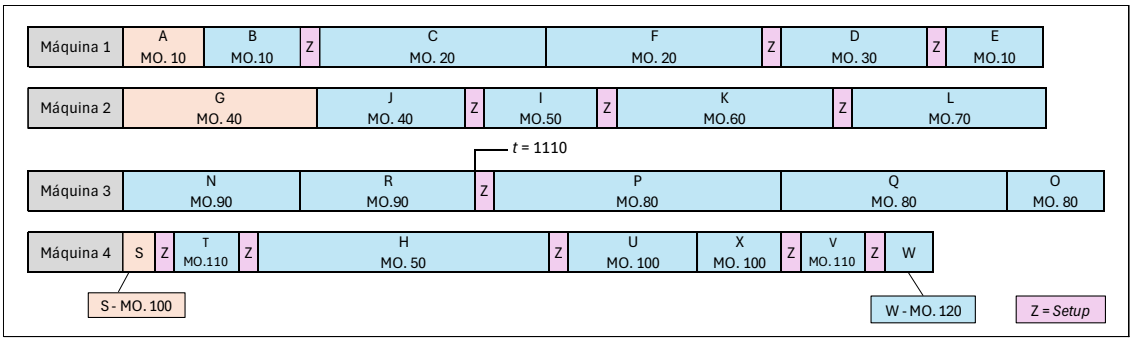


Figura 20 – Gráfico de *Gantt* do exemplo em análise antes da fase de reparação

O procedimento *Sobrecarga_Moldes* é acionado quando o valor *conflitos_moldes* é verdadeiro, ou seja, quando, no instante *t*, existe um molde que está a ser utilizado simultaneamente por mais do que um trabalho, processados em máquinas diferentes. Para assegurar que o molde em sobrecarga é utilizado apenas numa única máquina nesse instante, é necessário analisar o escalonamento nas máquinas em conflito, antes e depois dos trabalhos que partilham o molde de forma conflituosa.

Começa-se por verificar, para cada máquina, se o trabalho anterior ao trabalho em conflito também utiliza o molde em sobrecarga. Se o molde partilhado não estiver a ser utilizado em nenhuma das máquinas em conflito nos trabalhos anteriores aos trabalhos em conflito, aplica-se um processo semelhante ao utilizado nas funções *Sobrecarga_Setup* e *Sobrecarga_Operadores*. Nestes casos, o trabalho em conflito é adiado na máquina que apresenta o valor mais elevado de folga mínima em relação aos trabalhos processados após o instante *t*. Esta abordagem garante que a máquina com maior flexibilidade adia o trabalho em conflito, minimizando o impacto do adiamento na programação e no cumprimento dos prazos de entrega.

Caso contrário, se algum dos trabalhos anteriores aos trabalhos em conflito também utilizar o molde em sobrecarga, define-se que o planeamento dessa máquina se mantém, enquanto os trabalhos das restantes máquinas em conflito são adiados.

Para adiar os trabalhos nas restantes máquinas, dado que não faz sentido interromper a produção "em série" de trabalhos que utilizam o mesmo molde, identifica-se a operação de *setup* que ocorre após o trabalho em conflito na máquina onde o molde será mantido. Em seguida, determina-se que os *setups* que antecedem os trabalhos a adiar nas respetivas máquinas só podem iniciar um minuto após o término do *setup* na máquina onde o molde foi mantido. Esse instante corresponde ao momento em que o molde em conflito fica disponível para ser utilizado noutra máquina.

```
procedure Sobrecarga_Moldes

trabalhos_ativos ← selecionar trabalhos onde 't_i' ≤ t e 't_f' ≥ t
trabalhos_mesmo_molde ← trabalhos_ativos que utilizam o mesmo molde
molde_partilhado ← molde do primeiro trabalho em trabalhos_mesmo_molde
máquinas ← máquinas dos trabalhos que utilizam o molde_partilhado
máquina_A, máquina_B ← máquinas[0], máquinas[1]
trabalho_A, trabalho_B ← trabalho em máquina_A, trabalho em máquina_B
trabalho_ant_A ← último trabalho em máquina_A antes de trabalho_A['t_i']
trabalho_ant_B ← último trabalho em máquina_B antes de trabalho_B['t_i']
if trabalho_ant_A ['Molde'] ≠ molde_partilhado and trabalho_ant_A ['Molde'] ≠ molde_partilhado:
    máquina_com_mais_folga ← máquina cujo trabalho após t_i apresenta o maior 'Tempo_Folga' mínimo
    if máquina_com_mais_folga = máquina_A:
        trabalho_a_adiar, trabalho_a_manter ← trabalho_A, trabalho_B
    else:
        trabalho_a_adiar, trabalho_a_manter ← trabalho_B, trabalho_A
    end if;
else:
    if trabalho_ant_A['Molde'] = ferramenta partilhada:
        trabalho_a_adiar, trabalho_a_manter ← trabalho_B, trabalho_A
    else:
        trabalho_a_adiar, trabalho_a_manter ← trabalho_A, trabalho_B
    end if;
end if;
trabalhos_seguintes ← selecionar trabalhos após trabalho_a_manter['t_f'] em máquina_a_manter
próximo_trabalho ← primeiro trabalho dos trabalhos_seguintes que não usa o molde partilhado
setup_próximo_trabalho ← setup que antecede o próximo_trabalho
setup_a_adiar ← solução_setup onde 'ID_trabalho_seguinte' = trabalho_a_adiar['ID']
novo_ti ← setup_próximo_trabalho['t_f'] + 1
diferença_tempo ← novo_ti - setup_a_adiar ['t_i']
atualizar setup e tempos de trabalho_a_adiar e trabalhos subsequentes em máquina_a_adiar
return solução, solução_setup

end Sobrecarga_Moldes.
```

Figura 21 - Pseudocódigo da função *Sobrecarga_Moldes*

A Figura 22 representa o gráfico de *Gantt* do exemplo em análise após a aplicação da função *Sobrecarga_Setup* que corrige a sobreutilização de equipas de *setup* no instante $t = 1110$, apresentada anteriormente.

Efetivamente, a análise do gráfico de *Gantt* corrigido evidencia a utilização simultânea do Molde MO.50 para o processamento de dois trabalhos em máquinas diferentes a partir do instante $t = 1131$. Consequentemente, aplica-se a função *Sobrecarga_Moldes* para corrigir a sobrecarga na utilização do Molde MO.50.

Conforme mencionado anteriormente, o primeiro passo da função *Sobrecarga_Moldes* consiste em verificar se o molde partilhado é utilizado pelos trabalhos que antecedem os trabalhos em

conflito em alguma das máquinas envolvidas. No presente caso, o Trabalho J, que antecede o Trabalho I na Máquina 2, utiliza o Molde MO.40, e o Trabalho T, que antecede o Trabalho H na Máquina 3, utiliza o Molde MO.110. Assim, conclui-se que nenhum dos trabalhos que antecedem os trabalhos em conflito utiliza o molde partilhado (MO.50).

Desta forma, procede-se ao adiamento do trabalho em conflito na máquina que apresenta a maior folga mínima, conforme o processo descrito, de forma a garantir que o molde MO.50 é utilizado apenas por uma máquina em cada instante.

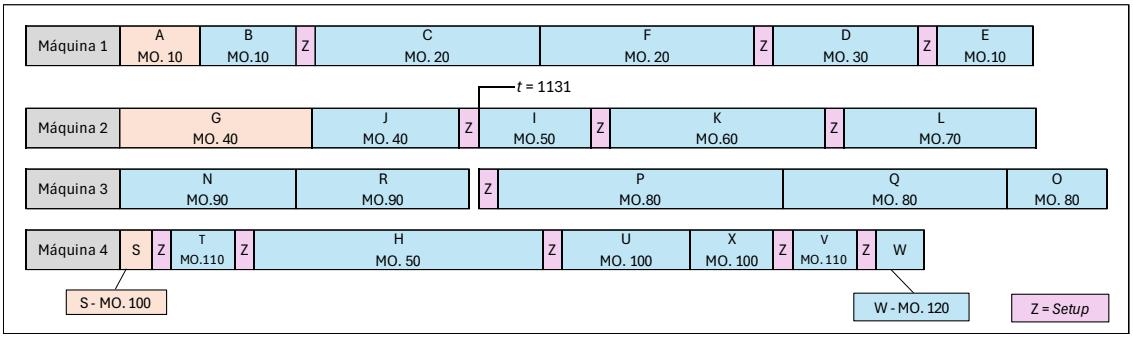


Figura 22 - Gráfico de *Gantt* do exemplo em análise corrigido após *Sobrecarga_Setup* (até $t = 1110$)

Como mencionado anteriormente, a definição de qual trabalho será mantido e qual será adiado baseia-se no maior valor de folga mínima dos trabalhos em cada uma das máquinas em conflito. Analisando as Tabelas 13 e 14, que apresentam as folgas dos trabalhos em conflito e dos trabalhos subsequentes aos mesmos para as Máquinas 2 e 3, verifica-se que o maior valor de folga mínima ocorre na Máquina 2. Assim, fica definido que o Trabalho I será adiado, enquanto o Trabalho H mantém o seu escalonamento.

Para adiar o Trabalho I, prorroga-se a operação de *setup* que o antecede e que insere o Molde MO.50 na Máquina 2, de modo que ela inicie logo após o término do *setup* que sucede o Trabalho H e que remove o mesmo molde da Máquina 3. Por outras palavras, o *setup* que antecede o Trabalho I, que estava previsto para iniciar no instante $t = 1070$, é reprogramado para iniciar no instante $t = 1381$, logo após o molde MO.50 ser liberado pela Máquina 3.

Esta estratégia assegura que o Molde MO.50 é utilizado de forma sequencial entre as máquinas, evitando conflitos e respeitando as restrições de utilização de recursos.

Tabela 13 – Análise das folgas dos trabalhos da Máquina 2

Máquina 2	Data de entrega (Dia)	Data de entrega (Minutos)	t_f	Folga
Trabalho I	2	2880	1470	1410
Trabalho K	2	2880	2200	680
Trabalho L	3	4320	2860	1460
Folga Mínima	680			

Tabela 14 - Análise das folgas dos trabalhos da Máquina 3

Máquina 3	Data de entrega (Dia)	Data de entrega (Minutos)	t_f	Folga
Trabalho H	1	1440	1320	120
Trabalho U	2	2880	1780	1100
Trabalho X	3	4320	2040	2280
Trabalho V	2	2880	2300	580
Trabalho W	3	4320	2510	1810
Folga Mínima	120			

De forma semelhante à função *Sobrecarga_Setup*, a função *Sobrecarga_Operadores* é responsável por reduzir a utilização de operadores no instante t em análise. Neste caso, é admissível interromper o processamento de um trabalho para iniciar outro. Assim, os *trabalhos_ativos* são definidos como os trabalhos que iniciam até ao instante t (inclusive) e que terminam após ou nesse mesmo instante.

O processo é semelhante ao aplicado no adiamento dos *setups*: os trabalhos a adiar são escolhidos com base no maior valor de folga mínima das máquinas às quais pertencem os trabalhos ativos.

Contudo, o facto de ser possível adiar um trabalho já em curso introduz uma ligeira variação ao processo. Nestes casos, o trabalho é interrompido no instante t, e cria-se um novo trabalho, com o mesmo ID, cujo tempo de processamento corresponde ao tempo de processamento restante. Este novo trabalho será iniciado no instante imediatamente após o término do trabalho ativo (excluindo o trabalho a adiar) que termina mais cedo.

Por fim, a diferença entre o instante original de conclusão do trabalho adiado e o novo instante de conclusão é propagada para os *setups* e trabalhos programados após esse trabalho, na mesma máquina. Este ajuste garante que a sequência de produção é atualizada de forma coerente.

procedure Sobrecarga_Operadores

trabalhos_ativos ← trabalhos na solução onde $t_i \leq t$ e $t_f \geq t$
máquinas_ativas ← máquinas únicas às quais pertencem os trabalhos_ativos
máquina_a_adiar ← máquina que pertence às máquinas ativas e tem o maior tempo de folga mínima
trabalho_a_adiar ← trabalho ativo da solução que pertence à máquina_a_adiar
tempo_produção_original ← tempo de produção do trabalho_a_adiar
original_ti ← t_i do trabalho_a_adiar
tf_primeiro_trabalho_a_terminar ← trabalho ativo com o menor valor de t_f (excluindo o trabalho_a_adiar)
novo_ti ← tf_primeiro_trabalho_a_terminar + 1
if trabalho_a_adiar inicia em t:
    atualizar t_i e t_f do trabalho_a_adiar considerando novo_ti e tempo_produção_original
    incremento_makespan ← t_f atualizado - t_f original
else:
    terminar trabalho_a_adiar em $t-1$
    tempo_produção_remanescente ← tempo_produção_original - ($t-1$ - original_ti)
    atualizar t_f e tempo de produção do trabalho_a_adiar
    novo_trabalho ← trabalho com o mesmo ID do trabalho_a_adiar e tempo_produção_remanescente
    adicionar à solução o novo_trabalho com o novo_ti e t_f = novo_ti + tempo_ produção_remanescente
    incremento_makespan ← t_f novo_trabalho - t_f original
end if;
atualizar t_i and t_f dos trabalhos e setups na máquina_a_adiar que começam após t com incremento_makespan
return solução, solução_setup

end Sobrecarga_Operadores.

Figura 23 - Pseudocódigo da função *Sobrecarga_Operadores*

Retomando o exemplo em análise, após a aplicação da correção referente à utilização do mesmo molde no processamento de trabalhos em mais de uma máquina no instante $t = 1131$, obtém-se o gráfico de Gantt representado na Figura 24.

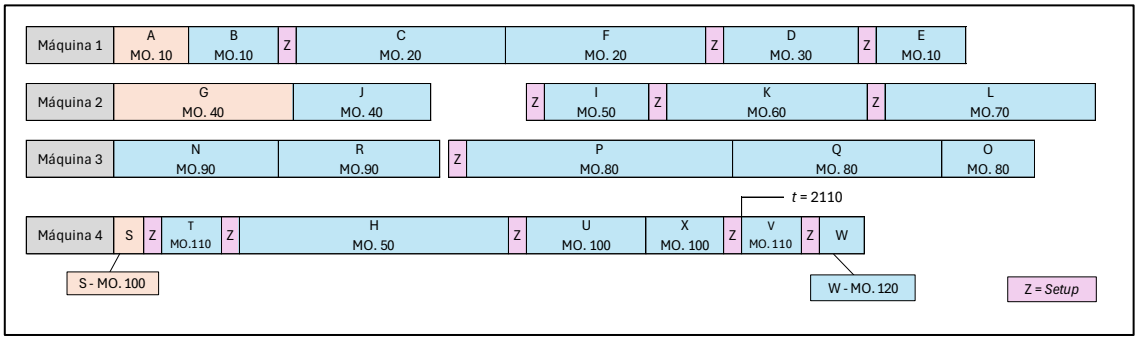


Figura 24 - Gráfico de *Gantt* do exemplo em análise corrigido após *Sobrecarga_Moldes* (até $t = 1131$)

À semelhança do problema apresentado no presente estudo, o exemplo em análise também requer a utilização de recursos adicionais para o processamento dos trabalhos. Nesse sentido, a Tabela 15 apresenta o número de operadores necessários para o processamento de cada trabalho considerado no exemplo. Para o horizonte temporal analisado, há sempre 10 operadores disponíveis para apoiar o processamento dos trabalhos, permitindo que as operações sejam realizadas desde que respeitem este limite de capacidade de operadores.

Tabela 15 – Número de operadores necessários para o processamento de cada trabalho

Trabalho	Nº Operadores
A	2
B	1
C	2
D	2
E	1
F	4
G	2
H	3
I	1
J	1
K	3
L	2
N	2
O	3
P	1
Q	2
R	2
S	2
T	1
U	3
V	5
W	2
X	2

Observando a Tabela 16, que apresenta o número total de operadores utilizados em cada instante do escalonamento, verifica-se que o limite de operadores é respeitado até ao instante $t = 2110$, quando se inicia o Trabalho V. Neste instante, o processamento simultâneo dos Trabalhos D, K, Q e V requer a utilização de 12 operadores, excedendo o limite máximo de 10 operadores disponíveis. De forma a resolver este problema, a função *Sobrecarga_Operadores* é acionada, e os quatro trabalhos em processamento nesse instante são adiados conforme necessário, até que o número total de operadores utilizados não ultrapasse o limite de 10 operadores.

Tabela 16 – Nº Total de operadores em cada instante

Trabalhos que ocorrem em simultâneo	Nº Total Operadores
A - G - N - S	8
A-G - N - T	7
B - G - N - T	6
B - G - N - H	8
B - G - R - H	8
C - J - R - H	8
C - P - H	6
F - P - U	8
F - I - P - U	9
F - I - P - X	8
F - K - P - X	10
D - K - Q - V	12
D - K - Q - W	7
E - L - Q	5
E - L - O	6
L - O	5

De acordo com o procedimento descrito, a decisão sobre qual trabalho adiar na primeira iteração recai sobre o trabalho que pertence à máquina com a maior folga mínima. Analisando a Tabela 17, observa-se que a Máquina 3 é a que apresenta maior folga mínima.

Assim, o Trabalho Q, bem como os trabalhos subsequentes ao mesmo, serão adiados. Para efetuar o adiamento do Trabalho Q, define-se o seu instante inicial como uma unidade de tempo após o valor de t_f mais baixo entre os *trabalhos_ativos* (restantes trabalhos em processamento, no instante em que ocorre a sobrecarga de utilização de operadores – Trabalhos D, K e V). Neste contexto, como a Tabela 17 indica que o Trabalho V é o trabalho ativo que termina mais cedo, o Trabalho Q será reprogramado para iniciar uma unidade de tempo após o instante em que o Trabalho V termina, ou seja, no instante $t = 2511$.

Tabela 17 - Análise das folgas dos trabalhos em processamento no instante $t = 2110$

		Data de entrega (Dia)	Data de entrega (Minutos)	t_f	Folga
Máquina 1	**Trabalho D**	2	2880	2490	390
	Trabalho E	3	4320	2850	1470
	Folga Mínima	390			
Máquina 2	**Trabalho K**	2	2880	2520	360
	Trabalho L	3	4320	3180	1140
	Folga Mínima	360			
Máquina 3	**Trabalho Q**	3	4320	2770	1550
	Trabalho O	3	4320	3070	1250
	Folga Mínima	1250			
Máquina 4	**Trabalho V**	2	2880	2300	580
	Trabalho W	3	4320	2510	1810
	Folga Mínima	580			

Assim, após a primeira iteração da função *Sobrecarga_Operadores*, o número de operadores utilizados no instante $t = 2110$ passa a ser de 10, respeitando o limite de utilização de recursos adicionais imposto pelo problema. Com essa correção, não se processa mais nenhuma iteração na função *Sobrecarga_Operadores* e obtém-se o gráfico de Gantt apresentado na Figura 25, ajustado para garantir a conformidade até ao instante $t = 2110$.

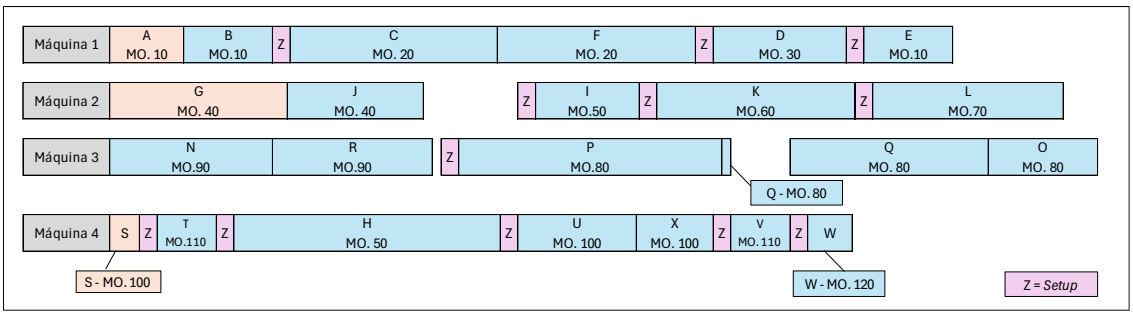


Figura 25 - Gráfico de *Gantt* do exemplo em análise corrigido (até $t = 2110$)

No final da função *Fase_de_Reparação* obtém-se uma solução que assegura o cumprimento das restrições de utilização de operadores, equipas de *setup* e utilização de moldes para todos os instantes do horizonte temporal considerado no escalonamento. A solução totalmente reparada obtida para o exemplo em análise encontra-se apresentada na Figura 26.

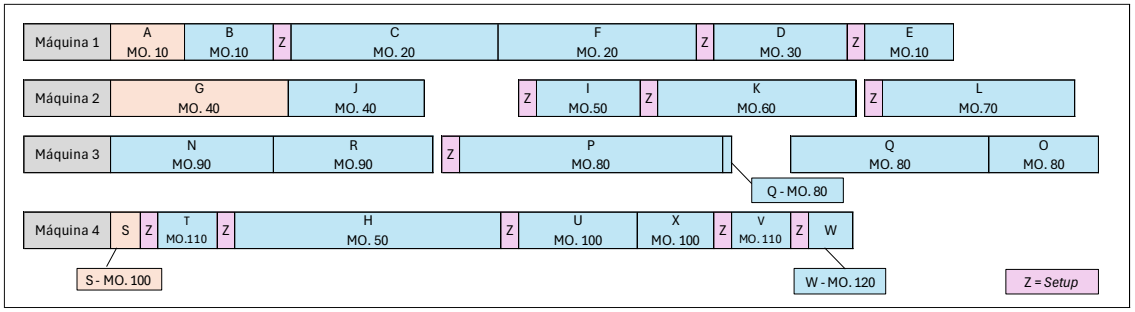


Figura 26 - Gráfico de *Gantt* do exemplo em análise totalmente corrigido

Tal como apresentado no pseudocódigo da Figura 16, após cada aplicação da função *Fase_de_Reparação*, é ativada a função *Outros_Conflitos_Moldes*, que garante que ao longo do horizonte temporal da solução reparada, as operações de *setup* responsáveis por inserir os moldes utilizados em mais do que uma máquina, apenas ocorrem após as operações de *setup* que os retiram da máquina na qual foram utilizados anteriormente.

Note-se que na função *Outros_Conflitos_Moldes* apenas são corrigidos os trabalhos não sobrepostos, uma vez que os casos em que existe sobreposição, são corrigidos na função *Sobrecarga_Moldes*. Adicionalmente, em cada iteração é permitida a correção de, no máximo, um conflito por máquina. Esta restrição assegura que cada máquina não seja sujeita a múltiplas correções simultâneas, prevenindo o surgimento de dependências cíclicas entre operações de *setup*. Além disso, garante que a resolução de um conflito não compromete as correções efetuadas em iterações anteriores, contribuindo para a estabilidade do processo de resolução de conflitos e evitando a reversão de decisões previamente tomadas.

```
procedure Outros_Conflitos_Moldes

moldes_com_múltiplas_máquinas ← filtrar na solução moldes usados em mais de uma máquina
for each molde in moldes_com_múltiplas_máquinas:
    trabalhos_molde ← obter trabalhos que usam o molde, ordenados por ordem crescente de t_i
    for each i in trabalhos_molde:
        trabalho_anterior ← trabalho no índice i
        trabalho_seguinte ← trabalho no índice i + 1
        if trabalho_anterior ['Máquina'] ≠ trabalho_seguinte ['Máquina']
            if trabalho_anterior ['t_f'] < trabalho_seguinte ['t_i']: (apenas trabalhos não sobrepostos)
                setup_trabalho_seguinte ← solução_setup onde 'ID_trabalho_seguinte' = trabalho_seguinte ['ID']
                próximo_setup ← setup após trabalho_anterior que ocorre na mesma Máquina
                if setup_trabalho_seguinte ['t_i'] e próximo_setup ['t_f'] < 61 minutos:
                    novo_ti ← próximo_setup ['t_f'] + 1
                    tempo_adicional ← novo_ti - setup_trabalho_seguinte ['t_i']
                    atualizar trabalhos na máquina onde 't_i' ≥ setup_trabalho_seguinte ['t_i'] com tempo_adicional
                    atualizar setups na máquina onde 't_i' ≥ setup_trabalho_seguinte ['t_i'] com tempo_adicional
                end if;
            end if;
        end if;
    end for;
end for;
return solução, solução_setup

end Outros_Conflitos_Moldes.
```

Figura 27 - Pseudocódigo da função *Outros_Conflitos_Moldes*

No exemplo em análise, não existe nenhum momento em que a aplicação direta da função *Outros_Conflitos_Moldes* seja necessária. No entanto, a Figura 28 apresenta uma versão manipulada do gráfico de *Gantt* original obtido após a função *Fase_de_Reparação* (Figura 26), que permite demonstrar a aplicação dessa função.

Supondo que o escalonamento reparado corresponde ao apresentado na Figura 28, é possível observar que a sequência respeita todas as restrições consideradas na função *Fase_de_Reparação*. No entanto, a utilização do Molde MO.50, usado para o processamento de trabalhos nas Máquinas 2 e 4, apresenta um comportamento fisicamente inviável. Observe-se que o Molde MO.50 é inicialmente utilizado na Máquina 4 para processar o Trabalho H. Após o término do processamento do Trabalho H, o Molde MO.50 é transferido para a Máquina 2, onde é utilizado no processamento do Trabalho I, sem que os dois trabalhos utilizem o molde simultaneamente (respeitando as condições impostas pela função *Sobrecarga_Moldes*).

Contudo, a operação de *setup* que sucede o Trabalho H e que retira o Molde MO.50 da Máquina 4 não ocorre imediatamente após o término do processamento do Trabalho H. Assim, o Molde MO.50 só estará disponível para ser inserido na Máquina 2 para o processamento do Trabalho I após o término dessa operação de *setup* que segue o Trabalho H, e não imediatamente após o fim do processamento do Trabalho H.

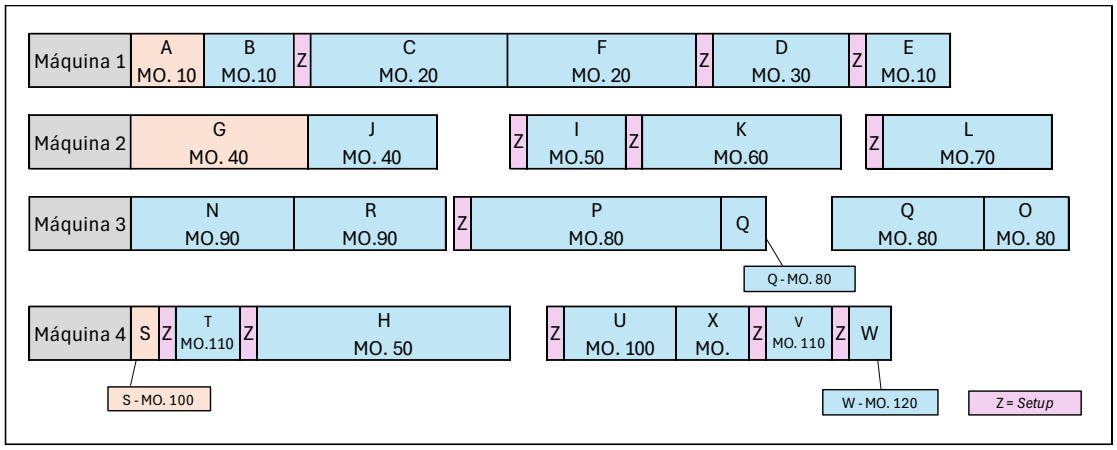


Figura 28 – Exemplo de procedimento realizado pela função *Outros_Conflitos_Moldes*

Assim, a operação de *setup* que antecede o Trabalho I e que insere o Molde MO.50 na Máquina 2 deve ser adiada para iniciar apenas após o término do *setup* que sucede o Trabalho H na Máquina 4, obtendo-se o escalonamento apresentado na Figura 29.

