Faculdade de Engenharia da Universidade do Porto

Otimização do *Job Shop Scheduling* integrando ferramentas *Lean* e TOC

Luís Filipe da Silva Tunes

Dissertação realizada no âmbito do
Mestrado Integrado em Engenharia Mecânica
Gestão da Produção

Orientador: Prof. Dr. Jorge Freire de Sousa

Sexta-feira, 23 de julho de 2021

Resumo

Um dos principais objetivos das empresas industriais nacionais modernas é o cumprimento de um plano a curto prazo que seja, simultaneamente, estável e flexível. Um dos aspetos da estabilidade tem a ver com o aumento dos tamanhos dos lotes, com a consequente diminuição dos custos operacionais, e a flexibilidade é desejada para dar resposta às múltiplas solicitações do mercado.

Este trabalho apresenta o processo de criação de uma ferramenta que automatiza parcialmente o planeamento da produção de uma empresa PME especializada na assemblagem e teste de equipamento eletrónico e de telecomunicações. Na base do sequenciamento de encomendas, encontra-se o problema *Job Shop Scheduling* para o qual são propostas diversas abordagens, integrando-se as filosofias *Lean* e *Theory of Constraints* e gerindo-se a alocação de recursos humanos e restrições associadas ao contexto prático.

Em aberto é deixado o desenvolvimento de ferramentas estocásticas e a análise do lucro e do risco associado. Menciona-se, como considerações futuras, algoritmos de dimensionamento de lotes e um modelo baseado em sistemas adaptativos complexos a ser implementado em sistemas de *machine learning*.

Abstract

One of the main objectives of modern national industrial companies is the fulfilment of a short-term plan that is both stable and flexible. Stability is desired by increasing batch sizes, with the consequent decrease in operating costs, and flexibility is desired to respond to multiple market requests. One of the aspects of stability has to do with increasing batch sizes, with the consequent decrease in operating costs, and flexibility is desired to respond to multiple market requests.

This work presents the process of creating a tool that automates the production planning of an SME company specializing in the assembly and testing of electronic and telecommunications equipment. At the base of order sequencing, there is the problem Job Shop Scheduling" for which different approaches are proposed, integrating the Lean and Theory of Constraints philosophies, and managing the allocation of human resources and restrictions linked to the practical context.

It is left in open the development of stochastic tools and the analysis of the profit and associated risk. As future considerations, batch sizing algorithms and a model based on complex adaptive systems to be implemented in machine learning systems are mentioned.

Agradecimentos

Agradeço, em primeiro lugar, aos meus pais, pelo constante apoio e motivação ao longo de todo o meu percurso académico, assim como pelo esforço de ambos, que possibilitou a minha formação académica.

Ao meu orientador, o Professor Dr. Jorge Freire de Sousa, pela oportunidade que me ofereceu de realizar este projeto, bem como pela constante disponibilidade e indicações sobre as melhores direções a tomar.

Ao meu co-orientador, MSc Artur Matos que me acompanhou diariamente na empresa HFA (Henrique Fernando Alves), pelo apoio e motivação, assim como pela ajuda prestada na compreensão e resolução de problemas que surgiram ao longo deste projeto.

Aos administradores da HFA pela oportunidade que me ofereceram de realizar este projeto e, em particular, ao Eng.º Carlos Alves pela informação e ajuda prestada

Aos restantes elementos da HFA deixo também o meu agradecimento, não apenas pelo acolhimento, mas também pela disponibilidade e paciência mostrada no esclarecimento de dúvidas e auxílio sempre que foi necessário na recolha e obtenção de informação.

À Inova-Ria pela excelente organização e apoio no âmbito do Programa GENIUS Investigação.

Índice

Resumo ..... **iii**

Abstract ..... **iv**

Agradecimentos ..... **v**

Índice ..... **vi**

Lista de figuras ..... **viii**

Lista de tabelas ..... **i**

Abreviaturas e Símbolos ..... **ii**

Capítulo 1 ..... **1**

Introdução 1
1.1 - Contexto da dissertação 1
1.2 - Problema e metodologia 1
1.3 - Organização da tese 2

Capítulo 2 ..... **3**

Contexto da dissertação 3
2.1 - *Lean* 3
2.2 - TOC – Teoria das Restrições 6
2.3 - Sequenciamento 10

Capítulo 3 ..... **19**

Encomendas e Produção 19
3.1 - Planta da Fábrica 19
3.2 - Encomendas abertas 19
3.3 - Etapas da produção 20
3.4 - Matriz Carga 21
3.5 - Pressupostos 22
3.6 - Gargalo 23
3.7 - Equipamento 23
3.8 - Matriz Carga - Equipamentos 24
3.9 - Turnos 24
3.10 - Clientes 25
3.11 - Mão de Obra 26
3.12 - Outros Dados 28

