Universidade de Aveiro

2020

José Filipe Mota Ferreira

Otimização e apoio à decisão no escalonamento da produção de cápsulas de rosca vinícolas em ambiente *Flexible Job Shop*

Universidade de Aveiro

2020

José Filipe Mota Ferreira

Otimização e apoio à decisão no escalonamento da produção de cápsulas de rosca vinícolas em ambiente *Flexible Job Shop*

Relatório de Projeto apresentado à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Engenharia e Gestão Industrial, realizada sob a orientação científica do Doutor Rui Jorge Ferreira Soares Borges Lopes, Professor Auxiliar do Departamento de Economia, Gestão, Engenharia Industrial e Turismo da Universidade de Aveiro

Dedico este trabalho à minha família por todo apoio incondicional.

o júri

presidente	Doutora Ana Raquel Xambre Professora Auxiliar, Universidade de Aveiro
vogal	Doutora Tânia Rodrigues Pereira Ramos Professora Auxiliar, Instituto Superior Técnico de Lisboa
vogal	Doutor Rui Jorge Ferreira Soares Borges Lopes Professor Auxiliar, Universidade de Aveiro

agradecimentos

Agradeço à organização Américo Coelho Relvas Sucrs, S.A e Dr. Ana Ferreira, pela oportunidade de estágio, em especial à minha orientadora Eng.ª Ana Mata por toda a disponibilidade, apoio e sobretudo pela sua boa disposição, mas também ao encarregado Victor Nuno e ao colaborador Joel Gomes que sempre se mostraram disponíveis para me ajudar.

Ao meu orientador na Universidade de Aveiro, Professor Rui Borges Lopes.

Aos meus colegas e amigos que fiz na Universidade de Aveiro que me acompanharam neste percurso universitário, por todos trabalhos, exames e experiências vividas em conjunto.

Por fim, à minha família e aos meus amigos, que me acompanharam durante toda a vida, sem eles muito provavelmente não teria chegado até aqui.

palavras-chave

Escalonamento, CPLEX, *Flexible Job Shop Problem* (FSJP), Métodos exactos e aproximados, heuristicas.

resumo

Hoje em dia, as organizações industriais realizam investimentos recorrentes para melhorar os seus processos produtivos e procuram não só a constante redução de custos como também a maximização da sua eficiência e competitividade.

O presente trabalho foi elaborado na organização, Américo Coelho Relvas Sucessores, SA. esta tem como atividade principal a produção de variadas cápsulas personalizáveis para garrafas. Todavia, o departamento onde este trabalho é desenvolvido é ainda bastante recente, possui menos de uma década de existência e é portanto, o setor mais recente da empresa. Este está a ter um elevado crescimento não só no nº de encomendas, mas também em investimento de máquinas. Tudo isto leva a um aumento da dificuldade de escalonamento.

Devido à enorme complexidade de problemas de escalonamento em ambiente industrial, houve uma necessidade de desenvolver um modelo recorrendo ao *software* IBM ILOG CPLEX e um conjunto de ferramentas de suporte para modelar o chão de fabrica deste setor e dar uma melhor resposta ao planeamento de trabalhos.

Na parte final deste projecto, o problema e o modelo utilizado para o ultrapassar serão apresentados em pormenor. Simultaneamente, será apresentado um estudo sobre duas funções objectivo a utilizar no modelo, e a que apresentar os melhores resultados será a utilizada no modelo final

Finalmente, é feito um estudo comparativo entre o escalonamento real e o escalonamento feito através do algoritmo criado. Estes resultados são apresentados em detalhe e os mesmos representam uma considerável melhoria de 20% no cumprimento dos prazos de entrega, para além de outros benefícios.

keywords

Scheduling, CPLEX, Flexible Job Shop Problem (FSJP), exact and approximate methods, heuristics.

abstract

Nowadays, industrial organizations frequently allocate investments to improve their production processes and seek not only the constant reduction of costs but also the maximization of their efficiency and competitiveness.
This project has been carried out in the organization, Américo Coelho Relvas Sucessores, SA., its main activity is the production of various customizable capsules for bottles. However, the department where this project was carried out is still quite recent, it has less than a decade of existence and is therefore the most recent sector of the company. It is having a high growth not only in the number of orders, but also in machine investment. All of this leads to an increase in the difficulty of scheduling.
Due to the enormous complexity of scheduling problems in an industrial environment, there was a need to develop a model using IBM ILOG CPLEX software and a set of support tools to model the manufacturing floor of this sector and provide a better response to job planning.
Towards the end of this project, the problem and the model used to overcome the problem will be presented in detail. Simultaneously, a study will be presented on two objective functions that can be used in the model, and the one that presents the best results will be the one used in the final model.
Finally, a comparative study is made between the actual scheduling and the scheduling made through the developed algorithm. These results are presented in detail and they represent a considerable 20% improvement in meeting delivery deadlines, in addition to other benefits.

Índice

ÍNDICE DE FIGURAS

ÍNDICE DE TABELAS

ACRÓNIMOS

ACO	*Ant Colony Optimization*
B&B	*Branch and Bound*
EDD	*Earliest Due Date*
FIFO	*First In, First Out*
FJSP	*Flexible Job-Shop Problem*
GA	*Genetic Algorithm*
GRASP	*Greedy Randomized Adaptive Search Procedure*
ILS	*Iterated Local Search*
JSP	*Job-Shop Problem*
LIFO	*Last In, First Out*
LNS	*Large Neighborhood Search*
LOT	*Longest Operation Time*
LPT	*Longest Processing Time*
MILP	*Mixed Integer Linear Programming*
MIP	*Mixed Integer Programming*
PLIM	Programação Linear Inteira Mista
PSO	*Particle Swarm Optimization*
SA	*Simulated Annealing*
SOT	*Shortest Operation Time*
SPT	*Shortest Processing Time*
TS	*Tabu Search*
VND	*Variable Neighborhood Descent*

1. Introdução

1.1 Contextualização

Num contexto global cada vez mais competitivo, as organizações industriais alocam recorrentemente investimentos para melhorar os seus processos produtivos. Estas procuram não só a constante redução de custos como também a maximização da sua eficiência e competitividade. No caso de um aumento repentino do número de ordens de fabrico, aumento da diversidade de personalização de produtos ou a introdução de novos equipamentos tais como máquinas, verifica-se com frequência que o sistema de planeamento da produção não consegue dar resposta a estas evoluções.

Empresas de produção por encomenda, ou, em inglês *make-to-order*, caracterizam-se pela fabricação de uma grande variedade de produtos personalizáveis e em pequenos lotes. Como consequência desta personalização, resulta em produtos com sequências de processamento muito variadas e com diferentes datas de entrega. Neste tipo de empresas é necessário apresentar equipamentos flexíveis. Sistemas de produção *make-to-order* geralmente pertencem à família de produção em ambiente *job-shop*.

A otimização de recursos é fundamental para o sucesso de todas as organizações. Posto isto, uma gestão eficiente dos recursos normalmente traz diversas vantagens para os *stakeholders* da organização, tais como a redução de custos, melhoria do serviço ao cliente, aumento da produtividade, entre outras. Ainda que a otimização de recursos seja importante para todo o tipo de organizações, é nas indústrias de manufatura, particularmente nos seus departamentos de produção, que a aplicação de ferramentas de programação e escalonamento assume uma maior importância.

A programação de tarefas, ou, em outras palavras um escalonamento requer total conhecimento do processo produtivo, das suas limitações de recursos, e respetivamente das restrições de sequência das operações (Varela & Carmo-Silva, 2008). Escalonar consiste na ordenação de operações e alocá-las em espaços de tempo específicos tendo em conta restrições técnicas e de capacidade. De acordo com Soleimani et al. (2019) escalonar a produção pode levar a uma redução significativa no tempo de produção, bem como um melhor aproveitamento dos recursos fabris.

Nas últimas décadas, o problema de escalonamento da produção em ambiente *Job-Shop* tem recebido enorme atenção por parte de investigadores. Tal deve-se a duas razoes: a primeira deve-se ao facto da sua enorme complexidade, este é considerado como NP-Difícil (Garey et al., 1976). E a segunda deve-se ao facto que este tipo de ambiente está cada vez mais presente muitas indústrias de manufatura. Este consiste em sequenciar um conjunto de trabalhos num grupo de máquinas, sob uma determinada ordem de processamento.

O foco principal deste trabalho trata-se numa análise de métodos de resolução do problema de escalonamento em ambiente de *Job-Shop* Flexível (FJSP), este consiste numa extensão do problema clássico de *Job-Shop* (JSP). Contrariamente ao JSP, é possível que uma operação seja executada ou processada em máquinas alternativas. É considerado por

muitos, um problema mais próximo do mundo real industrial e por outro lado, esta flexibilidade permite que uma operação seja processa em máquinas alternativas, aumenta o número de caminhos alternativos de um trabalho e consequentemente aumenta significativamente o número de soluções admissíveis.

1.2 Objetivos

Os principais objetivos a serem alcançados com este trabalho são os seguintes:

- Abordar um problema de escalonamento da produção de uma organização e simultaneamente abordar métodos de resolução deste tipo de problemas;
- Desenvolver um modelo para a obtenção de um escalonamento da produção e comparar com o atual, de forma a aumentar a eficiência da mesma;
- Comparar a solução dos problemas em termos de *makespan* e outras medidas de desempenho tais como minimizar soma do atraso máximo (*tardiness)*;
- Testar e validar as abordagens desenvolvidas para compreender melhor o seu desempenho e fornecer exemplos de referência para estudos futuros de problemas de escalonamento com características semelhantes.

1.3 Metodologia

Com vista a alcançar os objetivos definidos, foi seguida a seguinte metodologia.

Numa primeira fase, iniciou-se um processo de conhecimento do funcionamento da organização, desde o conhecimento de todos os departamentos, conhecimento total do processo produtivo, equipamentos do setor, cadências de produção e turnos de trabalho.

De seguida, foi levado a cabo uma pesquisa sobre as encomendas em atraso e as razões pelas quais um planeamento semanal é, por vezes, insuficiente.

Posto isto, iniciou-se uma pesquisa sobre como resolver este problema e, devido à enorme complexidade de escalonamento em *job-shop,* optou-se pela procura de um *software* onde fosse possível modelar o chão-de-fabrica do setor *Relwine* e obter um escalonamento eficiente. Após encontrado o *software*, foi necessário familiarizar-se com este e deu-se início à elaboração do modelo e ferramentas de apoio a este modelo.

Em paralelo a este conjunto de fases, foi também realizada uma pesquisa bibliográfica com o objetivo de reunir e aprofundar o conhecimento sobre os tópicos relacionados com o trabalho.

Por fim, ao longo da realização deste trabalho o modelo final sofreu várias iterações até conseguir representar de forma fidedigna o chão-de-fabrica do setor pretendido e os escalonamentos obtidos foram durante semanas comparados com o escalonamento real.

1.4 Estrutura

Este documento está dividido em cinco capítulos.

O primeiro capítulo dará uma sucinta introdução e contextualização à temática que levou à realização deste trabalho, bem como os objetivos pretendidos deste trabalho e a metodologia definida para os atingir.

O segundo capítulo, corresponde à revisão de literatura. Aqui é abordado o problema de *job-shop* clássico, e então, a sua extensão, *Job Shop Scheduling* Flexível (FJSP). De seguida, são apresentadas abordagens exatas e aproximadas, dentro dos quais estão inseridas abordagens heurísticas e meta-heurísticas.

O terceiro capítulo apresentará a organização onde foi desenvolvido este trabalho, de modo a dar a conhecer a organização em questão, bem como o seu processo produtivo e parte dos produtos produzidos por esta organização.

O quarto capítulo consiste na apresentação do problema em questão e todo o trabalho realizado para superar este problema. Tendo em vista todas as técnicas discutidas nos capítulos anteriores bem como uma análise e discussão dos resultados obtidos.

Por último, no quinto capítulo apresentam-se as conclusões obtidas neste trabalho, bem como possíveis propostas de melhorias a aplicar no futuro.

2. Revisão de literatura

2.1 O problema Job-Shop Clássico

O termo *job-shop* é utilizado em ambientes de produção cujas características são a produção de produtos altamente variáveis, normalmente em pequenas quantidades e de acordo com determinadas especificações do cliente, ou seja, normalmente são produtos altamente personalizáveis.

De acordo com Sonmez & Baykasoglu (1998), problemas de escalonamento tipo *job-shop* (JSP) pertencem a um ramo de programação de produção, que está entre os mais difíceis problemas de otimização combinatória. Estes surgiram por volta da década de 50, correspondem a problemas de otimização combinatória que alocam várias operações a determinados recursos e têm como objetivo encontrar uma sequência de operações para cada máquina/recurso, respeitando as restrições estabelecidas, com vista a maximizar ou minimizar um determinado objetivo (Huang, 2009).

Encontrar uma solução ideal para este tipo de problema num período de tempo considerado aceitável é bastante desafiador devido à enorme complexidade do ambiente de trabalho em *job-shop*. Sendo provado por Garey et al. (1976) como NP-Difícil.

Muito sinteticamente, o problema clássico de JSP é enunciado da seguinte forma (ver Figura 1). Considere-se um conjunto de *n* trabalhos, ou, em inglês *jobs,* cada trabalho é constituído por uma sequência previamente definida de várias operações. Cada operação é processada individualmente e ininterruptamente numa máquina, pertencentes a um conjunto M formada por m máquinas e toda a operação tem um tempo de processamento associado.

São várias as medidas ou métricas de desempenho deste sequenciamento. Contudo, as mais frequentes são a minimização do *makespan*, minimização do tempo total de fluxo, a minimização do atraso máximo também conhecido como *tardiness*, minimização do número de tarefas atrasadas e, por fim, o balanceamento ou equilíbrio das cargas de trabalho das máquinas.

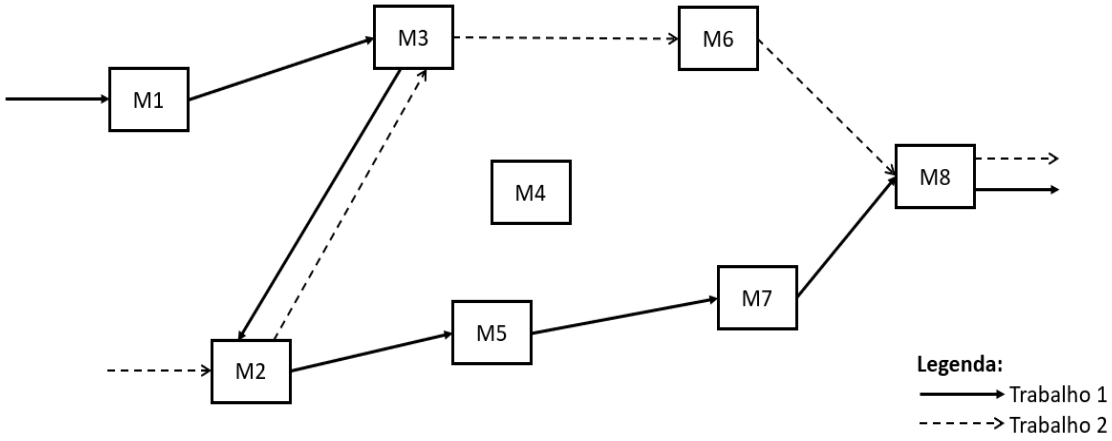


Figura 1 - Ambiente *Job-Shop* baseado em İliş (2004)

2.2 O problema de *Job-Shop* Flexível

Quando se consideram casos reais, o JSP possui a limitação de possuir uma sequência fixa de operações para cada trabalho. Surge então uma extensão denominada de problema *Flexible Job Shop Scheduling* (FJSP). Nesta perspetiva, é possível que uma operação de um dado trabalho possa ser processado em máquinas alternativas, isto é, um mesmo trabalho possa vir a ter variações no sequenciamento da sua produção (Ho & Tay, 2004a).

Gholami & Sotskov (2014) referem que a utilização de um conjunto de máquinas em paralelo permite aumentar a taxa de produção e evita a paragem da linha quando ocorre uma falha do recurso. Por sua vez, as máquinas em paralelo podem ser idênticas, uniformes ou não relacionadas. E como faz todo o sentido, os recursos/máquinas idênticas operam à mesma velocidade, os restantes podem ou não operar a velocidades diferentes (Potts & Kovalyov, 2000). Na Figura 2, é possível observar as tais máquinas idênticas em paralelo (M1 e M2; M3, M4 e M5; M7 e M8).
De acordo com a flexibilidade, Kacem et al. (2002) classifica o FJSP em dois tipos, conforme se segue:

1. Total FJSP (T-FJSP): cada operação pode ser processada numa das M máquinas disponíveis.
2. Parcial FJSP (P-FJSP): cada operação pode ser processada numa máquina pertencente a um sub-conjunto de máquinas M disponíveis.

Neste relatório o Parcial FJSP terá mais ênfase e de acordo com Kacem et al. (2002), o P-FJSP é mais complexo que o T-FJSP, este faz com que o esforço de procura e o tempo de computação aumentem.

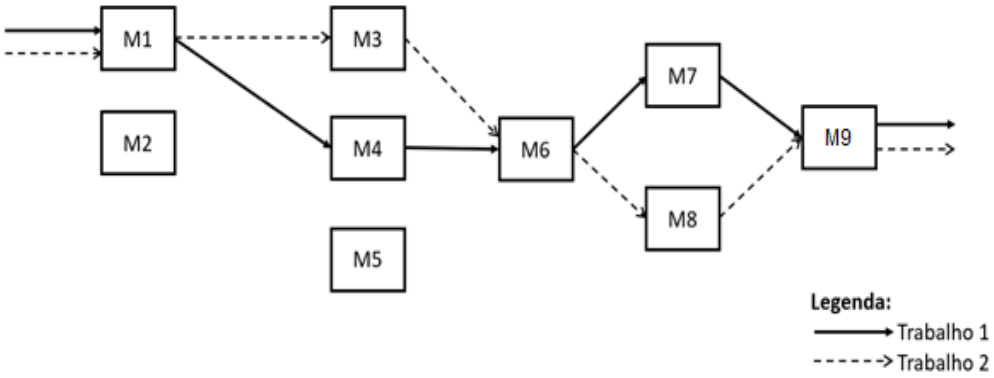


Figura 2 - Ambiente *Job-Shop* Flexível

Formulação do problema

De acordo com Pezzella et al. (2008), o FJSP pode ser descrito da forma seguinte. Dado um conjunto n trabalhos $J = \{J1, J2, ..., Jn\}$ de trabalhos ou encomendas para serem processados num conjunto m máquinas $M = \{M1, M2, ..., Mm\}$. Cada trabalho Ji é constituído por uma sequência de operações O_{i1}, O_{i2}, …, O_{ij}. Estas operações são realizadas uma após a outra de acordo com a sequência previamente definida (restrição de precedência).

Para toda a operação O_{ij} há um conjunto de equipamentos ou máquinas alternativas capazes de realizar essa operação. Este problema tem como principal objetivo encontrar um escalonamento que minimize o tempo total de produção, ou seja, o *makespan*. O *makespan* corresponde ao instante ou momento final de processamento da última operação do sistema e é normalmente definido por C_{max}.

Mais ainda, este tipo de problema está associado às seguintes preposições (Nouri et al., 2018):

- Todas as máquinas e todos os trabalhos estão disponíveis no momento inicial, ou seja, T=0;
- Os períodos de processamento de todas as operações nas respetivas máquinas são conhecidos com antecedência, normalmente são dados em horas ou minutos;
- Cada operação poderá ser realizada por qualquer máquina pertencente subconjunto de máquinas especificas e cada operação é realizada apenas uma vez;
- Cada máquina pode apenas processar uma operação de cada vez;
- Nenhuma máquina pode ser libertada enquanto a execução da operação não esteja concluída.
- Operações pertencentes a diferentes trabalhos podem ser processadas em paralelo/simultâneo;
- Só é realizada uma operação de um trabalho quando a operação antecedente desse trabalho desta esteja completa.

Tendo em conta as características apresentadas acima, o problema pode ser resumido de acordo com a Tabela 1. As linhas correspondem às operações a serem realizadas e as

colunas às máquinas. As entradas são os tempos de processamento correspondentes a cada operação, por fim o símbolo "-" significa que as operações não podem ser executadas na máquina correspondente.

Tabela 1 - Exemplo de um P-FJSP com respetivos tempos de processamento.

Trabalho	Operação	M_1	M_2	M_3	M_4
J_1	O_{11}	2	9	4	5
	O_{12}	–	6	–	4
J_2	O_{21}	1	–	5	–
	O_{22}	3	8	6	–
	O_{23}	–	5	4	3
J_3	O_{31}	–	6	6	–
	O_{32}	–	3	–	5
	O_{33}	1	2	5	–

2.3 Representação de problemas e soluções de escalonamento do tipo *Job-Shop*

Existem diversas maneiras de representar as soluções dos problemas de escalonamento do tipo *Job-Shop*, nesta seção serão apresentadas as duas formas mais comuns na literatura: os gráficos de *Gantt* e os grafos disjuntivos. Ambos têm a finalidade de facilitar uma visualização dos resultados dos problemas. De seguida, analisam-se ambas abordagens com um exemplo prático para cada e também com uma breve contextualização.

2.3.1 Gráfico de *Gantt*

Segundo Cox et al. (1992), o gráfico de *Gantt* é o tipo mais antigo e mais conhecido de gráfico concebido especificamente para mostrar graficamente a relação entre o desempenho previsto e o desempenho real. De acordo com Porter (1929), estes diagramas foram bastante revolucionários, pois relacionavam as atividades com o tempo num modo gráfico que permitia que se calculasse o horário de trabalho.

O gráfico de *Gantt* caracteriza-se pela sua facilidade de leitura e de análise clara. Foram desenvolvidos em 1911 por *Henry Gantt* com o objetivo de visualizar produção programada e comparar esta com a produção real. É um gráfico constituído por um sistema de eixos, o eixo das abscissas consiste numa escala temporal que representa o tempo de duração das operações e eixo das ordenadas representa o conjunto de máquinas. Cada retângulo representa uma operação e o seu comprimento é definido pelo tempo de processamento dessa operação.

Em jeito de demonstração, a figura 3 consiste numa solução admissível para o problema de *job-shop* flexível parcial (P-FJSP) encontrado na tabela 1, neste o *makespan* é de 14 unidades temporais. Já a figura 4 consiste noutra possível solução admissível onde o *makespan* é de 12 unidades temporais, ou seja, em termos de *makespan* é uma solução preferível. É então possível ter uma noção que para num exemplo tão básico como o da Tabela 1 representado por 4 máquinas e 3 trabalhos podem existir diversas soluções para o mesmo problema.

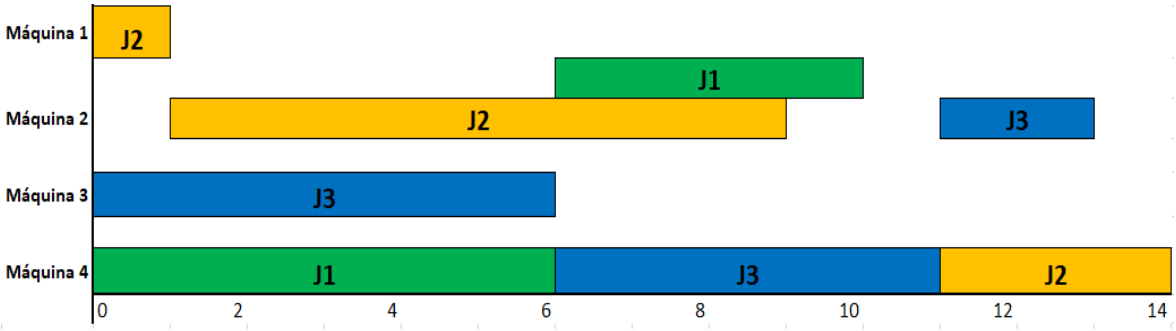


Figura 3 - Representação de uma solução admissível para o P-FJSP - Solução 1

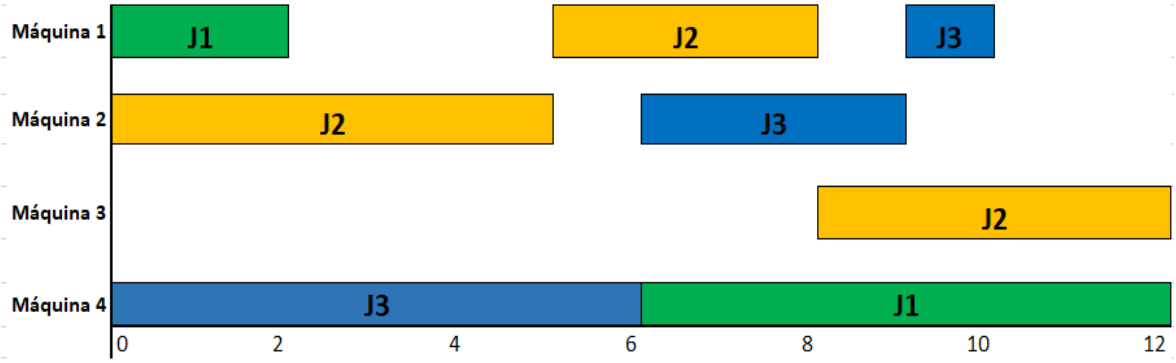


Figura 4 - Representação de uma solução admissível para o P-FJSP - Solução 2

2.3.2 Grafos Disjuntivos

O grafo disjuntivo de Roy & Sussmann (1964) é uma representação gráfica geralmente utilizada para representar problemas de escalonamento ou sequenciação. Mais tarde, Błażewicz et al. (1996b) apresenta uma nova abordagem especialmente desenvolvida para problemas de *job-shop*. Em seguida formula-se a representação deste modelo.

Existe um nó para cada operação, adicionalmente foram adicionados dois nós fictícios, os quais representam o início (I) e o fim (F). O grafo disjuntivo é representado por *G = (O, A, E).* Tal que:

- "**O"**: refere-se aos nós do grafo, para cada operação existe um nó com peso igual ao tempo de processamento dessa operação. Além disso, como referido acima existe nós com peso nulo, neste caso correspondem à operação inicial (I) e a operação final (F).
- **"A"**: corresponde ao conjunto de arcos conjuntivos relativos à sequenciação de operações. Estes respeitam as condições de precedência entre operações de um mesmo trabalho.
- **"E"**: consiste no conjunto de arcos disjuntivos. Os arcos disjuntivos representam o conjunto ou grupo de operações a realizar pela mesma máquina.

De modo a garantir uma solução válida, o grafo resultante da orientação dos arcos disjuntivos deve ser acíclico, ou seja, não deve formar caminhos em ciclo, de maneira a garantir a precedência das operações. O caminho crítico é definido como o caminho mais longo da disjuntiva gráfico. As operações no caminho crítico são denominadas operações críticas (Vital-Soto et al., 2020). O valor do *makespan* é exatamente igual ao do caminho crítico.

Apesar de ser menos utilizado pela comunidade científica, os grafos são outra forma de representar problemas de *job-shop* para além dos diagramas de *Gantt*. Para exemplificar o funcionamento dos grafos disjuntivos será apresentado o seguinte problema de JSP, uma vez que é menos complexo que um problema de FJSP.

A tabela 2 contem um exemplo que será usado para demonstrar um o funcionamento dos grafos disjuntivos.