Conforme se observa, após a correção implementada pela função *Outros_Conflitos_Moldes*, algumas das correções anteriormente garantidas pela função *Fase_de_Reparação* são revogadas. Esse fenômeno justifica a aplicação cíclica dessas duas funções durante a etapa de *Fase de Reparação e Outros Conflitos de Moldes*, permitindo que ambas as funções atuem em conjunto para assegurar que o escalonamento final atenda a todas as restrições impostas pelo problema.

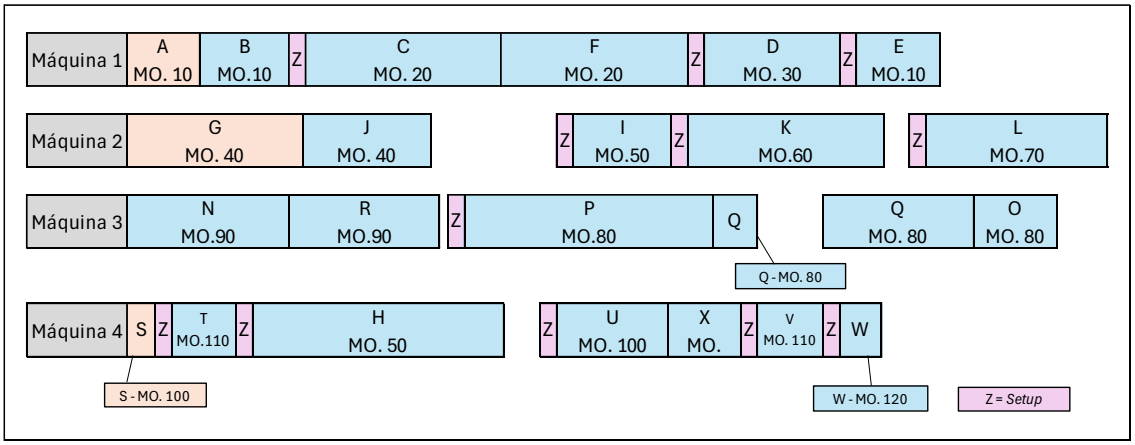


Figura 29 – Gráfico de *Gantt* da solução corrigida após a aplicação da função *Outros_Conflitos_Moldes*

3.4.4 Fase de Pesquisa Local

As soluções iniciais obtidas através da fase construtiva e, posteriormente, reparadas, são consideradas soluções possíveis para o problema em estudo. No entanto, tal como mencionado em (Feo & Resende, 1995) "as soluções geradas pela fase construtiva do algoritmo GRASP, nem sempre representam resultados ótimos", existindo, portanto a possibilidade de serem melhoradas. Neste sentido, a aplicação de técnicas de pesquisa local a cada uma das soluções obtidas pela fase construtiva, garante uma melhoria da qualidade da solução e uma exploração do espaço de soluções mais abrangente, tal como comprovado nos artigos de (Yepes-Borrero et al., 2020) e (Lopez-Esteve et al., 2023).

A fase de pesquisa local apresentada neste estudo é composta por dois momentos distintos, conforme ilustrado na Figura 30. Neste contexto, a melhor solução obtida na Fase de Pesquisa Local 1 é posteriormente refinada no segundo momento, através da aplicação da Fase de Pesquisa Local 2. A principal diferença entre estas duas fases reside no método de criação da *solução_candidata* utilizado em cada uma. Desta forma, cada fase contribui de maneira complementar para melhorar a qualidade da solução, maximizando a eficiência do processo de otimização.

Importa salientar que a *solução_candidata*, independentemente de ser criada com base na função *Inserção_Interna* ou *Troca_Interna*, é sempre gerada de forma dinâmica partir da solução construtiva (não reparada) que apresenta os melhores resultados dentro da mesma iteração do GRASP, encontrada até aquele momento.

```
procedure Fase de Pesquisa Local1 (Fase de Pesquisa Local 2)

for _ in 1 to max_iter do:
    solução_candidata, solução_setup_candidata ← Inserção_Interna (solução, solução_setup) (Fase1)
    solução_candidata, solução_setup_candidata ← Troca_Interna (solução_candidata, solução_setup_candidata)
                                                 (Fase2)
    Fase de Reparação (solução_candidata, solução_setup_candidata )
    if solução_candidata > melhor_solução_atual
        melhor_solução_atual, melhor_solução_setup_atual ← solução_candidata, solução_setup_candidata
        atualizar indicadores de desempenho
    end if;
end for;
return melhor_solução_atual, melhor_solução_setup_atual

end Fase de Pesquisa Local1 (Fase de Pesquisa Local 2).
```

Figura 30 - Pseudocódigo da fase de pesquisa local da metaheurística GRASP

Como mencionado anteriormente, a *Inserção_Interna* (Figura 31) é aplicada na Fase de Pesquisa Local 1 para gerar a *solução_candidata* em cada iteração. Esta função inicia-se com a seleção de um trabalho em atraso em cada máquina onde existam atrasos. Em seguida, para cada máquina identificada com trabalhos em atraso, o trabalho selecionado é retirado da posição onde se

encontra e reinserido na melhor posição disponível na solução dessa mesma máquina, definida com base no coeficiente λ, utilizado na fase construtiva. Este procedimento permite não só que a nova posição escolhida contribua para minimizar os atrasos e aumentar a eficiência do escalonamento, mas também visa corrigir potenciais ineficiências que possam ter sido introduzidas pela randomização do processo na fase construtiva.

Após a reinserção dos trabalhos em atraso em todas as máquinas, a *solução_candidata* é submetida ao mesmo processo de reparação utilizado anteriormente para garantir o cumprimento das restrições. Em seguida, a *solução_candidata* reparada é avaliada. Caso a *solução_candidata* reparada apresente melhores resultados do que a melhor solução encontrada até então, esta é assumida como a nova melhor solução. Nas iterações seguintes, os processos de pesquisa local passam a ser aplicados à fase construtiva dessa nova *solução_candidata*, aprimorando iterativamente a qualidade da solução final.

Assim, a *Inserção_Interna* funciona como uma etapa de refinamento, ajustando a solução inicial para reduzir o impacto de escolhas menos eficientes feitas aleatoriamente, e melhorando a qualidade da solução de forma sistemática.

```
procedure Inserção_Interna

solução_candidata ← cópia da melhor_solução_atual
solução_setup_candidata ← cópia da melhor_solução_setup_atual
máquinas_com_atraso ← máquinas da solução_candidata com atraso
for each máquina in máquinas_com_atraso:
    trabalhos_da_máquina ← trabalhos da solução_candidata que pertencem à máquina
    trabalhos_atrasados ← trabalhos_da_máquina com atraso (exceto trabalhos início de dia)
    trabalho_aleatório ← trabalho aleatório selecionado a partir dos trabalhos_atrasados
    próximo_trabalho ← trabalho dos trabalhos_da_máquina  produzido a seguir a trabalho_aleatório
    anterior_trabalho ← trabalho dos trabalhos_da_máquina  produzido antes do trabalho_aleatório
    solução_candidata ← solução_candidata sem trabalho_aleatório
    ajustar os valores de t_i e t_f da solução_candidata sem trabalho_aleatório
    remover setup para trabalho_aleatório
    criar novo setup entre anterior_trabalho e próximo_trabalho (se necessário)
    ajustar valores de t_i e t_f na solução_setup_candidata
    CL ← Lista de Candidatos construída com o mesmo método utilizado na Fase Construtiva
    posição_escolhida ← Primeiro valor da CL (e não escolhida aleatoriamente uma posição de RCL)
    inserir trabalho_aleatório na posição_escolhida na solução_candidata
end for;
return solução_candidata, solução_setup_candidata

end Inserção_Interna.
```

Figura 31 - Pseudocódigo da função *Inserção_Interna*

No segundo momento, o objetivo é reduzir o número de tempos inativos presentes na solução, que são provocados, essencialmente, pela indisponibilidade de equipas de *setup*. Para isso, aplica-se a função *Troca_Interna*, que, como o próprio nome indica, consiste na troca de dois grupos de trabalhos da mesma família, com o intuito de reorganizar a sequência e dissociar a realização de operações de *setup* na solução.

Esta função começa com a seleção de um subconjunto de 8 máquinas, escolhidas aleatoriamente. Note-se que este número foi determinado após análise de diferentes opções, sendo escolhido com base no valor que apresentou o melhor desempenho nos testes realizados (ou seja, corresponde ao número de máquinas envolvidas na aplicação da *Troca_Interna* que resulta numa maior probabilidade de melhoria da solução). Para cada uma das máquinas selecionadas, caso existam tempos inativos, seleciona-se aleatoriamente um desses períodos de inatividade. Estes momentos de inatividade são identificados quando, num par de trabalhos consecutivos que utilizam ferramentas diferentes, o instante inicial do segundo trabalho é maior do que o instante final do primeiro trabalho somado ao tempo de *setup* entre eles. Em seguida, registam-se os trabalhos que ocorrem antes e após cada período inativo identificado e selecionam-se os grupos de trabalhos consecutivos da mesma família (ou seja, que utilizam o mesmo molde) em relação a esses trabalhos. Finalmente, procede-se à troca desses dois grupos de trabalhos na sequência da solução, avaliando-se a qualidade da nova solução obtida.

Este processo visa reduzir os tempos inativos, otimizando o escalonamento e promovendo uma maior eficiência no uso das equipas de *setup*, o que pode resultar numa solução mais eficaz e menos suscetível a períodos de ociosidade.

procedure Troca_Interna

solução_candidata ← cópia da melhor_solução_atual
solução_setup_candidata ← cópia da melhor_solução_setup_atual
maquinas_selecionadas ← escolha aleatória de 8 máquinas presentes na solução_candidata
for each maquina **in** maquinas_selecionadas:
    tempos_inativos ← [ ]
    solução_reparada_filtrada ← melhor_solução_setup_atual onde 'Máquina' == maquina e 'Dia' ≠ 0
    for each par de trabalhos consecutivos **in** solução_reparada_filtrada:
        if t_i do 2º trabalho > t_f do 1º trabalho + 62 **and** 'Molde' do 1º trabalho ≠ 'Molde' do 2º trabalho:
        adicionar o t_i do 1º trabalho à lista de tempos inativos
        end if;
    end for;
    if tempos_inativos **not empty**:
        tempo_inativo_selecionado ← tempo inativo selecionado aleatoriamente de tempos_inativos
        trabalho_troca1 ← trabalho da solução_candidata com t_i== tempo_inativo_selecionado
        trabalho_troca2 ← trabalho processado a seguir a trabalho_troca1
        grupo1 ← trabalhos consecutivos (sem *setup*) antes de trabalho_troca1
        grupo2 ← trabalhos consecutivos (sem *setup*) após o trabalho_troca2
        anterior_trabalho ← trabalho imediatamente antes do trabalho com t_i mais baixo do groupo1
        seguinte_trabalho ← trabalho imediatamente após o trabalho com t_i mais alto do groupo2
        apagar setup para o trabalho com t_i mais baixo do grupo2
        criar novo setup entre o anterior trabalho e o trabalho com t_i mais baixo do grupo2 (se necessário)
        ajustar valores de t_i e t_f dos trabalhos do grupo2 de acordo com o novo *setup* criado
        criar novo *setup* entre o trabalho com t_i mais alto do grupo2 e trabalho com t_i mais alto do grupo1
        ajustar valores de t_i e t_f dos trabalhos do grupo1 de acordo com o novo *setup* criado
        criar novo setup entre o trabalho com t_i mais alto do grupo1 e o seguinte_trabalho (se necessário)
        ajustar valores de t_i e t_f dos trabalhos e *setups* após o seguinte trabalho
        atualizar solução_candidata e solução_setup_candidata com novos t_i e t_f do grupo1 e grupo2
    end if;
    return solução_candidata, solução_setup_candidata

end Troca_Interna.

Figura 32 - Pseudocódigo da função *Troca_Interna*

Utilizando o exemplo em análise para demonstrar a aplicação da função *Troca_Interna*, através da análise da versão corrigida do gráfico de *Gantt* apresentado na Figura 26, verifica-se que, na Máquina 2, existe um tempo inativo após o término do Trabalho J.

De acordo com o procedimento descrito anteriormente, a função *Troca_Interna* começa por registar os trabalhos processados antes e depois do tempo inativo identificado, neste caso, os Trabalhos I e J. De seguida, selecionam-se os grupos de trabalhos da mesma família que são processados de forma consecutiva com esses trabalhos registados anteriormente. No caso do Trabalho I, não existe nenhum trabalho da sua família processado juntamente com ele, sendo o grupo constituído exclusivamente por si. No caso do Trabalho J, embora o Trabalho G pertença à sua família e seja processado juntamente com o mesmo, este corresponde a um trabalho de início de dia, não podendo ser incluído na troca. O grupo do Trabalho J é também composto exclusivamente pelo Trabalho J.

A etapa seguinte passa por trocar os dois grupos identificados na solução não reparada à qual a *Troca_Interna* é aplicada obtendo-se a *solução_candidata* apresentada na Figura 33.

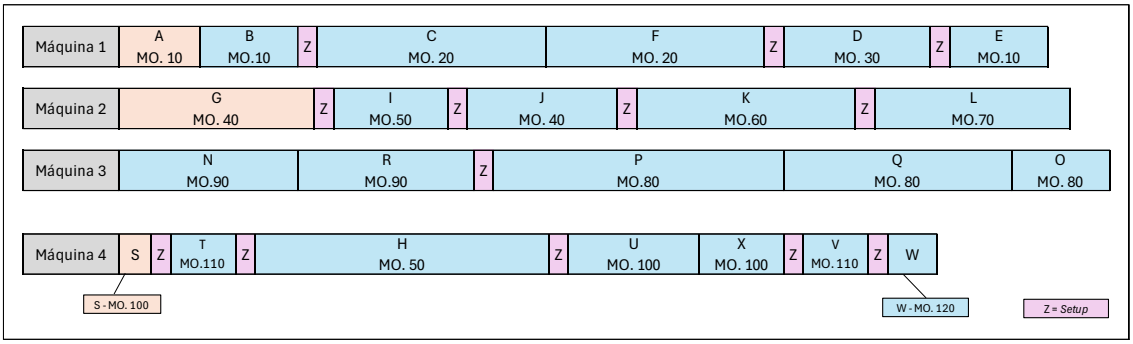


Figura 33 – Exemplo da aplicação da função *Troca_Interna* a um tempo inativo

4 Resultados e Discussão

O Capítulo 4 centra-se na aplicação prática da metaheurística GRASP desenvolvida em instâncias reais e na subsequente avaliação do seu desempenho, relevância e aplicabilidade ao problema em estudo. Este capítulo visa demonstrar como a metodologia proposta responde a cenários reais, analisando os resultados obtidos para avaliar a eficácia do GRASP na resolução das dificuldades específicas apresentadas pelo problema.

A análise dos resultados é realizada em dois momentos distintos: num primeiro momento, procede-se a uma análise crítica detalhada da solução obtida para uma instância específica; num segundo momento, são analisados vários indicadores de desempenho para um conjunto de 9 instâncias. Enquanto o primeiro momento se foca na interpretação dos resultados obtidos e na apresentação da metodologia correta para analisar as soluções devolvidas pelo Sistema Inteligente de Apoio à Decisão (SIAD), o segundo momento avalia a capacidade e a adequação do SIAD desenvolvido para resolver o problema que deu origem ao presente estudo, verificando a sua eficácia e aplicabilidade em múltiplos cenários.

Os resultados foram obtidos através da implementação da metaheurística GRASP em *Python* utilizando o *IDE Microsoft Visual Studio Code 2024, versão 1.93.1*. Os testes computacionais foram executados em um processador Intel Core i7-1165G7 CPU 2.80GHz, com 16.00 GB de memória RAM.

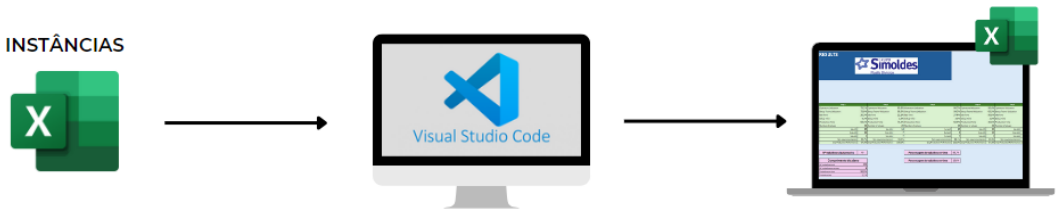


Figura 34 – Arquitetura de dados

4.1 Análise Crítica de Uma Instância Única

Neste subcapítulo, realiza-se uma análise aprofundada de uma instância específica, através de uma avaliação detalhada das soluções obtidas e dos comportamentos do algoritmo GRASP desenvolvido em cada um dos seus momentos. Este estudo detalhado permite observar como o algoritmo responde a esta instância em particular, identificando os ajustes e decisões feitas ao

longo das suas fases, e avaliando a eficácia e os resultados de cada etapa na obtenção de uma solução otimizada para o problema em análise.

A instância selecionada para a análise crítica representa as necessidades dos clientes para um horizonte temporal de 7 dias, com início no dia 04/09/2024. Dado que esta instância segue exatamente os mesmos padrões dos *inputs* utilizados pelo Departamento de Planeamento de Produção, considera-se que esta instância representa com precisão os dados de entrada do problema em estudo. Desta forma, é possível avaliar a aplicabilidade e a eficácia do algoritmo GRASP desenvolvido em condições que refletem o cenário real enfrentado pelo Departamento, assegurando que as soluções obtidas são relevantes e alinhadas com as exigências práticas do sistema de produção.

Adicionalmente, o SIAD é alimentado com as informações que caracterizam o sistema de produção no horizonte temporal em que é aplicado (Tabela 18). Estas informações incluem o número de operadores e equipas de *setup* disponíveis em cada turno dos 7 dias considerados, a duração de cada turno, o tempo mínimo de produção (240 minutos, neste exemplo), o tempo de *setup* (60 minutos) e o stock máximo das referências que não operam em *picking* (Tabela 19).

Tabela 18 - Informações de turno

Dia	Turno	Duração (min)	Equipas Setup	Operadores
1	1	480	2	45
1	2	480	2	45
1	3	480	1	45
2	1	480	2	45
2	2	480	2	45
2	3	480	1	45
3	1	480	2	45
3	2	480	2	45
3	3	480	1	45
4	1	480	2	45
4	2	480	2	45
4	3	480	1	45
5	1	480	2	45
5	2	480	2	45
5	3	480	1	45
6	1	480	2	45
6	2	480	2	45
6	3	480	1	45
7	1	480	2	45
7	2	480	2	45
7	3	480	1	45

Tabela 19 - Stock máximo

PARÂMETROS Stock Máximo		
CARGA MOLDE SEMANA (DIAS)		Stock Máximo (DIAS)
>=	2,5	1,5
>	0,3	3
<	0,3	5

A instância utilizada é composta por 350 trabalhos, que visam satisfazer as necessidades dos clientes no horizonte temporal considerado, e por 27 trabalhos adicionais que representam as ordens de fabrico que transitam, inacabadas, do dia anterior ao primeiro dia de planeamento (03/09/2024). Note-se que, ao longo desta análise, aos trabalhos que transitam inacabados do dia anterior ao primeiro dia em análise é atribuída a data de entrega de dia 0, de forma a evidenciar a sua classificação como um trabalho de início de dia, no entanto, a data de entrega real destes trabalhos é o dia 1.

Tabela 20 - Ordens de fabrico que transitam inacabadas do dia anterior ao dia 04/09/202024

Máquina	Referência	Molde	Quantidade
00K8004	F04016094005A	MO.9202	684
00K8003	F04016098005A	MO.9195	432
00K5009	F04016048001A	MO.9206	336
00K50011	F04016096001A	MO.9199	300
00K50010	F04916006001B	MO.9162	240
00K3509	F04016024001A	MO.9205	968
00K2006	I01822198001A	MO.10592	4020
00K16001	F04016023001A	MO.9209	136
00K10006	F04016050001A	MO.9212	729
00K10001	F04016101002A	MO.9204	128
00E8007	-	-	-
00E8006	F02917004004A	MO.9415	700
00E8005	F02917006003A	MO.9414	456
00E8004	-	-	-
00E6004	I05415162001A	MO.8613	153
00E4005	F02917012002A	MO.9418	320
00E40011	F00418054001A	MO.9289	1761
00E40010	F02821025001A	MO.10423	1080
00E3009	-	-	-
00E3008	I02717004001A	MO.9620	92
00E3007	I04016014002A	MO.9197	1000
00E3006	I05415044001A	MO.8646	720
00E2759	-	-	-
00E27510	F04916069001G	MO.9458	1008
00E17006	F04916019001A	MO.9290	860
00E15007	F04916026002A	MO.9298	80
00E15006	-	-	-
00E15005	-	-	-
00E11007	F02717006002A	MO.9437	210
00E11006	F04016097001A	MO.9200	1008
00E11004	F03315004005A	MO.8540	336
00EN2003	F04916070001B	MO.9457	2520
00K1508	I01617090001A	MO.9692	1400

Conforme demonstrado na Tabela 20, no início do primeiro dia de planeamento, apenas 6 das 33 máquinas que compõem o sistema de produção estão sem produção em curso. Para as máquinas que possuem ordens de fabrico inacabadas, são partilhadas informações adicionais que servem também como *input* para o SIAD, tais como o molde em carga e a quantidade restante para completar as ordens de fabrico em curso. Estas informações são indispensáveis para a definição dos primeiros trabalhos a serem sequenciados na fase construtiva da metaheurística GRASP, permitindo otimizar a sequência de operações com base no estado atual das máquinas e dos moldes.

A Tabela 21 resume as necessidades de produção totais para o horizonte temporal considerado, apresentando os tempos de processamento dos 377 trabalhos que compõe a instância selecionada agrupados por data de entrega e por máquina. Note-se que, nesta Tabela, os tempos de processamento dos trabalhos de início de dia estão incluídos nos tempos de processamento associados à data de entrega de dia 1. De seguida, explica-se, a forma correta de interpretar a Tabela 21.

Observa-se, por exemplo, que a Máquina 00E11004 tem 336 minutos de produção de trabalhos com data de entrega para o dia 1, 512 minutos de produção de trabalhos com data de entrega para o dia 2, 1765 minutos de produção de trabalhos com data de entrega para o dia 3, e assim sucessivamente. As células preenchidas a vermelho indicam um potencial conflito de capacidade, que surge quando o volume de produção de trabalhos alocados à mesma máquina, com a mesma data de entrega, excede a capacidade diária da máquina (1440 minutos). No entanto, esses potenciais conflitos não significam necessariamente que a máquina em questão não conseguirá produzir todos os trabalhos com essa data de entrega, uma vez que, caso exista folga suficiente nos dias anteriores ao dia com sobrecarga, a máquina pode continuar a ter capacidade para completar a produção de todos os trabalhos dentro do prazo estabelecido. Assim, para a realocação de trabalhos, é essencial identificar, para cada máquina, se existe algum dia no horizonte do planeamento em que a soma cumulativa do tempo de produção dos trabalhos com data de entrega igual ou inferior a esse dia excede a capacidade de produção acumulada da máquina até essa data.

Esta organização permite uma visão mais clara da distribuição das cargas de trabalho de cada máquina ao longo do horizonte de planeamento, facilitando a identificação de possíveis sobrecargas em dias específicos e auxiliando no ajuste da alocação de trabalhos de forma a otimizar o cumprimento das datas de entrega estipuladas.

Esta organização permite uma visão mais clara da distribuição das cargas de trabalho de cada máquina ao longo do horizonte de planeamento, facilitando a identificação de possíveis sobrecargas em dias específicos e auxiliando no ajuste da alocação de trabalhos para otimizar o cumprimento das datas de entrega estipuladas.

Para uma realocação eficaz de trabalhos, é essencial identificar, para cada máquina, se existe algum dia no horizonte de planeamento em que a soma cumulativa do tempo de produção dos

trabalhos com data de entrega igual ou inferior a esse dia excede a capacidade de produção acumulada da máquina até essa data.

As situações críticas identificadas exigem, assim, que sejam tomadas decisões que regularizem a utilização da máquina em questão, assegurando que o escalonamento de produção respeita os limites de capacidade e cumpre os prazos de entrega estabelecidos. A seguir, serão demonstradas as estratégias e critérios aplicados para a realocação dos trabalhos, visando otimizar o fluxo de produção e minimizar os conflitos de capacidade.

Tabela 21 - Necessidades de produção organizadas por data de entrega

	1	2	3	4	5	6	7	Total Máquina
00E11004	336	512	1765	710	711	1420	711	6165
00E11006	996	1558	1577	5	1334	1316	1910	8696
00E11007	172	986	935	1444	1466	1444	2903	9350
00E15005	0	212	611	635	635	635	682	3410
00E15006	0	0	561	910	920	910	931	4232
00E15007	155	814	1450	1450	1478	1450	1450	8247
00E17006	717	228	1084	680	990	1033	968	5700
00E27510	412	0	215	956	485	1397	2179	5644
00E2759	1920	240	240	480	0	0	1535	4415
00E3006	300	240	0	240	1440	726	1475	4421
00E3007	350	0	0	0	240	240	240	1070
00E3008	53	0	240	240	862	894	857	3146
00E3009	0	0	0	0	0	0	0	0
00E40010	936	0	330	0	510	552	490	2818
00E40011	1236	165	1289	1709	991	1958	1201	8549
00E4005	614	0	0	0	0	1223	0	1837
00E6004	873	0	540	540	540	803	1020	4316
00E8004	0	0	0	0	0	0	0	0
00E8005	616	0	0	0	191	584	0	1391
00E8006	946	0	0	0	338	973	0	2257
00E8007	240	817	0	720	0	262	2900	4939
00EN2003	504	0	0	0	240	1208	1176	3128
00K10001	96	1122	2088	0	1458	1431	1386	7581
00K10006	608	0	800	10	1067	1157	894	4536
00K1508	584	0	0	0	848	364	2168	3964
00K16001	121	128	1418	540	1239	1370	1130	5946
00K2006	2043	240	480	1908	0	240	852	5763
00K3509	597	799	407	0	272	543	1689	4307
00K50010	2652	0	104	2173	345	345	345	5964
00K50011	245	982	1748	0	1289	1391	1145	6800
00K5009	267	808	1624	0	1600	1583	1572	7454
00K8003	339	99	929	669	961	1083	974	5054
00K8004	1473	1788	1878	720	1554	1740	2353	11506
Total Diário	20401	11738	22313	16739	24004	30275	37136	162606

Começando pela máquina 00E2759, como ilustrado na Tabela 22, a soma dos tempos de produção dos trabalhos com data de entrega para o dia 1 é de 1920 minutos, excedendo o seu limite diário de capacidade de produção (1440 minutos). Dado que nenhum dos trabalhos pode ser produzido em máquinas alternativas, torna-se necessário realocar para uma Máquina Fictícia um número suficiente de trabalhos para garantir que os limites de capacidade sejam respeitados.

É importante observar que, na análise de capacidade das máquinas, além do tempo de produção dos trabalhos, deve-se considerar também o tempo necessário para realizar o número mínimo de *setups* entre os trabalhos alocados à máquina. Esse tempo adicional é fundamental para assegurar uma sequência de produção viável, respeitando as necessidades de *setup* e evitando sobrecargas que possam comprometer o cumprimento das datas de entrega.

