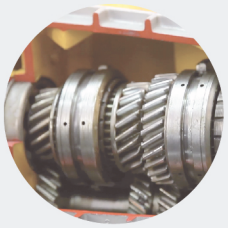
INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO
MESTRADO EM ENGENHARIA E GESTÃO INDUSTRIAL






DESENVOLVIMENTO DE UMA FERRAMENTA DE APOIO À DECISÃO AO ESCALONAMENTO DINÂMICO DA PRODUÇÃO

LUÍS PAULO MAGALHÃES FERREIRINHA
outubro de 2019

POLITÉCNICO
DO PORTO

Instituto Superior de Engenharia do Porto

DESENVOLVIMENTO DE UMA FERRAMENTA DE APOIO À DECISÃO AO ESCALONAMENTO DINÂMICO DA PRODUÇÃO

Luís Paulo Magalhães Ferreirinha

2018/2019

Instituto Superior de Engenharia do Porto

Departamento de Engenharia Mecânica

POLITÉCNICO DO PORTO

Instituto Superior de Engenharia do Porto

DESENVOLVIMENTO DE UMA FERRAMENTA DE APOIO À DECISÃO AO ESCALONAMENTO DINÂMICO DA PRODUÇÃO

Luís Paulo Magalhães Ferreirinha

1160297

Dissertação apresentada ao Instituto Superior de Engenharia do Porto para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Engenharia e Gestão Industrial, realizada sob a orientação do Professor André Borges Guimarães Serra e Santos e coorientação do Professor Doutor João Augusto de Sousa Bastos.

2018/2019

Instituto Superior de Engenharia do Porto

Departamento de Engenharia Mecânica

POLITÉCNICO DO PORTO

JÚRI

Presidente

Doutor Manuel Joaquim Pereira Lopes

Professor Adjunto, Instituo Superior de Engenharia do Porto

Orientador

Mestre André Borges Guimarães Serra e Santos

Assistente Convidado, Instituo Superior de Engenharia do Porto

Co-orientador

Doutor João Augusto de Sousa Bastos

Professor Adjunto, Instituo Superior de Engenharia do Porto

Arguente

Doutora Dorabela Gamboa

Presidente da ESTG, Escola Superior de Tecnologia e Gestão

AGRADECIMENTOS

Foram várias as pessoas que ao longo destes anos tornaram este trabalho possível e a todas elas devo o meu profundo agradecimento.

Quero deixar os meus sinceros agradecimentos ao Professor André Santos e ao Professor Doutor João Bastos pela sua orientação, motivação, disponibilidade, valores transmitidos e inspiração em que se tornaram desde o primeiro momento.

A todos os professores que de alguma forma marcaram a minha aventura pelo Mestrado em Engenharia e Gestão Industrial, com um especial agradecimento à Professora Doutora Maria Teresa Pereira.

À Ângela Pereira, Ivo Camelo, Cristiana Magalhães, Soraia Campelo e Rita Martins pela partilha de conhecimento e pelos momentos de descontração, apoio constante e amizade sincera.

À minha namorada, Sara Baptista, pelo companheirismo na vida, apoio, paciência, amizade e entreajuda ao longo da minha formação académica e pessoal.

À minha irmã e aos meus pais pelo apoio constante, por toda a educação, pela paciência e por sempre acreditarem em mim.

A todos, um eterno obrigado!

PALAVRAS CHAVE

Escalonamento dinâmico da Produção, Ferramenta de apoio à decisão, Meta-Heurísticas, Planeamento de experiências de Taguchi, Modelo de Kano.

RESUMO

A atual exigência e instabilidade do mercado global provocam um grande impacto nos problemas industriais, podendo destacar-se o escalonamento e o sequenciamento de trabalhos. Esta situação implica que mais recursos, para além do fator tempo, sejam considerados críticos, como máquinas, mão-de-obra e instalações. O objetivo geral das organizações prende-se com a satisfação dos clientes, quer em termos de qualidade de produtos quer no cumprimento das datas estabelecidas.

Por norma, os problemas de escalonamento são classificados como problemas de otimização combinatória sujeitos a restrições, com uma natureza dinâmica e de resolução bastante complexa, cujos elementos básicos são as máquinas e as tarefas. Uma maneira de promover a sobrevivência e a sustentabilidade das organizações é o uso eficiente dos seus recursos. Uma falha completa na preparação adequada das tarefas pode facilmente levar a um desperdício de tempo e recursos, o que pode resultar num baixo nível de produtividade e altas perdas monetárias. Diante do exposto, é essencial analisar e desenvolver continuamente novos modelos de programação de produção.

O escalonamento da produção na presença de eventos em tempo real é de grande importância para a implementação bem-sucedida dos sistemas de escalonamento no mundo real. A maioria das empresas opera em ambientes dinâmicos, vulneráveis a vários eventos estocásticos em tempo real, o que força a contínua reconsideração e revisão de programas pré-estabelecidos. Num ambiente incerto, maneiras eficientes de adaptar as soluções atuais a eventos inesperados são preferíveis às soluções ótimas que logo se tornam obsoletas assim que são lançadas para o chão de fábrica. Tal realidade

foi a principal motivação para o desenvolvimento de uma ferramenta de apoio ao escalonamento dinâmico, a qual tenta começar a preencher a lacuna entre a teoria e a prática do escalonamento.

A presente dissertação pretende contribuir para facilitar a compreensão dos problemas e melhorar o processo de escalonamento na indústria, apresentando contribuições no domínio do escalonamento da produção em duas envolventes principais: uma a um nível teórico, conceptual e outra ao nível prático da resolução de problemas através do desenvolvimento de uma ferramenta dinâmica de apoio à decisão. Ao nível conceptual contribui para uma ontologia de problemas de escalonamento da produção e conceitos relacionados, tais como ambiente de escalonamento, características dos trabalhos e dos recursos, critérios de otimização, medidas de desempenho, ferramentas de escalonamento, tipos de escalonamento e técnicas de resolução de problemas combinatórios e sua parametrização, providenciando um enquadramento comum para a compreensão e partilha de conhecimento acerca destes conceitos. Ao nível da resolução de problemas, desenvolveu-se uma ferramenta simples, moderna e autónoma de apoio à decisão ao escalonamento dinâmico onde os critérios de desempenho são classificados através do modelo de Kano. Ou seja, o protótipo desenvolvido simula a sua conexão ao software MRP e usa meta heurísticas para gerar um escalonamento preditivo. Assim, sempre que ocorrem eventos não planeados, como a chegada de novas tarefas ou cancelamento de outras, a ferramenta começa a reagendar através de um módulo de eventos dinâmicos que combina regras de despacho que melhor se ajustam às medidas de desempenho pré-classificadas pelo modelo de Kano. A ferramenta proposta foi testada em um estudo computacional aprofundado com entradas dinâmicas de tarefas com tempos de execução estocásticos. Além disso, foi realizada uma análise mais robusta, que demonstrou que há evidência estatística de que o desempenho do protótipo proposto é melhor que o método único de programação e comprovou a eficácia do mesmo.

O conceito da presente dissertação já deu origem a três publicações em revistas indexadas, o que motivou ainda mais ao desenvolvimento e aperfeiçoamento do sistema desenvolvido (L. Ferreirinha et al., 2019; Luis Ferreirinha et al., 2019; Luís Ferreirinha et al., 2020).

KEYWORDS

Dynamic Production Scheduling, Decision Support Tool, Meta-Heuristics, Design of Experiments of Taguchi, Kano's Model.

ABSTRACT

The current demand and instability of the global market have a major impact on industrial problems, including the staggering and sequencing of jobs. This situation means that more resources than time are considered critical, such as machines, labor and facilities. The overall goal of organizations is customer satisfaction, both in terms of product quality and meeting set dates.

Usually, scheduling problems are classified as constrained combinatorial optimization problems, with a dynamic nature and very complex resolution, whose basic elements are machines and tasks. One way to promote the survival and sustainability of organizations is the efficient use of their resources. A complete failure to properly prepare tasks can easily lead to a waste of time and resources, which can result in low productivity and high monetary losses. Given the above, it is essential to continuously analyze and develop new production scheduling models.

Production scheduling in the presence of real-time events is of great importance for the successful implementation of real-world scheduling systems. Most companies operate in dynamic environments that are vulnerable to multiple stochastic events in real time, which forces continuous reconsideration and review of pre-established programs. In an uncertain environment, efficient ways to adapt current solutions to unexpected events are preferable to optimal solutions that soon become obsolete as they are dropped to the shop floor. Such reality was the main motivation for the development of a dynamic scheduling support tool, which attempts to begin to bridge the gap between scheduling theory and practice.

This dissertation aims to contribute to facilitate the understanding of the problems and improve the process of scheduling in the industry, presenting contributions in the field of production scheduling in two main environments: one at a theoretical, conceptual level, and another at the practical level of problem solving through the development of a dynamic decision support tool. At the conceptual level it contributes to an ontology of production scheduling problems and related concepts such as scheduling environment, job and resource characteristics, optimization criteria, performance measures, scheduling tools, scheduling types, and resolution techniques regarding combinatorial problems and their parameterization, providing a common framework for understanding and sharing knowledge about these concepts. In terms of problem solving, a simple, modern and autonomous dynamic scheduling decision support tool has been developed where performance criteria are classified through Kano's model. This is, the developed prototype simulates its connection to MRP software and uses metaheuristics to generate predictive schedules. Thus, whenever unplanned events such as new tasks arrive or others are canceled, the tool begins to reschedule through a dynamic event module that combines dispatch rules that best fit the performance measures pre-classified by the Kano's model. The proposed tool was tested in an in-depth computational study with dynamic task inputs with stochastic runtimes. In addition, a more robust analysis was performed, which showed that there is statistical evidence that the performance of the proposed prototype is better than the single programming method and proved its effectiveness.

The concept of this dissertation has already given rise to three publications in indexed journals, which further motivated the development and improvement of the developed system (L. Ferreirinha et al., 2019; Luis Ferreirinha et al., 2019; Luís Ferreirinha et al., 2020).

LISTA DE SÍMBOLOS E ABREVIATURAS

ABC	Artificial bee colony
ACO	Ant colony optimization
BD	Base de dados
CO	Critério de otimização
DOE	Design of experiments
DER	Diagrama entidade relação
EDD	Earliest due date
FA	Fonte de alimento
FO	Função objetivo
GA	Genetic algorithm
GLS	Guided local search
ILS	Iterative Local Search
Ln	Estado de equilíbrio
LPT	Longest Processing Time
LS	Local search
MA	Movimentação aleatória de tarefas
MRP	Material Requirement Planning
OC	Otimização combinatória
PL	Pesquisa local
PSO	Particle swarm optimization
QAP	Quadratic assignment problems
SA	Simulated Annealing
TA	Troca de tarefas aleatoriamente
Tf	Temperatura final
Ti	Temperatura inicial
TMF	Tempo médio de fluxo
TS	Tabu search
TSP	Problema do caixeiro viajante
TTA	Troca de tarefas adjacentes
WIP	Work In Progress

GLOSSÁRIO DE TERMOS

batch(b)	produção por lote
brkdwn	Indisponibilidade da máquina
Cj	Instante de conclusão do trabalho j.
Cmax	Instante máximo de conclusão dos trabalhos ou Makespan.
dj	data de entrega da tarefa j
Ej	Antecipação de uma tarefa j
Emax	Antecipação máxima de uma sequência
FFSm	Flexible flow shop
Fj	Tempo de fluxo da tarefa j
Fm	Flow shop
fm/s	família de tarefas
Jm	Job shop
Lj	Atraso ou antecipação da tarefa j
Lmax	Atraso ou antecipação máxima de uma sequência
NP	Classe de problemas NP
nwt	sem espera
Om	Oficina aberta
pij	Tempo de processamento do trabalho j na máquina i
Pm	Máquinas paralelas idênticas
prec	Relações de precedência
prmp	interrupções
prmu	permutações
Qm	Máquinas paralelas uniformes
rcrc	recirculação
rj	data de lançamento de uma tarefa j
Rm	Máquinas paralelas não relacionadas
sij	Tempo de setup de uma tarefa j na máquina i
T	Atraso de uma tarefa j em relação ao dj
Tmax	Atraso máximo de uma sequência
wj	peso de uma tarefa j
ΣCj	Somatório dos instantes de conclusão dos trabalhos

ÍNDICE DE FIGURAS

ÍNDICE DE TABELAS

ÍNDICE

INTRODUÇÃO

1 INTRODUÇÃO

Está na natureza de qualquer organização a constante procura por melhorias e maiores rentabilidades dos seus processos produtivos. Dada a globalização do mercado, o mesmo tem vindo a tornar-se cada vez mais competitivo, impondo a redução de custos e melhores níveis de produtividade e qualidade. A globalização da economia e a agilidade tecnológica impôs-se como forma de mobilizar as organizações para a obtenção do grau máximo de competitividade, modernidade e qualidade, de modo a assegurarem a sua sobrevivência e o crescimento sustentável estabelecida pela nova formação económica da sociedade.

Tal cenário provoca uma diminuição nos ciclos de vida dos produtos o que leva a que a indústria sofra cada vez mais com flutuações nos seus produtos, os prazos de entrega sejam cada vez mais apertados e que as encomendas, embora mais frequentes, sejam de uma menor volumetria e com uma constante alteração das especificações, mesmo durante a produção (Artiba & Elmaghraby, 1996).

As exigências de celeridade e variedade do cliente provocam um grande impacto nos problemas industriais, podendo destacar-se o escalonamento e o sequenciamento de trabalhos. Este cenário implica que mais recursos, para além do fator tempo, sejam considerados críticos, como máquinas, mão-de-obra e instalações (Sule, 2007).

O escalonamento da produção afeta todos os ramos quer das organizações industriais, quer dos serviços, sendo que no contexto industrial tem normalmente implícito o objetivo da satisfação dos clientes, principalmente quanto às datas de entrega estabelecidas. Contudo, uma vez que o ambiente de produção é dinâmico, eventos inesperados podem ocorrer a qualquer momento, o que revela a importância da rapidez de soluções que requerem uma agilidade e flexibilidade numa parte da produção (Artiba & Elmaghraby, 1996).

Torna-se, então, necessário estudar métodos que sejam rápidos e flexíveis para a resolução destes problemas, integrando uma análise que não contemple, apensas, a satisfação dos interesses do cliente como também os interesses da empresa. A capacidade de análise de várias medidas de desempenho em simultâneo, levam a um maior balanceamento entre os interesses de ambas as partes, evitando tanto atrasos nas encomendas como custos desnecessários de armazenamento (Shabtay & Steiner, 2008). Por norma, os problemas de escalonamento são classificados como problemas de otimização combinatória sujeitos a restrições, com uma natureza dinâmica e de resolução bastante complexa, cujos elementos básicos são as máquinas e as tarefas. O escalonamento visa a afetação dos recursos às atividades e determina em que sequência devem ser executadas, de modo a otimizar uma dada medida de desempenho (Pereira, 2009; Maria Leonilde Rocha Varela, 2007).

O tema de otimização combinatória consiste num conjunto de problemas fulcrais para as áreas da ciência da computação e engenharia. A investigação nesta área visa

desenvolver técnicas eficientes para encontrar valores mínimos ou máximos de uma função (objetivo) constituída por diversas variáveis independentes (Kirkpatrick, Gelatt, & Vecchi, 1983).

Os problemas de otimização, os quais incluem os problemas de otimização combinatória, podem ser divididos em várias categorias, dependendo se são contínuos ou discretos, restritos ou não, mono ou multi objetivos, estáticos ou dinâmicos. De entre as diversas técnicas de resolução de problemas de otimização combinatória, onde se inclui o problema de escalonamento, destacam-se os métodos de aproximação, cujos objetivos passam por encontrar soluções satisfatórias em tempos de execução aceitáveis. Dentro destes métodos aproximativos encontram-se as Meta Heurísticas, i.e., métodos que pesquisam o espaço de soluções de um determinado problema com o intuito de encontrar soluções o mais próximas possíveis do ótimo global, de preferência ao ótimo global. A meta-heurística é um algoritmo projetado para resolver aproximadamente uma ampla gama de problemas de otimização difíceis sem ter que se adaptar profundamente a cada problema. De facto, o prefixo grego ''meta'', presente no nome, é usado para indicar que esses algoritmos são heurísticas de 'nível mais alto', em contraste com as heurísticas específicas do problema. As meta heurísticas são geralmente aplicadas a problemas para os quais não há um algoritmo satisfatório específico para resolvê-los em tempo polinomial. São amplamente utilizadas para resolver problemas complexos na indústria e nos serviços, em áreas que vão desde finanças a gestão de produção e engenharia (Boussaïd, Lepagnot, & Siarry, 2013).

Contudo, a aplicação destes métodos requere um conhecimento prévio dos parâmetros mais adequados ao problema a tratar, o que nem sempre é fácil e normalmente requer a utilização de alguns métodos de parametrização, como por exemplo Design of Experiments (DOE) (Pereira, 2009).

O escalonamento dinâmico nas indústrias pode ser extremamente difícil. A maioria dos sistemas industriais opera em ambientes dinâmicos vulneráveis a vários eventos estocásticos em tempo real, como quebras de máquinas, pedidos urgentes, material indisponível e muitos outros que tornam facilmente os pré-escalonamento obsoletos (Ana Madureira, Ramos, & Silva, 2003; O'Donovan, Uzsoy, & McKay, 1999; Ihsan Sabuncuoglu & Karabuk, 1999). Tais indústrias geram e atualizam os programas de produção, que são planos que servem de base para muitas atividades externas (por exemplo, o fluxo de materiais, manutenção preventiva). Num ambiente incerto, as indústrias podem beneficiar de uma melhor compreensão de como as estratégias de (re)escalonamento afetam o desempenho do sistema, pois nesse ambiente o foco do gestor passa por encontrar maneiras eficientes e eficazes de adaptar as soluções atuais a eventos inesperados, ao invés de encontrar uma programação de alta qualidade que se rapidamente se torna obsoleta assim que é liberta para o chão de fábrica. Na prática, o reescalonamento é feito de maneira híbrida, ou seja, de uma forma periódica onde os planos para o próximo período são definidos com base no status atual do sistema ou,

ocasionalmente, em resposta a eventos não planeados (O'Donovan et al., 1999; I Sabuncuoglu & Bayız, 2000; Vieira, Herrmann, & Lin, 2003).

Quando se aborda escalonamento em ambientes dinâmicos, destacam-se dois elementos principais: a geração do programa, que atua como um mecanismo preditivo e serve como um plano geral para outras atividades da empresa, e a monitoração / atualização do programa, que é vista como a parte reativa do sistema o qual tenta minimizar o efeito das perturbações no desempenho do sistema (I Sabuncuoglu & Bayız, 2000; M. L. R. Varela & Ribeiro, 2014; Vieira et al., 2003).

Em vista do exposto, a presente dissertação propõe uma ferramenta de apoio à decisão para ambientes dinâmicos que tenta preencher a lacuna entre a teoria e a prática do escalonamento declarada pelo autor em (Cowling & Johansson, 2002). Do presente relatório resulta um protótipo que não requer interação com o utilizador para além da definição dos critérios de desempenho. Os critérios de desempenho são classificados pelo grau de satisfação do Modelo de Kano, que classifica as medidas de desempenho e equilibra os interesses das partes interessadas. A ferramenta também simula a conexão ao MRP (Material Requirement Planning) da empresa e, sempre o MRP lança trabalhos no sistema, uma meta heurística é usada para gerar um novo plano. Quando ocorrem eventos não planeados, a ferramenta começa a reagendar as tarefas de acordo com as regras de prioridade que melhor se ajustam ao desempenho pretendido. Para validar a ferramenta, várias instâncias compostas por um vasto conjunto de tarefas com características estocásticas normalmente distribuídas foram executadas. De seguida, foi realizada uma análise estatística para encontrar evidências da eficácia do protótipo.

1.1 Objetivo

Como objetivo inicial pretende-se realizar um levantamento e descrição de problemas e respetivo métodos de escalonamento de produção, dando especial ênfase a regras de prioridade e meta heurísticas.

Posteriormente, o objetivo principal consiste em desenvolver uma ferramenta de apoio à decisão ao escalonamento dinâmico na produção, independentemente do ambiente máquina. A principal ferramenta poderá ser conectada ao MRP da empresa e escalonar de forma autónoma as diversas tarefas, presentes em todos os postos de trabalho em curso, sempre ocorra um qualquer evento não planeado ou, quando novas tarefas entrem no sistema. Tais escalonamentos autónomos devem visar certas medidas de desempenho, medidas essas classificadas pelo modelo de Kano.

Tem-se assim por objetivos:

- Identificação clara de problemas de escalonamento da produção;
- Aquisição e pesquisa de conhecimento, acerca de métodos de escalonamento dinâmico da produção;
- Desenvolvimento da ferramenta e aplicação de métodos de escalonamento para a produção;

- Análise comparativa das medidas de desempenho definidas no Modelo de Kano entre do modelo desenvolvido e um modelo de método único de escalonamento de produção.

1.2 Metodologia

No sentido de cumprir os objetivos delineados inicialmente para o presente relatório, torna-se essencial definir a metodologia a ser seguida. Considera-se como melhor método de abordagem o uso da metodologia dedutiva e indutiva, associada a uma lógica que permita a pesquisa de informação e tratamento da mesma voltando de seguida a pesquisa de dados para preencher as lacunas verificadas no desenvolvimento dos capítulos teóricos. A pesquisa bibliográfica deste projeto engloba os problemas de escalonamento da produção, características dos trabalhos e dos recursos produtivos, critérios e técnicas de otimização, medidas de desempenho e regras de sequenciamento, problemas de otimização combinatória, o conceito de meta-heurística e suas características, alguns algoritmos meta-heurísticos e ainda métodos de parametrização. Toda a informação que está relacionada com estes temas foi obtida através de dados e documentos cedidos por fontes como artigos, livros e informações credíveis através da internet referenciados no presente relatório.

Para validar a ferramenta, instâncias compostas por um vasto conjunto de tarefas com características estocásticas normalmente distribuídas foram executadas. Posteriormente, recorreu-se a testes paramétricos para evidenciar estatisticamente o desempenho do modelo.

1.3 Estrutura

A presente dissertação encontra-se estruturada em seis capítulos, sendo que, numa primeira parte é feito o enquadramento temático, definido os objetivos a atingir, descrita a metodologia de abordagem adotada e descrita a presente estrutura. A revisão bibliográfica é constituída pela segunda e terceira parte na qual é apresentada toda a revisão de literatura relativa problemas de escalonamento e técnicas de resolução do mesmo, respetivamente. Na quarta parte apresenta-se a ferramenta desenvolvida de forma detalhada, isto é, é apresentando o seu conceito, descrita a sua funcionalidade e modo de operação. Na quinta parte é apresentado o estudo computacional efetuado com a ferramenta desenvolvida e discutidos os resultados encontrados. Por fim, na sexta parte, são apresentadas as considerações finais e as conclusões a retirar do trabalho realizado, bem como as perspetivas de trabalho futuro.

ESCALONAMENTO DA PRODUÇÃO

2 Escalonamento da Produção

Neste capítulo serão abordados vários temas inerentes ao problema do escalonamento. Antes de analisar os sistemas de escalonamento existentes, faz sentido perceber as abordagens que até então têm vindo a ser utilizadas ao longo dos anos (Herrmann, 2006). Posto isto, o capítulo abordará com algum detalhe os elementos que constituem os problemas de escalonamento, desde a sua nomenclatura à sua implementação industrial. Posteriormente, serão apresentadas as restrições e objetivos que se encontram neste tipo de problemas bem como abordagens para a resolução dos mesmos. Finalmente, dar-se-á a conhecer medidas de desempenho relevantes a cada ambiente industrial bem como a complexidade subjacente a estes problemas.

2.1 Enquadramento Histórico

Com a revolução industrial o processo produtivo pouco especializado e até mesmo artesanal foi tomado por indústrias com estruturas organizacionais mais detalhadas. Contudo, as empresas pioneiras eram simples e relativamente pequenas, onde a quantidade de produtos produzidos era significativamente superior à diversidade dos mesmos. Tais empresas responsabilizavam os chefes de fábrica pela coordenação de todas as atividades necessárias ao desenvolvimento dos produtos, desde a contratação de funcionários, aquisição de matérias primas, planeamento, controlo do processo de fabrico e até mesmo da expedição do produto final (Herrmann, 2006).

Herrmann, J. W. (2006) indica ainda que no século XIX, o planeamento da produção não disponibilizava qualquer informação acerca de quanto tempo uma ordem de produção deveria demorar, ou até sobre o tempo necessário de cada operação. Quando utilizado, o planeamento apenas indicava quando é que um trabalho deveria começar ou até quando deveria estar pronto.

Pouco antes do início do século XX, as empresas começaram a atender a uma grande variedade de produtos, provocando um incremento significativo na complexidade do planeamento da produção. Por volta do tempo da 1ª guerra mundial, Frederick Taylor propõe uma separação de funções de planeamento e de produção, com recurso à criação de um departamento dedicado não só ao planeamento e controlo de operações, como também a algumas atividades logísticas, como por exemplo a gestão de inventário. Henry Gantt, por volta de 1916, propõe a entrega de ordens de trabalho com informações sobre as atividades que devem ser realizadas naquele dia. Várias foram as empresas que adotaram a proposta de Taylor, contudo, o chefe de fábrica continuava a possuir um papel importante no escalonamento da produção, decidindo, em conjunto com o responsável pelo planeamento, qual o trabalho que deveria ser efetuado e de que forma é que este último deveria ser afetado aos trabalhadores (Herrmann, 2006).

Por volta de 1940 começaram a surgir modelos de programação linear aplicados ao planeamento da produção onde aparece por, George Dantzig, o modelo *simplex* para a resolução de problemas de programação linear. Em meados do século XX, motivada

pelos problemas de escalonamento da produção, a investigação conduziu ao desenvolvimento de alguns algoritmos importantes, como por exemplo o algoritmo de Johnson para duas máquinas em *flowshop*, as regras de prioridade *earliest due date* (EDD), para a minimização dos atrasos máximos, e *shortest processing time* (SPT), para a minimização do tempo médio de fluxo. Posteriormente, em 1960, começaram a surgir técnicas de otimização, como o *Branch-and-Bound*, e técnicas de geração de colunas para a resolução de problemas de programação inteira. Em 1970, a teoria da complexidade mostra a dificuldade de alguns problemas de escalonamento, tornando a procura pela solução ótima em tempo útil muito complexa. Daí surgirem métodos mais eficientes para procura de solução perto da ótima em curto espaço de tempo, tais como a pesquisa local, *simulated annealing, tabu search*, entre outras (Herrmann, 2006).

O desenvolvimento de novas tecnologias teve uma grande importância na história do escalonamento da produção. As tecnologias de informação possibilitaram a execução de algoritmos de escalonamento de forma autónoma, contudo tornava-se necessário desenvolver melhores algoritmos para a resolução de problemas de escalonamento mais complexos (Herrmann, 2006).

Atualmente os problemas de escalonamento são cada vez mais alvo de investigação não só pelo mundo académico como pelo industrial. Vários autores adotam ou, até mesmo, adaptam diversas técnicas para a resolução deste tipo de problemas às suas necessidades. Posteriormente publicam não só os resultados encontrados bem como as vantagens e dificuldades confrontadas, inspirando outros autores na procura de novas e eficientes abordagens.

2.2 Problema de escalonamento

O planeamento e escalonamento da produção são problemas difíceis com um longo histórico nas áreas de investigação operacional e inteligência artificial, e continuam a ser áreas ativas de investigação (Mesghouni, Hammadi, & Borne, 1996).

Sendo uma das principais tarefas de gestão da produção por parte das indústrias, o escalonamento da produção tem sido um grande alvo de investigação, onde muito esforço tem sido dedicado ao desenvolvimento de algoritmos e métodos eficazes e mais eficientes para a resolução dos problemas inerentes a esta temática (Liu & Yan, 2012).

O escalonamento da produção (EP) implica a definição de um momento inicial e final do processamento de cada trabalho de um determinado conjunto e a sua alocação aos recursos, cumprindo determinadas restrições, que podem envolver os trabalhos e/ou recursos. O objetivo do escalonamento consiste em otimizar uma determinada medida de desempenho de caráter económico e operacional (Artiba & Elmaghraby, 1996; Ana Madureira et al., 2003).

Também designada por programação da produção, esta tarefa encontra-se no nível mais abaixo da função planeamento e controlo da produção de um sistema produtivo. Ou

seja, esta função é inerente ao chão de fábrica, trabalhando com um grande nível de detalhe e com um carácter de curto prazo, como demonstra a Figura 1.

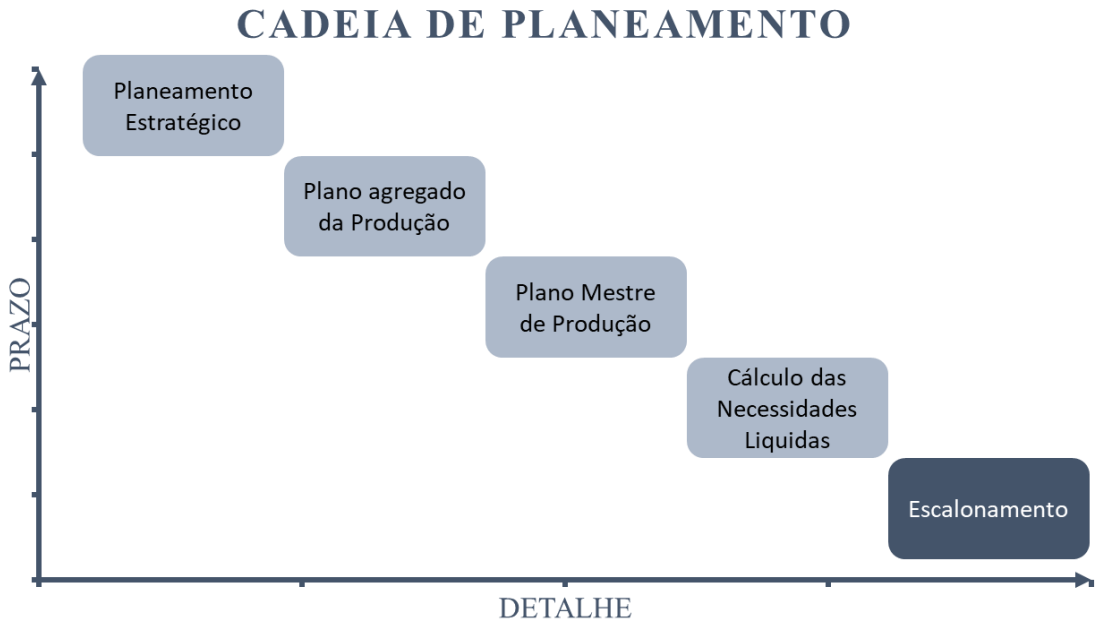


Figura 1 Posicionamento do escalonamento na cadeia de planeamento

A correta resolução de um problema de escalonamento implica o conhecimento em detalhe das tarefas que devem ser executadas bem como os recursos disponíveis para a execução de tais tarefas. Assim, os problemas de escalonamento visam responder a (Baker & Trietsch, 2009):

- Quais recursos a serem utilizados para executar determinada tarefa?
- Qual o momento em que cada tarefa deve ser executada?

Nesta perspetiva, a primeira pergunta permite definir a afetação das operações e a segunda permite definir a sequenciação das operações (Pinedo, 2012), temas esses que serão de seguida abordados.

2.2.1 Problemas de afetação

Em ambiente industrial é possível encontrar vários recursos capazes de executar as mesmas tarefas, pelo que se torna necessário tomar uma decisão relativamente à atribuição de tais recursos a tais tarefas. No entanto, existem *n* tarefas pertencentes a um conjunto *T ={T1, T2,…, Tn}*, *m* máquinas pertencentes a *P={P1, P2,…, Pm}* e *s* recursos adicionais pertencentes a *R = {R1, R2,…, R3}*, e a pergunta que se levanta é, de que forma é que as máquinas *P* e os recursos *R* devem ser afetados às tarefas *T* de modo a otimizar uma determinada medida de desempenho (A. B. G. S. e Santos, 2015). Tal problema de decisão é designado por problema de afetação, onde é necessário determinar que recursos executarão que tarefas de modo a maximizar/minimizar uma ou várias medidas de desempenho (Blazewicz, Ecker, Pesch, Schmidt, & Weglarz, 2007). Na Figura 2 encontra-se uma representação básica deste tipo de problema.

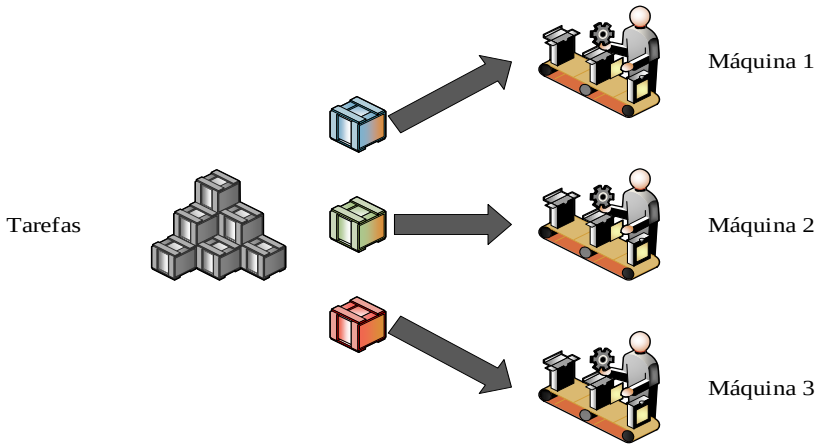


Figura 2 Problema de Afetação

2.2.2 Problemas de Calendarização/ sequenciação

É muito frequente integrar a atividade de afetação com a atividade de sequenciamento na programação da produção. Por norma, estes tipos de problemas sucedem a afetação, mas em muitos cenários os mesmos reduzem-se apenas ao sequenciamento. Neste problema de decisão o objetivo é distribuir as tarefas num intervalo de tempo de modo a otimizar uma determinada medida de desempenho (Baker & Trietsch, 2009; Maria Leonilde Rocha Varela, 2007).

Em Santos (2015) é percetível a diferença entre calendarização e sequenciação, onde a primeira determina o momento em que as tarefas devem ser executadas e a segunda define uma sequência de execução de tarefas. Portanto, dado um conjunto de tarefas *T = {T1, T2, …, Tn}*, a sequência *S = {S1, S2, …, Sm}* e/ ou o intervalo de tempo [a, b], o objetivo deste problema é sequenciar as tarefas *T* em *S* de modo a maximizar/minimizar uma ou várias medidas de desempenho. Aparentemente a resolução destes problemas pode parecer simples, no entanto, à medida que o conjunto *T* aumenta, estes problemas tornam-se cada vez mais complexos dada a sua natureza combinatória (A. B. G. S. e Santos, 2015). Na Figura 3 encontra-se uma representação básica deste tipo de problema.

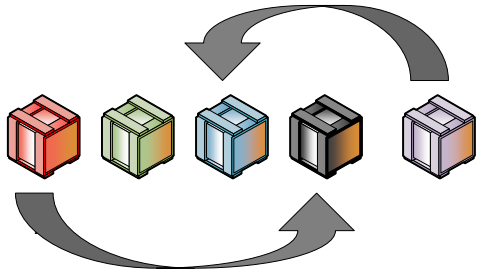


Figura 3 Problemas de Sequenciação

2.3 Nomenclatura - Pinedo

Dada a popularidade e dimensão dos problemas de escalonamento da produção, houve a necessidade de desenvolver sistemas de representação capazes de representar de forma simples o modelo em questão, sistemas esses inicialmente introduzidos por Graham et al., (1979). Existem diversas formas de representação na literatura científica, como a nomenclatura de Pinedo (2012), Brucker (1995) e Blazewicz et al., (2007), semelhantes entre si e que permitem descrever o problema com detalhe.

Em todos os problemas de escalonamento, considera-se que o número de tarefas e máquinas são finitos. Pinedo, M. L. (2012) representa o número de tarefas por *n,* e por *m* o número de máquinas. Por sua vez, discrimina cada tarefa através de um índice *j,* e cada máquina através de um índice *i*. Tal representação permite associar a tarefa j à máquina i. Assim, é possível representar mais características das tarefas da seguinte forma (Pinedo, 2012):

- **Data de entrega** *(d_j)* – Representa uma data limite de entrega de uma tarefa *j*. Dependendo do acordado com o cliente, o seu incumprimento pode levar a penalizações para a organização.
- **Peso** *(w_j)* – Representa o peso da tarefa *j* no sistema produtivo relativamente a outras tarefas. Este parâmetro define a relevância de uma tarefa, numa medida de desempenho, ou pode não existir quando todas as tarefas apresentam o mesmo peso;
- **Tempo de processamento** *(p_{ij})* – representa o tempo de processamento da tarefa *j* na máquina *i*. O índice *i* pode ser omitido caso o tempo de processamento da tarefa *j* não depende da máquina onde vai ser processada.