Tabela 2 - Trabalhos e respetivos tempos de fabrico (JSP) baseado em Yamada & Nakano (2000)

Operação			**Tempo de Processamento**
Trabalho	**No. Operação**	**Máquina**	
1	1	1	3
	2	2	3
	3	3	3
2	1	2	4
	2	3	3
	3	1	2
	1	1	3

3	2	2	2
	3	3	1

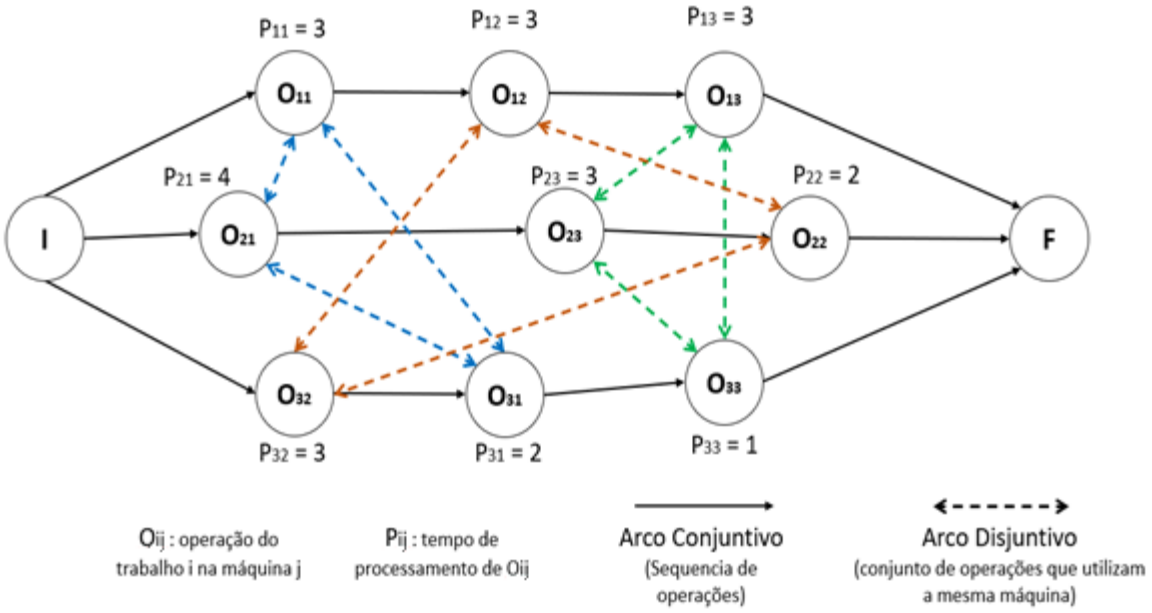


Figura 5 - Grafo Disjuntivo baseado em Yamada & Nakano (2000)

Na Figura 6 é representado o caminho crítico, relativo ao problema da Tabela 1. O caminho critico consiste na soma de 0 + 4 + 3 + 2 + 1 + 3 + 3 + 0 que resulta num *makespan* de 16 unidades temporais. Para uma melhor compreensão dos resultados deste problema foi criado a Figura 7 onde é possível fazer a comparação entre os grafos disjuntivos e os gráficos de Gantt.

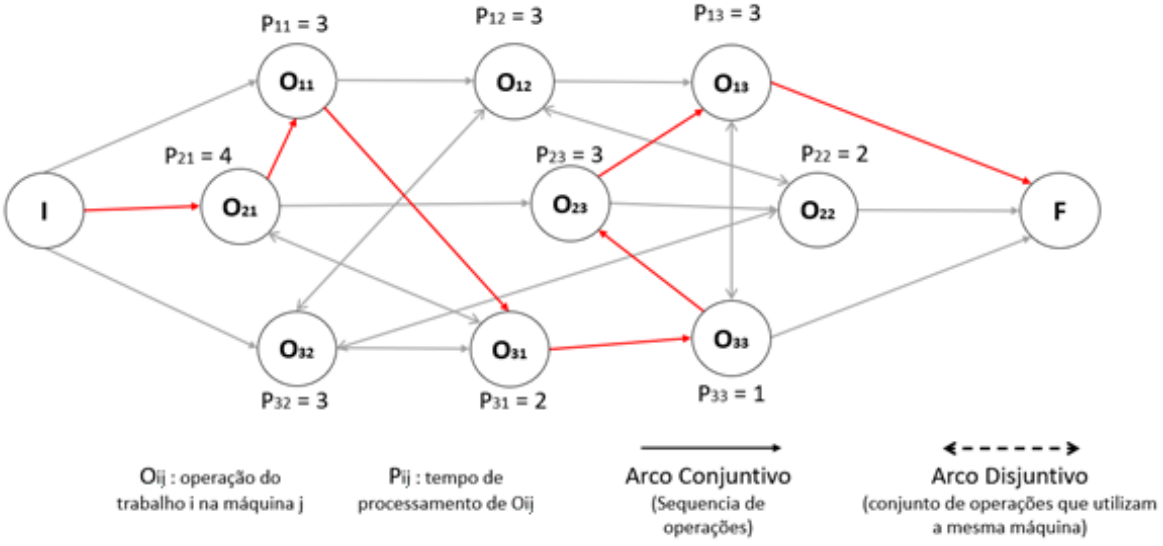


Figura 6 - Caminho critico do Grafo Disjuntivo

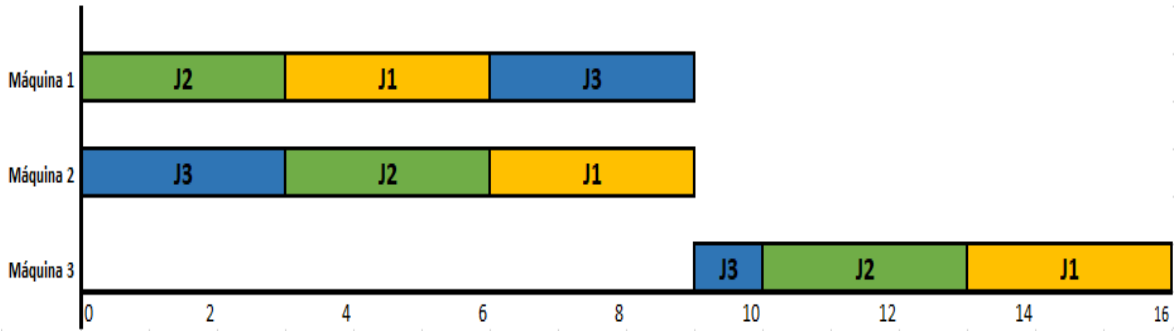


Figura 7 - Gráfico de *Gantt* e respetivo ao caminho crítico

2.4 Medidas de desempenho

Medidas de desempenho estão normalmente associadas às estratégias de negócio adotadas pelas organizações. Os critérios são definidos na função objetivo do modelo e expressos de forma a maximizar ou minimizar um ou, até mesmo, vários determinados critérios. Estes critérios permitem avaliar e comparar um sequenciamento ou escalonamento.

A maioria dos modelos concentra-se no cumprimento de apenas um objetivo, mas no mundo real, normalmente é necessário satisfazer mais do que um objetivo em simultâneo. De acordo com Pinedo (2012), os sistemas de programação considerados mais eficientes devem ser capazes de atingir simultaneamente objetivos tais como: minimizar o tempo de conclusão de todos os trabalhos (*makespan*); maximizar a utilização das máquinas; minimizar os custos e tempos de *setup*, manter o equilíbrio das cargas de trabalho; e cumprir o prazo de entrega.

Medidas de desempenho ou funções objetivo, estão muitas vezes na origem da classificação de um problema. Lawler et al. (1982) desenvolveu uma notação de classificação para os diferentes modelos de escalonamento utilizando três variáveis ($\alpha \mid \beta \mid \gamma$) que especificam diferentes elementos inerentes aos problemas de escalonamento. A variável α descreve o ambiente do chão de fábrica; β descreve as características do processo bem como as restrições; γ descreve as medidas de desempenho, ou seja, a função objetivo do problema.

Segundo Sen & Gupta (1984), as medidas ou métricas de desempenho mais frequentemente adotadas na função objetivo de problemas de escalonamento em ambientes *job shop* correspondem ao *makespan* e as restantes são baseados em datas ou prazos de entrega tal como, o *lateness*, o atraso máximo denominado na literatura como *tardiness* e a quantidade de trabalhos atrasados. Piroozfard et al. (2018) afirma também que na área de programação, os investigadores dão sobretudo maior atenção a funções objetivo como o *makespan*, *lateness*, *tardiness* e carga de trabalho nos equipamentos (em inglês, *workload*).

De acordo com Rinnooy Kan (1976), é possível concluir que os critérios de sequenciação e de escalonamento são divididos conforme três parâmetros:

1) Tempos de conclusão, dentro destes temos como mais conhecido o *makespan*;
2) Datas de entrega, dos quais *lateness*, *tardiness*, *earliness,* entre outros;
3) Inventario e utilização de equipamento/máquinas, dos quais os mais comuns são o número médio de trabalhos à espera de máquinas, número médio de trabalhos concluídos, tempo máximo de inatividade da máquina, entre outros;

Os seguintes termos, equações e conclusões estão presentes nos trabalhos de Alharkan (2005) e de Valente & Alves (2008).

2.4.1 Minimização do *makespan*

O critério *makespan* é tipicamente representado por C_{max} e representa o instante final de processamento da última operação do sistema.

Dado que Ci representa o instante final de processamento do trabalho i, então:

$$C_{max} = \max (C_1; \ldots; C_n) \quad (1)$$

A função objetivo com maior número de publicações na literatura, consiste em:

$$\text{Min } C_{max} \quad (2)$$

A tabela 3 contêm diversos artigos com modelos matemáticos acerca o problema de *Flexible Job Shop* cujo objetivo é a minimização do *makespan*.

Tabela 3 - FSJP com o objetivo de minimizar o *makespan*

Publicação	**Objetivo**
(Fattahi, Saidi Mehrabad, et al., 2007)	Minimização do *makespan.*
(Chan et al., 2006)	Minimização do *makespan* e em simultâneo os custos de paragem das máquinas.
(Özgüven et al., 2010)	Minimização do makespan.

2.4.2 Baseado em datas de entrega

Um atraso no trabalho, *lateness* (Li), é definido como:

$$L_i = C_i - D_i \quad (3)$$

Onde Di representa a data de entrega do trabalho i. A terminologia mais frequente na literatura é denominada de "*lateness*". É de realçar que o *lateness* possa apresentar valores negativos nos casos em que a tarefa é terminada mais cedo que a respetiva data ou prazo de entrega.

Geralmente, procura-se reduzir o atraso máximo, este é denominado por Tmax e corresponde ao trabalho (Ti) com maior diferença entre o momento final de processamento (Ci) e a correspondente data de entrega (Di). Assim sendo:

$$Ti = \max \{0; Ci - Di\} \quad (4)$$

Ou seja:

$$Ti = \max \{0, Li\} \quad (5)$$

O atraso máximo (Tmax) é então caraterizado por:

$$Tmax = \max \{Ti\} \quad (6)$$

A função objetivo mais frequente na literatura e que tem em consideração datas de entrega é a seguinte:

$$\text{Min } Tmax \quad (7)$$

Exemplos de problemas de escalonamento que utilizam estas funções objetivo apresentadas acima, podem ser encontrados na tabela 4.

Tabela 4 - FSJP com diferentes objetivos

Publicação	**Objetivo**
(Scrich et al., 2004)	Minimização total do *tardiness.*
(Low & Wu, 2001)	Minimização total do *tardiness.*
(Vilcot & Billaut, 2008)	Minimização do makespan e do atraso máximo (*lateness)*.

2.5 Metodologias para a resolução dos problemas de escalonamento do tipo *job-shop*

De acordo com Sha & Hsu (2006), o problema de escalonamento *job-shop* obteve destaque na literatura devido ao facto de não só estar entre os mais complexos problemas de escalonamento da produção, como também por ser um dos mais difíceis problemas de análise combinatória. O mesmo aplica-se ao FSJP, sendo este uma extensão do JSP clássico.

A flexibilidade do FJSP permite que uma operação seja processada em máquinas alternativas, isto leva a um aumenta do número de caminhos (*routing*) que um trabalho pode passar e consequentemente aumenta significativamente o número de soluções admissíveis, o que torna este problema ainda mais complexo (Birgin et al., 2014a).

Fisher & Thompson (1963) apresentaram um problema de 10 trabalhos com 10 máquinas, este problema manteve-se sem solução durante 25 anos e levou inúmeros investigadores a desenvolver diversos tipos de métodos para encontrar soluções. Atualmente, existe ainda enorme dificuldade para encontrar uma solução para problemas com 20 trabalhos e 20 máquinas num tempo considerado prático (C. Y. Zhang et al., 2008).

Para problemas da classe NP-difícil, métodos de resolução do tipo exatos podem fornecer soluções ótimas, porém demoram bastante tempo a serem computados. Por outro lado, métodos aproximados não garantem uma solução ótima mas podem fornecer soluções muito próximas desta em um curto espaço de tempo (Ghédira & Ennigrou, 2000). Ou seja, métodos de aproximação são ideais para problemas de grande dimensão devido à sua eficiência em termos de tempo.

Relativamente aos métodos de resolução para os problemas do tipo FJSP, estes podem ser distinguidos em duas principais classes: **métodos exatos** e **métodos de aproximação**. Os métodos exatos fornecem soluções, tal como o seu nome indica, exatas. Contudo, são métodos que exigem elevada capacidade de computação, isto significa que o tempo de processamento cresce exponencial de acordo com o tamanho do problema. Devido a este pormenor, são apenas aplicáveis para problemas de pequena dimensão (Kundakci & Kulak, 2016). Em contrapartida, os métodos aproximados podem fornecer soluções quase ótimas num curto espaço de tempo. Os métodos aproximados são utilizados para problemas de maior dimensão e fazem uso de métodos heurísticos, procurando assim por um conjunto de soluções admissíveis, não garantem uma solução ótima (Shen, 2002).

Posto isto, existem diversas abordagens e métodos para lidar com o problema de escalonamento em *job-shop*. Na figura 8, são apresentados os diferentes métodos de otimização em ambiente *job-shop*.

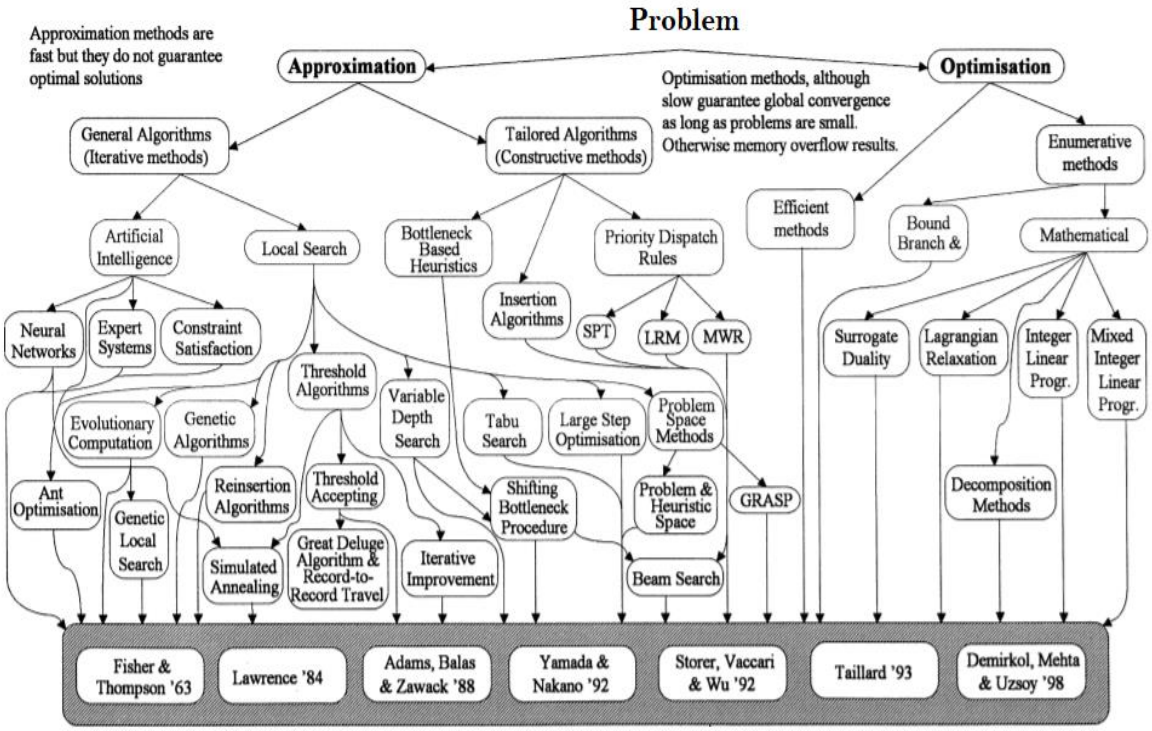


Figura 8 - Diferentes métodos de otimização em ambiente *job-shop* (Jain & Meeran, 1999)

Neste capítulo, é apresentada uma breve revisão das diferentes abordagens mais comuns para resolver o FJSP.

2.5.1 Métodos Exatos

De acordo, com Laporte & Nobert (1987) os métodos exatos podem ser divididos em duas categorias: métodos de pesquisa em árvore binaria (*branch and bound*) e programação linear inteira mista (do inglês *Mixed Integer Linear Programming*). O objetivo destes é a obtenção da solução ótima, ou seja, obter o ótimo global do problema, o que normalmente exige elevado tempo computacional que cresce exponencialmente com o tamanho do problema. Estes algoritmos atingem a solução ótima através de regras simples aplicadas sobre o conjunto que contem todas as soluções possíveis. Em suma, consistem em chegar a uma solução ótima dentro do espectro de soluções, através de uma pesquisa minuciosa ou exaustiva, este tipo de método apenas é utilizado em problemas de menores dimensões.

2.5.1.1 Algoritmos *Branch and Bound*

Os primórdios dos algoritmos de *Branch and Bound* (em português, ramificar e limitar) remontam à década de sessenta com as publicações de Brooks & White (1965), mais tarde foram submetidos a testes e avaliados em ambientes *job-shop* por Balas (1969), dando início a muitas outras investigações sobre esta temática.

De acordo com Morrison et al. (2016), a ideia principal do B&B é a decomposição de um problema em sub-problemas. Isto é, consiste na divisão do espaço de solução em dois ou mais subespaços, tal é visível na Figura 9, a estas ramificações dá-se o nome de *branching*. Cada ramificação será objeto de analise até que se obtenha a solução desse nó.

Existem também limites (*bound*) inferiores ou superiores para cada nó, que identifica a melhor solução que aquele nó pode gerar. Como se trata de problemas de *Job Shop* a função objetivo passa nomeadamente pela minimização do *makespan*, ou seja, é necessário determinar os limites inferiores em cada nó da árvore de pesquisa (Jurisch, 1995). Estes limites tornam o algoritmo mais eficiente pois permitem excluir nós da árvore de pesquisa que não foram plenamente explorados, tendo como garantia que não darão origem a soluções de maior qualidade do que as anteriormente obtidas.

Por fim, este procedimento termina quando os sub-problemas originam soluções não admissíveis ou não produzem soluções melhores do que as já determinadas, ou seja, a melhor solução encontrada é a solução ótima.

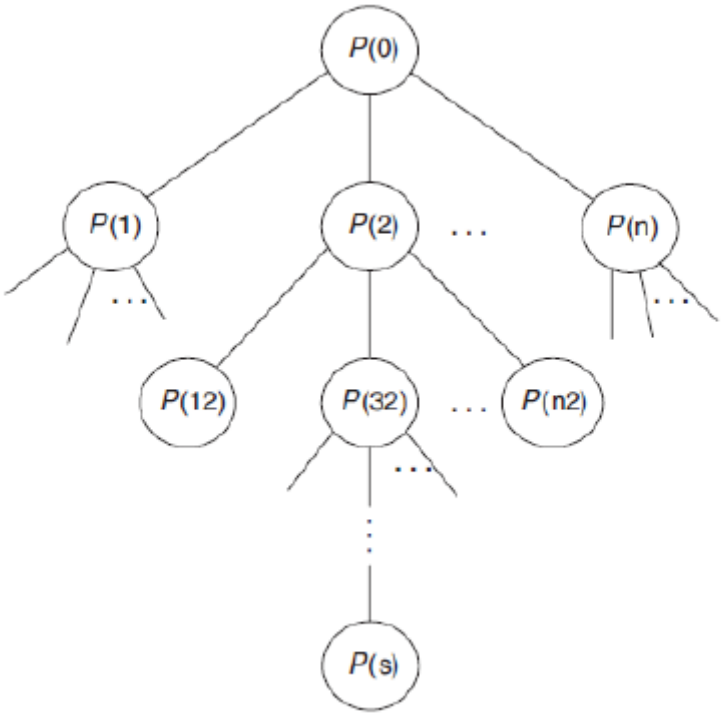


Figura 9 – Exemplo de uma árvore de *branch and bound* (Baker & Trietsch, 2009)

Recentemente, este método foi proposto por Soto et al. (2019) para resolver um problema de escalonamento de *flexible job shop* com múltiplos objetivos sendo estes a minimização do *makespan*, minimização da carga de trabalho da máquina mais carregada e também a minimização da soma das cargas de trabalho da totalidade das máquinas (carga de trabalho total). Outro exemplo, pode ser encontrado em Cheng et al. (2011), este desenvolveu um

algoritmo de *Branch and Bound* para um problema de programação em apenas numa máquina com tempos de *setup* cujo objetivo era minimizar o atraso global.

2.5.1.2 Programação Linear Inteira Mista

Embora a formulação da programação matemática não seja um método eficiente de solução devido à estrutura NP-Hard dos problemas de otimização, é um primeiro passo fundamental antes de desenvolver uma heurística eficaz e útil para compreender a estrutura do problema (Unlu & Mason, 2010).

A Programação Linear Inteira Mista (PLIM ou, em inglês MILP) é frequentemente o método padrão para solucionar problemas de escalonamento ou programação de uma indústria, uma vez que é um método flexível e poderoso para resolver problemas complexos (Blazewicz et al., 1991).

Todos os modelos de programação linear inteira mista são constituídos por três partes, nomeadamente a função objetivo que pretende maximizar ou minimizar determinado objetivo, as variáveis de decisão e por fim restrições. As variáveis de decisão podem ainda ser divididas em variáveis de decisão binárias e variáveis de decisão contínuas.
Os principais fatores que influenciam o desempenho de um modelo MILP são o número de variáveis de decisão binária ou continuas e o número de restrições (Pan, 1997). Estes fatores podem fazer com que o modelo fique demasiado "pesado" e também podem levar a diversos conflitos.

Inúmeros estudos comparativos de programação em máquinas têm surgido na literatura. Blazewicz et al. (1991) é o primeiro que se concentra em modelos matemáticos para problemas de escalonamento. Este apresentou formulações matemáticas de programação para problemas de programação *single-machine*, *parallel machine e job shop*. Uns anos mais tarde, Pan (1997) também fornece uma revisão e comparação de formulações de modelos MILP para problemas de *job shop* e *flow-shop.*

Relativamente ao campo de *job-shop,* os pioneiros e também considerados os principais trabalhos em formulações na área de *job-shop* são Manne (1960) e Wagner (1959). Após os trabalhos destes investigadores, várias outras formulações surgiam com base nestas. Mais tarde, Wilson (1989) propôs uma alternativa à formulação apresentada por Wagner (1959). Estas formulações estão presentes para comparação na tabela 5, onde é possível comparar o nº de variáveis binarias, variáveis continuas e também o nº de restrições. É de notar a semelhança do modelo Wagner e do Wilson, uma vez que um é criado a partir do outro.

Notação utilizada:

- n refere-se ao número de trabalhos para processar;
- m refere-se ao número de máquinas.

Tabela 5 - Comparação da dimensão das formulações de *job-shop* baseado em Pan (1997)

Modelo	Nº de variáveis binárias	Nº de restrições	Nº de variáveis continuas
(Manne, 1960)	$mn(n-1)/2$	$mn(n+1)/2$	$mn(n+1)/2+1$
(Wagner, 1959)	mn^2	$n^3(m-1)+3mn+m$	$2mn+1$
(Wilson, 1989)	mn^2	$n^3(m-1)+3mn$	$mn+1$

Recentemente, os modelos de programação linear inteira e mista (PLIM) para o problema *job-shop* flexível mais conhecidos da literatura e também os mais analisados são o de Özgüven et al. (2010) e do Fattahi et al. (2007), ambos têm o objetivo em comum, este é a minimização do *makespan* (C_{max}).

Fattahi et al. (2007) apresentou um dos trabalhos mais relevantes neste campo. Propôs então, um modelo de PLIM e utilizou-o para resolver um conjunto de 20 instâncias de pequenas e médias dimensões com o software LINGO, os resultados foram confrontados com os do obtidos através de métodos heurísticos. Estas instâncias tornaram-se uma referência da literatura para comparação de resultados para problemas de *job-shop* flexível.

A formulação de Özgüven et al. (2010) deriva da formulação apresentada acima de Manne (1960) e segundo o próprio investigador, obteve melhores resultados em termos de tamanho de modelo, tempos de computação e também em valores de C_{max} em relação à formulação de Fattahi et al. (2007). Da mesma forma, Demir & Kürşat İşleyen (2013) comparam as quatro formulações da literatura de FJSP mais conhecidas e chegam à conclusão que a formulação de Özgüven et al. (2010) oferece melhores desempenhos e resultados em comparação com as restantes.

Note-se que a resolução destes problemas apenas é viável para problemas de pequena escala com menor complexidade, uma vez que a maioria dos computadores existentes atualmente ainda são muito limitados a nível computacional quando se trata da resolução de problemas de maior dimensão, e podem mesmo prolongar-se por dias em bastantes ocasiões. O uso destes métodos é interessante no mundo académico, mas pode ser considerado inadequado para problemas de manufatura reais. Nestas situações, há necessidade de utilizar métodos aproximados e/ou métodos heurísticos, que permitam obter resultados de elevada qualidade e de uma maneira eficiente.

2.5.2 Métodos Aproximados: Métodos Heurísticos

Métodos heurísticos (ou aproximados) têm se destacado como uma excelente alternativa comparativamente aos restantes métodos falados neste trabalho. Heurística é uma palavra derivada do grego e que representa a arte de descobrir ou resolver problemas. De acordo com Shen (2002), uma pesquisa heurística permite-nos encontrar soluções que nos

apontam em direções interessantes (isto é, em direção à solução ótima) embora estas soluções possam ser consideradas como fracas, na medida em que podem falhar o verdadeiro ótimo global.

Um bom método heurístico pode alcançar boas soluções para problemas de grandes instâncias e computacionalmente difíceis num prazo razoável. Estes métodos tentam substituir as buscas exaustivas, reduzindo assim a dificuldade computacional. O mesmo conclui Zobolas et al. (2008), os métodos heurísticos são geralmente mais eficazes na obtenção de soluções de elevada qualidade no que diz respeito a problemas de tamanho considerável em tempos de computação razoáveis, contudo, não há garantias de encontrar a solução ótima.

Existem diferentes tipos de heurísticas (ver Figura 10), dependendo da forma como procuram e constroem soluções. Uma possível classificação divide-os em heurísticas construtivas e heurísticas de melhoramento.