A decisão sobre quais trabalhos devem ser mantidos na máquina original e quais devem ser realocados para a Máquina Fictícia baseia-se nas seguintes prioridades:

1. Volume de utilização dos moldes: Dá-se prioridade à realocação de trabalhos que utilizam moldes exclusivos (ou seja, moldes que não são partilhados com outros trabalhos) ou moldes menos utilizados. Esta abordagem visa minimizar os conflitos de uso de moldes e garantir uma melhor gestão dos recursos na máquina original.

2. Tempo de produção: Em segundo lugar, prioriza-se a realocação de trabalhos com maior tempo de produção. Esta estratégia tem como objetivo minimizar o número de trabalhos a serem realocados, o que reduz a quantidade de *setups* necessários e mantém a eficiência operacional na máquina original.

Como estão alocados à máquina trabalhos que utilizam moldes exclusivos, numa primeira fase, procura-se corrigir o excesso de utilização da máquina apenas através da realocação desses trabalhos para a Máquina Fictícia (trabalhos com IDs 1, 2, 3 e 5).

Após a realocação dos trabalhos com os Ids 1, 2 e 3 para a Máquina Fictícia, a soma do tempo de produção dos trabalhos com data de entrega para o dia 1 que permanecem na máquina 00E2759 reduz-se para 1200 minutos. Considerando que a produção dos trabalhos que permanecem alocados na máquina utiliza 3 moldes diferentes, estima-se que sejam necessários, no mínimo, 2 *setups*, o que adiciona um tempo de *setup* total de 120 minutos ao tempo de utilização total da máquina. Assim, o tempo total de utilização da máquina passa a ser de 1320 minutos, indicando claramente que os limites de utilização são cumpridos, não sendo necessário realocar mais trabalhos.

Esta abordagem permite que a máquina 00E2759 opere dentro da sua capacidade, assegurando que todos os trabalhos com data de entrega para o dia 1 possam ser produzidos sem ultrapassar os limites de tempo e recursos estabelecidos.

Tabela 22 – Trabalhos da máquina 00E2759 com data de entrega do dia 1

ID	Máquina Original	Máquina Alt. 1	Máquina Alt. 2	Data Entrega (Dias)	Data Entrega (Minutos)	Molde	Tempo Produção(min)	Tempo Produção Acumulado pré-realocação (min)	Tempo Produção Acumulado pós-realocação (min)
0	00E2759	-	-	1	1440	MO.10958	240	240	0
1	00E2759	-	-	1	1440	MO.10959	240	480	0
2	00E2759	-	-	1	1440	MO.10960	240	720	0
3	00E2759	-	-	1	1440	MO.10962	240	960	240
4	00E2759	-	-	1	1440	MO.10963	240	1200	480
5	00E2759	-	-	1	1440	MO.10962	240	1440	720
6	00E2759	-	-	1	1440	MO.10810	240	1680	960
7	00E2759	-	-	1	1440	MO.10810	240	1920	1200
TOTAL								1920	1200

A Tabela 23 representa a análise dos tempos de produção dos trabalhos com data de entrega de dia 1, alocados à máquina 00K2006. É importante relembrar que a data de entrega de dia 0 serve apenas para indicar que se trata de um trabalho de início de dia, sendo que a sua data de entrega real é o dia 1. Esta identificação é importante, na medida em que os trabalhos de início de dia não podem ser realocados, pois considera-se que o seu processamento já está em curso.

Este exemplo é semelhante ao anterior. Dado que nenhum dos trabalhos pode ser transferido para uma máquina alternativa, a única solução é a realocação de trabalhos para a Máquina Fictícia. O processo de realocação inicia-se pelo trabalho que utiliza um molde de forma exclusiva e possui o maior tempo de produção, com o objetivo de minimizar o número de trabalhos a serem realocados.

Assim, ao realocar o trabalho com o ID 11, verifica-se que a soma do tempo de produção e do tempo mínimo de *setup* ainda excede a capacidade diária da máquina. Este cenário indica que será necessário continuar a realocação de mais trabalhos para a Máquina Fictícia, seguindo o mesmo critério, até que o tempo total de utilização da máquina respeite o limite de capacidade diário.

Em seguida, ao realocar o trabalho com o ID 9, obtém-se um tempo total de produção de 1117 minutos. Somando o tempo mínimo de *setup* de 60 minutos, o total de utilização da máquina passa a ser de 1177 minutos, respeitando assim o limite de capacidade diária da máquina. Desta forma, a alocação fica dentro dos limites de capacidade, garantindo que todos os trabalhos com data de entrega para o dia 1 possam ser processados na máquina sem ultrapassar a capacidade disponível.

Tabela 23 - Trabalhos da máquina 00K2006 com data de entrega do dia 1

ID	Máquina Original	Máquina Alt. 1	Máquina Alt. 2	Data Entrega (Dias)	Data Entrega (Minutos)	Molde	Tempo Produção(min)	Tempo Produção Acumulado pré-realocação (min)	Tempo Produção Acumulado pós-realocação (min)
8	00K2006	-	-	0	1440	MO.10592	637	637	637
9	00K2006	-	-	1	1440	MO.10815	240	877	637
10	00K2006	-	-	1	1440	MO.10592	240	1117	877
11	00K2006	-	-	1	1440	MO.10593	686	1803	877
12	00K2006	-	-	1	1440	MO.10947	240	2043	1117
TOTAL								2043	1117

O cenário dos trabalhos com data de entrega de dia 1 alocados à máquina 00K50010 (Tabela 24) apresenta uma particularidade em relação aos exemplos anteriores, pois inclui dois trabalhos que podem ser produzidos numa máquina alternativa. De acordo com o procedimento descrito no Capítulo 3 - *Heurística para Alocação Trabalho-Máquina*, o processo de realocação, respeitando a hierarquia de máquinas alternativas, prioriza a alocação de trabalhos nas suas Máquinas Alternativas 1. Assim, antes de considerar a realocação de trabalhos para a Máquina Fictícia, verifica-se a possibilidade de alocar esses trabalhos às suas alternativas diretas.

Numa análise inicial, verifica-se que o trabalho com o ID 14, que corresponde ao trabalho com o maior tempo de processamento que utiliza um molde de forma exclusiva, possui um tempo de produção superior à capacidade diária da máquina. Por esse motivo, a máquina alternativa 00K50011 não tem capacidade para alocar este trabalho. De acordo com o procedimento, passa-se para a realocação do trabalho seguinte que possa ser produzido numa Máquina Alternativa 1.

De acordo com o procedimento estabelecido, avança-se para a realocação do próximo trabalho que possa ser produzido numa Máquina Alternativa 1. Assim, ao considerar o trabalho com o ID

15, que possui um tempo de produção de 240 minutos, verifica-se que este pode ser realocado para a sua Máquina Alternativa 1, uma vez que, conforme indicado na Tabela 21, a utilização da máquina 00K50011 para o processamento de trabalhos com data de entrega para o dia 1 representa apenas 245 minutos.

Após esta realocação, a máquina 00K50010 continua com excesso de carga. Como o trabalho com ID 14, que possui o maior tempo de processamento, não pode ser alocado em nenhuma das alternativas disponíveis, é automaticamente realocado para a Máquina Fictícia, sendo, assim, excluído do planeamento na máquina 00K50010.

Tabela 24 - Trabalhos da máquina 00K50010 com data de entrega do dia 1

ID	Máquina Original	Máquina Alt. 1	Máquina Alt. 2	Data Entrega (Dias)	Data Entrega (Minutos)	Molde	Tempo Produção(min)	Tempo Produção Acumulado pré-realocação (min)	Tempo Produção Acumulado pós-realocação (min)
13	00K50010	-	-	0	1440	MO.9162	192	192	192
14	00K50010	00K50011	-	1	1440	MO.10366	2220	2412	192
15	00K50010	00K50011	-	1	1440	MO.10363	240	2652	192
TOTAL							2652		192

Passando para a análise da máquina 00K8004, verifica-se que a soma dos tempos de produção dos trabalhos com data de entrega para o dia 1 é de 1473 minutos, o que indica a necessidade de realocação de alguns trabalhos para respeitar a capacidade diária. Neste contexto, como os tempos de produção dos trabalhos com data de entrega para o dia 1 e que utilizam um molde exclusivo são todos iguais, seleciona-se aleatoriamente um desses trabalhos para realocação na Máquina Fictícia.

Dessa forma, o trabalho com o ID 27 é realocado, reduzindo a soma dos tempos de produção dos trabalhos com data de entrega para o dia 1 para 1233 minutos, conforme apresentado na Tabela 25.

Posteriormente, observa-se que a soma dos tempos de produção dos trabalhos com data de entrega para o dia 2 é de 1788 minutos, o que faz com que a soma cumulativa dos tempos de produção dos dois primeiros dias exceda a capacidade da máquina para esse período. No entanto, agora existem trabalhos que podem ser produzidos na sua Máquina Alternativa 1. Assim, inicia-se a realocação desses trabalhos, priorizando aqueles que utilizam um molde com menor frequência de utilização até ao dia 2. Entre os trabalhos que atendem a esse critério, a prioridade é dada aos que possuem maior tempo de produção.

Após a realocação do trabalho com o ID 32 para a máquina 00K50010, que apresenta folga suficiente na capacidade de produção para os dois primeiros dias, o tempo de produção dos trabalhos com data de entrega para o dia 2 é reduzido para 1260 minutos. Desta forma, a soma do tempo total de processamento dos trabalhos com data de entrega até ao dia 2 passa a ser de 2493 minutos. Ao somar o tempo mínimo de *setup* necessário entre os quatro moldes utilizados por esses trabalhos, obtém-se um tempo total de utilização da máquina de 2673 minutos, assegurando que os limites de capacidade da máquina 00K8004 são respeitados até ao segundo dia.

Por fim, a soma dos tempos de produção dos trabalhos com data de entrega para o dia 3 volta a exceder a capacidade cumulativa da máquina 00K8004 até esse dia. Nesse contexto, procede-se à realocação do trabalho com o maior tempo de produção que utiliza um molde de forma exclusiva (ID 38) para a máquina 00K50010. Essa realocação resolve o excesso de capacidade da máquina 00K8004 até ao dia 3, assegurando que os limites de capacidade de produção nas máquinas envolvidas são respeitados.

Tabela 25 - Trabalhos da máquina 00K8004 com data de entrega do dia 1,2 e 3

ID	Máquina Original	Máquina Alt. 1	Máquina Alt. 2	Data Entrega (Dias)	Data Entrega (Minutos)	Molde	Tempo Produção(min)	Tempo Produção Acumulado pré-realocação (min)	Tempo Produção Acumulado pós-realocação (min)
26	00K8004	-	-	0	1440	MO.9202	513	513	513
27	00K8004	-	-	1	1440	MO.10483	240	753	513
28	00K8004	-	-	1	1440	MO.10484	240	993	753
29	00K8004	-	-	1	1440	MO.10484	240	1233	993
30	00K8004	-	-	1	1440	MO.10637	240	1473	1233
31	00K8004	00K10001	-	2	2880	MO.9202	540	2013	1773
32	00K8004	00K50010	-	2	2880	MO.9162	528	2541	1773
33	00K8004	-	-	2	2880	MO.10483	240	2781	2013
34	00K8004	-	-	2	2880	MO.10484	240	3021	2253
35	00K8004	-	-	2	2880	MO.10484	240	3261	2493
36	00K8004	00K10001	-	3	4320	MO.9202	918	4179	3411
37	00K8004	00K10001	-	3	4320	MO.9202	240	4419	3651
38	00K8004	00K50010	-	3	4320	MO.9162	720	5139	3651
TEMPO PRODUÇÃO TOTAL							5139		3651
TEMPO SETUP MÍNIMO							240		180
TEMPO UTILIZAÇÃO TOTAL							5379		3831

A análise da Tabela 26 revela que, embora a soma dos tempos de produção dos trabalhos com data de entrega para o dia 2 na máquina 00E11006 exceda os 1440 minutos, a soma cumulativa dos tempos de produção dos trabalhos com data de entrega para os dias 1 e 2 é inferior à capacidade da máquina para esses dois dias. Neste sentido, não é necessária nenhuma intervenção até esse ponto.

No entanto, a soma dos tempos de produção dos trabalhos com data de entrega para o dia 3 na máquina 00E11006 volta a ultrapassar o seu limite diário de produção. Neste caso, a soma cumulativa dos tempos de produção dos trabalhos com data de entrega até esse dia (4131 minutos), somada ao tempo mínimo de *setup* necessário para os processar (240 minutos), excede a capacidade total da máquina para esses três dias (4320 minutos).

Tendo em conta este cenário, e respeitando a hierarquia das máquinas alternativas, a primeira opção é realocar os trabalhos que podem ser produzidos numa Máquina Alternativa 1. O processo inicia-se pela verificação da viabilidade de produzir o trabalho que pode ser alocado numa Máquina Alternativa 1, priorizando aquele cujo molde é utilizado o menor número de vezes e que apresenta o maior tempo de produção. Desta forma, o trabalho com o ID 24 é realocado para a máquina 00K8004, uma vez que esta já dispõe de folga suficiente até ao dia 3 para o acomodar, tal como se verifica pelo processo de realocação apresentado anteriormente.

Tabela 26 - Trabalhos da máquina 00E11006 com data de entrega do dia 1,2 e 3

ID	Máquina Original	Máquina Alt. 1	Máquina Alt. 2	Data Entrega (Dias)	Data Entrega (Minutos)	Molde	Tempo Produção(min)	Tempo Produção Acumulado pré-realocação (min)	Tempo Produção Acumulado pós-realocação (min)
16	00E11006	-	-	0	1440	MO.9200	756	756	756
17	00E11006	-	-	1	1440	MO.10485	240	996	996
18	00E11006	00K8004	-	2	2880	MO.9212	82	1078	1078
19	00E11006	-	-	2	2880	MO.9200	756	1834	1834
20	00E11006	-	-	2	2880	MO.10486	240	2074	2074
21	00E11006	-	-	2	2880	MO.10486	240	2314	2314
22	00E11006	-	-	2	2880	MO.10485	240	2554	2554
23	00E11006	-	-	3	4320	MO.10425	240	2794	2794
24	00E11006	00K8004	-	3	4320	MO.9212	428	3222	2794
25	00E11006	-	-	3	4320	MO.9200	909	4131	3703
TEMPO PRODUÇÃO TOTAL							4131		3703
TEMPO SETUP MÍNIMO							240		240
TEMPO UTILIZAÇÃO TOTAL							4371		3943

Por último, ao observar a Tabela 27, que apresenta todos os trabalhos alocados à Máquina 00K8004, verifica-se que esta máquina também enfrenta um excesso de capacidade para a produzir os trabalhos com data de entrega de dia 7.

Para resolver este conflito, realocando alguns dos trabalhos originalmente alocados à Máquina 00K8004 (excluindo o trabalho com o ID 47), começa-se por identificar o molde menos utilizado para a produção de trabalhos com data de entrega até ao dia 7. Embora existam moldes menos utilizados, o Molde MO. 9162 é o menos utilizado entre aqueles que possuem trabalhos que podem ser processados numa Máquina Alternativa 1.

Assim, o próximo passo é selecionar os trabalhos que utilizam o Molde MO. 9162 e que apresentam o maior tempo de processamento. Os trabalhos com IDs 50, 53 e 57 têm o mesmo tempo de processamento, 720 minutos. O trabalho com ID 50 é então selecionado aleatoriamente para ser realocado para a sua Máquina Alternativa 1, a Máquina 00K50010.

Com esta realocação, conforme apresentado na Tabela 27, o limite de utilização da capacidade da máquina 00K8004 passa a ser respeitado, garantindo a conformidade com as restrições de capacidade.

Tabela 27 - Trabalhos da máquina 00K8004 com data de entrega até ao dia 7

ID	Máquina Original	Máquina Alt. 1	Máquina Alt. 2	Data Entrega (Dias)	Data Entrega (Minutos)	Molde	Tempo Produção(min)	Tempo Produção Acumulado pré-realocação (min)	Tempo Produção Acumulado pós-realocação (min)
39	00K8004	-	-	0	1440	MO.9202	513	513	513
40	00K8004	-	-	1	1440	MO.10484	240	753	753
41	00K8004	-	-	1	1440	MO.10484	240	993	993
42	00K8004	-	-	1	1440	MO.10637	240	1233	1233
43	00K8004	00K10001	-	2	2880	MO.9202	540	1773	1773
44	00K8004	-	-	2	2880	MO.10483	240	2013	2013
45	00K8004	-	-	2	2880	MO.10484	240	2253	2253
46	00K8004	-	-	2	2880	MO.10484	240	2493	2493
47	00E11006	00K8004	-	3	4320	MO.9212	428	2921	2921
48	00K8004	00K10001	-	3	4320	MO.9202	918	3839	3839
49	00K8004	00K10001	-	3	4320	MO.9202	240	4079	4079
50	00K8004	00K50010	-	4	5760	MO.9162	720	4799	4079
51	00K8004	00K10001	-	5	7200	MO.9202	594	5393	4673
52	00K8004	00K10001	-	5	7200	MO.9202	240	5633	4913
53	00K8004	00K50010	-	5	7200	MO.9162	720	6353	5633
54	00K8004	00K10001	-	6	8640	MO.9202	540	6893	6173
55	00K8004	00K10001	-	6	8640	MO.9202	240	7133	6413
56	00K8004	00K50010	-	6	8640	MO.9162	240	7373	6653
57	00K8004	00K50010	-	6	8640	MO.9162	720	8093	7373
58	00K8004	00K10001	-	7	10080	MO.9202	513	8606	7886
59	00K8004	00K10001	-	7	10080	MO.9202	240	8846	8126
60	00K8004	00K50010	-	7	10080	MO.9162	640	9486	8766
61	00K8004	00K50010	-	7	10080	MO.9162	240	9726	9006
62	00K8004	-	-	7	10080	MO.10483	240	9966	9246
63	00K8004	-	-	7	10080	MO.10484	240	10206	9486
64	00K8004	-	-	7	10080	MO.10484	240	10446	9726
TEMPO PRODUÇÃO TOTAL							10446		9726
TEMPO SETUP MÍNIMO							240		240
TEMPO UTILIZAÇÃO TOTAL							10686		9966

A Tabela 28 resume as necessidades de produção de cada máquina após as realocações aplicadas. Note-se que, continuam a existir, para algumas das máquinas, dias para os quais a soma do tempo de processamento dos trabalhos com data de entrega desse dia excede a capacidade diária de produção. No entanto, todas as máquinas possuem, agora, capacidade para concluir todos os trabalhos a elas atribuídos antes da respetiva data de entrega, sem considerar os limites de recursos.

Tabela 28 - Necessidades de produção organizadas por data de entrega após otimização da alocação

	1	2	3	4	5	6	7	Total Máquina
00E11004	336	512	1765	710	711	1420	711	6165
00E11006	996	1558	1149	5	1334	1316	1910	8268
00E11007	172	986	935	1444	1466	1444	2903	9350
00E15005	0	212	611	635	635	635	682	3410
00E15006	0	0	561	910	920	910	931	4232
00E15007	155	814	1450	1450	1478	1450	1450	8247
00E17006	717	228	1084	680	990	1033	968	5700
00E27510	412	0	215	956	485	1397	2179	5644
00E2759	1200	240	240	480	0	0	1535	3695
00E3006	300	240	0	240	1440	726	1475	4421
00E3007	350	0	0	0	240	240	240	1070
00E3008	53	0	240	240	862	894	857	3146
00E3009	0	0	0	0	0	0	0	0
00E40010	936	0	330	0	510	552	490	2818
00E40011	1236	165	1289	1709	991	1958	1201	8549
00E4005	614	0	0	0	0	1223	0	1837
00E6004	873	0	540	540	540	803	1020	4316
00E8004	0	0	0	0	0	0	0	0
00E8005	616	0	0	0	191	584	0	1391
00E8006	946	0	0	0	338	973	0	2257
00E8007	240	817	0	720	0	262	2900	4939
00EN2003	504	0	0	0	240	1208	1176	3128
00K10001	96	1122	2088	0	1458	1431	1386	7581
00K10006	608	0	800	10	1067	1157	894	4536
00K1508	584	0	0	0	848	364	2168	3964
00K16001	121	128	1418	540	1239	1370	1130	5946
00K2006	1117	240	480	1908	0	240	852	4837
00K3509	597	799	407	0	272	543	1689	4307
00K50010	192	528	824	2893	345	345	345	5472
00K50011	485	982	1748	0	1289	1391	1145	7040
00K5009	267	808	1624	0	1600	1583	1572	7454
00K8003	339	99	929	669	961	1083	974	5054
00K8004	1233	1260	1586	0	1554	1740	2353	9726
Fictional	4106	0	0	0	0	0	0	4106
Total Diário	20401	11738	22313	16739	24004	30275	37136	162606

No final do processo de realocação, obtêm-se 7 trabalhos que, devido aos limites de capacidade impostos ao problema não podem ser produzidos e, por isso, são alocados na Máquina Fictícia, tal como representa a Tabela 29. Repare-se que todos estes trabalhos tem a data de entrega de dia 1, o que pode ser justificado pelo facto de quanto mais próxima seja a data de entrega, menor flexibilidade existe para ajustar o sistema de produção no sentido de o capacitar para a produção de todas as necessidades existentes.

Tabela 29 – Trabalhos alocados na Máquina Fictícia

ID	Máquina Original	Máquina Alt. 1	Máquina Alt. 2	Data Entrega (Dias)	Data Entrega (Minutos)	Molde	Tempo Produção(min)
F01920008001A	00K2006	-	-	1	1440	MO.10815	240
F03423004001A	00E2759	-	-	1	1440	MO.10958	240
F03423005001A	00E2759	-	-	1	1440	MO.10959	240
F03423006001A	00E2759	-	-	1	1440	MO.10960	240
I01822199001A	00K2006	-	-	1	1440	MO.10593	686
I02021032001A	00K50010	00K50011	-	1	1440	MO.10366	2220
I02621014001A	00K8004	-	-	1	1440	MO.10483	240
TEMPO PRODUÇÃO TOTAL							**4106**

Adicionalmente, a Tabela 29 revela que apenas um dos trabalhos não planeados poderia ser produzido numa máquina alternativa, indicando uma limitação na flexibilidade de alocação

para os trabalhos restantes. A Figura 35 ilustra a distribuição da soma dos tempos de produção dos trabalhos com data de entrega para o dia 1, por máquina, evidenciando a capacidade potencial do sistema de produção para absorver a maior parte dos trabalhos inicialmente alocados à Máquina Fictícia.

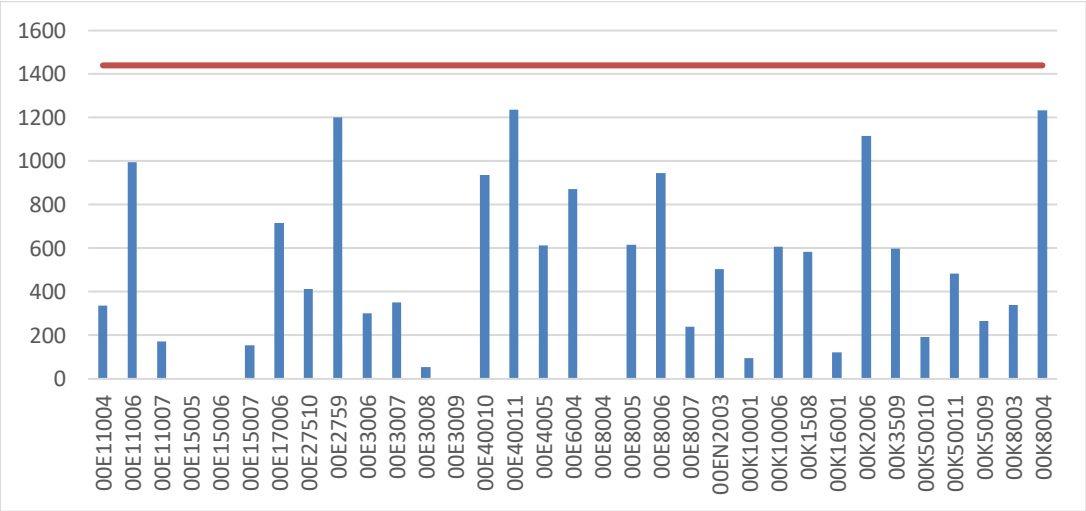


Figura 35 – Soma do tempo de produção dos trabalhos com data de entrega de dia 1 por máquina

A análise detalhada da folga disponível em cada máquina do sistema de produção, apresentada na Tabela 30, acrescenta relevância à existência de uma importante oportunidade para melhorar o planeamento da produção. Os dados indicam que existem folgas consideráveis no sistema que não estão a ser devidamente aproveitadas. À exceção do trabalho não planeado com a referência I02021032001A (cujo tempo de produção excede a capacidade diária do sistema) todos os outros trabalhos não planeados poderiam ser incorporados no planeamento atual.

Especificamente, há 30 máquinas com capacidade para realizar trabalhos com um tempo de produção de 240 minutos (incluindo o tempo de *setup* de 60 minutos), e 25 máquinas que poderiam executar trabalhos com um tempo de produção de 686 minutos (incluindo o tempo de *setup* de 60 minutos). Neste sentido, os resultados sugerem que o planeamento e a alocação de trabalhos poderiam ser significativamente melhorados, se fosse possível distribuir um maior número de trabalhos entre um conjunto mais amplo de máquinas alternativas. Esta alteração não só reduziria a dependência da Máquina Fictícia, mas também maximiza a eficiência do sistema e melhora o cumprimento das datas de entrega estipuladas.

Tabela 30 – Informação detalhada dos trabalhos com data de entrega de dia 1 por máquina

Máquina	Utilização	Capacidade Total	Folga	> 300	> 716
00E11004	336	1440	1104	Sim	Sim
00E11006	996	1440	444	Sim	Não
00E11007	172	1440	1268	Sim	Sim
00E15005	0	1440	1440	Sim	Sim
00E15006	0	1440	1440	Sim	Sim
00E15007	155	1440	1285	Sim	Sim
00E17006	717	1440	723	Sim	Sim
00E27510	412	1440	1028	Sim	Sim
00E2759	1200	1440	240	Não	Não
00E3006	300	1440	1140	Sim	Sim
00E3007	350	1440	1090	Sim	Sim
00E3008	53	1440	1387	Sim	Sim
00E3009	0	1440	1440	Sim	Sim
00E40010	936	1440	504	Sim	Não
00E40011	1236	1440	204	Não	Não
00E4005	614	1440	826	Sim	Sim
00E6004	873	1440	567	Sim	Não
00E8004	0	1440	1440	Sim	Sim
00E8005	616	1440	824	Sim	Sim
00E8006	946	1440	494	Sim	Não
00E8007	240	1440	1200	Sim	Sim
00EN2003	504	1440	936	Sim	Sim
00K10001	96	1440	1344	Sim	Sim
00K10006	608	1440	832	Sim	Sim
00K1508	584	1440	856	Sim	Sim
00K16001	121	1440	1319	Sim	Sim
00K2006	1117	1440	323	Sim	Não
00K3509	597	1440	843	Sim	Sim
00K50010	192	1440	1248	Sim	Sim
00K50011	485	1440	955	Sim	Sim
00K5009	267	1440	1173	Sim	Sim
00K8003	339	1440	1101	Sim	Sim
00K8004	1233	1440	207	Não	Não
TOTAL	16295	47520	31225	30	25

Após o processo de realocação, a metaheurística GRASP, conforme descrito na Subsecção *3.4 – GRASP para UPMSR-PS*, é aplicada com o objetivo de gerar um plano de produção para os 370 trabalhos previstos (350 trabalhos que refletem as necessidades no horizonte temporal considerado, 27 trabalhos de início de dia e excluindo-se os trabalhos alocados na Máquina Fictícia). Deste modo, o algoritmo trabalha para otimizar o planeamento de forma a satisfazer, da melhor forma possível, os indicadores de desempenho definidos na função multiobjetivo do problema, promovendo uma alocação eficiente dos recursos e respeitando as restrições e objetivos estabelecidos para a solução.