Capítulo 4 30

Sequenciamento de tarefas 30
4.1 - Regras de prioridade - avaliação 30
4.2 - Problema com n tarefas e uma máquina 30
4.3 - Problema com *n* tarefas e duas máquinas 36
4.4 - Problema com *n* tarefas e *m* máquinas 38
4.5 - COVERT modificado 40

Capítulo 5 42

Programa e Resultados 42
5.1 - Novas Considerações 42
5.2 - Complexidade do Programa 42
5.3 - Simulação 1 43
5.4 - Simulação 2 46

Capítulo 6 48

Conclusão 48
6.1 - Conclusões 48
6.2 - Recomendações futuras 49
6.3 - Considerações finais 52

Referências 53

ANEXO A 56

Planta do chão de fábrica 56

ANEXO B 58

VSM – *Value Stream Mapping* 58

ANEXO C 62

Código fonte do programa desenvolvido 62

Lista de figuras

Figura 2.1 – PDCA 4
Figura 2.2 – Exemplo de planta industrial no início do sec. XX. 4
Figura 2.3 - Produção *Lean* - Sistema *Pull*. 5
Figura 2.4 - Fluxo Linear com um Gargalo 8
Figura 2.5 - Fluxo Linear com um Recurso de Capacidade Limitada 8
Figura 2.6 – Diagrama de Euler – Tempo execução 11
Figura 2.7 – Tarefas processadas sequencialmente 12
Figura 2.8 – Solução Ótima 12
Figura 2.9 – Problema da mochila. 15
Figura 2.10 – Método SA. 17
Figura 2.11 – Simulação do método SA para o problema LA01 18
Figura 3.1 – Etapas de produção 21
Figura 3.2 – Recursos Humanos – Formação 27
Figura 3.3 – Layout 28
Figura 3.4 – Gantt 28
Figura 5.1 – Simulação 1 vs Plano HFA semana 18 45
Figura 5.2 – Comparação do tempo de ciclo médio 45
Figura 5.3 – Simulação 2 vs Plano HFA semana 21 47
Figura 6.1 – Modelo Taxonómico – Perspetiva dos CAS 51
Figura Anexo A.1 – Visão geral 56
Figura Anexo A.2 –Linhas de Produção 57
Figura Anexo B.1 –Linha 2 58

Figura Anexo B.2 – Linha 2 – 1/3 59
Figura Anexo B.3 – Linha 2 – 2/3 60
Figura Anexo B.4 – Linha 2 – 3/3 61
Figura Anexo C.1 – Diagrama de sequência 62

Lista de tabelas

Tabela 2-1 — Diferenças entre *Push* e *Pull* 5

Tabela 2-2 — Problema LA01 18

Tabela 3-1 — Matriz Carga 22

Tabela 3-2 — Custos hora de atividade por equipamento 23

Tabela 3-3 — Matriz Carga – Equipamento 24

Tabela 3-4 — Turnos 25

Tabela 3-5 — Mão-de-obra 26

Tabela 3-6 — Soldadura Seletiva vs Soldadura Onda 27

Tabela 3-7 — Conversão da cadência na linha 1 para as outras linhas de produção 29

Tabela 4-1 — Índice OEE de referência por linha 32

Tabela 4-2 — Carga por operação 34

Tabela 4-3 — Número de pessoas por operação 34

Tabela 4-4 — SPT vs EDD 35

Tabela 4-5 — Máquina 1 (SMT) e Máquina 2 (THT + Outros) 36

Tabela 4-6 — Máquina 1 (SMT + THT) e Máquina 2 (Outros) 37

Tabela 5.1 — *Big* O *Notation - Timsort*. 43

Abreviaturas e Símbolos

Lista de abreviaturas (ordenadas por ordem alfabética)

AI	Artificial Intelligence
CAS	Complex Adaptive Systems
CBSD	Community Based System Dynamics
CCR	Capacity-Constrained Resource
CCTF	Collaborative Causal Theory Formation
CDS	Campbell, Dudek e Smith
COVERT	Cost Over Time
CRP	Capability Resource Planning
EDD	Earliest Due Date
ERP	Enterprise Resource Planning
JIT	Just in Time
JSSP	Job Shop Scheduling Problem
MIP	Mixed-Integer Linear Program
ML	Machine Learning
MRP	Material Requirement Planning
MTO	Make to Order
MTS	Make to Stock
OEE	Overall Equipment Effectiveness
OPT	Optimized Production Technology
PCB	Printed Circuit Board
SA	Simulated Annealing
SKU	Stock Keeping Unit
SMD	Surface Mounted Device
SMED	Single Minute Exchange of Dies
SMT	Surface Mounted Technology
SPT	Shortest Processing Time