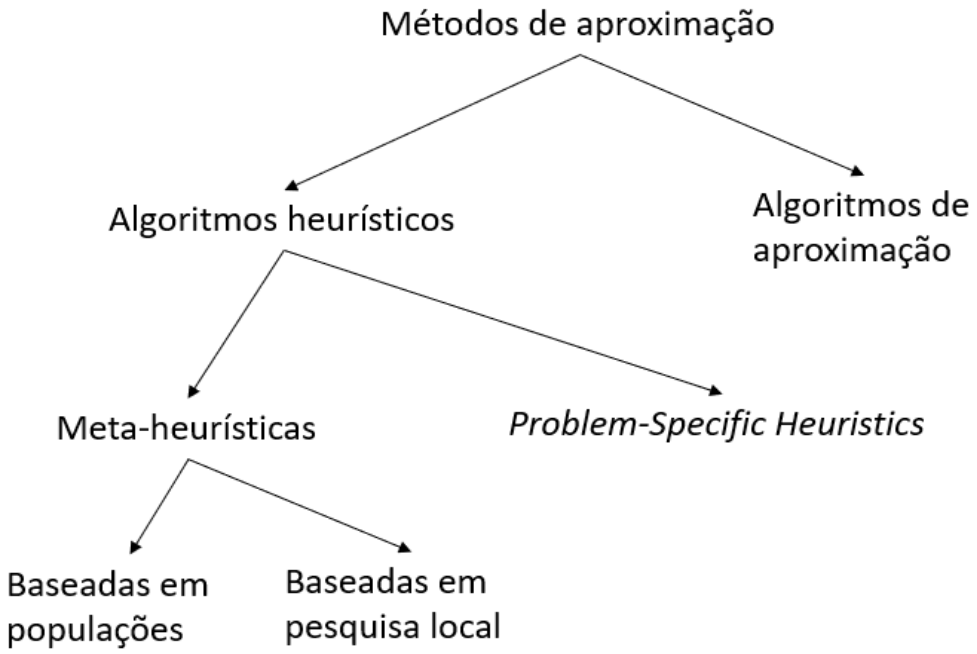


Figura 10 – Métodos de aproximação baseado em Trabelsi et al. (2010)

Heurísticas construtivas ou heurísticas gulosas

Têm como objetivo determinar uma solução inicial viável, estes métodos partem de um conjunto de solução vazio. De acordo com (Talbi & El-Ghazali, 2009), os algoritmos gulosos são populares por serem simples de desenvolver e aplicar. Além disso, os algoritmos gulosos apresentam, geralmente, uma complexidade reduzida em comparação com os algoritmos iterativos. Embora estas heurísticas possam ser rápidas na obtenção de uma solução admissível, a qualidade das soluções geralmente deixa bastante espaço para melhorias (C. Zhang et al., 2007).

As heurísticas construtivas mais utilizadas e conhecidas pelo mundo académico para problemas de otimização são as regras de prioridade.

2.5.2.1 Regras de prioridade

Regras de prioridade, também denominadas por regras de sequenciamento, são provavelmente as heurísticas mais utilizadas para resolver problemas práticos de programação em *job shop*, devido à sua facilidade de implementação e à sua baixa complexidade (Błażewicz et al., 1996a). A primeira regra de prioridade foi introduzida por Jackson (1956), mais tarde Giffler & Thompson(1960) fazem enormes contribuições nesta área, considerado por muitos como a base de todas as regras de prioridade baseadas em heurísticas.

A grande vantagem das regras de prioridade deve-se ao facto de serem, na sua maioria, de natureza intuitiva e poderem ser facilmente aplicadas no chão de fábrica. Podem ser usadas para sequenciar as encomendas/trabalhos numa sequência, de acordo com parâmetros de prioridade, por exemplo, data de entrega mais próxima ou então selecionar a seguinte operação a processar numa máquina de acordo com, por exemplo, maior tempo de processamento.

A tabela 6 contém algumas das regras de prioridade encontradas em Haupt (1989) e Blackstone et al. (1982). A primeira coluna contém a sigla e o nome da regra, enquanto que a segunda coluna contém uma breve descrição da regra.

Tabela 6 - Regras de prioridade

Regra	Descrição
SPT ("shortest processing time")	Regra que prioriza o trabalho com o menor tempo total de processamento.
LPT ("longest processing time")	Dá prioridade ao trabalho com o maior tempo de processamento total.
EDD ("earliest due date")	As tarefas são processadas de acordo com a proximidade da data de entrega.
SOT ("shortest operation time")	A operação com o tempo de processamento mais curto é processada na máquina considerada.
LOT ("longest operation time")	A operação com o tempo de processamento mais longo é processada na máquina considerada.
Aleatória / "Random"	A operação para a máquina em questão é escolhida aleatoriamente.
FIFO ("First In, First Out")	A prioridade dos trabalhos é dada de acordo com a chegada dos pedidos ao sistema.
LIFO ("Last In, First Out")	A prioridade dos trabalhos é dada ao último pedido que entra no sistema.
Rácio critico / "Critical ratio"	A prioridade de processamento dos trabalhos é dada pelo menor valor da razão crítica (RC), este valor é obtido através da seguinte formula: $$RC = \frac{(\text{data entrega}) - (\text{data atual})}{(tempo\ de\ processamento)}$$
Folga / "Slack"	A prioridade de processamento dos trabalhos é dada pela menor folga. Este valor é obtido seguinte modo: $$Folga = \frac{(\text{data de entraga}) - (\sum \text{tempos de processamento})}{(número\ de\ operações\ restantes)}$$

É de notar que a seleção da regra de prioridade a usar num sistema de produção depende dos objetivos da empresa. De uma maneira geral, estas regras são uteis quando o objetivo passa apenas por reduzir o *makespan*. No entanto, na prática os objetivos são bem mais complexos e podem passar não só pela redução do *makespan* mas também pela uma

distribuição de cargas de trabalho equivalentes quando se trata de linhas ou maquinas paralelas.

Heurísticas de melhoramento ou heurísticas de procura local

Ao contrário das heurísticas de construção, as heurísticas de melhoramento (ou também denominadas por heurísticas de procura local) partem de uma solução inicial viável e tentam melhorá-la iterativamente através da modificação da solução atual, realizando pesquisas locais para melhores soluções vizinhas.

Geralmente, as soluções iniciais são geradas de forma aleatória ou através de heurísticas construtivas, depois a heurística de melhoria executa o processo de melhoria da solução.

Para além dos métodos heurísticos, existem ainda os métodos meta-heurísticos. Estes últimos também foram abordados nesta revisão de literatura e encontram-se explícitos no Anexo A. Mais especificamente, no Anexo A são abordados três meta-heurísticas sendo estes Algoritmos Genéticos, Pesquisa Tabu e *Simulated Annealing*.

2.6 Aplicações no mundo real e *software* para a resolução de problemas de otimização

O FJSP tem aplicações relevantes no mundo real. Na literatura, apesar de todos variarem na sua forma de resolução, vários problemas realistas têm sido apresentados.

Calleja & Pastor (2014) apresentam um algoritmo baseado nas regras de prioridade para a resolução de um problema real de escalonamento em ambiente *job-shop* flexível numa instalação de fabrico de componentes para automóveis. Gomes et al. (2005) apresenta um novo modelo de programação linear inteira para escalonar um problema de FJSP, encontrado numa fábrica de produção de peças eletrónicas que opera numa perspetiva de *make-to-order*. Este autor admite ter encontrado boas soluções utilizando pacotes de *software* comerciais de resolução de problemas MILP, afirma ainda que os recentes avanços no poder computacional estão a tornar esta abordagem bastante competitiva em relação a outros métodos tradicionalmente aplicados.

Birgin et al. (2014a) propôs um novo modelo MILP para FJSP e recorreu ao *software* IBM CPLEX para resolver as instâncias baseadas nos dados reais de uma indústria tipográfica, este modelo tem como objetivo a minimização do *makespan*.

Segundo Kallrath (2013), os primeiros computadores desenvolvidos para resolver problemas de programação linear apareceram nos finais da década de 50 pela mão de engenheiros da IBM. Este mesmo autor refere que os sistemas de modelação percorreram três etapas:

1. Problemas computacionais, no passado, os limites computacionais eram a restrição dominante;
2. Devido ao aparecimento de novas linguagens de modelagem e sistemas computacionais mais poderosos, já é possível dar resposta a muitas questões que no passado não eram possíveis;

3. Atualmente e cada vez mais no futuro, aplicações reais ou novos problemas tornaram-se cada vez mais fáceis de resolver e de analisar para suportar o apoio à decisão, por parte do utilizador final.

Segundo Bartolacci et al. (2012), a área de otimização cresceu de tal maneira nas ultimas décadas que o tempo para encontrar uma solução de um determinado problema em 1984 levou cerca de dois meses e em 2004 este mesmo problema era resolvido numa questão de segundos. É possível concluir que as capacidades versáteis de modelação em conjunto com as capacidades computacionais modernas, permitem a capacidade de modelar uma grande variedade de problemas do mundo real (Kronqvist et al., 2018).

Pacotes de software são constituídos por um motor de resolução (ou, em inglês *solver*), que contêm um ou mais algoritmos, de forma a permitirem resolver certos problemas de otimização. Entre os mais conhecidos, destacam-se o CPLEX, LINDO/LINGO, XPRESS, GAMS, OptiMax 2000, entre outros (Sarker & Newton, 2007).

Arenales & Armentano (2006) destacam ainda que, métodos como o *Branch-and-Bound* ou *Branch-and-Cut* são os métodos com maior taxa de sucesso na resolução de problemas de programação inteira. Por este motivo, são adotados pelos pacotes de software comerciais mais conhecidos como o IBM CPLEX e o LINDO. De acordo com Ku & Beck (2016), embora as formulações de PLIM existam desde a década de 60, é nos últimos anos que tem ganho enorme importância devido a softwares como CPLEX, GUROBI e SCIP.

3. Caraterização da organização

Neste capítulo é apresentada uma descrição da empresa onde este trabalho foi desenvolvido. Primeiramente é feita uma breve alusão a aspetos mais gerais da empresa, como a sua história, missão, valores e políticas. Seguidamente, procede-se a uma caracterização do processo produtivo do sector onde este trabalho foi concebido, este setor é denominado de *Relwine* e, finalmente, são mostradas imagens dos produtos produzidos pela organização.

3.1 Descrição e breve história da empresa

A organização Américo Coelho Relvas, Sucessores, S.A. é uma sociedade anónima, de tipo familiar, fundada no ano de 1932. Encontra-se atualmente localizada em Rio Meão, uma vila pertencente ao Concelho de Santa Maria da Feira, a 27 Km do Porto. Conta já com muitos anos de história naquilo que diz respeito à produção de cápsulas para diversas garrafas. Contam com umas instalações que ocupam uma área coberta de cerca de 15.000 m² e empregam aproximadamente 100 colaboradores, com uma capacidade de produção de 300 milhões de cápsulas anuais.

A empresa deve a sua designação social ao nome do seu fundador, Américo Coelho Relvas, que iniciou a sua atividade na indústria da cortiça, fabricando rolhas para garrafas de espumante.

Dando-se conta do vazio que existia no mercado e face ao crescente interesse dos seus clientes de rolhas, resolveu diversificar a atividade da empresa a outros produtos que, aliás, complementavam os que já fabricava, introduzindo o fabrico de cápsulas para garrafas. Esta diversificação operou-se no ano de 1953 para o fabrico de cápsulas de chumbo estanhado utilizadas nos vinhos de qualidade.

Nas duas primeiras décadas, a totalidade da produção foi destinada ao mercado nacional, mas a partir da década de 70, A.C. Relvas alargou a sua área comercial aos mercados de exportação, e hoje pode orgulhar-se de ter um nome reconhecido nos 5 continentes como fornecedor credenciado de cápsulas, quer na qualidade oferecida pelos seus produtos, quer no serviço prestado.

Posteriormente, em 1965, foi alargado o leque dos produtos a outros tipos de cápsulas, nomeadamente para champanhe, de alumínio normal e complexo para vinhos de mesa, de P.V.C. e ainda de rosca. Estava assim praticamente coberta toda a gama de cápsulas que a indústria de vinhos e espirituosos exigia.

3.2 Missão, valores e políticas

A missão da empresa segue a filosofia do seu fundador, a A.C Relvas propõe-se continuar a abastecer o mercado vitivinícola oferecendo produtos de embalagem de qualidade e

inovadores que vão ao encontro das necessidades dos seus clientes e consolidando a sua posição de líder no mercado nacional.

A empresa assenta em quatro pilares que definem a sua forma de relacionamento com os clientes e seus parceiros de negócio:

- Tradição
- Garantia
- Qualidade
- Inovação.

Tem como objetivo o crescimento da empresa através da área da investigação, com vista à constante melhoria da sua qualidade, procurando sempre encontrar novas soluções para a organização, zelando sempre pelo ambiente.

Relativamente às políticas, assentam nos seguintes princípios da qualidade, segurança e ambiente.

3.3 Produtos

As designações atribuídas às cápsulas diferem de acordo com o material a partir do qual são fabricadas. Posto isto, é possível dividir os produtos em seis tipos:

1. Cápsulas de Rosca em Alumínio (Figura 11 A) - Formadas em alumínio, estas cápsulas destinam-se a vinhos, bebidas com elevado teor alcoólico ou mesmo ao sector da oleicultura.

2. Cápsulas de PVC (Figura 11 B) - É a cápsula mais barata oferecida pela organização, tal deve-se ao custo da matéria prima. O revestimento da cápsula é composto por uma pelicula de policloreto de vinil, já o topo é composto por PVC conferindo uma maior resistência.

3. Cápsulas de Alumínio (Figura 11 C) – tal como o nome indica, são fabricados em alumínio puro.
4. Cápsulas de Alumínio complexo (Figura 11 D) – Considerado um produto de média-gama e com imensa procura. São cápsulas formadas por duas peças, uma de alumínio simples no topo outra de alumínio complexo no corpo.

5. Cápsulas de Estanho (Figura 11 E) - são fabricadas em estanho puro. O estanho atribuiu um toque mais premium ao produto, contudo o custo é relativamente superior às restantes cápsulas. O processo de produção desta cápsula tem ainda uma singularidade, uma vez que o seu processo de fabrico produz o mínimo de sucata ou desperdício, visto que o restante estanho é reciclado.

6. Cápsulas de Espumante (Figura 11 F) – O seu fabrico recorre ao alumínio complexo ou puro. São diferenciáveis das restantes cápsulas pelas suas dimensões e efeito *sparkling*. Apresenta uma textura lisa ou granitada conforme a preferência do cliente.

Este trabalho tem origem num setor que produz as primeiras cápsulas apresentadas, ou seja, cápsulas de rosca em alumínio (Figura 11 A).

É também importante mencionar que estas cápsulas podem ser personalizadas de acordo com o pedido do cliente, ou seja, podendo ter várias personalizações o que implica processos de produção variáveis de encomenda em encomenda.

A

B

C

D

E

F

Figura 11 – Produtos: A - Cápsulas de rosca em alumínio; B -Cápsula de PVC ; C -Cápsulas em alumínio ; D - Cápsulas de alumínio complexo; E - Cápsulas de estanho; F - Cápsulas de espumante;

3.4 Processo produtivo das cápsulas de rosca em alumínio

O setor denominado por *Relwine* produz, como anteriormente referido, cápsulas de rosca em alumínio. Este foi o setor da organização que mais cresceu e, por conseguinte, mais investimento teve. O investimento neste sector destinou-se a apresentar um variado leque de opções de customização de produtos de forma a diferenciá-los dos seus concorrentes diretos, tal como o *slogan* da empresa indica "uma empresa, múltiplas soluções em capsulas". Como consequência o processo de produção poderá ser variar e ser diferente, de acordo com as exigências do cliente.

Seguidamente, apresenta-se o fluxograma do processo de produção das cápsulas *Relwine*:

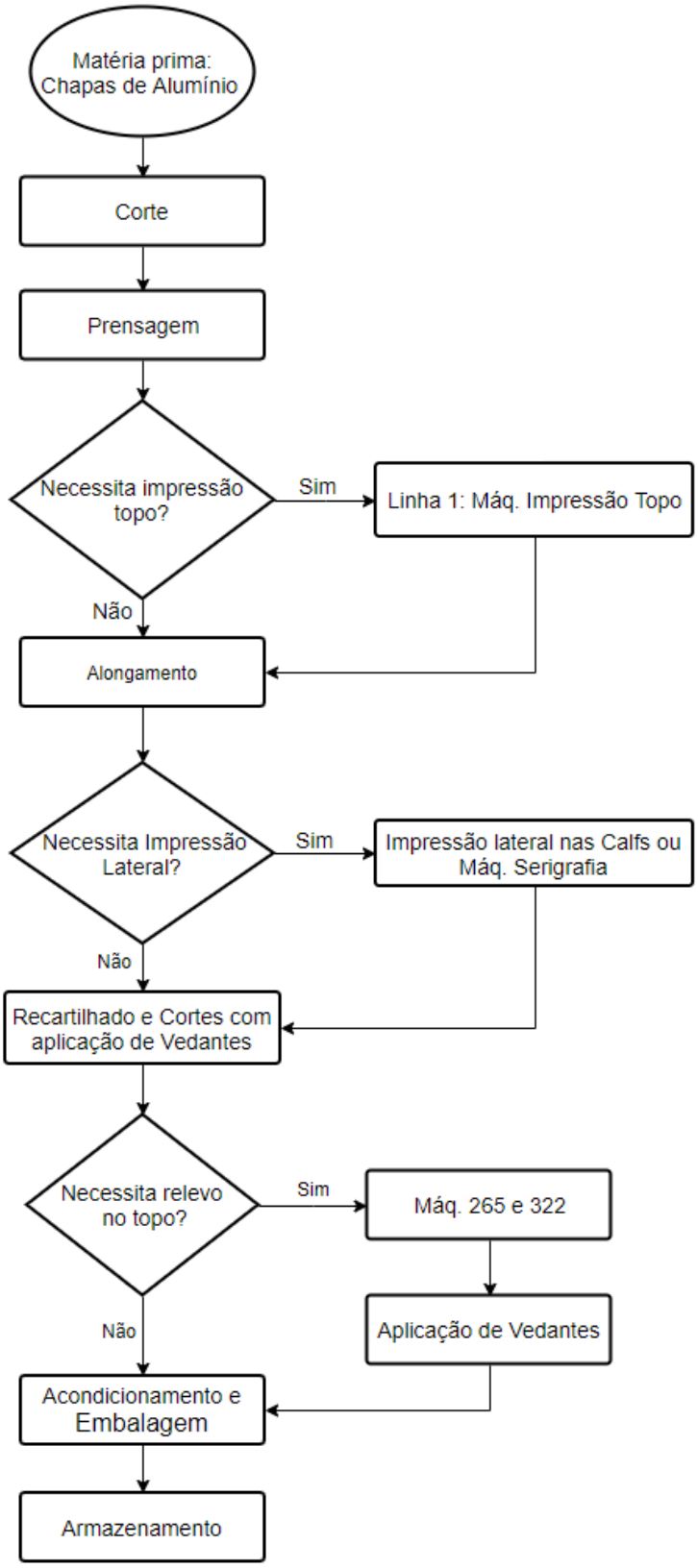


Figura 12 - Fluxograma do processo produtivo

Quando se inicia a produção de uma encomenda, é garantida a disponibilidade de toda a matéria-prima e em quantidades suficientes para que o fluxo de produção se possa realizar normalmente, sem escassez de material. A matéria prima consiste em chapas de alumínio, tintas/vernizes e discos/vedantes interiores, para além das caixas e sacas para embalamento.

O processo produtivo tem início no **corte** de uma chapa de alumínio em duas chapas de dimensões idênticas (Figura 13), realizada numa guilhotina automática. De seguida, as chapas cortadas nas dimensões ideias são transportadas para uma das três linhas possíveis, conforme a indicação que é dada ao operador. É de notar que as três linhas possuem o mesmo tipo de máquinas, sendo então possível realizar o segundo passo do processo produtivo em qualquer uma delas. As linhas são compostas por uma prensa, uma máquina de alongamento e, por fim, uma recartilhados e cortes dotada de visão artificial para detetar peças imperfeitas.

De seguida, é realizada um processo de conformação mecânica denominado por estampagem profunda ou ”repuxo”, onde a chapa é deformada numa operação de **prensagem** criando 55 peças ocas denominadas de “copola” (Figura 14 - A).

A terceira operação, consiste no **alongamento** da “copola” formada no processo anterior e tal como o nome indica, a “copola” é alongada para o tamanho pretendido da cápsula final. Esta adquire a forma final de 30 mm por 60 mm (diâmetro x cumprimento) (Fig.18 - B). Caso o cliente tenha pretendido uma pintura superior da cápsula com recurso à flexografia, este é necessário ir para uma linha especifica (Linha 1) e a pintura é realizada antes do alongamento. Contudo, é uma etapa facultativa e relativamente menos utilizada.

Após o alongamento, é possível realizar de novo etapas facultativas nomeadamente a **impressão lateral**. As cápsulas têm de ser transportadas das linhas de produção para uma das duas máquinas de impressão lateral denominadas por Calf, o transporte é feito através de palotes com capacidade aproximadamente de 8000 cápsulas. É também possível utilizar outra técnica de impressão lateral para além do uso das Calfs, esta técnica é a Serigrafia.

Por fim, segue-se a operação de **recartilhado e cortes,** na qual se efetua os cortes de abertura fácil na cápsula (Fig.18 - C), o que mais tarde vai permitir que as cápsulas sejam abertas. A **aplicação do disco ou vedante** no interior da cápsula permite selar a garrafa, sem que o conteúdo no seu interior se derrame ou entre em contacto com o exterior. Esta operação está também dotada de uma máquina de visão artificial que separa cápsulas com defeitos das cápsulas finais. De novo, existe mais uma etapa facultativa caso o cliente peça um relevo no topo da cápsula. Este é feito apos a operação de recartilhado e cortes e no final da operação de relevo, existe a necessidade da aplicação do disco ou do vedante.

Uma vez o disco ou vedante, o produto final é automaticamente expelido para caixas de cartão onde se acumulam 1.200 cápsulas por caixa. Posteriormente, as caixas serão amontoadas em paletes e **embaladas**.

Em suma, as etapas essenciais na formação de uma cápsula são:

Etapas facultativas, meramente decorativas (ver Figura 15):

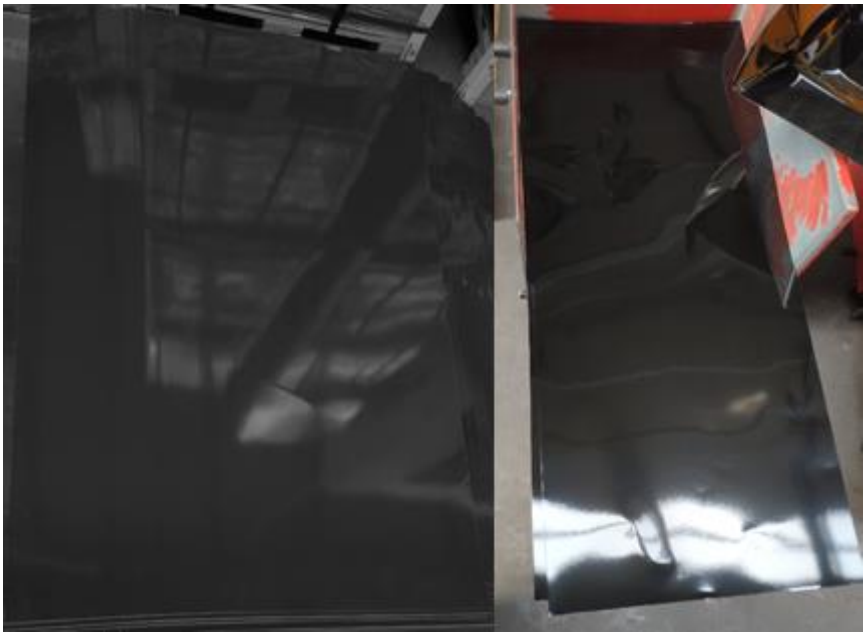


Figura 13 - Corte de Chapa

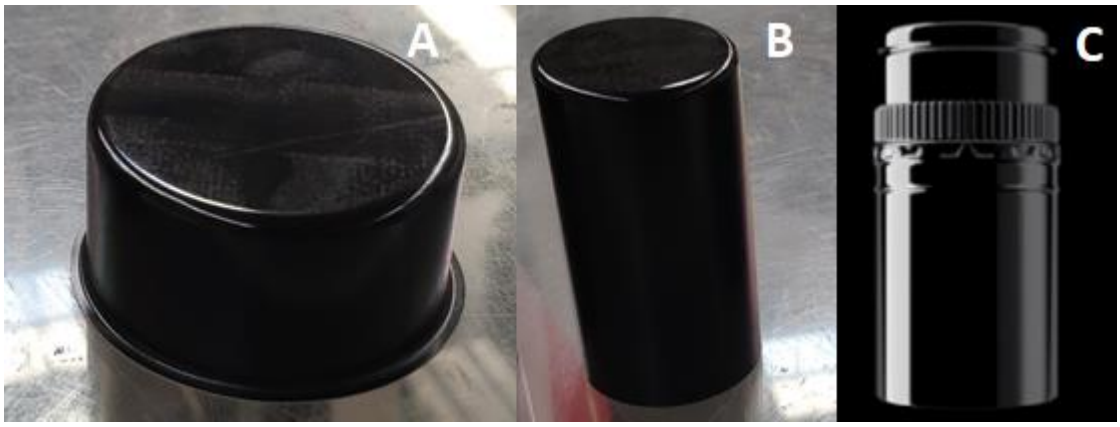


Figura 14 - Etapas do processo produtivo

Figura 15 – Etapas facultativas

4. Proposta de melhoria e resultados obtidos

O trabalho que será apresentado a seguir foi desenvolvido para melhorar o processo de planeamento produtivo e dar uma resposta superior ao escalonamento da produção para prevenir o atraso das encomendas. Posto isto, neste capítulo será apresentada a situação inicial deste setor produtivo, os eventuais problemas encontrados, as ferramentas utilizadas, os modelos desenvolvidos e, por fim, os resultados obtidos com estes.

4.1 Descrição do problema

Atualmente, as organizações industriais alocam de forma recorrente investimentos para melhorar os seus processos de produção e procuram não só a constante redução de custos, mas também a maximização da sua eficiência e competitividade.

Perante um aumento da procura, um aumento da diversidade de produtos e a introdução de novos equipamentos, é muito frequente que o sistema de Planeamento e Controlo da Produção não se mantenha a par destas evoluções e dê origem a uma redução do desempenho produtivo. Tal está a acontecer com a A.C Relvas S.A. Não há quaisquer dúvidas de que um adequado sistema de planeamento e controlo da produção seja imprescindível para o desempenho global de uma empresa industrial, sendo assim capaz de responder a questões tais como: o quê, onde e quando produzir. Assim sendo, é possível aumentar a eficácia da utilização dos equipamentos bem como das instalações de modo a garantir a satisfação dos clientes, encontrando resposta para as suas necessidades a tempo e horas.

Por outras palavras, este é o setor mais recente da organização e por conseguinte, está a ter um elevado crescimento não só no número de encomendas, mas também em investimento. Este investimento pode ser visto na introdução de novas máquinas o que leva a um aumento da dificuldade de programação. Como já foi referido, os produtos são altamente personalizáveis de acordo com o pedido do cliente e como consequência o processo produtivo é diferente para cada encomenda. Tudo isto leva a um aumento da complexidade de programação de produção, o que se traduz em certas situações num atraso de encomendas e por sua vez num desagrado do cliente. Outro facto importante é que atualmente a programação/escalonamento é feito à base da experiência por parte do responsável deste setor, ou seja, é realizada através da sua intuição e este tem várias funções para além desta. E é por isso, comum encontrar erros tais como parar o processamento de uma operação antes que esta seja concluída para processar outra encomenda que apresenta uma data de entrega mais próxima. Não quero dizer com isto que o escalonamento seja feito de forma errada, pelo contrário este apresenta resultados satisfatórios, mas penso que possa ser melhorado e existem margens de progresso.