A Tabela 31 sumariza os resultados globais obtidos em 1236 segundos (aproximadamente 20 minutos) através da aplicação da metaheurística GRASP à instância de dia 04/09/2024, enquanto a Tabela 32 apresenta os resultados detalhados diários para a mesma instância. Durante o tempo de processamento da metaheurística GRASP, foram geradas 7 soluções construtivas distintas. Note-se que os valores totais apresentados na Tabela 31, consideram os trabalhos não planeados alocados à Máquina Fictícia. Por fim, a Figura 36 representa o gráfico de *Gantt* que representa a solução obtida e o Anexo A apresenta os gráficos de utilização de operadores e equipas de *setup* ao longo do horizonte temporal de planeamento.

Este processo assegura um planeamento robusto, com o mínimo de atrasos e *setups* desnecessários, otimizando a produtividade do sistema de produção.

Tabela 31 – Resultados globais obtidos com a aplicação da metaheurística à instância única

Descrição	Valor
Nº trabalhos não planeados	7
Nº trabalhos planeados	370
Nº trabalhos planeados c/ atraso	6
Número de trabalhos planeados *on-time*	364
% trabalhos planeados c/ atraso	1,6%
% trabalhos planeados *on-time*	98,4%
% TOTAL trabalhos *on-time*	96,6%
% TOTAL trabalhos c/ atraso	3%
Tempo computacional (segundos)	1236

Tabela 32 - Resultados globais obtidos com a aplicação da metaheurística à instância única

Métrica	Dia 1	Dia 2	Dia 3	Dia 4	Dia 5	Dia 6	Dia 7	Total (7 dias)
Utilização Operadores (%)	86,6%	83,5%	73,3%	56,8%	35,5%	23,9%	10,3%	52,8%
Utilização Equipas Setup (%)	61,4%	67,8%	74,1%	50,8%	38,1%	23,3%	14,8%	47,2%
Tempo Ocioso (%)	15,6%	22,4%	30,7%	44,9%	67,5%	78,2%	89,2%	49,8%
Tempo Setup (%)	3,7%	4,0%	4,4%	3,0%	2,3%	1,4%	0,9%	2,8%
Tempo Produção (%)	80,7%	73,6%	64,9%	52,1%	30,3%	20,4%	9,9%	47,4%
Número Setups	29	32	35	24	18	11	7	156
Setups 1º Turno	8	14	14	10	10	5	4	65
Setups 2º Turno	14	12	14	10	4	2	3	59
Setups 3º Turno	7	6	7	4	4	4	0	32
Necessidades Líquidas (min)	0,343	0,247	0,47	0,352	0,505	0,637	0,781	0,48
Performance de Produção Diária	1	0,973	0,94	1	1	1	0,975	0,98

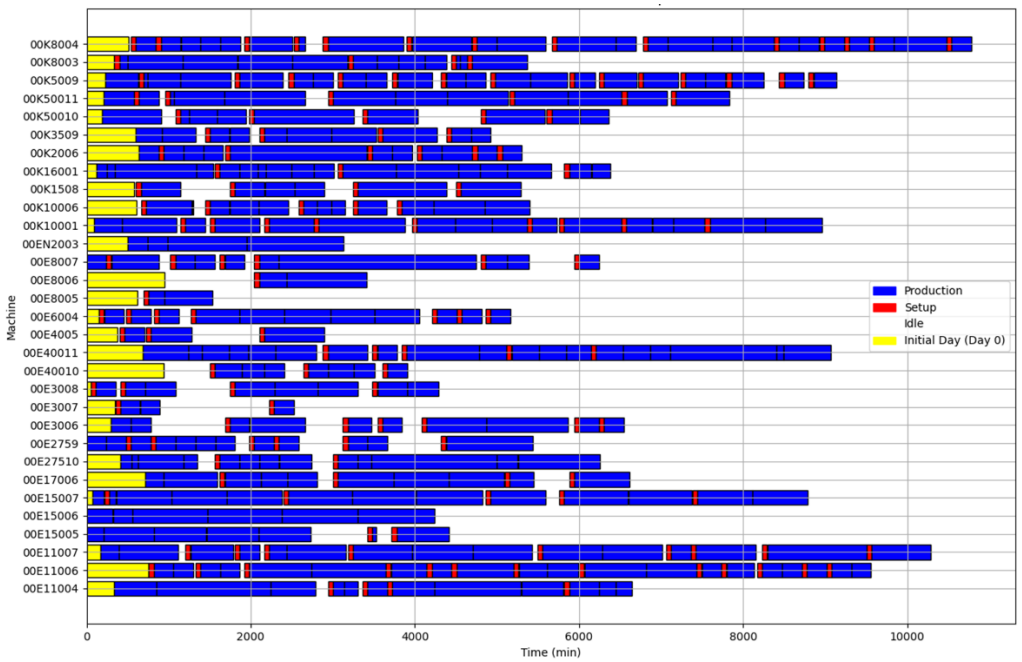


Figura 36 – *Gantt* do planeamento obtido através da metaheurística GRASP para a instância 04/09/2024

Por fim, a Tabela 33 apresenta os trabalhos em atraso obtidos na solução final para a instância do dia 04/08/2024.

Tabela 33 - Trabalhos planeados com atraso

ID	Máquina Original	Máquina Alt. 1	Máquina Alt. 2	Data Entrega (Dias)	Data Entrega (Minutos)	Molde	Tempo Produção(min)	t_i	t_f	Atraso (minutos)
65	00K8004	-	-	3	4320	MO.9202	240	4757	4997	677
66	00K10001	-	-	3	4320	MO.10420	378	7155	7533	3213
67	00E11006	-	-	3	4320	MO.10425	240	9320	9560	5240
68	00E11007	-	-	7	10080	MO.9437	722	9565	10287	207
69	00K8004	-	-	7	10080	MO.9162	640	9837	10477	397
70	00K8004	-	-	2	2880	MO.10483	240	10539	10779	7899

Com o objetivo de analisar o comportamento da metaheurística GRASP ao longo de todo o processo de obtenção de uma solução, a Tabela 34 e os gráficos subsequentes ilustram a evolução dos indicadores de desempenho após cada fase do algoritmo ao longo das 6 iterações realizadas. É importante destacar que, embora o menor número de trabalhos em atraso apareça na Figura 37 após a Fase Construtiva da segunda iteração, essa não representa a melhor solução final, uma vez que as soluções obtidas após as fases construtivas não atendem integralmente às restrições impostas pelo problema.

Os valores mínimos para o número de trabalhos em atraso são obtidos após as fases de Pesquisa Local 1 e Pesquisa Local 2, na terceira iteração. No entanto, a solução obtida após a Fase de Pesquisa Local 2 apresenta o menor número de setups, qualificando-a como a melhor solução encontrada para a instância em análise.

Tabela 34 – Comportamento dos indicadores de desempenho ao longo do GRASP (Instância 1)

Fase	Nº Trabalhos Atrasados	Nº Setups	Makespan	Nº Periodos Ociosos	Total de Tempo Improdutivo
CPhase - iter 1	7	172	10564	0	0
RepPhase - iter 1	27	172	12825	146	36328
LSPhase 1 - iter 1	12	167	11351	138	32719
LSPhase 2 - iter 1	12	150	10865	111	19240
CPhase - iter 2	5	171	10625	0	0
RepPhase - iter 2	11	171	10906	122	21569
LSPhase 1 - iter 2	9	167	11064	125	20903
LSPhase 2 - iter 2	9	163	11064	121	18035
CPhase - iter 3	8	157	10442	0	0
RepPhase - iter 3	11	157	11120	115	15079
LSPhase 1 - iter 3	6	157	10667	108	15007
LSPhase 2 - iter 3	**6**	**154**	**10679**	**108**	**15281**
CPhase - iter 4	7	160	10381	0	0
RepPhase - iter 4	13	160	10604	132	23650
LSPhase 1 - iter 4	10	160	10620	130	24425
LSPhase 2 - iter 4	8	150	10620	112	17642
CPhase - iter 5	7	168	10625	0	0
RepPhase - iter 5	12	168	10969	123	22176
LSPhase 1 - iter 5	10	166	10793	125	23229
LSPhase 2 - iter 5	10	154	10793	114	20126
CPhase - iter 6	9	168	10564	0	0
RepPhase - iter 6	22	168	11584	138	28937
LSPhase 1 - iter 6	14	165	11162	141	29388
LSPhase 2 - iter 6	13	152	10935	112	20161

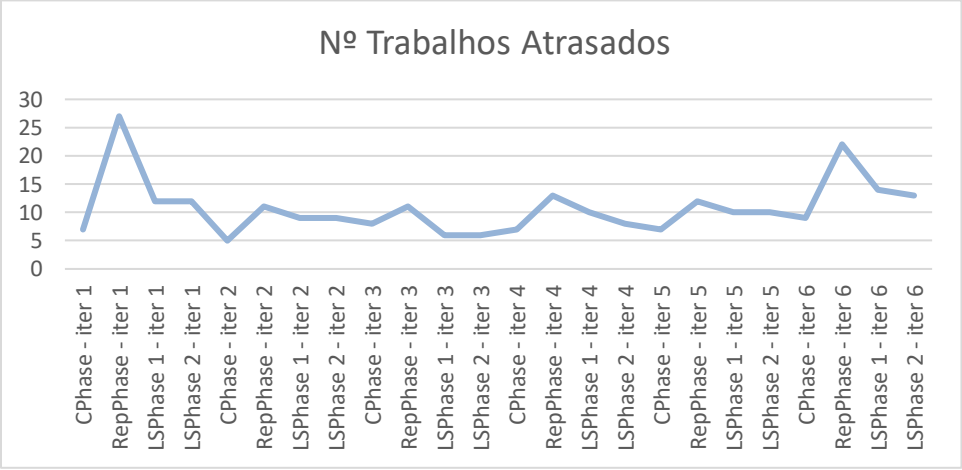


Figura 37 – Comportamento da variação do Nº Trabalhos Atrasados ao longo do GRASP (Instância 1)

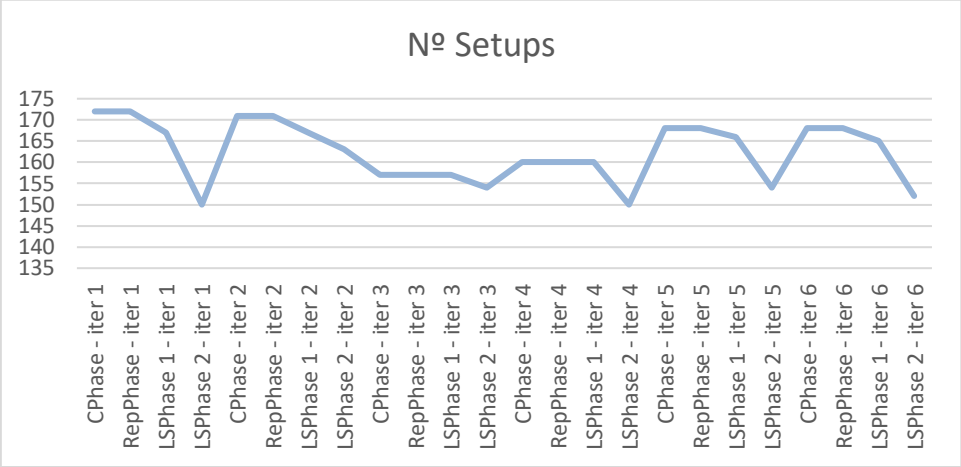


Figura 38 - Comportamento da variação do Nº *Setups* ao longo do GRASP (Instância 1)

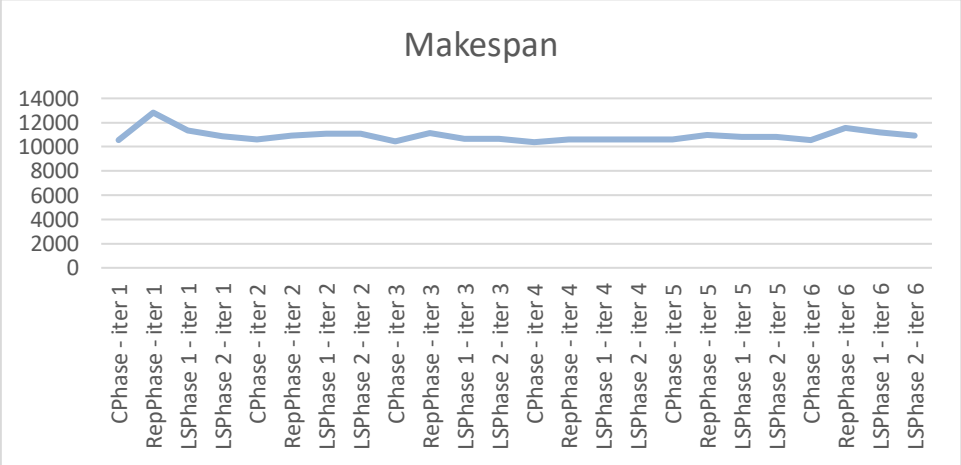


Figura 39 - Comportamento da variação do *Makespan* ao longo do GRASP (Instância 1)

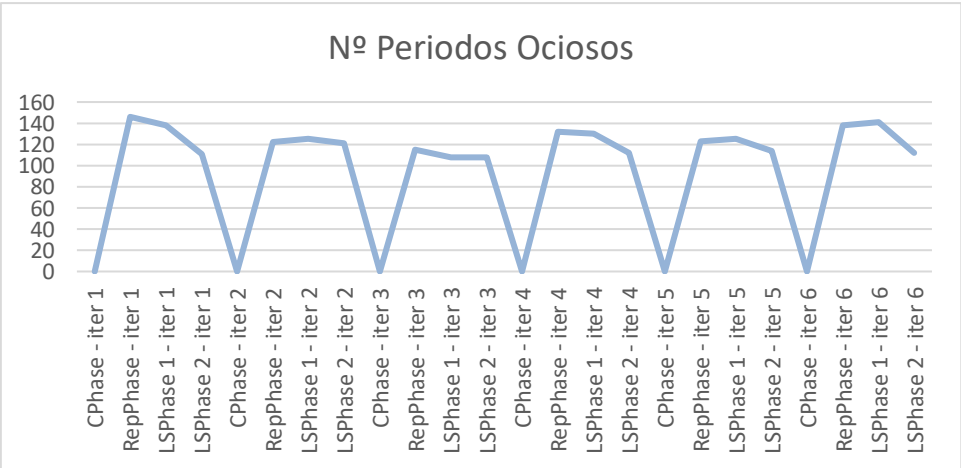


Figura 40 - Comportamento da variação do Nº Tempos Ociosos ao longo do GRASP (Instância 1)

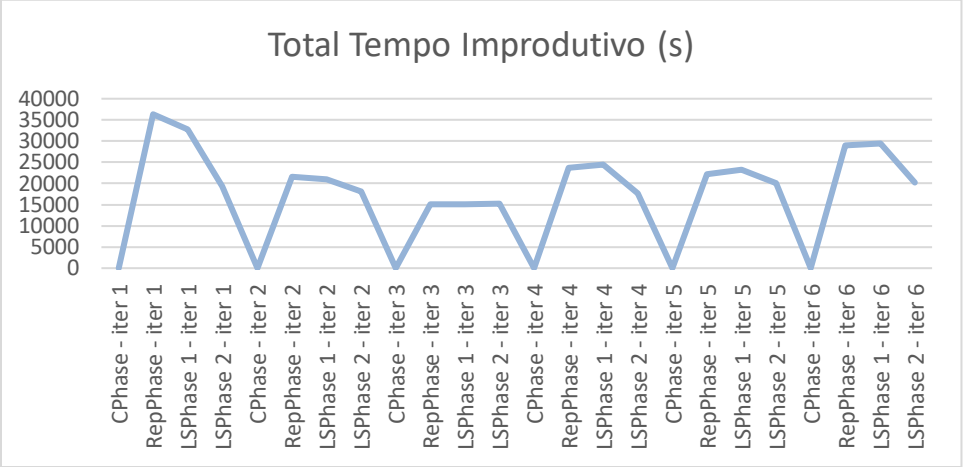


Figura 41 - Comportamento da variação do Total Tempo Improdutivo ao longo do GRASP (Instância 1)

A Tabela 35 e os gráficos subsequentes ilustram a evolução da melhor solução obtida pela metaheurística GRASP ao longo de seu procedimento. O comportamento visualizado nos gráficos reflete um padrão típico da metaheurística GRASP, em que, após as fases de reparação, observa-se uma melhoria significativa nos indicadores de desempenho. Essa redução gradual é fruto da aplicação das fases de pesquisa local, que permitem explorar o espaço de soluções de forma eficaz e identificar as melhores alternativas para cada iteração.

A estabilização do número de trabalhos em atraso nas últimas iterações é um sinal claro de convergência para uma solução robusta, indicando que o GRASP é eficaz em aproximar-se de uma solução próxima do ótimo.

Tabela 35 – Variação da melhor solução encontrada ao longo do GRASP (Instância 1)

Fase	Melhor Valor Número Trabalhos em Atraso	Melhor Valor Número Setups	Melhor Valor Makespan
RepPhase - iter 1	27	172	12825
LSPhase 1 - iter 1	12	167	11351
LSPhase 2 - iter 1	12	150	10865
RepPhase - iter 2	11	171	10906
LSPhase 1 - iter 2	9	167	11064
LSPhase 2 - iter 2	9	163	11064
RepPhase - iter 3	9	163	11064
LSPhase 1 - iter 3	6	157	10667
LSPhase 2 - iter 3	6	154	10679
RepPhase - iter 4	6	154	10679
LSPhase 1 - iter 4	6	154	10679
LSPhase 2 - iter 4	6	154	10679
RepPhase - iter 5	6	154	10679
LSPhase 1 - iter 5	6	154	10679
LSPhase 2 - iter 5	6	154	10679
RepPhase - iter 6	6	154	10679
LSPhase 1 - iter 6	6	154	10679
LSPhase 2 - iter 6	6	154	10679

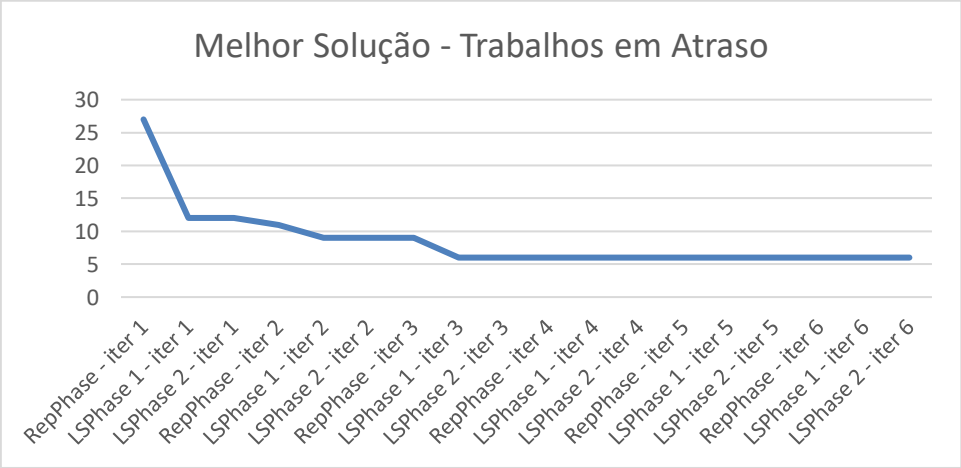


Figura 42 – Comportamento do Nº Trabalhos em Atraso da melhor solução ao longo do GRASP (Instância 1)

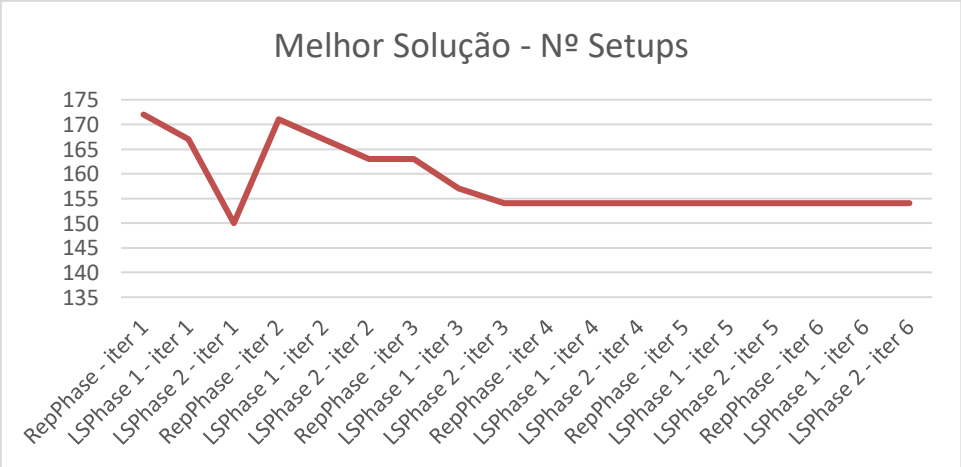


Figura 43 - Comportamento do Nº *Setups* da melhor solução ao longo do GRASP (Instância 1)

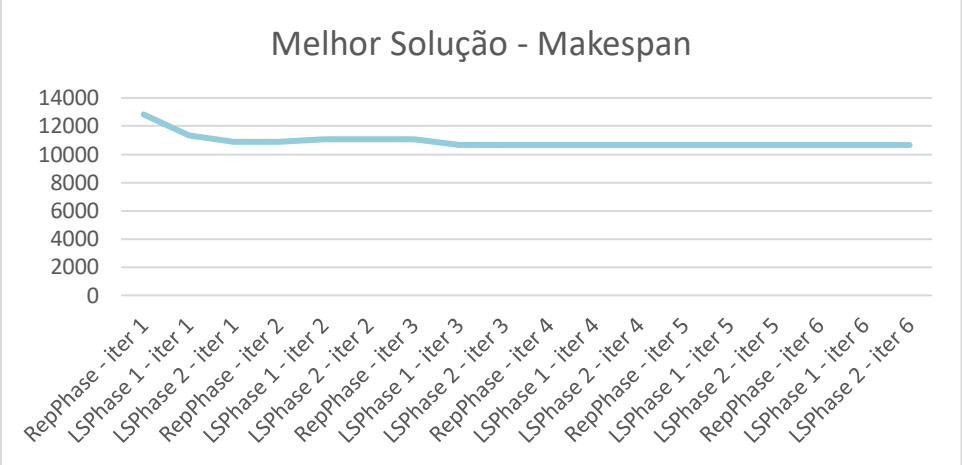


Figura 44 - Comportamento do *Makespan* da melhor solução ao longo do GRASP (Instância 1)

4.2 Avaliação de Desempenho em Múltiplas Instâncias

O objetivo deste subcapítulo é avaliar o desempenho da metaheurística GRASP mediante a análise da qualidade das soluções obtidas com a aplicação do método a 9 instâncias reais. Esta avaliação visa aferir a eficácia do GRASP no cumprimento do principal objetivo do Sistema Inteligente de Apoio à Decisão aplicado ao problema em estudo: a elaboração de um plano de produção que responda, de forma otimizada, às necessidades dos clientes dentro de um horizonte temporal de sete dias, minimizando o número de trabalhos em atraso. A análise dos Tabela 36, permite ter uma visão detalhada do desempenho do Sistema Inteligente de Apoio à Decisão nas nove instâncias testadas.

A heurística construtiva de alocação do SIAD revela-se bastante eficiente, conseguindo incluir, em média, 409 dos 415 trabalhos solicitados pelos clientes para um horizonte de sete dias. Esta inclusão de aproximadamente 98,6% dos trabalhos evidencia a elevada capacidade do sistema em atender à maioria das necessidades dos clientes, garantindo uma resposta robusta e alinhada com os objetivos de produção.

Este número, embora pequeno, revela a existência de limitações no sistema de produção que impedem a alocação de uma parcela mínima dos pedidos, sugerindo oportunidades de melhoria na atribuição de máquinas alternativas a cada trabalho, conforme discutido na subsecção anterior. Efetivamente, ao reforçar a flexibilidade do sistema de produção através de ajustes na alocação de máquinas alternativas, torna-se possível aumentar a capacidade de resposta e minimizar a exclusão de trabalhos do planeamento.

A análise dos resultados do GRASP revela uma performance robusta, com a metaheurística conseguindo, em média, escalonar 393 trabalhos antes da data de entrega, de um total de 409 trabalhos planeados. Desta forma, os valores apresentados, representam, em média, uma taxa de cumprimento de prazos de cerca de 96,2%, reforçando a eficiência do GRASP em priorizar a execução dos trabalhos de modo a reduzir atrasos e assegurar que a maioria dos prazos de entrega é respeitada.

Adicionalmente, ao focar-se no top 5 dos melhores desempenhos do GRASP (o que representa mais de 50% das instâncias analisadas), observa-se uma taxa de cumprimento ainda mais elevada, com 97,2% dos trabalhos produzidos dentro do prazo. Uma vez mais, estes resultados evidenciam o elevado potencial do GRASP em otimizar o planeamento de produção, garantindo que, mesmo em cenários com restrições de capacidade exigentes, a maioria dos trabalhos é concluída de forma pontual, contribuindo para um planeamento eficiente e fiável.

De forma global, ao analisar a combinação dos resultados da fase de alocação com o desempenho da metaheurística GRASP, conclui-se que o Sistema Inteligente de Apoio à Decisão (SIAD) desenvolvido neste estudo é altamente eficaz, demonstrando a capacidade de gerar um planeamento que permite, em média, a produção de 94,8% dos trabalhos dentro do prazo de entrega estipulado.

Em termos de tempo computacional, o SIAD demonstra uma performance eficiente, com o tempo de processamento a variar entre 18 e 24 minutos para gerar o plano de produção semanal. A média de 22 minutos por instância confirma a viabilidade do sistema para aplicações em ambientes de produção real, garantindo uma resposta rápida e uma integração fluida no processo de planeamento.

Assim, a Tabela 36 ilustra a capacidade do SIAD em planear de forma eficiente e com elevada precisão, atendendo à maioria dos pedidos dos clientes dentro do prazo e com tempos de processamento adequados, reforçando a adequação do sistema para o problema em análise.

Tabela 36 – Resumo resultados GRASP

Descrição	I1	I2	I3	I4	I5	I6	I7	I8	I9	Média
Nº trabalhos total	377	363	390	398	428	453	458	435	435	415
Nº trabalhos excluídos do planeamento	7	7	6	7	4	7	4	5	5	6
Nº trabalhos incluídos do planeamento	370	356	384	391	424	446	454	430	430	409
% trabalhos incluídos no planeamento	98,1%	98,1%	98,5%	98,2%	99,1%	98,5%	99,1%	98,9%	98,9%	98,6%
Nº trabalhos planeados c/ atraso	6	7	9	8	15	26	24	11	12	13
Número de trabalhos planeados *on-time*	364	349	375	383	409	420	430	419	418	396
% trabalhos planeados c/ atraso	1,6%	2,0%	2,3%	2,0%	3,5%	5,8%	5,3%	2,6%	2,8%	3,1%
% trabalhos planeados *on-time*	98,4%	98,0%	97,7%	98,0%	96,5%	94,2%	94,7%	97,4%	97,2%	96,9%
% TOTAL trabalhos *on-time*	96,6%	96,1%	96,2%	96,2%	95,6%	92,7%	93,9%	96,3%	96,1%	95,5%
% TOTAL trabalhos c/ atraso	3,4%	3,9%	3,8%	3,8%	4,5%	7,3%	6,1%	3,7%	4,0%	4,5%
Tempo computacional (segundos)	1236	1337	1262	1180	1325	1420	1292	1323	1352	1303,0
Tempo computacional (minutos)	21	22	21	20	22	24	22	22	23	22

Finalmente, à semelhança das análises apresentadas no subcapítulo anterior, os gráficos no Anexo B ilustram o comportamento dos diversos indicadores de desempenho ao longo do

processo do GRASP, para as 9 instâncias. Adicionalmente, no Anexo C, encontra-se representada a trajetória da melhor solução ao longo do procedimento para as mesmas 9 instâncias.