TAT	Turnaround time
THT	Through-hole technology
TOC	Theory of Constraints
TPM	Total Productive Maintenance
TPS	Toyota Production System
TQC	Total Quality Control
VSM	Value Stream Mapping
WIP	Work in Progress
WT	Waiting time

Capítulo 1

Introdução

Neste capítulo, apresenta-se o projeto proposto e a empresa em que foi desenvolvido. Elabora-se uma metodologia de abordagem para o problema, desenvolvem-se parâmetros de avaliação dos objetivos definidos e enuncia-se a estrutura da dissertação.

1.1 - Contexto da dissertação

Esta dissertação do Mestrado Integrado em Engenharia Mecânica, área de Gestão da Produção, foi efetuada em ambiente empresarial ao abrigo do programa GENIUS investigação da Inova-Ria, associação de empresas na área TICE, na empresa de acolhimento HFA.

A HFA - Henrique, Fernando e Alves, S.A. é uma PME especializada na assemblagem e teste de equipamento eletrónico e de telecomunicações, em regime de subcontratação.

O trabalho realizado consistiu na análise e sugestão de melhorias do processo de planeamento existente bem como da gestão do *picking*, integrando ferramentas *Lean* e TOC.

1.2 - Problema e metodologia

Um dos objetivos que nos propomos atingir é o aumento da satisfação dos Clientes através da diminuição dos tempos de entrega. A falta de um plano de produção global efetivo para o sequenciamento de tarefas traduz-se num esforço excessivo na decisão e em resultados com uma margem significativa de melhoria. Consequentemente, não há qualquer reflexão no longo termo, tomando-se decisões num ambiente incerto.

Inicialmente, a resolução da questão do sequenciamento consistiu na simulação de heurísticas progressivamente mais complexas. A sua eficácia foi comparada na prática, adaptando-as às restrições que foram surgindo. Posteriormente, os resultados obtidos foram

avaliados segundo diversos indicadores, dos quais de forma indireta: a taxa de cumprimento, a carga, os atrasos *backlog*[1], o volume de negócios e a satisfação do Cliente.

Após a comparação entre elas e a escolha de uma heurística que satisfaça os requisitos práticos e possa ser utilizada consistentemente, a próxima etapa passa por uma análise semelhante relativamente à TOC. Sendo os recursos humanos disponíveis uma importante restrição, é desenvolvido, também, um algoritmo para a gestão permanente da sua alocação.

Futuramente, esta ferramenta deve ser expandida com mais técnicas e variáveis aqui discutidas. Por um lado, a otimização e planeamento dinâmico em relação ao custo de posse de inventário e ao custo de *setup* seria feito através de algoritmos de dimensionamento de lotes. Por outro lado, é possível ainda ultrapassar os gargalos e outras restrições através de um novo *layout*. Ao organizar as pessoas em "ilhas de produção" nestes locais críticos, é concebível um algoritmo que ajuste o número de pessoas nas secções imediatamente antes e depois do estrangulamento.

1.3 - Organização da tese

A estrutura da tese segue a sequência lógica da metodologia apresentada.

- Capítulo 2 - Estabelece-se um enquadramento teórico nas áreas de conhecimento fundamentais, que sustentam as ferramentas postas em prática. Menciona-se também a bibliografia que serviu de inspiração e apoio para o desenvolvimento de novos algoritmos.
- Capítulo 3 - Apresenta-se o sistema de encomendas e o mapa do chão de fábrica. Além disso, é demonstrado o cálculo da matriz de carga e a identificação do gargalo.
- Capítulo 4 - Apresenta-se a metodologia de sequenciamento de tarefas e mostra-se o desenvolvimento do planeamento ao longo das semanas. É desenvolvido, ainda, um algoritmo teórico para a resolução do problema genérico de atrasos JSSP.
- Capítulo 5 - Desenvolve-se um programa baseado nas informações do capítulo anterior e discutem-se os resultados teóricos e práticos obtidos.
- Capítulo 6 - Apresenta-se as conclusões e as perspetivas de evolução futura.

[1] *Backlog*: Acumulação de encomendas pendentes inacabadas ou não atendidas.

Capítulo 2

Contexto da dissertação

Neste capítulo introduzem-se as ferramentas *Lean* e TOC. Além disso, define-se o problema job shop scheduling e propõe-se diversas abordagens heurísticas como COVERT, Moore-Hodgson, Grasp e Simulated Annealing.