4.2 Metodologia

Depois de identificado o problema, estabeleceu-se uma metodologia de acordo com as seguintes etapas:

- Conhecimento total do processo produtivo;
- Levantamento de dados relevantes tais como número de máquinas e seus nomes, tempos de produção destas, encomendas com etapas especiais;
- Conhecimento do sistema encomendas, desde a chegada, passagem pelo sistema informático e expedição;
- Familiarização com o *software* de otimização, sendo este o *IBM ILOG CPLEX* e com a linguagem *CP Optimizer*;
- Elaboração do modelo e documentos de apoio ao modelo;
- Obtenção e validação de soluções de planeamento semanal.

4.3 Ferramenta utilizada: IBM ILOG CPLEX - CP Optimizer

A ferramenta utilizada para dar uma resposta ao problema acima mencionado, será o IBM ILOG CPLEX. Mais especificamente, será utilizado a linguagem *CP Optimizer*.

O *CP Optimizer* fornece uma linguagem de modelação para Problemas de Optimização Combinatória que inclui a programação linear inteira e também a programação por restrições. Segundo Laborie et al. (2018), este proporciona uma linguagem com conceitos matemáticos simples para captar a dimensão temporal dos problemas de escalonamento num contexto de otimização combinatória. O *CP Optimizer* contém uma serie de restrições e funções especializadas para problemas de escalonamento reais na indústria. Foi criado tendo em consideração os seguintes requisitos:

- Ser simples, acessível e utilizar um número mínimo de conceitos de modo a reduzir a curva de aprendizagem para novos utilizadores;
- Deve efetuar uma procura automática robusta e eficiente para que o utilizador apenas se concentre no modelo sem necessidade de escrever qualquer algoritmo de pesquisa complexo, sendo então possível atribuir mais importância à modelação do problema.

A procura automática referida anteriormente, faz uso das seguintes heurísticas ou meta-heuristicas: *Large Neighborhood Search* (LNS) para produzir soluções de boa qualidade; *Failure-Directed Search* para comprovar a inviabilidade; aleatoriedade; *machine learning;* árvores de pesquisa como *Branch and Bound*; entre outros…

Relativamente ao desempenho é possível observar em Laborie (2016), que o *CP Optimizer* em comparação com modelos MIP, apresenta tempos de pesquisa de uma solução admissível ou até mesmo ótima muito inferiores. Outro fator importante de comparação é

o GAP, este foi comparado com diferentes instâncias da literatura em função de dimensões e complexidades e obteve valores próximos do 0. O mesmo não se pode dizer dos diferentes modelos de MIP onde o GAP aumentou à medida que a complexidade e dimensão do problema cresceu. Conclui-se assim que os fatores que levaram à escolha desta ferramenta se devem ao facto de esta ser relativamente acessível no contexto de aprendizagem para um novo utilizador e também porque apresenta resultados muito interessantes com instâncias de literatura.

4.4 Apresentação do problema e do modelo utilizado

Como referido anteriormente, o problema de *job-shop* flexível permite que uma operação seja processada por uma qualquer máquina a partir de um determinado grupo de máquinas. O verdadeiro desafio passa pela atribuição de cada operação a uma máquina e ordenar as operações nas máquinas de modo a minimizar o tempo máximo de conclusão de todas as operações (*makespan*).

Numa primeira fase procedeu-se ao levamento das linhas e seus equipamentos como é possível observar na Figura 16.

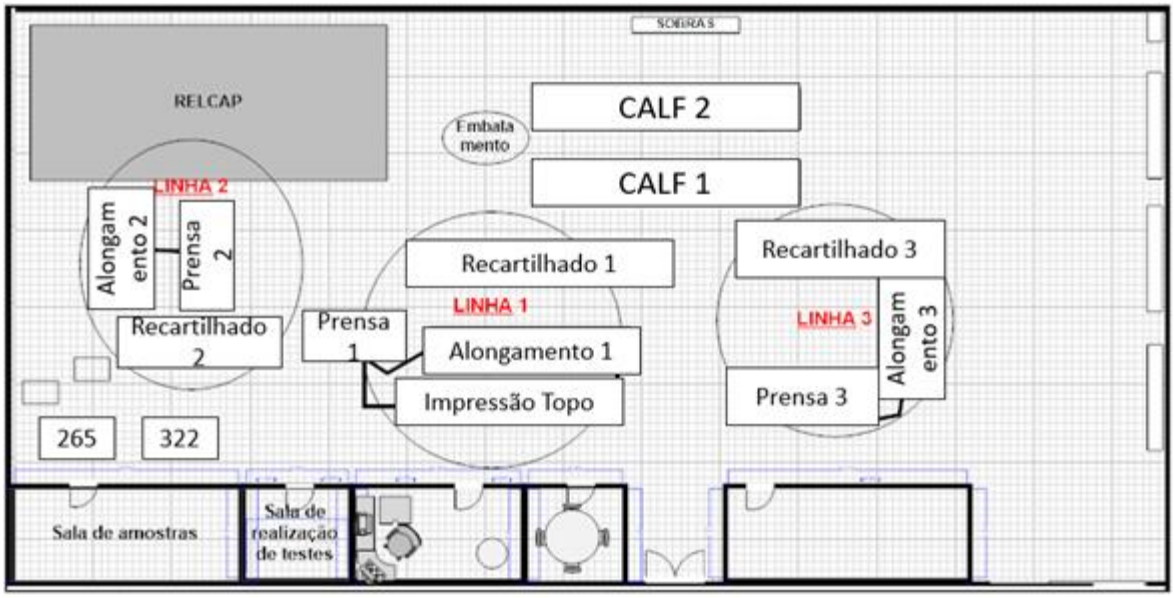
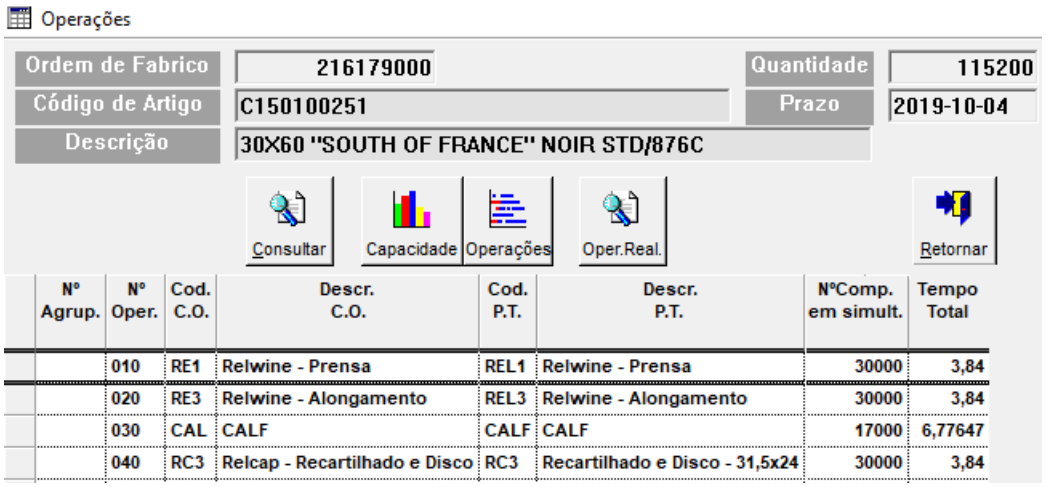


Figura 16 - Layout do Relwine

Através da Figura 16, é possivel concluir que existem três linhas de produção (Linha 1,2 e 3) das quais apresentam maquinas semelhantes, ou seja, máquinas alternativas. Todas as três linhas são compostas por uma maquina de prensagem, alongamento e por ultimo, uma máquina de recartilhado e cortes.

O próprio sistema informático da organização contém um programa que permite ter acesso às encomendas que devem ser entregues numa semana especifica, bem como é possível consultar as operações que cada encomenda deve efetuar até estar concluída (ver Figura 17). Estes dados são reaproveitados para posteriormente serem transformados em *inputs*.



Nº Agrup.	Nº Oper.	Cod. C.O.	Descr. C.O.	Cod. P.T.	Descr. P.T.	NºComp. em simult.	Tempo Total
	010	RE1	Relwine - Prensa	REL1	Relwine - Prensa	30000	3,84
	020	RE3	Relwine - Alongamento	REL3	Relwine - Alongamento	30000	3,84
	030	CAL	CALF	CALF	CALF	17000	6,77647
	040	RC3	Relcap - Recartilhado e Disco	RC3	Recartilhado e Disco - 31,5x24	30000	3,84

Figura 17 - Sistema informático (SEGIN) e exemplo de encomenda

Tendo em consideração a encomenda da Figura 17, é possível observar que contém quatro operações. A primeira e segunda operação são realizadas em simultâneo e podem ser realizadas numa das três linhas visto que estas estão equipadas com máquinas semelhantes ou alternativas. A terceira operação trata-se de uma pintura lateral e é realizada nas CALF's, nesta situação existem dois eventuais caminhos que este trabalho pode tomar, estes são CALF1 ou CALF2. Por fim, a última operação é o recartilhado e inserção de discos e tal como a primeira operação, existem três linhas possíveis em que pode ser realizada. Conclui-se desta forma que esta encomenda contêm 18 caminhos possíveis.

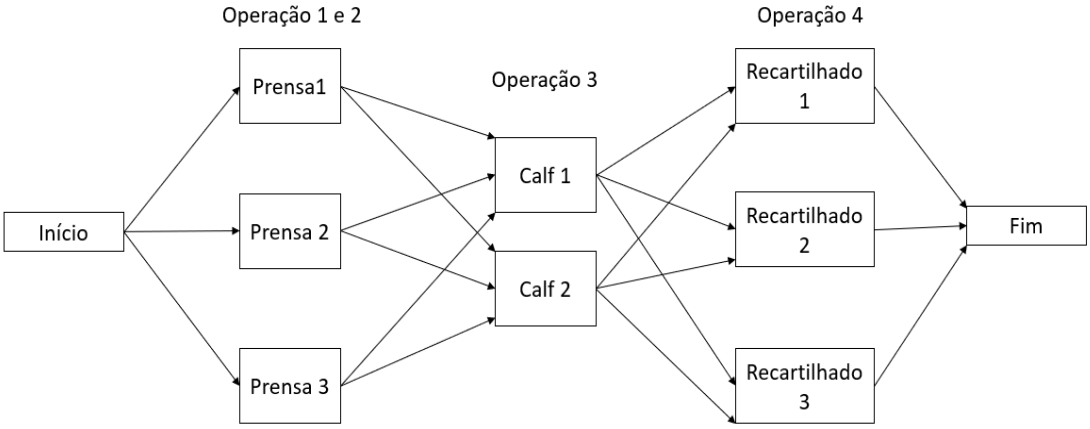


Figura 18 - Caminhos possíveis

Aqui se encontra o verdadeiro desafio de otimização no que toca à programação da produção num ambiente de *job-shop* flexível. Programar uma encomenda pode até parecer simples, mas à medida que o número de encomendas aumenta, também aumenta exponencialmente o número de soluções possíveis.

Para solucionar este problema ou neste caso, ajudar o encarregado deste setor de maneira a ter algo por onde se guiar ao escalonar as encomendas seguintes a produzir, optou-se pelo desenvolvimento um algoritmo que permitisse modelar o chão de fábrica do *Relwine*.

4.4.1 Modelo proposto

Relativamente ao modelo proposto, este é feito na linguagem de programação CP e teve de base um exemplo fornecido pelo próprio CPLEX. Este exemplo foi modificado e recebeu alterações significativas até estar totalmente adaptado à realidade do chão de fábrica do *Relwine*.

4.4.1.1 Teste do modelo inicial com instâncias da literatura

Antes de mais, convêm testar e analisar os resultados do modelo inicial antes deste ser adaptado ao contexto do chão de fábrica do *Relwine*. Para tal, serão utilizadas as instâncias mais conhecidas da literatura.

A seguinte tabela contém o tamanho das instâncias (nº de trabalhos x nº de máquinas), os melhores resultados LB (*Lower Bounds)* e UB (*Upper Bounds*) conhecidos na literatura até ao momento, como também contém as soluções obtidas através do CP.

Brandimarte (1993) forneceu 10 instâncias com flexibilidade média, contudo nem todas conseguiram ser testadas devido ao tamanho dessas instâncias e devido a limite de licença de *software*.

Tabela 7 – Instâncias de Brandimarte (1993)

Instância	**Tamanho**	**LB**	**UB**	***CP Optimizer***
MK01	10x6	36	42	40
MK02	10x6	26	32	26
MK03	15x8	204	211	204
MK04	15x8	48	81	60
MK05	15x4	168	186	173
MK06	10x10	33	86	57

Fattahi et al. (2007) desenvolveu dez instâncias de dimensão pequena conhecidas como SFJS e também desenvolveu outras dez instâncias de dimensão média conhecidas como MFJS.

Tabela 8 – Pequenas e médias instâncias de Fattahi et al. (2007)

Instância	Tamanho	LB	UB	*CP Optimizer*
SFJS01	2x2	66	66	66
SFJS02	2x2	107	107	107
SFJS03	3x2	212	221	221
SFJS04	3x2	331	355	355
SFJS05	3x2	107	119	119
SFJS06	3x2	310	320	320
SFJS07	3x5	397	397	397
SFJS08	3x4	216	253	253
SFJS09	3x3	210	210	210
SFJS10	4x5	427	516	516
MFJS01	5x6	403	468	468
MFJS02	5x7	396	446	446
MFJS03	6x7	396	465	465
MFJS04	7x7	496	554	554
MFJS05	7x7	414	514	514
MFJS06	8x7	614	635	634
MFJS07	8x7	764	879	831
MFJS08	9x8	764	884	884
MFJS09	11x8	807	1088	1070
MFJS10	12x8	999	1267	1199

Kacem et al. (2002) desenvolveu quatro instâncias para o problema de FJSP com total flexibilidade (T-FJSP) com diversos números de operações por trabalho.

Tabela 9 - Kacem et al. (2002)

Instância	Tamanho	LB	UB	*CP Optimizer*
Kacem1	4x5	11	11	11
Kacem2	10x7	11	11	11
Kacem3	10x10	7	7	7
Kacem4	10x15	11	11	11

Com a informação apresentada nas tabelas acimas, é possível concluir que este modelo serve como uma excelente base de trabalho visto que em certos casos obteve exatamente o valor igual às melhores soluções encontradas na literatura e nos restantes casos apresentou resultados bastantes próximos das melhores soluções encontradas na literatura.

De acordo com um dos criadores deste e outros exemplos em linguagem CP, Vilím et al. (2015), é possível confirmar várias melhorias de soluções conhecidas noutras instâncias da literatura mais especificamente são instâncias de Hurink et al. (1994). Contudo, apenas apresentei as instâncias de autores referidos na revisão de literatura deste relatório, como é o caso de Kacem, Fahatti e Brandimarte.

4.4.1.2 O modelo em linguagem CP e o seu funcionamento

Não só foi o modelo adaptado como foi criado um sistema que permitisse uma rápida importação e exportação de dados, de forma a facilitar o uso deste para que seja acessível por qualquer indivíduo dentro da organização.

De seguida, será explicado em que consiste o modelo. Este pode ser dividido em quarto partes: definição de variáveis, função objetivo, restrições, e por fim, pós-processamento.

Definição de variáveis:

No início do modelo estão representadas as variáveis do modelo, como por exemplo o número de encomendas a escalonar, o número de máquinas do chão de fábrica. Este número inteiro é guardado no ficheiro *.dat* ou então é importado de um *excel* específico que efetua este cálculo. O mesmo pode ser dito para as variáveis inteiras id, jobId, pos, opId, maq e tempoprocessamento. Para uma melhor compressão desta seção é necessário ter em atenção o exemplo presente em 4.4.2.

```
using CP;  //https://www.ibm.com/analytics/cplex-cp-optimizer

//Definição de Variáveis
int nEncomendas=...;
int nMaq=...;
range Encomendas = 1..nEncomendas;
range Maquinas = 1..nMaq;
```

```
tuple Operacoes {
  int id;    // ID da Operação
  int jobId; // ID do trabalho
  int pos;   // Posição no trabalho
};

{Operacoes} Ops   = ...;

tuple Alternativas {
  int opId; // ID da Operação
  int maq;  // Máquina
  int tp;   // Tempo de Processamento
};

{Alternativas} Modes = ...;

// Posição da última operação da encomenda j
int jlast[j in Encomendas] = max(o in Ops: o.jobId==j) o.pos;

//int DataEntrega[Encomendas]=...; Datas de entrega (Modelo B)

//Variaveis de decisão:
dvar interval ops  [Ops];
dvar interval modes[md in Modes] optional size md.tp;
dvar sequence maqs[m in Maquinas] in all(md in Modes: md.maq == m) modes[md];
```

Função objetivo e respetivas restrições:

Este modelo pode adotar duas funções objetivo com propósitos diferentes, sendo estas a minimização do tempo máximo de conclusão de todas as operações, por outros palavras a redução do *makespan* (C_{max}) e a minimização do atraso máximo ou também conhecido como *tardiness* (T_{max}). Mais tarde, será apresentado um estudo e comparação entre ambas as funções objetivo.

Relativamente aos conjuntos de restrições:

- O conjunto de restrição (c1) trata-se de restrições de precedências entre operações consecutivas da mesma encomenda. Por outras palavras, a operação 1 só começa quando a operação 0 esteja terminada, para tal utiliza mais uma vez uma função característica desta linguagem *endBeforeStart*.
- O conjunto de restrição c2, está relacionada com as máquinas alternativas que uma determinada operação possa ser efetuada.
- O conjunto de restrição c3, não permite que operações estejam sobrepostas numa máquina. Ou seja, uma máquina realiza apenas uma operação. É utilizada mais uma vez uma função característica desta linguagem *noOverlap.*
- O conjunto de restrição c4 é uma restrição estritamente criada para o caso do *Relwine*. Trabalhos com apenas 2 operações (operação 0 e operação 1), devem ser processados na mesma linha de produção. Por outras palavras, esta restrição está relacionada com encomendas mais simples em que apenas é realizada a prensagem e os recortes com introdução do disco. O modelo escolhe a primeira máquina para realizar a operação 0 (prensagem) contudo a operação 1, tem de ser realizada na

máquina (recartilhado) da mesma linha que a primeira. O mesmo se aplica ao caso das operações serigrafias de topo e corpo em simultâneo.

- Por fim, o conjunto de restrição c5 também é estritamente criado para o caso do *Relwine* e está relacionada com a máquina de impressão topo da linha 1 e com encomendas mais simples como o caso acima, ou seja, trata-se de um caso especial. Apenas existe uma máquina que realize impressão de topo e está presente na linha 1, trabalhos que necessitem de impressão do topo sem relevo terão obrigatoriamente de ser processados na prensa da linha 1.

```
//Função Objetivo (função objetivo do modelo A):
minimize max(j in Encomendas, o in Ops: o.pos==jlast[j]) endOf(ops[o]);

//minimize max(j in Encomendas, o in Ops: o.pos==jlast[j])  (endOf(ops[o]) -
DataEntrega[j]); (função objetivo do modelo B)

//Restrições:
subject to {
// Restrição de precedência entre operações consecutivas o1,o2 de uma encomenda
j
  forall (j in Encomendas, o1 in Ops, o2 in Ops: o1.jobId==j && o2.jobId==j &&
o2.pos==1+o1.pos)
   c1:
    endBeforeStart(ops[o1],ops[o2]);

// Máquinas alternativas para uma determinada operação o
  forall (o in Ops)
    c2:
    alternative(ops[o], all(md in Modes: md.opId==o.id) modes[md]);

// Operações na maquina m não podem sobrepor-se/overlap
  forall (m in Maquinas)
    c3:
    noOverlap(maqs[m]);

//Restrição para trabalhos pequenos (realizados de modo continuo na mesma
linha)
   forall(md1 in Modes, md2 in Modes: md2.opId==1+md1.opId &&
md2.mch==5+md1.maq)
            c4:
      presenceOf(modes[md1])==presenceOf(modes[md2]);



// Restrição para máquina do Topo (Linha 1) e pequenos trabalhos
forall (j in Encomendas, o1 in Ops, o2 in Ops: o1.jobId==j && o2.jobId==j &&
o2.pos==o1.pos)
   c5:
    startAtStart(ops[o1],ops[o2]);
}
```

Definição do pós-processamento:

Após o modelo apresentar resultados estes serão apresentados da seguinte forma: operação; máquina onde a operação foi realizada; o início e fim. Uma vez que permite uma

leitura mais facilitada dos resultados, permite também exportar e transformar estes gráficos num gráfico de Gantt como de seguida será apresentado.

```
tuple solutionT{   //Transformar a sequencia em tuplo (para exportar para
excel)
       int operacao;
       int maquina;
       int inicio;
       int fim;
};
{solutionT} solution = {<m.opId, m.maq, startOf(modes[m]), endOf(modes[m])> | m
in Modes : endOf(modes[m]) != 0};


//Mensagem escrita no scripting log
execute {
  for (var m in Modes) {
    if (modes[m].present)
      writeln("A Maquina " + m.maq + " realiza a Operacao " + m.opId + " no
instante " +modes[m].start + " e acaba no instante " +modes[m].end);
  }
}
```

4.4.2 Leitura de dados/instâncias e explicação do funcionamento do modelo

De seguida, será apresentado um simples exemplo de três encomendas (Enc.1, Enc.2 e Enc.3) de modo a que se perceba como é que os *inputs* e *outputs* do algoritmo funcionam.

Cada operação tem um *id* único que é usado posteriormente para identificar as máquinas alternativas e será incrementado a cada posição. Os trabalhos ou encomendas, também têm um número de identificado denominado de *jobId*. Cada operação contém uma posição (*pos*), as posições parecem ser consideradas a partir de 0. Relativamente às alternativas, contêm também três campos. O campo *opId* será usado para identificar a operação especifica que contêm exatamente o mesmo número e essa operação poderá ter várias máquinas (*maq*) alternativas com variados tempos de processamentos (*tp*).

Relativamente à encomenda 1, é fácil entender que é identificada por *jobId=1* e contém três posições (0,1,2), ou seja, contém 3 operações. A posição 0 que tem o id=1 apresenta três máquinas alternativas (como já referido anteriormente são as três prensas) denominadas de 1,2,3 (no Anexo C estão especificados o número e nome dado às máquinas utilizadas no sistema) com o respetivo tempo de processamento de 5 horas. Já a operação id=2 e posição 1 tem apenas duas máquinas alternativas cujo tempo de processamento é de 9 horas. Por fim, o recartilhado e disco que tem como id=3 e posição 2, apresenta também três máquinas alternativas.

Tabela 10 - Encomenda 1 (exemplo)

Encomenda 1 - Operações:	**Tempo de processamento (horas)**	**Operações**			**Alternativas**		
Prensa e Alongamento (0)	5	**id**	**jobId**	**pos**	**opId**	**maq**	**tp**
		1	1	0	1	1	5
		2	1	1	1	2	5
CALF - Decoração lateral (1)	8,8	3	1	2	1	3	5
					2	4	9
					2	5	9
					3	6	5
Recartilhado e Disco (2)	5				3	7	5
					3	8	5

Tabela 11 - Encomenda 2 (exemplo)

Encomenda 2 - Operações:	**Tempo de processamento (horas)**	**Operações**			**Alternativas**		
Prensa e Alongamento (0)	1,1	**id**	**jobId**	**pos**	**opId**	**maq**	**tp**
		4	2	0	4	1	2
		5	2	1	4	2	2
Serigrafia de Corpo (1)	9,6	6	2	2	4	3	2
					5	11	10
					5	12	10
Recartilhado e Disco (2)	1,1				6	6	2
					6	7	2
					6	8	2

A encomenda 3 é um caso especial que já foi referido acima, trabalhos com apenas 2 operações, devem ser processados na mesma linha de produção tal como é feito no chão de fábrica. Devido a isto, foi criado certas restrições que permitem modelar tal e qual é efetuado na realidade. Ambas apresentam a posição 0 pois são efetuadas em simultâneo.

Tabela 12 - Encomenda 3 (exemplo)

Encomenda 3 - Operações:	Tempo de processamento (horas)	Operações			Alternativas		
Prensa e Alongamento (0)	1,4	**id**	**jobId**	**pos**	**opId**	**maq**	**tp**
		7	3	0	7	1	2
		8	3	0	7	2	2
Recartilhado e Disco (0)	1,4				7	3	2
					8	6	2
					8	7	2
					8	8	2

Para tal, foi criado um ficheiro de Microsoft Excel que consegue transformar os dados que se encontram no formato de texto como na Figura 17, neste tipo de instâncias. Este ficheiro permite ao utilizador apenas selecionar as encomendas pretendidas e no final colar estas no próprio ficheiro *.dat* sem qualquer tipo de problemas ou percas de tempo.

Nota: Em todos os tempos de processamento é realizado um arredondamento por excesso. O algoritmo consome apenas ler números inteiros.

Transformação e exportação dos dados

Após o algoritmo correr, os resultados ou *outputs* são apresentados da seguinte forma:

```
// solution with objective 19
A Maquina 3 realiza a Operacao 1 no instante 0 e acaba no instante 5
A Maquina 5 realiza a Operacao 2 no instante 5 e acaba no instante 14
A Maquina 8 realiza a Operacao 3 no instante 14 e acaba no instante 19
A Maquina 2 realiza a Operacao 4 no instante 0 e acaba no instante 2
A Maquina 12 realiza a Operacao 5 no instante 2 e acaba no instante 12
A Maquina 8 realiza a Operacao 6 no instante 12 e acaba no instante 14
A Maquina 1 realiza a Operacao 7 no instante 0 e acaba no instante 2
A Maquina 6 realiza a Operacao 8 no instante 0 e acaba no instante 2
```

Figura 19 - Resultados obtidos - *Scripting log*

A Figura acima, demonstra a maneira de como os dados da solução são apresentados e esta solução tem conta com um objetivo de 19. Como é obvio os dados representados desta forma são muito pouco úteis.

Para tal, foi necessário desenvolver outra ferramenta de apoio para conseguir uma fácil leitura destes resultados. Apesar do IBM ILOG CPLEX apresentar um sistema de soluções com recurso a gráficos de *Gantt*, este provou ser demasiado ineficiente. Pois, continha diversos erros, falta de organização e também é bastante difícil de ler devido à falta de cores.

A ferramenta foi desenvolvida com ajuda do *Google Charts*. Os resultados são exportados para um *Excel* (ver Figura 20) e com o recurso a uma macro estes dados são num gráfico de *Gantt*.