5 Conclusão

Para manter a competitividade num mercado cada vez mais exigente e dinâmico, as empresas devem ser flexíveis e adaptáveis, assegurando simultaneamente a tomada de decisões informadas e fundamentadas, que garantam a eficácia operacional e o alcance de objetivos de longo prazo. No atual cenário de inovação tecnológica acelerada, torna-se essencial que as empresas invistam continuamente em métodos que lhes permitam desenvolver soluções tecnológicas capazes de promover decisões rápidas e fundamentadas, preparando-as para responder aos desafios e oportunidades emergentes no mercado.

Neste contexto, o presente projeto surge como resposta ao desafio da INPLAS de identificar oportunidades de melhoria no sistema de apoio à decisão (SIAD) atualmente implementado, com o objetivo de aumentar a sua adaptabilidade às necessidades específicas da INPLAS e ao seu sistema de produção. Deste modo, após um período de análise e mapeamento detalhado do processo de planeamento, incluindo as interações a montante e a jusante deste, identificou-se que a consideração de máquinas alternativas teria um impacto significativo e positivo no desempenho do SIAD. Com base nessa conclusão, foi desenvolvido um sistema de apoio à decisão para a programação de máquinas paralelas dedicadas com múltiplas alternativas, *setups*, recursos adicionais e datas de entrega, conforme descrito no presente estudo.

A resolução deste problema levanta duas principais questões para investigação. A primeira questão prende-se com a análise de como a inclusão do subproblema de alocação dos trabalhos a uma das suas máquinas alternativas pode melhorar o Sistema Inteligente de Apoio à Decisão atualmente em vigor, aumentando a sua adaptabilidade às necessidades específicas do departamento de planeamento da produção da INPLAS. A segunda questão centra-se na avaliação da adequabilidade da aplicação da metaheurística GRASP a um problema de escalonamento em máquinas paralelas dedicadas com múltiplas alternativas, *setups*, recursos adicionais e datas de entrega, estudando o impacto desta abordagem na eficiência e na qualidade das soluções obtidas.

O presente trabalho apresenta várias contribuições quer ao nível da heurística construtiva desenvolvida para o subproblema de alocação, quer para a metaheurística GRASP desenvolvida para a resolução deste tipo de problemas.

Em primeiro lugar, dado que não foram encontradas abordagens na literatura para a resolução de problemas específicos de escalonamento em máquinas paralelas dedicadas com múltiplas alternativas, a heurística construtiva de alocação foi desenvolvida de forma original no âmbito deste estudo. Adicionalmente, a particularidade do problema em relação à hierarquia das máquinas alternativas acrescenta uma camada extra de originalidade à heurística desenvolvida. Assim, considera-se que a heurística, na sua totalidade, representa uma contribuição significativa, ao propor uma abordagem inovadora e adaptada para lidar com as complexidades específicas deste tipo de problema, colmatando uma lacuna existente na literatura sobre o tema.

No que diz respeito ao GRASP identificam-se cinco contribuições. A primeira contribuição refere-se à adaptação dos modelos de referência para o caso específico das máquinas paralelas dedicadas com múltiplas alternativas. Esta adaptação, juntamente com a particularidade do problema dos tempos de processamento dos trabalhos e dos *setups* serem independentes tanto da máquina em que são executados quanto da sequência de processamento, permite separar o processo de alocação do processo de sequenciação, ao contrário do que acontece nos modelos de referência, onde ambos são realizados de forma simultânea. Importa salientar que, até ao momento, não foi encontrada na literatura nenhuma aplicação prática deste tipo de abordagem em problemas que envolvem máquinas paralelas dedicadas com múltiplas alternativas, reforçando a originalidade e a relevância desta contribuição. Adicionalmente, a inclusão de datas de entrega comuns, para além de adicionar complexidade ao problema, cria a necessidade de introdução de uma nova variável (o atraso de cada trabalho) e exige a adaptação da função objetivo, que passa a ter como objetivo principal a minimização do número de trabalhos com atraso. De seguida, sendo o sistema de produção considerado, um sistema de laboração contínua, tornou-se necessário adaptar o GRASP a esta particularidade. Deste modo, na fase de sequenciação, a heurística construtiva desenvolvida não inicia com uma solução parcial vazia (à semelhança dos modelos de referência considerados), mas sim, com os trabalhos que transitam inacabados do dia anterior, sequenciados na primeira posição de cada máquina. Este aspeto reduz a flexibilidade da fase de sequenciação e, por isso, adiciona também complexidade ao problema. A quarta adaptação consiste na adaptação da heurística construtiva de sequenciação desenvolvida, de forma original, num dos modelos de referência. A singularidade desta heurística construtiva de sequenciação prende-se com a exclusividade do coeficiente λ utilizado para a classificação das posições nas quais cada trabalho pode ser inserido nas soluções parciais. Conforme mencionado, ao considerar datas de entrega, o objetivo principal do problema deixa de ser a minimização do *makespan* e passa a focar-se na minimização do número de trabalhos com atraso. No entanto, verificou-se que a exclusividade deste objetivo resulta num elevado número de soluções ótimas, limitando a capacidade de discriminação entre as alternativas viáveis. Assim, desenvolveu-se uma função multiobjetivo que, além de minimizar o número de trabalhos com atraso, também procura uma solução eficiente e compacta, reduzindo o número de *setups* e o *makespan* total.

Os resultados obtidos permitem responder de forma clara à primeira questão de investigação, evidenciando a relevância e o impacto positivo da realocação de trabalhos para as Máquinas Alternativas 1 e para a Máquina Fictícia. Na análise da instância específica, foram realocados um total de 12 trabalhos: 7 para a Máquina Fictícia (excluídos do planeamento) e 5 para as respetivas Máquinas Alternativas 1. Sendo a estratégia de realocação implementada apenas quando as máquinas não conseguem cumprir os prazos de entrega dos trabalhos que lhes estão alocados, conclui-se que, caso estas realocações não fossem efetuadas, o desempenho do algoritmo GRASP sofreria uma deterioração substancial, resultando num aumento estimado de, no mínimo, 200% no número de trabalhos em atraso (de 6 para 18 trabalhos).

Adicionalmente, verificou-se que o sistema de produção poderia, potencialmente, acomodar 6 dos trabalhos inicialmente alocados à Máquina Fictícia, assegurando o seu processamento dentro dos prazos estabelecidos. Este facto revela uma oportunidade clara de melhoria no

sistema, sugerindo que a inclusão de um maior número de máquinas alternativas para cada referência poderia aumentar significativamente a flexibilidade e a capacidade de resposta do sistema de produção, contribuindo para uma redução ainda mais acentuada no número de trabalhos em atraso.

Relativamente à segunda questão de investigação, a análise dos resultados obtidos com o GRASP evidencia uma performance robusta na aplicação desta metaheurística a problemas de programação de máquinas paralelas dedicadas com múltiplas alternativas, *setups*, recursos adicionais e datas de entrega. Em média, o algoritmo demonstrou a capacidade de escalonar 96,9% dos trabalhos planeados dentro do prazo estipulado, o que reforça a eficácia do GRASP em priorizar a execução dos trabalhos, minimizando atrasos e rentabilizando a utilização dos recursos adicionais para assegurar o cumprimento dos prazos de entrega. Adicionalmente, ao focar-se no top 5 dos melhores desempenhos do GRASP (representando mais de 50% das instâncias analisadas), observa-se uma taxa de cumprimento de prazos ainda mais elevada, com 98% dos trabalhos concluídos dentro do prazo. Estes resultados reforçam o elevado potencial do GRASP para otimizar a tipologia de problemas em estudo, mostrando que, mesmo em cenários com restrições rigorosas, a maioria dos trabalhos é concluída dentro do seu prazo estipulado, o que contribui para uma capacidade de planeamento ágil, eficiente e fiável.

Em termos de tempo computacional, o Sistema Inteligente de Apoio à Decisão (SIAD) apresenta uma performance eficiente, com o tempo de processamento a variar entre 21 e 24 minutos para gerar o plano de produção semanal. Com uma média de 22 minutos por instância, confirma-se a viabilidade do sistema para aplicações em ambientes de produção real, proporcionando uma resposta rápida e uma integração fluida no processo de planeamento.

Assim, de forma global, ao avaliar a combinação dos resultados da fase de alocação com o desempenho da metaheurística GRASP, conclui-se que o SIAD desenvolvido neste estudo se revela altamente eficaz, demonstrando a capacidade de gerar um planeamento que permite, em média, a produção de 95,5% do número total de trabalhos dentro do prazo de entrega estipulado. Estes resultados reforçam a pertinência e a robustez do modelo proposto, apresentando-se como uma ferramenta fiável e eficiente para o suporte ao planeamento em ambientes produtivos com características semelhantes.

5.1 Limitações e Investigação Futura

Os trabalhos futuros poderão centrar-se na melhoria do desempenho do SIAD, abordando as diversas vertentes que o compõem. Primeiramente, existe margem para aperfeiçoamento na fase de alocação, uma vez que ao substituir a abordagem heurística por modelos matemáticos exatos, torna-se possível a obtenção resultados ótimos num menor tempo computacional.

Relativamente à metaheurística GRASP desenvolvida, é recomendável explorar outras estratégias de pesquisa local, que permitam expandir o espaço de pesquisa de soluções e, simultaneamente, acelerar a convergência para a melhor solução em cada conjunto de soluções. Para além disso, será fundamental aplicar exaustivamente o Sistema Inteligente de Apoio à Decisão a várias instâncias reais e adaptá-lo às possíveis particularidades de cada uma, aumentando assim a sua robustez e capacidade de resolução de qualquer instância que surja no departamento de planeamento de produção.

Existe ainda espaço para melhorias na programação em *Python* desenvolvida, com a possibilidade de reduzir substancialmente o tempo computacional da metaheurística.

Este projeto constituiu um verdadeiro desafio a nível pessoal, académico e técnico. A complexidade associada ao problema em estudo exigiu um esforço significativo para compreender profundamente as suas características e particularidades, enquadrá-lo na literatura, traduzir todos os fatores influentes num algoritmo e desenvolver computacionalmente as ideias que foram surgindo. Este processo revelou-se moroso e exigente. Contudo, o percurso, embora repleto de obstáculos, proporcionou momentos de aprendizagem contínua, e as dificuldades superadas permitiram-me desenvolver competências avançadas em otimização combinatória e programação heurística, assim como uma maior resiliência e capacidade de resolução de problemas complexos.

Referências

Akyol Ozer, E., & 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

Allahverdi, A. (2015). The third comprehensive survey on scheduling problems with setup times/costs. *European Journal of Operational Research*, *246*(2), 345–378. https://doi.org/10.1016/J.EJOR.2015.04.004

Allahverdi, A., Gupta, J. N. D., & Aldowaisan, T. (1999). A review of scheduling research involving setup considerations. *The International Journal of Management Science*, 219–239.

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

Alvarez-Valdes, R., Crespo, E., Tamarit, J. M., & Villa, F. (2008). GRASP and path relinking for project scheduling under partially renewable resources. *European Journal of Operational Research*, *189*(3), 1153–1170. https://doi.org/10.1016/j.ejor.2006.06.073

Anticona, M. T. (2006). A GRASP algorithm to solve the problem of dependent tasks scheduling in different machines. In M. Bramer (Ed.), *Artificial Intelligence in Theory and Practice* (Vol. 217, pp. 325–334). Boston: Springer.

Arnaout, J. P., Musa, R., & Rabadi, G. (2014). A two-stage Ant Colony optimization algorithm to minimize the makespan on unrelated parallel machines - Part II: Enhancements and experimentations. *Journal of Intelligent Manufacturing*, *25*(1), 43–53. https://doi.org/10.1007/s10845-012-0672-3

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. *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 and Operations Research*, *103*, 46–63. https://doi.org/10.1016/j.cor.2018.10.010

Blazewicz, J., Brauner, N., & Finke, G. (2004). Scheduling with Discrete Resource Constraints. *Handbook of Scheduling: Algorithms, Models, and Performance Analysis*, 508–525.

Błażewicz, J., Ecker, K. H., Pesch, E., Schmidt, G., & Węglarz, J. (2001). Scheduling Computer and Manufacturing Processes. In *Scheduling Computer and Manufacturing Processes*. Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-662-04363-9

Blazewicz, J., Ecker, K., Pesch, E., Schmidt, G., Sterna, M., & Weglaz, J. (2007). *International Handbooks on Information Systems Handbook on Scheduling From Theory to Practice Second Edition*. http://www.springer.com/series/3795

Cheng, T. C. E., & Sin, C. C. S. (1990). A state-of-the-art review of parallel-machine scheduling research. In *European Journal of Operational Research* (Vol. 47).

Diana, R. O. M., de França Filho, M. F., de Souza, S. R., & de Almeida Vitor, J. F. (2015). An immune-inspired algorithm for an unrelated parallel machines' scheduling problem with sequence and machine dependent setup-times for makespan minimisation. *Neurocomputing*, *163*, 94–105. https://doi.org/10.1016/j.neucom.2014.06.091

Dillen, W., Lombaert, G., & Schevenels, M. (2021). Performance Assessment of Metaheuristic Algorithms for Structural Optimization Taking Into Account the Influence of Algorithmic Control Parameters. *Frontiers in Built Environment*, *7*. https://doi.org/10.3389/fbuil.2021.618851

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. (2012). Solution approaches for a real-life resource-constrained parallel machine scheduling problem. *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*. https://doi.org/10.1016/j.eswax.2020.10

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

Feo, T. A., & Resende, M. G. C. (1989). A PROBABILISTIC HEURISTIC FOR A COMPUTATIONALLY DIFFICULT SET COVERING PROBLEM *. In *Operations Research Letters* (Vol. 8).

Feo, T. A., & Resende, M. G. C. (1995). Greedy Randomized Adaptive Search Procedures. *Journal of Global Optimization*, *6*(2), 109–133. https://doi.org/10.1007/BF01096763

Ferreira, A. R. (2023). *PROGRAMAÇÃO DE MÁQUINAS PARALELAS DEDICADAS COM SETUPS DEPENDENTES DA SEQUÊNCIA DE FAMÍLIAS, RECURSOS ADICIONAIS E DATAS DE ENTREGA*.

Garey, M. R., & Johnson, D. S. (1979). *Computers and intractability: A guide to the theory of NP-Completeness*.

Graham, R. L., Lawler, E. L., Lenstra, J. K., & Kan, A. H. G. R. (1979). Optimization and approximation in deterministic sequencing and scheduling: A survey. *Annals of Discrete Mathematics*, 287–326.

Lopez-Esteve, A., Perea, F., & Yepes-Borrero, J. C. (2023). GRASP algorithms for the unrelated parallel machines scheduling problem with additional resources during processing and setups. *International Journal of Production Research*, *61*(17), 6013–6029. https://doi.org/10.1080/00207543.2022.2121869

Mokotoff, E. (2001). Parallel Machine Scheduling Problems - A Survey. *Asia-Pacific Journal of Operational Research 18*, 193–242.

Parreño, F., Alvarez-Valdes, R., Oliveira, J. F., & Tamarit, J. M. (2010). A hybrid GRASP/VND algorithm for two- and three-dimensional bin packing. *Annals of Operations Research*, *179*(1), 203–220. https://doi.org/10.1007/s10479-008-0449-4

Pereira Coutinho, C. (2014). *Metodologia de Investigação em Ciências Sociais e Humanas: teoria e prática*. www.almedina.net

Pinedo, M. (1999). *Planning and Scheduling in Manufacturing and Services*. Scholars Portal.

Pinto, J. M., & Grossmann, I. E. (1997). A logic-based approach to scheduling problems with resource constraints. In *Computers chem. Engng* (Vol. 21, Issue 8).

Resende, M. G. C., & Ribeiro, C. C. (2008). *GRASP: GREEDY RANDOMIZED ADAPTIVE SEARCH PROCEDURES*.

Saunders, M. N., Lewis, P., & Thornhill, A. (2023). *Research Methods for Business Students*. https://www.researchgate.net/publication/367780349

Slowinski, R. (1980). Two Approaches to Problems of Resource Allocation Among Project Activities - a Comparative Study. *Journal of the Operational Research Society*, *31*, 711–723.

Stockemer, D. (2018). Quantitative Methods for the Social Sciences: A Practical Introduction with Examples in SPSS and Stata. In *Quantitative Methods for the Social Sciences: A Practical Introduction with Examples in SPSS and Stata*. Springer International Publishing. https://doi.org/10.1007/978-3-319-99118-4

T'kindt, V., & Billaut, J.-C. (2006). *Multicriteria Scheduling: Theory, Models and Algorithms* (Second). Springer.

Traqueia, A., Euzébio, C., Soares Diana, Pacheco, E., Taveira, E., Bernardo, I., Rios, J., Sousa, L., Bento Lopes, M., & Soares, T. (2021). *Reflexões em torno de Metodologias de Investigação: métodos*. https://www.researchgate.net/publication/350314320

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

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

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*. https://doi.org/10.1016/j.eswa.2019.112959

Ying, K. C., Lee, Z. J., & Lin, S. W. (2012). Makespan minimization for scheduling unrelated parallel machines with setup times. *Journal of Intelligent Manufacturing*, *23*(5), 1795–1803. https://doi.org/10.1007/s10845-010-0483-3

Yunusoglu, P., & Topaloglu Yildiz, S. (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

Anexo A

Figura 45 – Utilização de equipas de *setup* na instância de 04/09/2024

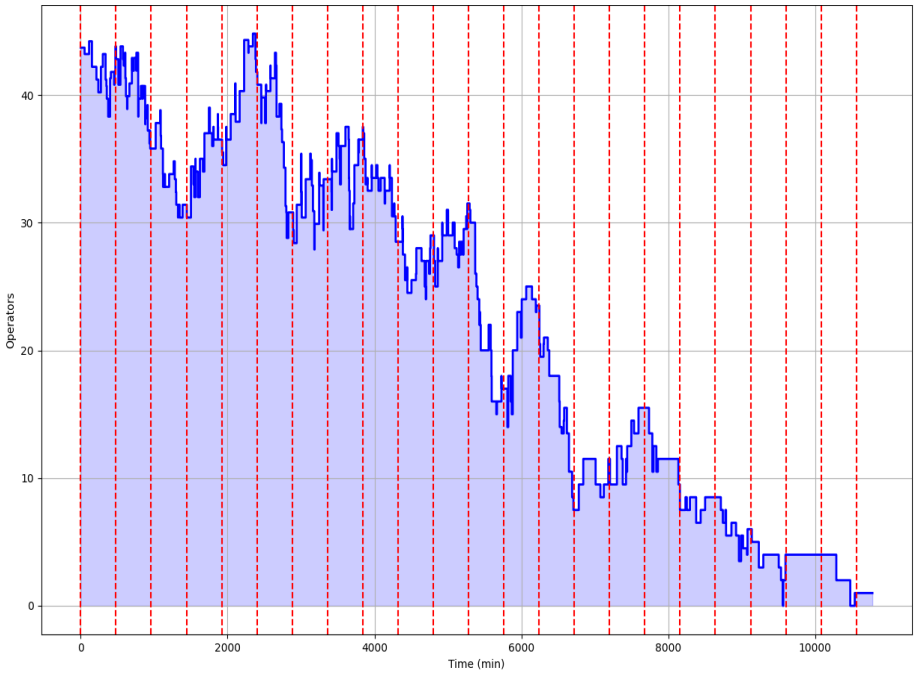


Figura 46 - Utilização de operadores na instância de 04/09/2024

Anexo B

Tabela 37 - Comportamento dos indicadores de desempenho ao longo do GRASP (Instância 2)

Fase	Nº Trabalhos Atrasados	Nº Setups	Makespan	Nº Periodos Ociosos	Total de Tempo Improdutivo
CPhase - iter 1	11	142	10454	0	0
RepPhase - iter 1	14	142	10510	103	12350
LSPhase 1 - iter 1	8	142	10597	94	11869
LSPhase 2 - iter 1	8	135	10696	89	9858
CPhase - iter 2	8	140	10576	0	0
RepPhase - iter 2	12	140	10756	104	12522
LSPhase 1 - iter 2	11	140	10748	107	12929
LSPhase 2 - iter 2	10	130	10659	85	9654
CPhase - iter 3	12	144	10515	0	0
RepPhase - iter 3	17	144	10657	106	14325
LSPhase 1 - iter 3	12	144	10797	116	15393
LSPhase 2 - iter 3	11	134	10689	84	7679
CPhase - iter 4	9	140	10637	0	0
RepPhase - iter 4	12	140	10847	97	11978
LSPhase 1 - iter 4	7	139	10862	102	12301
LSPhase 2 - iter 4	7	138	10796	84	10538
CPhase - iter 5	6	148	10698	0	0
RepPhase - iter 5	14	148	10957	108	12017
LSPhase 1 - iter 5	8	145	10736	105	11286
LSPhase 2 - iter 5	7	143	10660	104	9628
CPhase - iter 6	5	139	10576	0	0
RepPhase - iter 6	13	139	10746	107	11451
LSPhase 1 - iter 6	9	138	10628	101	10944
LSPhase 2 - iter 6	8	123	10585	65	8810
CPhase - iter 7	6	146	10698	0	0
RepPhase - iter 7	17	146	12627	102	15087
LSPhase 1 - iter 7	13	146	10951	109	13600
LSPhase 2 - iter 7	12	144	10951	82	11701
CPhase - iter 8	10	150	10657	0	0
RepPhase - iter 8	17	150	11342	117	14870
LSPhase 1 - iter 8	14	150	10862	110	13853
LSPhase 2 - iter 8	13	147	10862	104	12973

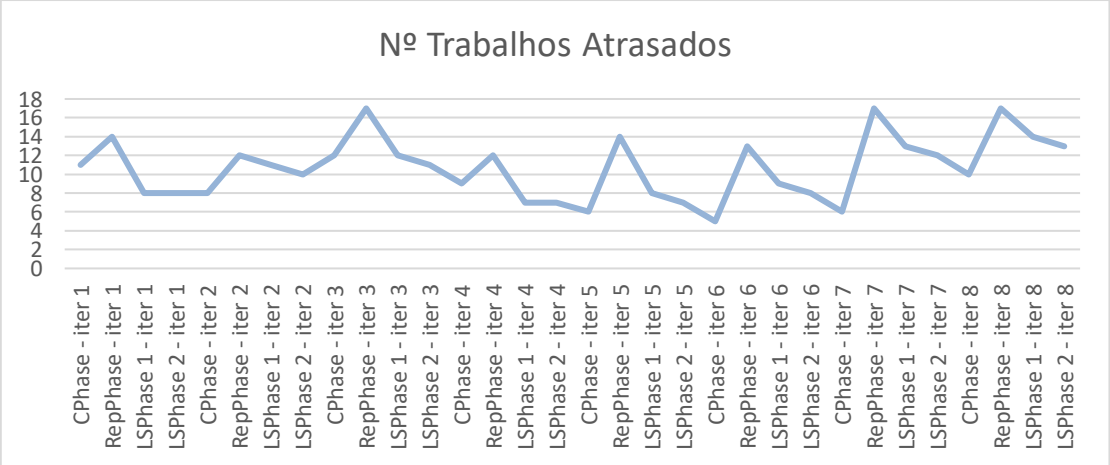


Figura 47 - Comportamento da variação do Nº Trabalhos Atrasados ao longo do GRASP (Instância 2)

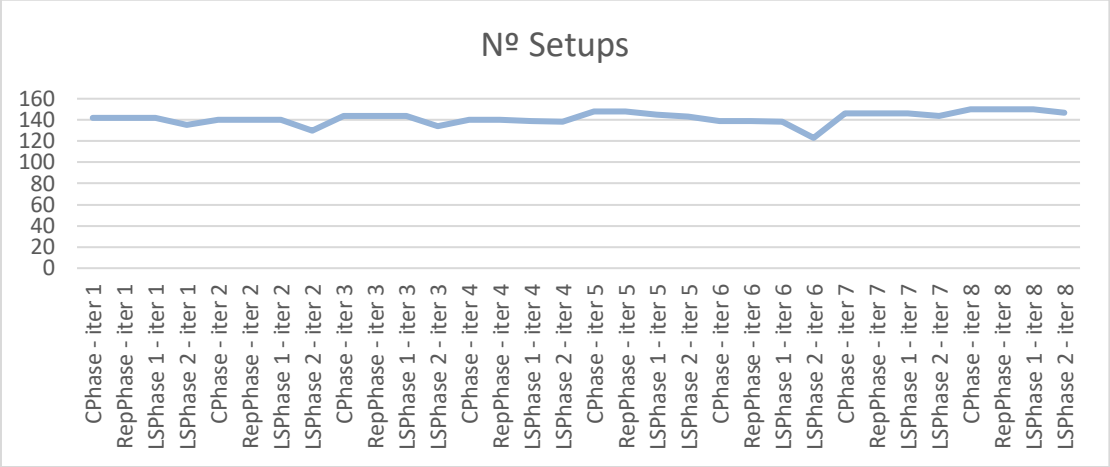


Figura 48 - Comportamento da variação do Nº *Setups* ao longo do GRASP (Instância 2)

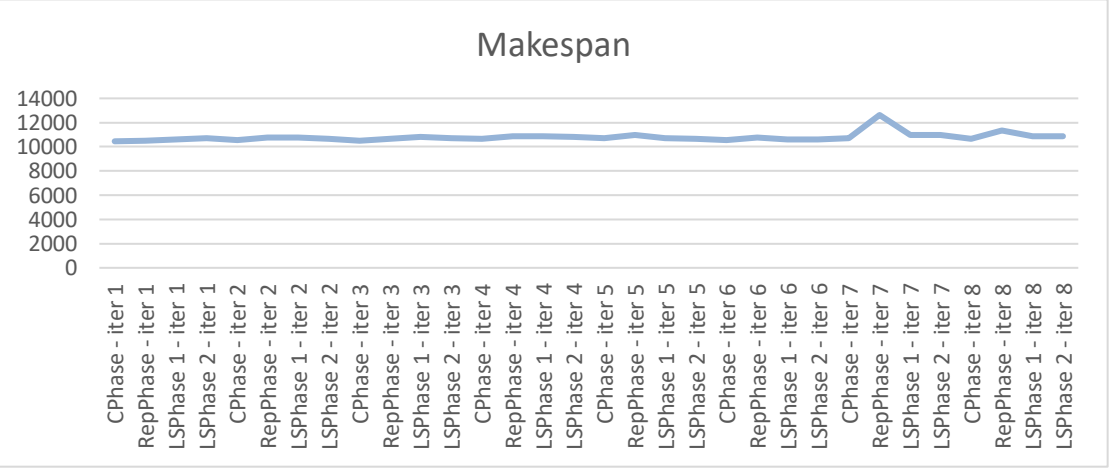


Figura 49 - Comportamento da variação do Makespan ao longo do GRASP (Instância 2)