2.1 - *Lean*

A metodologia de gestão da produção com maior impacto nos últimos 50 anos foi a *Lean* (Womack, Jones, & Roos, 1990). O sistema de produção *Lean* desenvolvido pela Toyota (TPS) foca-se em eliminar o desperdício, sendo que nada é produzido até ser necessário. O resultado é a melhor qualidade, o menor custo e o menor lead time.

Sakichi Toyoda, fundador do grupo Toyota, inventou o conceito de *Jidoka*. O *Jidoka* conta com quatro princípios para garantir a entrega de produtos sem defeitos e acompanhar o *takt time*:

1. Descoberta de uma não conformidade
2. Interrupção do processo
3. Resolução imediata do problema
4. Procura e correção da causa raiz

O *takt time* é o ritmo no qual é preciso completar um produto para suprir a procura do consumidor, conforme definido na equação (2.1) e define-se como:

$$Takt\ Time\ = \frac{\text{tempo total de produção disponível}}{\text{procura média do Cliente}} \tag{2.1}$$

Kiichiro Toyoda, filho de Sakichi, desenvolveu o conceito de JIT-*Just-in-Time* na década de 1930. O TPS é, de facto, construído sobre os dois pilares, JIT e *Jidoka*, que têm raízes no período pré-guerra. Estes são continuamente melhorados por interações entre um trabalho padronizado e *kaizen*.

Kaizen significa "mudar para melhor" e o seu ciclo de atividades pode ser definido através do ciclo PDCA (*Plan, Do, Check, Act*) conforme Figura 2.1. É comum planear (*Plan*) e executar (*Do*), mas a inovação surge na melhoria continua proporcionada pelas atividades verificar (*Check*) e agir (*Act*).

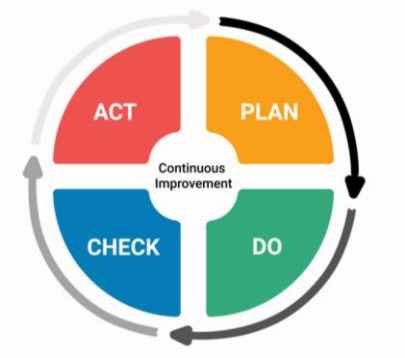

Figura 2.1 – PDCA

O desenvolvimento do TPS é atribuído a Taiichi Ohno, chefe de produção da Toyota no período posterior à Segunda Guerra Mundial. Este sistema é associado a duas filosofias fundamentais na cultura japonesa: eliminação de desperdícios e respeito pelas pessoas (Wantuck, 1983).

Figura 2.2 – Exemplo de planta industrial no início do sec. XX.

Fora do Japão, alguns destes conceitos remontam ao início dos anos 1900 nos Estados Unidos, como ilustrado na Figura 2.2. Henry Ford foi um dos seus pioneiros nas linhas de montagem de automóveis. Por exemplo, para eliminar desperdício, ele usava o fundo das caixas de embalagem dos assentos como tábuas para o chão dos carros.

Uma das ferramentas principais da produção *Lean* é o sistema *pull*. Quando um produto é vendido, em teoria, o mercado "puxa" uma substituição da última posição no sistema. De modo a permitir que este processo seja viável, a produção exige altos níveis de qualidade em cada etapa do processo, fortes relações com o fornecedor e uma procura previsível para o produto final. Este sistema *pull* é evidenciado na Figura 2.3 (o serviço *A* é composto pelos bens B e C).

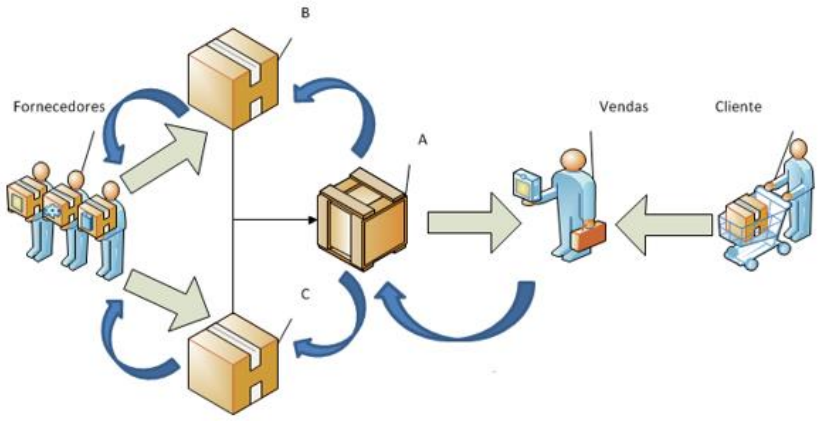

Figura 2.3 - Produção *Lean* - Sistema *Pull*.