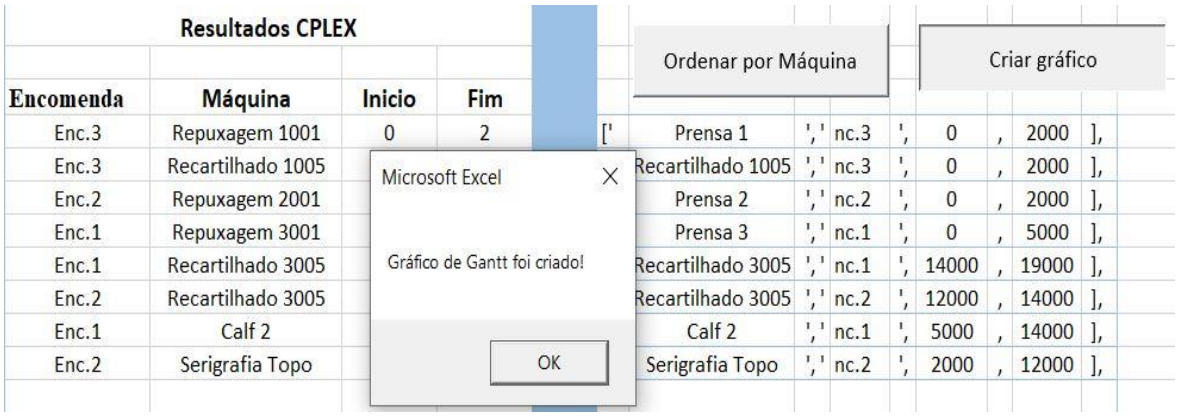


Figura 20 - Exportação de dados Excel e Criação do Gráfico de Gantt

Após a transformação os dados serão apresentados conforme se segue:

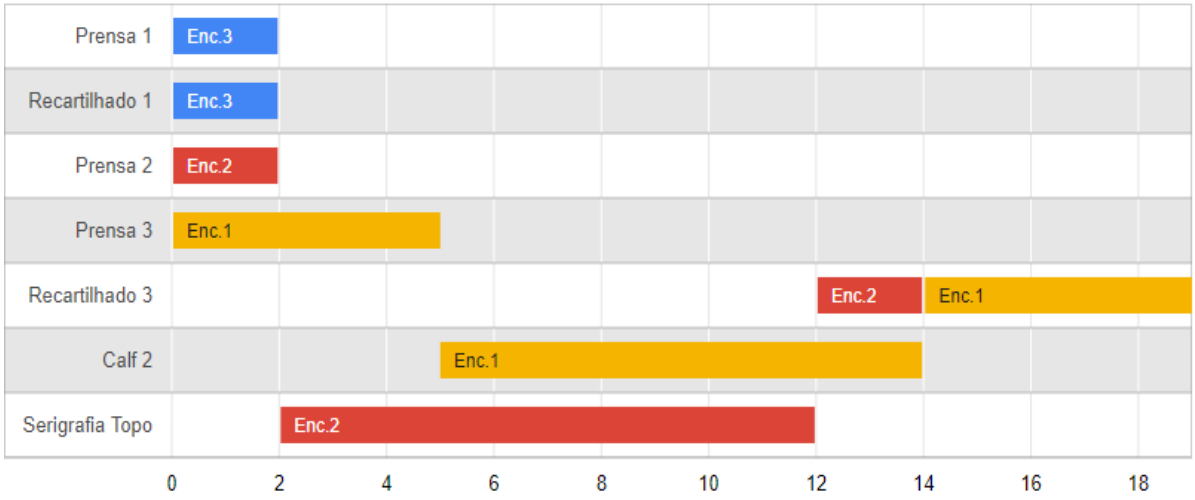


Figura 21 - Resultados representados num gráfico de Gantt

Através deste pequeno exemplo é possível concluir que este escalonamento apresenta um *makespan* de 19 horas, a última operação a ser executada tem fim na máquina "Recartilhado 3". Tal como foi referido a Encomenda 3 é um caso especial e que tem apenas duas operações obrigatoriamente executadas na mesma linha, neste caso linha 1, em simultâneo.

4.5 Análise da função objetivo

Nesta secção será apresentado numa primeira fase os resultados e análise das soluções de dois escalonamentos que apresentam as mesmas instâncias, contudo o primeiro escalonamento é executado num modelo cuja função objetivo consiste na minimização do *makespan* (C_{max}) presente no capítulo 2.4 (2) deste trabalho. Por outro lado, o segundo modelo tem em consideração datas de entrega, ou seja, janelas temporais e como tal, a função objetivo consiste em minimizar a soma do atraso máximo das encomendas (em inglês: *tardiness*) presente no capítulo 2.4 (7) deste trabalho. Numa segunda fase, as conclusões obtidas são utilizadas para determinar que tipo de função objetivo é melhor para a organização em questão e é apresentada uma comparação do escalonamento que realmente foi aplicado vs. o escalonamento através deste modelo.

O computador utilizado para efetuar as experiências possui um processador *Intel Core i5* de 8ª geração, memória RAM de 24 GB e sistema operativo *Windows* 10. O modelo anteriormente apresentado foi executado no *software IBM ILOG CPLEX Optimization Studio 12.9.* Este *software* permitirá obter uma solução ótima para este problema. As instâncias traduzem-se nos inputs do problema e neste caso, representam as encomendas reais de uma semana que necessitam ser entregues.

É de notar que as instâncias são reais e tratam de 20 encomendas para entregar na semana de 03/02/2020 a 07/02/2020 (Anexo D) e podem ser consultadas no Anexo E.

Em termos de complexidade de dimensão de um modelo, estes podem ser comparados em função do número variáveis e do número de restrições. Relativamente à complexidade computacional, os modelos podem ser avaliados com base, no número de soluções óptimas, no tempo de CPU e no GAP. Quanto menor for o tempo de CPU, melhor o modelo. O GAP é usado para avaliar se a solução obtida é ou não ótima, este também serve para avaliar soluções de diferentes modelos. Quanto menor for o valor do GAP, melhor será a solução, uma solução que apresenta GAP = 0 uma solução ótima.

4.5.1 Escalonamento A - na minimização do *makespan* (C_{max})

Como já foi referido, o escalonamento A trata do modelo apresentado no ponto 4.4.1.

Os resultados obtidos estão apresentados na tabela seguinte:

Tabela 13 - Resultados escalonamento A

Função Objetivo	Nº de variáveis	Nº de restrições	Tempo de execução (seg)	GAP
45	244	214	2.41seg	0%

Para uma análise da solução, o gráfico de *Gantt* deste escalonamento encontra-se na Figura 22.

4.5.2 Escalonamento B - na minimização do atraso máximo (*tardiness* - T_{max})

Relativamente ao modelo do escalonamento B, foi necessário fazer umas pequenas alterações face ao modelo apresenta no ponto 4.4.1.

Estas alterações passam pela adição de variáveis inteiras como as datas de entregas. Estas serão mais tarde introduzidas no ficheiro .dat.

```
int DataEntrega[Encomendas]=...;
```

Também foi necessário introduzir um limite de paragem do CP Optimizer. Caso contrário, o programa só parava de efetuar uma pesquisa quando fosse encontrado uma solução ótima e provado que é ótima. Assim, o motor de pesquisa irá parar antes de provar a otimalidade e reportar a melhor solução encontrada até agora. Neste caso, o CP irá parar após atingir um determinado número de falhas.

```
execute { //limite no número de falhas durante a procura (realizado ao CP)
            cp.param.FailLimit = 5000000;
}
```

Por último, a função objetivo passou a ser a seguinte:

```
minimize sum(j in Encomendas, o in Ops: o.pos==jlast[j])  maxl(0,(endOf(ops[o])
- DataEntrega[j]));
```

É de notar a semelhança desta função objetivo relativamente à função objetivo encontrada na revisão de literatura presenta no capítulo 2.4 (7) da revisão de literatura. De acordo, com o Anexo D as datas de entregas são introduzidas da seguinte maneira:

DataEntrega = [12, 12, 24, 32, 32, 24, 24 ,32, 32, 32, 44, 44, 44, 32, 56, 56, 56, 24, 24, 12];

Um dia representa 12h de laboração no setor *Relwine*, ou seja, as encomendas para o dia o dia 03/02 terão uma janela de entrega [0,12], as do dia 04/02 terão uma janela [0,24] e por aí adiante.

Os resultados obtidos estão apresentados na tabela seguinte:

Tabela 14 - Resultados escalonamento B

Função Objetivo	Nº de variáveis	Nº de restrições	Tempo de execução (seg)	GAP
1191	244	214	39.91seg	81%

Este escalonamento encontra-se representando na forma de gráfico de *Gantt* na Figura 23.

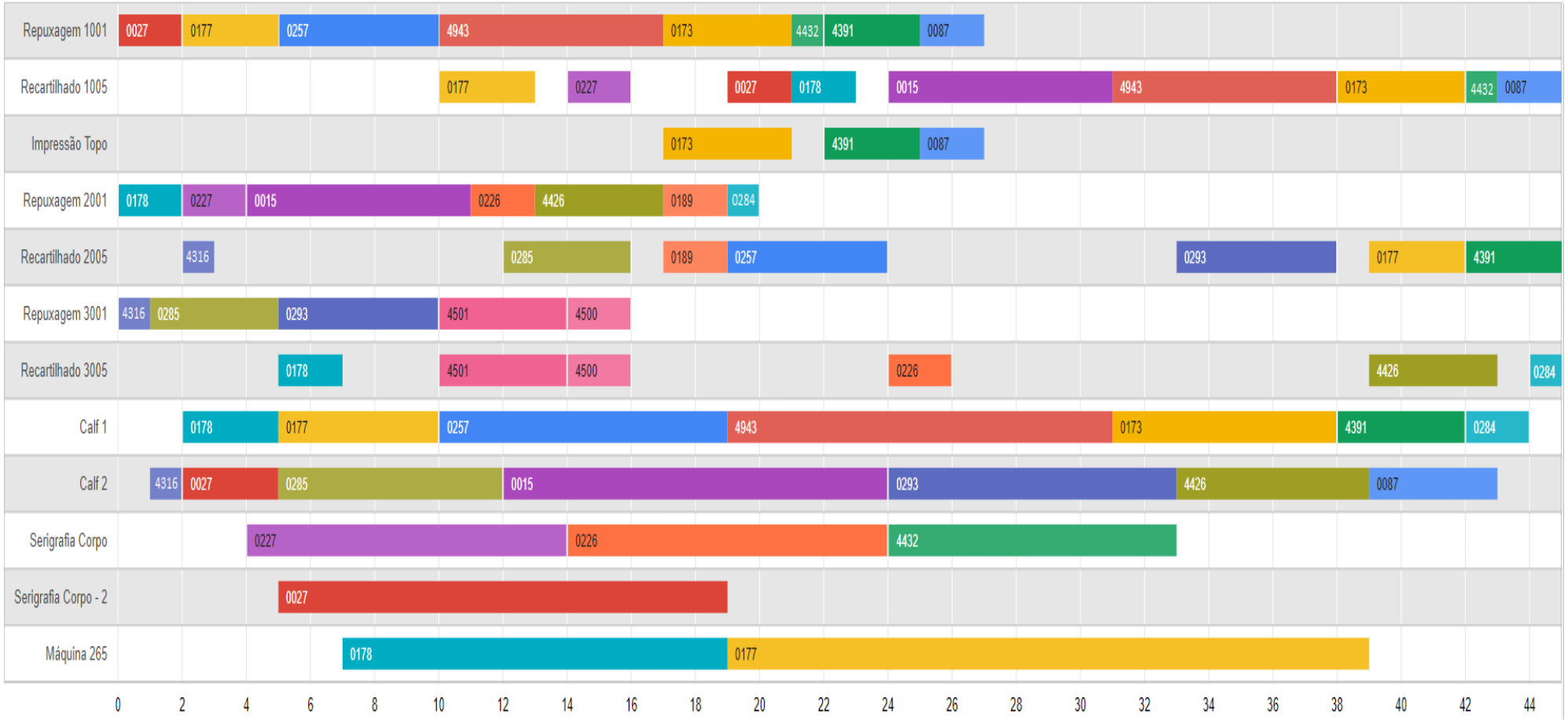


Figura 22 - Escalonamento A

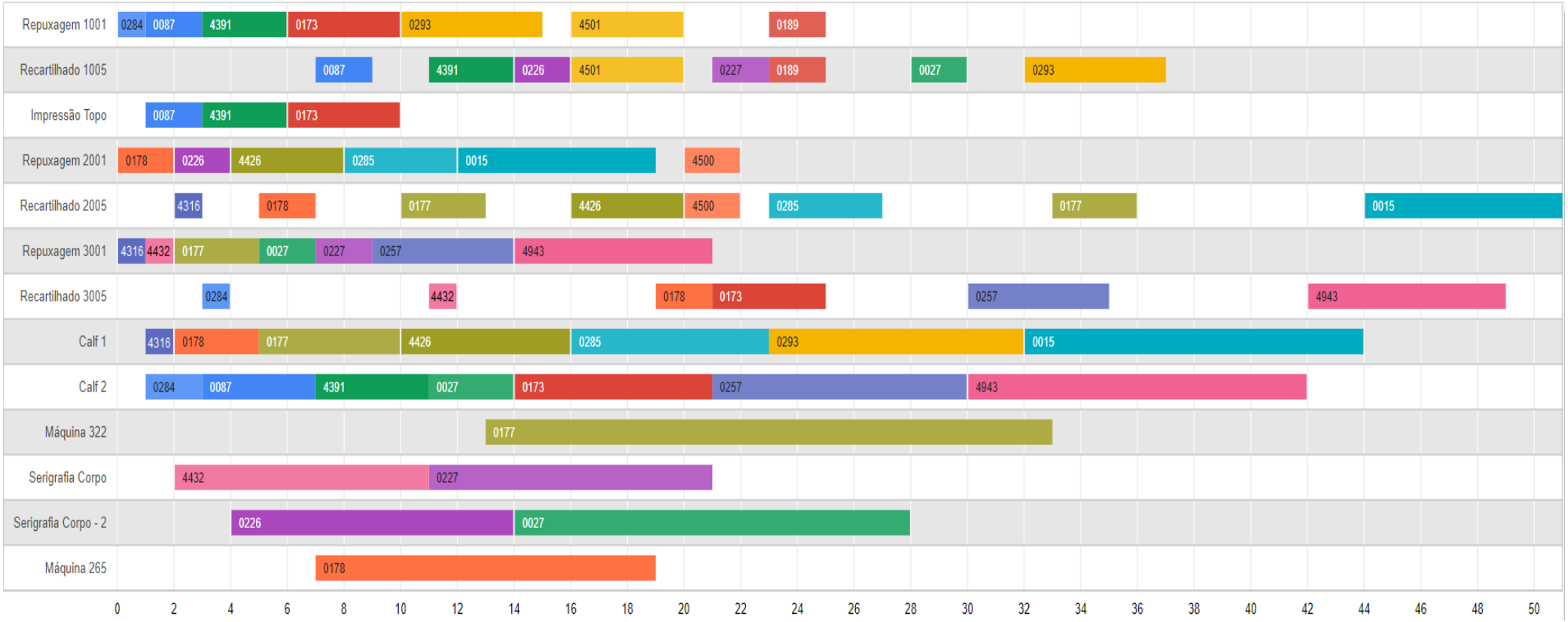


Figura 23 - Escalonamento B

4.5.3 Análise da função objetivo e discussão de resultados

De um lado, está presente um escalonamento com o objetivo de reduzir o *makespan* e do outro tem-se um escalonamento com o objetivo de reduzir o número de encomendas em atraso.

Tendo em consideração ambos gráficos de Gantt, é possível perceber que o escalonamento A tal como esperado, consegue "despachar" as encomendas num curto espaço de tempo em comparação com escalonamento B. No escalonamento A, a última operação termina no instante 45 e é esse o valor da função objetivo deste escalonamento. Já no escalonamento B, a última operação termina no instante 51. Em termos práticos, para as mesmas encomendas o último modelo demorou mais de 6 horas até concluir todas as encomendas.

Contudo, o objetivo do escalonamento B é reduzir o número de encomendas em atraso e este cumpriu com este objetivo como é possível analisar de seguida.

Na primeira coluna encontra-se o nome das encomendas, na segunda a sua duração de fabrico prevista (em horas), na terceira coluna encontram-se as janelas temporais das datas de entrega e nas restantes colunas encontram-se a hora de conclusão da encomenda no determinado escalonamento e por último, as encomendas em atraso estão sombreadas.

Tabela 15 - Encomendas em atraso

Encomenda	Duração prevista (h)	Janela temporal: Data de entrega	Escalonamento A: Conclusão (h)	Escalonamento B: Conclusão (h)
0257	20h	[0,12]	24	35
0087	8h	[0,12]	45	9
0015	26h	[0,24]	31	51
0027	21h	[0,32]	21	30
4943	26h	[0,32]	38	49
0284	4h	[0,24]	44	4
0226	12h	[0,24]	26	16
0227	12h	[0,32]	16	23
0173	15h	[0,32]	42	25
0189	2h	[0,32]	19	25
0177	32h	[0,44]	42	36
0285	16h	[0,44]	16	27
0177	20h	[0,44]	23	21
0178	19h	[0,32]	38	37
0283	4h	[0,56]	14	20
4501	2h	[0,56]	43	22
4426	13h	[0,56]	43	20
4391	9h	[0,24]	45	14
4432	11h	[0,24]	43	12
4316	2h	[0,12]	3	3

Na tabela acima, são apresentadas as 20 encomendas a serem entregues na semana 03/02 a 07/02, supondo que ambos os escalonamentos foram efetuados no dia 03/02 pelas 8h da manhã. É possível concluir que o escalonamento A, apresenta 45% das encomendas em atraso. Por outro lado, o escalonamento B, apresenta apenas 20% de encomendas em atraso e uma delas, neste caso a encomenda 0293, apenas conta com um atraso de 5 horas. Muito provavelmente se permitisse o algoritmo correr mais tempo, ou seja, se aumentasse

o parâmetro de paragem (limite no número de falhas durante a procura), o algoritmo demoraria mais tempo até encontrar uma solução mas essa solução na teoria seria superior à solução apresentada e neste caso é possível prever que a encomenda 0293 pode até se encontrar dentro da janela temporal.

Nota: A encomenda 0257 e 0015 obviamente encontram-se atrasadas pois, a sua duração de fabrico prevista é superior às datas de entrega. Ou seja, a encomenda 0257 deveria ser entregue na segunda-feira (03/02/2020) contudo o seu tempo de fabrico é superior a um dia de trabalho (12h), logo é tecnicamente impossível entregar esta encomenda dentro deste prazo. O que pode ser considerado como uma crítica de fazer um escalonamento semanal, um escalonamento para este tipo de empresa deveria ser feito quinzenalmente.

Relativamente à complexidade o modelo A é de facto, muito superior. Este demorou apenas cerca de 3 segundos até encontrar uma solução ótima, já o modelo B foi necessário implementar um parâmetro de paragem para que este reportasse a melhor solução até então encontrada.

No que diz respeito ao GAP, este representa entre a melhor solução inteira e a melhor solução inteira atingida e pode ser calculado da seguinte maneira:

$$GAP\ (\%) = \frac{\text{Upper Bound} - \text{Lower Bound}}{\text{Lower Bound}} \times 100$$

O modelo A conseguiu encontrar uma solução ótima daí o seu GAP ser de 0%, o que é um valor excelente no que diz respeito à dificuldade destes problemas. Por outro lado, o GAP da otimalidade do escalonamento B é de 81%, prova que está longe do valor da solução ótima. Contudo, tal deve-se à natureza da introdução de datas de entrega, na literatura modelos em CP que apresentam datas de entrega, normalmente apresentam um GAP superior a 50%. Tanto a função objetivo como o GAP poderiam ter apresentado melhores resultados se parâmetro de paragem fosse alargado e por exemplo, este modelo corre-se durante vários minutos ou, até mesmo, horas. Contudo, o objetivo desta dissertação passa por de uma forma geral arranjar uma forma rápida e eficiente de efetuar um planeamento diversas encomendas. E neste sentido, tal foi cumprido pois com recurso a estas ferramentas é possível efetuar um escalonamento semanal em menos de 30 minutos.

Conclui-se então, que apesar do primeiro modelo apresentar resultados computacionais significativamente melhores, o segundo destaca-se no que toca a entregar encomendas dentro do prazo definido. Posto isto, acredito seriamente que o modelo B cujo função objetivo é a minimização do *tardiness* seja mais relevante no mundo industrial e também o modelo ideal para ser utilizado por esta organização.

4.6 Comparação do escalonamento real vs. escalonamento do algoritmo

Até chegar ao modelo final apresentado neste trabalho foram feitas bastantes comparações com a realidade e melhorias deste modelo. A seguinte semana que servirá de termos de comparação ocorreu no mês de fevereiro durante os dias 10 a 14. Para tal, os dados foram recolhidos através de uma análise presencial e também se encontram presentes no *software* da organização. Contudo, estes por vezes são mal introduzidos no sistema pelos operadores e podem conter alguma lacuna.

Na Figura 24 está presente o escalonamento real do setor que foi efetuado no dia 10 (segunda-feira) e, por sua vez, toda a semana foi compilada na totalidade na Figura 27 sendo assim possível comparar com o escalonamento teórico obtido através do *CPLEX*.

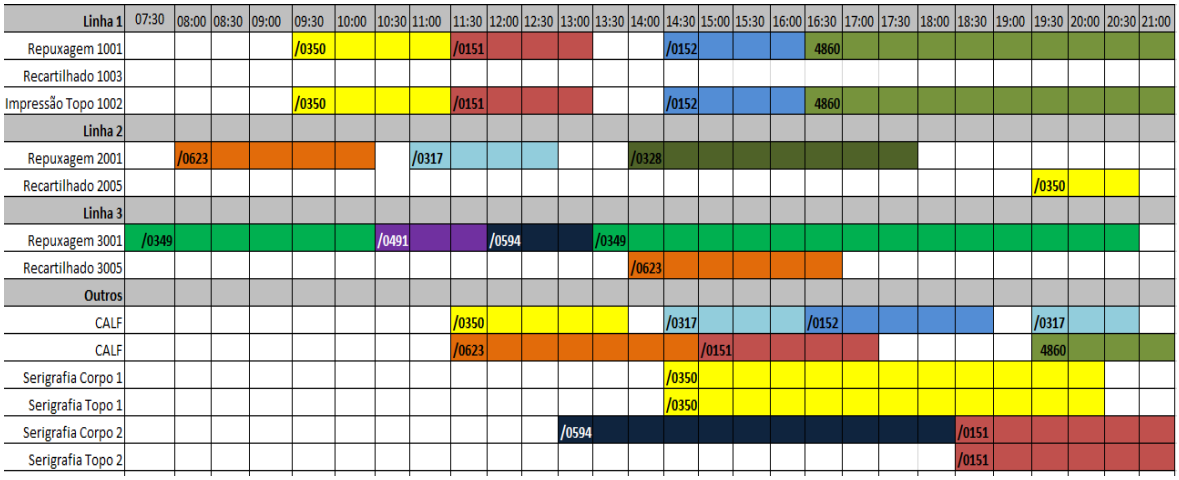


Figura 24 - Escalonamento real do dia 10

Apenas num dia é possível observar duas diferenças relativamente ao escalonamento teórico efetuado pelo programa. A principal diferença deve-se ao facto de em ambiente real em certas ocasiões uma operação de um trabalho pode dar início sem que a operação antecedente esteja totalmente acabada. Por outras palavras, caso uma operação esteja a decorrer e já exista um palote cápsulas (cerca de 8000 cápsulas) preparadas para a próxima fase/operação, não é necessário esperar pelo processamento total dessa operação e podem seguir para a próxima operação. Tal é possível observar na figura seguinte.

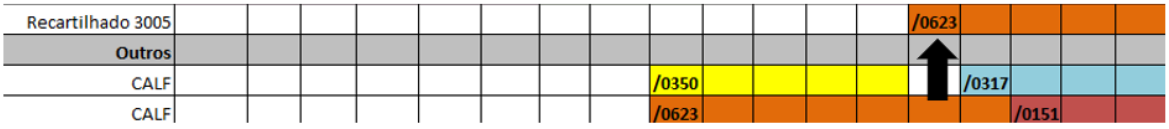


Figura 25 - Relaxação em ambientes reais

Ou seja, em ambiente real caso exista uma máquina disponível não é necessário esperar pela conclusão total da operação antecedente. Visto que já existem cápsulas disponíveis para processar na próxima operação. Esta relaxação não existe no modelo atual

apresentado, uma operação só dá início quando a operação antecedente desse trabalho esteja completamente terminada.

A segunda diferença é que uma operação pode ser interrompida para que uma determinada máquina fique disponível para outra operação de outra encomenda. Este caso verifica-se principalmente quando a pessoa responsável pelo sector percebe que uma encomenda vai demorar mais tempo que o esperado e é necessário processar uma encomenda mais prioritária. Como é obvio o programa não liberta uma máquina enquanto que essa operação não fique totalmente concluída. A interrupção pode ser observada na Figura 26.

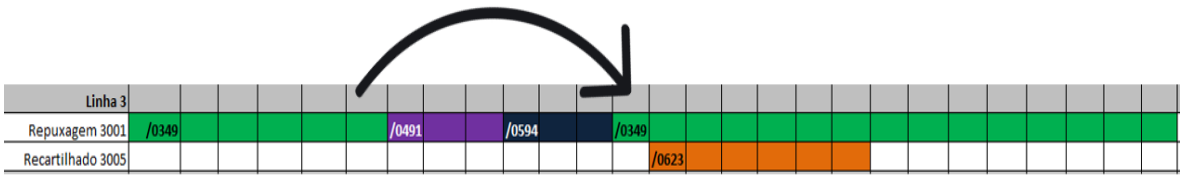


Figura 26 - Interrupção e continuação de uma operação após serem processadas outras operações

Certos desperdícios ocorrem devido a estas interrupções e são bastante normais de as ver no chão de fábrica. Isto gera diversos palotes (cerca de 8000 cápsulas) que serão acumulados no centro do chão de fábrica sem qualquer necessidade e mais tarde terão de ser movimentados (desperdícios de transportação). Vai então contra vários princípios da filosofia *Lean*.

Na Figura 27, encontra-se o escalonamento real da semana 10 a 14 de fevereiro tal como foi efetuado no chão de fábrica, para conseguir uma comparação realística e fiável relativamente à Figura 28.