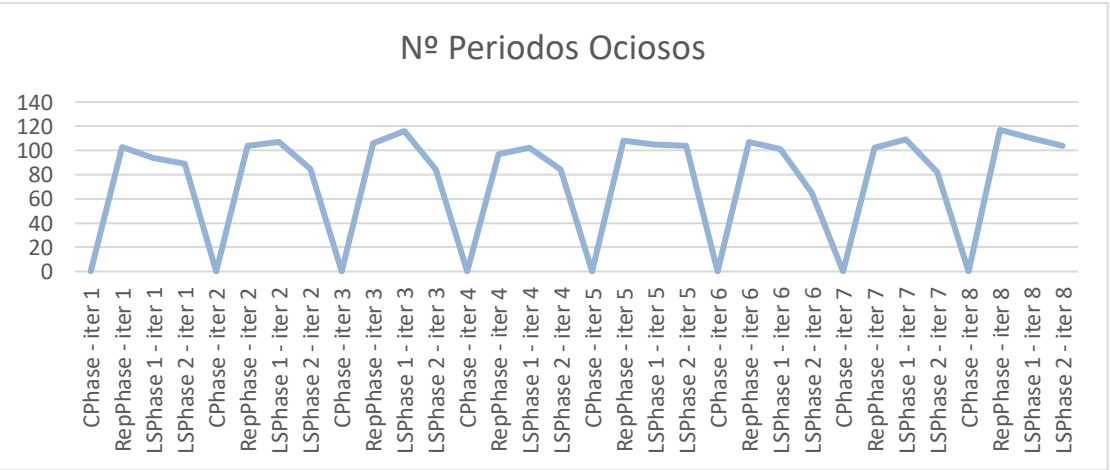


Figura 50 - Comportamento da variação do Nº Períodos Ociosos ao longo do GRASP (Instância 2)

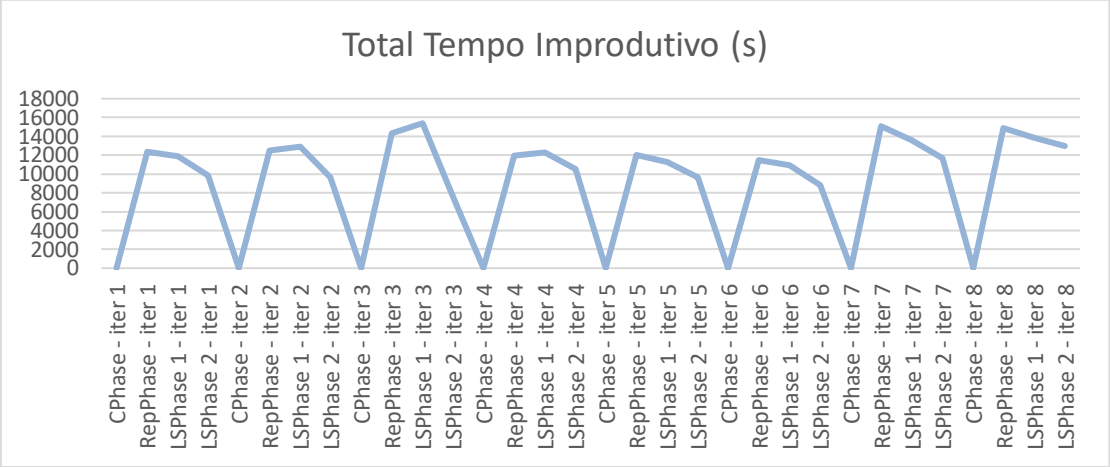


Figura 51 - Comportamento da variação do Total Tempo Improdutivo ao longo do GRASP (Instância 2)

Tabela 38 - Comportamento dos indicadores de desempenho ao longo do GRASP (Instância 3)

Fase	Nº Trabalhos Atrasados	Nº Setups	Makespan	Nº Periodos Ociosos	Total de Tempo Improdutivo
CPhase - iter 1	9	166	10174	0	0
RepPhase - iter 1	30	166	11427	137	21138
LSPhase 1 - iter 1	12	165	11229	123	15808
LSPhase 2 - iter 1	11	161	10575	122	15340
CPhase - iter 2	8	163	10229	0	0
RepPhase - iter 2	18	163	10690	132	15904
LSPhase 1 - iter 2	9	160	11372	116	12554
LSPhase 2 - iter 2	9	157	11394	114	13746
CPhase - iter 3	10	166	10351	0	0
RepPhase - iter 3	41	166	11869	124	19492
LSPhase 1 - iter 3	30	159	14112	114	18824
LSPhase 2 - iter 3	18	147	10755	97	7854
CPhase - iter 4	3	165	10229	0	0
RepPhase - iter 4	37	165	13781	129	21161
LSPhase 1 - iter 4	9	164	11730	128	18456
LSPhase 2 - iter 4	9	155	11633	109	14570
CPhase - iter 5	8	160	10290	0	0
RepPhase - iter 5	47	160	14446	120	20890
LSPhase 1 - iter 5	20	162	10956	128	14317
LSPhase 2 - iter 5	20	153	10684	107	14629
CPhase - iter 6	9	159	10229	0	0
RepPhase - iter 6	15	159	10703	124	15262
LSPhase 1 - iter 6	12	159	10694	123	13359
LSPhase 2 - iter 6	12	155	10585	114	12517
CPhase - iter 7	11	165	10357	0	0
RepPhase - iter 7	28	165	11449	138	21044
LSPhase 1 - iter 7	18	165	11156	119	17488
LSPhase 2 - iter 7	17	150	10736	109	13199
CPhase - iter 8	9	166	10229	0	0
RepPhase - iter 8	33	166	12518	129	22092
LSPhase 1 - iter 8	15	167	10706	133	17828
LSPhase 2 - iter 8	15	161	10650	123	15697

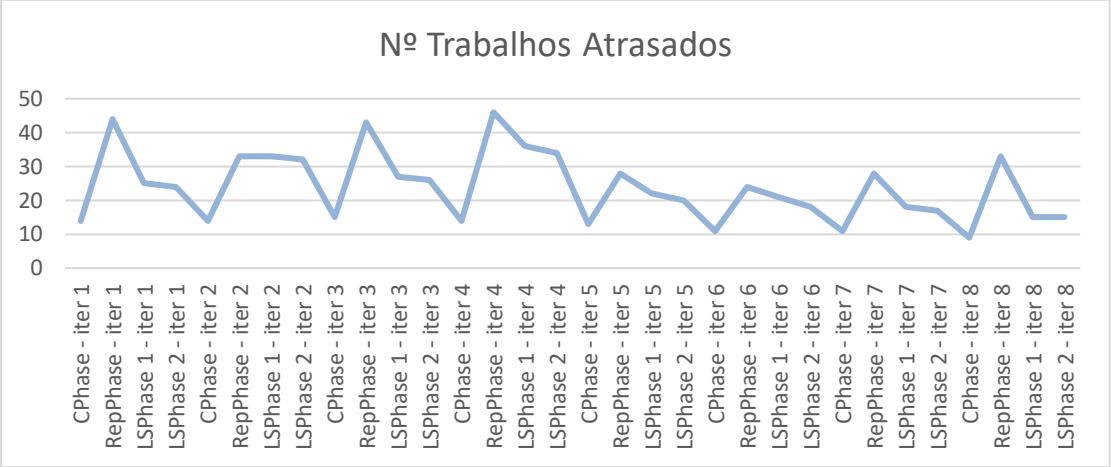


Figura 52 - Comportamento da variação do Nº Trabalhos Atrasados ao longo do GRASP (Instância 3)

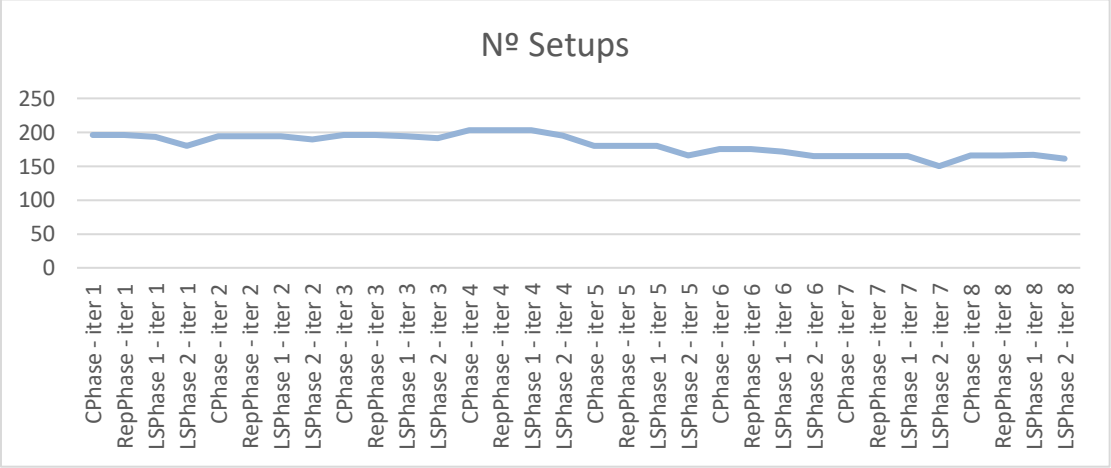


Figura 53 - Comportamento da variação do Nº Setups ao longo do GRASP (Instância 3)

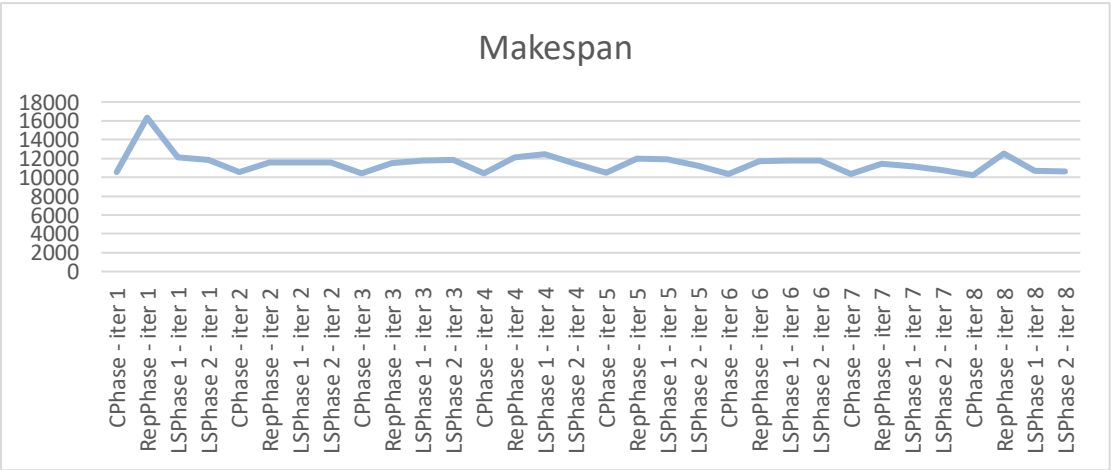


Figura 54 - Comportamento da variação do *Makespan* ao longo do GRASP (Instância 3)

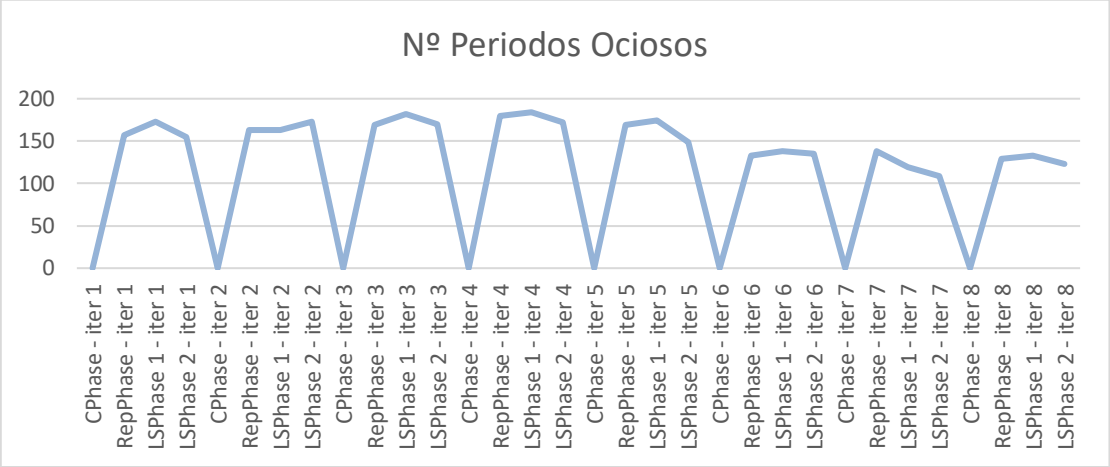


Figura 55 - Comportamento da variação do Nº Períodos Ociosos ao longo do GRASP (Instância 3)

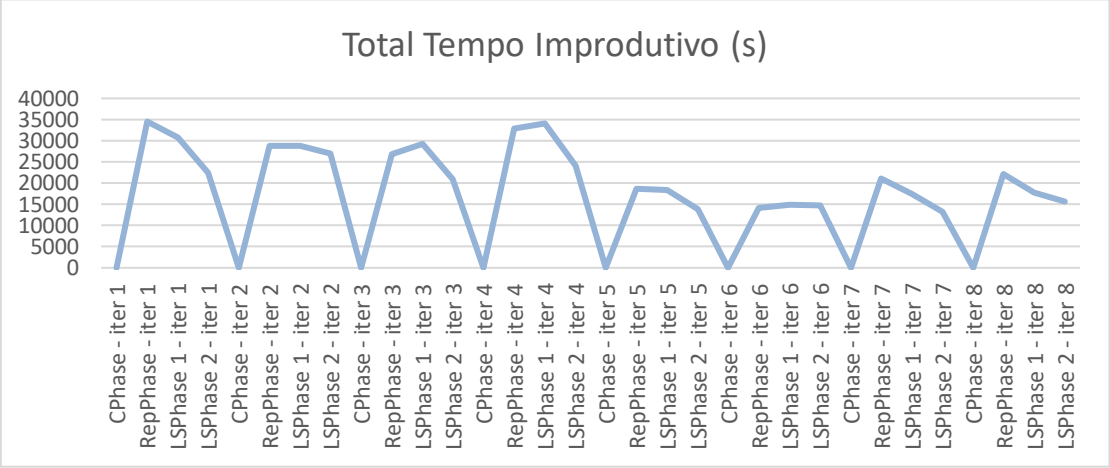


Figura 56 - Comportamento da variação do Total Tempo Improdutivo ao longo do GRASP (Instância 3)

Tabela 39 - Comportamento dos indicadores de desempenho ao longo do GRASP (Instância 4)

Fase	Nº Trabalhos Atrasados	Nº Setups	Makespan	Nº Periodos Ociosos	Total de Tempo Improdutivo
CPhase - iter 1	7	173	10726	0	0
RepPhase - iter 1	18	173	11028	139	19246
LSPhase 1 - iter 1	10	169	11042	139	19281
LSPhase 2 - iter 1	10	162	10789	127	17252
CPhase - iter 2	8	167	10543	0	0
RepPhase - iter 2	19	167	10773	125	14171
LSPhase 1 - iter 2	19	165	10689	122	13784
LSPhase 2 - iter 2	17	159	10711	113	10931
CPhase - iter 3	7	172	10665	0	0
RepPhase - iter 3	15	172	10967	142	18138
LSPhase 1 - iter 3	12	172	11011	135	17056
LSPhase 2 - iter 3	11	163	10904	122	14070
CPhase - iter 4	8	176	10726	0	0
RepPhase - iter 4	34	176	15953	132	19000
LSPhase 1 - iter 4	16	175	11477	130	12413
LSPhase 2 - iter 4	14	174	11416	128	13315
CPhase - iter 5	6	172	10665	0	0
RepPhase - iter 5	19	172	10912	137	19516
LSPhase 1 - iter 5	8	170	11023	146	20427
LSPhase 2 - iter 5	8	166	10912	133	15036
CPhase - iter 6	8	165	10543	0	0
RepPhase - iter 6	29	165	11505	118	15817
LSPhase 1 - iter 6	13	166	11515	126	14960
LSPhase 2 - iter 6	11	156	11481	113	11870

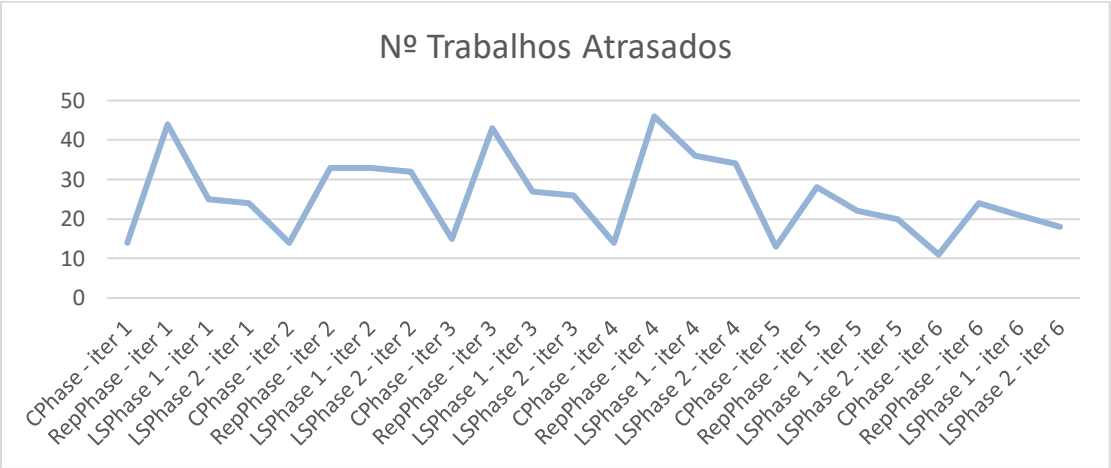


Figura 57 - Comportamento da variação do Nº Trabalhos Atrasados ao longo do GRASP (Instância 4)

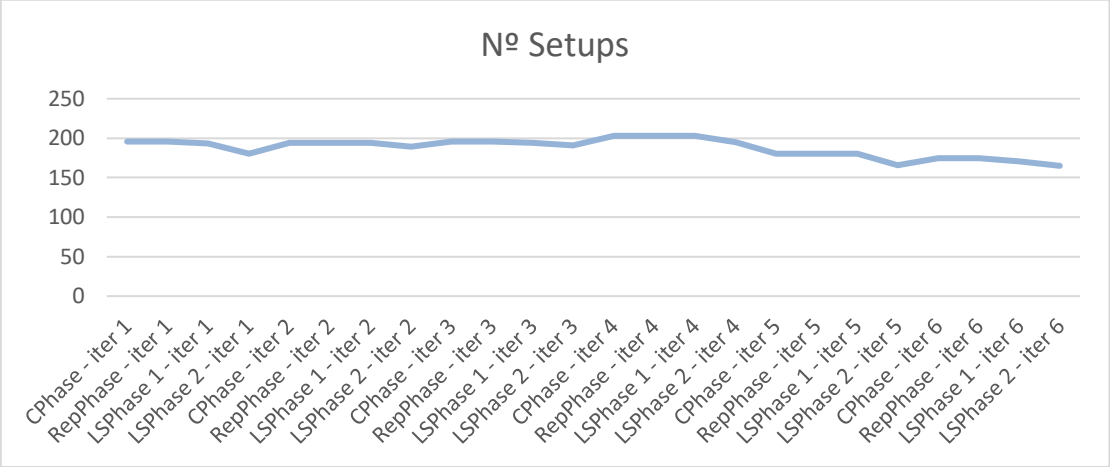


Figura 58 - Comportamento da variação do Nº *Setups* ao longo do GRASP (Instância 4)

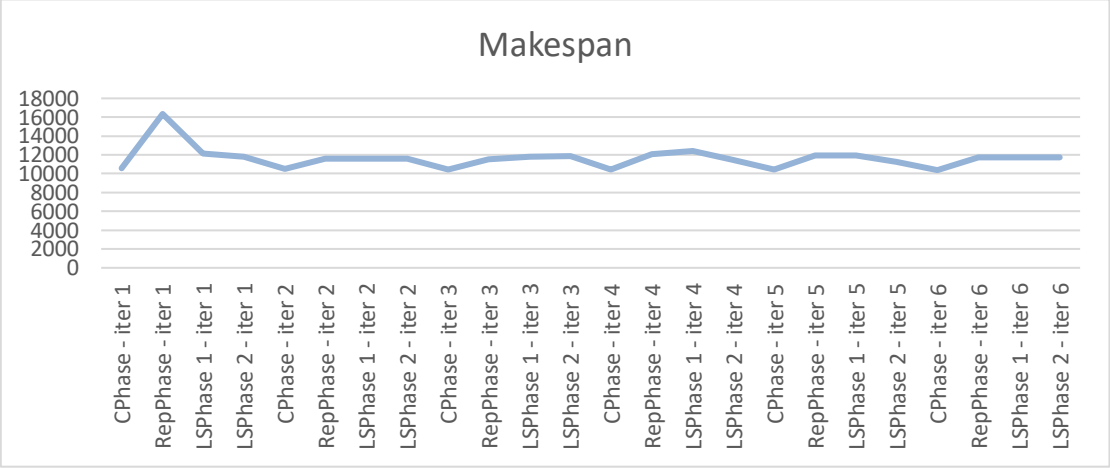


Figura 59 - Comportamento da variação do Makespan ao longo do GRASP (Instância 4)

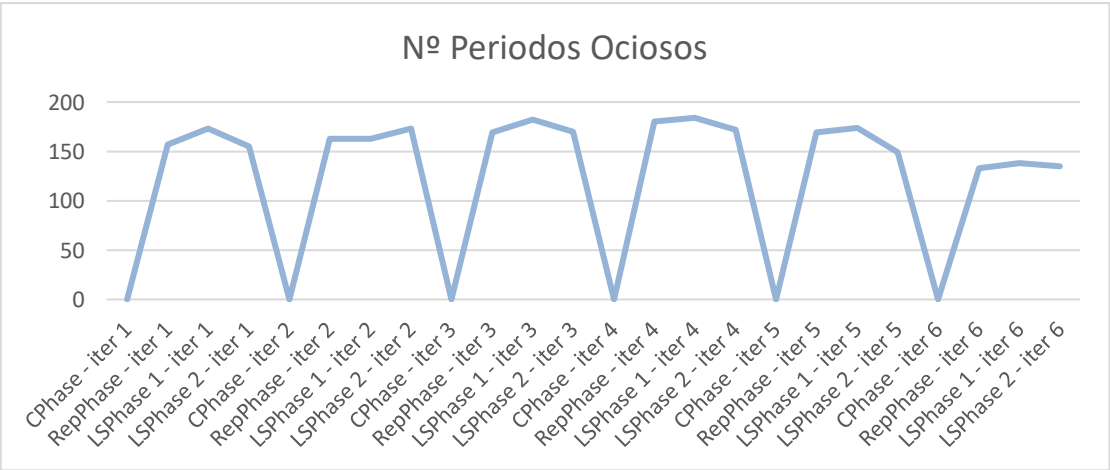


Figura 60 - Comportamento da variação do Nº Períodos Ociosos ao longo do GRASP (Instância 4)

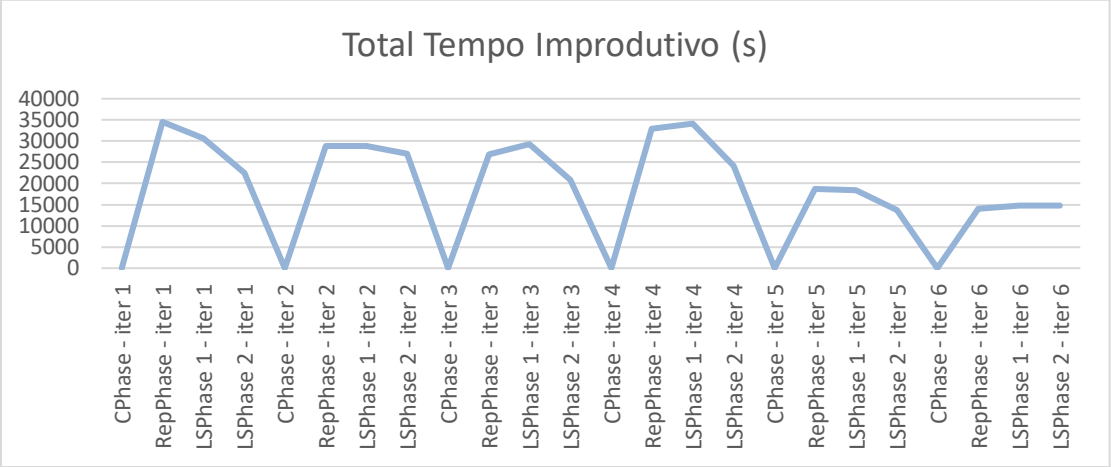


Figura 61 - Comportamento da variação do Total Tempo Improdutivo ao longo do GRASP (Instância 4)

Tabela 40 - Comportamento dos indicadores de desempenho ao longo do GRASP (Instância 5)

Fase	Nº Trabalhos Atrasados	Nº Setups	Makespan	Nº Periodos Ociosos	Total de Tempo Improdutivo
CPhase - iter 1	15	180	10408	0	0
RepPhase - iter 1	31	180	11420	157	22372
LSPhase 1 - iter 1	20	181	11424	163	18798
LSPhase 2 - iter 1	19	177	10944	136	16360
CPhase - iter 2	13	182	10469	0	0
RepPhase - iter 2	43	182	17585	166	28060
LSPhase 1 - iter 2	30	173	17639	152	23351
LSPhase 2 - iter 2	24	163	10853	147	12083
CPhase - iter 3	13	170	10347	0	0
RepPhase - iter 3	34	170	11680	146	16523
LSPhase 1 - iter 3	17	168	12122	156	19358
LSPhase 2 - iter 3	17	168	12122	156	19358
CPhase - iter 4	14	174	10652	0	0
RepPhase - iter 4	23	174	10989	156	12897
LSPhase 1 - iter 4	15	171	10810	156	13099
LSPhase 2 - iter 4	15	165	10782	130	11333
CPhase - iter 5	13	180	10469	0	0
RepPhase - iter 5	28	180	11952	169	18737
LSPhase 1 - iter 5	22	180	11910	174	18348
LSPhase 2 - iter 5	20	166	11260	149	13762
CPhase - iter 6	11	175	10378	0	0
RepPhase - iter 6	24	175	11737	133	14062
LSPhase 1 - iter 6	21	171	11752	138	14844
LSPhase 2 - iter 6	18	165	11765	135	14733

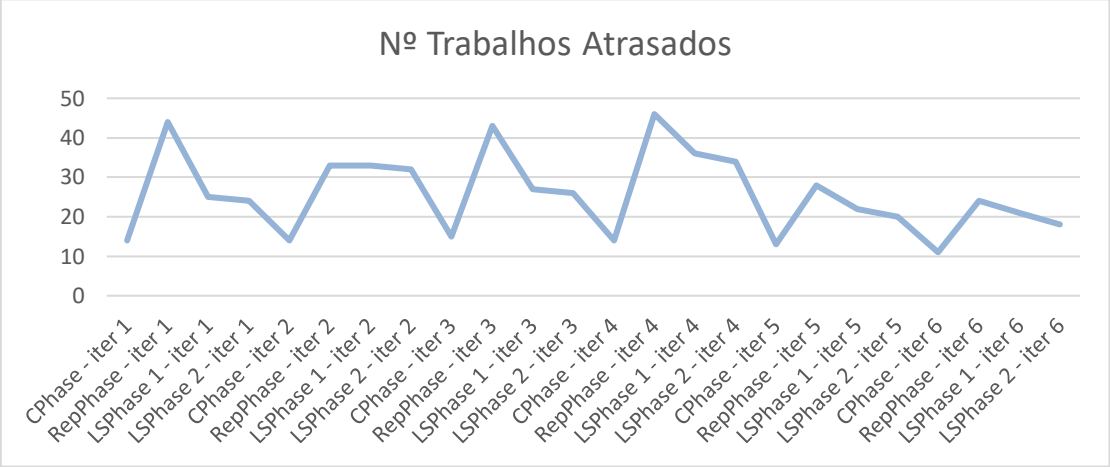


Figura 62 - Comportamento da variação do Nº Trabalhos Atrasados ao longo do GRASP (Instância 5)