Pinedo, M. L. (2012) descreve os problemas de escalonamento em três campos α| β| γ. Nesta representação α descreve como estão dispostos os recursos afetos à produção, isto é, o ambiente de produção, β descreve as restrições do processo e γ descreve o critério de otimização, ou seja, a medida de desempenho que se pretende maximizar/minimizar. Dado que é esta a nomenclatura que será adotada no presente relatório, os próximos pontos serão abordados com algum detalhe.

Tal como se referiu, existem outro tipo de representações, recomenda-se a leitura de Varela, M. (2007) onde várias nomenclaturas são apresentadas e discutidas com grande detalhe e até onde uma proposta de nomenclatura é apresentada.

2.3.1 Ambientes de Produção- Campo α

Máquina Única (1)

O escalonamento da produção em máquinas únicas é, fundamentalmente, um problema de sequenciação/ calendarização, e reside na existência de um único recurso, ou máquina. Trata-se de um problema de análise combinatória onde as várias tarefas em fila de espera permutam entre si por forma a otimizar uma determinada medida de desempenho. Apesar da sua aparente simplicidade, este tipo de problemas é frequentemente utilizado na resolução de sub-problemas de implementações mais complexas onde mais tarde se incorpora o resultado no problema maior (Baker & Trietsch, 2009). Este recurso de máquina única pode em muitas circunstâncias estar inserido em implantações como linhas de produção ou oficinas, onde este recurso se torna numa situação de estrangulamento que afeta todo o sistema simplificando a resolução de problemas deste tipo (A. Madureira, Pereira, Pereira, & Abraham, 2014).

Máquinas paralelas (Pm/Qm/Rm)

Na indústria é muito comum encontrar várias máquinas capazes de executar um determinado número de trabalhos, a isto denomina-se por máquinas paralelas. O escalonamento em máquinas paralelas representa um problema de escalonamento completo, isto é, numa primeira instância reside na afetação e posteriormente na sequenciação das tarefas. Ou seja, os trabalhos são afetados às máquinas ou recursos e de seguida são sequenciados, reduzindo-se assim a vários problemas de máquina única (Lopez & Roubellat, 2008; Pinedo, 2012).

Dentro das máquinas paralelas, dependendo da sua velocidade de processamento das tarefas, pode-se distinguir três tipos de máquinas diferentes (Leung, 2004; Pinedo, 2012):

- **Máquinas paralelas idênticas (P_m):** quando os tempos de processamento das tarefas são idênticos para todas as máquinas;
- **Máquinas paralelas uniformes (Q_m):** diferentes velocidades de processamento das tarefas, mas a velocidade de cada máquina é constante e independente da tarefa a processar;
- **Máquinas paralelas não relacionadas (R_m):** quando apresentam características distintas, isto é, velocidade de processamento dependente da tarefa particular a ser executada.

Linhas de Fabrico ou *Flow Shop* (Fm)

Por vezes as empresas possuem no seu chão de fábrica um conjunto de máquinas dispostas em série onde *n* tarefas vão sendo executadas sequencialmente em *m* máquinas. Neste cenário, as linhas de fabrico compreendem apenas decisões de sequenciação, pois basta determinar uma sequência para as tarefas que a solução é obtida. No entanto, torna-se relevante destacar que se considera uma linha de fabrico

puramente sequencial aquela onde a sequência de tarefas atribuída à primeira máquina do sistema não possa ser alterada entre máquinas. Tal cenário ocorre quando não existem armazéns intermédios entre as *m* máquinas.

No entanto, nem sempre é possível obter um fluxo contínuo do material em curso, e a armazenagem de tarefas entre máquinas é necessária. Nestes cenários, a sequenciação das tarefas que aguardam pela operação seguinte, pode ser alterada (Baker & Trietsch, 2009; Pinedo, 2012; A. B. G. S. e Santos, 2015).

- **Linhas de Fabrico Flexíveis ou Flexible Flow Shop (FFSm):** Embora as linhas de fabrico sejam comummente tratadas como problemas de sequenciamento, existem linhas de fabrico que, à semelhança das máquinas paralelas, incluem problemas de afetação. Tais linhas são designadas por linhas de fabrico flexíveis, onde as mesmas são constituídas por várias máquinas em paralelo em cada nível. Isto é, as tarefas não necessitam de percorrer por completo uma linha, as mesmas podem deslocar-se entre níveis por forma a serem executadas por várias máquinas diferentes (A. B. G. S. e Santos, 2015).

Oficina de Fabrico ou Job Shop (J_m)

Este tipo de implantação é muito comum em empresas que produzam de um modo descontínuo em pequenas séries, onde os produtos são personalizados consoantes as especificações do cliente. Isto é, ao contrário das linhas de fabrico, onde os materiais possuem todos a mesma rota, nas oficinas de fabrico cada material possui o seu próprio fluxo produtivo. Tal significa que as tarefas não necessitam de percorrer todas as máquinas da oficina, mas apenas aquelas que acrescentarão valor ao produto final.

Normalmente cada tarefa visita uma máquina uma única vez em todo o seu processo produtivo, no entanto, poderão existir cenários onde uma tarefa necessita de visitar mais que uma vez um determinado recurso. Neste caso, o tipo de implantação designa-se de oficina de fabrico com recirculação (Baker & Trietsch, 2009; Pinedo, 2012).

Uma vez que o fluxo que cada tarefa tem de percorrer no chão de fábrica é conhecido à priori, é possível planear um método eficiente de utilizar os diversos recursos. No entanto, uma vez que cada tarefa possui a sua própria rota, está-se perante um problema de escalonamento completo, onde é necessária a afetação dos recursos às tarefas bem como o sequenciamento das mesmas.

Oficinas abertas (O_m)

Como se referiu no ponto anterior, nas implantações em oficinas de trabalho, cada tarefa percorre a sua própria rota de fabrico e, por norma, essa rota é conhecida à priori. No entanto, é comum encontrar cenários onde não existam rotas estandardizadas para as tarefas, e cabe ao planeador tomar tal decisão. A este tipo de implantação designa-se de oficinas abertas- *open shops*, onde a complexidade é acrescida pelo facto de ser

necessário definir um plano de operações para cada tarefa, além dos problemas de afetação e sequenciação (Pinedo, 2012).

2.3.2 Restrições- Campo β

Na representação de Pinedo (2012) as especificações associadas ao processo de fabrico são descritas no campo β. Neste campo, é possível encontrar:

- **Data de lançamento** (r_j)- Representa a impossibilidade da tarefa *j* iniciar o seu processamento antes da data r_j. Caso esta restrição não exista em β, possibilita que que todos os trabalhos, assim que chegam ao sistema, podem começar de imediato a serem processados, isto é, estão todos disponíveis no instante de tempo *t=0*;
- **Interrupções** (*prmp*)- Representa a possibilidade de interromper o processamento de uma tarefa antes de a mesma estar concluída. O trabalho interrompido não é perdido, o que possibilita a conclusão da respetiva tarefa à posteriori;
- **Relações de precedência** (*prec*)- Representa a existência de relações de precedência entre a execução de tarefas. Uma tarefa *A* que possui uma relação de precedência sobre outra, *B*, terá de ser processada completamente antes da tarefa que a sucede. Neste exemplo, *A* é processado antes de *B*;
- **Tempos de *setup* dependentes da sequência** (s_{jk})- Representa a existência de tempos de preparação dependentes da sequência em que as tarefas são processadas. Ou seja, s_{jk} representa o tempo de preparação necessário entre a execução das tarefas *j* e *k*. Caso *k* seja o primeiro trabalho na sequência, denota-se o tempo de preparação como s_{0k}. Caso *j* seja o último elemento da sequência, denota-se como s_{j0} o tempo de preparação final após o trabalho j. Se s_{jk} não existir no existir em β, assume-se que todos os tempos de preparação são zero ou que são independentes da sequência, o que neste caso podem ser incluídos nos tempos de processamento dos trabalhos;
- **Família de tarefas** (*fm/s*) - Representa a existência de famílias de tarefas. Dentro de cada família, as tarefas podem ter tempos de processamento diferentes, mas não necessitam de tempo de preparação entre a execução das mesmas;
- **Produção por lote** (*batch(b)*) – Representa a possibilidade de mais que uma tarefa possa, simultaneamente, ser processada numa máquina. Ou seja, a máquina poderá executar *b* tarefas em simultâneo, e apenas estarão concluídas quando a tarefa com maior tempo de processamento terminar;
- **Indisponibilidade das máquinas** (*brkdwn*) – Representa a indisponibilidade das máquinas para o processamento de qualquer trabalho. Tais períodos de indisponibilidade das máquinas, por vezes podem ser fixos, devido a turnos ou manutenção preventiva, por exemplo;
- **Permutações** (*prmu*) – Este tipo de restrições pode existir em configurações do tipo linha de fabrico e representa a impossibilidade de alterar a sequência de tarefas após a execução na primeira máquina;

- **Sem espera** (*nwt*) – Representa a proibição de esperas entre duas máquinas em configurações do tipo linha de fabrico. Isto implica que uma tarefa apenas pode iniciar a ser processada quando estiverem reunidas as condições para poder circular através da linha sem qualquer tempo de espera;
- **Recirculação** (*rcrc*) – Representa a possibilidade de uma tarefa poder visitar uma máquina mais do que uma vez. Esta característica apenas ocorre em configurações do tipo oficinas de fabrico simples ou híbridas.

Existem outras restrições específicas para certas configurações, o que implica que para caracterizar β é necessário conhecer as especificidades do problema de escalonamento a ser estudado. Ao contrário de α, o campo β pode conter mais do que uma variável ou parâmetro se o problema assim o exigir, nomeadamente entre o conjunto de restrições acima apresentado. Caso não existam restrições ao problema, o mesmo encontra-se vazio.

2.3.3 Função Objetivo- Campo γ

Existem organizações que preferem minimizar os tempos de conclusão das tarefas, outras preferem minimizar os atrasos, e haverá com certeza outros objetivos para o qual outras organizações dão prioridade. Ou seja, consoante os objetivos em causa o escalonamento determina o modo a empresa deve utilizar os seus recursos para atender os objetivos. Assim, é necessário identificar o/ou os objetivos em causa, para tal Pinedo (2012) recorre ao campo γ para alocar tal informação. Os mais comuns são:

- ***Completion time (Cj)*** – Corresponde ao momento em que a tarefa é concluída, ou seja, representa a data de conclusão da tarefa *j*.
  - **Min** $\sum_{j=1}^{n} \boldsymbol{Cj}$ – corresponde a problemas de minimização do tempo total de execução, ou seja, pretende-se minimizar o tempo de execução das tarefas.
  - **Min** $^{1}/_{n} \sum_{j=1}^{n} \boldsymbol{Cj}$ – corresponde a problemas de minimização de tempo médio de execução, ou seja, minimizar o tempo médio de todas as tarefas
- ***Makespan (Cmax)*** – Representa o tempo máximo de conclusão de todas as tarefas de determinada ordem de fabrico.

$$C_{max} = \max Cj$$

  - **Min Cmax** – corresponde a problemas de minimização de *makespan*, ou seja, problemas que se pretende reduzir o tempo de conclusão de uma ordem de fabrico.
- ***Fj*** – corresponde ao tempo de execução de uma tarefa j, determinado pela diferença entre a data de conclusão e a data de entrada no sistema produtivo.

$$Fj = Cj - rj$$

- $\overline{\boldsymbol{F}}$ – corresponde ao tempo médio de execução de um conjunto de tarefas. Determina-se através da média das diferenças entre as datas de conclusão e lançamento.

$$\bar{F} = \frac{1}{n}\sum_{j=1}^{n} Fj$$

Os critérios de pontualidade visam a avaliação do cumprimento dos tempos de entrega definidos. Estes critérios são:

- ***Lateness (Lj)*** – representa o atraso ou antecipação relativamente à data definida, calculando-se através da diferença entre a data de entrega e data de conclusão de uma tarefa:

$$Lj = Cj - dj$$

  - **Min** $\sum_{j=1}^{n} \boldsymbol{Lj}$ **–** corresponde a problemas de minimização de atraso ou antecipação.
  - **Min** $\sum_{j=1}^{n} \boldsymbol{wj\ Lj}$ **-** corresponde a problemas de minimização de atraso ou antecipação ponderadas.

- ***Maximum Lateness (Lmax)*** – corresponde ao atraso ou antecipação máximo de uma sequência.

$$L_{max} = \max |Lj|$$

  - **Min Lmax** - corresponde a problemas de minimização de atraso ou antecipação máxima.

- ***Earliness (Ej)*** – Corresponde conclusão de uma tarefa antes da data definida, ou seja, antecipação.

$$Ej = \max(|Lj|, 0)$$

- ***Maximum Earliness (Emax)*** – corresponde à antecipação máximo de uma sequência de tarefas, ou seja, antecipação máxima.

$$E_{max} = \max Ej$$

- ***Medium Earliness (Ē)*** – corresponde à antecipação média de uma determinada sequencia de fabrico, ou seja, adiantamento médio das tarefas:

$$\bar{E} = \frac{1}{n}\sum_{j=1}^{n} Ej$$

- ***Tardiness (Tj)*** – corresponde a um atraso de uma tarefa em relação a uma data definida. Neste critério não contabiliza a antecipação das datas definidas.

$$Tj = \max(Lj\ , 0)$$

- ***Maximum Tardiness (Tmax)*** – corresponde ao atraso máximo de uma sequência de fabrico, ou seja, representa o atraso máximo das tarefas.

$$T_{max} = \max |Tj|$$

- ***Medium Tardiness*** $(\bar{T})$ – corresponde ao atraso médio de uma sequência de fabrico, representando-se por:

$$\bar{T} = \frac{1}{n}\sum_{j=1}^{n} Tj$$

- **Min** $\sum_{j=1}^{n} Uj$ **–** corresponde a problemas de minimização do número de tarefas em atraso, ou seja, pretende minimizar o número de tarefas que não cumprem as datas de entrega.

Os critérios de desempenho de produtividade visam a avaliação da produtividade do sistema. Estes critérios são:

- ***Uri –*** corresponde à taxa de utilização de uma máquina, calculando-se:

$$URi = \frac{1}{Cmax}\sum_{j=1}^{n} Pij\ x\ 100$$

- ***UR –*** corresponde á taxa de utilização do sistema produtivo na globalidade. Calculando-se através do tempo de execução de todas as máquinas e o tempo disponível da máquina.

$$UR = \frac{1}{Cmax\ .m}\sum_{i=1}^{m}\ \ \sum_{j=1}^{n} Pij\ x\ 100$$

2.4 Teoria da Complexidade

Quando um novo problema de escalonamento surge, a primeira ideia que ocorre é desenvolver um algoritmo que resolva otimamente o problema em causa. Contudo, para muitos problemas de escalonamento, não se consegue, a curto prazo, determinar um algoritmo mais eficiente que um que enumere todas as soluções possíveis e escolha a melhor (Leung, 2004). No entanto, tal como Santos, A. B. G. S. (2013) refere, a dimensão (*n*) do problema, por norma, determina o esforço necessário que um algoritmo necessita de executar para solucionar o problema.

As funções de complexidade nos problemas de escalonamento podem ser de duas naturezas distintas. Existem funções de complexidade que podem ser descritas como funções polinomiais, onde os algoritmos são designados como algoritmos polinomiais ou de classe P. No entanto, nem sempre é possível representar a função de complexidade como uma função polinomial, pelo que se recorre a algoritmos não polinomiais, isto é, classe NP. Para demonstrar a complexidade subjacente a estas naturezas distintas de algoritmos, apresenta-se um exemplo. Dadas duas funções, *n* (P) e 2^n (NP), quando *n* = 100 a primeira necessita de 100 ciclos de processamento computacional, já a segunda precisa de $1{,}27 \times 10^{30}$ ciclos. Esta discrepância de número de ciclos observada deve-se ao facto de a primeira função *n* crescer polinomialmente com *n*, enquanto a segunda função 2^n cresce exponencialmente com a dimensão *n* do problema (Blazewicz et al., 2007; A. B. G. S. e Santos, 2015).

Blazewicz, J. *et al.* (2007) refere ainda duas subclasses dos problemas de pesquisa, os problemas de otimização e de decisão. Nos problemas de otimização, os mesmos procuram a melhor solução sob determinadas restrições. No caso dos problemas de decisão, os mesmos avaliam se uma solução deve ou não ser aceite como solução do problema, ou seja, respondem de um modo binário.

Em termos de complexidade computacional, não existem diferenças entre problemas de otimização e decisão, ou seja, se um problema de otimização puder ser eficientemente resolvido, o problema de decisão também o poderá (Blazewicz et al., 2007; A. B. G. S. e Santos, 2015).

Como se tem vindo a referir, os problemas de escalonamento podem apenas ser de sequenciamento, de afetação ou completos (afetação e sequenciação). Para o caso mais simples, onde apenas a sequenciação é necessária, o número de soluções possíveis é $n!$, onde n representa o número de tarefas no sistema. No entanto, e agora abordando o cenário mais complexo, nas oficinas de fabrico existem n tarefas e m máquinas para as executar, logo o número de soluções possíveis é agora $(n!)^m$. A Tabela 1 mostra a variação do número de soluções com a variação do número de tarefas e máquinas num ambiente *job shop.*

Tabela 1 Variação do número de soluções

Número de entidades (n)	Número de máquinas (m)	Número de soluções
5	1	120
5	5	$2{,}5\text{x}10^{10}$
5	10	$6{,}19\text{x}10^{20}$
10	10	$3{,}96\text{x}10^{65}$

Como se conclui através da tabela acima, mesmo com problemas relativamente pequenos, a procura pela solução ótima na programação da produção torna-se extremamente complexa.

2.5 Ambientes de Escalonamento

O desenvolvimento do escalonamento da produção apresenta elevada complexidade devido à natureza combinatória como também à elevada incerteza e alteração das condições do sistema. Ou seja, os recursos disponíveis podem rapidamente sofrer alterações (avarias dos recursos) ou até as alterações ou desconhecimento dos tempos de execução dos trabalhos. Estas alterações inviabilizam rapidamente o escalonamento desenvolvido. Assim, os modelos de escalonamento recorrem à utilização de distribuições estatísticas para os tempos de execução e probabilidade de avarias das máquinas. O escalonamento pode ser classificado de acordo com as seguintes características (Pinedo, 2012):

- **Escalonamento Determinístico:** Neste ambiente de escalonamento todos os parâmetros e condições do problema são conhecidos à priori, bem como constantes ao longo do tempo. É muito raro encontrar este tipo de ambientes nas indústrias, mas serve para simplificar cenários reais que trazem consigo alguma aleatoriedade e incerteza;
- **Escalonamento Estocástico:** Ao contrário do escalonamento determinístico, neste ambiente a maior parte das condições e parâmetros do problema não são conhecidos. Para modelar a incerteza inerente a este ambiente, recorre-se a distribuições estatísticas;
- **Escalonamento Estático:** Em ambientes estáticos, a solução de escalonamento inicialmente definida para um dado conjunto de tarefas, é cumprida independentemente da chegada de novos trabalhos. Ou seja, sempre que novas tarefas cheguem para serem lançadas, o plano não irá alterar;
- **Escalonamento Dinâmico:** Neste caso, não se conhece desde início todas as tarefas do problema. Assim, programa-se as tarefas que atualmente se conhece, e à medida que novas tarefas vão dando entrada no sistema, ajusta-se o plano de modo a incorporar todas as tarefas. Por norma, o escalonamento dinâmico reordena uma fila de espera de tarefas que ainda não foram executadas, sempre que uma nova tarefa dá entrada no sistema (Pinedo, 2012; A. B. G. S. e Santos, 2015; Maria Leonilde Rocha Varela, 2007).

Tem havido um crescente interesse em modelar e resolver problemas de programação em ambientes dinâmicos e estocásticos. Em ambientes industriais, o escalonamento é um processo em constante mudança onde eventos em tempo real forçam a reconsideração e a revisão de planos pré-estabelecidos. Como mencionado anteriormente, neste ambiente de escalonamento as tarefas não são conhecidas no início do problema, o que torna o sistema vulnerável a eventos em tempo real inevitáveis e imprevisíveis, que causam uma mudança nos planos idealizados, e que fazem com que um plano anteriormente viável se torne obsoleto assim que é libertado para o chão de fábrica. Esses eventos inesperados podem ocorrer por vários motivos, como os relacionados aos recursos (por exemplo, interrupções, retrabalho, doença do operador) ou os trabalhos (por exemplo, cancelamento de encomendas, alterações nos prazos de entrega, chegadas tardias).

O principal objetivo na resolução deste problema já não passa por encontrar os melhores planos, uma vez que estes últimos rapidamente se tornarão obsoletos, mas sim pela capacidade de se adaptarem eficientemente as soluções atuais ao ambiente dinâmico, uma vez que soluções satisfatórias facilmente adaptáveis, serão preferíveis para a soluções ótimas que dificilmente se conseguem encontrar. Assim, quando perturbações não planeadas ocorrem, é necessário encontrar um novo programa com a qualidade próxima do programa que poderia ter sido executado se toda a incerteza tivesse sido revelada à priori (L. Ferreirinha et al., 2019; Terekhov, Down, & Beck, 2014)

O escalonamento dinâmico pode ser definido sob três categorias (Aytug, Lawley, McKay, Mohan, & Uzsoy, 2005; Ouelhadj & Petrovic, 2009; M. L. R. Varela & Ribeiro, 2014; Vieira et al., 2003), que estão resumidas na Tabela 2.

Tabela 2 Categorias do Escalonamento Dinâmico

Escalonamento completamente reativo	Escalonamento preditivo-reativo
No escalonamento completamente reativo, nenhum plano é gerado antecipadamente. As decisões são tomadas localmente em tempo real, geralmente por regras prioridade.	O escalonamento preditivo-reativo é um processo iterativo e possui duas etapas principais. O primeiro passo gera um programa de produção. A segunda etapa atualiza o programa em resposta a um evento em tempo real.
Escalonamento pró-ativo robusto	
O escalonamento proativo robusto é baseado em escalonamento preditivos que satisfazem os requisitos de desempenho de maneira previsível num ambiente dinâmico.	

Quando se trata de reprogramar devido a eventos em tempo real, dois grandes problemas emergem: como e quando reagir a esses eventos. A primeira questão é a estratégia a ser usada no reescalonamento, que geralmente leva à adaptação do escalonamento ou ao escalonamento completo (I Sabuncuoglu & Bayız, 2000; Vieira et al., 2003). A adaptação do escalonamento refere-se ao ajuste do programa atual, economizando tempo de CPU e sem comprometer a estabilidade do sistema. Em relação ao escalonamento completo, este refere-se literalmente à programação do zero. Embora este último possa ser melhor para manter a solução ideal, na prática geralmente não há tempo para reagendar (Aytug et al., 2005; Ouelhadj & Petrovic, 2009; M. L. R. Varela & Ribeiro, 2014; Vieira et al., 2003). Em relação à segunda questão, quando reprogramar, basicamente visa responder quando um evento tem impacto suficiente para que um novo escalonamento seja necessário. Três políticas podem ser encontradas na literatura (Aytug et al., 2005; Ouelhadj & Petrovic, 2009; I Sabuncuoglu & Bayız, 2000; Vieira et al., 2003): *Reescalonamento contínuo ou orientado a eventos*, que reprograma cada vez que um evento ocorre, como a chegada de novas tarefas; *reescalonamento periódico*, onde os planos são gerados em intervalos regulares *T* e quaisquer interrupções entre os períodos de reprogramação são ignorados até o próximo período, onde reúne todas as informações disponíveis no chão de fábrica. E por último o *híbrido*, que reescalona o sistema periodicamente e também quando ocorrem alguns eventos particulares, como avaria de máquinas, chegada de trabalhos urgentes, cancelamento de trabalhos, entre outros (Aytug et al., 2005; CHURCH & UZSOY, 1992; I Sabuncuoglu & Bayız, 2000; Ihsan Sabuncuoglu & Karabuk, 1999).

MÉTODOS DE OTIMIZAÇÃO

3 Métodos de Otimização

Neste capítulo pretende-se introduzir o conceito de problemas de otimização, em especial, os de otimização combinatória, bem como métodos de resolução dos mesmos. Uma vez que a presente dissertação se enquadra em problemas de escalonamento, a apresentação e explicação de alguns métodos de resolução são focados para esta respetiva área. Um dos métodos de resolução passa pela utilização de meta heurísticas, e como grande parte das mesmas tem como base o conceito de pesquisa de vizinhança, começar-se-á por apresentar os métodos de pesquisa local e os seus conceitos. Para aprofundar um pouco o conceito das meta heurísticas, o presente capítulo conta com uma distinção entre os conceitos de ótimos locais e globais, bem como de intensificação e diversificação, uma vez que ajudam a perceber melhor aquilo que são as meta heurísticas. São ainda apresentados o Algoritmo Genético, o *Simulated Annealing* e o *Discrete Artificial Bee Colony* e analisadas as técnicas de parametrização mais comuns.

3.1 Enquadramento

Muitos problemas de otimização de importância tanto ao nível prático como teórico consistem na procura do melhor arranjo de um conjunto de variáveis de forma a alcançar um determinado objetivo. Tais problemas tendem a dividir-se naturalmente em duas categorias: aqueles onde as soluções são codificadas com variáveis de valor real e aquelas onde as soluções são codificadas com variáveis discretas. Entre os últimos encontramos uma classe de problemas denominados de problemas de Otimização Combinatória (OC) (Blum & Roli, 2003). Exemplos deste tipo de problemas são o problema do caixeiro viajante (TSP), *Quadratic assignment problems (QAP), e muitos problemas de* escalonamento *(Scheduling)*, entre outros.

Dada a importância prática dos problemas de OC, diversos algoritmos que permitem a sua resolução foram desenvolvidos. Tais algoritmos podem ser classificados como algoritmos completos/exatos ou aproximados. No que diz respeitos aos algoritmos completos existe a garantia de encontrar a solução ótima através de algoritmos com tempo de resolução polinomial do tamanho do problema (Sen, Sulek, & Dileepan, 2003). No entanto, existem problemas de OC que são *NP-Hard*, o que significa que não existe um algoritmo para a resolução dos mesmos em tempo polinomial, o que faz com que os métodos de otimização, no pior caso, necessitem de tempo de computação exponencial. Tais cenários levam a tempos de computação muito morosos para fins práticos, pelo que o recurso a algoritmos aproximados para resolver problemas de OC tem ganho cada vez mais atenção.

Em algoritmos aproximados sacrifica-se a garantia da obtenção da solução ótima por uma solução razoável num período de tempo útil (Blum & Roli, 2003).

Nos capítulos seguintes irão ser apresentadas diversas técnicas de otimização que permitem a resolução de problemas de OC. No entanto, dado o enquadramento do presente trabalho nos problemas de escalonamento, a apresentação das diversas

técnicas de otimização será voltada para problemas desse tipo. Contudo, vale a pena realçar que os métodos que serão mencionados não são limitados aos problemas de escalonamento, pelo que podem ser aplicados na resolução de outro tipo de problemas, como o problema do TSP, por exemplo.

Dependendo das características do problema, os métodos de otimização podem ser abordados por diversas técnicas, como mostra a Figura 4. No que toca aos métodos do escalonamento, os mesmos são geralmente desenvolvidos para resolver uma classe de problemas específica (Maria Leonilde Rocha Varela, 2007).

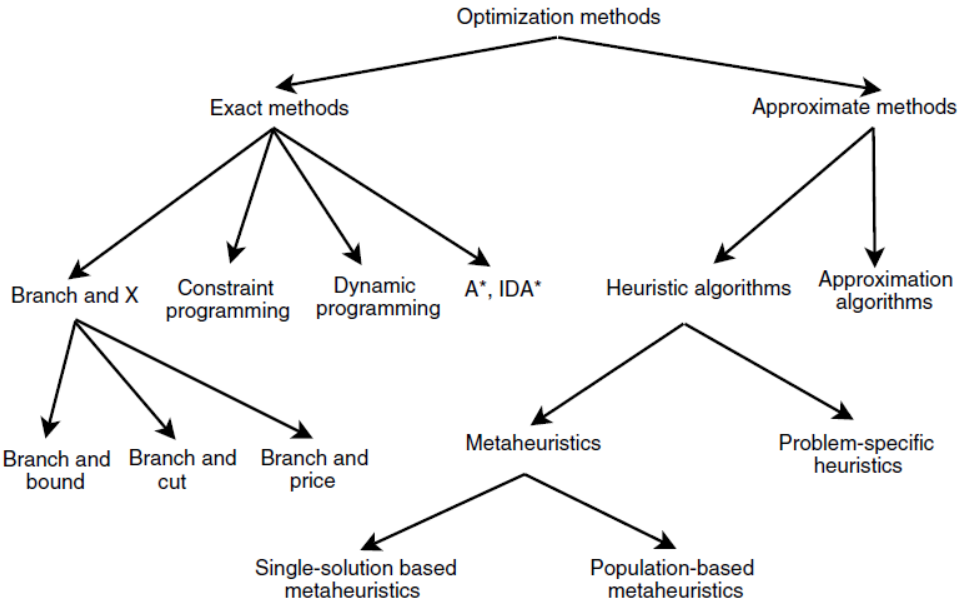


Figura 4 Técnicas de Otimização clássicas (Talbi, 2009)

Maioritariamente, os problemas de escalonamento são de natureza combinatória pelo que tradicionalmente se recorre a técnicas enumerativas para encontrar a solução ótima. Tal como se referiu no ponto 3.1., para problemas de natureza combinatória, existem métodos exatos, que garantem o ótimo, e técnicas aproximadas, que apesar de não garantirem o ótimo, encontram soluções bastante satisfatórias em períodos de tempo razoáveis. A programação da produção inclui problemas computacionalmente complexos, o que impossibilita, na maior parte dos casos, a utilização de métodos exatos para solucionar otimamente o problema em causa em tempo útil. Neste tipo de problemas, é muito usual recorrer a métodos aproximativos, que, tal como já se referiu, apesar de não garantirem uma solução ótima, possibilitam, em reduzido tempo computacional, encontrar uma solução bastante satisfatória (Maria Leonilde Rocha Varela, 2007).

3.2 Métodos de otimização clássicos

3.2.1 Métodos Exatos

Para qualquer problema de escalonamento, a solução mais interessante e que geralmente se procura é a solução ótima. No entanto, dada a complexidade inerente ao escalonamento da produção recorre-se maioritariamente a métodos heurísticos.

Os métodos exatos, ou de otimização, são métodos que encontram garantidamente a solução ótima para o problema em causa em função do critério de otimização. Ou seja, são métodos que exploram a totalidade do espaço de soluções possíveis do problema, com o intuito de encontrar a melhor solução para o objetivo pretendido (Maria Leonilde Rocha Varela, 2007). Uma técnica é exata se garantir que todas as restrições subjacentes ao problema são cumpridas, e é chamada de completa se num tempo de computação polinomial do tamanho do problema, for possível encontrar a solução ótima (A. B. G. S. e Santos, 2015).

De seguida, aborda-se de um modo superficial alguns métodos exatos:

- **Programação Dinâmica**

  Desenvolvida por Bellman nos anos 50, este método exato resolve os problemas através de um processo multi-estágio e recursivo. Baseia-se no princípio de Bellman que afirma que uma subsolução de uma solução ótima, é também ótima. Ou seja, o problema é dividido em vários subproblemas e a solução é construída recursivamente através de uma sequência de soluções desses subproblemas. Tal método evita a enumeração de subsoluções não ótimas, que segundo o princípio de Bellman, não irão resultar numa solução ótima (A. B. G. S. Santos, 2013; Maria Leonilde Rocha Varela, 2007).

- ***Branch and X***

  Representam uma família de métodos de otimização de enumeração implícita de soluções, a qual inclui o *Branch and Bound, Branch and Cut* e o *Branch and Price*. O problema em questão é dividido em vários subproblemas e resolvido através de uma árvore de decisão onde cada ramo representa um subproblema e cada folha uma solução. O *Branch* representa a divisão do problema em diversos subproblemas, enquanto o complementar, *bound, cut* ou *price*, representam as técnicas de redução de pesquisa da árvore de decisão, ou seja, os modos de encontrar uma solução ótima sem explorar todos os ramos da árvore (A. B. G. S. Santos, 2013).

3.2.2 Métodos Heurísticos

Dada a complexidade dos problemas de escalonamento e à escassez de tempo que geralmente se dispõe para se resolver tais problemas, na maioria dos casos não é possível encontrar a solução ótima. Daí recorrer-se a técnicas de aproximação, ou

heurísticas, que apesar de não garantirem o ótimo, encontram soluções bastantes próximas do mesmo em espaços de tempo computacionais aceitáveis. Nem sempre é percetível, mas as expressões "métodos heurísticos" e "métodos de aproximação" representam duas realidades distintas. No caso dos métodos heurísticos, não se consegue ter uma perceção de quão boa é a solução final, isto é, não existe nada que permite saber o quão afastada da solução ótima se encontra a solução encontrada. Já quanto aos métodos aproximados, é possível avaliar analiticamente o grau de aproximação que a solução encontrada está da solução ótima (Maria Leonilde Rocha Varela, 2007). Dentro dos métodos heurísticos, é possível destacar:

- **Heurísticas construtivas**

  Tratam-se de métodos muito simples de aplicação que nem sempre geram soluções "boas", mas a sua utilização justifica-se quando existem decisões a tomar em curtos espaços de tempo. São habitualmente baseados em algoritmos "gulosos", ou seja, que optam pelo ótimo local sem ter em consideração os níveis de decisão seguintes. Uma vez que estas heurísticas constroem uma solução desde o início, são muitas vezes utilizadas como ponto de partida para outros métodos que necessitam de uma solução à priori, como por exemplo as heurísticas de pesquisa local ou em métodos de *branch and bound* (A. B. G. S. Santos, 2013).

- **Heurísticas de pesquisa local**

  São métodos que, dada uma solução inicial, manipulam a mesma de modo a obterem uma melhor solução. Ou seja, tratam-se de heurísticas que pesquisam parte do espaço de soluções possíveis, com o intuito de encontrarem uma solução com melhor desempenho. A principal dificuldade destes métodos é o facto de bloquearem sempre que encontram um ótimo local, aspeto que será abordado com mais detalhe no ponto 3.5.

- **Meta Heurísticas**

  Representam a maioria dos métodos utilizados para resolver os problemas de escalonamento mais complexos. Estes métodos regem-se por mecanismos estocásticos que permitem obter uma maior diversidade na pesquisa do espaço de soluções possíveis, ultrapassando, assim, os ótimos locais. As meta heurísticas são habitualmente inspiradas em fenómenos naturais, e podem ser divididas em meta heurísticas de solução única ou meta heurísticas de população.

3.2.3 Regras de prioridade

Tal como se referiu anteriormente, as heurísticas são métodos de adaptação do escalonamento específicas do problema, que têm a capacidade de encontrar soluções quase ótimas rapidamente e com pouco esforço computacional. As regras de despacho são também heurísticas com grande importância no escalonamento completamente

reativo e são usadas para classificar os trabalhos na fila de máquinas por alguns critérios. Portanto, semelhante aos mecanismos pull, como os cartões Kanban, as regras de despacho são usadas para controlar a produção.

Como o próprio nome indica, estas regras estabelecem uma prioridade às tarefas que aguardam para ser processadas numa determinada máquina. São muito comuns em problemas de sequenciamento e o modo como a prioridade é atribuída às tarefas pode estar relacionada com as próprias tarefas, com as propriedades das máquinas, ou com os objetivos da empresa. Ou seja, assim que uma tarefa acaba de ser processada, é a regra de prioridade selecionada que irá determinar a próxima tarefa a ser executada.

Torna-se ainda interessante destacar a existência de regras estáticas e dinâmicas. As regras de prioridade estáticas são aquelas cujo valor da prioridade atribuído a uma determinada tarefa é independente do instante de tempo atual. Por exemplo, a regra Shortest Processing Time (SPT) sequencia as tarefas por ordem crescente do seu tempo de processamento, ou seja, a tarefa com o menor tempo de processamento, têm prioridade sobre as outras. Tal regra é claramente estática, pois, por norma, os tempos de processamento não variam com o tempo. Já as regras de prioridade dinâmicas representam o outro lado do espelho, isto é, o valor da prioridade varia com o tempo. A título de exemplo, analise-se a regra “folga para processamento”, a qual diz respeito ao tempo que ainda há para processamento atá à data de entrega. O valor desta regra, como é percetível, varia continuamento no tempo, e por isso pertence ao grupo de regras de prioridade dinâmica.