Por outro lado, no sistema *push*, ou seja, o sistema de fabricação tradicional, a produção é processada numa data bem definida. O produto, a partir do instante de lançamento, fica sujeito a uma sequência de operações, e quando uma operação termina, o *output* é "empurrado" para a operação seguinte, independentemente de ser ou não necessário naquele momento. O plano de produção é elaborado com base na capacidade da fábrica e a produção é realizada com o intuito de ser vendida num futuro mais ou menos próximo. Além disso, quando a oferta é maior que a procura, existe a criação de *stock* e, por vezes, um abrandamento na produção, até se atingir um equilíbrio entre estas (Dias, 2005).

As principais diferenças entre o sistema *push* e o sistema *pull* estão presentes na tabela 2-1.

Tabela 2-1 — Diferenças entre *Push* e *Pull*

Push	***Pull***
Antecipação à procura	Resposta à procura
Produção para *stock*	Produção a pedido
Segmentação de mercado	Fragmentação do mercado
Grande número de Clientes com necessidades semelhantes	Número de Clientes reduzido e com necessidades distintas
Produtos genéricos	Produtos à medida
Séries de produção longas	Séries de produção curtas

Há ainda várias ferramentas do *Lean* consideradas, mas não explorados ao longo desta tese, das quais salientamos *Kaizen* (melhoria contínua), *Jidoka* (Automação), *Kanban* (JIT - *Just in Time*), VSM - *Value Stream Mapping* (mapeamento do fluxo do valor), SMED - *Single-Minute Exchange of* Dies, TPM - *Total Productive Maintenance*, 5S's-Seiri (Classificação), *Seiton* (Ordem), *Seiso* (Limpeza), *Seiketsu* (Padronização) e *Shitsuke* (Disciplina).

A ferramenta VSM em particular é já utilizada pela empresa onde decorreu a dissertação. Na Figura Anexo B.1 é exemplificado um caso real do seu uso numa das linhas de produção. Um mapa global do chão de fábrica é também apresentado no Anexo B.

2.2 - TOC – Teoria das Restrições

A TOC, teoria das restrições, é um paradigma de gestão introduzido por Goldratt[2] no livro *The Goal* (1984). As suas cinco etapas basilares são:

1. Identificar as restrições do sistema. (Qualquer progresso é impossível, a menos que se encontre a restrição ou o elo mais fraco.)
2. Decidir como explorar as restrições do sistema.
3. Subordinar tudo a essa decisão. (Alinhar todas as outras partes do sistema de modo a suportar as restrições, mesmo que isso reduza a eficiência dos recursos não restritivos.)
4. Anular as restrições do sistema. (Se a produção ainda for inadequada, pode-se, por exemplo, duplicar o recurso ou dividir a operação, para que não seja mais uma restrição.)
5. Se, nas etapas anteriores, as restrições foram ultrapassadas, então voltar à etapa 1, mas nunca deixar que a inércia se torne a restrição do sistema. (Este é um processo de melhoria contínua: identificar restrições, removê-las e, depois, tornar a identificar as novas.)

Subjacente ao trabalho de Goldratt, está a noção de **fabricação síncrona**, ou seja, o processo de produção deve funcionar em harmonia para se alcançar o objetivo da empresa.

Goldratt argumenta que, embora uma organização possa ter vários propósitos - aumentar a empregabilidade, aumentar as vendas e a influência no mercado, desenvolver tecnologia ou produzir produtos de elevada qualidade - nenhum garante perspetivas positivas a longo termo. Estes são meios de se atingir um **fim.** O fim (objetivo) de uma empresa é ter lucro. Quando uma empresa ganha dinheiro, prospera e pode-se focar noutros vertentes da sua atuação.

[2] Esta Secção é baseada nos ensinamentos do Dr.Eliyahu M.Goldratt. Goldratt fundou o instituto Avraham Y.Goldratt e foi autor em particular do livro *Critical Chain* (1997), que serviu de especial inspiração.

Tipicamente, a **produtividade**[3] pode ser medida em termos de produção por hora de trabalho. Contudo, esta métrica não é suficiente, por exemplo, quando a produção adicional é acumulada em *stock*. Assim, a produtividade deve ser definida como todas as ações que levam uma empresa a ficar mais perto dos seus objetivos.

A ênfase está no desempenho total do sistema e não em métricas locais como a utilização de máquinas. Na fabricação síncrona, balancear a capacidade[4] é visto como uma má decisão. Este equilíbrio só seria possível se os tempos de *output* em todas as estações fossem constantes ou tivessem uma curva de distribuição muito estreita. Uma distribuição normal destes tempos causa que as estações a jusante tenham *idle time* quando as a montante demoram mais tempo na produção. O efeito da variação estatística é cumulativo. Por isso, devem ser realizadas tentativas para equilibrar o fluxo do produto através do sistema; quando o fluxo está equilibrado, as capacidades estão desequilibradas.