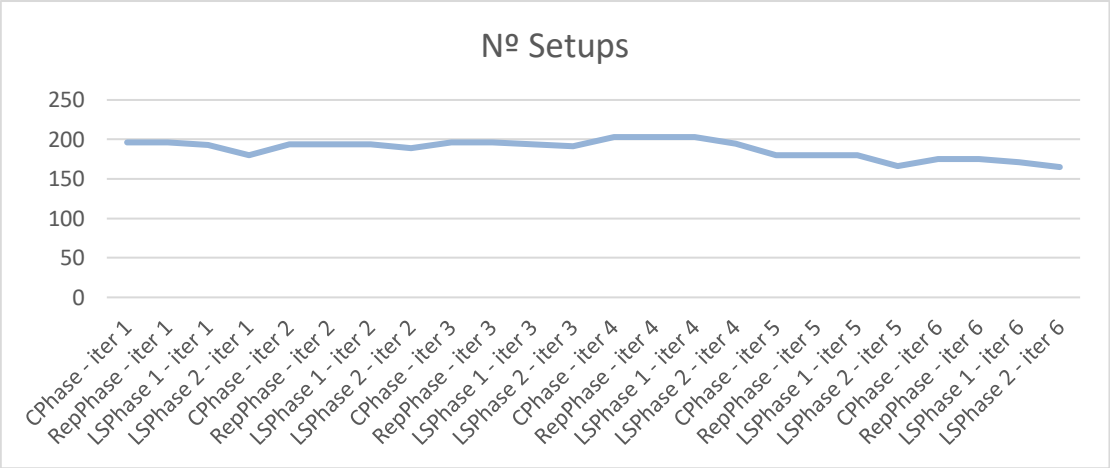


Figura 63 - Comportamento da variação do Nº *Setups* ao longo do GRASP (Instância 5)

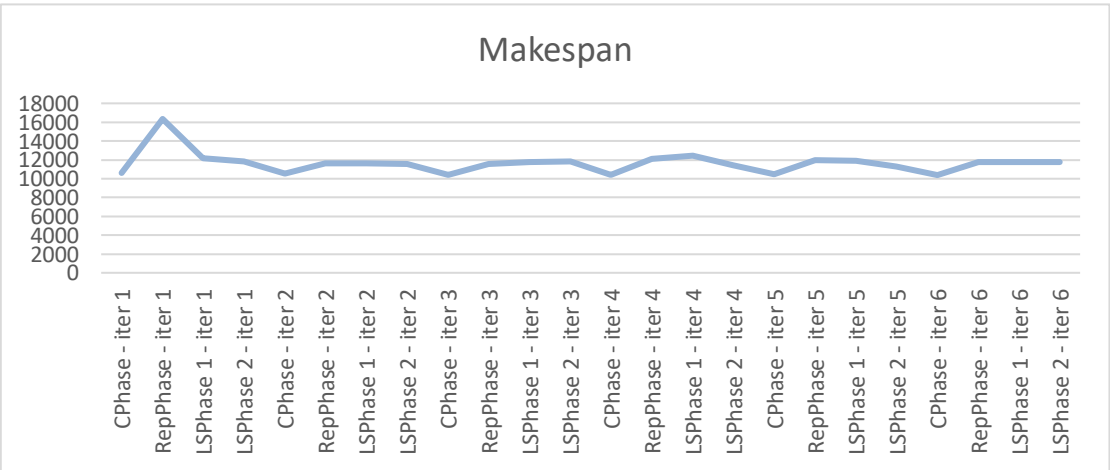


Figura 64 - Comportamento da variação do *Makespan* ao longo do GRASP (Instância 5)

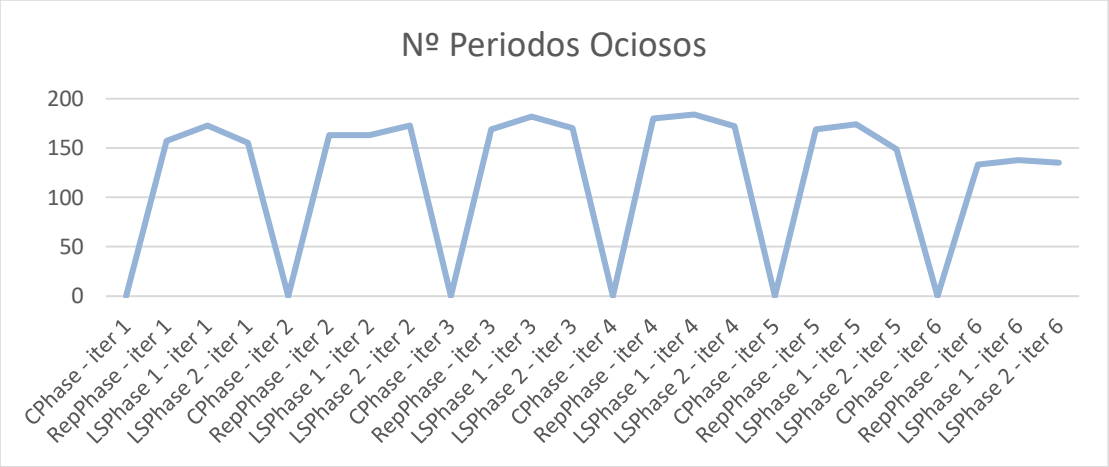


Figura 65 - Comportamento da variação do Nº Períodos Ociosos ao longo do GRASP (Instância 5)

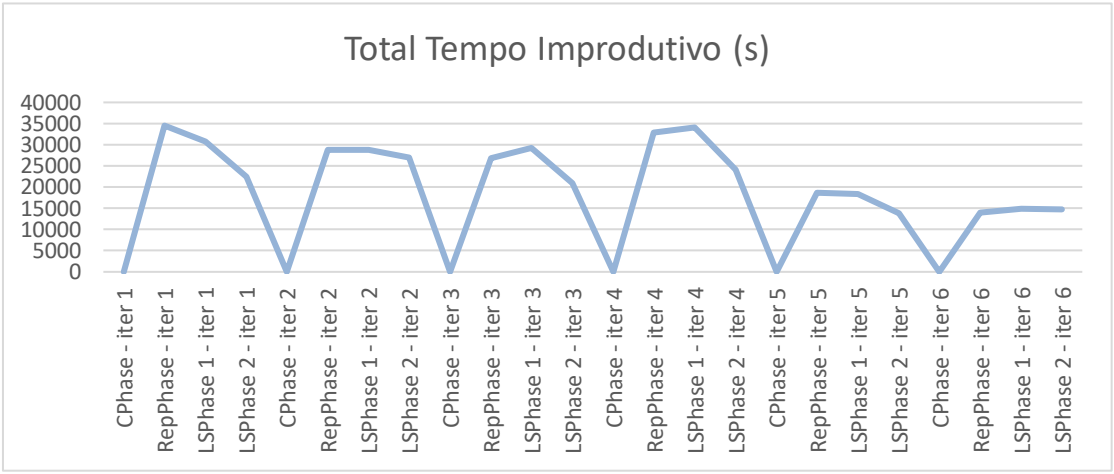


Figura 66 - Comportamento da variação do Total Tempo Improdutivo ao longo do GRASP (Instância 5)

Tabela 41 - Comportamento dos indicadores de desempenho ao longo do GRASP (Instância 6)

Fase	Nº Trabalhos Atrasados	Nº Setups	Makespan	Nº Periodos Ociosos	Total de Tempo Improdutivo
CPhase - iter 1	18	215	10800	0	0
RepPhase - iter 1	85	215	13925	198	39129
LSPhase 1 - iter 1	43	213	12236	181	31736
LSPhase 2 - iter 1	41	198	11758	167	20803
CPhase - iter 2	21	206	10922	0	0
RepPhase - iter 2	69	215	11589	161	40949
LSPhase 1 - iter 2	26	205	11890	197	41304
LSPhase 2 - iter 2	26	198	11758	185	31789
CPhase - iter 3	21	203	10922	0	0
RepPhase - iter 3	82	206	13072	193	40949
LSPhase 1 - iter 3	43	213	12236	181	31736
LSPhase 2 - iter 3	41	198	11758	167	20803

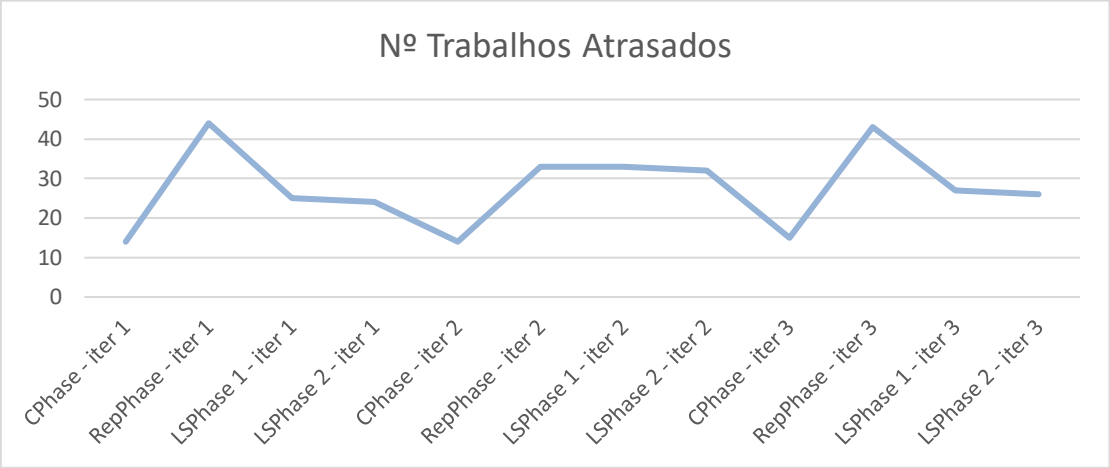


Figura 67 - Comportamento da variação do Nº Trabalhos Atrasados ao longo do GRASP (Instância 6)

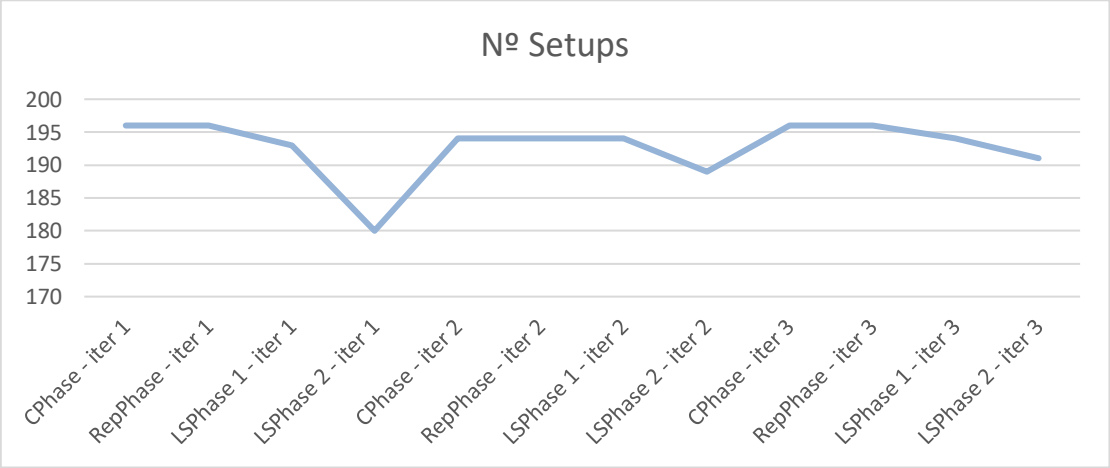


Figura 68 - Comportamento da variação do Nº *Setups* ao longo do GRASP (Instância 6)

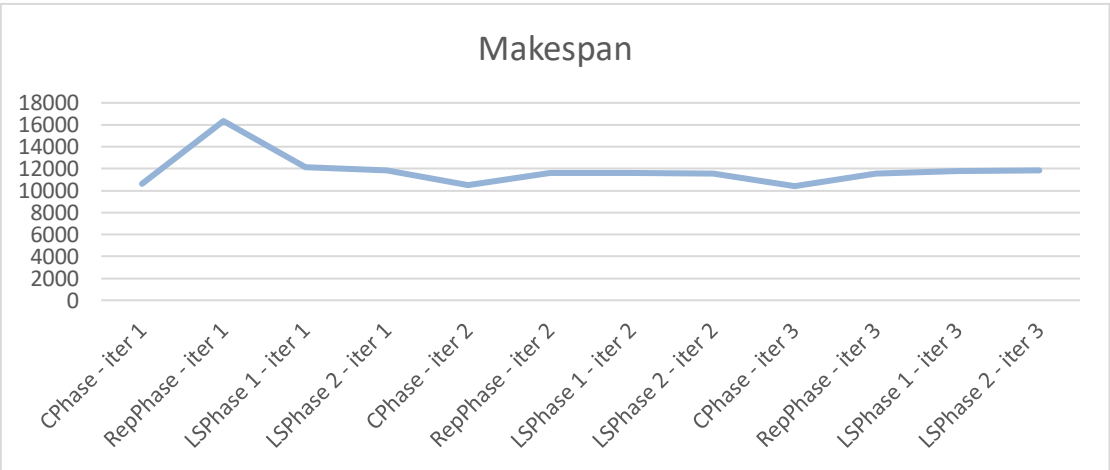


Figura 69 - Comportamento da variação do *Makespan* ao longo do GRASP (Instância 6)

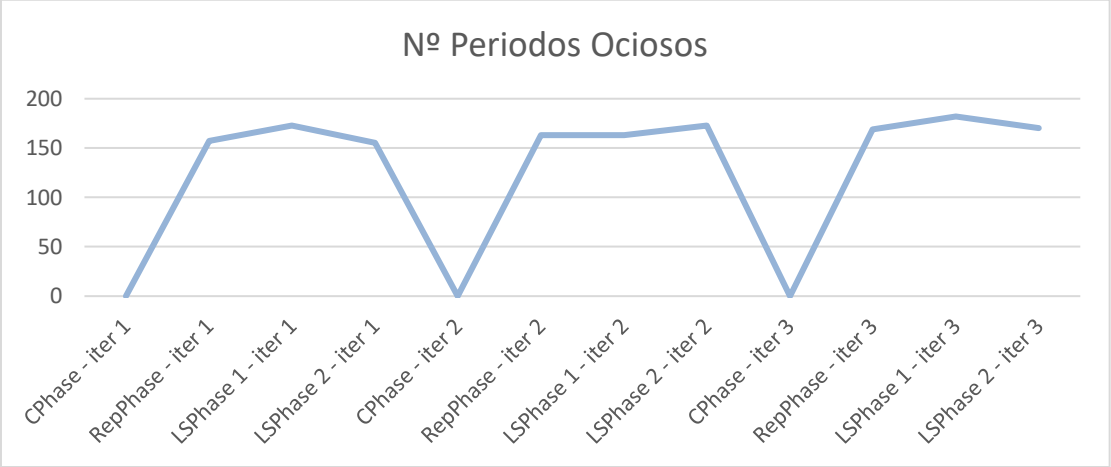


Figura 70 - Comportamento da variação do Nº Períodos Ociosos ao longo do GRASP (Instância 6)

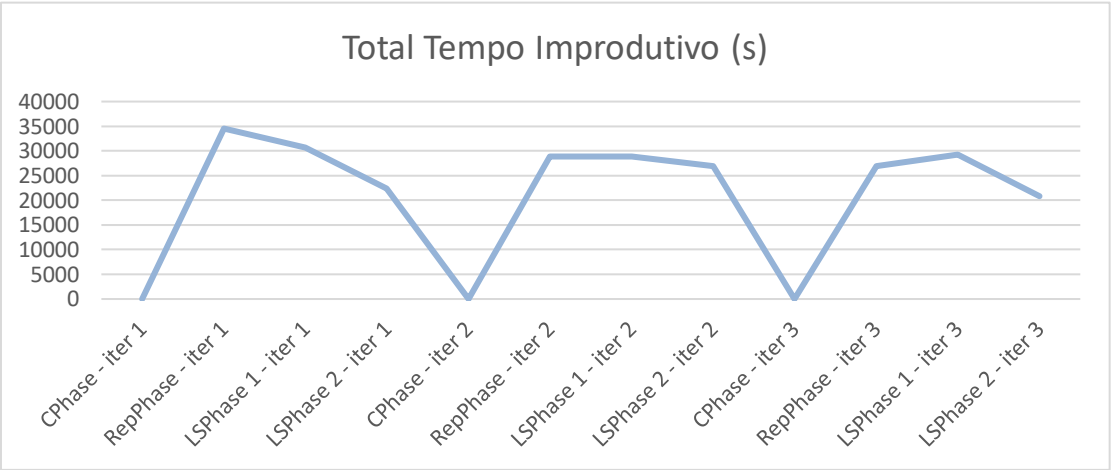


Figura 71 - Comportamento da variação do Nº Períodos Ociosos ao longo do GRASP (Instância 6)

Tabela 42 - Comportamento dos indicadores de desempenho ao longo do GRASP (Instância 7)

Fase	Nº Trabalhos Atrasados	Nº Setups	Makespan	Nº Periodos Ociosos	Total de Tempo Improdutivo
CPhase - iter 1	14	196	10590	0	0
RepPhase - iter 1	44	196	16344	157	34528
LSPhase 1 - iter 1	25	193	12151	173	30710
LSPhase 2 - iter 1	24	180	11822	155	22437
CPhase - iter 2	14	194	10529	0	0
RepPhase - iter 2	33	194	11600	163	28851
LSPhase 1 - iter 2	33	194	11600	163	28851
LSPhase 2 - iter 2	32	189	11567	173	26965
CPhase - iter 3	15	196	10407	0	0
RepPhase - iter 3	43	196	11537	169	26870
LSPhase 1 - iter 3	27	194	11782	182	29261
LSPhase 2 - iter 3	26	191	11835	170	20879
CPhase - iter 4	14	203	10407	0	0
RepPhase - iter 4	46	203	12101	180	32946
LSPhase 1 - iter 4	36	203	12429	184	34132
LSPhase 2 - iter 4	34	195	11441	172	24120

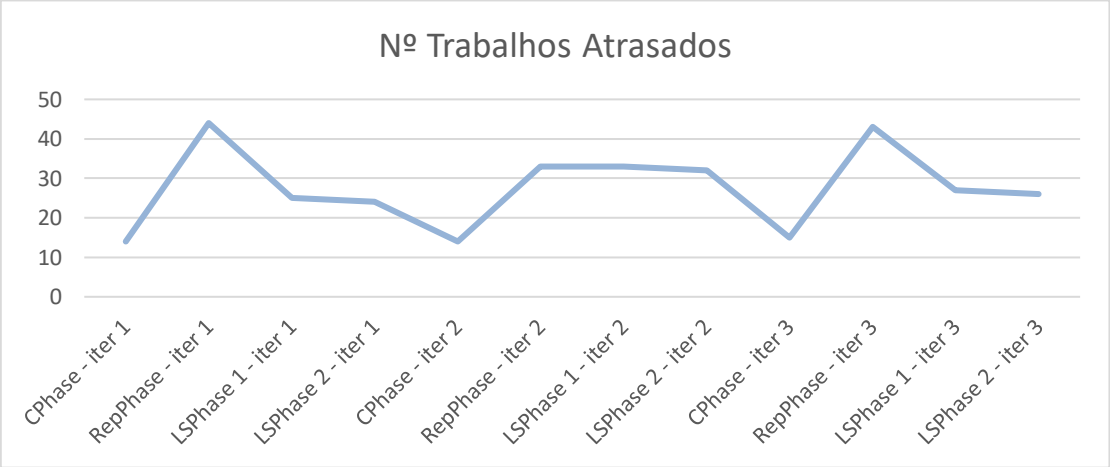


Figura 72 - Comportamento da variação do Nº Trabalhos Atrasados ao longo do GRASP (Instância 7)

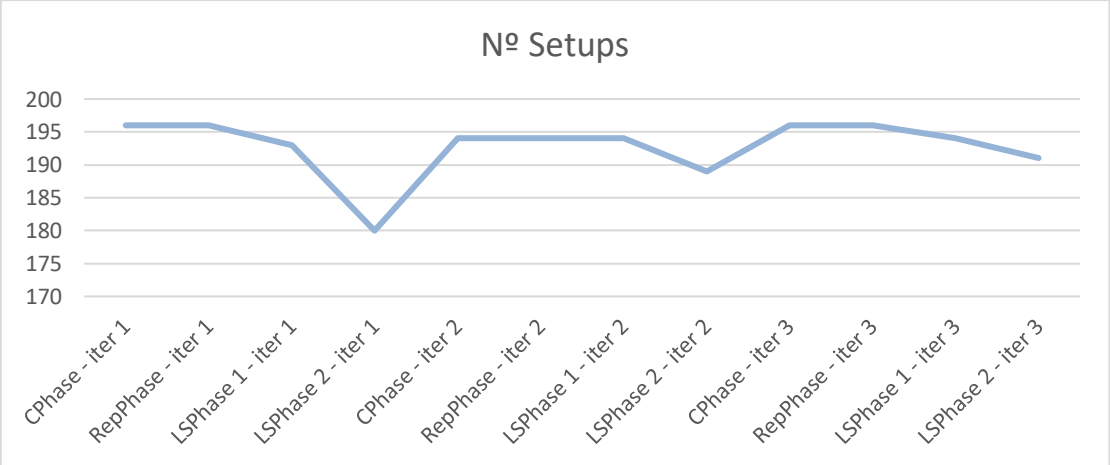


Figura 73 - Comportamento da variação do Nº *Setups* ao longo do GRASP (Instância 7)

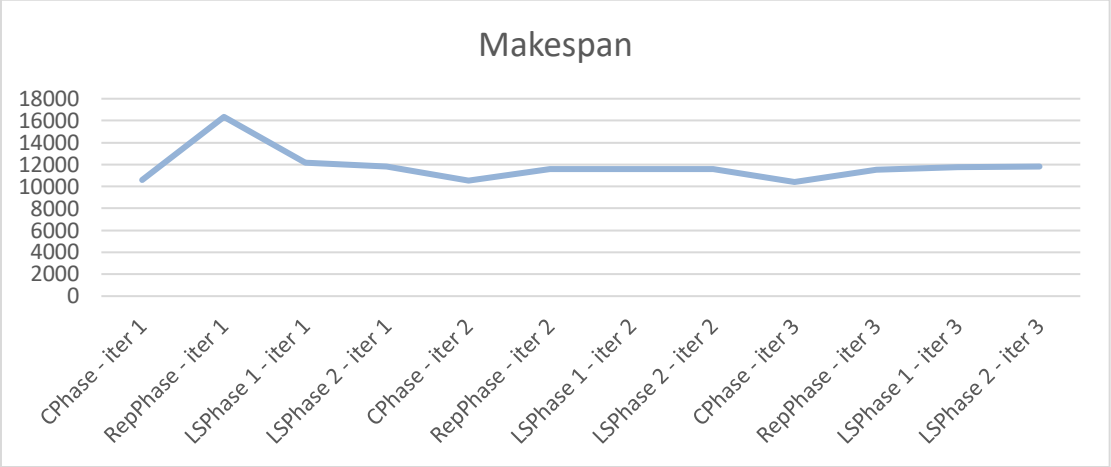


Figura 74 - Comportamento da variação *Makespan* ao longo do GRASP (Instância 7)

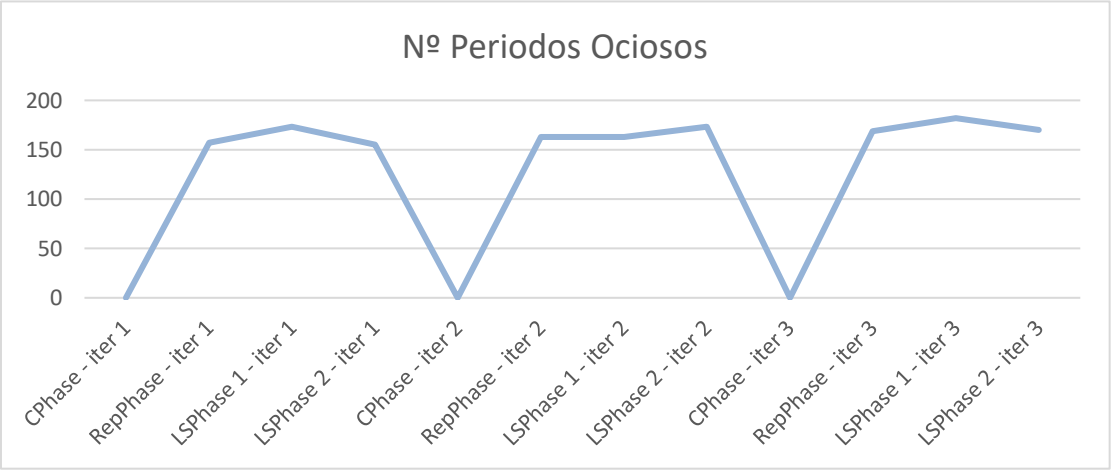


Figura 75 - Comportamento da variação do Nº Períodos Ociosos ao longo do GRASP (Instância 7)

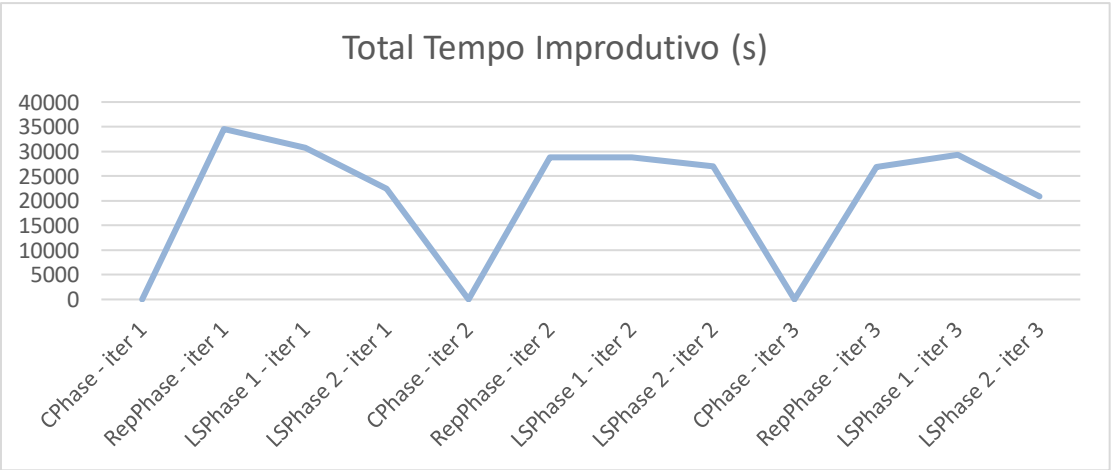


Figura 76 - Comportamento da variação do Total Tempo Improdutivo ao longo do GRASP (Instância 7)