Uma extensa lista de regras de despacho é apresentada e a dificuldade da escolha de uma regra de despacho surge do fato de que existem n! maneiras de sequenciar n tarefas (Panwalkar & Iskander, 1977). Na literatura, o desempenho das regras de despacho é geralmente avaliado experimentalmente, sendo que em alguns casos, tais regras de prioridades mostram-se ótimas, como a “Shortest Processing Time” (SPT) que minimiza o Tempo Médio de Fluxo e a “Earliest Due Date” (EDD) que minimiza o Atraso Máximo, entre outros (L. Ferreirinha et al., 2019; Ouelhadj & Petrovic, 2009; Rajendran & Holthaus, 1999; Ihsan Sabuncuoglu & Karabuk, 1999; Terekhov et al., 2014; Maria Leonilde Rocha Varela, 2007). Um escalonamento dinâmico pode ser visto como uma coleção de problemas estáticos interligados, o que implica que os métodos desenvolvidos para problemas de escalonamento estáticos se tornam aplicáveis aos dinâmicos. Tais métodos podem efetivamente lidar com problemas complexos e otimizar a qualidade dos escalonamentos para cada sub-problema estático. (Akers & Friedman, 1955) Os autores de (Panwalkar & Iskander, 1977; Rajendran & Holthaus, 1999; Ramasesh, 1990) apresentam um levantamento de última geração das regras de despacho.

3.3 Meta Heurísticas

Tal como se referiu anteriormente, tem-se vindo a procurar continuamente melhores soluções para os problemas de OC. Tais problemas têm-se tornado cada vez mais

complexos pelo que a procura exaustiva por soluções ótimas tornou-se, de um modo geral, impraticável, dado o tempo computacional necessário para achar as mesmas.

Nas últimas décadas surgiu uma nova família de algoritmos aproximados que dominou a pesquisa de soluções de problemas de otimização combinatória. O objetivo desta nova metodologia é eficientemente e eficazmente explorar o espaço de soluções impulsionado por movimentos lógicos e pela perceção do efeito de um determinado movimento, facilitando assim a procura de soluções localmente ótimas. Tais métodos são denominados de Meta Heurísticas, um termo que foi inicialmente introduzido por Glover nos anos 80 (F. Glover, 1986). De acordo com (Xhafa & Abraham, 2008) as meta heurísticas são mais vantajosas em relação às heurísticas mais comuns em termos de robustez de solução. No entanto, são mais difíceis de implementar e sintonizar, uma vez que precisam de outro tipo de informações sobre o problema a ser resolvido de modo a obter bons resultados. Devido à complexidade computacional dos problemas de otimização combinatória, os resultados moderados por métodos heurísticos, as limitações de tempo para a aplicação de algoritmos exatos e a implementação de métodos meta-heurísticos para resolver tais problemas, tem sido um campo de investigação importante nas áreas de engenharia e ciência (Xhafa & Abraham, 2008).

Talbi, E. G. (2009) afirma ainda que ao contrário dos métodos exatos, as meta heurísticas permitem lidar com instâncias de problemas de elevada dimensão e encontram soluções satisfatórias num espaço de tempo razoável. À semelhança das heurísticas mais comuns, nos métodos meta-heurísticos não há garantias de que a solução encontrada representa a solução ótima. De acordo com o mesmo autor, o uso das meta heurísticas em diversas aplicações, tem demonstrado a sua eficiência e eficácia na resolução de diversos problemas de diversas áreas de investigação.

A Figura 5 mostra a evolução das meta heurísticas no seculo XX.

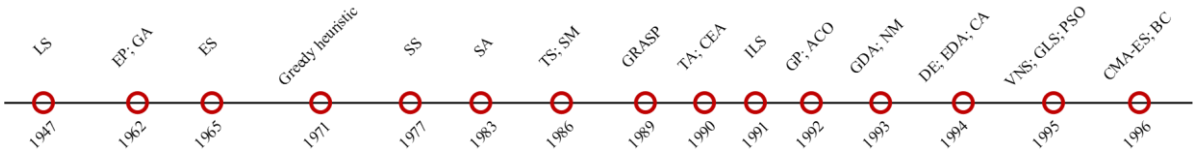


Figura 5 Evolução das Meta heurísticas (Adaptado de (Talbi, 2009))

Atualmente, as meta heurísticas representam as principais ferramentas para solucionar problemas computacionalmente complexos, tais como problemas de otimização combinatória, uma vez que tem vindo a demonstrar-se como técnicas com melhor desempenho (A. B. G. S. e Santos, 2015).

3.4 Pesquisa Local

A pesquisa local é provavelmente o método heurístico mais antigo e simples de aplicar (Talbi, 2009). Os algoritmos de pesquisa local (*Local Search*-LS) partem de uma solução inicial, a qual na maioria das vezes é gerada por uma heurística construtiva ou aleatoriamente, e tenta iterativamente substituir parte, ou a solução inteira, por uma

melhor num apropriado conjunto definido de populações vizinhas (Xhafa & Abraham, 2008). O processo continua até não ser possível encontrar uma solução com melhor desempenho na vizinhança atual. Ou seja, percebe-se aqui que estes métodos utilizam o conceito de vizinhança para pesquisar por soluções com melhor desempenho, sendo que estas últimas dependem da escolha do modo como as soluções vizinhas são geradas (de forma determinística ou estocástica) e da estratégia de seleção da solução vizinha (Talbi, 2009). Na Tabela 3 encontra-se o procedimento da Pesquisa Local para um problema de minimização (A. B. G. S. e Santos, 2015).

Tabela 3 Método de Pesquisa Local

Escolher uma solução inicial $x_n = x_1 \in x$;

Encontrar a melhor solução na vizinhança da solução atual $v(x_n)$;

If *$f(x_n+1) < f(x_n)$*

$x_n = x_n+1$ e voltar ao 2º passo;

else

Terminar.

Contudo, Xhafa & Abraham (2008) realçam que a principal desvantagem dos métodos básicos da pesquisa local consiste nas soluções convergirem facilmente para um ótimo local, Figura 6. Ou seja, a pesquisa local, com movimentos apropriados, pode ser muito eficaz na exploração de uma vizinhança de uma solução inicial (x), Figura 6, no entanto não possui nenhum mecanismo que a faça levar para outras vizinhanças distantes no espaço de soluções onde o ótimo global possa existir. Para contornar a fraqueza percetível nestes algoritmos, novos métodos foram desenvolvidos com o intuito de guiar o processo de procura pela solução ótima, os tais chamados de métodos meta-heurísticos.

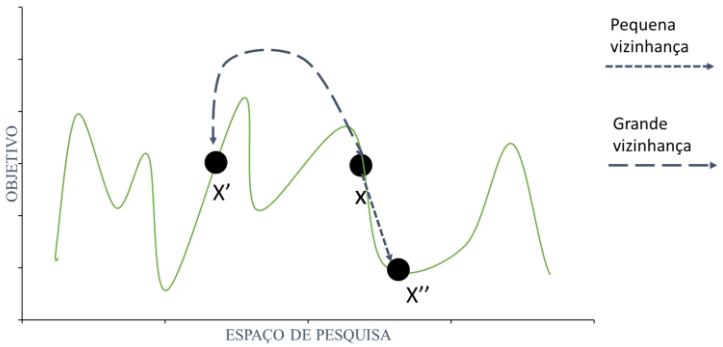


Figura 6 Comportamento da pesquisa local (adaptado de Talbi, 2009)

Para ilustrar o funcionamento desta técnica segue-se um pequeno exemplo de minimização do *makespan* em *flowshop* com as seis primeiras tarefas do problema em estudo, apresentadas na Tabela 4.

Tabela 4 Tarefas utilizadas para o exemplo ilustrativo

J_i	P_{i1}	P_{i2}	P_{i3}	P_{i4}	P_{i5}
1	38	1	50	92	60
2	85	86	78	30	37
3	34	37	88	7	60
4	36	5	13	5	98
5	82	62	12	26	24
6	68	78	18	73	88

Partindo então da solução inicial 1 2 3 4 5 6 com um *makespan* de 652, o problema vai ser abordado pela técnica da pesquisa local com recurso ao mecanismo de vizinhança de troca de pares adjacentes, Tabela 5.

Tabela 5 Exemplo de aplicação de Pesquisa Local

	Solução	f(x)		Solução	f(x)		Solução	f(x)		Solução	f(x)		Solução	f(x)
1ª Iteração	2 1 3 4 5 6	721	2ª Iteração	3 1 2 4 5 6	608	3ª Iteração	3 1 2 4 6 5	608	4ª Iteração	3 1 4 2 6 5	608	5ª Iteração	3 1 4 6 2 5	608
	1 3 2 4 5 6	600		1 2 3 4 5 6	652		1 2 3 4 6 5	652		1 4 3 2 6 5	572		1 4 3 6 2 5	548
	1 2 4 3 5 6	624		1 3 4 2 5 6	600		1 3 4 2 6 5	572		1 3 2 4 6 5	598		1 3 6 4 2 5	556
	1 2 3 5 4 6	652		1 3 2 5 4 6	600		1 3 2 6 4 5	634		1 3 4 6 2 5	548		1 3 4 2 6 5	572
	1 2 3 4 6 5	652		1 3 2 4 6 5	598		1 3 2 4 5 6	600		1 3 4 2 5 6	600		1 3 4 6 5 2	574

No exemplo resolvido foram encontradas duas soluções com melhor desempenho, 1 3 4 6 2 5 e 1 4 3 6 2 5 ambas com um *makespan* de 548, que são um ótimo local.

3.5 Ótimos Locais e Ótimos Globais

No ponto anterior abordaram-se conceitos relacionados com a caracterização do ótimo encontrado, ou seja, se o mesmo consistia num ótimo local ou global. Neste ponto pretende-se aprofundar mais um pouco acerca desses dois conceitos.

Neste momento, e como já se abordou o comportamento dos métodos de pesquisa local, a melhor forma para explicar as diferenças entre ótimos passa pela análise da Figura 7.

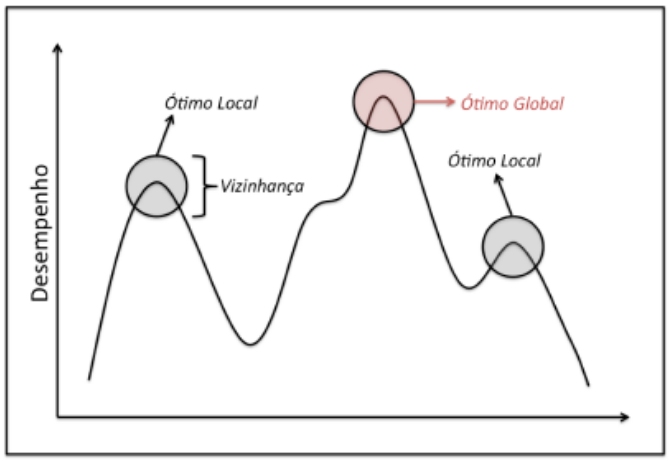


Figura 7 Ótimos locais e globais (A. B. G. S. e Santos, 2015)

Pela análise da Figura 7 são percetíveis as diferenças entre estes dois conceitos. Ou seja, uma vez que a pesquisa da LS é sempre realizada numa determinada vizinhança, apenas

são consideradas como próximas soluções aquelas que efetivamente se encontram próximas da solução atual. Dentro das elegíveis, apenas a solução com melhor desempenho ao da solução atual é que será selecionada. Tal cenário faz com que as técnicas de LS, caso não se encontrem na vizinhança do ótimo global, convirjam sempre para ótimos locais (A. B. G. S. e Santos, 2015; Talbi, 2009).

Matematicamente, para problemas de maximização, diz-se que x_1 é um ótimo local se o resultado de x_1 for superior ao resultado de qualquer x que pertença à vizinhança de x_1, ou seja:

$$f(x_1) \geq f(x), \forall\, x \ \in v(x_1)$$

Já um ótimo global, será aquele cuja solução tenha o melhor desempenho entre todas as soluções possíveis do problema. Ou seja, x_1 é um ótimo global se e só se:

$$f(x_1) \geq f(x), \forall\, x$$

3.6 Intensidade e Diversidade

Na implementação de uma meta-heurística deve-se ter em conta dois critérios contraditórios, a exploração do espaço de soluções (diversificação) e a exploração de soluções na vizinhança (intensificação). Na intensificação, as regiões promissoras são exploradas mais detalhadamente com o objetivo de encontrar melhores soluções. Já na diversificação, as regiões não exploradas são visitadas com o intuito de garantir que todas as regiões do espaço de soluções são exploradas uniformemente e que a pesquisa não fique restrita a uma única região de vizinhança (Talbi, 2009).

A grande questão que se levanta é quando se deve recorrer à exploração de regiões inexploradas (diversificação) e/ou quando é que se deve abrandar na pesquisa e explorar com mais detalhe o espaço da vizinhança (intensificação) (Dréo, Chatterjee, Pétrowski, Siarry, & Taillard, 2006). Por outras palavras, pode-se afirmar que esses dois conceitos representam a relação entre eficiência e eficácia da técnica de otimização (A. B. G. S. e Santos, 2015). Isto é, enquanto uma pesquisa muito intensa vai convergir muito rapidamente para um ótimo local da vizinhança da solução inicial, uma pesquisa muito diversificada, aleatória, irá movimentar-se através de regiões sem confluir para um único ponto.

Embora a intensidade e a diversidade possam ser controladas através dos parâmetros das meta heurísticas, existem técnicas que pelas suas características são mais intensas ou têm mais diversidade, como se mostra na Figura 8.

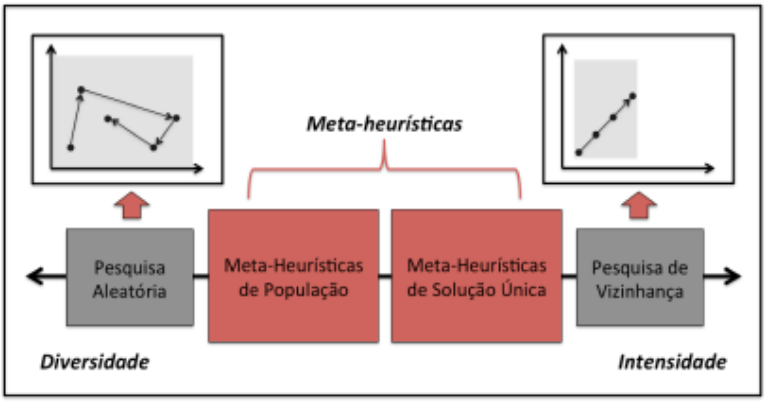


Figura 8 Intensidade e Diversidade (A. B. G. S. e Santos, 2015)

3.7 Descrição de Meta Heurísticas

Através da análise da Figura 8 percebe-se que existem dois tipos de meta heurísticas, as de solução única e as de população. Esta classificação existe pelo facto de existirem meta heurísticas que utilizam apenas uma solução para explorar o espaço de soluções (meta heurísticas de solução única) e outras que utilizam n soluções (meta heurísticas de população). Tal como se pode observar também através da Figura 8, as meta heurísticas de solução única são mais intensas que as meta heurísticas de população, sendo que estas últimas ao manipularem mais que uma solução, exploram mais regiões do espaço de soluções, tendo por isso mais diversidade (A. B. G. S. e Santos, 2015; Talbi, 2009).

De acordo com A. B. G. S. e Santos (2015), as meta heurísticas que manipulam uma só solução comportam-se de forma idêntica aos métodos de pesquisa local simples, acrescentados de uma componente estocástica de forma a ultrapassar os ótimos locais. Ou seja, estas partem de uma solução inicial e exploram o espaço de soluções da vizinhança, mas têm a capacidade de afastar-se do caminho pré-determinado pela pesquisa local simples, deslocando-se para além de ótimos locais. Existem várias meta heurísticas de solução única, sendo que as mais utilizadas são:

- Simulated Annealing (SA);
- Tabu Search (TS);
- Iterative Local Search (ILS);
- Guided Local Search (GLS).

Quanto às meta heurísticas de população, as mesmas manipulam várias soluções em simultâneo de forma a controlar a pesquisa. Estas partem de múltiplas soluções iniciais (população inicial) e evoluem a atual população de modo a otimizar uma determinada medida de desempenho. Dentro das diversas meta heurísticas de população, destaca-se:

- Genetic Algorithms (GA);
- Ant Colony Optimization (ACO);
- Artificial Bee Colony (ABC);
- Particle Swarm Optimization (PSO).

Para uma fase inicial da ferramenta a desenvolver na presente dissertação, decidiu-se implementar duas meta heurísticas de população (GA e ABC) e uma de solução única (SA), pelo que as mesmas serão de seguida apresentadas com detalhe.

3.7.1 Algoritmos Genéticos

Os algoritmos genéticos (AG) foram originalmente propostos por Holland em 1975. Tal algoritmo genético baseia-se na teoria da origem das espécies de Charles Darwin, a qual diz que numa competição entre indivíduos de cada espécie animal apenas os mais aptos conseguem sobreviver, sendo a natureza responsável por efetuar esta seleção natural (Ana Madureira et al., 2003; Pereira, 2009).

Os algoritmos genéticos são métodos de pesquisa estocásticos amplamente utilizados em diversas áreas de investigação. Os mesmos são teoricamente e empiricamente provados para fornecer uma pesquisa robusta em problemas não lineares complexos. Posto isto, pode-se dizer que os algoritmos genéticos estão enraizados na genética e na ciência da computação, pelo que a terminologia dos AGs têm uma mistura de termos naturais e artificiais (F. W. Glover & Kochenberger, 2006; Siriwardene & Perera, 2006).

Num algoritmo genético, o conjunto de possíveis soluções é denominado de população. O conjunto de parâmetros do algoritmo é definido como cromossoma, sendo que cada parâmetro presente no cromossoma é conhecido como um gene (Siriwardene & Perera, 2006).

Os componentes básicos de um AG são a população de indivíduos, em que cada um deles representa uma potencial solução para o problema considerado, o mecanismo de avaliação dos indivíduos da população e os operadores genéticos que pesquisam novas soluções.

Os AGs trabalham sobre uma população de potenciais soluções, ou indivíduos, às quais é aplicado o princípio da sobrevivência das melhores, i.e., os indivíduos competem entre si pela sobrevivência. Após serem avaliados, os melhores indivíduos têm maior probabilidade de serem escolhidos (para progenitores) e gerarem novos indivíduos (os descendentes). A geração de novos indivíduos é feita através de mecanismos baseados na genética. Assim, os descendentes são gerados a partir da recombinação dos progenitores pelo que herdam algumas das suas características. Para além disso, de uma forma probabilística é aplicada a mutação com o objetivo de possibilitar o aparecimento de algumas características que não existam na população. Os descendentes competem entre si e com os progenitores. Este processo é repetido ao longo de um determinado número de gerações.

Ao longo das gerações, uma vez que os melhores indivíduos têm maior probabilidade de serem selecionados para gerar descendentes, e, possivelmente, de gerarem melhores descendentes, a população tende a ter cada vez melhores indivíduos. Este processo de procura genética conduz à evolução da população e os melhores indivíduos, ou, pelo menos as características dos melhores indivíduos, tendem a sobreviver, tal como sucede

na adaptação natural. A procura utilizando um AG, tal como foi descrito, processa-se de acordo com o procedimento da Tabela 6 (Costa, 2003).

Tabela 6 Algoritmo Genético (Costa, 2003)

A população inicial de indivíduos é criada aleatoriamente, ou através de uma heurística construtiva.
Cada indivíduo na população é avaliado de acordo com uma medida de desempenho.
Um conjunto de indivíduos é selecionado para gerar descendentes de tal forma que os que tiverem melhor desempenho têm maior probabilidade de serem selecionados.
Ao conjunto de indivíduos selecionados são aplicados operadores genéticos como a recombinação e a mutação.
***if** não se verificar o critério de paragem;*
voltar ao passo 2
else
Terminar

Os operadores do algoritmo genético, i.e., os parâmetros tais como a população, o tipo de seleção, o cruzamento e a mutação, desempenham um papel fundamental na eficiência e capacidade de otimização do algoritmo em alcançar a solução ótima. Um dos aspetos mais difíceis de usar os AGs consiste na seleção dos melhores valores de parametrização para o problema em causa.

População

O tamanho da população representa o número de cromossomas presentes numa população. Tamanhos de população elevados aumentam a quantidade de variação da população atual, ou seja, a diversidade populacional, contudo irá exigir mais avaliações de aptidão (*fitness*). Além disso, quando o tamanho da população é muito elevado, há uma tendência por parte do utilizador do algoritmo de reduzir o número de gerações, a fim de reduzir o esforço computacional, já que o esforço de computação depende tanto do tamanho da população como do número de gerações. Tal redução do número de gerações, provocará um impacto negativo na qualidade da solução final. Por outro lado, um tamanho de população pequeno pode fazer com que o algoritmo genético convirja prematuramente para um ótimo local. De acordo com (Siriwardene & Perera, 2006) tamanhos de população entre 30 a 200, foram a escolha geral de muitos investigadores deste tipo de algoritmos. Além disso, o autor refere que o tamanho da população estava relacionado com o tamanho dos cromossomas, pelo que para cromossomas mais longos, eram necessários tamanhos populacionais maiores, pois permitiam uma melhor exploração do espaço de soluções.

Seleção

De acordo Pereira (2009) o operador de seleção tem um papel fundamental nos AGs, dado que é através dele que são selecionados os indivíduos para reprodução. Existem

várias formas de efetuar uma seleção, tendo em conta vários aspetos importantes. Um desses aspetos é a seleção ser efetuada baseada em elitismo. Tal escolha aparenta ser a mais óbvia, em analogia ao mundo real, onde os mais fortes prevalecem sobre os mais fracos. Assim, os mais fracos poderão ser descartados, escolhendo-se assim os mais aptos para a geração seguinte e para reprodução. No entanto, no caso de resolução de problemas de otimização o mesmo não acontece uma vez que os indivíduos descartados podem ser uteis para chegar a um melhor indivíduo, sendo que para tal é necessário recorrer a cruzamentos e mutações. Assim, os operadores de seleção mais utilizados são a seleção por etilismo, o método da roleta, o método da seleção estocástica e o método de seleção por torneios.

Cruzamento

Outro operador dos AGs é o operador de cruzamento (*crossover*) que como já foi referido, tem a função de criar novos indivíduos a partir de outros selecionados para o efeito (progenitores ou descendentes). Estes novos indivíduos desenvolvidos possuem uma combinação genética dos seus progenitores, tal como ocorre na recombinação natural. Se os cromossomas forem apenas cadeias de genes, essas combinações serão feitas através de um ou mais pontos de cruzamento nos cromossomas dos seus ascendentes, sendo misturadas as partes resultantes dos cromossomas para cada descendente (Pereira, 2009).

A taxa de cruzamento diz respeito à probabilidade definida pelo utilizador à qual o cruzamento ocorrerá. Ou seja, uma taxa de cruzamento de 0,9 significa que em média, 90% da população sofrerá um cruzamento. Uma alta taxa de cruzamento estimulará uma boa mistura dos cromossomas. Existem vários métodos de cruzamento na literatura, nomeadamente o cruzamento de ponto único, multiponto, uniforme, entre outros (Siriwardene & Perera, 2006).

Mutação

O parâmetro da mutação consiste em trocar o valor de um ou mais genes num determinado cromossoma com uma probabilidade definida, resultante de uma taxa de mutação, de forma a tornar as populações mais diversificadas em termos genéticos. O objetivo deste parâmetro é gerar pontos aleatórios no espaço de soluções para prevenir a convergência prematura para ótimos locais e, consequentemente, poderem ser exploradas outras regiões do espaço de soluções.

A taxa de mutação é definida pelo utilizador e a mesma determina a probabilidade de que a mutação ocorrerá. Por exemplo, se o tamanho da população for 100, o tamanho do cromossoma for 20 e a taxa de mutação for 0,001, em média, apenas dois genes irão alterar em todo o cromossoma, isto é, 100 x 20 x 0,001 =2.

Os operadores de mutação mais conhecidos em problemas de escalonamento são o *swap*, o qual consiste em trocar dois genes selecionados aleatoriamente e o *shift*, que

permite deslocar um gene relativamente à sua posição, para a direita ou para a esquerda.

Considera-se uma má taxa de mutação aquela que estiver entre [0; 0,001] ou entre [0,02;1]. Já uma boa taxa de mutação deverá estar compreendida entre [0,005; 0,01] (Sadegheih, 2006).

Vale a pena realçar que é importante que seja mantida uma diversidade suficiente na população, de forma a permitir a exploração de muitas regiões do espaço das soluções, e não restringir a pesquisa à vizinhança do ótimo (Ana Madureira et al., 2003).

3.7.2 Simulated Annealing

O Simulated Annealing (SA) é uma meta-heurística usada para resolver problemas complexos, com um grande espaço de soluções e capaz de escapar a ótimos locais de forma a encontrar resultados próximos do ótimo global, num razoável período de tempo (Gendreau & Potvin, 2010; Haridass, Valenzuela, Yucekaya, & McDonald, 2014). Tal meta heurística é baseada num método, da área da metalurgia, de Metropolis *et al.,* (1953) e posteriormente adaptada à resolução de problemas de otimização por Kirkpatrick, S., *el al* (1983) e Černý, V. (1985). A inspiração do algoritmo provém dos princípios da mecânica estatística onde o processo de recozimento (*annealing*) requer inicialmente um aquecimento e posteriormente um arrefecimento lento da substância de modo a obter uma forte estrutura cristalina. Caso a temperatura inicial não seja suficientemente alta ou ocorra um arrefecimento muito rápido, imperfeições surgirão na estrutura final do material (Černý, 1985; Kirkpatrick et al., 1983; Metropolis, Rosenbluth, Rosenbluth, Teller, & Teller, 1953; Talbi, 2009).

O SA tem recebido uma atenção significativa no que diz respeito à resolução de problemas de otimização. Foi aplicado a diversos problemas de otimização tais como o escalonamento da produção, por exemplo em *flow shop* e *job shop*, escalonamento de camiões, roteamento, *cut and packing*, entre muitos outros (Wu, Wang, Pedrycz, Li, & Wang, 2017).

Numa primeira fase o SA parte de uma solução inicial, por norma aleatória ou adquirida através de uma heurística, e é inicializada uma temperatura T com um valor predeterminado. Posteriormente, em cada iteração, o algoritmo procura soluções na vizinhança da solução atual sendo que melhores soluções são imediatamente aceites para substituir a mesma. Na esperança de escapar a ótimos locais, piores soluções podem ser aceites consoante uma probabilidade dependente da temperatura atual e da degradação da solução atual, tal como mostra a expressão 1. Ou seja, dada a solução com pior desempenho, a probabilidade de aceitação da mesma é calculada através da expressão 1 e de forma estocástica a mesma é, ou não, aceite para substituir a solução atual (Gendreau & Potvin, 2010; Talbi, 2009).

$$P(f(x'), f(x), T) = e^{-\frac{f(x')-f(x)}{T}} \quad (1)$$

Através da expressão 1 é possível concluir que quando a temperatura é muito elevada, a probabilidade de aceitação é próxima de 1, o que faz com que soluções com pior desempenho sejam facilmente aceites para substituir a solução atual. Por outro lado, quanto a temperatura apresenta valores pequenos, o oposto acontece, dificultando a aceitação de piores soluções. Por outras palavras, no início do procedimento da meta heurística diversas soluções com pior desempenho são aceites, contribuindo para a exploração do espaço de soluções/diversidade da pesquisa. Contudo, à medida que a temperatura vai diminuindo a probabilidade de aceitação aproxima-se de 0, dificultando a aceitação de piores soluções e tornado a pesquisa de soluções mais intensa. No final do procedimento do algoritmo, a solução atual poderá não ser a melhor solução encontrada ao longo do procedimento do mesmo, pelo que também existe a necessidade de manter a melhor solução até então encontrada (A. B. G. S. e Santos, 2015; Talbi, 2009). O algoritmo do SA pode ser analisado na Tabela 7.

Tabela 7 Algoritmo do Simulated Annealing (A. B. G. S. e Santos, 2015)

```
Selecionar a solução inicial x ϵ X
Inicializar a temperatura T = Ti
While T> TF
        repeat
                Selecionar aleatoriamente x´ ϵ v(x)
                If f(x') ≤ f(x) then
                        x=x'
                else
                        x=x' com probabilidade p (T, f(x'), f(x))
                End
        until tenham sido analisadas Ln soluções
        T=αT
end
return melhor solução
```

3.7.2.1 Programa de Arrefecimento

Segundo Talbi, E. G., (2009) o SA é muito sensível ao programa de arrefecimento selecionado, uma vez que o mesmo tem um grande impacto no sucesso do algoritmo. Para a definição do programa de arrefecimento é necessário ter em conta a temperatura inicial, o estado de equilíbrio (o número de iterações à mesma temperatura), o mecanismo de arrefecimento e a temperatura final, que define o critério de paragem do algoritmo. Em Talbi, E. G., (2009) é possível encontrar uma explicação detalhada sobre estes parâmetros, pelo que no presente relatório se abordarão os mesmos de um modo mais superficial.

Temperatura Inicial (T_i)

De acordo com os pioneiros da utilização do SA, Kirkpatrick, S., *et al.,* (1983), a temperatura inicial deve ser alta o suficiente de modo a que a probabilidade de aceitação de todos os vizinhos da solução inicial seja elevada na fase inicial do algoritmo. Uma das sugestões para a definição da T_i Talbi, E. G., (2009) passa pela utilização da expressão 2, onde σ diz respeito ao desvio padrão da diferença entre valores de funções objetivos e *k= -3 / ln(p)*, com a probabilidade de aceitação *p*. Tal expressão resulta num valor superior a 3σ.

$$T_i = k\ \sigma \tag{2}$$

Em Santos, A. B. G. S., (2015) constata-se que a probabilidade de aceitação de soluções piores poderá estar entre [0,7, 0,8] para a definição da temperatura inicial.

Estado de Equilíbrio (*Ln*)

Na tentativa de levar o sistema a um estado correspondente ao equilíbrio térmico na analogia física, um número de transições suficiente, à mesma temperatura *T,* deve ser aplicado (Eglese, 1990). De acordo com Talbi, E. G., (2009), o número de iterações à mesma temperatura deve ser definido consoante o tamanho do problema e particularmente proporcional ao tamanho da vizinhança, sendo que este último depende do mecanismo de vizinhança que se esteja a utilizar.

Mecanismo de Arrefecimento (α)

Tal como se pode visualizar na Tabela 7, ao fim de um determinado número de iterações à mesma temperatura, a temperatura do sistema diminui, de acordo com a expressão 3.

$$T_i > 0\ \forall i\ \&\ \lim_{i \to +\infty} T_i = 0 \tag{3}$$

Em coerência com o que se referiu anteriormente, um arrefecimento muito rápido produz impurezas na estrutura final, isto é, o algoritmo poderá convergir muito rapidamente para um ótimo local. Em Talbi, E. G., (2009) são apresentados vários mecanismos de arrefecimento, incluindo o apresentado na Tabela 7, o qual se irá utilizar na resolução do problema em estudo, o mecanismo de arrefecimento geométrico α, apresentado na expressão 4.

$$T = \alpha T\ onde\ \alpha \in]0,1[\ \sigma \tag{4}$$

De acordo com o mesmo autor, tal mecanismo é o mais popular entre os restantes, devendo o mesmo estar entre [0,5;0,99]. Já Eglese, R. W., (1990) encurta tal intervalo para valores entre [0,8;0,9].

Já em A. B. G. S., (2015) é possível encontrar uma métrica para o cálculo da constante de arrefecimento α, de acordo com a expressão 5.

$$\alpha = \left({}^{T_f}/_{Ti}\right)^{\frac{1}{(M-1)}} \sigma \qquad (5)$$

Onde T_f eT_i representam temperatura final e inicial, respetivamente e *M* representa o número de patamares de temperatura, ou seja, o número total de iterações à mesma temperatura. Contudo, tal métrica aplica-se em casos onde o critério de paragem diz respeito ao número total de iterações de forma a poder calcular *M*. Caso o critério de paragem seja até $T<T_f$, o número total de iterações à mesma temperatura está dependente do mecanismo de vizinhança, pelo que o mesmo deixa de poder ser calculado.

Temperatura Final (T_f)- Critério de Paragem

São diversos os critérios de paragem propostos na literatura. No caso do algoritmo da Tabela 7, o critério de paragem ocorrerá assim que a temperatura do sistema seja inferior à temperatura final, sendo esta última definida à priori e com valores pequenos, como 0,01 de acordo com. Talbi, E. G., (2009). Já em Eglese, R. W., (1990) o autor refere que o critério de paragem deverá ocorrer ao fim de um determinado número de iterações consecutivas sem melhorias na solução atual. Apresentado por Park, M. W., & Kim, Y. D. (1998), o SA poderá ser interrompido ao fim do número total de patamares da temperatura predeterminado ou simplesmente ao fim de um determinado número de iterações (Park & Kim, 1998).

Dado que nesta fase inicial do modelo desenvolvido na presente dissertação serão implementadas as meta heurísticas SA e ABC, neste capítulo apresentar-se-ão exemplos do funcionamento das mesmas.

Assim, para ilustrar o funcionamento do SA apresenta-se de seguida um exemplo de um problema de minimização do *makespan* em *flowshop* com as seis primeiras tarefas do problema em estudo. Para tal partiu-se da solução inicial [1 2 3 4 5 6] com um *makespan* de 652 com uma temperatura inicial de $T_i=20$, um esquema de arrefecimento linear de β=5, um número de iterações à mesma temperatura de $L_n=5$ e um critério de paragem baseado na Temperatura atual (T≤10). Utilizou-se um mecanismo de vizinhança de troca de pares adjacentes. Na Tabela 8 encontra-se o procedimento da meta heurística SA.

Tabela 8 Exemplo de aplicação do SA

	Solução x	*f(x)*	*Solução x'*	*f(x')*	*p(x')*	*p(x')*
T=20	1 2 3 4 5 6	652	1 2 4 3 5 6	624	1	-
	1 2 4 3 5 6	624	2 1 4 3 5 6	721	0,0078	0,81
	1 2 4 3 5 6	624	1 4 2 3 5 6	610	1	-
	1 4 2 3 5 6	610	1 4 2 5 3 6	611	0,95	0,51
	1 4 2 5 3 6	611	4 1 2 5 3 6	611	1	-
T=15	4 1 2 5 3 6	611	4 1 2 5 6 3	626	0,37	0,55
	4 1 2 5 3 6	611	4 1 5 2 3 6	672	0,017	0,62
	4 1 2 5 3 6	611	4 1 2 3 5 6	610	1	-
	4 1 2 3 5 6	610	4 1 3 2 5 6	600	1	-
	4 1 3 2 5 6	600	4 1 3 2 6 5	572	1	-
T=10	4 1 3 2 6 5	572	1 4 3 2 6 5	572	1	-
	1 4 3 2 6 5	572	1 4 3 6 2 5	548	1	-
	1 4 3 6 2 5	548	4 1 3 6 2 5	516	1	-
	4 1 3 6 2 5	516	4 1 6 3 2 5	520	0,67	0,3
	4 1 6 3 2 5	520	4 6 1 3 2 5	546	0,074	0,48

Como se pode observar, ao contrário da Pesquisa Local que resultou num ótimo local com valor de 548 [1 4 3 6 2 5], o SA resultou numa solução [4 1 3 6 2 5] com um desempenho de 516. Vale a pena realçar que as probabilidades *p(x')* foram calculadas através da expressão 1 e as restantes de forma aleatória. É percetível a diminuição da probabilidade de aceitação de piores soluções à medida que a temperatura vai baixando.

3.7.3 Artificial Bee Colony

A meta heurística *Artificial Bee Colony* (ABC) enquadra-se no grupo de *Swarm Intelligence* (SI), o qual é constituído por meta heurísticas inspiradas no comportamento coletivo de uma colónia social de insetos e de outras sociedades animais que permitem a resolução de problemas de otimização. Proposta por Karaboga, D., (2005) e Pham, D. T., *et al.,* (2006), a meta heurística é inspirada no comportamento de uma colónia de abelhas na procura de fontes de alimento (Ghanbarzadeh, Koç, Otri, Rahim, & Zaidi, 2006; Karaboga, 2005). De acordo com A. B. G. S., (2015), a presente meta heurística só permitia a resolução de problemas de otimização contínuos, só mais tarde sugeriram versões do ABC que permitia a resolução de tais problemas, mas agora discretos, denominada de *Discrete Artificial Bee Colony* (DABC). Contudo, é possível aplicar meta heurísticas continuas em problemas discretos, para tal basta ter o cuidado de a codificar de modo a obter soluções discretas. De acordo com o mesmo autor, a diferença entre ABC e DABC reside apenas no modo como as abelhas se movimentam na vizinhança de uma fonte de alimentos. Na Tabela 9 é apresentada a estrutura habitual do ABC.