2.2.1 - *Drum, Buffer, Rope*

O sistema de produção precisa de um ponto ou pontos para controlar o fluxo do produto. Caso o sistema contenha um **gargalo,** restrição do sistema que limita o *throughput*[5] (TH), esse será o melhor ponto para o controlar. Na Figura 2.4, este é denominado de tambor ou *drum*, porque marca o ritmo que o resto do sistema segue para funcionar. O gargalo sendo um recurso que não tem capacidade para responder à procura, funciona, idealmente, de forma ininterrupta e o seu uso como ponto de controlo garante que as operações a montante não produzem em excesso e acumulem trabalho em curso (**WIP**) não suportado.

Por exemplo, a produção atual de carros elétricos é limitada pela fabricação de baterias. O tambor é todo o processo envolvido com estas, na fabricação. O produto final segue o seu ritmo. Assim, as baterias são o alvo de maior esforço económico para se aumentar a cadência associada ao seu processo e tornar, futuramente, o seu custo mais viável.

Se não houver gargalo, o melhor lugar para ser definido como tambor seria um recurso de capacidade limitada (CCR - *Capacity Constraint Resource*). Um **CCR** é um recurso que opera perto da capacidade limite, mas, em média, tem capacidade adequada, desde que não seja sequenciado incorretamente.

[3] Aumentar a produtividade pode ser definido como aumentar a quantidade de tarefas entregues enquanto simultaneamente se reduz o inventário e os gastos com as operações.

[4] Capacidade é o tempo disponível para produção. Em vez das capacidades, deve ser balanceado o fluxo do produto através do sistema.

[5] O *thoughput* representa a quantidade de tarefas concluídas num determinado período de tempo. TH = WIP / Lead Time.

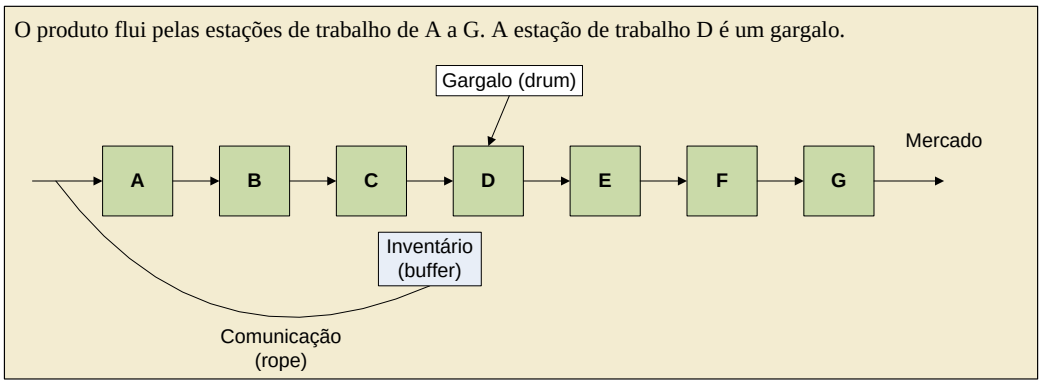

Figura 2.4 - Fluxo Linear com um Gargalo

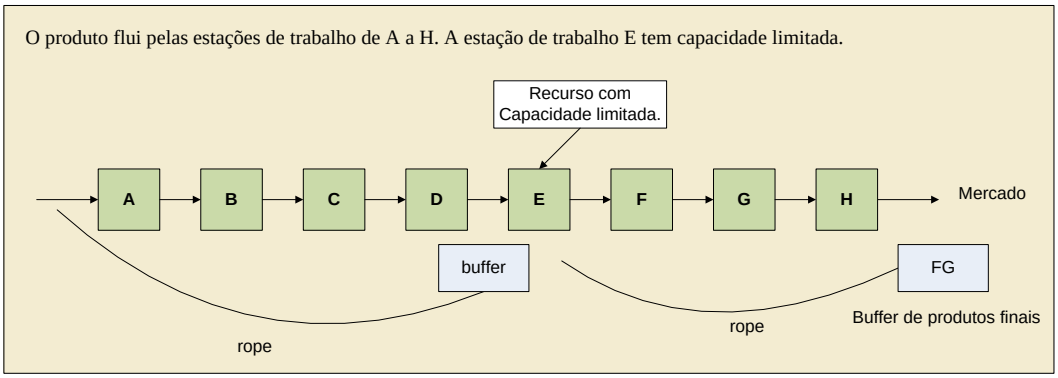

Figura 2.5 - Fluxo Linear com um Recurso de Capacidade Limitada

O *buffer* de WIP à frente de um gargalo é um time *buffer*[6]. Assim, pretende-se certificar que este está sempre em funcionamento, independentemente da ordem em que os produtos foram planeados. Na Figura 2.5, o inventário de produtos acabados protege o mercado, e o *time buffer* à frente do CCR protege o *throughtput*. O propósito da corda ou *rope* é servir de comunicação entre as várias zonas. Assim, no segundo caso, são precisas duas cordas: (1) uma corda que comunica do inventário de produtos acabados de volta ao tambor para aumentar ou diminuir o output e (2) uma corda desde o tambor ao ponto inicial, especificando quanto material é necessário.