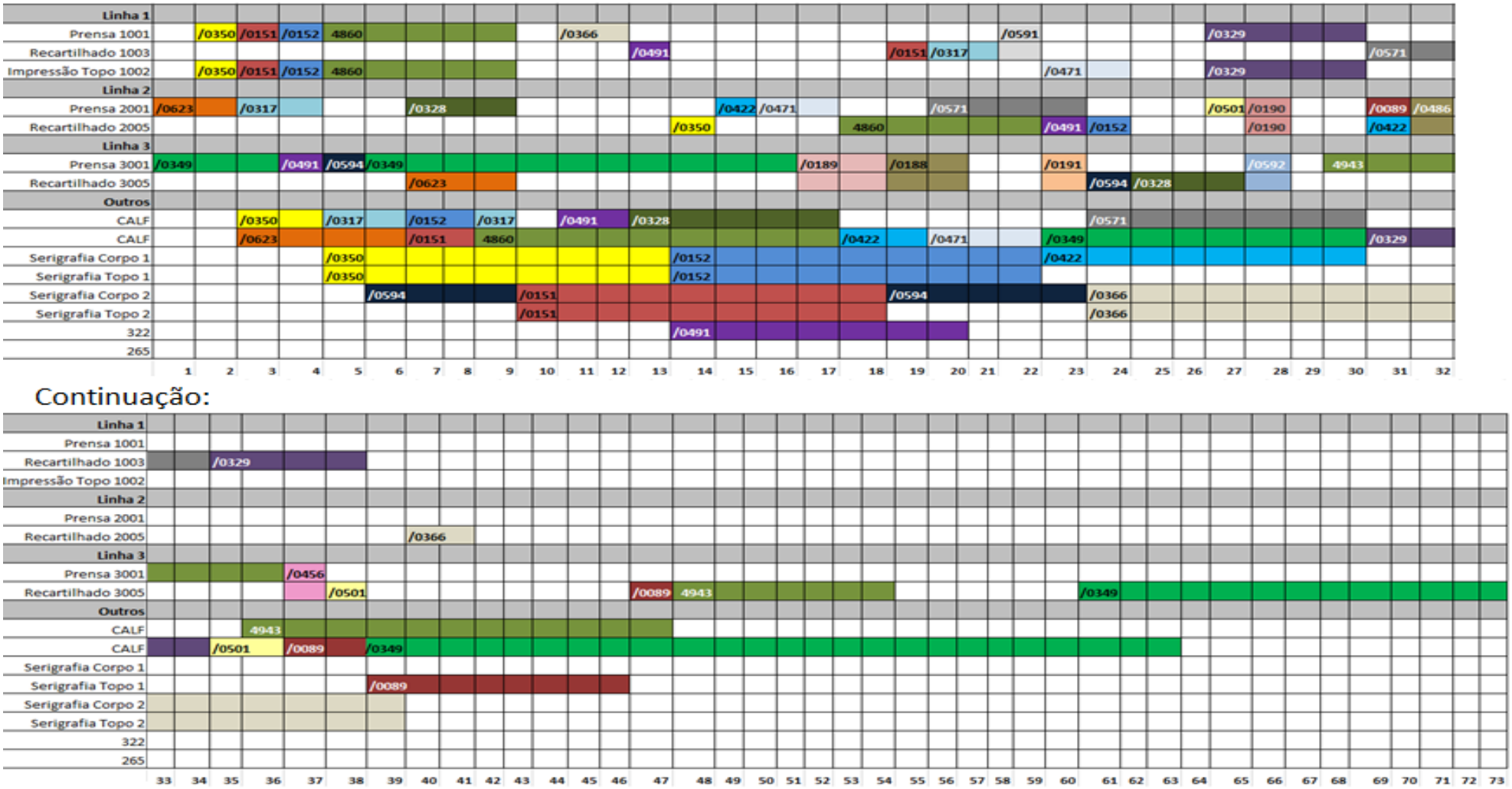


Figura 27 - Escalonamento real da semana de 10 a 14 de fevereiro

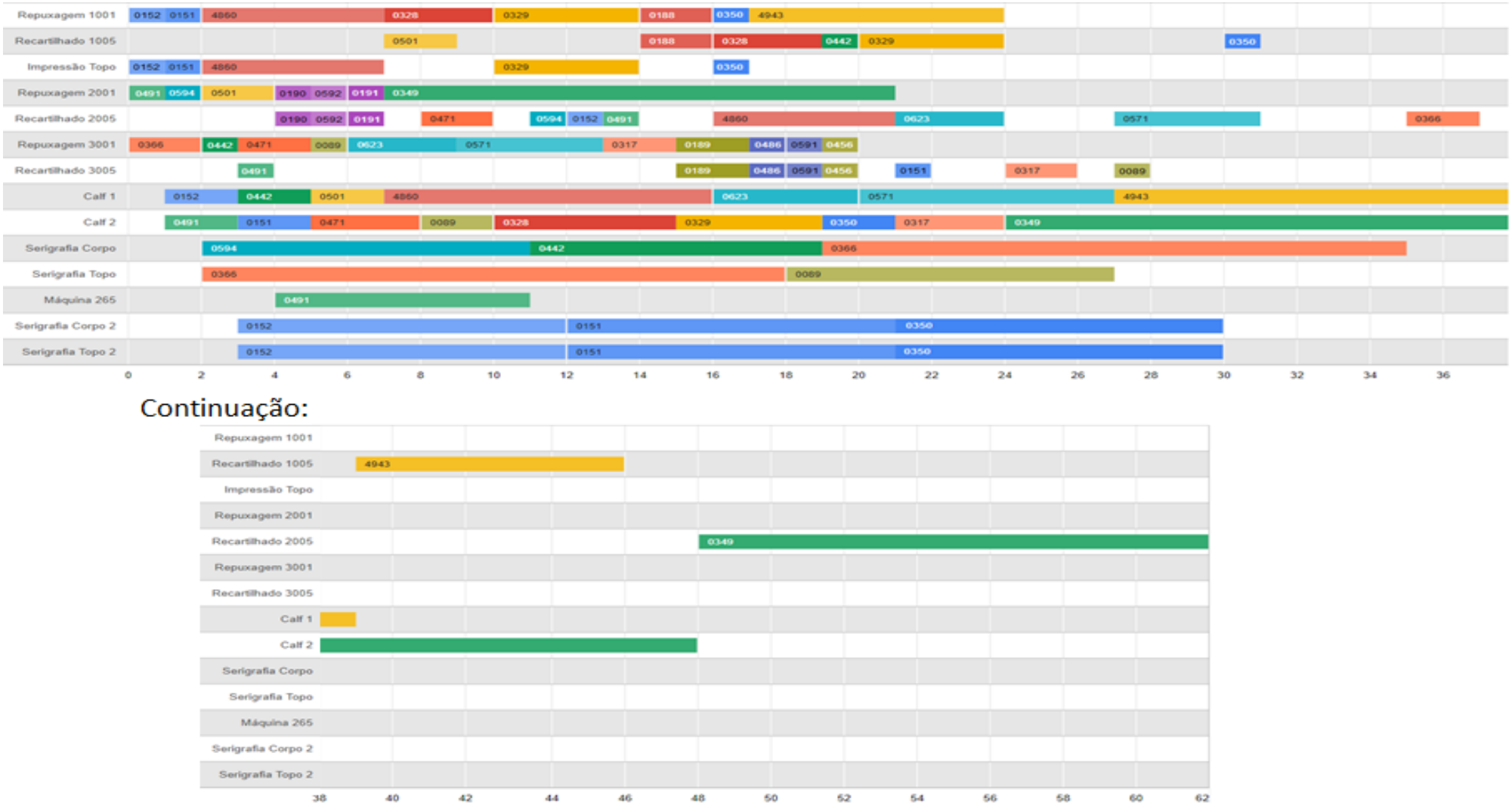


Figura 28 - Escalonamento com recurso ao algoritmo na semana de 10 a 14 de fevereiro

Análise e discussão de resultados

É possível ver as semelhanças durante as primeiras horas de ambos escalonamentos, principalmente na linha 1 (Repuxagem 1001 + Recartilhado 1005) onde são escolhidas praticamente às mesmas encomendas, contudo como esperado após as primeiras horas ambos escalonamentos tornam-se totalmente diferentes isto deve-se ao elevado número de máquinas alternativas que este departamento apresenta. Também é interessante que as últimas encomendas as serem processadas são exatamente as mesmas (4943 e 0349).

Como referido várias vezes, o escalonamento real foi feito no chão de fábrica à base da experiência. Além disso, algumas diferenças de um escalonamento para outro podem dever-se aos seguintes critérios:

- Encomendas atrasadas de semanas anteriores, ou seja, não há um momento inicial com todas as máquinas livres;
- Avarias de equipamentos que podem ter acontecido durante essa semana;
- Ausência ou falta de determinados colaboradores;
- Preferência de certo colaborador presente em determinada linha, para produzir um produto com maior grau de exigência.

Para conseguir comparar ambos o escalonamento foi necessário desenvolver a seguinte tabela que contém as 26 encomendas processadas durante essa semana, juntamente com os seus prazos de entrega e o momento final de processamento das mesmas em ambos os escalonamentos.

Tabela 16 - Encomendas em atraso

Encomenda	Duração prevista (h)	Janela temporal: Data de entrega	Escalonamento CPLEX: Conclusão (h)	Escalonamento Real: Conclusão (h)
0350	12h	[0,12]	31	14
0151	12h	[0,24]	22	19
0152	12h	[0,24]	13	24
0623	9h	[0,32]	24	9
0317	6h	[0,32]	26	21
0328	10h	[0,24]	19	25
0349	51h	[0,44]	62	73
0491	12h	[0,32]	14	23
0188	2h	[0,56]	16	20
4860	19h	[0,32]	21	22
0191	1h	[0,44]	7	23
0189	2h	[0,44]	17	18
0571	15h	[0,44]	31	34
0471	6h	[0,32]	10	24
0456	1h	[0,32]	20	37
0089	12h	[0,44]	28	47
0329	11h	[0,44]	24	38
0366	20h	[0,44]	37	41
0190	1h	[0,44]	5	28
0501	5h	[0,44]	9	38
4943	25h	[0,56]	46	54
0486	1h	[0,56]	18	32
0591	1h	[0,40]	19	22
0592	1h	[0,24]	6	28
0594	11h	[0,24]	12	24
0442	10h	[0,24]	20	31

De forma a avaliar estes escalonamentos não há melhor maneira do que comparar se os prazos de entrega foram cumpridos e é aqui que o escalonamento através do modelo de CPLEX se destaca.

O escalonamento real apresenta 7 encomendas num total de 26 encomendas em atraso, ou seja, aproximadamente 27% dos prazos de entrega não foram cumpridos. Por outro lado, o escalonamento realizado pelo modelo de CPLEX apresenta apenas 2 encomendas em atraso. Traduz-se num atraso de apenas 7% dos prazos de entrega, ou seja, numa melhoria significativa de 20% de cumprimento de prazos de entrega.

Em jeito de conclusão, este resultado já era esperado pois este modelo foi desenvolvido e testado durante vários meses. Durante estes meses também foi possível ver na prática alguns erros efetuados na programação real, tal como as paragens de operações para dar lugar a outras operações de encomendas mais prioritárias mencionadas na Figura 26 que vai contra certos princípios da filosofia *lean* pois geram desperdícios de movimentação e de ocupação de espaço desnecessário. Certos desperdícios como os mencionados podem ser resolvidos recorrendo a este modelo desenvolvido. Apesar de tudo, é de louvar o escalonamento atual pois, mesmo assim tem conseguido dar resposta à maioria das encomendas, contudo existe sempre espaço para melhoria daí ter sido realizado este trabalho com o intuito de otimizar a produção.

5. Conclusões e trabalhos futuros

O presente capítulo apresenta as conclusões gerais deste trabalho, tendo em consideração o principal objetivo proposto e também serão apresentadas propostas de melhorias futuras.

Para melhorar os seus processos de produção, as organizações industriais alocam, de forma recorrente, investimentos com vista à redução constante dos custos, bem como à maximização da sua eficiência e competitividade. Para tal, recorrem cada vez mais ao uso de *software* que permita ganhos de eficiência.

Perante o problema definido pela A.C Relvas, ao longo de sete meses foi desenvolvido um projeto com vista a melhorar o planeamento industrial de um setor desta organização. Foi possível realizar-se um estudo e uma revisão de literatura de um problema de otimização combinatória de recursos num ambiente de *job-shop* flexível numa organização onde este tipo de abordagem nunca tinha sido antes abordado até então.

Nas últimas décadas, os problemas de escalonamento têm constituído o alvo de vários estudos. O principal interesse destes baseia-se no facto de que uma programação de produção otimizada se traduz num processo de grande eficiência produtiva, em que se retira maior rendimento dos equipamentos da organização.

Quanto aos problemas de escalonamento analisados nos meios académicos, são muito diferentes da realidade, uma vez que em ambientes industriais reais existem inúmeras situações e desafios únicos que se podem manifestar. Cada caso é um caso particular e o modelo desenvolvido neste trabalho, serve apenas para modelar o chão de fábrica do *Relwine*. Inicialmente, o processo de planeamento produtivo não era sustentado por qualquer modelo e critério, realizando-se de forma totalmente aleatória e/ou com base da experiência de apenas uma pessoa.

A introdução do CPLEX possibilita o desenvolvimento de um modelo em linguagem CP capaz de dar resposta ao problema que, pelo seu carácter combinatório, apresenta um elevado grau de dificuldade na sua resolução. Além deste *software* possuir uma série de restrições e funções específicas para problemas reais de escalonamento na indústria, apresenta ainda a possibilidade de trabalhar em simultâneo com outras ferramentas e programas externos. Torna-se então possível definir como *inputs* informações proveniente de um ficheiro *excel* assim como exportar os resultados obtidos do modelo para este formato (*outputs*) e transformar estes outputs num gráfico de *Gantt* para uma fácil leitura. Deste modo, este trabalho consistiu num enorme apoio informático para a área de produção e é especialmente útil para os responsáveis deste setor que podem planear e analisar escalonamentos futuros.

Adicionalmente, a implementação deste modelo, conduziu à maximização da eficiência e uma redução significativa de encomendas atrasadas que, por sua vez, se traduz num maior valor acrescentado para a organização.

Ao longo deste trabalho, foram inicialmente tratados dois possíveis cenários (modelo A e modelo B) para testar qual a melhor função objetivo para a organização em questão. As

instâncias utilizadas são relativas a uma semana de trabalho e os resultados dos vários modelos foram comparados. Apesar do primeiro modelo apresentar resultados computacionais significativamente melhores, o segundo destaca-se no que toca a entregar encomendas dentro do prazo definido. Posto isto, acredito que o modelo B cujo função objetivo consiste na minimização do atraso máximo (*tardiness*) seja mais adequada a um contexto industrial.

De seguida, recorreu-se à comparação do escalonamento real de uma semana vs. o escalonamento realizado pelo algoritmo apresentado neste relatório. Os resultados traduzem-se numa redução de 20% nos atrasos de encomendas dessa semana.

É possível chegar à conclusão que os objetivos anteriormente propostos foram, na sua maioria, alcançados com sucesso, relativamente às ferramentas desenvolvidas no contexto do presente trabalho é de esperar que tenham impactos positivos nos campos de eficiência do processo produtivo. Com recurso a estas ferramentas é possível efetuar um escalonamento semanal ou, até mesmo, um escalonamento com diversas encomendas em menos de 30 minutos. O que de facto, torna-se numa enorme ajuda na tomada de decisão, visto é que excelente ter acesso a um planeamento semanal ou quinzenal efetuado por um algoritmo que provou ser eficiente, sendo possível prever e planear diversas situações.

Em relação às limitações, foi considerado a resistência à mudança ou o receio de implementação de um algoritmo ou de uma certa informatização do sistema de planeamento por parte da chefia e do encarregado deste setor. Pois, teria certas implicações ao nível do funcionamento interno. Contudo, a maior limitação deste trabalho diz respeito à duração do mesmo, visto que foi interrompido antes do previsto e não permitiu uma duração de testes mais prolongada nem uma comparação de dados mais rigorosa.

Por último, no que diz respeito a propostas de melhoria futuras na organização, sobressaem as seguintes:

- uma integração destas ferramentas no programa informático da organização;
- melhoria de todo o processo de receção de encomendas, este atualmente apresenta algumas lacunas e várias vezes só chega ao setor produtivo fichas de encomendas a produzir em cima da hora e também é de esperar que o escalonamento tenha de ser realizado quinzenalmente para evitar atrasos;
- a correção de operações presentes no sistema informático que certas encomendas devem seguir;
- apesar de o modelo estar muito próximo da realidade pode existir algum espaço de melhoria, posto isto, era interessante que o modelo especificasse o número de encomendas em atraso, bem como referir quais as que estão em atraso. Da mesma forma que seria útil adicionar uma restrição de equilíbrio das cargas de trabalho nas linhas de produção. Evitando assim sobrecarregar uma determinada linha.

REFERÊNCIAS BIBLIOGRÁFICAS

Ab Wahab, M. N., Nefti-Meziani, S., & Atyabi, A. (2015). A comprehensive review of swarm optimization algorithms. *PLoS ONE*, *10*(5), 1–36. https://doi.org/10.1371/journal.pone.0122827

Alharkan, I. M. (2005). Algorithms for sequencing and scheduling. In *Industrial Engineering Department, King Saud University, Riyadh, Saudi Arabia*.

Ali, M., Haque, M., & Rahman, M. (2018). *Machine Utilization Technique for Job Shop Scheduling using Tabu Search Algorithm*. *6*, 200–205. https://doi.org/10.14741/ijaie/v.6.3.1

Arenales, M., & Armentano, V. (2006). *Pesquisa operacional*. Elsevier Brasil.

Baker, K., & Trietsch, D. (2009). Principles of Sequencing and Scheduling. In *Principles of Sequencing and Scheduling*. https://doi.org/10.1002/9781119262602

Bakos, P., Katzenberger, P., Mitcsenkov, A., Paksy, G., & Cinkler, T. (2011). *Topology Optimization for Next Generation Access Networks: Algorithmic Overview*.

Balas, E. (1969). Machine Sequencing Via Disjunctive Graphs: An Implicit Enumeration Algorithm. *Operations Research*, *17*(6), 941–957. https://doi.org/10.1287/opre.17.6.941

Bartolacci, M., LeBlanc, L., Kayikci, Y., & Grossman, T. (2012). Optimization Modeling for Logistics: Options and Implementations. *Journal of Business Logistics*, *33*, 118–127. https://doi.org/10.1111/j.0000-0000.2012.01044.x

Birgin, E. G., Feofiloff, P., Fernandes, C. G., de Melo, E. L., Oshiro, M. T. I., & Ronconi, D. P. (2014a). A MILP model for an extended version of the Flexible Job Shop Problem. *Optimization Letters*, *8*(4), 1417–1431. https://doi.org/10.1007/s11590-013-0669-7

Birgin, E. G., Feofiloff, P., Fernandes, C. G., de Melo, E. L., Oshiro, M. T. I., & Ronconi, D. P. (2014b). A MILP model for an extended version of the Flexible Job Shop Problem. *Optimization Letters*, *8*(4), 1417–1431. https://doi.org/10.1007/s11590-013-0669-7

Blackstone, J. H., Phillips, D. O. N. T., & Hogg, G. L. (1982). A state-of-the-art survey of dispatching rules for manufacturing job shop operations. *International Journal of Production Research*, *20*(1), 27–45. https://doi.org/10.1080/00207548208947745

Błażewicz, J., Domschke, W., & Pesch, E. (1996a). The job shop scheduling problem: Conventional. *European Journal Of Operational Research*, *93*(1), 1–33.

Błażewicz, J., Domschke, W., & Pesch, E. (1996b). The job shop scheduling problem: Conventional and new solution techniques. *European Journal of Operational Research*, *93*(1), 1–33.

Blazewicz, J., Dror, M., & Weglarz, J. (1991). Mathematical programming formulations for machine scheduling: A survey. *European Journal of Operational Research*, *51*(3), 283–300. https://doi.org/https://doi.org/10.1016/0377-2217(91)90304-E

Brandimarte, P. (1993). Routing and scheduling in a flexible job shop by tabu search. *Annals of Operations Research*, *41*(3), 157–183. https://doi.org/10.1007/BF02023073

Brooks, G. H., & White, C. . (1965). An algorithm for finding optimal or near optimal solutions to the production scheduling problem. *The Journal of Industrial Engineering*, *16*(1), 34–40.

Calleja, G., & Pastor, R. (2014). A dispatching algorithm for flexible job-shop scheduling with transfer batches: an industrial application. *Production Planning & Control*, *25*(2), 93–109. https://doi.org/10.1080/09537287.2013.782846

Chan, F. T. S., Wong, T. C., & Chan, L. Y. (2006). Flexible job-shop scheduling problem under resource constraints. *International Journal of Production Research*, *44*(11), 2071–2089. https://doi.org/10.1080/00207540500386012

Cheng, T. C. E., Hsu, C. J., Huang, Y. C., & Lee, W. C. (2011). Single-machine scheduling with deteriorating jobs and setup times to minimize the maximum tardiness. *Computers and Operations Research*, *38*(12), 1760–1765. https://doi.org/10.1016/j.cor.2010.11.014

Consoli, S. (2008). The development and application of metaheuristics for problems in graph theory: A computational study. In *4OR quarterly journal of the Belgian, French and Italian Operations Research Societies* (Vol. 10).

Cox, J. F., Blackstone Jr, J. H., & Spencer, M. S. (1992). American Production and Inventory Control Society. *American Production and Inventory Control Society, Falls Church, Virginia*.

Davis, L. (1985). Job Shop Scheduling with Genetic Algorithms. *Proceedings of the 1st International Conference on Genetic Algorithms*, 136–140.

Demir, Y., & Kürşat İşleyen, S. (2013). Evaluation of mathematical models for flexible job-shop scheduling problems. *Applied Mathematical Modelling*, *37*(3), 977–988. https://doi.org/https://doi.org/10.1016/j.apm.2012.03.020

Fattahi, P., Saidi-Mehrabad, M., & Jolai, F. (2007). Mathematical modeling and heuristic approaches to flexible job shop scheduling problems. *Journal of Intelligent Manufacturing*, *18*, 331–342. https://doi.org/10.1007/s10845-007-0026-8

Fattahi, P., Saidi Mehrabad, M., & Jolai, F. (2007). Mathematical modeling and heuristic approaches to flexible job shop scheduling problems. *Journal of Intelligent Manufacturing*, *18*(3), 331–342. https://doi.org/10.1007/s10845-007-0026-8

Fisher, R. D., & Thompson, G. L. (1963). *Probabilistic Learning Combinations of Local Job-Shop Scheduling Rules*.

Gao, J., Sun, L., & Gen, M. (2008). A hybrid genetic and variable neighborhood descent algorithm for flexible job shop scheduling problems. *Computers & Operations Research*, *35*, 2892–2907. https://doi.org/10.1016/j.cor.2007.01.001

Garey, M. R., Johnson, D., & Sethi, R. (1976). The complexity of flowshop and jobshop

scheduling. *Mathematics of Operations Research*, *1*, 117–129. https://doi.org/https://doi.org/10.1287/moor.1.2.117

Gen, M., & Cheng, R. (1997). Genetic Algorithms & Engineering Design. John Wiley& Sons. *Inc., New York*.

Ghédira, K., & Ennigrou, M. (2000). *How to Schedule a Job Shop Problem through Agent Cooperation BT - Artificial Intelligence: Methodology, Systems, and Applications*. 132–141.

Gholami, O., & Sotskov, Y. N. (2014). Solving parallel machines job-shop scheduling problems by an adaptive algorithm. *International Journal of Production Research*, *52*(13), 3888–3904. https://doi.org/10.1080/00207543.2013.835498

Giffler, B., & Thompson, G. L. (1960). Algorithms for Solving Production-Scheduling Problems. *Oper. Res.*, *8*(4), 487–503. https://doi.org/10.1287/opre.8.4.487

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

Goldberg, D. E. (1989). Genetic algorithms in search. *Optimization, and MachineLearning*.

Gomes, M., Barbosa-Povoa, A., & Novais, A. (2005). Optimal scheduling for flexible job shop operation. *International Journal of Production Research*, *43*, 2323–2353. https://doi.org/10.1080/00207540412331330101

Haupt, R. (1989). A survey of priority rule-based scheduling. *OR Spektrum*, *11*(1), 3–16. https://doi.org/10.1007/BF01721162

Ho, N. B., & Tay, J. C. (2004a). GENACE: An efficient cultural algorithm for solving the flexible job-shop problem. *Proceedings of the 2004 Congress on Evolutionary Computation, CEC2004*, *2*, 1759–1766. https://doi.org/10.1109/cec.2004.1331108

Ho, N. B., & Tay, J. C. (2004b). GENACE: An efficient cultural algorithm for solving the flexible job-shop problem. In *Proceedings of the IEEE Congress on Evolutionary Computation* (Vol. 2). https://doi.org/10.1109/CEC.2004.1331108

Holland, J. H. (1992). Genetic Algorithms. *Scientific American*, *267*(1), 66–72. https://doi.org/10.1038/scientificamerican0792-66

Huang, H. (2009). Component-based design for job-shop scheduling systems. *International Journal of Advanced Manufacturing Technology*, *45*(9–10), 958–967. https://doi.org/10.1007/s00170-009-2022-y

Hurink, J., Jurisch, B., & Thole, M. (1994). Tabu search for the job-shop scheduling problem with multi-purpose machines. *Operations-Research-Spektrum*, *15*(4), 205–215. https://doi.org/10.1007/BF01719451

İliş, E. Ö. (2004). *Heuristic approaches to scheduling problems in a flexible job shop environment*.

Jackson, J. R. (1956). An extension of Johnson's results on job IDT scheduling. *Naval Research Logistics Quarterly*, *3*(3), 201–203. https://doi.org/10.1002/nav.3800030307

Jain, A. S., & Meeran, S. (1999). Deterministic job-shop scheduling: Past, present and future. *European Journal of Operational Research*, *113*(2), 390–434. https://doi.org/10.1016/S0377-2217(98)00113-1

Jurisch, B. (1995). Lower bounds for the job-shop scheduling problem on multi-purpose machines. *Discrete Applied Mathematics*, *58*(2), 145–156. https://doi.org/10.1016/0166-218X(93)E0124-H

Kacem, I., Hammadi, S., & Borne, P. (2002). Pareto-optimality approach for flexible job-shop scheduling problems: Hybridization of evolutionary algorithms and fuzzy logic. *Mathematics and Computers in Simulation*, *60*(3–5), 245–276. https://doi.org/10.1016/S0378-4754(02)00019-8

Kacem, P. D. I., Hammadi, S., & Borne, P. (2002). Pareto-optimality Approach for Flexible Job-shop Scheduling Problems: Hybridization of Evolutionary Algorithms and Fuzzy Logic. *Mathematics and Computers in Simulation*, *60*, 245–276. https://doi.org/10.1016/S0378-4754(02)00019-8

Kallrath, J. (2013). *Modeling languages in mathematical optimization* (Vol. 88). Springer Science & Business Media.

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

Kronqvist, J., Bernal, D., Lundell, A., & Grossmann, I. (2018). A review and comparison of solvers for convex MINLP. *Optimization and Engineering*, *20*. https://doi.org/10.1007/s11081-018-9411-8

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

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

Laborie, P. (2016). *An introduction to CP Optimizer An introduction to CP Optimizer*. *May*. https://doi.org/10.13140/RG.2.1.4976.9200

Laborie, P., Rogerie, J., Shaw, P., & Vilím, P. (2018). IBM ILOG CP optimizer for scheduling: 20+ years of scheduling with constraints at IBM/ILOG. *Constraints*, *23*(2), 210–250. https://doi.org/10.1007/s10601-018-9281-x

Laporte, G., & Nobert, Y. (1987). Exact algorithms for the vehicle routing problem. In *North-Holland Mathematics Studies* (Vol. 132, pp. 147–184). Elsevier.

Lawler, E. L., Lenstra, J. K., & Rinnooy Kan, A. H. G. (1982). *Recent Developments in*

Deterministic Sequencing and Scheduling: A Survey BT - Deterministic and Stochastic Scheduling (M. A. H. Dempster, J. K. Lenstra, & A. H. G. Rinnooy Kan (eds.); pp. 35–73). Springer Netherlands.

Li, J., Pan, Q., & Liang, Y.-C. (2010). An effective hybrid tabu search algorithm for multi-objective flexible job-shop scheduling problems. *Computers & Industrial Engineering*, *59*(4), 647–662. https://doi.org/https://doi.org/10.1016/j.cie.2010.07.014

Low, C., & Wu, T.-H. (2001). Mathematical modelling and heuristic approaches to operation scheduling problems in an FMS environment. *International Journal of Production Research*, *39*(4), 689–708. https://doi.org/10.1080/00207540150504403

Manne, A. (1960). On the Job-Shop Scheduling Problem. *Operations Research*, *8*(2), 219–223. https://econpapers.repec.org/RePEc:inm:oropre:v:8:y:1960:i:2:p:219-223

Marzouki, B., Driss, O. B., & Ghédira, K. (2017). Decentralized Tabu Searches in Multi Agent System for Distributed and Flexible Job Shop Scheduling Problem. *2017 IEEE/ACS 14th International Conference on Computer Systems and Applications (AICCSA)*, 1019–1026. https://doi.org/10.1109/AICCSA.2017.133

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.