Tabela 9 Algoritmo da Artificial Bee Colony (A. B. G. S. e Santos, 2015)

```
Inicializar fontes de alimentos
While ≠ Critério de interrupção
        Fase das Abelhas Trabalhadoras
        Fase das Abelhas Oportunistas
        If Fonte de alimentos exausta
                Fase das abelhas trabalhadoras
        end
end
return melhor solução
```

A colónia de abelhas é constituída pela soma das abelhas trabalhadoras e oportunistas. Por norma, o número de abelhas trabalhadoras iguala o número de abelhas oportunistas, mas nem sempre é necessário utilizar uma proporção entre as mesmas de 50%. Numa primeira fase, cada abelha trabalhadora é enviada para uma fonte de alimento aleatória. As posições das fontes de alimento no algoritmo representam possíveis soluções do problema de otimização. Quanto à quantidade de néctar da fonte de alimento, a mesma corresponde à qualidade da solução representada por essa fonte de alimento (Karaboga, 2005). Posteriormente, cada abelha trabalhadora explora uma solução vizinha da sua fonte de alimento e comunica os resultados às abelhas oportunistas. Dado o desempenho de cada fonte de alimento, as abelhas oportunistas distribuem-se probabilissimamente pelas fontes de alimento. Quando as fontes de alimento são consideradas exaustas, as abelhas trabalhadoras convertem-se em abelhas exploradoras e procuram uma nova fonte de alimento aleatoriamente (Boussaïd et al., 2013; A. B. G. S. e Santos, 2015).

Ou seja, o ABC consiste nos três grupos de abelhas seguidamente apresentados

Abelhas Trabalhadoras

Inicialmente é gerada aleatoriamente, ou parcialmente aleatória, uma fonte de alimento (x_i) para cada abelha trabalhadora, i.e., o número de fontes de alimento é igual ao número de abelhas trabalhadoras. Cada abelha trabalhadora procura aleatoriamente uma solução vizinha (v_i) da sua fonte de alimento. Posteriormente, partilha a informação da respetiva fonte de alimento com uma certa probabilidade com as abelhas oportunistas (Karaboga, 2005).

Abelhas Oportunistas

As abelhas oportunistas esperam na colónia pela informação das abelhas trabalhadoras acerca da fonte de alimento. Posteriormente, escolhem probabilissimamente qual a fonte de alimento que irão explorar. Em Santos, A. B. G. S., (2015) é apresentada a expressão 6 para calcular a probabilidade de aceitação da fonte de alimento para

problemas de maximização, onde f_i representa o desempenho da fonte de alimento (x_i). Para problemas de minimização, é preciso ter em conta que o desempenho é representado por $1/f_i$, tal como apresentado na expressão 7.

$$pi = \frac{f_i}{\sum f_i} \text{ para problemas de maximização} \quad (6)$$

$$pi = \frac{1/f_i}{\sum 1/fi} \text{ para problemas de minimização} \quad (7)$$

Determinada a fonte de alimento, cada abelhas oportunistas explora uma solução vizinha da fonte de alimento onde se encontra.

Abelhas Exploradoras

Por fim, ocorre a fase das abelhas exploradoras, a qual consiste em explorar novas fontes de alimento quando uma fonte de alimento atual seja abandonada. Quando uma fonte de alimento é abandonada, a abelha trabalhadora da respetiva fonte converte-se em abelha exploradora e procura por uma nova fonte aleatoriamente. Uma fonte de alimento é abandonada quando ocorrem *l* iterações sem melhoria do desempenho. Santos, A. B. G. S., (2015) afirma que o número limite de iterações (*l*) está relacionado com o tamanho da colónia e apresenta uma métrica para o cálculo do mesmo, expressão 8.

$$l = \frac{cs \times D}{3} \quad (8)$$

Resumidamente, *l* elevado resulta em pesquisas mais intensas, enquanto valores pequenos de *l* permitem uma pesquisa mais diversa.

Para ilustrar o funcionamento do *DABC*, um exemplo de minimização do *makespan* em *flowshop* com as sei primeiras tarefas do problema em estudo será apresentado de seguida. Pra o exemplo assumiu-se um tamanho da colónia de 4 abelhas, com uma proporção de 50/50 entre abelhas trabalhadoras e oportunistas. Considerou-se ainda que uma fonte de alimento é abandonada ao fim de 2 iterações sem melhoria, o critério de paragem seria de 6 iterações e o mecanismo de vizinhança utilizado seria a troca de pares adjacentes. Na Figura 9 é possível analisar o funcionamento da *DABC*.

	Fontes e Abelhas	Soluções	f(x)		Fontes e Abelhas	Soluções	f(x)
1ª Iteração	F. Alimento A	3 4 5 1 2 6	630	2ª Iteração	F. Alimento A	3 4 5 1 2 6	630
	A. Trabalhadora	4 3 5 1 2 6	630		A. Trabalhadora	3 4 1 5 2 6	630
	A. Oportunista	3 5 4 1 2 6	630		A. Oportunista	3 4 5 1 6 2	574
					A. Oportunista	3 4 5 2 1 6	704
	F. Alimento B	2 3 6 4 1 5	698		F. Alimento B	3 2 6 4 1 5	656
	A. Trabalhadora	2 3 4 6 1 5	674		A. Trabalhadora	3 2 6 4 5 1	656
	A. Oportunista	3 2 6 4 1 5	656				
3ª Iteração	F. Alimento A	3 4 5 1 6 2	574	4ª Iteração	F. Alimento A	3 4 5 1 6 2	574
	A. Trabalhadora	3 5 4 1 6 2	574		A. Trabalhadora	3 4 5 6 1 2	578
	A. Oportunista	4 3 5 1 6 2	574		A. Oportunista	3 4 1 5 6 2	574
	A. Oportunista	3 4 5 1 2 6	630				
	F. Alimento B	3 2 6 4 1 5	656		F. Alimento C	1 5 3 6 2 4	636
	A. Trabalhadora	3 2 6 1 4 5	660		A. Trabalhadora	1 5 3 6 4 2	621
					A. Oportunista	1 3 5 6 2 4	636
5ª Iteração	F. Alimento D	4 3 1 2 6 5	572	6ª Iteração	F. Alimento D	4 3 1 6 2 5	559
	A. Trabalhadora	4 3 1 6 2 5	559		A. Trabalhadora	4 1 3 6 2 5	516
	A. Oportunista	4 3 1 2 5 6	600		A. Oportunista	4 3 6 1 2 5	520
	A. Oportunista	3 4 1 2 6 5	572				
	F. Alimento C	1 5 3 6 4 2	621		F. Alimento C	1 5 3 4 6 2	597
	A. Trabalhadora	1 5 3 4 6 2	597		A. Trabalhadora	1 3 5 4 6 2	574
					A. Oportunista	1 5 4 3 6 2	574

Figura 9 Exemplo de aplicação do DABC

Com a aplicação do *DABC* foi encontrada uma solução [4 1 3 6 2 5] com um *makespan* de 516, que por acaso coincide com a melhor solução encontrada através do SA. Tal como se referiu existe maior probabilidade de as abelhas oportunistas se dirigirem para fontes de alimento com melhor desempenho. No caso da 5ª iteração, existe uma probabilidade de 52% de as abelhas oportunistas se dirigirem para a Fonte de Alimento D. Destaca-se ainda que a Fonte de alimento B é abandonada na 3ª iteração e a Fonte de alimento A é abandonada na 4ª iteração.

3.8 Parametrização de Meta Heurísticas

Os parâmetros de uma meta-heurística têm muita influência na qualidade das soluções encontradas. Por norma, o processo de parametrização é habitualmente muito moroso, sendo que uma parte importante do esforço computacional de aplicação de uma meta-heurística. Os processos de parametrização podem ser divididos em (Montero, Riff, & Neveu, 2014; A. B. G. S. e Santos, 2015):

- **Parametrização manual:** Os parâmetros são alterados iterativamente pelo utilizador, na procura por uma combinação de parâmetros que resulte no melhor desempenho da técnica de otimização para o problema em causa;
- **Parametrização por analogia:** Os parâmetros são definidos pela expectativa do desempenho numa instância do problema, através da identificação do impacto dos parâmetros nas características da técnica de otimização;

- **Parametrização por planeamento de experiências:** Os parâmetros são definidos estatisticamente, através de uma análise dos valores de parâmetros no desempenho da técnica de otimização, num determinado problema;
- **Parametrização por pesquisa:** É feita uma pesquisa pelo espaço de parâmetros, através de pesquisa local ou outras meta heurísticas, na procura dos parâmetros que resultam no melhor desempenho num determinado problema;
- **Parametrização híbrida:** os parâmetros são definidos através de uma mistura de técnicas de parametrização. Por exemplo, pela combinação de parametrização por planeamento de experiências e parametrização por pesquisa.

3.8.1 Planeamento de experiências - Taguchi

Uma das técnicas mais utilizadas para a parametrização das meta heurísticas consiste na utilização do planeamento de experiências (*Design of experiments-DOE*). DOE é o nome dado à técnica usada para guiar a escolha das experimentações a serem realizadas de maneira eficiente. Normalmente, os dados estão sujeitos a erros experimentais (ruído), pelo que os resultados podem ser significativamente afetados pelo ruído. Assim, a melhor forma de analisar os dados passa pelo uso de métodos estatísticos (Cavazzuti, 2012).

Para realizar um DOE é necessário definir o problema e escolher as variáveis, as quais são chamadas de fatores ou parâmetros. O número de níveis deve também ser selecionado de acordo com o número de experiências que pode ser oferecido. Como níveis entende-se o número de valores diferentes que a variável pode assumir de acordo com as suas características. Por norma, o número de níveis é igual para todas as variáveis, no entanto, existem algumas técnicas de DEO que permitem a diferenciação de níveis para cada variável (Cavazzuti, 2012).

Os métodos de DOE foram desenvolvidos originalmente por Fisher. No entanto, os métodos clássicos de DOE são muito complexos e difíceis de utilizar. Além disso, o número de experimentações aumenta à medida que o número de parâmetros e níveis começa a aumentar (Yang & Tarng, 1998).

Em Cavazzuti, M. (2012) vários métodos de DOE são apresentados e discutidos, entre eles:

- Randomized Complete Block Design;
- Full Factorial;
- Fractional Factorial;
- Taguchi.

Neste trabalho decidiu-se utiliza o método de Taguchi, pelo que se irá abordar um pouco o funcionamento do mesmo.

O método de Taguchi é uma técnica bem conhecida, única e poderosa para a melhoria da qualidade do produto/processos. Possui uma ampla aplicação no design de

engenharia e pode ser aplicado em muitas áreas, tais como otimização, estimação de parâmetros, previsão, entre outros. O DOE de Taguchi é uma técnica estruturada e eficiente que difere do DOE clássico (*Full Factorial)* (Marković, Petrović, Ćojbašić, & Marinković, 2013).

O método de Taguchi foi desenvolvido por Taguchi no Japão para melhorar a implementação do controlo da qualidade total. O método está preparado para encontrar os melhores valores das variáveis controláveis e tornar o problema menos sensível ao ruído causado pelas variáveis incontroláveis. Tal método é baseado em níveis mistos, projetos fatoriais altamente fracionários e tabelas/matrizes ortogonais (Cavazzuti, 2012).

A principal dificuldade da aplicação do método Taguchi consiste na escolha da matriz ortogonal adequada. A literatura tem relevado diversas matrizes ortogonais, no entanto, um esquema completo que inclui todos as possibilidade de matrizes ortogonais, mesmo para um pequeno número de execuções experimentais, ainda não são conhecidas (Bolboacă, Jäntschi, Bolboacă, & Jäntschi, 2007).

Tradicionalmente, os dados das experimentações são utilizados para analisar o valor médio. No entanto, no método Taguchi a média e a variância do resultado experimental em cada configuração são combinadas na matriz ortogonal numa única medida de desempenho, *Signal-to-Noise (S/N)*. Para problemas de minimização, o S/N é calculado através da expressão 9 (Marković et al., 2013). O desempenho do parâmetro deverá num determinado nível é representado por *yi* e *n* o número de observações.

$$\frac{S}{N} = -10log\ (\frac{1}{n}\sum_{i=1}^{n} y_i^2) \qquad (9)$$

Para demonstrar o funcionamento do planeamento de experiência de Taguchi um pequeno exemplo será apresentado.

Imagine-se um cenário de utilização de uma meta-heurística com quatro parâmetros (P_1, P_2, P_3 e P_4) e 3 níveis para cada parâmetro, tal como mostra a Tabela 10. Caso se pretendesse realizar uma factorização completa, para este cenário seriam necessárias 81 (3^4) experiências, que no caso do método de Taguchi resume-se a 9 corridas. Ou seja, nestas condições recorre-se à tabela ortogonal L_9, apresentada na Tabela 11.

Tabela 10 Representação dos parâmetros e dos níveis do exemplo

	1	2	3
P_1	P_{11}	P_{12}	P_{13}
P_2	P_{21}	P_{22}	P_{23}
P_3	P_{31}	P_{32}	P_{33}
P_4	P_{41}	P_{42}	P_{43}

Tabela 11 Matriz ortogonal L9 de Taguchi (Zandieh, Amiri, Vahdani, & Soltani, 2009)

Corrida	P_1	P_2	P_3	P_4
1	1	1	1	1
2	1	2	3	2
3	1	3	2	3
4	2	1	3	3
5	2	2	2	1
6	2	3	1	2
7	3	1	2	2
8	3	2	1	3
9	3	3	3	1

Construindo então a matriz L9 com os valores dos níveis de cada parâmetro, tem-se a Tabela 12.

Tabela 12 S/N das experiências do exemplo

Corrida	P_1	P_2	P_3	P_4	F(x)	S/N
1	P_{11}	P_{21}	P_{31}	P_{41}	Z_1	$-10\log(Z_1^2)$
2	P_{11}	P_{22}	P_{33}	P_{42}	Z_2	$-10\log(Z_2^2)$
3	P_{11}	P_{23}	P_{32}	P_{43}	Z_3	$-10\log(Z_3^2)$
4	P_{12}	P_{21}	P_{33}	P_{43}	Z_4	$-10\log(Z_4^2)$
5	P_{12}	P_{22}	P_{32}	P_{41}	Z_5	$-10\log(Z_5^2)$
6	P_{12}	P_{23}	P_{31}	P_{42}	Z_6	$-10\log(Z_6^2)$
7	P_{13}	P_{21}	P_{32}	P_{42}	Z_7	$-10\log(Z_7^2)$
8	P_{13}	P_{22}	P_{31}	P_{43}	Z_8	$-10\log(Z_8^2)$
9	P_{13}	P_{23}	P_{33}	P_{41}	Z_9	$-10\log(Z_9^2)$

Os resultados de cada experiência são representados por Z_i. Para facilitar o cálculo do S/N, na Tabela 12, a fórmula foi decomposta, pelo que agora se torna necessário calcular a média para cada nível para obter os valores S/N na totalidade, Tabela 13.

Tabela 13 Seleção dos parâmetros

	P_1	P_2	P_3	P_4
1	$(S/N_1+S/N_2+S/N_3)/3$	$(S/N_1+S/N_4+S/N_7)/3$	$(S/N_1+S/N_6+S/N_8)/3$	$(S/N_1+S/N_5+S/N_9)/3$
2	$(S/N_4+S/N_5+S/N_6)/3$	$(S/N_2+S/N_5+S/N_8)/3$	$(S/N_3+S/N_5+S/N_7)/3$	$(S/N_2+S/N_6+S/N_7)/3$
3	$(S/N_7+S/N_8+S/N_9)/3$	$(S/N_3+S/N_6+S/N_9)/3$	$(S/N_2+S/N_4+S/N_9)/3$	$(S/N_3+S/N_4+S/N_8)/3$

Calculados os valores de S/N, os níveis de cada parâmetro com os valores mais elevados são selecionados.

APRESENTAÇÃO DO PROTÓTIPO

4 Apresentação do Protótipo

Neste capítulo será apresentado o protótipo desenvolvido no âmbito da presente dissertação. Serão apresentados com o detalhe o seu conceito e as suas funcionalidades.

4.1 O Conceito

O protótipo desenvolvido pretende facilitar a gestão do planeamento da produção e a própria comunicação no chão de fábrica. Ou seja, esta aplicação foi desenvolvida com o intuito de ilustrar como é que uma qualquer alteração do plano pode ser efetuada e todas as partes interessadas podem ser notificadas em tempo real. Além disso, o presente projeto pretende ainda ilustrar como é que o plano de produção poder ser adaptado, tendo em conta os seus objetivos, sempre que eventos dinâmicos ocorram na produção. Isto é, a ferramenta desenvolvida é capaz de escalonar o WIP (*Work In Progress*) de todos os postos de trabalho de forma autónoma sempre que novas tarefas dão entrada, escalonamento esse que tem em conta as medidas de desempenho classificadas no modelo de Kano.

Portanto, a ferramenta desenvolvida foi projetada com o objetivo de não requerer qualquer interação com o utilizador além da definição dos critérios de desempenho. A definição dos critérios de desempenho, como foi referido, é classificada pelo grau de satisfação do Modelo de Kano. A Figura 10 exemplifica como as medidas de desempenho podem ser classificadas na ferramenta. Para os objetivos definidos na Figura 10, presume-se que o utilizador não deseja que o atraso máximo ultrapasse um determinado valor e, ao mesmo tempo, deseja que o tempo médio de fluxo seja o menor possível. Como a Figura 10 ilustra, o atraso máximo representa um atributo "*Must be*" no modelo de Kano, o que significa que caso não seja alcançado, provocará uma insatisfação extrema ao utilizador. Quanto ao Tempo Médio de Fluxo, o mesmo representa um atributo proporcional, pois corresponde a um grau de satisfação proporcional ao grau de desempenho do atributo, ou seja, quanto menor, melhor, neste caso (Högström, Rosner, & Gustafsson, 2010; Löfgren, Witell, & Gustafsson, 2011).

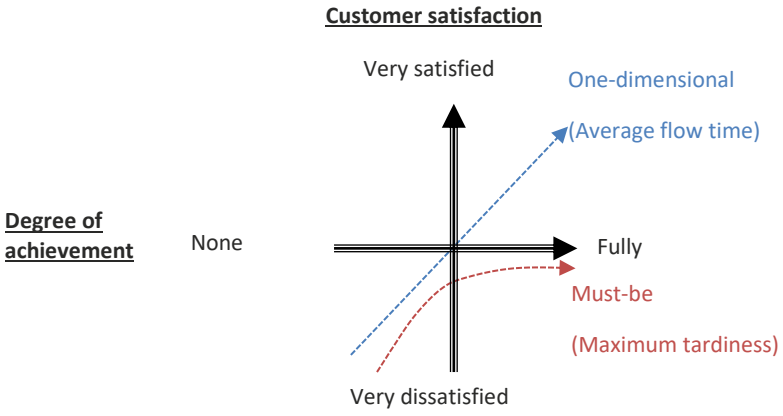


Figura 10 Exemplos de medidas de desempenho classificadas através do modelo de Kano (Luís Ferreirinha et al., 2020)

Após a definição dos critérios de desempenho, o próprio software escalona e reescalona as tarefas no sistema ao longo do tempo, na ocorrência de um evento em tempo real, como por exemplo a anulação de uma tarefa ou a alteração nas datas de entrega e assim por diante. Para gerar soluções aceitáveis em tais circunstâncias, a ferramenta encontra-se conectada ao MRP da empresa para gerar um plano preditivo usando as informações disponíveis e, em seguida, sempre que ocorrerem eventos no sistema durante a execução do plano pré-estabelecido, o reescalonamento é executado. Como o protótipo está conectado ao MRP e na maioria das vezes o MRP é executado periodicamente, sabe-se que podem ocorrer distúrbios entre dois períodos de tempo imediatos, o que nos leva a uma estratégia híbrida, ou seja, reescalonar o sistema periodicamente e também quando alguns eventos particulares acontecem.

Quando o MRP é executado, um grande número de tarefas geralmente entra no sistema, portanto a procura por uma solução satisfatória pode ser demorada e dificilmente permanecerá viável no ambiente industrial. Assim, a ferramenta proposta gera o plano preditivo por meta heurísticas, que pode lidar com problemas de larga escala e encontrar soluções satisfatórias dentro de um período de tempo razoável. Vale a pena referir que o ideal seria que todas as tarefas lançadas no período T do MRP terminassem antes do período T + 1. No entanto, tal cenário provavelmente não ocorrerá em ambientes industriais, portanto, quando o sistema atinge T + 1, a MH irá programar as novas tarefas que o MRP irá lançar, assim como as tarefas que ainda não foram processadas (WIP) (Luís Ferreirinha et al., 2020).

A parametrização da meta heurística é feita através da sensibilidade do utilizador (de modo manual), algo que futuramente poderá ser alterado para, por exemplo, um DOE, onde os valores que irão competir serão calculados por métricas ou gerados aleatoriamente dentro de intervalos que foram mostrados ser eficazes na resolução de problemas deste tipo na comunidade do escalonamento. Veja-se por exemplo (Eglese, 1990; Kirkpatrick et al., 1983; Park & Kim, 1998; Talbi, 2009) onde algumas métricas e intervalos são apresentados para o *Simulated Annealing* (SA). Sempre que ocorram interrupções/ eventos, o protótipo reprograma as tarefas com regras de prioridade. Um breve resumo do que foi descrito aqui é mostrado na Figura 11.

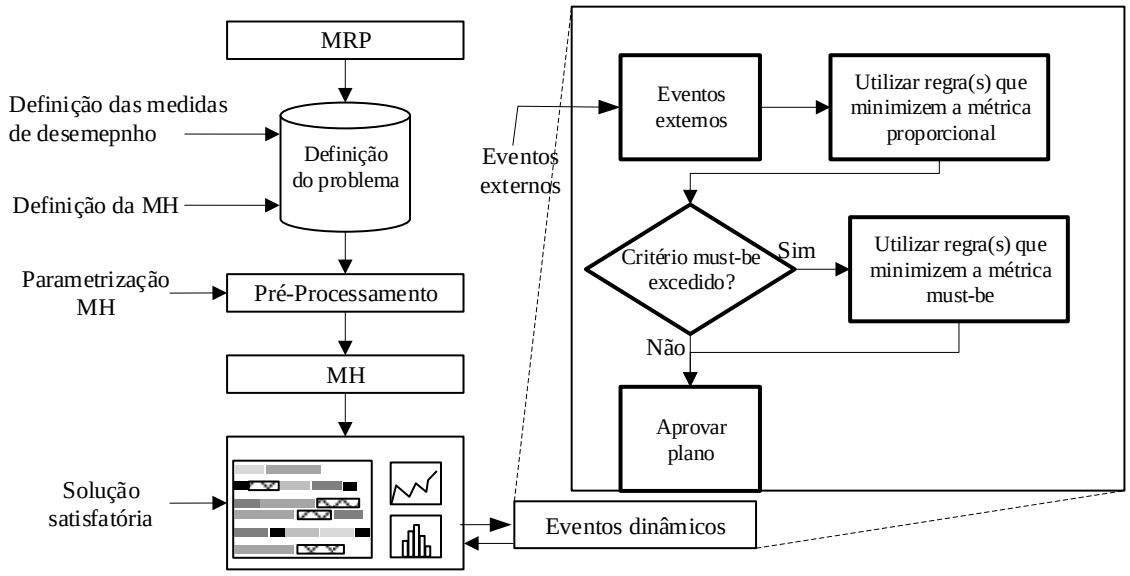


Figura 11 Esquema da lógica do protótipo (adaptado de Luís Ferreirinha et al., 2020)

Na Figura 11 é mostrada a lógica do protótipo. Quando o MRP é executado, o utilizador precisa de definir as medidas de desempenho e a MH que irá gerar um plano preditivo. Caso os critérios de otimização não sejam definidos, a ferramenta irá manter as escolhas anteriores. Após a parametrização, um plano preditivo é gerado e, em seguida, é libertado para o chão de fábrica por um painel (geral) onde vários indicadores de desempenho são apresentados, bem como as tarefas seguintes que irão entrar em curso. Quando eventos dinâmicos ocorrem, o módulo "Eventos dinâmicos" reprograma as tarefas. Apesar do protótipo apresentado idealizar uma ferramenta totalmente autônoma, haverá situações onde o gestor, devido à sua experiência e conhecimento do processo produtivo, poderá querer validar planos alternativos, e tal pode ser efetuado em qualquer momento. Através da ferramenta desenvolvida, o gestor pode sempre comparar planos alternativos, gerados pelo próprio, com planos gerados automaticamente pelo modelo.

A principal diferença do protótipo apresentado aqui em relação a outras metodologias na literatura, é que é independente do ambiente da máquina. Ou seja, o modelo desenvolvido tanto funcionará em ambientes de máquina única como em ambientes *flow-shop* e máquinas paralelas, sendo que neste último caso, devido a questões de tempo, não foi possível parametrizar o escalonamento no ambiente de máquinas paralelas, será algo a desenvolver como trabalho futuro. Grande parte da literatura que analisa o escalonamento dinâmico apresenta metodologias específicas de problemas e não um modelo genérico adaptável a vários cenários. Por exemplo, em Kundakcı, N., & Kulak (2016) os autores apresentam metodologias eficientes de Algoritmo Genético Híbrido (GA) com uma nova heurística (KK com base nos nomes dos autores) e regras de despacho bem conhecidas, para minimizar o makespan em problemas dinâmicos em ambientes *job-shop* (Kundakcı & Kulak, 2016). Em Ahmadi, E., et al. (2016) os autores propuseram uma metodologia multi-objetiva para o problema de planeamento em ambiente FJSP para situações de falha/ avarias de máquina (Ahmadi, Zandieh, Farrokh,

& Emami, 2016). Outro exemplo pode ser analisado em Rahmani, D., & Ramezanian, R. (2016) onde os autores propuseram um algoritmo de Pesquisa de Vizinhança Variável (VNS) para resolver um problema dinâmico de problema em Flexible Flow Shop (FFS) considerando a chegada de novas tarefas (Rahmani & Ramezanian, 2016). Embora as metodologias apresentadas pelos autores acima mencionados sejam eficientes para os problemas em questão, dificilmente são imediatamente aplicáveis a outros ambientes de máquinas.

4.2 Base de Dados

Para uma flexibilidade acrescida por parte da ferramenta desenvolvida, decidiu-se criar uma base de dados para guardar as informações mais relevantes para o protótipo. Na Figura 12 encontra-se o diagrama Entidade-Relação (DER) da base de dados desenvolvida já na 3ª fase de normalização, ou seja, sem atributos não-atómicos (1ª fase), sem dependências parciais (2ª fase normal) e sem dependências transitivas (3ª fase).

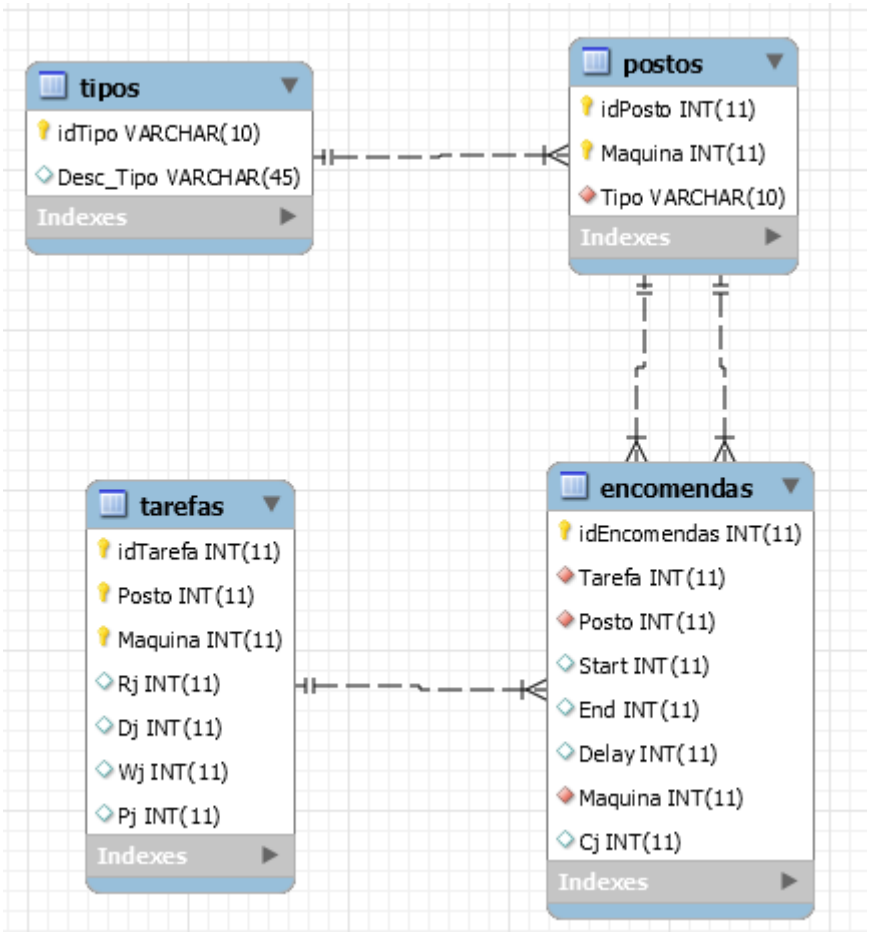


Figura 12 Diagrama Entidade-Relação da Base de Dados

Tal como se pode observar no DER existem quatro entidades: tarefas, postos, tipos e encomendas. De seguida os objetivos de cada um.

- **Tarefas:** Permite guardar informações sobre as tarefas criadas pelo utilizador. A informação armazenada passa pelo número da tarefa, pelo posto onde a tarefa pode ser trabalhada e as características da respetiva tarefa para cada máquina constituinte do posto de trabalho, tal como o tempo de processamento, o prazo, entre outros;
- **Posto:** Permite alocar informação sobre os postos de trabalho criados pelo utilizador. A identificação do posto de trabalho, o número de máquinas presentes no mesmo e o tipo/ ambiente do respetivo posto, são as informações armazenadas na base de dados;
- **Tipo:** Permite identificar o ambiente máquina associado a um determinado posto;
- **Encomendas:** Permite armazenar informações sobre as tarefas em produção, ou seja, de um determinado número de encomenda é possível saber que tarefa está a ser produzia, em que posto, quando iniciou/ terminou a produção, quando é que terminou numa determinada máquina e ainda o atraso associado a essa encomenda.

Criada a base de dados, passou-se ao desenvolvimento do protótipo por forma a manipular a mesma através de uma entidade externa.

4.3 Tarefas

A página principal da ferramenta desenvolvida pode ser visualizada na Figura 13. Nesta última é possível observar que, à semelhança da base de dados, o protótipo conta com três grandes campos de operação, sendo eles: tarefas, postos de trabalho e escalonamento.

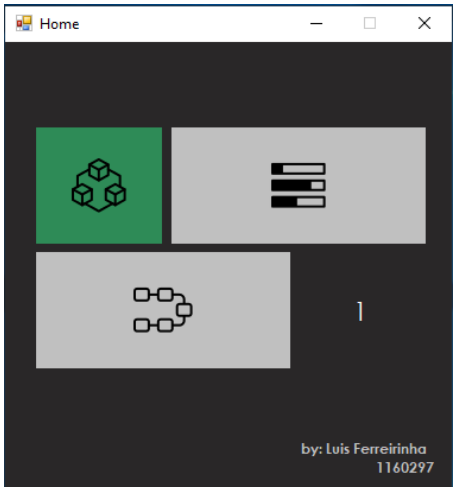


Figura 13 Página Inicial do Protótipo

Neste ponto irar-se-á abordar com detalhe o campo das tarefas, onde se irá demonstrar como é que o utilizador pode adicionar, remover ou atualizar tarefas através da aplicação.

4.3.1 Inserir Tarefas

Para inserir novas tarefas através do protótipo é necessário indicar o respetivo número e indicar em que posto de trabalho é que a mesma irá ser inserida. Ao clicar no campo "Tarefa", aplicação irá devolver de imediato uma proposta para o número da nova tarefa, posteriormente basta selecionar o posto pretendido e clicar no botão "criar novo", conforme ilustra a Figura 14.

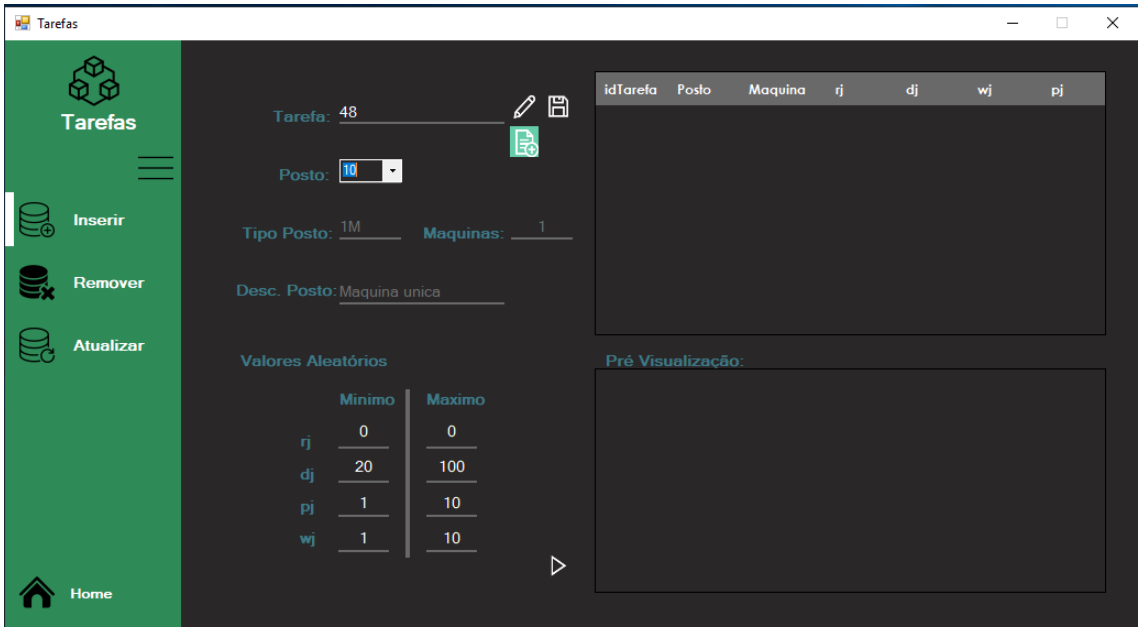


Figura 14 Inserção de uma nova tarefa num posto de trabalho vazio

Criada a nova tarefa, a aplicação irá preencher ambas as janelas existentes. A primeira diz respeito à forma como a tarefa está alocada no posto de trabalho escolhido, uma vez que se trata de uma tarefa criada de raiz, as características da mesma estarão com os valores por defeito. Na segunda janela "Pré-Visualização", é onde o utilizador pode editar essas mesmas características referidas anteriormente. Pode editá-las manualmente ou de uma forma aleatória com base nos limites estabelecidos no campo "Valores aleatórios" seguido do botão "*play*". Tal funcionalidade de aleatoriedade foi desenvolvida para gerar as tarefas que serão utilizadas para validar o modelo desenvolvido. Definidas as características pretendidas, basta clicar no ícone "guardar", Figura 15, que a primeira janela e a base de dados irão atualizar em conformidade.

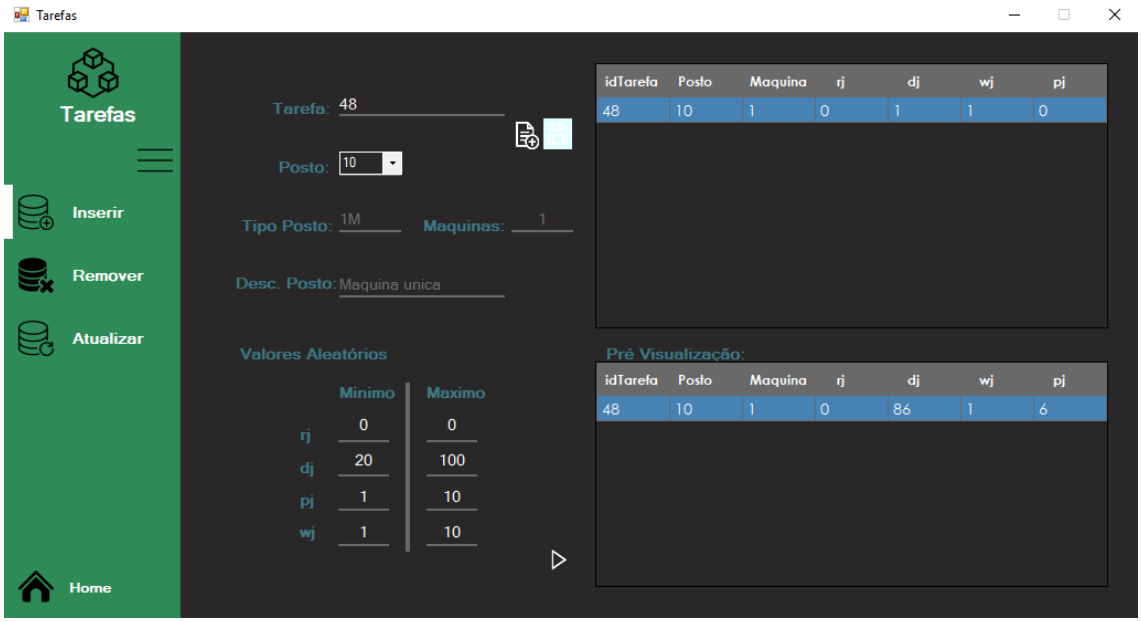


Figura 15 Guardar tarefa criada de raiz

No caso do posto de trabalho já possuir tarefas alocadas, a inserção de novas tarefas torna-se mais simples. Basta definir a respetiva referência e selecionar uma das tarefas existentes no respetivo posto de trabalho, aquela que talvez se assemelhará mais com a nova tarefa, seguido de pressionar o botão “copiar estilo”, Figura 16. Neste caso a aplicação irá copiar a informação da tarefa selecionada e irá criar uma tarefa com base nas características da primeira.