[6] *Time Buffer*, deve ser o quão grande for preciso de modo a garantir que o gargalo continue a trabalhar.

2.2.2 - Comparar Fabricação Síncrona com MRP e JIT

O MRP - planeamento das necessidades materiais, agenda a produção de forma regressiva, *backward scheduling*, depois de lhe ser fornecido o "agendamento mestre" desta. Funciona retrocedendo no tempo a partir da data de conclusão desejada. Como procedimento secundário, o MRP, através do seu módulo de planeamento, desenvolve perfis de capacidade dos centros de trabalho. Quando estes centros estão sobrecarregados, há apenas duas soluções: ou o agendamento mestre da produção deve ser ajustado, ou deve haver já programada folga de capacidade suficiente para que as tarefas possam ser suavizadas a nível local.

Uma abordagem por fabricação síncrona utiliza o princípio contrário, *forward scheduling*, porque se foca nos recursos críticos. Este agendamento assegura que há capacidade para as cargas colocadas. Todos os recursos que não são restrições ou gargalos são planeados de modo a suportar os mais críticos. Este procedimento possibilita um agendamento viável. Para ajudar a reduzir o tempo de espera e o WIP, o tamanho dos lotes[7] é variado - um procedimento que o MRP não é capaz.

Comparativamente à fabricação síncrona, o JIT tem um desempenho excelente na redução de prazos de entrega e do trabalho em processo (WIP), mas tem várias desvantagens. De facto, não só é limitado a uma fabricação repetitiva, que requer uma produção estável por um período significativo, como não é flexível aos produtos produzidos, ou seja, os produtos devem ser semelhantes com um número de opções limitado. Além disso, o WIP continua a ser um requisito quando usado com *kanban*[8] para que haja algo para "puxar". Isto significa que as tarefas completas devem ser armazenadas a jusante de cada estação de trabalho a ser "puxado" pelo posterior. Por fim, os vendedores devem estar situados num local próximo, porque o sistema depende de entregas pequenas e mais frequentes.

Como a fabricação síncrona usa um agendamento para atribuir as tarefas a cada estação de trabalho, não há necessidade de mais WIP para além do trabalho a já ser processado. A exceção é para *stock* especificamente colocado à frente de um gargalo de modo a assegurar o trabalho contínuo, ou em pontos específicos a jusante deste para garantir o fluxo do produto.

Em relação às melhorias contínuas do sistema, o JIT é um procedimento de tentativa e erro aplicado a um sistema real. Na fabricação síncrona, o sistema pode ser programado e simulado, uma vez que o agendamento é realizável e o tempo de execução por um computador é curto.

[7] Tamanho do Lote: Numa linha de montagem, o tamanho do lote, *transfer batch*, é "um", porque uma unidade é movida de cada vez, e "infinito", *process batch*, porque a linha produz continuamente o mesmo produto.

[8] Kanban: É um "cartão" de sinalização que controla os fluxos de produção, permite limitar o WIP e assegura políticas explicitas e melhoria colaborativa

2.3 - Sequenciamento

Há uma série de tentativas de pesquisa relativamente aos problemas tradicionais de sequenciamento. Os primeiros esforços remontam ao final dos anos 1970, conforme mostrado na pesquisa bibliográfica *Sequencing* (Baker & Scudder, 1990). O conjunto destes problemas cuja solução considera como critério a minimização de atrasos foi introduzido por Kanet (1981) e Sundararaghavan (1984) que mostraram que o agendamento pelo algoritmo de Kanet é viável para prazos de entrega "restritivamente urgentes". Hall (1991) caracterizou ainda as propriedades ótimas de fluxo para o problema de Kanet e provou que este é *NP-complete*[9].

Por outro lado, algumas metodologias abordaram o processamento por lote de tarefas. Ahmadi (1992) investigou uma classe de problemas com pelo menos um lote por máquina. Não obstante, realizou uma análise de complexidade para duas situações, tendo como métricas o tempo médio de fluxo e o *makespan*, tempo máximo de conclusão entre todas as tarefas, mas sem considerar o critério de atraso. Posteriormente, Mosheiov (2003) definiu o problema com m máquinas sob a condição das tarefas terem o mesmo tempo de processamento e uma determinada data de entrega em comum. Para isso, considerou o objetivo JIT de minimizar o custo máximo de atraso. Assim, introduziu uma abordagem com solução em tempo polinomial para o problema proposto.