Morrison, D. R., Jacobson, S. H., Sauppe, J. J., & Sewell, E. C. (2016). Branch-and-bound algorithms: A survey of recent advances in searching, branching, and pruning. *Discrete Optimization*, *19*, 79–102. https://doi.org/10.1016/j.disopt.2016.01.005

Nouri, H. E., Belkahla Driss, O., & Ghédira, K. (2018). Solving the flexible job shop problem by hybrid metaheuristics-based multiagent model. *Journal of Industrial Engineering International*, *14*(1), 1–14. https://doi.org/10.1007/s40092-017-0204-z

Osman, I. H. (1995). An introduction to meta-heuristics. *Operational Research Tutorial Papers*, 92–122. https://doi.org/10.1057/jors.1992.43

Özgüven, C., Özbakır, L., & Yavuz, Y. (2010). Mathematical models for job-shop scheduling problems with routing and process plan flexibility. *Applied Mathematical Modelling*, *34*(6), 1539–1548. https://doi.org/https://doi.org/10.1016/j.apm.2009.09.002

Pan, C. H. (1997). A study of integer programming formulations for scheduling problems. *International Journal of Systems Science*, *28*(1), 33–41. https://doi.org/10.1080/00207729708929360

Pezzella, F., Morganti, G., & Ciaschetti, G. (2008). A genetic algorithm for the Flexible Job-shop Scheduling Problem. *Computers and Operations Research*, *35*(10), 3202–3212. https://doi.org/10.1016/j.cor.2007.02.014

Pinedo, M. (2012). *Scheduling* (Vol. 5). Springer.

Piroozfard, H., Wong, K. Y., & Wong, W. P. (2018). Minimizing total carbon footprint and total late work criterion in flexible job shop scheduling by using an improved multi-

objective genetic algorithm. *Resources, Conservation and Recycling*, *128*, 267–283. https://doi.org/https://doi.org/10.1016/j.resconrec.2016.12.001

Ponnambalam, S. G., Aravindan, P., & Chandrasekaran, S. (2001). Constructive and improvement flow shop scheduling heuristics: An extensive evaluation. *Production Planning & Control*, *12*(4), 335–344. https://doi.org/10.1080/09537280152004950

Porter, D. B. (1929). Controlling the manufacture of parts on order and for stock by the Gantt progress chart. *ASME Transactions*, *51*, 105–109.

Potts, C., & Kovalyov, M. (2000). Scheduling with batching: A review. *European Journal of Operational Research*, *120*, 228–249. https://doi.org/10.1016/S0377-2217(99)00153-8

Ramezani, P., Rabiee, M., & Jolai, F. (2015). No-wait flexible flowshop with uniform parallel machines and sequence-dependent setup time: a hybrid meta-heuristic approach. *Journal of Intelligent Manufacturing*, *26*(4), 731–744. https://doi.org/10.1007/s10845-013-0830-2

Rinnooy Kan, A. H. G. (1976). Machine scheduling problems: Classification. In *Complexity and Computations, Nijhoff, The Hague*. Springer US. https://doi.org/10.1007/978-1-4613-4383-7

Roy, B., & Sussmann, B. (1964). Les problemes d'ordonnancement avec contraintes disjonctives. *Note Ds*, *9*.

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

Sarker, R. A., & Newton, C. S. (2007). *Optimization modelling: a practical approach*. CRC press.

Scrich, C. R., Armentano, V. A., & Laguna, M. (2004). Tardiness minimization in a flexible job shop: A tabu search approach. *Journal of Intelligent Manufacturing*, *15*(1), 103–115. https://doi.org/10.1023/B:JIMS.0000010078.30713.e9

Sen, T., & Gupta, S. K. (1984). A state-of-art survey of static scheduling research involving due dates. *Omega*, *12*(1), 63–76. https://doi.org/https://doi.org/10.1016/0305-0483(84)90011-2

Sha, D. Y., & Hsu, C. Y. (2006). A hybrid particle swarm optimization for job shop scheduling problem. *Computers and Industrial Engineering*, *51*(4), 791–808. https://doi.org/10.1016/j.cie.2006.09.002

Shahsavari-Pour, N., & Ghasemishabankareh, B. (2013). A novel hybrid meta-heuristic algorithm for solving multi objective flexible job shop scheduling. *Journal of Manufacturing Systems*, *32*(4), 771–780. https://doi.org/10.1016/j.jmsy.2013.04.015

Shen, W. (2002). Distributed Manufacturing Scheduling Using Intelligent Agents. *IEEE Intelligent Systems*, *17*(1), 88–94. https://doi.org/10.1109/5254.988492

Soleimani, H., Ghaderi, H., Tsai, P.-W., Zarbakhshnia, N., & Maleki, M. (2019). Scheduling of unrelated parallel machines considering sequence-related setup time, start time-dependent deterioration, position-dependent learning and power consumption minimization. *Journal of Cleaner Production*, *xxxx*, 119428. https://doi.org/https://doi.org/10.1016/j.jclepro.2019.119428

Sonmez, A. I., & Baykasoglu, A. (1998). A new dynamic programming formulation of (n x m) flowshop sequencing problems with due dates. *International Journal of Production Research*, *36*(8), 2269–2283. https://doi.org/10.1080/002075498192896

Soto, C., Dorronsoro, B., Fraire, H., Cruz-Reyes, L., Gomez-Santillan, C., & Rangel, N. (2019). Solving the multi-objective flexible job shop scheduling problem with a novel parallel branch and bound algorithm. *Swarm and Evolutionary Computation*, *53*(December 2019), 100632. https://doi.org/10.1016/j.swevo.2019.100632

Talbi, E.-G., & El-Ghazali. (2009). Metaheuristics: From Design to Implementation. In *Metaheuristics: From Design to Implementation* (Vol. 74). https://doi.org/10.1002/9780470496916

Trabelsi, K., Sevaux, M., Coussy, P., Rossi, A., & Sörensen, K. (2010). *Metaheuristics*.

Unlu, Y., & Mason, S. J. (2010). Evaluation of mixed integer programming formulations for non-preemptive parallel machine scheduling problems. *Computers and Industrial Engineering*, *58*(4), 785–800. https://doi.org/10.1016/j.cie.2010.02.012

Valente, J. M. S., & Alves, R. A. F. S. (2008). Heuristics for the single machine scheduling problem with quadratic earliness and tardiness penalties. *Computers & Operations Research*, *35*(11), 3696–3713. https://doi.org/https://doi.org/10.1016/j.cor.2007.04.006

Varela, L., & Carmo-Silva, S. (2008). An Ontology For A Model Of Manufacturing Scheduling Problems To Be Solved On The Web. *International Federation for Information Processing Digital Library; Innovation in Manufacturing Networks;*, *266*. https://doi.org/10.1007/978-0-387-09492-2_21

Vilcot, G., & Billaut, J.-C. (2008). A tabu search and a genetic algorithm for solving a bicriteria general job shop scheduling problem. *European Journal of Operational Research*, *190*(2), 398–411. https://doi.org/https://doi.org/10.1016/j.ejor.2007.06.039

Vilím, P., Laborie, P., & Shaw, P. (2015). *Failure-Directed Search for Constraint-Based Scheduling*. https://doi.org/10.1007/978-3-319-18008-3_30

Vital-Soto, A., Azab, A., & Baki, M. F. (2020). Mathematical modeling and a hybridized bacterial foraging optimization algorithm for the flexible job-shop scheduling problem with sequencing flexibility. *Journal of Manufacturing Systems*, *54*(May 2019), 74–93. https://doi.org/10.1016/j.jmsy.2019.11.010

Wagner, H. M. (1959). An integer linear-programming model for machine scheduling. *Naval Research Logistics Quarterly*, *6*(2), 131–140. https://doi.org/10.1002

/nav.3800060205

Wilson, J. M. (1989). Alternative formulations of a flow-shop scheduling problem. *Journal of the Operational Research Society*, *40*(4), 395–399. https://doi.org/https://doi.org/10.1057/jors.1989.58

Xia, W., & Wu, Z. (2005). An effective hybrid optimization approach for multi-objective flexible job-shop scheduling problems. *Computers and Industrial Engineering*, *48*(2), 409–425. https://doi.org/10.1016/j.cie.2005.01.018

Yamada, T., & Nakano, R. (2000). *Job-Shop Scheduling*.

Zhang, C., Li, P., Guan, Z., & Rao, Y. (2007). A tabu search algorithm with a new neighborhood structure for the job shop scheduling problem. *Computers & Operations Research*, *34*, 3229–3242. https://doi.org/10.1016/j.cor.2005.12.002

Zhang, C. Y., Li, P. G., Rao, Y. Q., & Guan, Z. L. (2008). A very fast TS/SA algorithm for the job shop scheduling problem. *Computers and Operations Research*, *35*(1), 282–294. https://doi.org/10.1016/j.cor.2006.02.024

Zhang, G., Yang, S., & Gao, L. (2008). A genetic algorithm and tabu search for solving flexible job shop schedules. *Proceedings of the 2008 International Symposium on Computational Intelligence and Design, ISCID 2008*, *1*, 369–372. https://doi.org/10.1109/ISCID.2008.202

Zhou, H., Feng, Y., & Han, L. (2001). The hybrid heuristic genetic algorithm for job shop scheduling. *Computers & Industrial Engineering*, *40*(3), 191–200. https://doi.org/https://doi.org/10.1016/S0360-8352(01)00017-1

Zobolas, G., Tarantilis, C., & Ioannou, G. (2008). Exact, Heuristic and Meta-heuristic Algorithms for Solving Shop Scheduling Problems. In *Studies in Computational Intelligence* (Vol. 128, pp. 1–40). https://doi.org/10.1007/978-3-540-78985-7_1

Anexos

ANEXO A

Métodos Aproximados: Meta-Heurísticas

O termo meta-heurística deriva de duas palavras gregas: "Heurística" (do verbo heuriskein) que significa "encontrar"; e o sufixo "Meta" que significa "para além". Antes da adoção generalizada deste termo, a meta-heurística era frequentemente designada por heurística moderna (Osman, 1995).

Segundo Consoli (2008), as meta-heurísticas são estratégias aproximadas, que guiam o processo de pesquisa para explorar eficientemente o espaço de pesquisa de forma a encontrar soluções quase ótimas, utilizam técnicas que vão desde simples procedimentos de pesquisa local a processos de aprendizagem complexos. Estas incorporam mecanismos para que as soluções não fiquem “presas” no ótimo local. Isto é, escapar aos ótimos locais é uma característica desejável para uma meta-heurística, é esta estratégia que caracteriza uma meta-heurística. Por outras palavras e observando a figura seguinte, fugir ao ótimo local implica aceitar soluções piores, permitindo uma degradação provisória da melhor solução obtida até então.

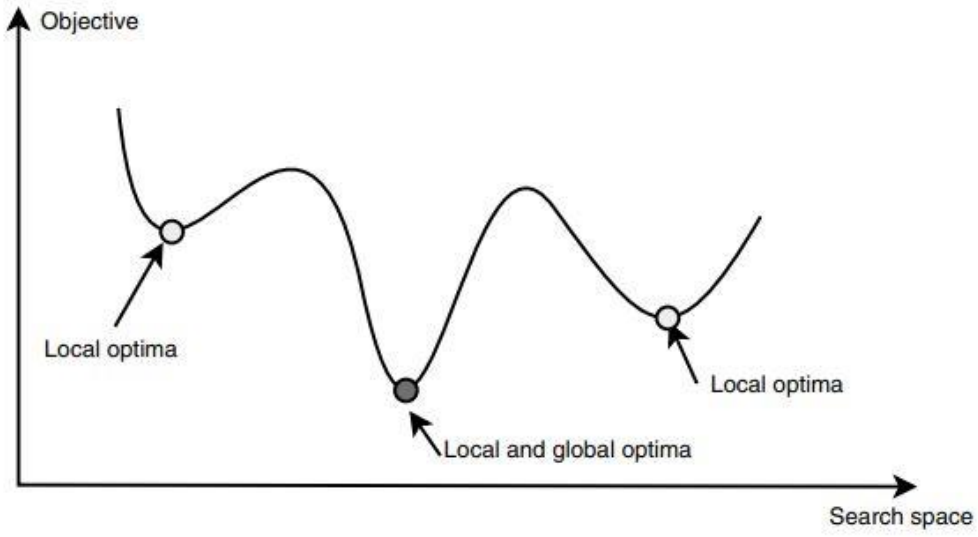


Figura 29 - Local ótimo e global ótimo no espaço de pesquisa retirado de (Talbi & El-Ghazali, 2009)

As meta-heurísticas podem ser distinguidas em duas classes (ver Figura 10), com base na forma de como as soluções são manipuladas. Sendo estas: meta-heurísticas baseadas em pesquisa local e meta-heurísticas baseadas em populações. As meta-heurísticas de

pesquisa local realizam iterativamente pequenas alterações a uma única solução, já as meta-heurísticas baseadas em populações combinam iterativamente soluções em novas soluções.

Meta-heurísticas baseadas em pesquisa local partem de uma solução inicial única, normalmente obtida recorrendo às regras de prioridade acima mencionadas, e procuram, através de iterações sucessivas, aperfeiçoar a solução atual.
O conjunto de soluções que se pode obter aplicando uma única mudança a uma determinada solução é designado por vizinhança (ou *neighborhood*) dessa solução. A cada iteração, a solução atual é substituída por uma solução da sua vizinhança (Trabelsi et al., 2010).

A solução que é melhor do que qualquer solução na sua vizinhança é chamada um ótimo local (ver Figura 31). Quando a solução atual é um ótimo local, uma meta-heurística utilizará uma estratégia para "escapar" a este ótimo local, procurando desta maneira chegar ao ótimo global. Um ótimo global é considerado a melhor solução possível para o problema.

Os exemplos mais populares de meta-heurística de pesquisa local são pesquisa tabu (TS), *simulated annealing* (SA), *variable neighborhood search* (VNS), *greedy randomized adaptive search procedure* (GRASP) e por fim, *iterated local search* (ILS). Neste capítulo, serão apresentados as duas primeiros meta-heurísticas.

Meta-Heurísticas baseadas em Populações

As meta-heurísticas baseadas nas populações encontram boas soluções, selecionando de forma iterativa e depois combinando soluções existentes a partir de um conjunto, designado por população. Estas começam a partir de uma população inicial de soluções e através de métodos iterativos como a seleção de soluções de alta qualidade da população e a recombinação destas soluções em novas soluções que serão reintroduzidas na população, substituindo assim a população atual.

Segundo Trabelsi et al. (2010), a mutação é outra estratégia deste tipo de algoritmo que de modo aleatório muda uma solução após ter sido recombinada, é também regularmente aplicada. A maioria dos algoritmos evolutivos iteram as fases de seleção, recombinação, mutação e reinserção várias vezes, até encontrarem a melhor solução na população.
Alguns dos exemplos mais populares de meta-heurística baseadas em populações são o algoritmo genético (GA), otimização por exame de partículas (em inglês: *particle swarm optimization* ou PSO) e otimização da colônia de formigas (ACO, do inglês ant colony optimization).

Nas ultimas décadas, a adoção de meta-heurísticas como *simulated annealing* (SA), a pesquisa tabu (TS) e os algoritmos genéticos (AG) de modo geral produziram produziram melhores resultados do que as clássicas regras de prioridade ou heurísticas gulosas (Pezzella et al., 2008).

Neste capítulo, será apresentado o algoritmo genético (AG) que pertence ao grupo das meta-heurísticas baseadas nas populações. E relativamente às meta-heurísticas baseadas em pesquisa local serão apresentados a pesquisa tabu (TS) e *simulated annealing* (SA). Estas escolhas devem-se ao elevado número de artigos, até ao momento, publicados relativamente para o FJSP.

Algoritmos Genéticos

Algoritmos genéticos é uma técnica de pesquisa, baseada no conceito de evolução de Darwin. Ao contrário das heurísticas construtivas, estes começam com um conjunto inicial de soluções aleatórias chamado população (ver Figura 32). Os AG manipulam uma população de indivíduos em cada geração (iteração) onde cada indivíduo, denominado de cromossoma, representa uma solução candidata para o problema. Dentro da população, os indivíduos aptos sobrevivem para se reproduzir e o seu respetivo material genético é recombinado para produzir novos indivíduos. Segundo Ponnambalam et al. (2001), para criar a próxima geração, os novos cromossomas, chamados descendentes, por um dos seguintes: (i) pela junção de dois cromossomas da geração atual utilizando um operador

de mutação; ou (ii) pela modificação de um cromossoma utilizando um operador de mutação. Cada solução está associada a um valor de *fitness* que reflete a sua qualidade em comparação com outras soluções na população. Após várias gerações, o algoritmo converge para o melhor cromossoma, o que se deseja que represente a solução óptima ou sub-óptima para o problema (Gen & Cheng, 1997). O processo é então repetido até serem cumpridos os critérios de paragem (Goldberg, 1989).

Foram propostos por Holland (1992) e têm sido utilizados com sucesso numa ampla gama de problemas práticos. Davis (1985) foi o primeiro a aplicar com sucesso um algoritmo genético ao problema de JSP, outros autores seguiram as suas pegadas como prova o número crescente de artigos sobre o tema. Apesar de semelhantes todas as abordagens diferem umas das outras no que respeita à geração inicial da população, seleção cromossómica e estratégias de geração de descendência. Zhou et al. (2001) propuseram um AG híbrido para o JSP, onde as regras de prioridade, como o menor tempo de processamento (SPT) e *most work remaining* (MWKR), foram incorporadas no processo de evolução genética. (Ho & Tay, 2004b) propuseram uma ferramenta baseada em AG chamada GENACE para resolver o FJSP para satisfazer o critério de menor *makespan* possível. Já Pezzella et al. (2008) propuseram um AG aprimorado com o objetivo de reduzir o *makespan* e segundo estes, o seu algoritmo tem um desempenho superior aos AG mais conhecidos para o mesmo problema. Por fim, também é comum encontrar algoritmos híbridos baseados em AG para o problema de FJSP com multiobjectivos como é o caso de Gao et al. (2008), este desenvolveram um algoritmo genético em simultâneo com outra meta-heurística denominada de *variable neighborhood descent* (VND) a fim de reforçar a capacidade de pesquisa.

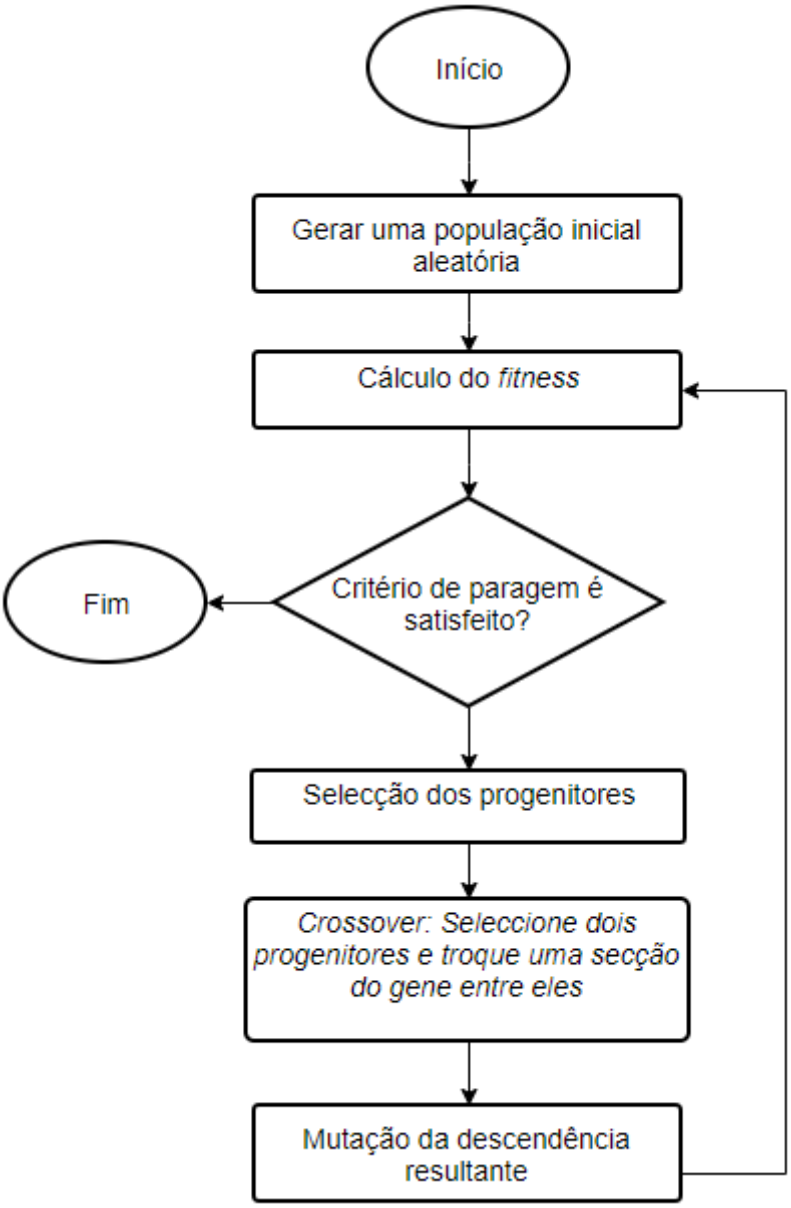


Figura 30 – Fluxograma do Algoritmo Genético baseado em (Ab Wahab et al., 2015)

Pesquisa Tabu

A Pesquisa Tabu ou do inglês *Tabu Search* (TS), inicialmente proposta por (Glover, 1986), tem sido aplicada com sucesso a vários problemas de otimização desde então. A principal característica da pesquisa tabu é a existência de uma estrutura de memórias flexíveis chamada lista tabu que permite que a pesquisa local ultrapasse o ótimo local.

Este metodo parte de uma solução inicial da forma de solução atual e depois move-se sucessivamente entre a vizinhança, aplicando um mecanismo de movimentação na expectativa de descobrir alguma solução melhorada até que se cumpram certos critérios de paragem. Em cada iteração, o processo consiste em selecionar a vizinhança mais adequada que pode até não ser uma solução melhorada. A pesquisa é então repetida a partir da melhor permutação encontrada como a nova solução atual. Para evitar ficar preso em regiões subótimas, a TS utiliza uma estrutura de memória que funciona através do armazenamento do movimento selecionado numa estrutura de dados chamada de lista tabu. Este mecanismo exclui movimentos que levariam a procura de volta onde estava em interações anteriores (Marzouki et al., 2017).

Segundo Li et al. (2010), existem vários parâmetros críticos (ver Figura 33) que afetam o processo de pesquisa e a qualidade da solução do algoritmo TS, nomeadamente:

1. Lista Tabu: A fim de evitar que a lista tabu fique retida no ótimo local, é introduzida a ideia de uma lista tabu. Vários tipos de elementos podem ser armazenados na lista tabu, tais como o valor da função objetivo de uma solução, a posição de um movimento, ou a própria solução. A lista de tabu será atualizada quando for encontrada uma nova solução vizinha ou quando uma solução antiga tiver expirado.
2. Estrutura de vizinhança: Uma estrutura de vizinhança é um mecanismo que pode obter um novo conjunto de soluções de vizinhança através da aplicação de uma pequena perturbação a uma determinada solução. Os movimentos desnecessários e inviáveis devem ser eliminados, se possível, para diminuir a dimensão do conjunto de soluções vizinhas e o tempo computacional.

3. Critério de aspiração: O objetivo do critério de aspiração, sempre que necessário, é evitar impasse quando todas as soluções são proibidas pela lista tabu. Ou seja, se a etapa produzir uma solução superior à melhor solução obtida até agora, então esta etapa é cumprida mesmo que seja tabu.
4. Regra de movimento: Uma regra de movimento é introduzida para evitar o bloqueio quando todas as soluções são proibidas pela lista tabu.

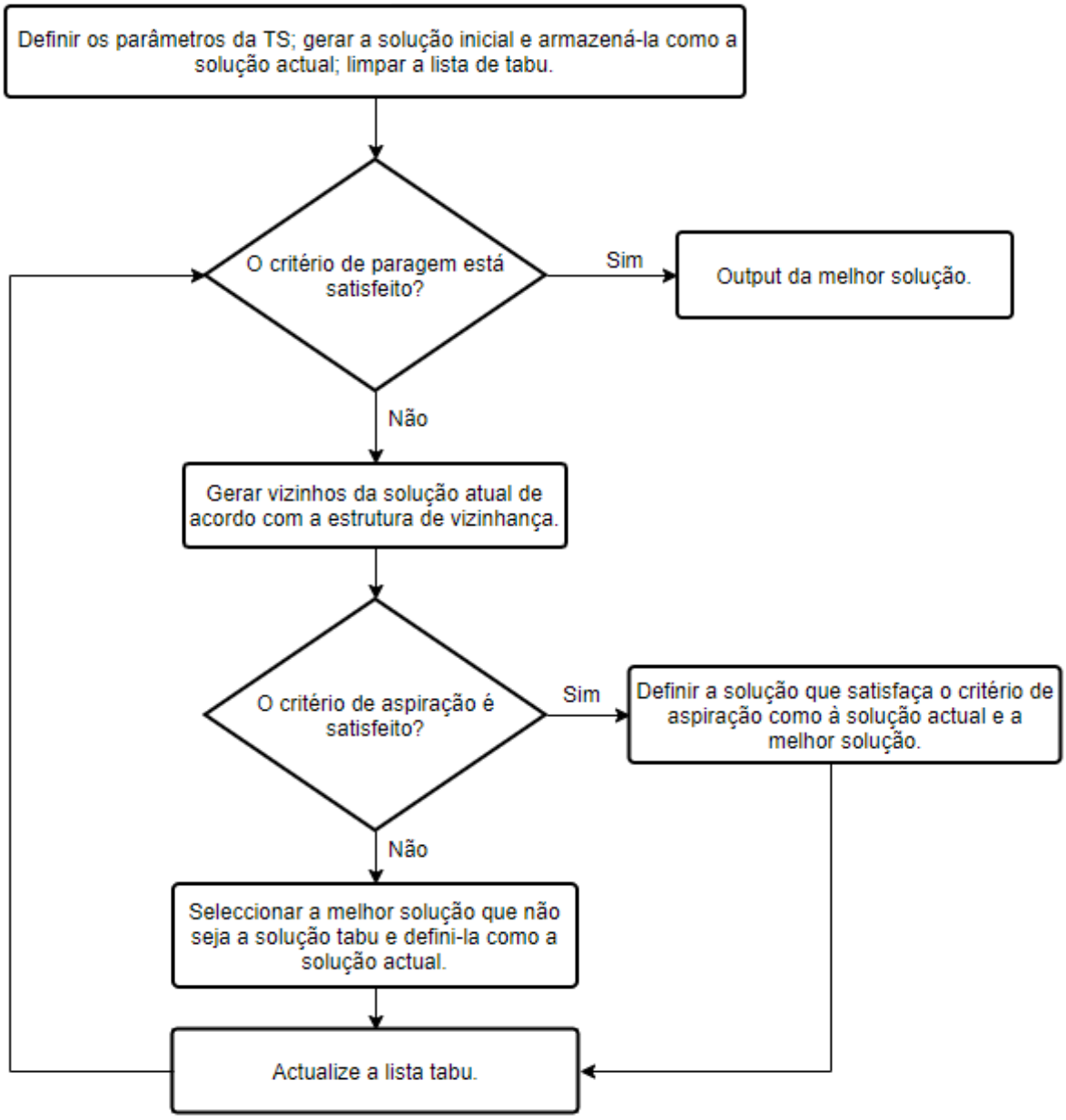


Figura 31 - Fluxograma do algoritmo de Pesquisa Tabu para a resolução do problema JSP de (Ali et al., 2018)

G. Zhang et al. (2008) propõem um algoritmo genético (AG) em conjunto com a pesquisa tabu (TS) com o intuito da minimização do *makespan* e comprovam resultados computacionais eficientes. Ali et al. (2018) utiliza as regras de prioridades de modo a produzir uma solução inicial e a partir dessa solução faz uso da pesquisa tabu. O mesmo acontece com Brandimarte (1993) que propôs um algoritmo hibrido utilizando as regras de sequenciamento juntamente com a pesquisa tabu. Saidi-Mehrabad & Fattahi (2007) apresentam um modelo e soluções detalhadas sobre o FJSP, o objetivo deste passa por minimizar o *makespan* e acreditam que este possa ser aplicado facilmente em ambientes reais.