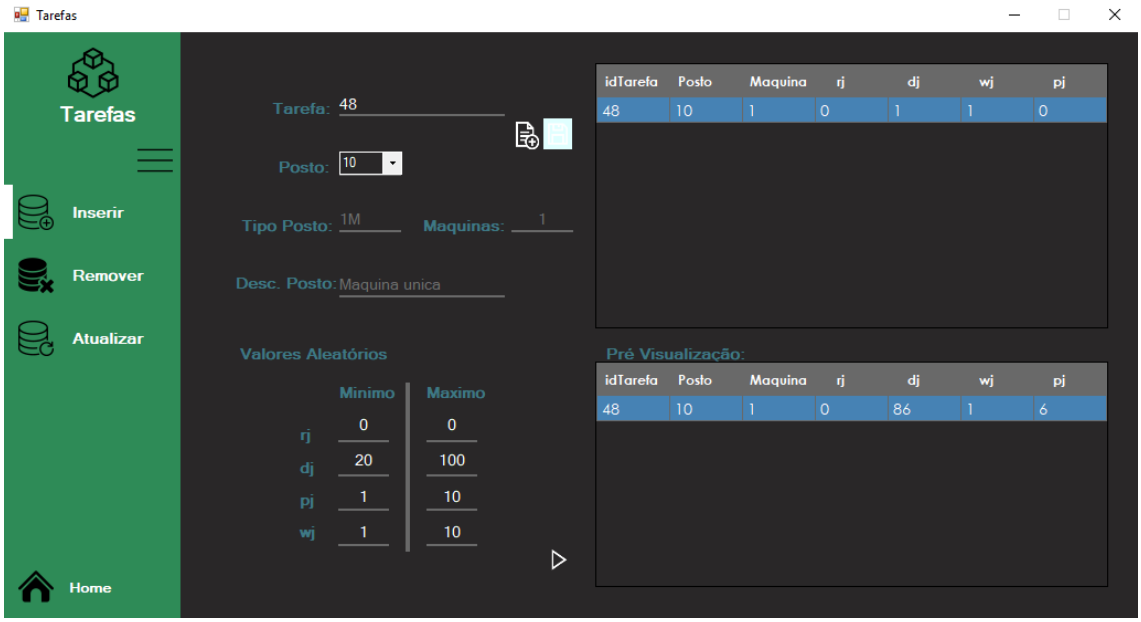


Figura 16 Criação de uma tarefa com base noutra

Definidas as características pretendidas para a nova tarefa, basta clicar no botão "Guardar", Figura 17.

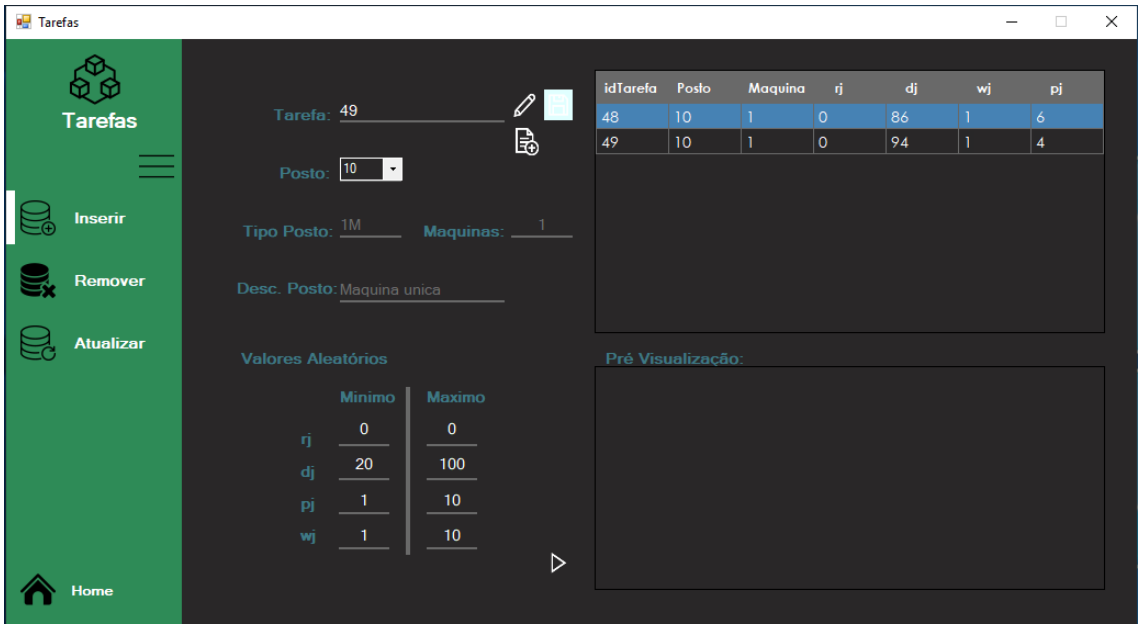


Figura 17 Nova tarefa adicionada ao respetivo posto de trabalho

4.3.2 Remover Tarefas

A remoção de tarefas da base de dados através da aplicação é bastante simples. Existem duas formas para procurar a tarefa que se pretende eliminar, ou se pesquisa diretamente pelo número de identificação da tarefa, ou seleciona-se o posto de trabalho ao qual a mesma está alocada. Nesta última opção a aplicação irá devolver todas as tarefas associadas ao respetivo posto, Figura 18.

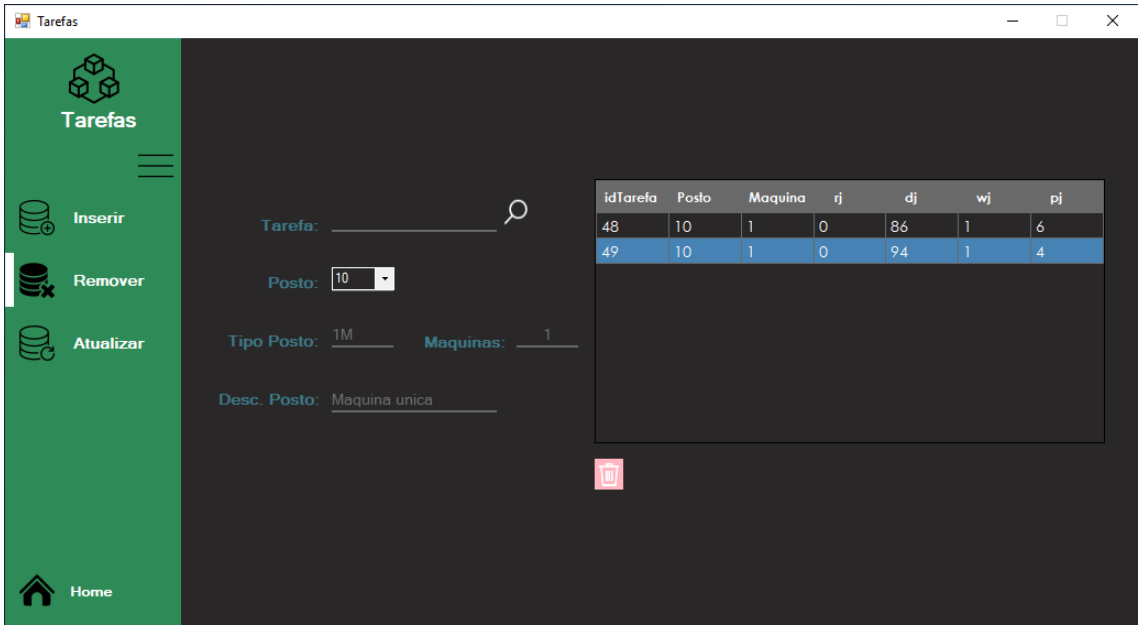


Figura 18 Procura da tarefa através do posto de trabalho

Posteriormente basta selecionar a tarefa pretendida e clicar no ícone “Lixo”, que a tarefa será removida, Figura 19.

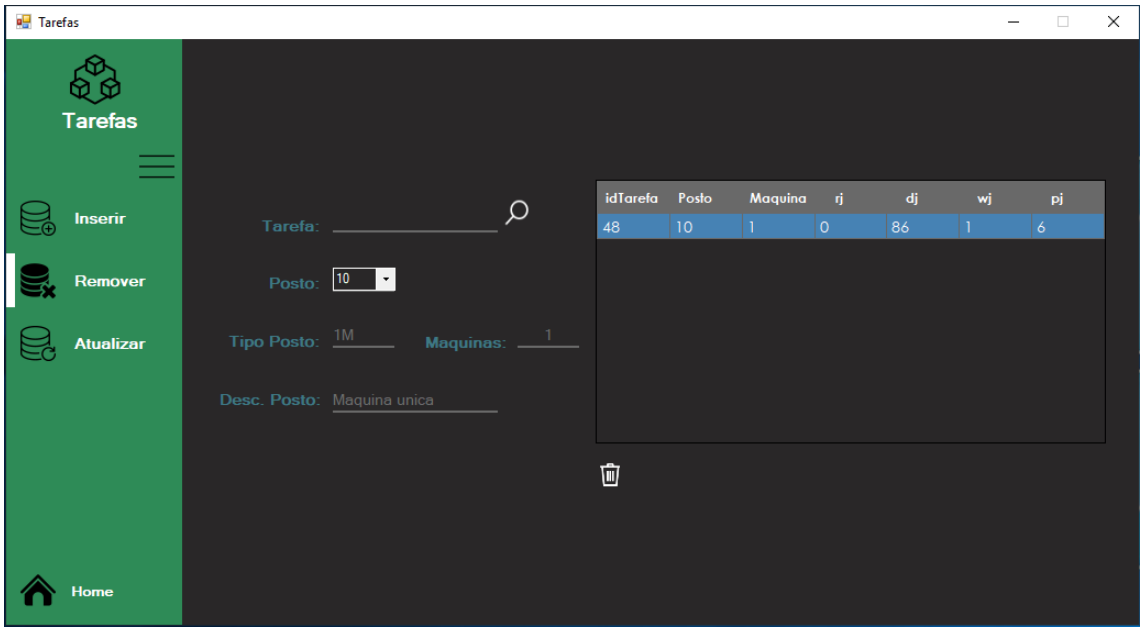


Figura 19 Remoção da tarefa através da procura por posto de trabalho

4.3.3 Atualizar Tarefas

O processo de atualização das características de uma determinada tarefa acaba por ser uma junção dos dois processos anteriormente explicados. Numa fase inicial procura-se a tarefa a atualizar, que à semelhança do campo remoção, tanto pode ser através de uma pesquisa individual como através do respetivo posto de trabalho, seguido da atualização das características, Figura 20.

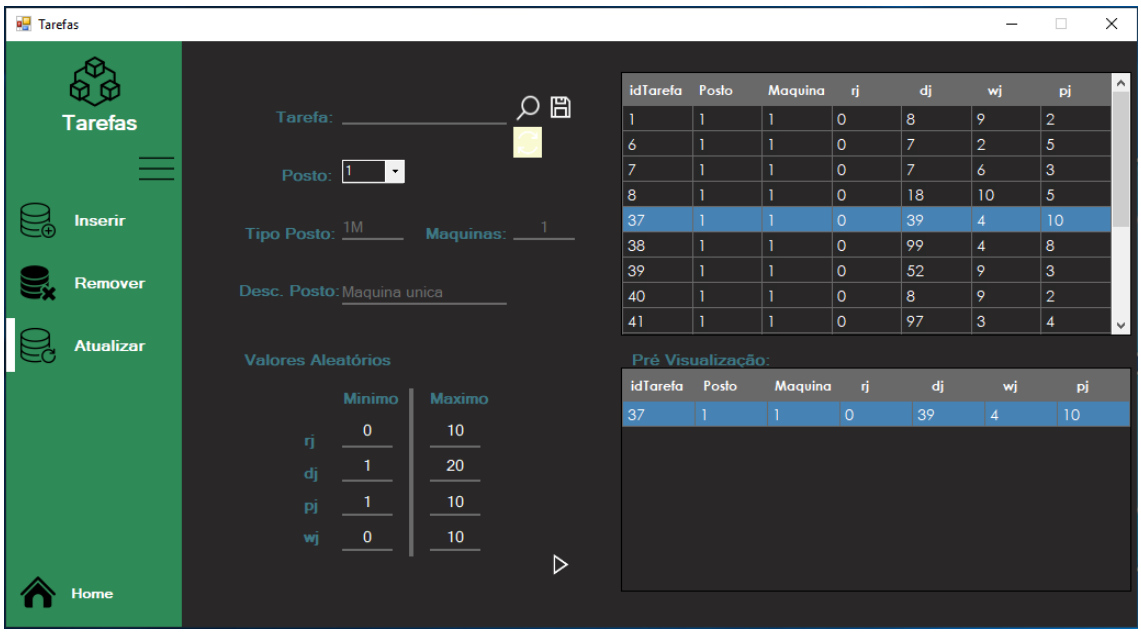


Figura 20 Procura da tarefa através do posto de trabalho

Selecionada a tarefa pretendida, basta editar as características conforme pretendido seguido da opção “Guardar”, onde a aplicação irá atualizar a respetiva tarefa, Figura 21.

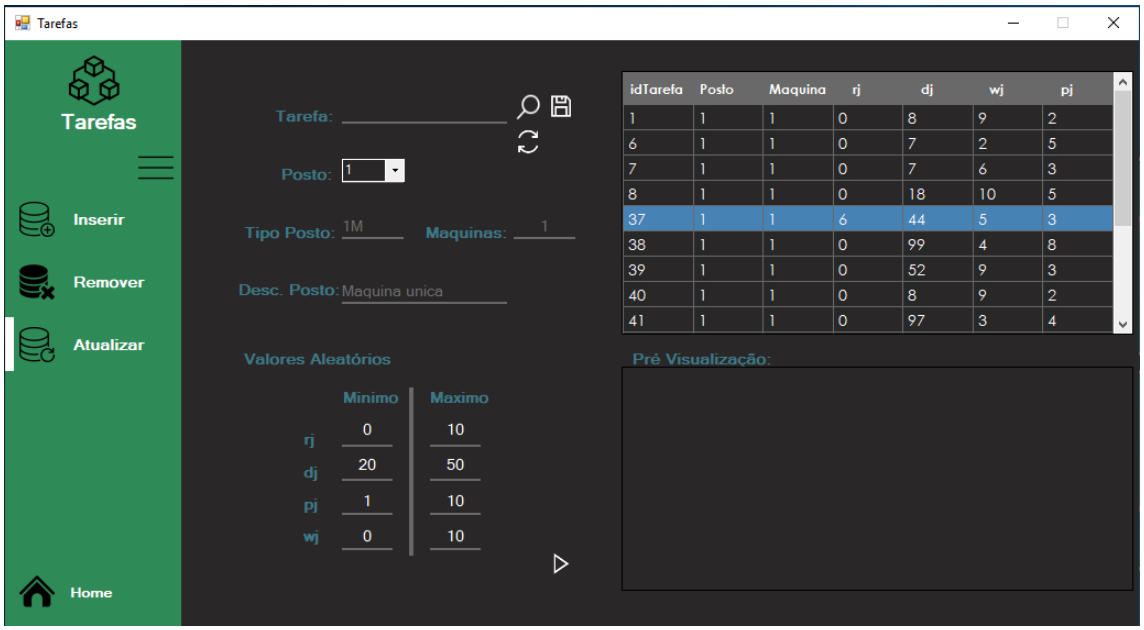


Figura 21 Guardar tarefa atualizada

4.4 Postos de Trabalho

Neste ponto irar-se-á abordar com detalhe o campo dos postos de trabalho, onde se irá demonstrar como é que o utilizador pode adicionar, remover ou atualizar postos de trabalho através da aplicação, Figura 22.

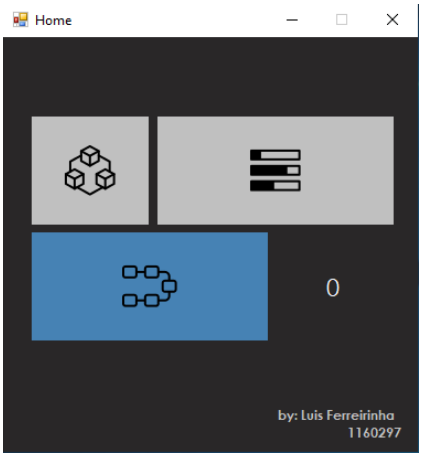


Figura 22 Campo postos de trabalho

4.4.1 Inserir Postos de Trabalho

À semelhança da criação de novas tarefas, ao clicar no campo “Posto”, a aplicação irá devolver uma sugestão para a identificação do mesmo. Posteriormente basta selecionar o tipo de posto que se pretende criar bem como o número de máquinas constituinte, Figura 23. Nesta fase inicial, a aplicação permite a criação de postos de trabalho de ambiente máquina única, ambiente *flow shop* e máquinas paralelas.

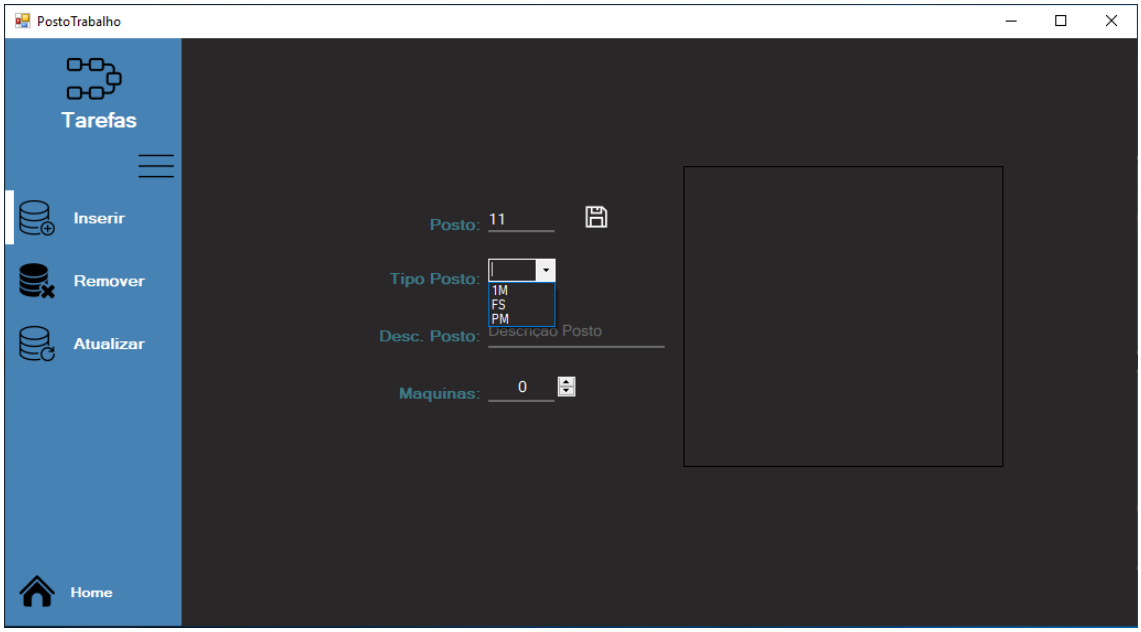


Figura 23 Tipos de postos de trabalho

Assim que se seleciona o tipo de posto pretendido, o protótipo devolve os respetivos postos já existentes na base de dados que correspondem ao ambiente escolhido, apenas para título informativo. Após reunir todas as informações acima mencionadas para a constituição de um novo posto de trabalho basta guardar que o software irá adicionar o novo elemento à base de dados, Figura 24.

Figura 24 Guardar posto de trabalho

4.4.2 Remover Postos de Trabalho

O processo de remoção de postos de trabalho através da aplicação é em tudo semelhante ao processo de remoção de tarefas anteriormente referido, Figura 25. Contudo, como se pode observar pelo DER no ponto 4.2, o posto de trabalho funciona

como chave-primária quer para a entidade tarefas quer para a entidade encomendas. Tal significa que não é possível eliminar um posto de trabalho se existir ou tarefas ou encomendas associadas ao mesmo.

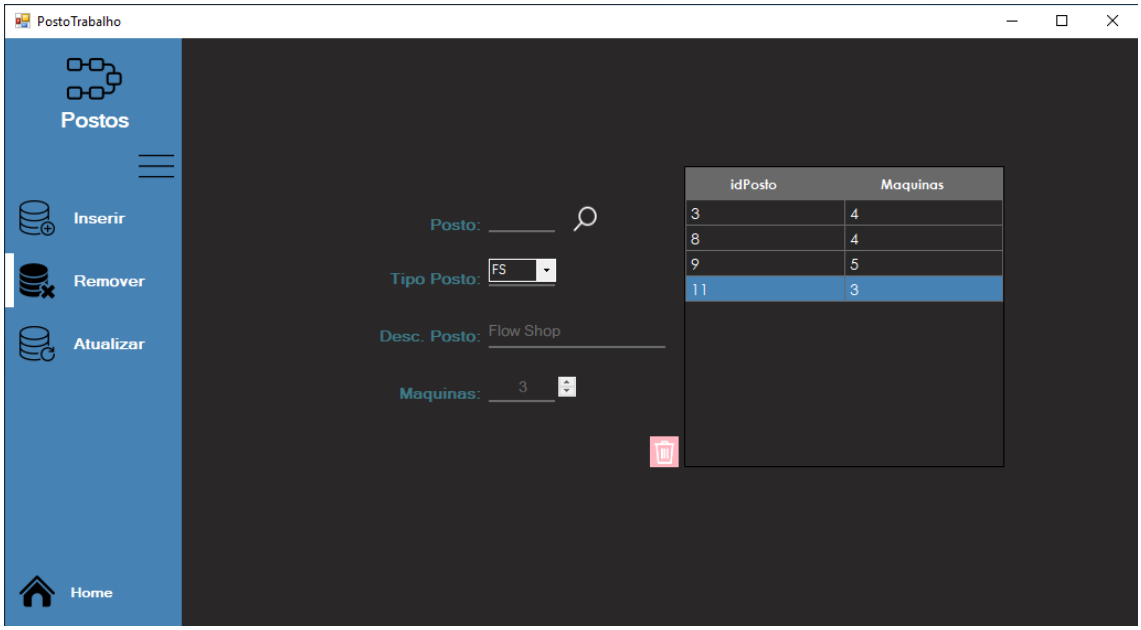


Figura 25 Remoção de um posto de trabalho

No caso de existirem encomendas associadas ao posto de trabalho, a aplicação informará, Figura 26, e não irá remover o posto até que o utilizar as cancele. Caso não existam encomendas e existam tarefas, a aplicação irá informar o utilizador e autonomamente irá remover as tarefas alocadas ao respetivo posto de trabalho, Figura 27. Por fim irá remover por completo o posto de trabalho.

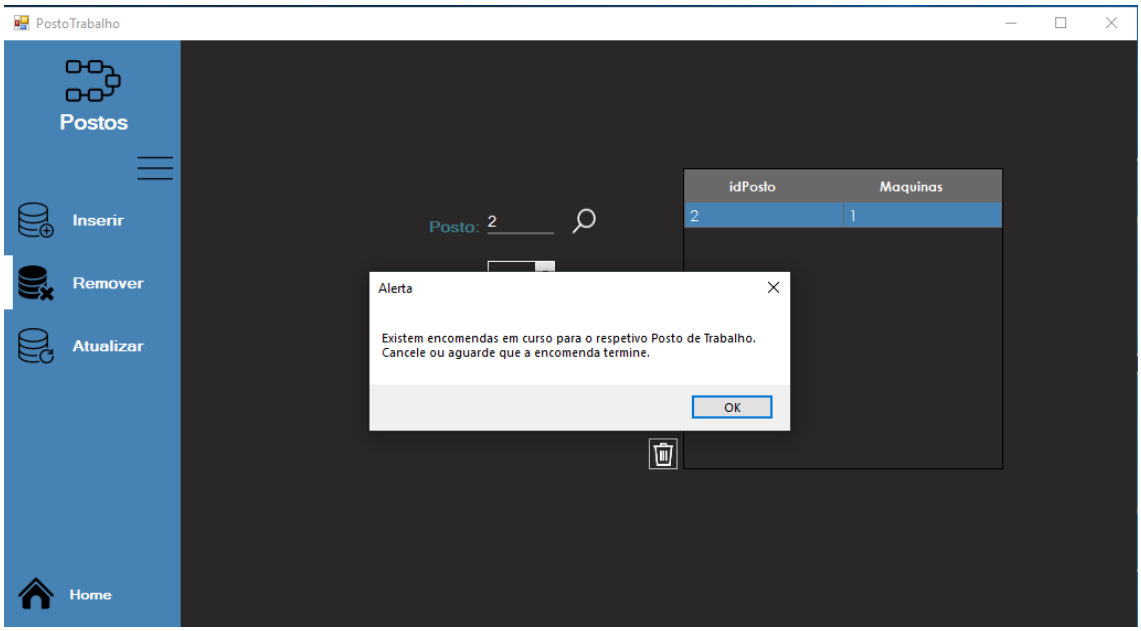


Figura 26 Informação sobre a existência de encomendas associadas ao posto de trabalho

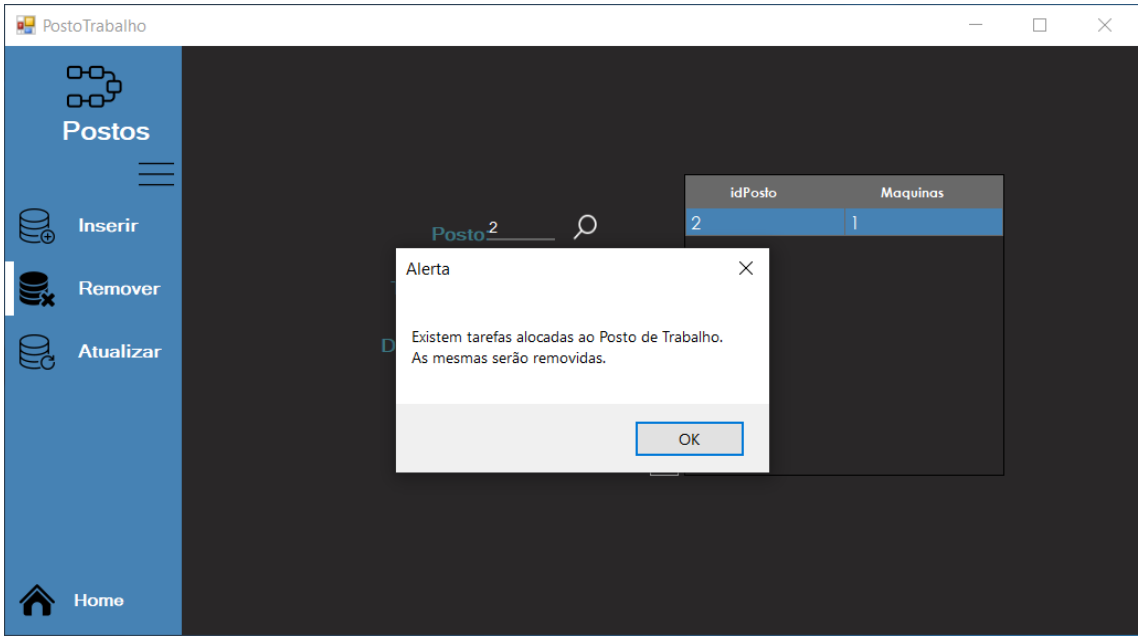


Figura 27 Informação sobre existência de tarefas alocadas ao posto de trabalho

4.4.3 Atualizar Postos de Trabalho

No que diz respeito ao processo de atualização de postos de trabalho, o mesmo é um pouco mais complexo que os restantes. Neste caso a única coisa que o utilizador pode atualizar é o número de máquinas associadas ao posto de trabalho. Para uma melhor explicação da lógica da remoção de postos de trabalho, veja-se o seguinte exemplo onde se pretende atualizar o posto 9 do tipo *flow shop*.

Como se pode observar pela Figura 28 o posto 9 possui duas tarefas, 25 e 27, com características definidas para todas as máquinas do respetivo posto, que neste caso são seis.

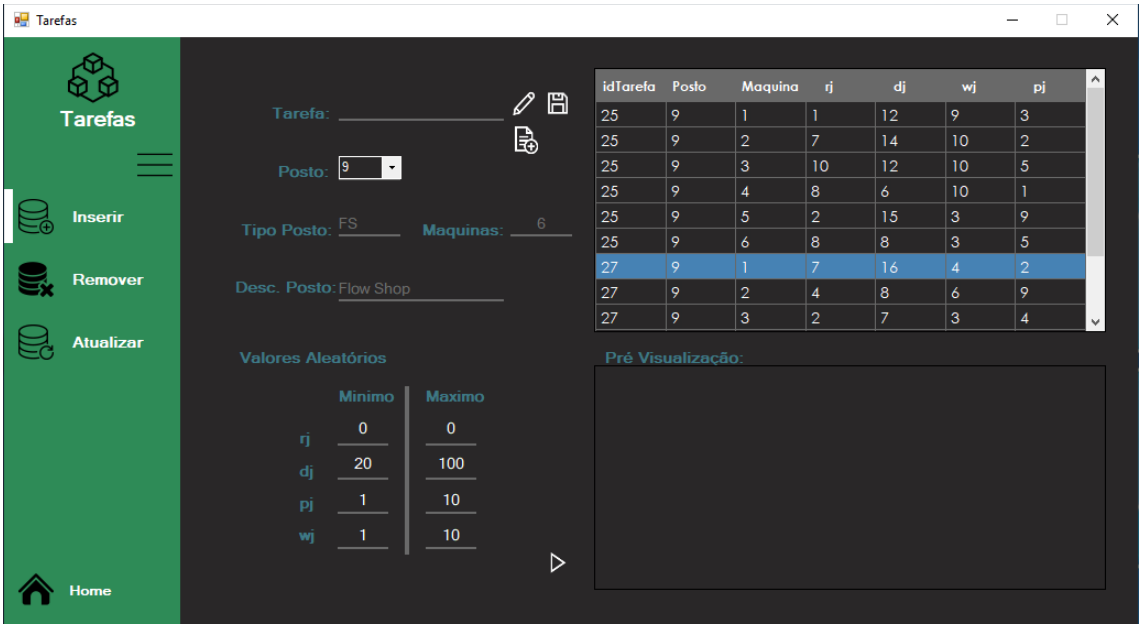


Figura 28 Tarefas alocadas ao posto de trabalho 9 com cinco máquinas

Neste primeiro exemplo pretende-se ajustar o número de máquinas de seis para três máquinas então, estando no campo de atualização de postos de trabalho basta indicar o posto de trabalho pretendido bem como definir o novo número de máquinas que passará a existir, seguido do ícone “atualizar”, Figura 29.

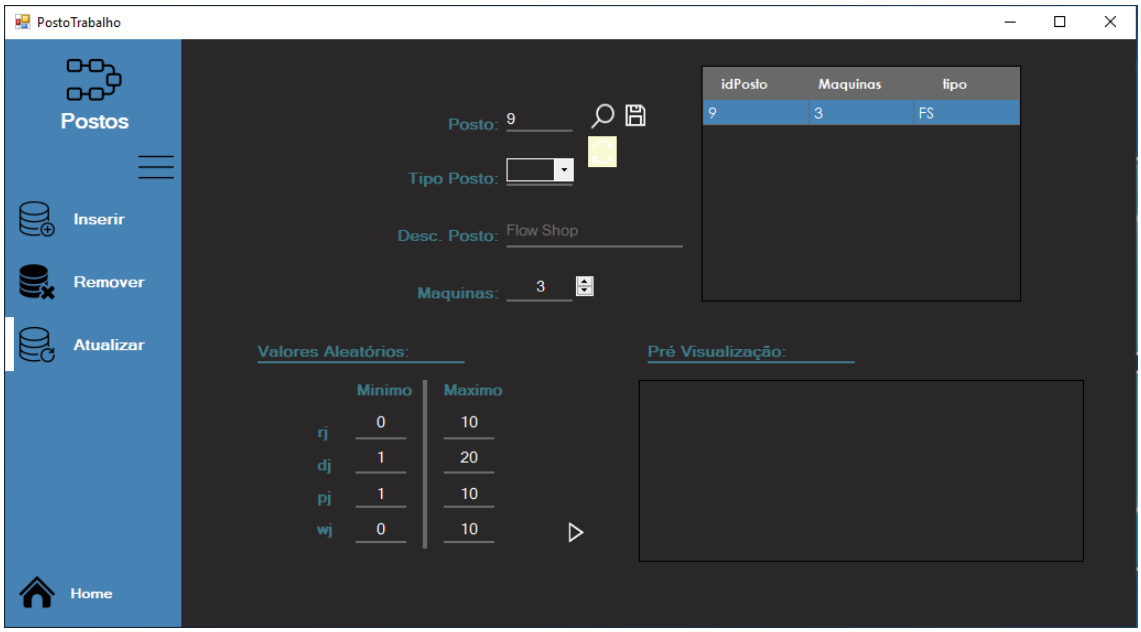


Figura 29 Diminuição do número de máquinas do posto 9

Uma vez que se trata de uma remoção de máquinas, a aplicação remove de todas as operações alocadas as máquinas que foram removidas. Ou seja, uma vez que o posto de trabalho 9 deixou de ter seis máquinas para passar a ter apenas três, a informação relacionada com as máquinas 4, 5 e 6 foi removida da base de dados tal como se pode verificar na Figura 30.

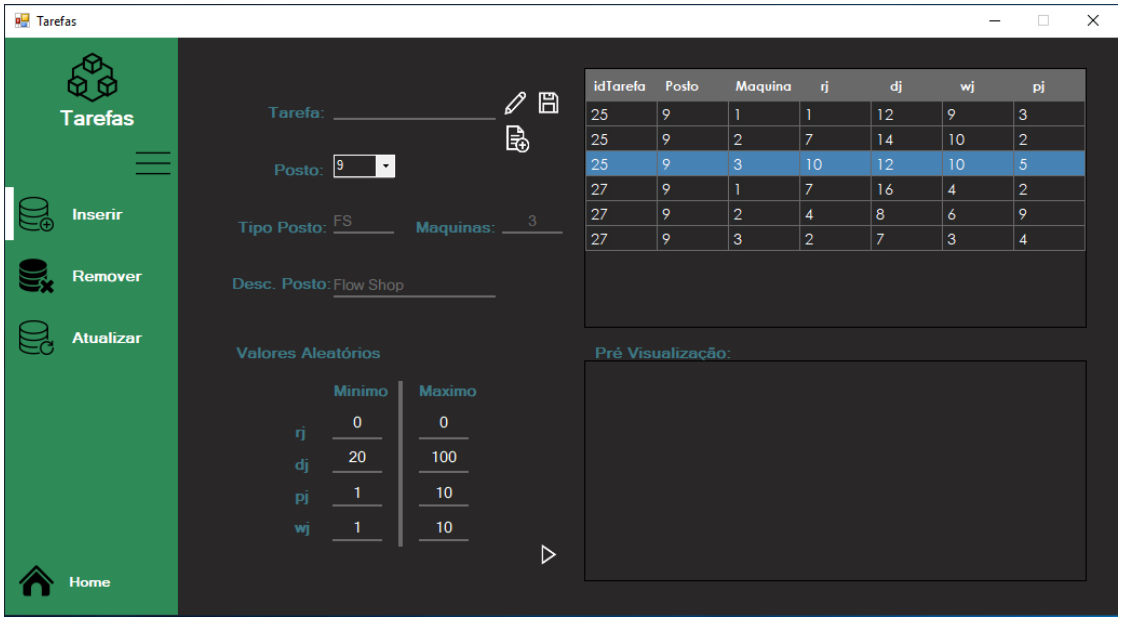


Figura 30 Tarefas alocadas ao posto de trabalho 9 com três máquinas

De seguida demonstra-se o processo inverso, ou seja, onde se adicionam máquinas a um posto de trabalho, que para facilitar a interpretação será novamente o 9. No exemplo que se segue irar-se-á adicionar mais duas máquinas, totalizando cinco. Seguindo o mesmo raciocínio de seleção do posto de trabalho pretendido e definido o novo número de máquinas, assim que o utilizador pressionar o botão “atualizar”, a aplicação irá mostrar na janela de “Pré-visualização” todas as tarefas associadas ao posto de trabalho com a informação das novas máquinas que entraram no sistema, Figura 31. Ou seja, uma vez que o número de máquinas do posto de trabalho aumentou de três para cinco, a informação que é mostrada na janela de “Pré-visualização” diz respeito às características das máquinas quatro e cinco.