Anexo C

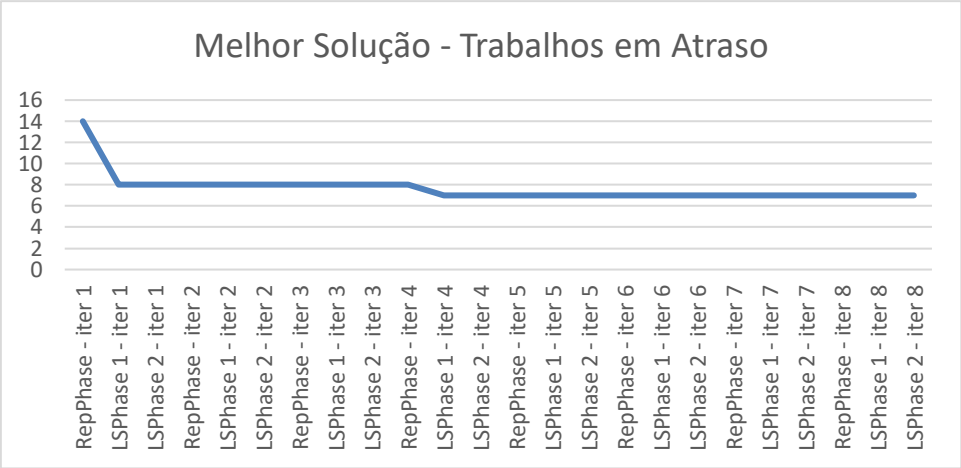


Figura 77 - Comportamento do Nº Trabalhos em Atraso da melhor solução ao longo do GRASP (Instância 2)

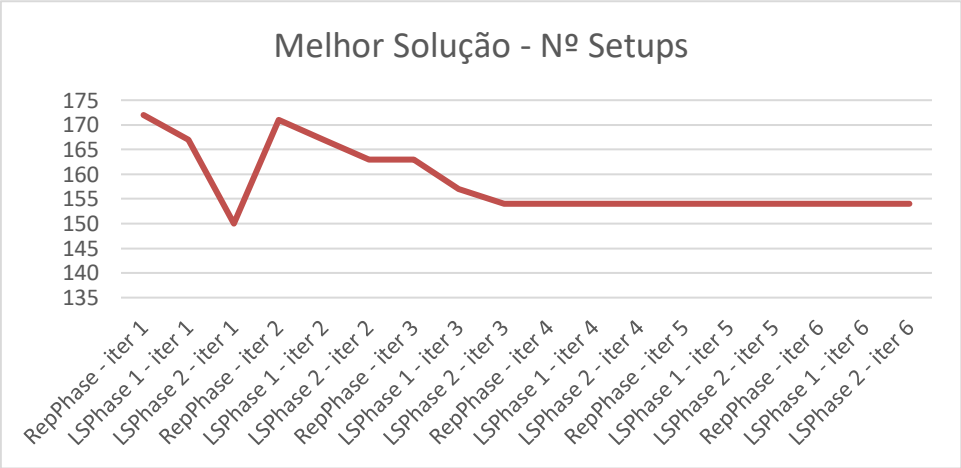


Figura 78 - Comportamento do Nº *Setups* da melhor solução ao longo do GRASP (Instância 2)

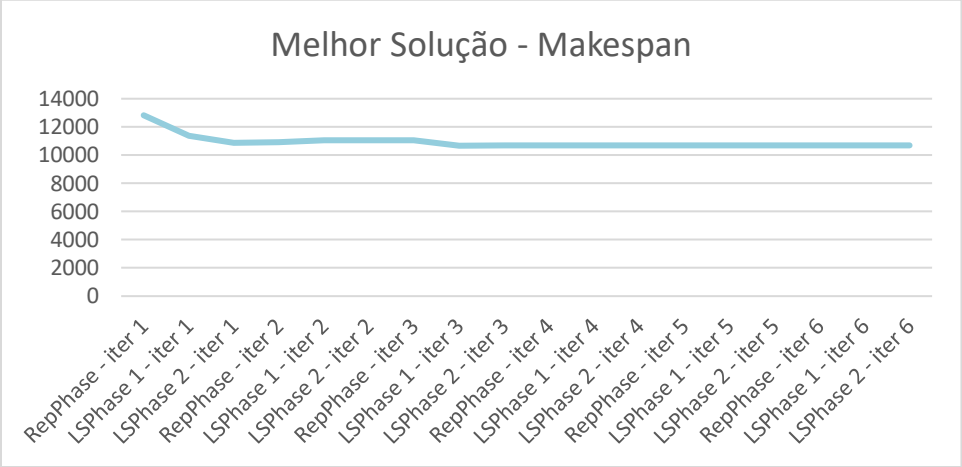


Figura 79 - Comportamento do *Makespan* da melhor solução ao longo do GRASP (Instância 2)

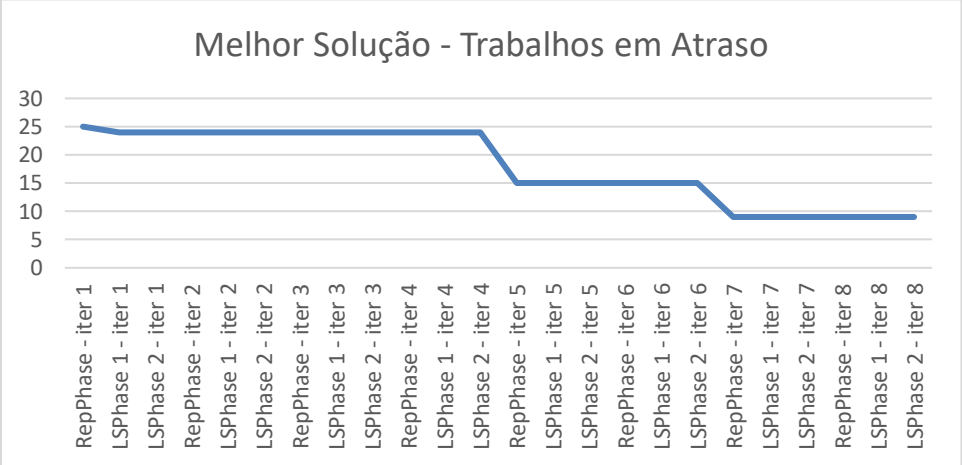


Figura 80 - Comportamento do Nº Trabalhos em Atraso da melhor solução ao longo do GRASP (Instância 3)

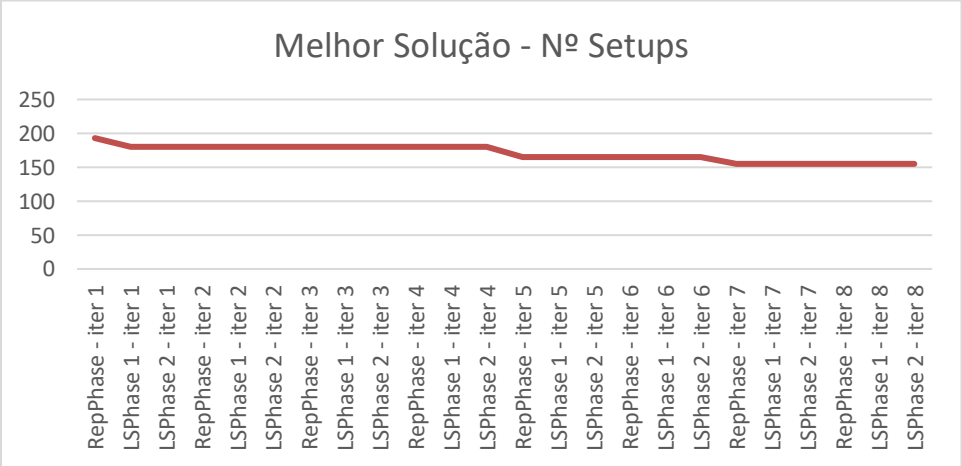


Figura 81 - Comportamento do *Makespan* da melhor solução ao longo do GRASP (Instância 3)

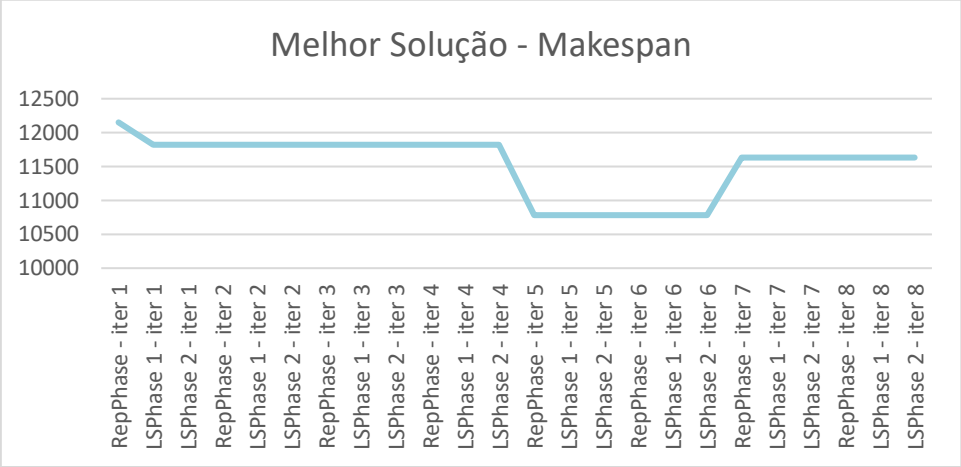


Figura 82 - Comportamento do *Makespan* da melhor solução ao longo do GRASP (Instância 3)

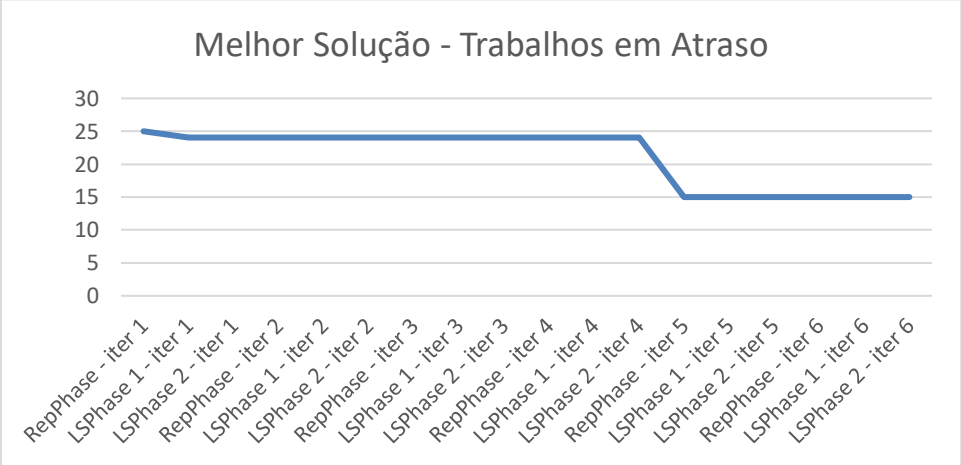


Figura 83 - Comportamento do Nº Trabalhos em Atraso da melhor solução ao longo do GRASP (Instância 4)

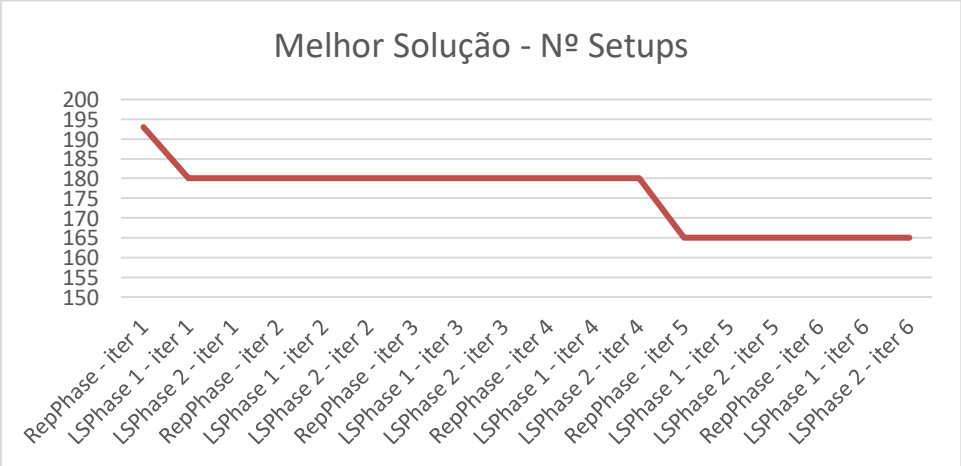


Figura 84 - Comportamento do Nº *Setups* da melhor solução ao longo do GRASP (Instância 4)

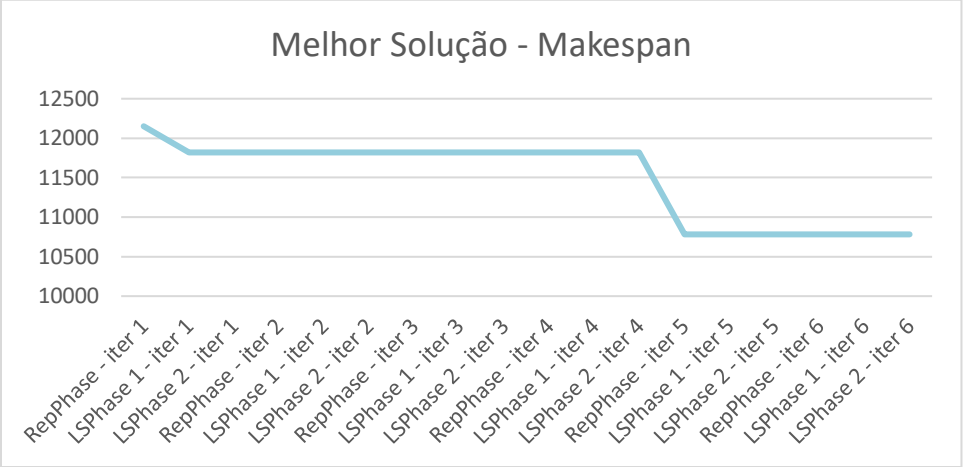


Figura 85 - Comportamento do *Makespan* da melhor solução ao longo do GRASP (Instância 4)

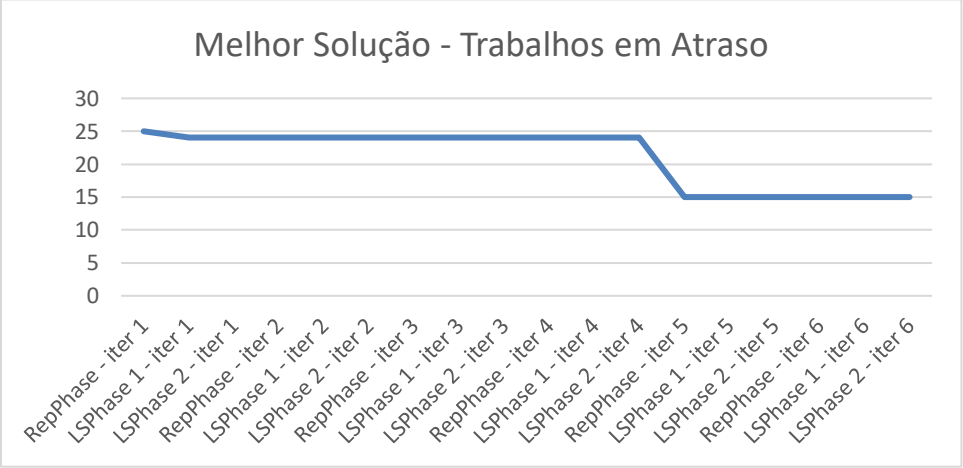


Figura 86 - Comportamento do Nº Trabalhos em Atraso da melhor solução ao longo do GRASP (Instância 5)

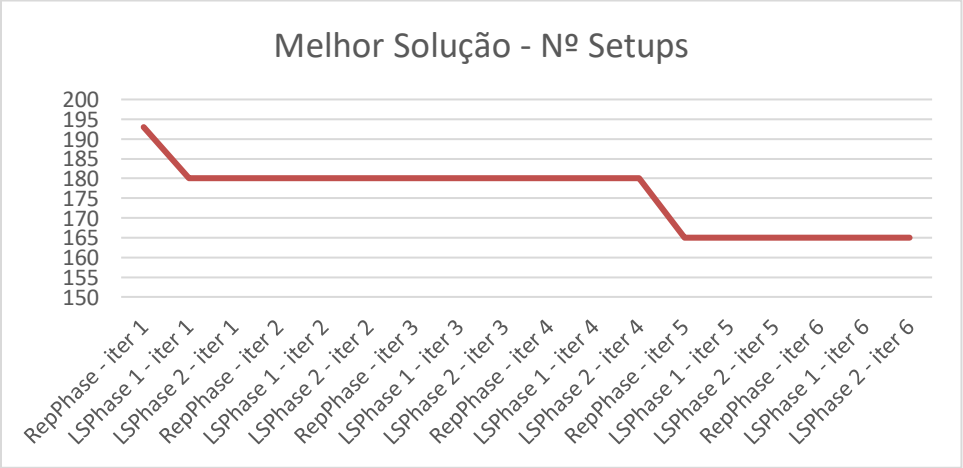


Figura 87 - Comportamento do Nº *Setups* da melhor solução ao longo do GRASP (Instância 5)

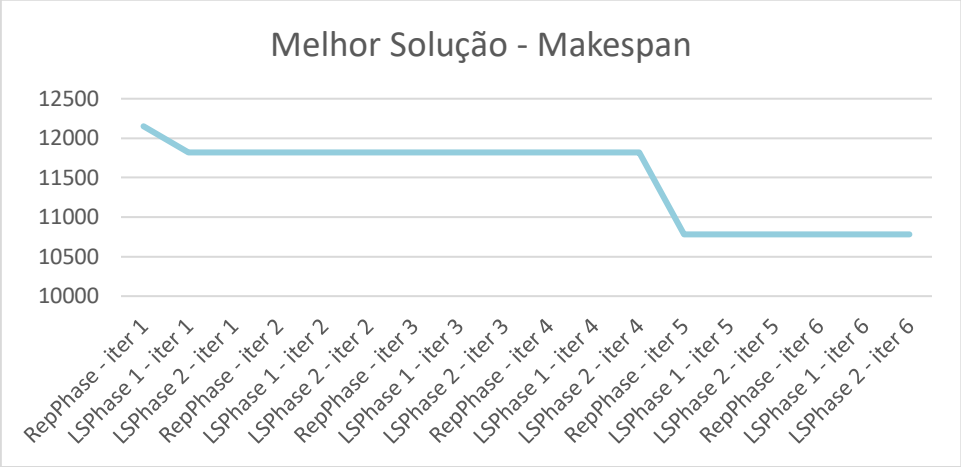


Figura 88 - Comportamento do *Makespan* da melhor solução ao longo do GRASP (Instância 5)

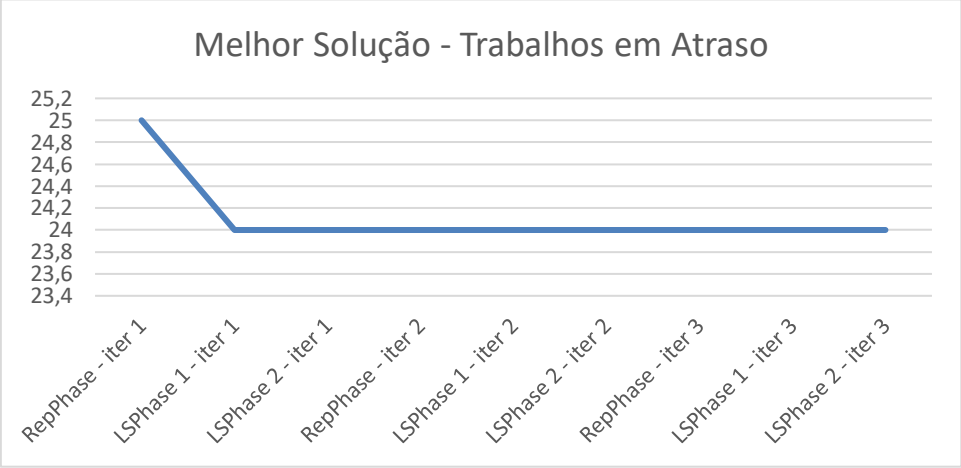


Figura 89 - Comportamento do Nº Trabalhos em Atraso da melhor solução ao longo do GRASP (Instância 6)

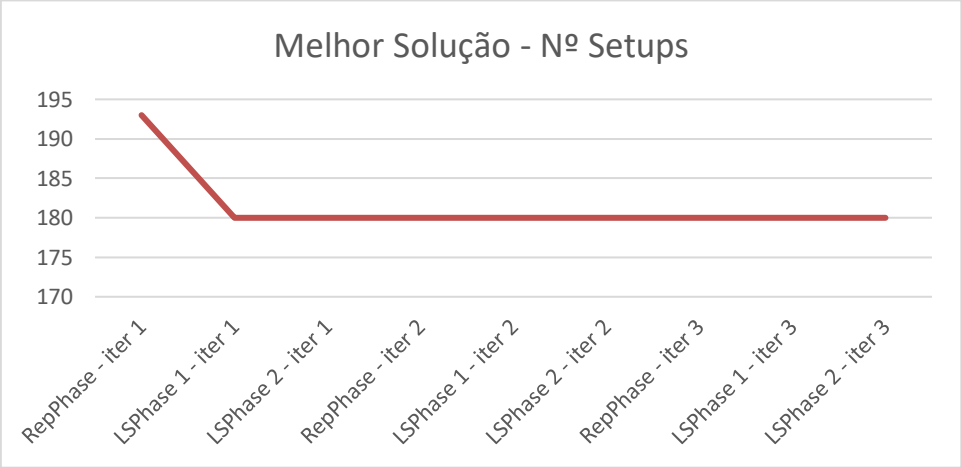


Figura 90 - Comportamento do Nº *Setups* da melhor solução ao longo do GRASP (Instância 6)

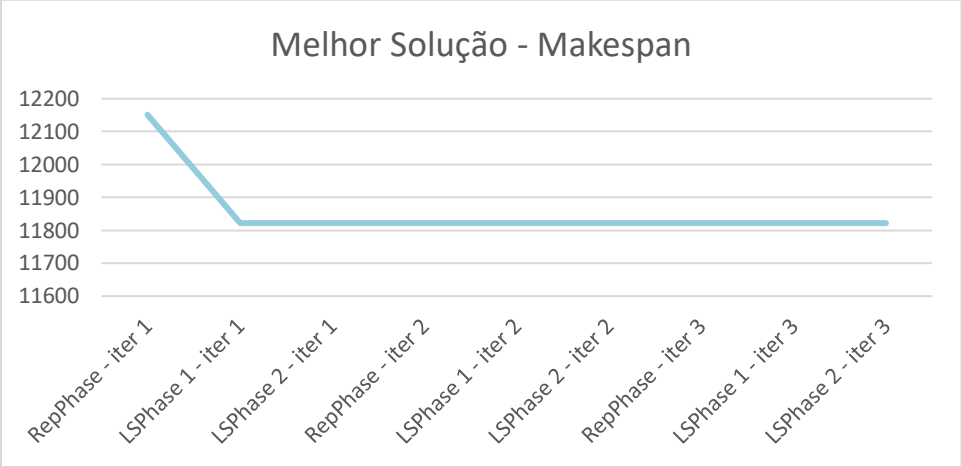


Figura 91 - Comportamento do *Makespan* da melhor solução ao longo do GRASP (Instância 6)

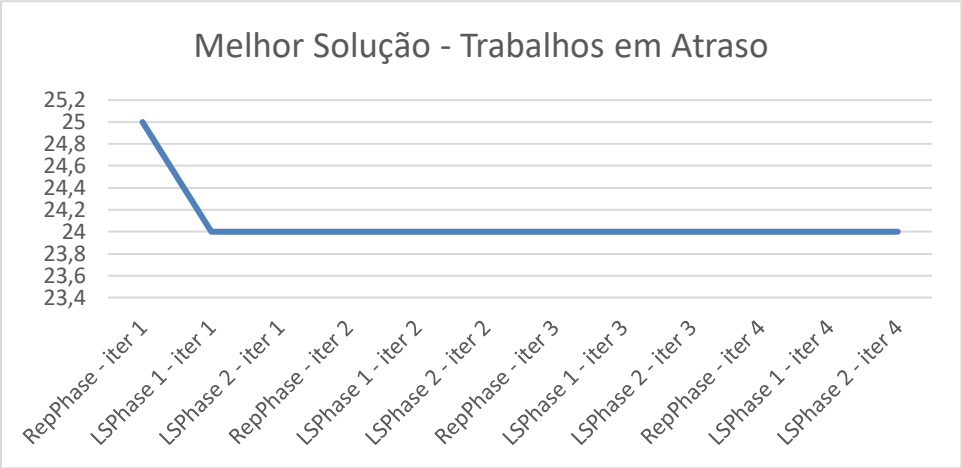


Figura 92 - Comportamento do Nº Trabalhos em Atraso da melhor solução ao longo do GRASP (Instância 7)

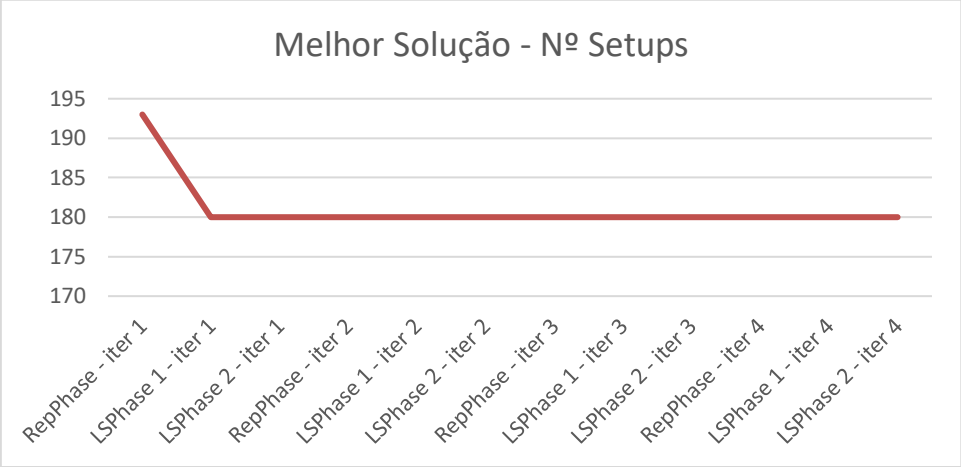


Figura 93 - Comportamento do Nº *Setups* da melhor solução ao longo do GRASP (Instância 7)

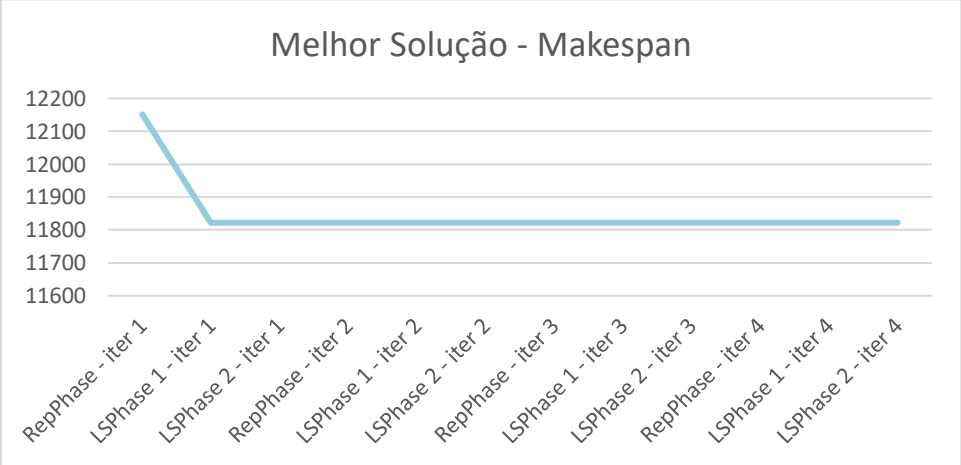


Figura 94 - Comportamento do *Makespan* da melhor solução ao longo do GRASP (Instância 7)