Simulated annealing

O *simulated annealing* (SA) (ou, em português arrefecimento simulado) é uma meta-heurística baseadas em pesquisa local, este foi inicialmente proposto por Kirkpatrick et al. (1983) e tem por base a analogia do arrefecimento termodinâmico e fundição de metal. Assim sendo, o processo vai diminuindo a cada passo os estados de energia até atingir um estado estável de equilíbrio sólido (Metropolis et al., 1953).

Apresenta a capacidade de escapar aos ótimos locais para a ótimo global. Esta capacidade passa por permitir soluções vizinhas piores do que a solução atual. A probabilidade de aceitação é determinada por um parâmetro de controlo (Temperatura), que diminui sucessivamente durante o procedimento de SA (Vital-Soto et al., 2020).

Ramezani et al. (2015), descreve o algoritmo da seguinte forma. O processo toma como ponto de partida uma solução inicial (ver Figura 34) e continua a produzir e avaliar soluções de vizinhança em diferentes fases, denominadas temperaturas. A cada nível de temperatura é permitido um determinado número de movimentos para se obter uma solução melhor. A temperatura inicial (T) é um parâmetro global que é utilizado para controlar a probabilidade de aceitação e o algoritmo começa também com um valor de temperatura elevado, e depois a temperatura diminui a cada passo. Quando uma solução

de vizinhança tem um melhor valor de função objetivo em relação à solução atual, esta é armazenada como a melhor solução encontrada até agora; caso contrário, o movimento é aceite de acordo com uma probabilidade de aceitação que depende da deterioração Δ da função objetivo. A probabilidade de aceitação é normalmente calculada como $e^{-\frac{\Delta}{T}}$. Segundo Fattahi et al. (2007), a temperatura (T) é progressivamente reduzida de acordo com um esquema de arrefecimento, de modo a que a probabilidade de aceitação de movimentos de deterioração diminua no decurso do processo de arrefecimento.

Tal como a pesquisa Tabu, no SA certos paramentos para a tomada de decisão vão influenciar a qualidade da solução, estes são:

1. Temperatura inicial (T): é inicialmente elevado, o que permite aceitar muitos movimentos inferiores, sendo gradualmente reduzido através da multiplicação por um parâmetro $R \prec 1$ de acordo com o esquema de arrefecimento;
2. A probabilidade de aceitar soluções piores $(e^{-\frac{\Delta}{T}})$;
3. Esquema de arrefecimento: Determina a temperatura (T) do algoritmo a cada iteração;
4. Número de iterações a mesma temperatura (N), normalmente este valor é definido em função do tamanho do problema;
5. Critério de paragem, uma vez atingido este critério o algoritmo termina. Normalmente, este critério vigora até se atingir uma série de iterações sem que a solução tenha sido melhorada ou até se atingir uma determinada temperatura.

Apesar de existir uma grande quantidade de literatura sobre a utilização da meta heurística *simulated annealing* para problemas de escalonamento JSP e de *flow shop*, o mesmo já não é verdade para problemas de FJSP, não existe uma literatura tão abundante. Daí ser comum encontrar este método em conjunto com demais métodos heurísticos, ou seja, a hibridização deste método é bastante comum.

Xia & Wu (2005) aplicaram com sucesso ao problema FJSP uma meta heurística hibrida constituída por SA e PSO (*Particle Swarm Optimization*) e segundo estes, ao hibridizar estas

duas metodologias obtiveram uma abordagem viável e eficaz para resolver o FJSP multiobjectivo. Outro caso de hibridização, pode ser encontrado com Shahsavari-Pour & Ghasemishabankareh (2013). Estes últimos introduzem um novo algoritmo genético em conjunto com o recozimento simulado destinado à resolução dos problemas do FJSP. O algoritmo é denominado de NHGASA (do inglês: *novel hybrid genetic algorithm and simulated annealing*) e o seu objetivo passa por minimizar o tempo máximo de realização de todas as operações (*makespan*), minimizar a carga de trabalho da máquina mais sobrecarregada e por último, minimizar a carga total de trabalho de todas as máquinas.

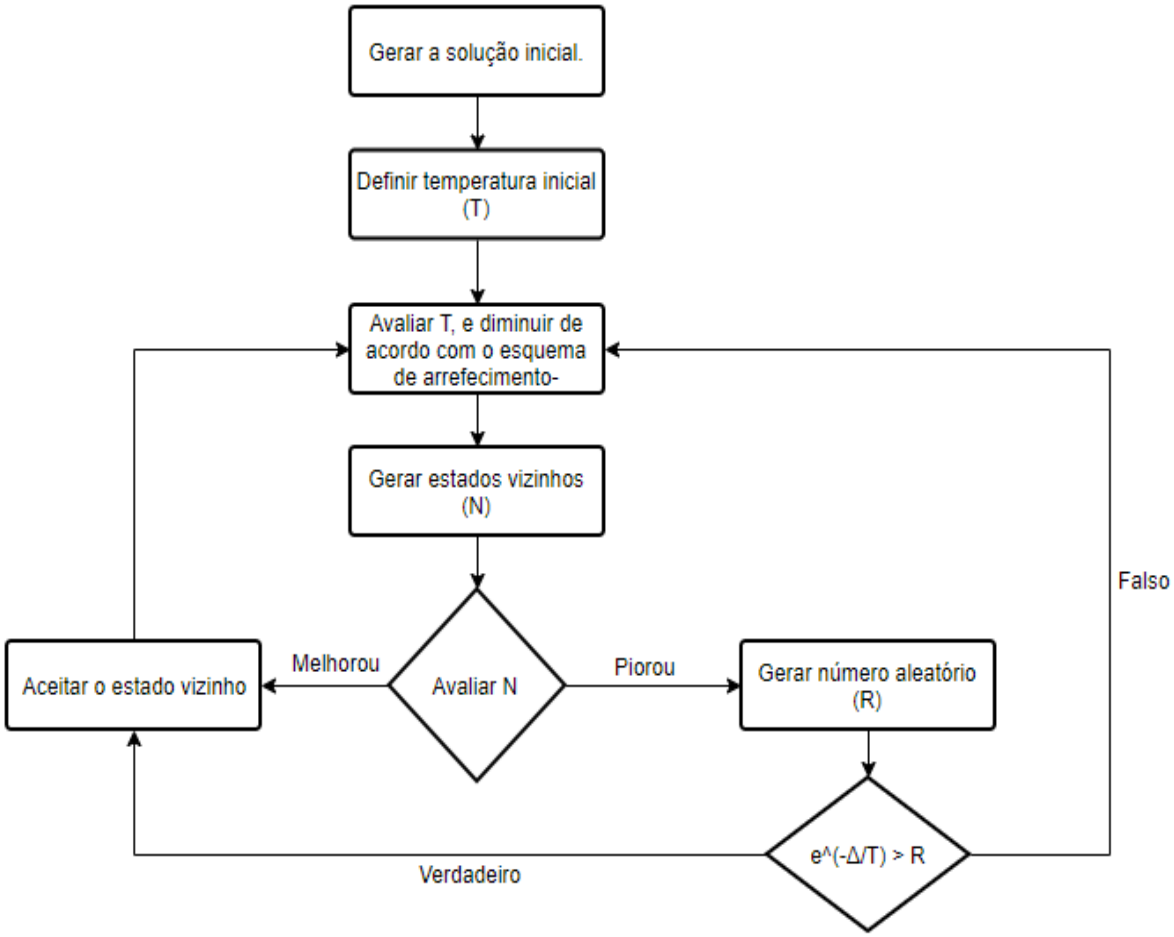


Figura 32 – Fluxograma do algoritmo *simulated annealing* (SA) baseado em (Bakos et al., 2011)

ANEXO B

Tabela 17 - Velocidades das máquinas do sistema

Máquina	Capacidade de produção de cápsulas por hora
Repuxagem/Prensagem	30000 cap/h
Recartilhado e cortes com introdução de disco	30000 cap/h
Calf's	17000 cap/h
Serigrafia corpo	3500 cap/h
Serigrafia Topo	3500 cap/h
Estampagem de lado	3500/4200 cap/h
Estampagem de Topo	3500/4200 cap/h
Máquina 322 (topo)	3600 cap/h
Máquina 265 (topo)	3600 cap/h

ANEXO C

Tabela 18 - Número de identificação dado às máquinas do sistema

Número correspondente:	Nome da máquina:
1	Prensa 1 (Linha 1)
2	Prensa 2 (Linha 2)
3	Prensa 3 (Linha 3)
4	Calf 1
5	Calf 2
6	Recartilhado 1 (Linha 1)
7	Recartilhado 2 (Linha 2)
8	Recartilhado 3 (Linha 3)
9	Máquina 322 (topo)
10	Máquina 265 (topo)
11	Serigrafia corpo
12	Serigrafia Topo
13	Impressão Topo
14	Estampagem de lado
15	Estampagem de Topo
16	Serigrafia corpo 2

ANEXO D

Semana De: 03/02/2020 Até: 07/02/2020 Data de Impressão: 03/02/2020 Rúbrica:_____

Setor: Relwine Mercado: Interno e Externo

CLIENTE		Nº ENC.	DATA PREV. ENTREGA	ARTIGO		QT. ENC.
N.º	NOME			CODIGO	DESCRIÇÃO	
21296	BODEGAS PRINCIPE DE VIA	20200257	03/02/2020	1560220004	30X60 PRINCIPE DE VIANA VERDE ANÓNIMO SARANEX	150 000
21450	SAS CR-DISTRIBUTION	20200087	03/02/2020	C150100262	30X60 SUD DE FRANCE ARGENT/8961C/447C SANS MOULET	57 600
10441	AVELEDA, S.A.	20200015	04/02/2020	C150300025	30X60 FONTE BRANCO RB VERDE 389C/P5815C	200 000
11563	SOC. VINHOS BORGES,S.A.	20200027	04/02/2020	C150300012	30X60 813228 DÃO MEB VERDE/BRANCO	50 000
10441	AVELEDA, S.A.	20194943	05/02/2020	1560211007	30X60 SARANEX 460 AZUL/BRANCO C.G.BRANCO	150 000
11563	SOC. VINHOS BORGES,S.A.	20200284	04/02/2020	C150300019	30X60 813406 GAVAREZ BRANCO/AZUL 7709	25 000
21450	SAS CR-DISTRIBUTION	20200226	04/02/2020	C150100395	30X60 "SUD DE FRANCE " NOIR STD/ROUGE P187C	33 600
21450	SAS CR-DISTRIBUTION	20200227	05/02/2020	C150100250	30X60 SATURNIN "SUD DE FRANCE" NOIR STD/PANT. 3975	33 600
21450	SAS CR-DISTRIBUTION	20200173	05/02/2020	C150100391	30X60 NARBONNAIS "ARTISANS VIGNERONS" OR ACR006/NO	115 200
31237	CORK PERU S. A.	20200189	05/02/2020	156021010	30X60 ROSCA ORO ACR007 ANÓNIMA SARANEX	48 000
31201	ACIC USA	20200177	06/02/2020	C150100270	30X60 INDIGENOUS W. W. BOCA001612 CREAM / BLACK	80 000
21450	SAS CR-DISTRIBUTION	20200285	06/02/2020	1560221003	30X60 VINS DE SATURNIN 3VCLISN75 CREME/OR	115 200
31201	ACIC USA	20200178	06/02/2020	C150100271	30X60 INDIGENOUS W. W. BOCA001613 BLACK/ SILVER	50 000
11563	SOC. VINHOS BORGES,S.A.	20200293	05/02/2020	C150500004	30X60 813516 VERAO EDEKA EVOH PRATA/VERDE	150 000
21440	BODEGAS LOPEZ MORENAS	20194501	07/02/2020	156021001	30X60 ROSCA TAP00032 ANÓNIMA NEGRO ACR001 SARANEX	100 800
21440	BODEGAS LOPEZ MORENAS	20194500	07/02/2020	156021015	30X60 ROSCA VERDE TAP00017 ACR010 ANÓNIMO SARANEX	33 600
11014	DIAGONAL TRADING, LDA.	20194426	07/02/2020	C150100193	30X60 MADXA SARANEX PRETO/DOURADO	100 000
10148	CASA SANTOS LIMA - COMP	20194391	03/02/2020	C150100078	30X60 SC0053 SARANEX AZUL NORUEGA PMS 7696U	68 000
10441	AVELEDA, S.A.	20194432	04/02/2020	C150100109	30X60 LARANJA/BRANCO C.G.TINTO	30 000
10466	SOGRAPE - VINHOS, S.A.	20194316	03/02/2020	1560221003	30X60 13000508 QUINTA DE AZEVEDO P10136C/10139C	10 000

Figura 33 - Mapa de encomendas de (03/02/2020) até (07/02/2020)

ANEXO E

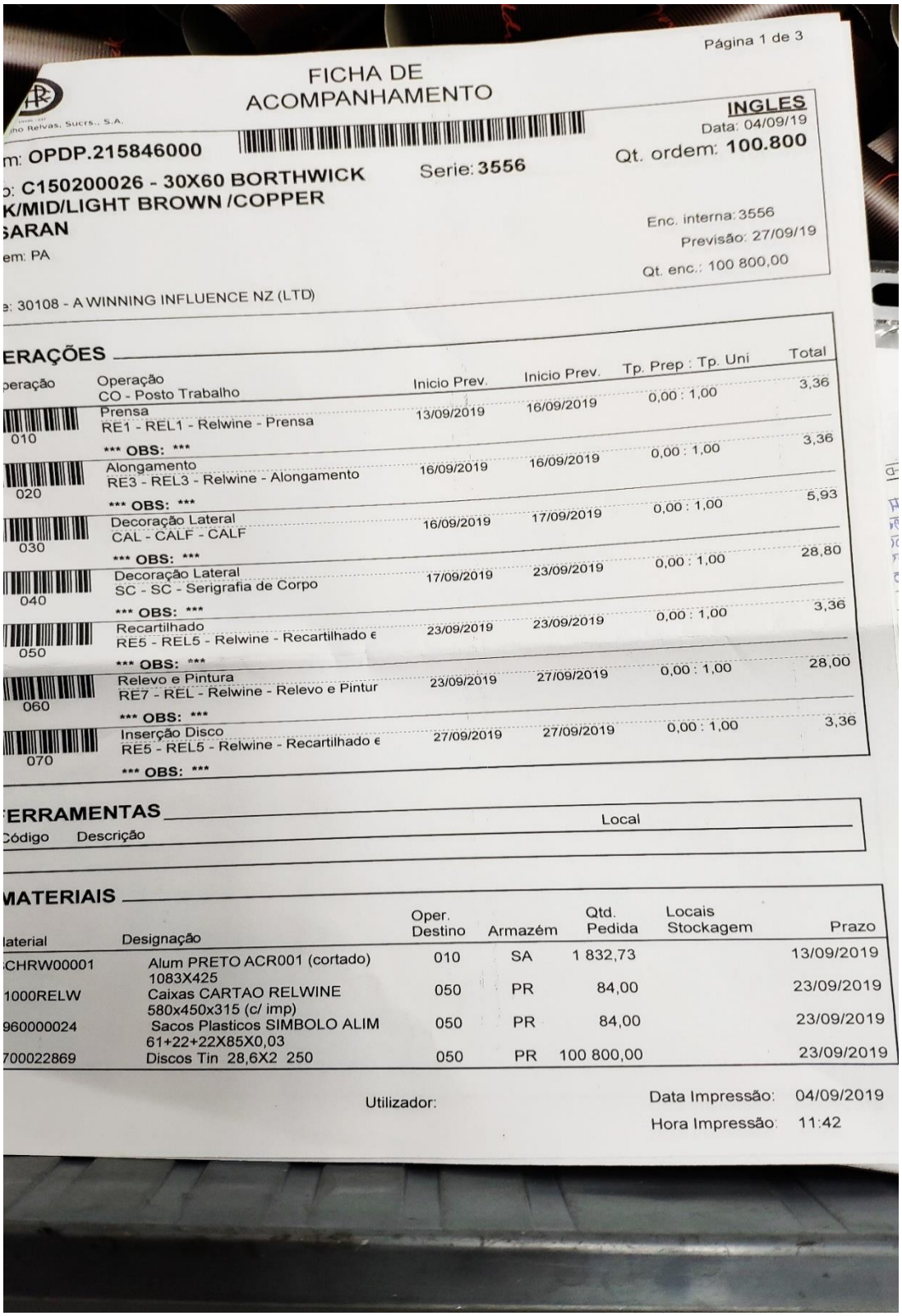
Operações							Alternativas							
Encomenda		**id**		**jobId**		**pos**			**opId**		**maq**		**tp**	
0257	<	1	,	1	,	0	>,	<	1	,	1	,	5	>,
	<	2	,	1	,	1	>,	<	1	,	2	,	5	>,
	<	3	,	1	,	2	>,	<	1	,	3	,	5	>,
0087	<	4	,	2	,	0	>,	<	2	,	4	,	9	>,
	<	5	,	2	,	0	>,	<	2	,	5	,	9	>,
	<	6	,	2	,	1	>,	<	3	,	6	,	5	>,
	<	7	,	2	,	2	>,	<	3	,	7	,	5	>,
0015	<	8	,	3	,	0	>,	<	3	,	8	,	5	>,
	<	9	,	3	,	1	>,	<	4	,	1	,	2	>,
	<	10	,	3	,	2	>,	<	5	,	13	,	2	>,
0027	<	11	,	4	,	0	>,	<	6	,	4	,	4	>,
	<	12	,	4	,	1	>,	<	6	,	5	,	4	>,
	<	13	,	4	,	2	>,	<	7	,	6	,	2	>,
	<	14	,	4	,	3	>,	<	7	,	7	,	2	>,
4943	<	15	,	5	,	0	>,	<	7	,	8	,	2	>,
	<	16	,	5	,	1	>,	<	8	,	1	,	7	>,
	<	17	,	5	,	2	>,	<	8	,	2	,	7	>,
0284	<	18	,	6	,	0	>,	<	8	,	3	,	7	>,
	<	19	,	6	,	1	>,	<	9	,	4	,	12	>,
	<	20	,	6	,	2	>,	<	9	,	5	,	12	>,
0226	<	21	,	7	,	0	>,	<	10	,	6	,	7	>,
	<	22	,	7	,	1	>,	<	10	,	7	,	7	>,
	<	23	,	7	,	2	>,	<	10	,	8	,	7	>,
0227	<	24	,	8	,	0	>,	<	11	,	1	,	2	>,
	<	25	,	8	,	1	>,	<	11	,	2	,	2	>,
	<	26	,	8	,	2	>,	<	11	,	3	,	2	>,
0173	<	27	,	9	,	0	>,	<	12	,	4	,	3	>,
	<	28	,	9	,	0	>,	<	12	,	5	,	3	>,
	<	29	,	9	,	1	>,	<	13	,	11	,	14	>,
	<	30	,	9	,	2	>,	<	13	,	16	,	14	>,
0189	<	31	,	10	,	0	>,	<	14	,	6	,	2	>,
	<	32	,	10	,	0	>,	<	14	,	7	,	2	>,
0177	<	33	,	11	,	0	>,	<	14	,	8	,	2	>,
	<	34	,	11	,	1	>,	<	15	,	1	,	7	>,
	<	35	,	11	,	2	>,	<	15	,	2	,	7	>,
	<	36	,	11	,	3	>,	<	15	,	3	,	7	>,
	<	37	,	11	,	4	>,	<	16	,	4	,	12	>,
0285	<	38	,	12	,	0	>,	<	16	,	5	,	12	>,
	<	39	,	12	,	1	>,	<	17	,	6	,	7	>,
	<	40	,	12	,	2	>,	<	17	,	7	,	7	>,

0178	< 41 , 13 , 0 >,	< 17 , 8 , 7 >,
	< 42 , 13 , 1 >,	< 18 , 1 , 1 >,
	< 43 , 13 , 2 >,	< 18 , 2 , 1 >,
	< 44 , 13 , 3 >,	< 18 , 3 , 1 >,
	< 45 , 13 , 4 >,	< 19 , 4 , 2 >,
0293	< 46 , 14 , 0 >,	< 19 , 5 , 2 >,
	< 47 , 14 , 1 >,	< 20 , 6 , 1 >,
	< 48 , 14 , 2 >,	< 20 , 7 , 1 >,
4501	< 49 , 15 , 0 >,	< 20 , 8 , 1 >,
	< 50 , 15 , 0 >,	< 21 , 1 , 2 >,
4500	< 51 , 16 , 0 >,	< 21 , 2 , 2 >,
	< 52 , 16 , 0 >,	< 21 , 3 , 2 >,
4426	< 53 , 17 , 0 >,	< 22 , 11 , 10 >,
	< 54 , 17 , 1 >,	< 22 , 16 , 10 >,
	< 55 , 17 , 2 >,	< 23 , 6 , 2 >,
4391	< 56 , 18 , 0 >,	< 23 , 7 , 2 >,
	< 57 , 18 , 0 >,	< 23 , 8 , 2 >,
	< 58 , 18 , 1 >,	< 24 , 1 , 2 >,
	< 59 , 18 , 2 >,	< 24 , 2 , 2 >,
4432	< 60 , 19 , 0 >,	< 24 , 3 , 2 >,
	< 61 , 19 , 1 >,	< 25 , 11 , 10 >,
	< 62 , 19 , 2 >,	< 25 , 16 , 10 >,
4316	< 63 , 20 , 0 >,	< 26 , 6 , 2 >,
	< 64 , 20 , 1 >,	< 26 , 7 , 2 >,
	< 65 , 20 , 2 >,	< 26 , 8 , 2 >,
		< 27 , 1 , 4 >,
		< 28 , 13 , 4 >,
		< 29 , 4 , 7 >,
		< 29 , 5 , 7 >,
		< 30 , 6 , 4 >,
		< 30 , 7 , 4 >,
		< 30 , 8 , 4 >,
		< 31 , 1 , 2 >,
		< 31 , 2 , 2 >,
		< 31 , 3 , 2 >,
		< 32 , 6 , 2 >,
		< 32 , 7 , 2 >,
		< 32 , 8 , 2 >,
		< 33 , 1 , 3 >,
		< 33 , 2 , 3 >,
		< 33 , 3 , 3 >,
		< 34 , 4 , 5 >,

	< 34 , 5 , 5 >, < 35 , 6 , 3 >, < 35 , 7 , 3 >, < 35 , 8 , 3 >, < 36 , 9 , 20 >, < 36 , 10 , 20 >, < 37 , 6 , 3 >, < 37 , 7 , 3 >, < 37 , 8 , 3 >, < 38 , 1 , 4 >, < 38 , 2 , 4 >, < 38 , 3 , 4 >, < 39 , 4 , 7 >, < 39 , 5 , 7 >, < 40 , 6 , 4 >, < 40 , 7 , 4 >, < 40 , 8 , 4 >, < 41 , 1 , 2 >, < 41 , 2 , 2 >, < 41 , 3 , 2 >, < 42 , 4 , 3 >, < 42 , 5 , 3 >, < 43 , 6 , 2 >, < 43 , 7 , 2 >, < 43 , 8 , 2 >, < 44 , 9 , 12 >, < 44 , 10 , 12 >, < 45 , 6 , 2 >, < 45 , 7 , 2 >, < 45 , 8 , 2 >, < 46 , 1 , 5 >, < 46 , 2 , 5 >, < 46 , 3 , 5 >, < 47 , 4 , 9 >, < 47 , 5 , 9 >, < 48 , 6 , 5 >, < 48 , 7 , 5 >, < 48 , 8 , 5 >, < 49 , 1 , 4 >, < 49 , 2 , 4 >, < 49 , 3 , 4 >, < 50 , 6 , 4 >,

	< 50 , 7 , 4 >, < 50 , 8 , 4 >, < 51 , 1 , 2 >, < 51 , 2 , 2 >, < 51 , 3 , 2 >, < 52 , 6 , 2 >, < 52 , 7 , 2 >, < 52 , 8 , 2 >, < 53 , 1 , 4 >, < 53 , 2 , 4 >, < 53 , 3 , 4 >, < 54 , 4 , 6 >, < 54 , 5 , 6 >, < 55 , 6 , 4 >, < 55 , 7 , 4 >, < 55 , 8 , 4 >, < 56 , 1 , 3 >, < 57 , 13 , 3 >, < 58 , 4 , 4 >, < 58 , 5 , 4 >, < 59 , 6 , 3 >, < 59 , 7 , 3 >, < 59 , 8 , 3 >, < 60 , 1 , 1 >, < 60 , 2 , 1 >, < 60 , 3 , 1 >, < 61 , 11 , 9 >, < 61 , 16 , 9 >, < 62 , 6 , 1 >, < 62 , 7 , 1 >, < 62 , 8 , 1 >, < 63 , 1 , 1 >, < 63 , 2 , 1 >, < 63 , 3 , 1 >, < 64 , 4 , 1 >, < 64 , 5 , 1 >, < 65 , 6 , 1 >, < 65 , 7 , 1 >, < 65 , 8 , 1 >,

ANEXO F



FICHA DE ACOMPANHAMENTO

Página 1 de 3

INGLES

Data: 04/09/19

m: OPDP.215846000 Serie: 3556 Qt. ordem: 100.800

o: C150200026 - 30X60 BORTHWICK
K/MID/LIGHT BROWN /COPPER
SARAN

em: PA

Enc. interna: 3556
Previsão: 27/09/19
Qt. enc.: 100 800,00

e: 30108 - A WINNING INFLUENCE NZ (LTD)

ERAÇÕES

peração	Operação CO - Posto Trabalho	Inicio Prev.	Inicio Prev.	Tp. Prep : Tp. Uni	Total
010	Prensa RE1 - REL1 - Relwine - Prensa	13/09/2019	16/09/2019	0,00 : 1,00	3,36
	*** OBS: ***				
020	Alongamento RE3 - REL3 - Relwine - Alongamento	16/09/2019	16/09/2019	0,00 : 1,00	3,36
	*** OBS: ***				
030	Decoração Lateral CAL - CALF - CALF	16/09/2019	17/09/2019	0,00 : 1,00	5,93
	*** OBS: ***				
040	Decoração Lateral SC - SC - Serigrafia de Corpo	17/09/2019	23/09/2019	0,00 : 1,00	28,80
	*** OBS: ***				
050	Recartilhado RE5 - REL5 - Relwine - Recartilhado €	23/09/2019	23/09/2019	0,00 : 1,00	3,36
	*** OBS: ***				
060	Relevo e Pintura RE7 - REL - Relwine - Relevo e Pintur	23/09/2019	27/09/2019	0,00 : 1,00	28,00
	*** OBS: ***				
070	Inserção Disco RE5 - REL5 - Relwine - Recartilhado €	27/09/2019	27/09/2019	0,00 : 1,00	3,36
	*** OBS: ***				

ERRAMENTAS

Código	Descrição	Local

MATERIAIS

laterial	Designação	Oper. Destino	Armazém	Qtd. Pedida	Locais Stockagem	Prazo
CHRW00001	Alum PRETO ACR001 (cortado) 1083X425	010	SA	1 832,73		13/09/2019
1000RELW	Caixas CARTAO RELWINE 580x450x315 (c/ imp)	050	PR	84,00		23/09/2019
960000024	Sacos Plasticos SIMBOLO ALIM 61+22+22X85X0,03	050	PR	84,00		23/09/2019
700022869	Discos Tin 28,6X2 250	050	PR	100 800,00		23/09/2019

Utilizador:

Data Impressão: 04/09/2019
Hora Impressão: 11:42

Figura 34 - Ficha de acompanhamento de uma encomenda com respetivas operações e tempo unitário previsto