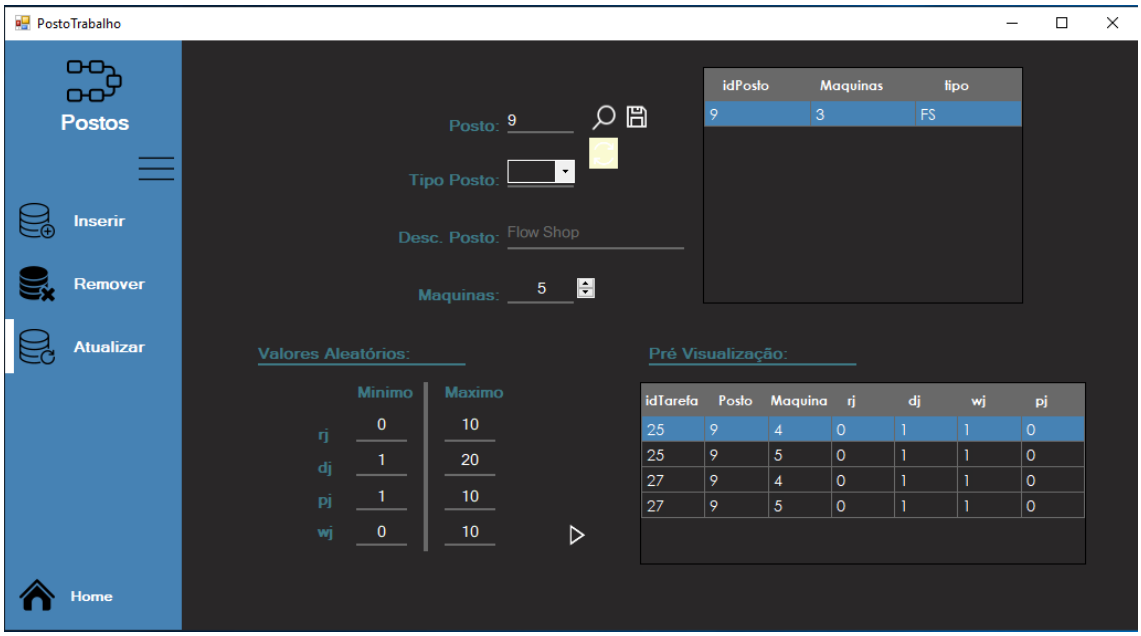


Figura 31 Adição de máquinas no posto 9

De notar que neste momento todas as tarefas associadas ao posto 9 já possuem informação sobre as cinco máquinas, contudo, as últimas máquinas associadas estarão com valores padrão, pelo que caso o utilizador não indique os respetivos valores, a informação será mantida conforme foi criada. O processo de definição da informação das respetivas tarefas é semelhante aos processos anteriormente referidos seguido do botão “Guardar”, Figura 32.

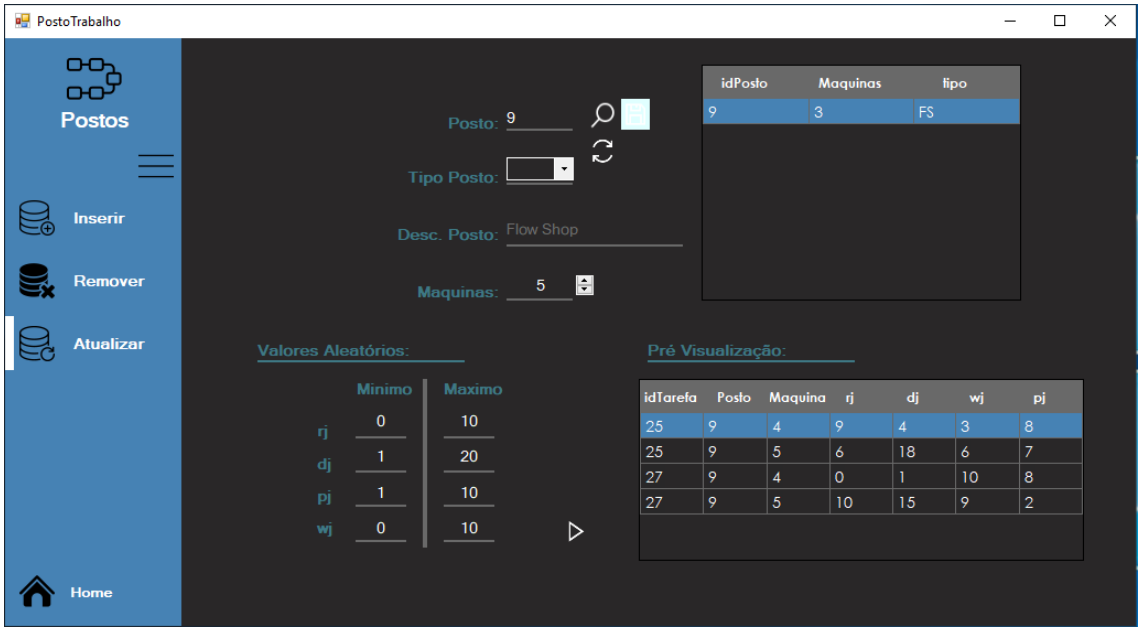


Figura 32 Atualização da informação das tarefas das máquinas adicionadas

Conforme se pode observar pela Figura 33, as tarefas associadas ao posto de trabalho 9 já contém a informação anteriormente atualizada relativamente às máquinas quatro e cinco.

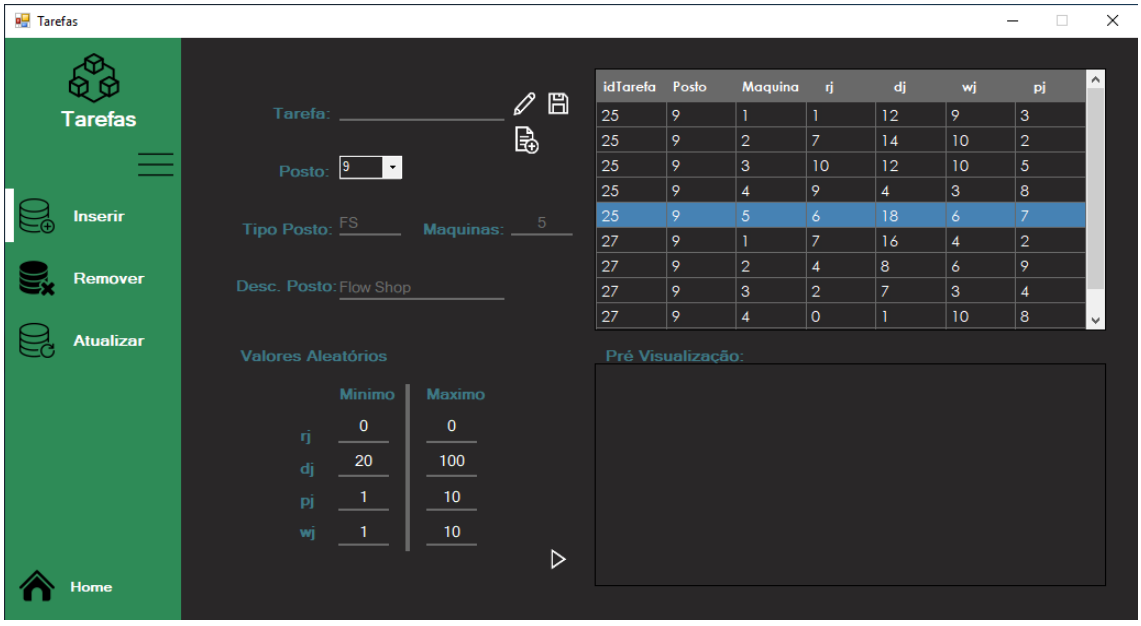


Figura 33 Tarefas associadas ao posto 9 com a informação sobre as máquinas adicionadas

4.5 Produção

Criados os postos de trabalho e as respetivas tarefas, está-se agora em condições de proceder à criação de encomendas e respetiva produção das tarefas, Figura 34.

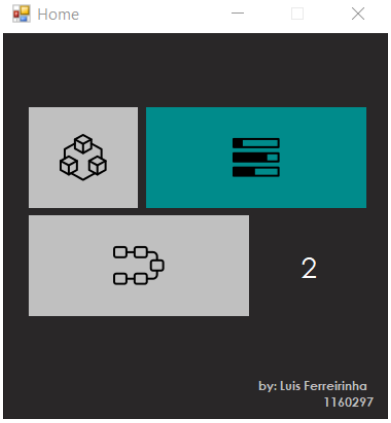


Figura 34 Campo escalonamento

Ao selecionar o campo escalonamento, o utilizador irá deparar-se com a janela da Figura 35, a qual mostra o estado da produção em todos os postos com encomendas. Neste caso os únicos postos com encomendas eram o 1, o 2 e o 3, estando eles com 40%, 33% e 75% do plano cumprido, respetivamente.

Figura 35 Campo estado da produção

Para indicar um novo posto para iniciar a produção basta selecioná-lo seguido de pressionar o botão “adicionar”, Figura 36.

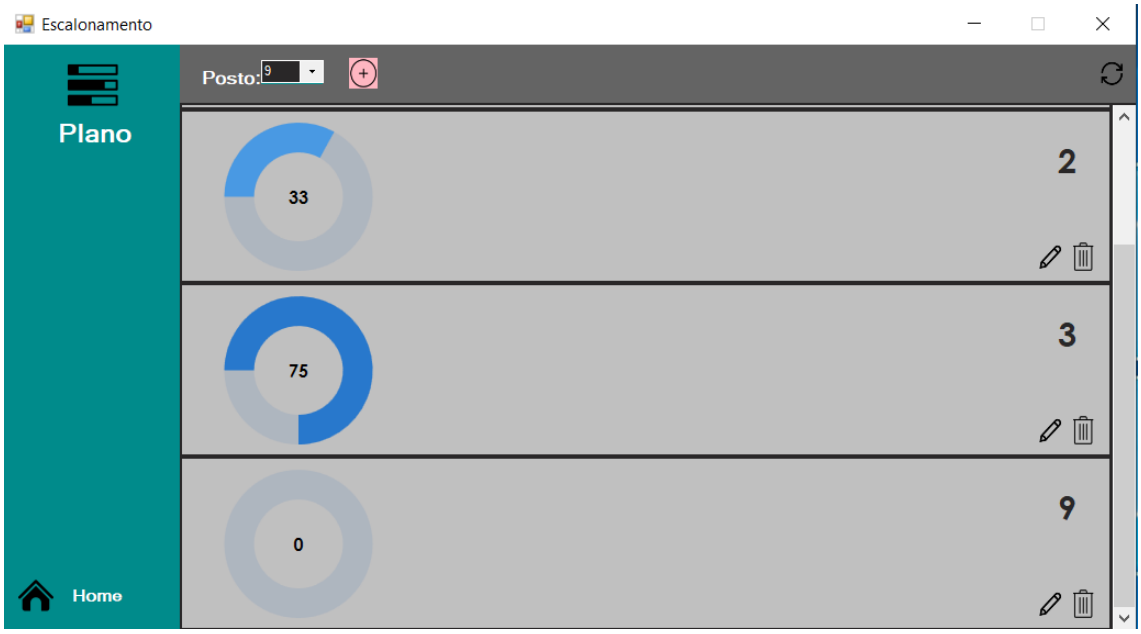


Figura 36 Adição de um posto de trabalho para produção

A paragem de um determinado posto, seja por cancelamento da produção seja por término do plano, pode ser feita através do ícone “Eliminar”, Figura 37.

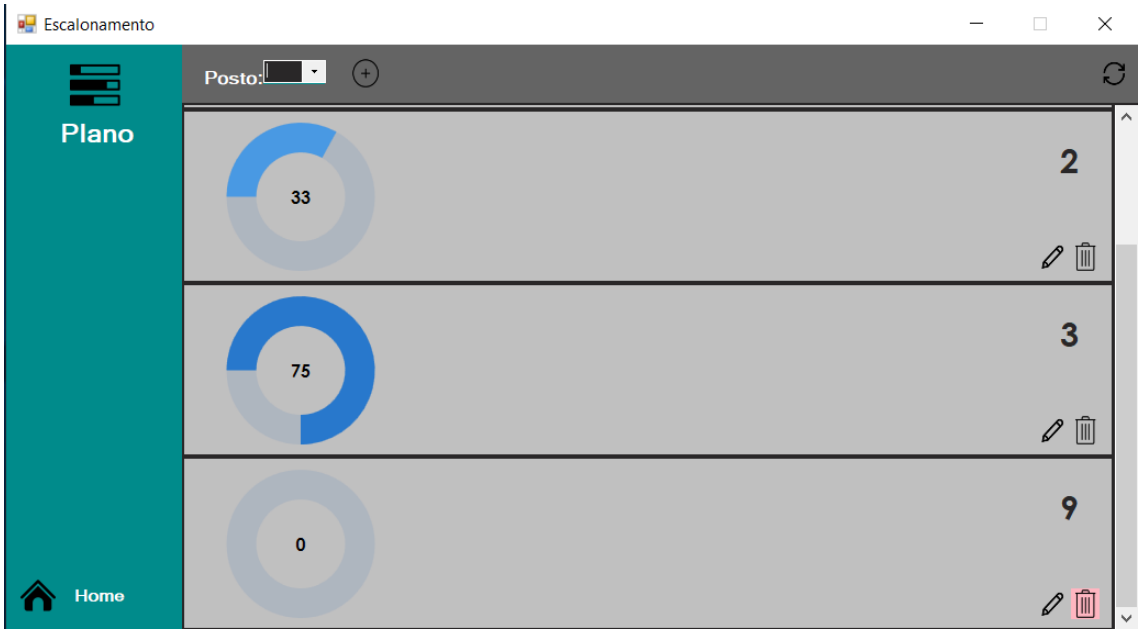


Figura 37 Remoção de um posto de trabalho da produção

Para ver em detalhe um determinado posto de trabalho bem como criar encomendas para o mesmo, escalonar as tarefas existes e parametrizar a autonomia de escalonamento basta clicar no botão “editar”, Figura 38.

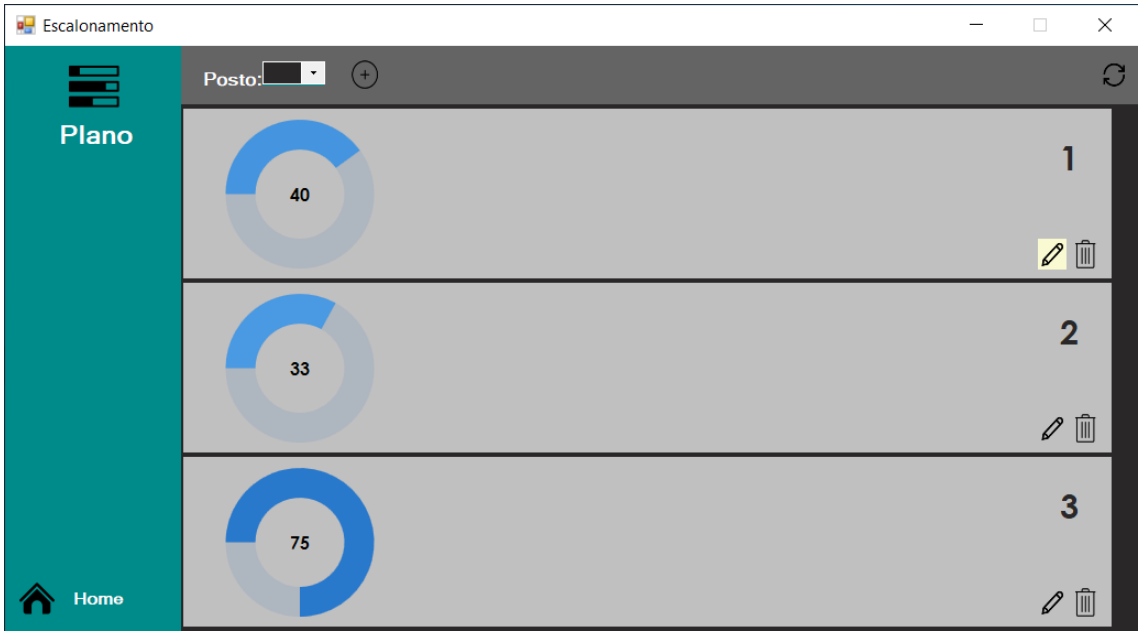


Figura 38 Ver em detalhe a produção do posto de trabalho

4.6 Escalonamento

Neste campo de operação do protótipo, é onde o utilizador pode escalonar as tarefas inerentes aos respetivos postos de trabalho segundo algumas regras de prioridade e meta heurísticas. Nele também é possível observar globalmente a performance do plano atual bem como o que vem a seguir para produzir, o que pode ser útil não só para o gestor como para qualquer operador fabril, principalmente o(s) que está(ão) a operar no respetivo posto de trabalho. No campo de escalonamento o utilizador pode adicionar e/ou remover encomendas associadas ao posto de trabalho, bem como parametrizar a ferramenta para escalonar as tarefas de forma autónoma segundo os critérios definidos no modelo de Kano.

4.6.1 Planeamento

Ao clicar na opção “editar” num determinado posto de trabalho no campo “Escalonamento”, o utilizador é encaminhado para o campo “Produção” do mesmo, onde a primeira janela diz respeito ao plano de produção em vigor, Figura 39.

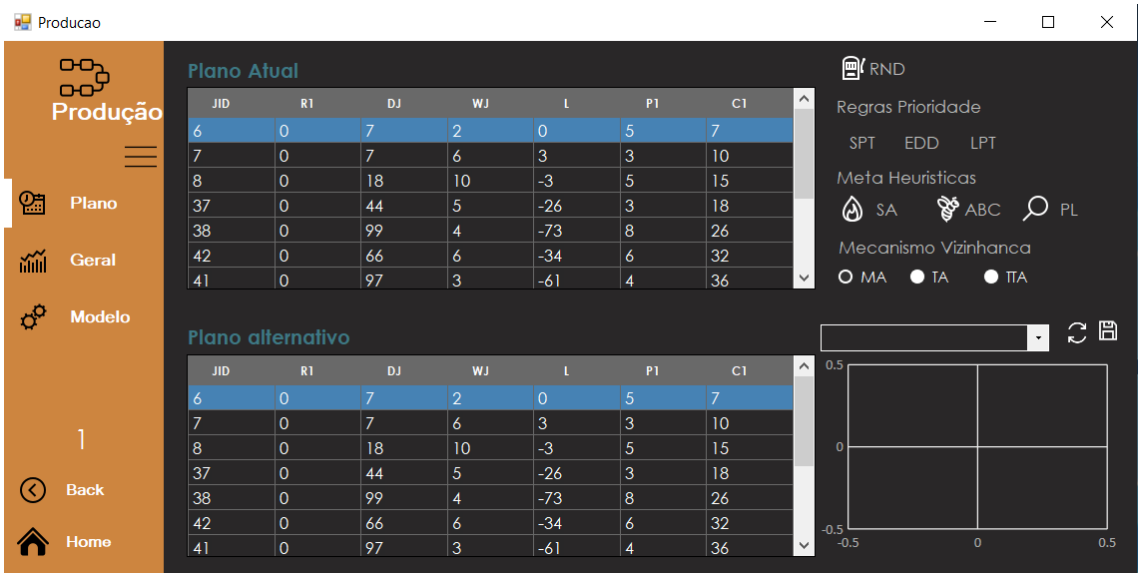


Figura 39 Campo plano

Neste campo do protótipo o utilizador tem a possibilidade de escalonar as tarefas no chão de fábrica de acordo com algumas regras de prioridades, como o SPT, EDD e LPT (*Longest ProcessingTime*) ou através de meta heurísticas como é o caso da Pesquisa Local (PL), do *Simulated Annealing (SA)* e da *Artificial Bee Colony (ABC)*.

Caso o utilizador pretenda observar que impacto é que um determinado escalonamento terá no plano atual basta selecionar a medida de desempenho que se pretende otimizar que a aplicação irá comparar de um modo gráfica ambos os planos, conforme ilustra a Figura 40.

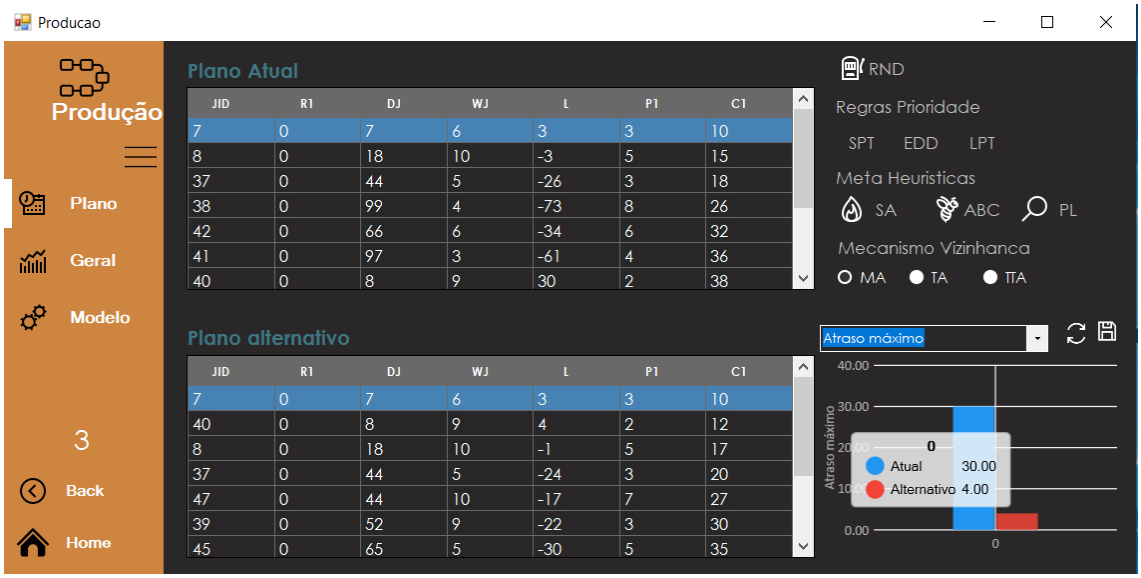


Figura 40 Comparação entre planos de produção

No caso da Figura 40 pretendia-se otimizar o atraso máximo das tarefas em chão de fábrica, e uma vez que o posto de trabalho selecionado tratava-se de um ambiente de máquina única, escalonaram-se as tarefas de acordo com a regra de prioridade que

minimiza tal medida de desempenho, ou seja, o EDD. Como se pode observar o plano alternativo apresenta um atraso máximo (4) inferior ao plano atual (30). Assim, caso se pretenda mudar o plano atual para o alternativo encontrado, basta guardar o mesmo que o corrente plano irá alterar, Figura 41.

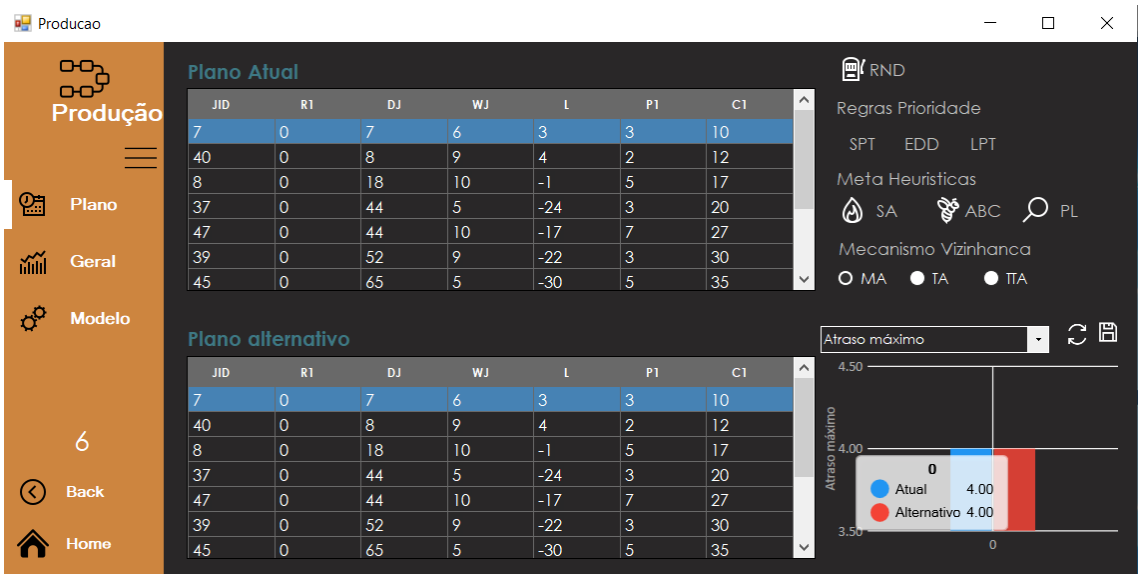


Figura 41 Atualização do plano atual

De seguida irar-se-á ilustrar o procedimento para utilizar as meta heurísticas como método de escalonamento das tarefas.

Tal como se reviu na revisão da literatura no ponto 3.3, a utilização de uma meta heurística carece de um ponto de partida, ou seja, de uma solução inicial, por modo a permitir ao método de otimização explorar soluções na sua vizinhança. No caso da aplicação, o utilizador partirá sempre que o utilizador se deslocar ao campo “Plano”, a solução em vigor é carregada para o plano alternativo, ou então partir dum uma solução baseada numa das regras de prioridade, ou ainda partir de uma solução aleatória, Figura 42.

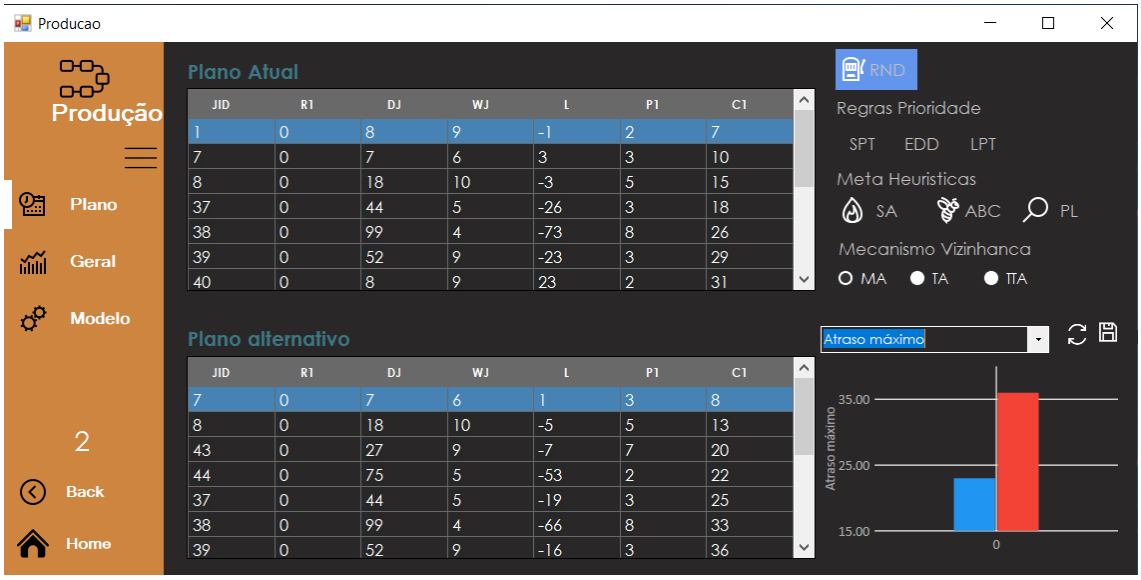


Figura 42 Geração de uma solução aleatória

Definida a medida de desempenho que se pretende otimizar, que para os seguintes exemplos será o atraso máximo, é necessário definir o modo como a meta heurística irá explorar as soluções vizinhas da solução inicial. Por defeito estará selecionado o mecanismo de vizinhança, "Mover Aleatório" (MA). Neste primeiro exemplo selecionou-se o mecanismo "Troca de Tarefas Adjacentes" (TTA) e a PL como metodologia de otimização de uma solução escalonada pela regra LPT com um atraso máximo de 45, Figura 43.

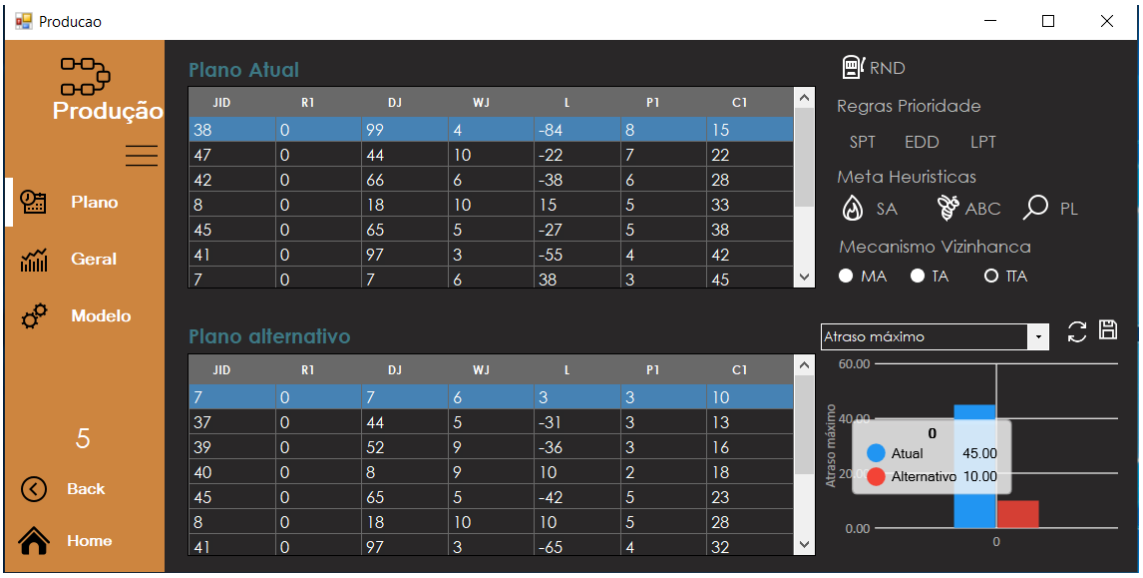


Figura 43 Seleção da pesquisa local como método de otimização

Assim, podemos reparar, na Figura 43, que a pesquisa local reduziu o atraso máximo para 10 unidades temporais, sendo que a solução ótima para o mesmo plano era 4, tal como se verificou na Figura 41. É importante referir que o ambiente no qual o protótipo trabalha trata-se de um ambiente dinâmico, ou seja, a qualquer momento podem dar entrada e/ou saída de tarefas para o respetivo plano. Ou seja, caso exista um plano onde

o atraso máximo seja de 3 e a tarefa que está a provocar esse mesmo atraso seja a próxima a entrar em produção, o próximo plano do respetivo posto de trabalho deixará de ter um atraso máximo de 3, pois a respetiva tarefa deixou de fazer parte do conjunto de tarefas no chão de fábrica. A tarefa em causa terminará, de facto, com atraso, mas o plano de produção restante terá um novo atraso máximo. Tal pode ser observado nesta mesma janela à medida que o tempo avança. O tempo está indicado acima da opção "*Back*" e assim que atingir um *Cj* de uma qualquer tarefa em curso que esteja num qualquer posto de trabalho, a próxima tarefa no plano entra em produção. A atual tarefa em curso pode ser observada no campo "Geral", como se verá adiante. Vale ainda a pena referir que, apesar de se estar a analisar o posto 1, Figura 38, todos os postos de trabalho com encomendas são atualizados em simultâneo, ou seja, estar a analisar o posto de trabalho 1, não implica que o posto de trabalho 2 e 3 estejam parados, pois tal não acontece nas industrias. Ou seja, à medida que se analisa o posto 1, ou qualquer outro posto, o tempo vai passando, pelo que sempre que o tempo atinge um "*cj*" de qualquer posto de trabalho, a ferramenta coloca a próxima tarefa do respetivo posto em curso, atualiza os critérios de desempenho e o progresso do respetivo posto, sem qualquer interação do utilizador e sem causar qualquer distúrbio na análise do posto em questão, neste caso, o posto 1.

Usando, ainda, o exemplo da Figura 41, onde o atraso máximo era de 4 e a próxima tarefa a entrar em produção era a tarefa 7. Sabendo que a tarefa em curso tem um Cj de 7, podemos observar pela Figura 44 que a tarefa 7 já entrou em produção e como a mesma não era a que provocava um atraso máximo de 4, esse mesmo valor manteve-se.

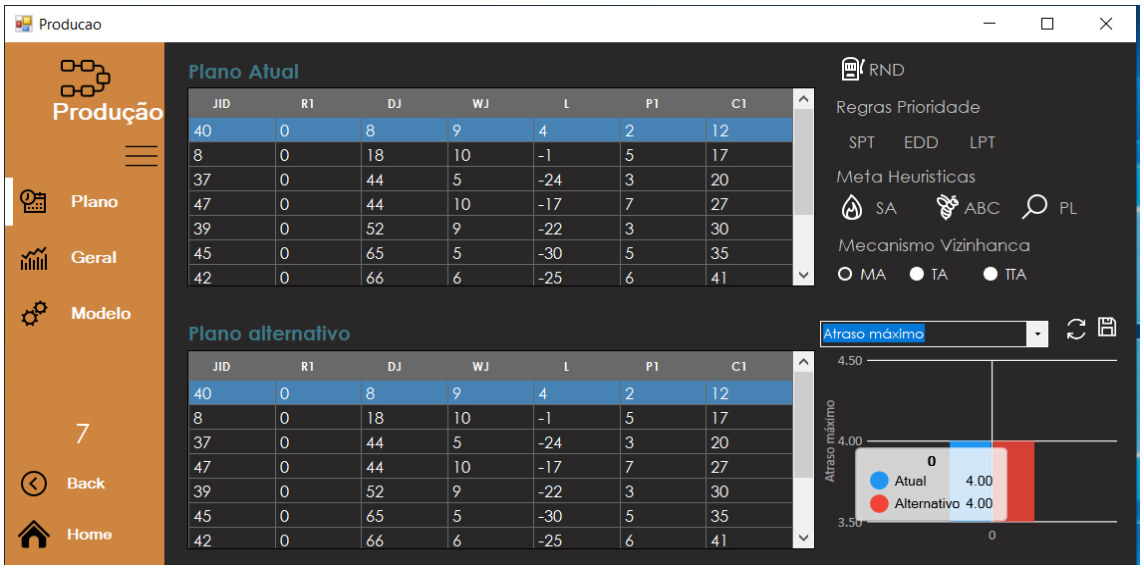


Figura 44 Entrada de uma tarefa em produção

4.6.2 Geral

O campo geral serve como base se informação quer para o gestor quer para os operadores do respetivo posto de trabalho. Neste campo da aplicação é onde se encontra toda a informação em tempo real não só sobre a performance do plano como também da tarefa que se encontra em curso e que tarefas irão entrar de seguida em produção, Figura 45.

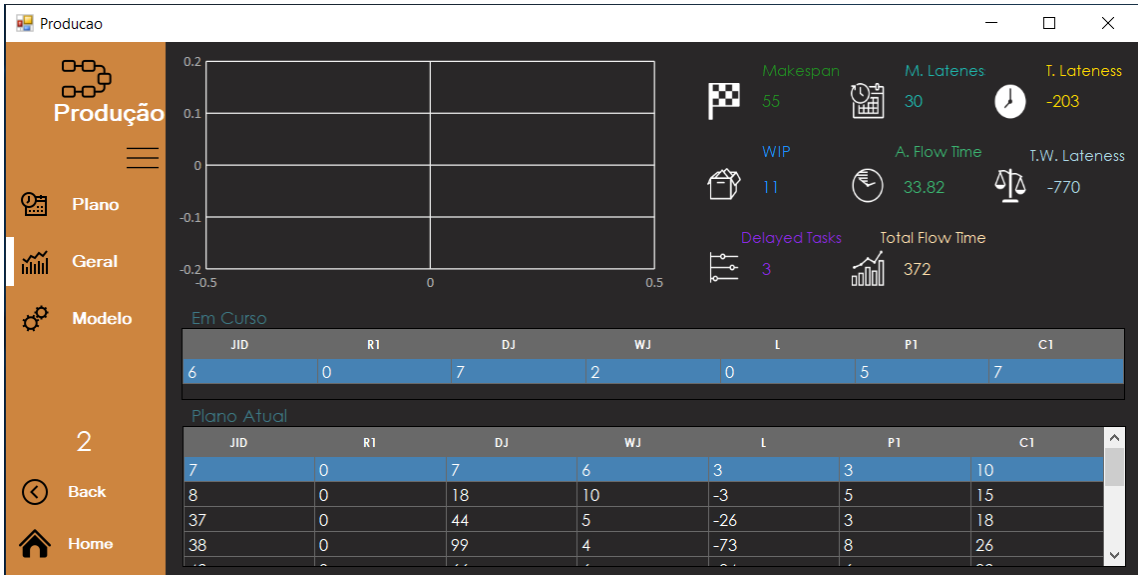


Figura 45 Campo geral

Caso o plano de produção altere, a informação é de imediato atualizada, como se irá ver mais adiante no presente relatório. Assim, é possível uma transmissão de informação segura e imediata para todos as partes interessadas. Como se pode observar pela Figura 46 em comparação com a Figura 45, a tarefa que iria entrar em produção era a 7 seguida da 8 e 37, e o plano incorria num atraso máximo de 30. Contudo, o plano foi escalonado pelo método EDD por forma a minimizar o atraso máximo, pelo que a tarefa que irá agora entrar em produção, após a 7, será a 40 em vez da 8, onde agora o plano possui um atraso máximo de 4 unidades temporais.

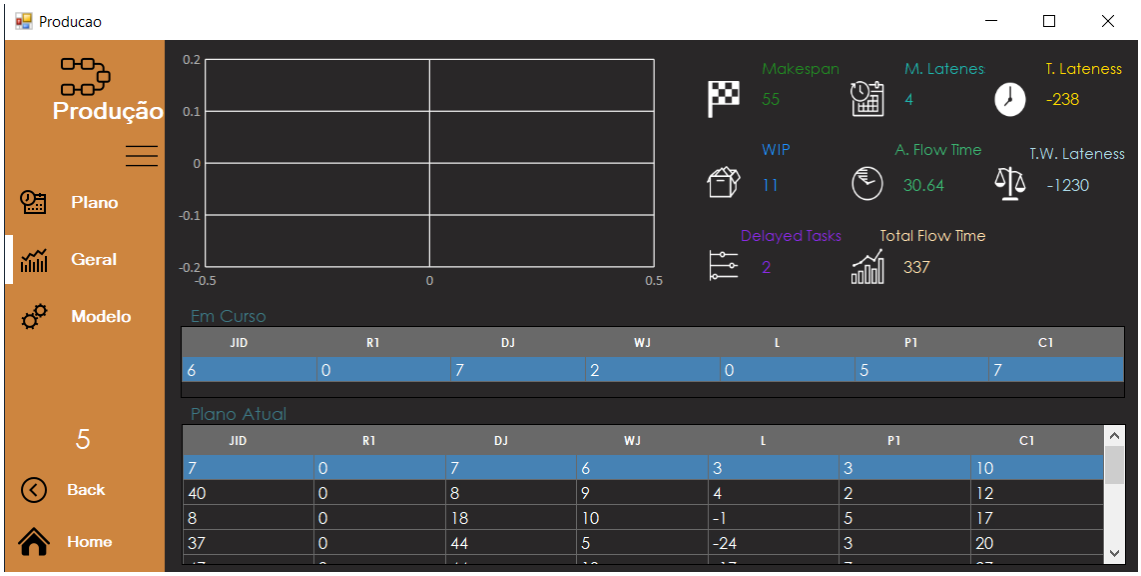


Figura 46 Alteração do plano em tempo real

O gráfico presente neste campo da aplicação só mostra informação caso o modelo autónomo de escalonamento esteja ativo. Este irá ilustrar o comportamento do plano à medida que novas tarefas dão entrada no sistema, seja por adição manual de novas encomendas, sejam encomendas geradas autonomamente pelo MRP associado, que neste caso o protótipo desenvolvido simula a existência de um MRP ligado à aplicação. Ou seja, um certo número de tarefas num determinado período no tempo dará entrada no chão de fábrica, sem qualquer interação por parte do utilizador, isto se o modelo estiver ativo como se verá no ponto seguinte.

4.6.3 Modelo

Tal como se referiu anteriormente, no campo “Modelo” é onde o utilizador pode adicionar e/ou remover encomendas associadas ao respetivo posto de trabalho selecionado, bem como parametrizar o escalonamento autónomo por parte do protótipo com base no modelo de Kano, Figura 47.

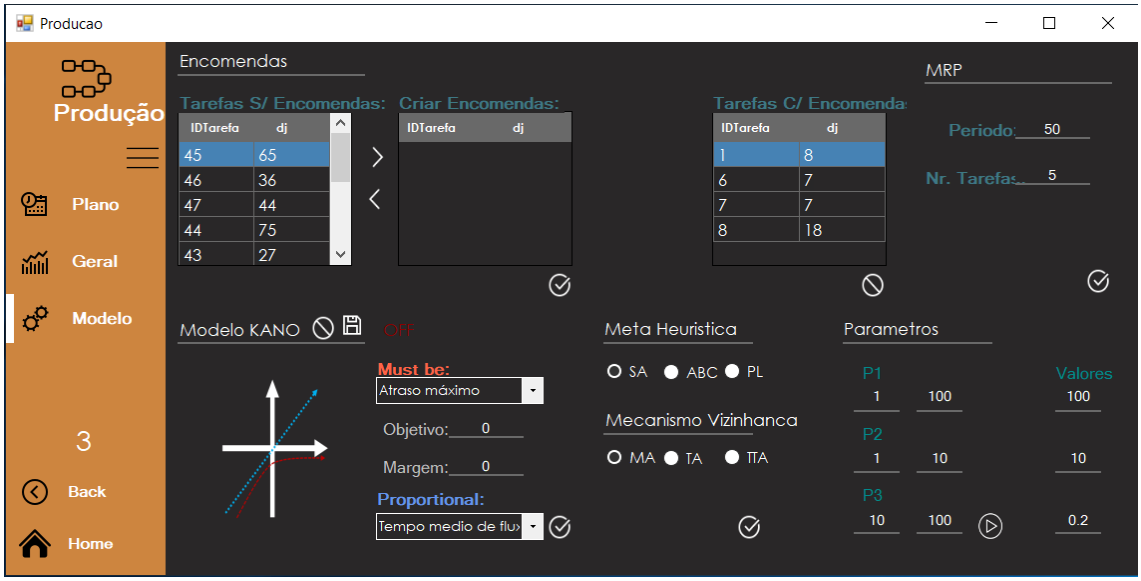


Figura 47 Campo modelo

Assim que o utilizador entra nesta funcionalidade da aplicação, a mesma carrega quer as tarefas alocadas ao respetivo posto de trabalho sem encomendas, quer as que já possuem encomenda. Caso se pretenda gerar novas encomendas para dar entrada no posto de trabalho, basta selecionar as diversas tarefas pretendidas e encaminhá-las para a janela "Criar encomendas", conforme ilustra a Figura 48, onde se pretende criar encomendas para as tarefas 44,45 e 47.

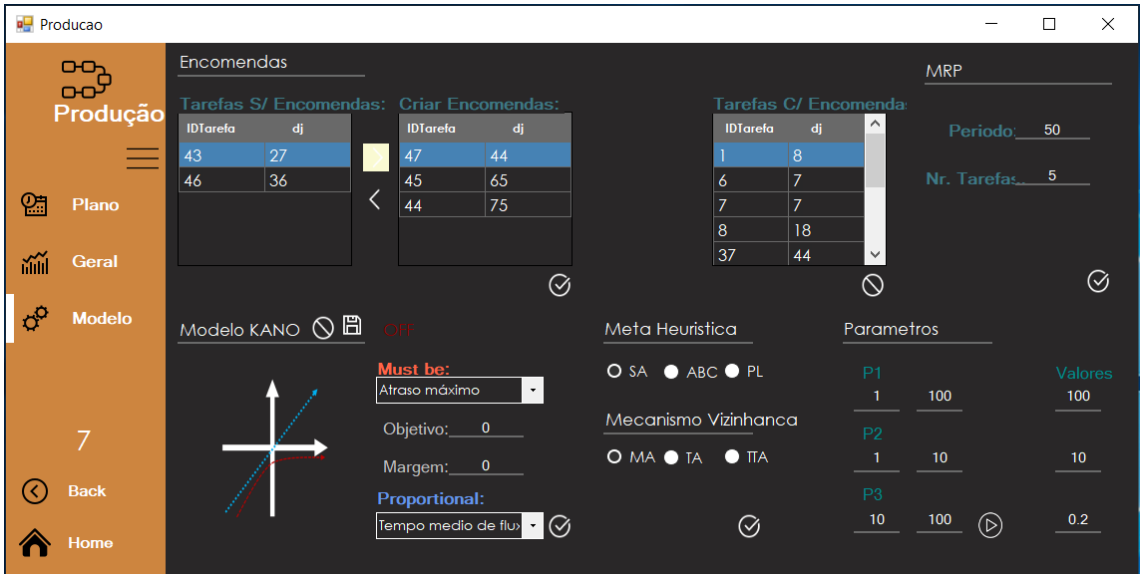


Figura 48 Escolha de tarefas para novas encomendas

O inverso também é válido, ou seja, se por lapso o utilizador tenha passado tarefas que para já não pretende gerar encomendas, basta selecionar as mesmas na janela "Criar encomendas" e enviá-las novamente para a janela "Tarefas S/ encomendas".

Por fim, para gerar as respetivas encomendas para as demais tarefas demonstradas na janela “Criar encomendas”, basta validar que as mesmas serão geradas na base de dados e demonstradas na janela “Tarefas C/ encomendas”, Figura 49.

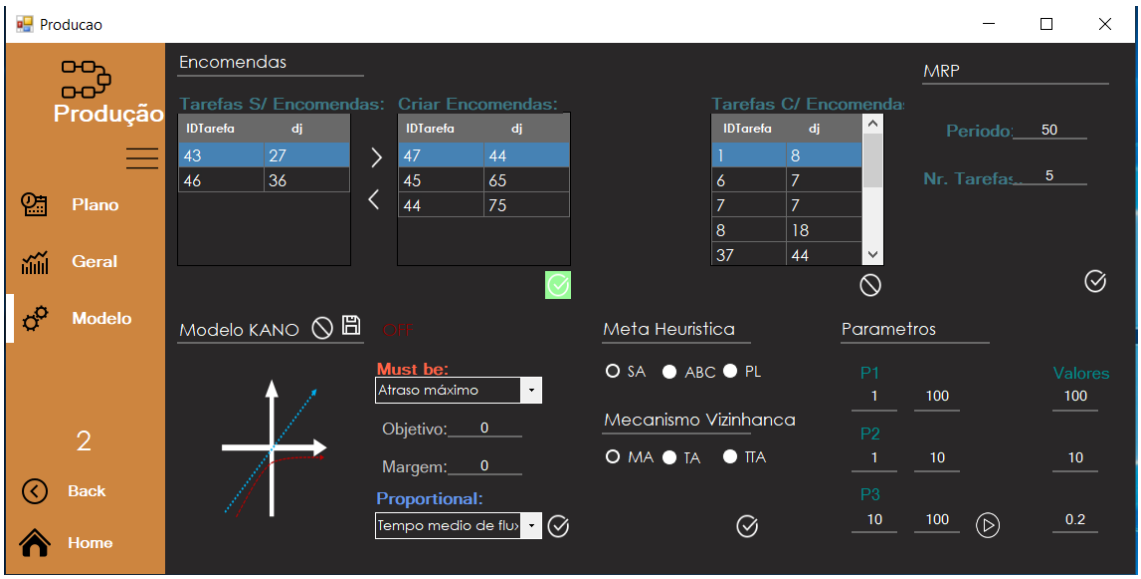


Figura 49 Criação de encomendas para as tarefas selecionadas

Como se pode observar pela Figura 49, o resto da funcionalidade do campo “Modelo” é dividido em três grupos: O grupo do MRP, do Modelo de Kano e das Meta heurísticas, de seguidos descritos individualmente.

4.6.3.1 MRP

Neste grupo do MRP, é onde o protótipo simula a conexão do mesmo com o MRP da empresa. É através da parametrização deste grupo que a aplicação saberá de quanto em quanto tempo, e quantas novas tarefas é que darão entrada no chão de fábrica. Para isso basta definir o “Período MRP”, que como o próprio nome indica este campo defini a periodicidade de entrada de novas tarefas no respetivo posto de trabalho, assim como o “Nr. Tarefas”, onde este último parametriza o número máximo de tarefas máximo que darão entrada por cada período de tempo decorrido. Por fim basta guardar a informação definida, Figura 50.

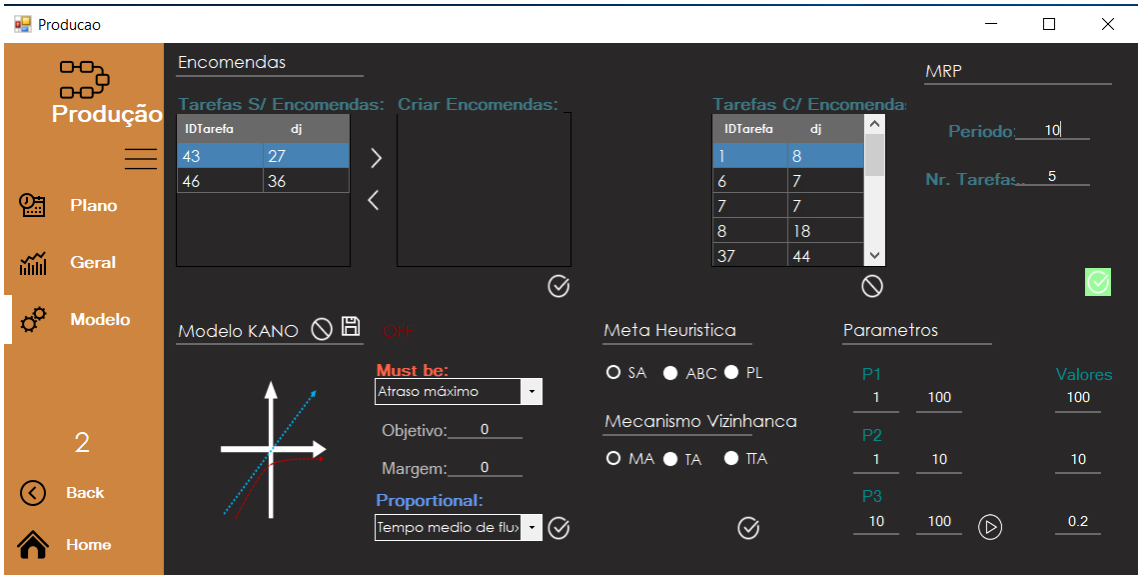


Figura 50 Definição dos parâmetros MRP

4.6.3.2 Modelo Kano

No campo do modelo de Kano, é onde o utilizador define as medidas de desempenho que pretende otimizar, isto é, as medidas pelas quais o escalonamento autónomo se irá guiar. Nesta primeira fase do projeto, a aplicação apenas permite a definição de uma medida de desempenho em ambos os critérios do modelo de Kano, critério "*must be*" e critério "*one dimensional*". Como trabalho futuro pretende-se dar a possibilidade de classificar várias medidas de desempenho em cada um dos campos do modelo de Kano. Assim que esteja definida a medida de desempenho "*must be*", o utilizador necessita de indicar o objetivo desejado para mesma, que por defeito é zero. O objetivo irá servir de suporte ao escalonamento autónomo da ferramenta, pois será o guia que irá definir se a aplicação deverá utilizar o método de otimização A ou o método de otimização B, Figura 51.

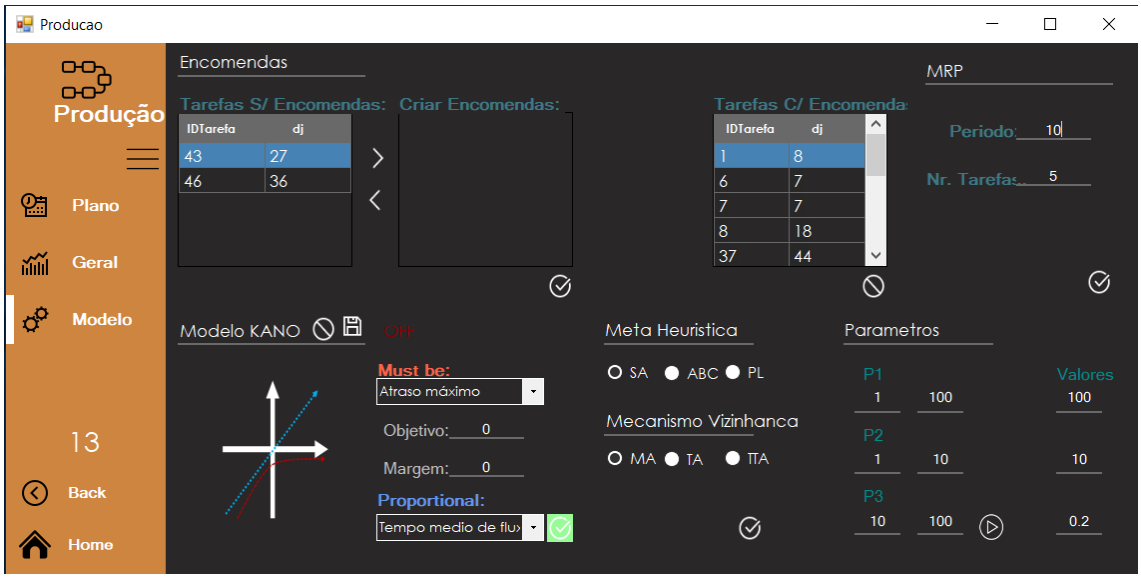


Figura 51 Definição do modelo de Kano

Por sua vez, e por forma a ajudar a cumprir o objetivo desejado, o utilizador pode também definir uma margem para o mesmo. Com uma margem definida, o modelo irá utilizar o método de escalonamento que otimiza a medida de desempenho definida em "*must be*" assim que o valor x=*(objetivo-margem)* seja ultrapassado. No entanto, é percetível que tal valor de margem comprometerá a longo prazo o critério proporcional, uma vez que escalonamentos que minimizam tal critério irão ser descartados com mais frequência, pois a ferramenta irá forçar o uso de métodos de escalonamento para satisfazer o critério *must be*. Dito isto, é possível perceber o impacto que o valor da margem pode causar no sistema, se for muito alto, a ferramenta ira cumprir com o objetivo definido; no entanto, a satisfação do utilizador quanto ao critério proporcional estará comprometida. Por outro lado, quanto menor o valor de margem, ou mesmo zero, e dependendo das características das tarefas, a ferramenta poderá não ter a capacidade de impedir que o plano exceda o limite máximo delineado o que provocará uma violação nos critérios *must be*. Portanto, um equilíbrio entre a margem e o objetivo realmente pretendido deve ser considerado.

Para uma melhor interpretação do descrito, analise-se o seguinte exemplo da Figura 51 que se trata de um posto de trabalho de ambiente em máquina única.

Da Figura 51 retira-se que não se pretende que o atraso máximo seja superior a zero unidades temporais e sempre que possível, deseja-se minimizar o tempo médio de fluxo. Da literatura sabe-se que as tais medidas de desempenho são otimizadas através das regras de prioridade EDD e SPT, respetivamente. Logo, sempre que entrarem novas tarefas no sistema, o modelo irá escalonar, inicialmente, todo o WIP em chão de fábrica do respetivo posto através do SPT e posteriormente irá verificar o valor do atraso máximo para a solução encontrada. Caso ultrapasse o valor do objetivo definido, o modelo reescalona as tarefas de acordo com o EDD, caso contrário o plano atual é atualizado para a solução encontrada.

4.6.3.3 Meta heurísticas

Por fim, no campo das meta-heurística é onde o utilizador parametriza o modo como o modelo irá escalonar as tarefas sempre que novas tarefas dão entrada no chão de fábrica através do MRP da aplicação. Ou seja, sempre que o MRP liberta novas encomendas para o chão de fábrica, o modelo irá escalonar o WIP de acordo com a meta heurística selecionada e respetivos parâmetros definidos, Figura 52.

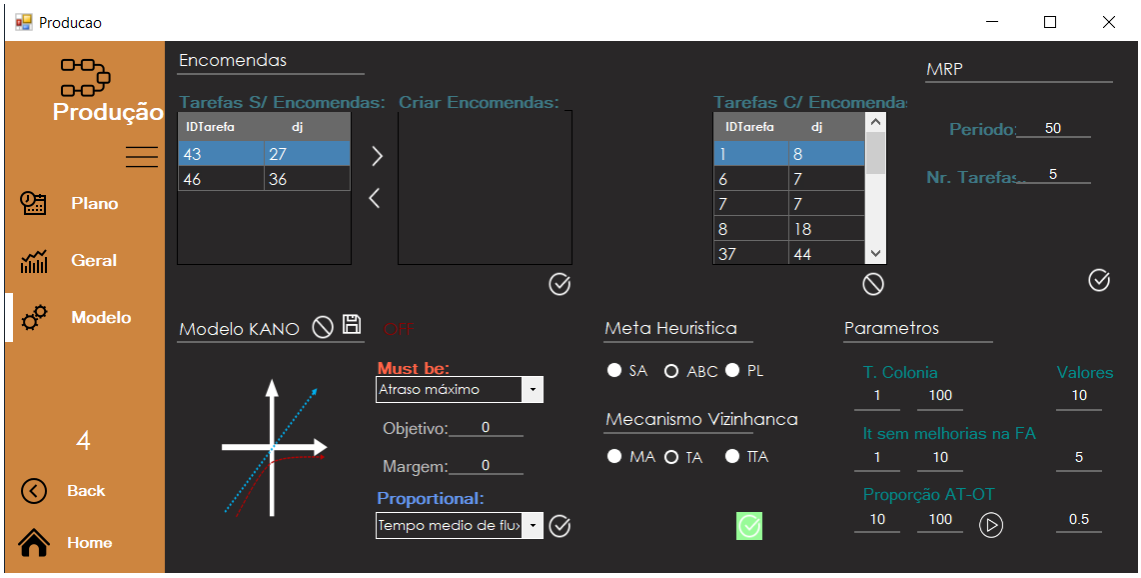


Figura 52 Definição da meta heurística

De notar que dependendo das medidas de desempenho selecionadas no modelo de Kano, a *fitness* da meta-heurística irá variar. Contudo, a mesma pode ser escrita genericamente através da expressão 10.

$$F(x) = \propto . A + B, onde \ \propto = \begin{cases} M, se\ A > (objetivo - margem) \\ 1, caso\ contrário \end{cases} \quad (10)$$

Onde A diz respeito à fórmula que permite o cálculo da medida de desempenho do critério "*must be*", B representa a fórmula para a medida de desempenho "*one-dimensional*" e M representa um majorante que força a meta heurística a minimizar o critério "*one- dimensional*". Ou seja, se M for diferente de 1, significa que a solução encontrada não cumpre com o objetivo definido, isto é, excede o valor pretendido para o critério "*must be*", logo a meta-heurística necessita de encontrar uma solução onde o critério "one- dimensional" é mínimo dentro do conjunto de soluções onde o critério "*must-be*" é violado. Pegando no exemplo anterior do ponto 4.6.3.2, a fórmula para a *fitness* da meta heurística ficaria de acordo com a expressão 11.

$$F(x) = \propto . Max(L) + \frac{\sum cj}{n}, onde \ \propto = \begin{cases} M, se\ Max(L) > (objetivo - margem) \\ 1, caso\ contrário \end{cases} \quad (11)$$

4.6.3.4 *Ativação do modelo*

Para a ativação do modelo autónomo de escalonamento tendo por base as diversas opções definidas nos três grupos anteriormente referidos, basta selecionar a opção "Guardar" e a ferramenta irá indicar de imediato que o modelo passa a estar ativo. Tal é possível de visualizar através do texto "ON", Figura 53.

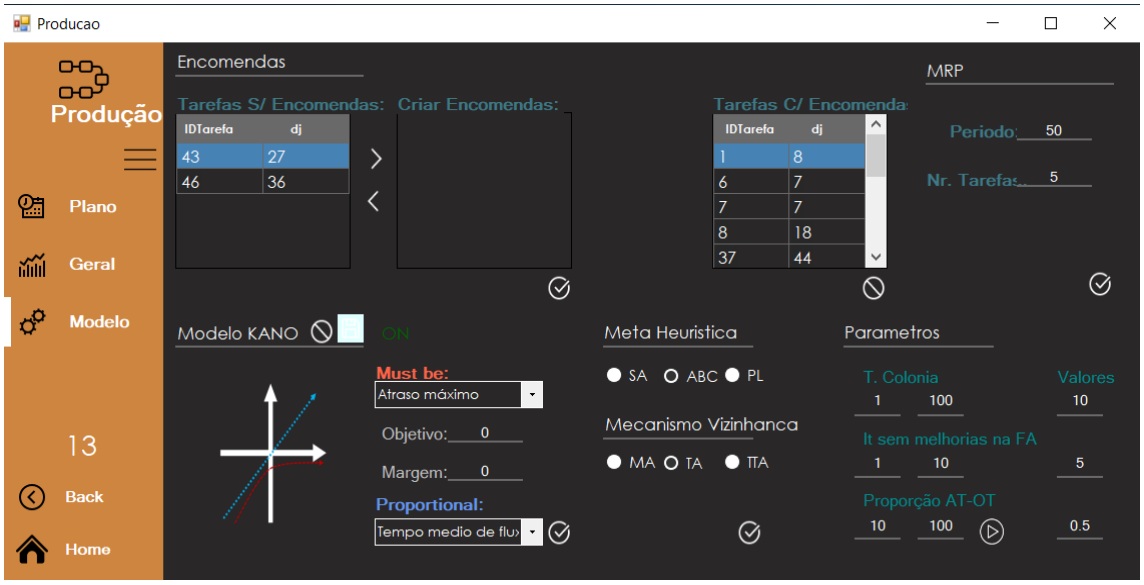


Figura 53 Ativação do escalonamento autónomo do protótipo

A qualquer momento o utilizador pode cancelar a autonomia da programação das tarefas, para isso basta clicar no botão “Cancelar”, Figura 54. A partir desse momento, o protótipo deixa de estar ligado ao MRP e de reescalonar novas tarefas que dão entrada no chão de fábrica. Cabe ao utilizador ter essa gestão.

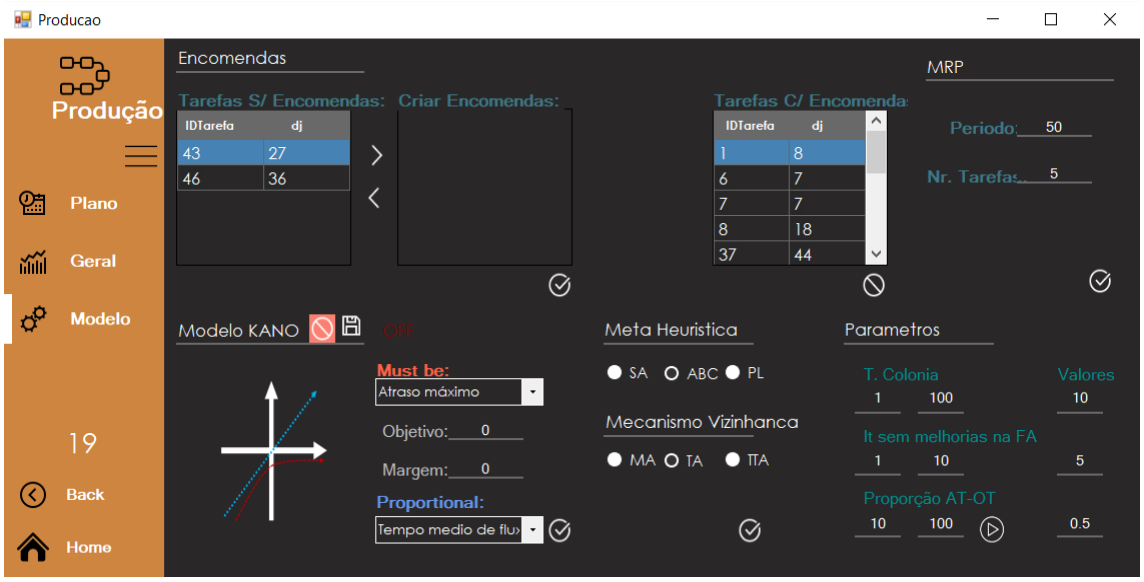


Figura 54 Cancelamento do escalonamento autónomo

ESTUDO COMPUTACIONAL

5.1 Estudo em Ambiente Máquina Única

5.1.1 Resultados

5.1.2 Análise estatística

5 Estudo Computacional

5.1 Estudo em Ambiente Máquina Única

Para demonstrar o funcionamento do protótipo, foram criados 100 trabalhos com atributos aleatórios, Tabela 14. Para este estudo foi definido que o período entre as execuções do MRP seria de 25 unidades temporais (21 tarefas por período), o critério "*one- dimensional*" seria o Tempo Médio de Fluxo e o critério "*must be*" o Atraso Máximo, onde o valor máximo seria zero com uma margem de 20 unidades. O SA foi escolhido como a meta heurística a utilizar sempre que o MRP liberta tarefas para o chão de fábrica. Ao longo do tempo foram também adicionadas aleatoriamente tarefas não planeadas e fora do conjunto de tarefas do MRP. A Figura 55 mostra a definição dos parâmetros no protótipo de acordo com o que foi acima descrito. Os resultados obtidos através do modelo, serão comparados com os resultados que seriam obtidos se apenas se utilizasse a metodologia que minimizasse o critério "*must be*", que no exemplo em questão seria o EDD.

Tabela 14 Distribuições usadas para a geração de tarefas

T	0	25	50	75	Aleatório
Nº de tarefas	24	21	21	21	14
rj	0	25	50	75	N(50,15)
Dj [N(μ,σ)+ +rj+p]	N(250,25)	N(350,25)	N(500,25)	N(550,25)	N(750,25)
Pj	N(5,1)				
Wj	N(10,3)				

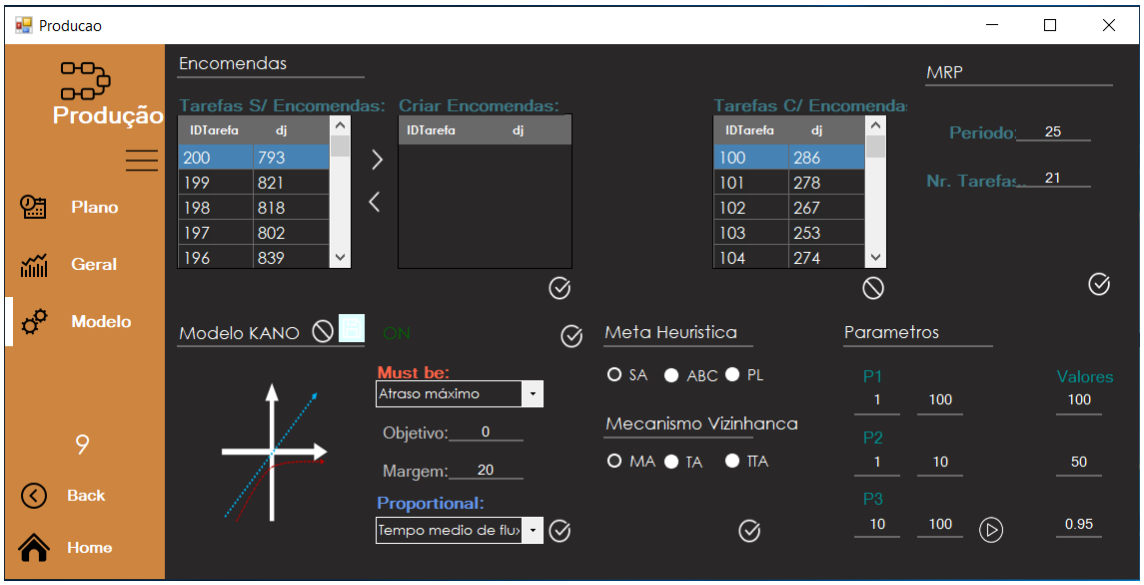


Figura 55 Definição do modelo a utilizar no protótipo

Tal como se referiu anteriormente, dependendo das medidas de desempenho selecionadas no modelo de Kano, a *fitness* da meta-heurística varia. Para os critérios definidos, a *fitness* a utilizar pelo SA será a da expressão 12.

$$F(x) = \propto . Max(L) + \frac{\sum cj}{n}, onde \ \propto = \begin{cases} M, se\ Max(L) > (objetivo - margem) \\ 1, caso\ contrário \end{cases} \quad (12)$$

Na expressão 12, M representa um majorante que força a meta heurística a minimizar o Tempo Médio de Fluxo sempre que acionado. Esse M deve ser pelo menos o atraso máximo de um plano escalonado através da regra LPT. Assim, sempre que o atraso máximo exceder o valor definido, a equação força a meta heurística a encontrar uma solução em que o Tempo Médio de Fluxo é mínimo do conjunto de soluções que não conseguem cumprir com os valores definidos (objetivo -margem). Caso contrário, a meta heurística tenta encontrar uma solução que minimize ambos. Em relação ao "Módulo de Eventos Dinâmicos", a ferramenta utilizará regras de despacho como SPT e EDD, uma vez que estas minimizam o Tempo Médio de Fluxo e o Máximo Atraso, respetivamente (L. Ferreirinha et al., 2019).

5.1.1 Resultados

Tal como mencionado anteriormente, os resultados do modelo foram comparados com os resultados de um sistema baseado em EDD. A Figura 56 compara o Tempo Médio de Fluxo e o WIP entre os dois sistemas. Tal como se pode observar, o Tempo Médio de Fluxo através do sistema baseado em EDD tende a aumentar muito mais rápido em comparação com o protótipo. Isto permite concluir que apesar do sistema baseado em EDD ir ao encontro do critério "must be", não satisfará o utilizador quanto ao critério proporcional. Tal como se pode observar, agora pela Figura 57, o modelo conseguiu evitar que o atraso máximo ultrapasse o valor definido inicialmente (-20), uma vez que os valores obtidos ao longo do tempo aquando a entrada de novas tarefas (linha azul), manteve-se inferior ao objetivo (linha vermelha), o que significa que para além de ter ido ao encontro do critério "must be", o modelo minimizou, sempre que possível, o tempo médio de fluxo (critério proporcional). Tal pode ser observado novamente na Figura 56, onde se repara no crescimento menos acentuado do Tempo Médio de Fluxo. De notar que desde o início do plano, onde tal medida de desempenho apresentava um valor de 65,25 unidades temporais, no final do plano apresentava um valor final de 112,22, o que se traduz num aumento de aproximadamente 47 unidades temporais, face ao crescimento de 120 unidades temporais no sistema baseado em EDD. O desempenho do modelo pode ainda ser refletido no WIP, tal como ilustra a Figura 56, isto é, à semelhança do Tempo Médio de Fluxo, o WIP, no sistema baseado em EDD, também tende a aumentar rapidamente ao longo do tempo. Isso se deve ao fato de que o EDD não priorizar tarefas com tempos de processamento mais curtos, o que faz com que tarefas longas tenham a possibilidade de serem processadas primeiro, criando um "bottleneck" na seção e, por sua vez, gerando um WIP mais rápido. Enquanto no caso

do modelo, o mesmo conseguiu diminuir o WIP ao longo do tempo, 18 tarefas no final, o sistema baseado em EDD acabou com 65 tarefas.

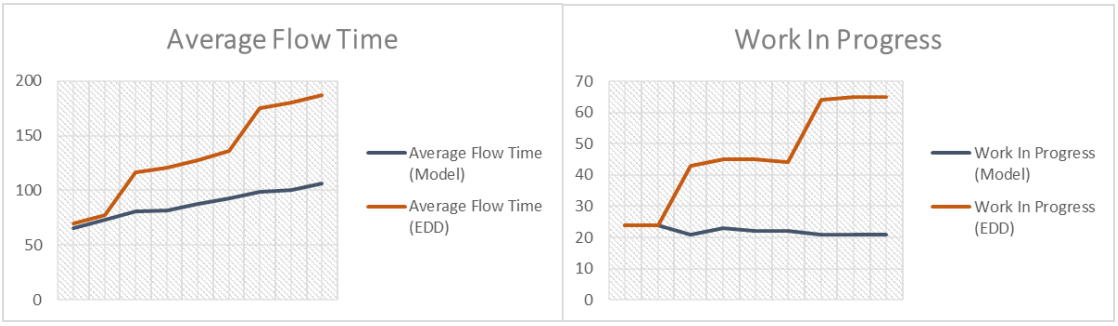


Figura 56 Comparação entre resultados obtidos

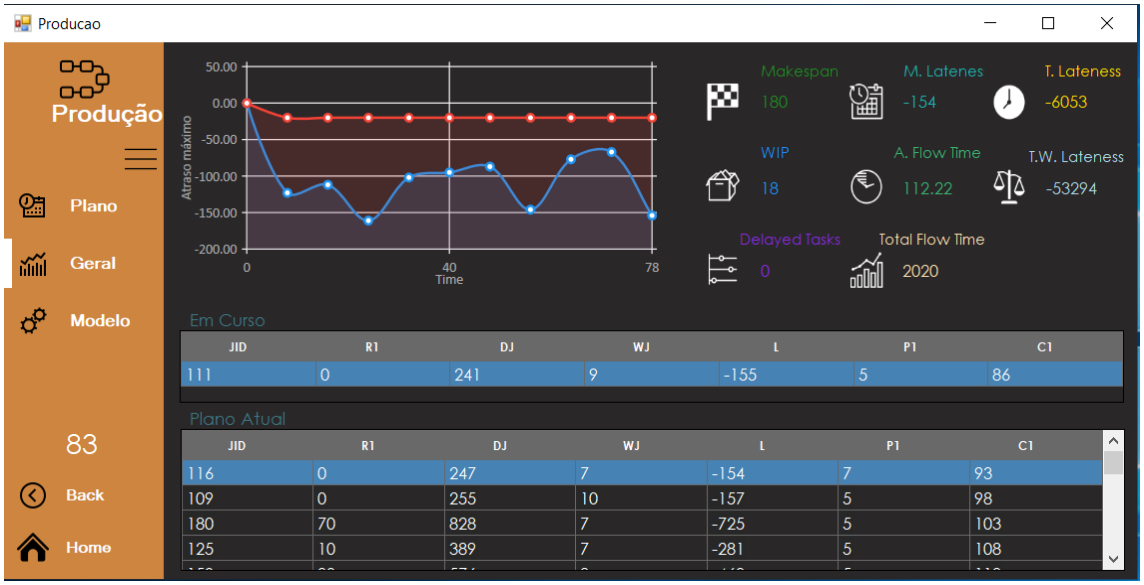


Figura 57 Resultados obtidos através do modelo

5.1.2 Análise estatística

Neste ponto, pretende-se analisar o desempenho da ferramenta desenvolvida. Ou seja, pretende-se encontrar evidências estatísticas que comprovem que o conceito do protótipo alcança um desempenho melhor que o método único de escalonamento, neste caso particular, em comparação com a regra EDD. Para isso, foram geradas 9 instâncias de 100 tarefas com as características estocásticas apresentadas anteriormente (Tabela 14). Aa Figura 58 e Figura 61 mostram os resultados obtidos, pela ferramenta e pelo uso contínuo de EDD, para cada instância em relação ao tempo médio de fluxo e ao trabalho em processamento, respetivamente.

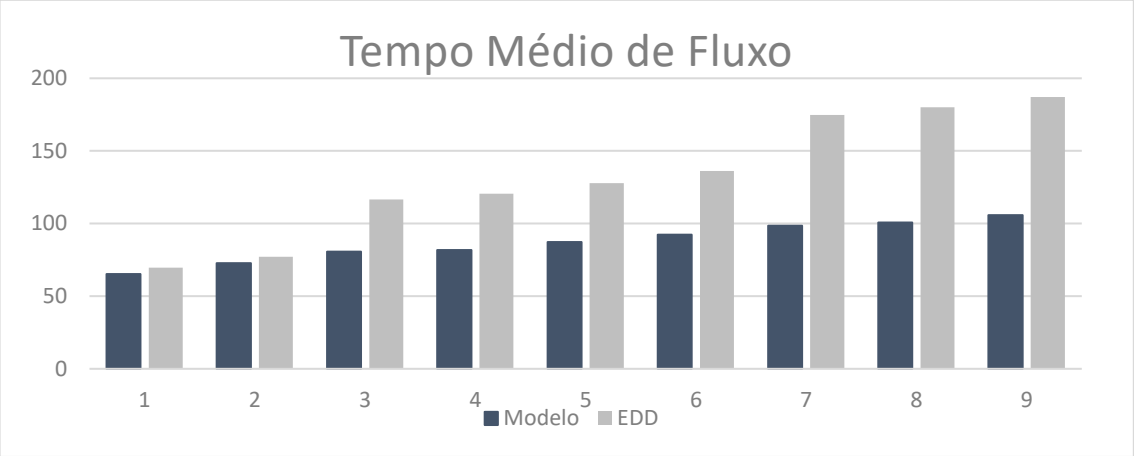


Figura 58 Resultados obtidos do tempo médio de fluxo para cada instância

Como pode ser visualizado na Figura 58, o tempo médio de fluxo para todas as instâncias obtidas pelo modelo são sempre inferiores à regra do EDD. De facto, esse resultado já era o esperado, pois sempre que a ferramenta tem oportunidade, a mesma utiliza o SPT para minimizar o tempo médio de fluxo, ao contrário do EDD, que se concentra apenas em minimizar o atraso máximo. A partir dos resultados apresentados na Figura 58, vale a pena destacar que no modelo, o valor mínimo registrado foi de 62.25, comparado ao valor máximo de 105.86, e apresentou uma média de 87.20 unidades de tempo com um desvio padrão de 4.48. Quanto aos resultados do sistema baseado em EDD, o mesmo registou um valor mínimo de 69.58, um valor máximo de 187.09 e um valor médio de 132.20 unidades de tempo, com um desvio padrão de 14.18. Perante tais resultados, aparentemente a ferramenta tem um melhor desempenho do que um sistema dinâmico escalonado em EDD relativamente à minimização do atraso máximo e à minimização do tempo médio de fluxo. No entanto, para verificar essa declaração, é necessário um teste t paramétrico para amostras independentes. Uma vez que ambas as distribuições são normais o teste paramétrico pode ser feito com as hipóteses:

$$H0: \mu\ TMF\_Protótipo = \mu\ TMF\_EDD$$

$$H1: \mu\ TMF\_Protótipo < \mu\ TMF\_EDD$$

Group Statistics

	Modelo	N	Mean	Std. Deviation	Std. Error Mean
TMF	Prototipo	9	87.2011	13.43509	4.47836
	EDD	9	132.2022	42.54482	14.18161

Independent Samples Test

		Levene's Test for Equality of Variances		t-test for Equality of Means						
									95% Confidence Interval of the Difference	
		F	Sig.	t	df	Sig. (2-tailed)	Mean Difference	Std. Error Difference	Lower	Upper
TMF	Equal variances assumed	7.320	.016	-3.026	16	.008	-45.00111	14.87191	-76.52815	-13.47407
	Equal variances not assumed			-3.026	9.580	.013	-45.00111	14.87191	-78.33585	-11.66637

Figura 59 Resultados do teste paramétrico para amostras independentes relativamente ao tempo médio de fluxo

Como a igualdade das variâncias é assumida, Figura 59, verifica-se que o tempo médio de fluxo através do modelo é inferior aos resultados obtidos pela regra EDD, t (16) = -3.026, p-value = 0.004. Ou seja, ao nível de 5%, existe evidência estatística de que o tempo médio de fluxo no modelo é menor do que o obtido num sistema baseado em EDD para ambiente de escalonamento dinâmico, tal como a Figura 60 demonstra.

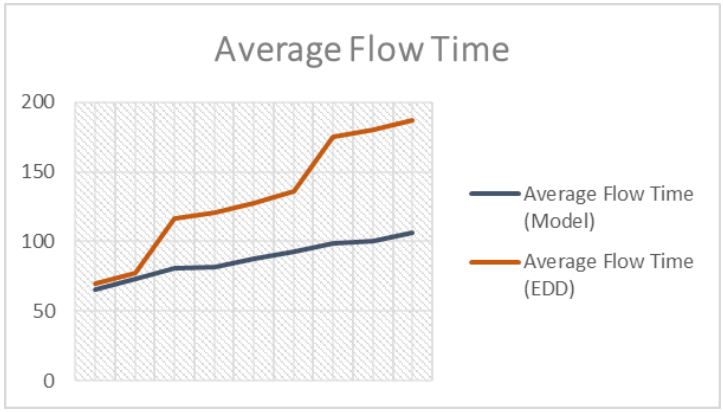


Figura 60 Resultados entre os modelos para o tempo médio de fluxo

Na Figura 61 a diferença do WIP entre o protótipo e o EDD, é mais percetível, e à semelhança do tempo médio do fluxo, o WIP registado pelo modelo é menor em todas as instâncias.

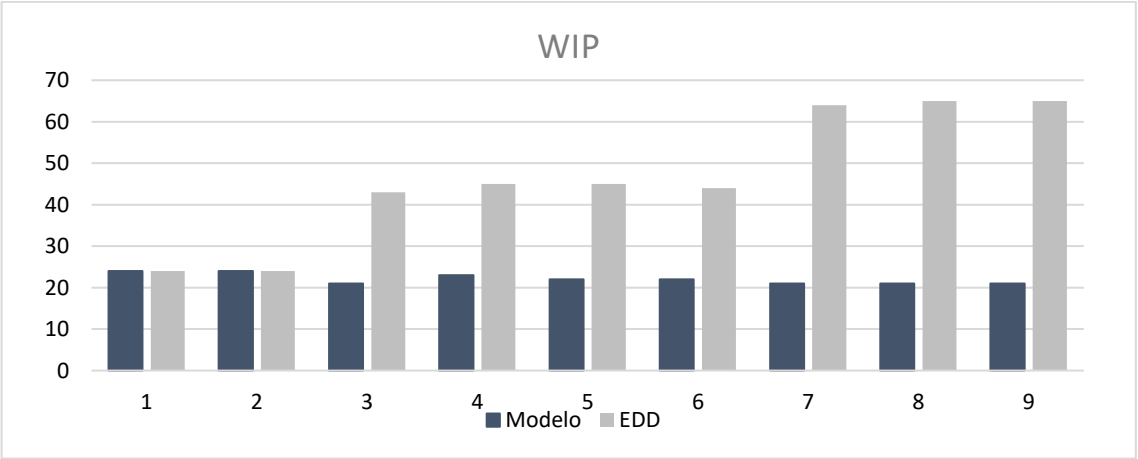


Figura 61 Resultados obtidos para o WIP para cada instância

Através da ferramenta obteve-se um valor mínimo de 21 e um valor máximo de 24 tarefas no sistema, bem como uma média de 22,11 com um desvio padrão de 1.27. Para o sistema baseado em EDD, registou-se um mínimo de 24, um valor máximo de 65 e uma média de 46.55 unidades no sistema com um desvio padrão de 15.91. Tal como efetuado para a medida de desempenho anterior, para verificar se o WIP obtido pelo sistema é menor do que o obtido através do EDD num ambiente de escalonamento dinâmico, efetuou-se um teste t paramétrico para amostras independentes. Uma vez que ambas as distribuições também são normais, o teste pode ser escrito através das hipóteses:

H0: μ WIP_Protótipo = μ WIP_EDD

H1: μ WIP_ Protótipo <μ WIP_EDD

Group Statistics

	Modelo	N	Mean	Std. Deviation	Std. Error Mean
WIP	Prototipo	9	22.1111	1.26930	.42310
	EDD	9	46.5556	15.91470	5.30490

Independent Samples Test

		Levene's Test for Equality of Variances		t-test for Equality of Means						
									95% Confidence Interval of the Difference	
		F	Sig.	t	df	Sig. (2-tailed)	Mean Difference	Std. Error Difference	Lower	Upper
WIP	Equal variances assumed	12.226	.003	-4.593	16	.000	-24.44444	5.32175	-35.72604	-13.16285
	Equal variances not assumed			-4.593	8.102	.002	-24.44444	5.32175	-36.68963	-12.19926

Figura 62 Resultados do teste paramétrico para amostras independentes relativamente ao WIP

Da Figura 62 conclui-se que a igualdade das variações é assumida, pelo que se verifica que o WIP através do modelo é inferior aos resultados obtidos pela regra EDD, t (16) = -4.593, p-value = 0.000. Ou seja, no nível de 5% existe evidência estatística de que o WIP através do protótipo é menor do que o obtido através do EDD para um ambiente de escalonamento dinâmico, tal como a Figura 63 demonstra.

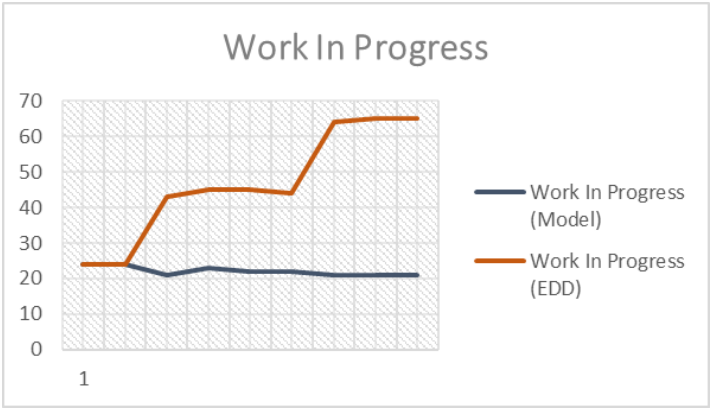


Figura 63 Resultados entre os modelos para o WIP

CONCLUSÕES

6.1 Trabalho Futuro

6 CONCLUSÕES

O escalonamento da produção afeta todos os ramos quer das organizações industriais, quer dos serviços, sendo que no contexto industrial tem normalmente implícito o objetivo da satisfação dos clientes, principalmente quanto às datas de entrega estabelecidas. A utilização eficiente dos recursos internos nas organizações (equipamentos) torna-se uma vantagem competitiva, podendo assim ditar a sua sobrevivência no mercado. Neste sentido, torna-se crucial a análise e desenvolvimento de modelos de escalonamento da produção uma vez que este último permite a definição dos trabalhos a executar pelos recursos disponíveis no sentido de otimizar uma ou mais determinadas medidas de desempenho. O escalonamento da produção torna-se complexo devido à natureza combinatória de sequências possíveis, difíceis de enumerar em tempo útil. Assim, para a resolução de problemas de escalonamento é frequente o recurso a métodos aproximativos. Entre as diversas técnicas de aproximação, as meta heurísticas têm sido as que mais suscitam maior interesse na teoria do escalonamento ao fazerem uma pesquisa orientada pelas características do espaço de soluções.

A presente dissertação teve como intuito contribuições no domínio do escalonamento da produção em duas envolventes principais: uma a um nível teórico, conceptual e outra ao nível prático da resolução de problemas através do desenvolvimento de uma ferramenta dinâmica de apoio à decisão. Ao nível conceptual contribuiu para uma ontologia de problemas de escalonamento da produção e conceitos relacionados, tais como ambiente de escalonamento, características dos trabalhos e dos recursos, critérios de otimização, medidas de desempenho, ferramentas de escalonamento, tipos de escalonamento e técnicas de resolução de problemas combinatórios e sua parametrização, providenciando um enquadramento comum para a compreensão e partilha de conhecimento acerca destes conceitos. Ao nível da resolução de problemas, desenvolveu-se uma ferramenta simples e moderna, capaz de escalonar autonomamente o WIP, sempre que ocorrem eventos não planeados em postos de trabalho, nomeadamente a entrada e/ou remoção de tarefas, tendo sempre em conta os critérios de desempenho definidos pelo utilizador.

Para além da sua simplicidade e comodidade de uso, o protótipo desenvolvido permite a gestão de escalonamento de todos os postos de trabalho com encomendas alocadas de um modo dinâmico. Isto é, dado que o sistema desenvolvido pretende servir como uma ferramenta de apoio à decisão aos problemas de escalonamento dinâmico, a mesma tem que operar como tal, portanto, à medida que o tempo passa, várias tarefas alocadas aos respetivos postos de trabalho vão sendo executadas/finalizadas, e outras vão entrando no sistema, pelo que a ferramenta é capaz de gerir todos esses eventos em simultâneo sem interferir na utilização da mesma. Tal deve-se à capacidade de *thread* introduzida na ferramenta, ou seja, independentemente do foco do utilizador, quer esteja a inserir tarefas, quer esteja a ajustar um determinado plano, ou a efetuar outra operação, as tarefas associadas a encomendas vão sendo executadas em paralelo. Assim, sempre que o utilizador entre no ambiente de escalonamento de qualquer posto

de trabalho, o mesmo estará atualizado e não terá necessidade de recalcular tudo o que aconteceu até à unidade de tempo atual. A ferramenta permite o escalonamento de tarefas nos postos de trabalho de duas formas, ou de uma forma manual, onde o próprio utilizador fica responsável por escalonar e comparar as soluções obtidas com o plano atual, ou de uma forma autónoma, onde o utilizador delega grande parte do escalonamento ao protótipo, tendo apenas como preocupação a definição das medidas de desempenho que pretende atingir/ cumprir no respetivo posto de trabalho. Tal definição de medidas de desempenho é feita através do modelo de Kano, onde o utilizador define qual a medida de desempenho que o plano deve cumprir (critério *must be*) por forma a evitar um extrema insatisfação no utilizador, e a medida de desempenho que o utilizador gostaria de otimizar sempre que possível (critério *one-dimensional*), uma vez que quanto melhor tal medida de desempenho, maior a satisfação do utilizador.

A ferramenta desenvolvida simula a sua conexão ao MRP da empresa e, de acordo com os objetivos definidos pelo modelo de Kano, sempre que ocorrem eventos não planeados, o protótipo programa e reprograma através de meta heurísticas e através do módulo dinâmico "Eventos dinâmicos", a fim de cumprir com os objetivos definidos. No escalonamento autónomo da ferramenta, uma vez que conta com a periodicidade do MRP, a mesma utiliza uma estratégia hibrida de escalonamento, ou seja, escalona o WIP sempre que eventos não planeados ocorrem no sistema, como o caso de entrada de novas tarefas e/ ou cancelamento de outras, e também de um modo periódico, que neste caso coincide com o lançamento de tarefas por parte do MRP.

Por norma, o MRP lança um grande número de tarefas para o chão de fábrica pelo que para escalonar todo o WIP associado ao posto de trabalho, o protótipo escalona o mesmo através de uma meta heurística selecionada, onde a parametrização desta última é efetuada de um modo manual. No caso de novas tarefas darem entrada no num determinado posto de trabalho sem ser através do MRP, o protótipo utiliza regras de prioridade (*módulo eventos dinâmicos*) para atender aos critérios definidos no modelo de Kano. Ou seja, à medida que novas tarefas entram no sistema, a ferramenta programa as mesmas de acordo com a medida de desempenho que otimiza o critério *one-dimensional*. Se o plano não violar o critério *must be* definido, a solução encontrada é aceite; caso contrário, a ferramenta recorre automaticamente à regra de prioridade que minimiza o critério *must be*, de modo a evitar que o utilizador incorra numa enorme insatisfação.

Para analisar o desempenho do modelo desenvolvido, foram geradas 9 instâncias com 100 tarefas estocásticas onde os critérios de otimização diziam respeito ao tempo médio de fluxo (*one-dimensional*) e o atraso máximo (*must be*) num posto de trabalho de ambiente em máquina única. Como objetivo do plano, pretendia-se que o atraso máximo não ultrapasse o valor zero, ao qual foi adicionado uma margem de 20 unidades temporais. Posteriormente, os resultados obtidos foram comparados com um cenário onde apenas o critério *must be* era considerado, ou seja, foi comparado com um cenário

onde o EDD seria o método de escalonamento sempre que novas tarefas davam entrada no sistema.

Os resultados mostram a eficácia de ferramenta onde o atraso máximo nunca foi violado e o tempo médio de fluxo foi minimizado sempre que possível. Através dos resultados obtidos, percebeu-se que o modelo baseado em EDD, apesar de ter cumprido com o objetivo definido no atraso máximo, tende a piorar de forma muito célere o critério proporcional e ainda o WIP no respetivo posto de trabalho. Após uma análise mais robusta, quer para o tempo médio de fluxo quer para o WIP, houve evidência estatística ao nível de 5%, de que a ferramenta tem melhor desempenho do que modelo baseado em EDD para um ambiente de escalonamento dinâmico onde os objetivos são classificados através do modelo de Kano. Tal como se mencionou na presente dissertação, o protótipo desenvolvido conta com um quadro geral de planeamento, onde é possível visualizar o plano atual, a tarefa atualmente em curso, a que irá entrar de seguida e ainda diversas medidas de desempenho. Tal quadro serve para informar às partes interessadas, principalmente aos operadores do respetivo posto de trabalho, o estado atual do posto, no que diz respeito ao que está planeado e qual a sua performance, o qual é atualizado sempre que alterações ocorrem no sistema.

O conceito da presente dissertação deu origem a três publicações em revistas indexadas, o que motivou e motiva ainda mais ao desenvolvimento e aperfeiçoamento do sistema desenvolvido (L. Ferreirinha et al., 2019; Luis Ferreirinha et al., 2019; Luís Ferreirinha et al., 2020).

Como todo e qualquer trabalho, ao longo desta dissertação, algumas dificuldades tiveram de ser enfrentadas e superadas. Uma vez que todo o presente projeto tinha por base o conceito “dinâmico”, técnicas de programação em tais ambientes tiveram de ser analisadas, interiorizadas e implementadas por forma atingir o principal objetivo.

6.1 Trabalho Futuro

Com relação a trabalhos futuros, pretende-se implementar o protótipo desenvolvido numa indústria e avaliar o seu desempenho, bem como preparar a ferramenta para outros eventos dinâmicos, tais como avarias na máquina, tempos de reparação, entre outros, a fim de diminuir a diferença entre a teoria do escalonamento e a prática. Como mencionado anteriormente, ainda é do interesse implementar outras regras de prioridade capazes de ir ao encontro de outras mediadas de desempenho, bem como ter a possibilidade de classificar mais que uma medida de desempenho em cada um dos critérios do modelo de Kano. Uma outra funcionalidade da ferramenta poderia ser a utilização do planeamento de experiências de Taguchi como técnica de parametrização da meta heurística no escalonamento autónomo onde os valores que iraim competir seriam calculados por métricas ou gerados aleatoriamente dentro de intervalos que foram mostrados ser eficazes na resolução de problemas deste tipo na comunidade do escalonamento. Algo que também poderia ser interessante implementar era se o valor da margem para o critério *must be* alterasse dinamicamente, dependendo do

desempenho do sistema, uma vez que se constatou que uma margem fixa poderá ter um grande impacto na qualidade do escalonamento. Portanto, metodologias que permitem o ajuste automático da margem também serão analisadas.

BIBLIOGRAFIA E OUTRAS FONTES DE INFORMAÇÃO

7 BIBLIOGRAFIA E OUTRAS FONTES DE INFORMAÇÃO

Ahmadi, E., Zandieh, M., Farrokh, M., & Emami, S. M. (2016). A multi objective optimization approach for flexible job shop scheduling problem under random machine breakdown by evolutionary algorithms. *Computers & Operations Research*, *73*, 56–66. https://doi.org/10.1016/J.COR.2016.03.009

Akers, S. B., & Friedman, J. (1955). A Non-Numerical Approach to Production Scheduling Problems. *Journal of the Operations Research Society of America*, *3*(4), 429–442. https://doi.org/10.1287/opre.3.4.429

Artiba, A., & Elmaghraby, S. E. (1996). *The Planning and Scheduling of Production Systems: Methodologies and applications*. Springer US.

Aytug, H., Lawley, M. A., McKay, K., Mohan, S., & Uzsoy, R. (2005). Executing production schedules in the face of uncertainties: A review and some future directions. *European Journal of Operational Research*, *161*(1), 86–110. https://doi.org/10.1016/J.EJOR.2003.08.027

Baker, K. R., & Trietsch, D. (2009). *Principles of Sequencing and Scheduling*. Wiley.

Blazewicz, J., Ecker, K. H., Pesch, E., Schmidt, G., & Weglarz, J. (2007). *Handbook on Scheduling: From Theory to Applications*. Springer Berlin Heidelberg. Retrieved from https://books.google.pt/books?id=9-WNzKdDzKIC

Blum, C., & Roli, A. (2003). Metaheuristics in combinatorial optimization. *ACM Computing Surveys*, *35*(3), 268–308. https://doi.org/10.1145/937503.937505

Bolboacă, S., Jäntschi, L., Bolboacă, S. D., & Jäntschi, L. (2007). Design of Experiments: Useful Orthogonal Arrays for Number of Experiments from 4 to 16. *Entropy*, *9*(4), 198–232. https://doi.org/10.3390/e9040198

Boussaïd, I., Lepagnot, J., & Siarry, P. (2013). A survey on optimization metaheuristics. *Information Sciences*, *237*, 82–117. https://doi.org/10.1016/J.INS.2013.02.041

Cavazzuti, M. (2012). *Optimization Methods: From Theory to Design Scientific and Technological Aspects in Mechanics*. Springer Berlin Heidelberg. Retrieved from https://books.google.pt/books?id=i9S_Ji6N0MoC

Černý, V. (1985). Thermodynamical approach to the traveling salesman problem: An efficient simulation algorithm. *Journal of Optimization Theory and Applications*, *45*(1), 41–51. https://doi.org/10.1007/BF00940812

CHURCH, L. K., & UZSOY, R. (1992). Analysis of periodic and event-driven rescheduling policies in dynamic shops. *International Journal of Computer Integrated Manufacturing*, *5*(3), 153–163. https://doi.org/10.1080/09511929208944524

Costa, L. (2003). *Algoritmos evolucionários em optimização uni e multi-objectivo*.

Cowling, P., & Johansson, M. (2002). Using real time information for effective dynamic scheduling. *European Journal of Operational Research*, *139*(2), 230–244. https://doi.org/10.1016/S0377-2217(01)00355-1

Dréo, J., Chatterjee, A., Pétrowski, A., Siarry, P., & Taillard, E. (2006). *Metaheuristics for Hard Optimization: Methods and Case Studies*. Springer Berlin Heidelberg. Retrieved from https://books.google.pt/books?id=I-acEfdFZ1MC

Eglese, R. W. (1990). Simulated annealing: A tool for operational research. *European Journal of Operational Research*, *46*(3), 271–281. https://doi.org/10.1016/0377-2217(90)90001-R

Ferreirinha, L., Baptista, S., Pereira, A., Santos, A. S., Bastos, J., Madureira, A. M., &

Varela, M. L. R. (2019). A Dynamic Selection of Dispatching Rules Based on the Kano Model Satisfaction Scheduling Tool (pp. 339–346). Springer, Cham. https://doi.org/10.1007/978-3-319-91334-6_46

Ferreirinha, Luis, Baptista, S., Pereira, Â., Santos, A. S., Bastos, J., Madureira, A. M., & Varela, M. L. R. (2019). An Industry 4.0 Oriented Tool for Supporting Dynamic Selection of Dispatching Rules Based on Kano Model Satisfaction Scheduling. *FME Transactions*, *47*, 757–764. https://doi.org/10.5937/fmet1904757F

Ferreirinha, Luís, Santos, A. S., Madureira, A. M., Varela, M. L. R., & Bastos, J. A. (2020). Decision Support Tool for Dynamic Scheduling (pp. 418–427). Springer, Cham. https://doi.org/10.1007/978-3-030-14347-3_41

Gendreau, M., & Potvin, J.-Y. (Eds.). (2010). *Handbook of Metaheuristics* (Vol. 146). Boston, MA: Springer US. https://doi.org/10.1007/978-1-4419-1665-5

Ghanbarzadeh, A., Koç, E., Otri, S., Rahim, S., & Zaidi, M. (2006). The Bees Algorithm — A Novel Tool for Complex Optimisation Problems. *Intelligent Production Machines and Systems*, 454–459. https://doi.org/10.1016/B978-008045157-2/50081-X

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

Glover, F. W., & Kochenberger, G. A. (2006). *Handbook of Metaheuristics*. Springer US. Retrieved from https://books.google.pt/books?id=P-HpBwAAQBAJ

Haridass, K., Valenzuela, J., Yucekaya, A. D., & McDonald, T. (2014). Scheduling a log transport system using simulated annealing. *Information Sciences*, *264*, 302–316. https://doi.org/10.1016/j.ins.2013.12.005

Herrmann, J. W. (2006). *Handbook of Production Scheduling*. Springer US.

Högström, C., Rosner, M., & Gustafsson, A. (2010). How to create attractive and unique customer experiences. *Marketing Intelligence & Planning*, *28*(4), 385–402. https://doi.org/10.1108/02634501011053531

Karaboga, D. (2005). *An idea based on honey bee swarm for numerical optimization*.

Kirkpatrick, S., Gelatt, C. D., & Vecchi, M. P. (1983). Optimization by simulated annealing. *Science (New York, N.Y.)*, *220*(4598), 671–680. https://doi.org/10.1126/science.220.4598.671

Kundakcı, N., & Kulak, O. (2016). Hybrid genetic algorithms for minimizing makespan in dynamic job shop scheduling problem. *Computers & Industrial Engineering*, *96*, 31–51. https://doi.org/10.1016/J.CIE.2016.03.011

Leung, J. Y. T. (2004). *Handbook of Scheduling: Algorithms, Models, and Performance Analysis*. CRC Press. Retrieved from https://books.google.pt/books?id=MAY1ZstmGPkC

Liu, W., & Yan, F. (2012). On the population diversity control of evolutionary algorithms for production scheduling problems. In *2012 IEEE Fifth International Conference on Advanced Computational Intelligence (ICACI)* (pp. 468–473). https://doi.org/10.1109/ICACI.2012.6463208

Löfgren, M., Witell, L., & Gustafsson, A. (2011). Theory of attractive quality and life cycles of quality attributes. *The TQM Journal*, *23*(2), 235–246. https://doi.org/10.1108/17542731111110267

Lopez, P., & Roubellat, F. (Eds.). (2008). *Production Scheduling*. London, UK: ISTE. https://doi.org/10.1002/9780470611050

Madureira, A., Pereira, I., Pereira, P., & Abraham, A. (2014). Negotiation mechanism for self-organized scheduling system with collective intelligence. *Neurocomputing*, *132*, 97–110. https://doi.org/10.1016/J.NEUCOM.2013.10.032

Madureira, Ana, Ramos, C., & Silva, S. do C. (2003). Using genetic algorithms for dynamic scheduling. In *I14th Annual Production and Operations Management Society Conference (POMS 2003)*.

Marković, D., Petrović, G., Ćojbašić, Ž., & Marinković, D. (2013). A comparative analysis of metaheuristic maintenance optimization of refuse collection vehicles using the Taguchi experimental design. *Transactions of FAMENA*, *36*(4), 25–38.

Mesghouni, K., Hammadi, S., & Borne, P. (1996). Production job-shop scheduling using genetic algorithms. In *1996 IEEE International Conference on Systems, Man and Cybernetics. Information Intelligence and Systems (Cat. No.96CH35929)* (Vol. 2, pp. 1519–1524 vol.2). https://doi.org/10.1109/ICSMC.1996.571372

Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H., & Teller, E. (1953). Equation of State Calculations by Fast Computing Machines. *The Journal of Chemical Physics*, *21*(6), 1087–1092. https://doi.org/10.1063/1.1699114

Montero, E., Riff, M.-C., & Neveu, B. (2014). A beginner's guide to tuning methods. *Applied Soft Computing*, *17*, 39–51. https://doi.org/10.1016/J.ASOC.2013.12.017

O'Donovan, R., Uzsoy, R., & McKay, K. N. (1999). Predictable scheduling of a single machine with breakdowns and sensitive jobs. *International Journal of Production Research*, *37*(18), 4217–4233. https://doi.org/10.1080/002075499189745

Ouelhadj, D., & Petrovic, S. (2009). A survey of dynamic scheduling in manufacturing systems. *Journal of Scheduling*, *12*(4), 417–431. https://doi.org/10.1007/s10951-008-0090-8

Panwalkar, S. S., & Iskander, W. (1977). A Survey of Scheduling Rules. *Operations Research*, *25*(1), 45–61. https://doi.org/10.1287/opre.25.1.45

Park, M.-W., & Kim, Y.-D. (1998). A systematic procedure for setting parameters in simulated annealing algorithms. *Computers & Operations Research*, *25*(3), 207–217. https://doi.org/10.1016/S0305-0548(97)00054-3

Pereira, I. (2009). *Aspectos de aprendizagem em optimização*. Instituto Politécnico do Porto. Instituto Superior de Engenharia do Porto.

Pinedo, M. (2012). *Scheduling : theory, algorithms, and systems*.

Rahmani, D., & Ramezanian, R. (2016). A stable reactive approach in dynamic flexible flow shop scheduling with unexpected disruptions: A case study. *Computers & Industrial Engineering*, *98*, 360–372. https://doi.org/10.1016/J.CIE.2016.06.018

Rajendran, C., & Holthaus, O. (1999). A comparative study of dispatching rules in dynamic flowshops and jobshops. *European Journal of Operational Research*, *116*(1), 156–170. https://doi.org/10.1016/S0377-2217(98)00023-X

Ramasesh, R. (1990). Dynamic job shop scheduling: A survey of simulation research. *Omega*, *18*(1), 43–57. https://doi.org/10.1016/0305-0483(90)90017-4

Sabuncuoglu, I, & Bayız, M. (2000). Analysis of reactive scheduling problems in a job shop environment. *European Journal of Operational Research*, *126*(3), 567–586. https://doi.org/10.1016/S0377-2217(99)00311-2

Sabuncuoglu, Ihsan, & Karabuk, S. (1999). Rescheduling frequency in an FMS with uncertain processing times and unreliable machines. *Journal of Manufacturing Systems*, *18*(4), 268–283. https://doi.org/10.1016/S0278-6125(00)86630-3

Sadegheih, A. (2006). Scheduling problem using genetic algorithm, simulated annealing and the effects of parameter values on GA performance. *Applied Mathematical Modelling*, *30*(2), 147–154. https://doi.org/10.1016/J.APM.2005.03.017

Santos, A. B. G. S. (2013). *Afetação de recursos em sistemas de produção*. Instituto Politécnico do Porto. Instituto Superior de Engenharia do Porto.

Santos, A. B. G. S. e. (2015). *Análise do Desempenho de Técnicas de Otimização no Problema de Escalonamento*.

Sen, T., Sulek, J. M., & Dileepan, P. (2003). Static scheduling research to minimize weighted and unweighted tardiness: A state-of-the-art survey. *International Journal of Production Economics*, *83*(1), 1–12. https://doi.org/10.1016/S0925-5273(02)00265-7

Siriwardene, N. R., & Perera, B. J. C. (2006). Selection of genetic algorithm operators for urban drainage model parameter optimisation. *Mathematical and Computer Modelling*, *44*(5–6), 415–429. https://doi.org/10.1016/J.MCM.2006.01.002

Sule, D. R. (2007). *Production Planning and Industrial Scheduling*. CRC Press. https://doi.org/10.1201/9781420044218

Talbi, E.-G. (2009). *Metaheuristics : from design to implementation*. John Wiley & Sons.

Terekhov, D., Down, D. G., & Beck, J. C. (2014). Queueing-theoretic approaches for dynamic scheduling: A survey. *Surveys in Operations Research and Management Science*, *19*(2), 105–129. https://doi.org/10.1016/J.SORMS.2014.09.001

Varela, M. L. R., & Ribeiro, R. A. (2014). Distributed Manufacturing Scheduling Based on a Dynamic Multi-criteria Decision Model (pp. 81–93). Springer, Cham. https://doi.org/10.1007/978-3-319-06323-2_6

Varela, Maria Leonilde Rocha. (2007). *Uma contribuição para o escalonamento da produção baseado em métodos globalmente distribuídos*.

Vieira, G. E., Herrmann, J. W., & Lin, E. (2003). Rescheduling Manufacturing Systems: A Framework of Strategies, Policies, and Methods. *Journal of Scheduling*, *6*(1), 39–62. https://doi.org/10.1023/A:1022235519958

Wu, G., Wang, H., Pedrycz, W., Li, H., & Wang, L. (2017). Satellite observation scheduling with a novel adaptive simulated annealing algorithm and a dynamic task clustering strategy. *Computers & Industrial Engineering*, *113*, 576–588. https://doi.org/10.1016/J.CIE.2017.09.050

Xhafa, F., & Abraham, A. (Eds.). (2008). *Metaheuristics for Scheduling in Industrial and Manufacturing Applications* (Vol. 128). Berlin, Heidelberg: Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-540-78985-7

Yang, W. H., & Tarng, Y. S. (1998). Design optimization of cutting parameters for turning operations based on the Taguchi method. *Journal of Materials Processing Technology*, *84*(1–3), 122–129. https://doi.org/10.1016/S0924-0136(98)00079-X

Zandieh, M., Amiri, M., Vahdani, B., & Soltani, R. (2009). A robust parameter design for multi-response problems. *Journal of Computational and Applied Mathematics*, *230*(2), 463–476. https://doi.org/10.1016/J.CAM.2008.12.019