Karimi (1992) propôs uma formulação de programação não linear inteira de sistemas de produção em série de múltiplas etapas sob procura constante e horizonte infinito. Estas etapas de produção operam com interrupções e arranques periódicos. Mostrou, então, que o algoritmo *branch-and-bound - state space search*[10], talvez a ferramenta mais utilizada na resolução de otimização de problemas ***NP-hard***, é a melhor abordagem para resolver o problema proposto. Faaland e Schmitt (1987) desenvolveram uma abordagem aproximada que minimiza as penalidades por atraso nos processos de fabricação e montagem. Através de uma simulação, mostraram que ocorrem melhorias no custo à medida que a complexidade do procedimento de programação aumentava.

O problema não resolvido da ciência da computação, P versus NP, é um dos sete problemas do milénio selecionados pelo *Clay Mathematics Institute*. A questão a responder é se cada problema cuja solução pode ser verificada rapidamente também pode ser resolvido rapidamente. O termo rapidamente significa em tempo polinomial "P". Além de ser um problema importante na teoria computacional, uma prova em qualquer sentido teria profundas implicações para a matemática, criptografia, pesquisa de algoritmos, inteligência artificial, teoria dos jogos, processamento de multimédia, filosofia, economia entre outros campos.

[9] *NP-complete*, embora uma solução para um problema *NP-complete* possa ser verificada "rapidamente", não há maneira conhecida de a encontrar em tempo polinomial.

[10] *Branch and Bound*, o algoritmo consiste na enumeração sistemática de candidatos a solução através de *space state search* e na exploração dos ramos da árvore que representam os subconjuntos do conjunto de soluções. Antes de serem enumerados, o ramo é comparado às estimativas para os limites superior e inferior e descartado se não for encontrada uma solução melhor que a atual.

A Figura 2.6 mostra os conjuntos de problemas através do Diagrama de Euler para cada uma das soluções. Além disso, o gráfico mostra o tempo [ms], usando um processador Pentium III de 933 MHz, vs o tamanho de problemas tipo mochila referidos na Secção 2.3.3. Uma regressão quadrática sugere que a complexidade do algoritmo nesta magnitude seja $O(\log(n))^2$ (Pisinger, 2003).

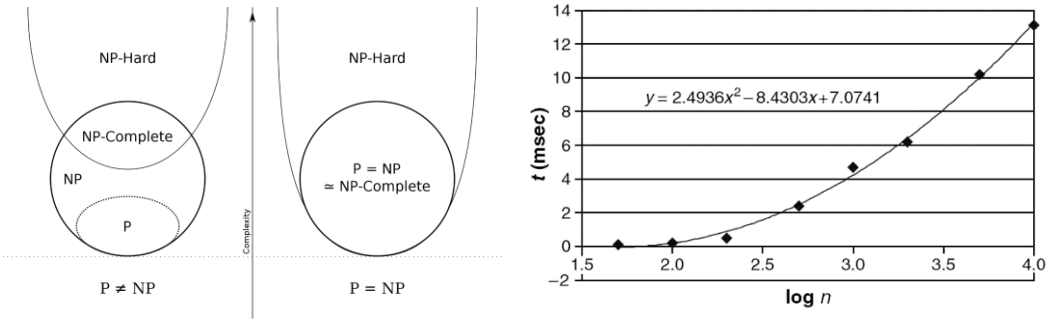

Figura 2.6 – Diagrama de Euler – Tempo execução

2.3.1 - JSSP – Problema de sequenciamento

O JSSP - *Job Shop Scheduling Problem*, é um problema *NP-hard* definido por um conjunto de tarefas que devem ser executadas por um conjunto de máquinas numa ordem específica. Cada tarefa tem um tempo de execução definido em cada máquina, sendo que cada máquina não pode executar mais do que uma tarefa ao mesmo tempo. Uma vez iniciada, a máquina não pode ser interrompida até à conclusão da tarefa **atribuída**[11]. O objetivo é minimizar o *makespan*.

As principais dificuldades do planeamento da produção na empresa HFA advêm da complexidade do sequenciamento das tarefas, que estão na raiz do problema JSSP. É por isso, um dos principais desafios que justifica o uso de heurísticas entre outras abordagens.

Suponha-se que temos três tarefas que são processadas cada uma numa ordem específica em três máquinas. O agendamento na Figura 2.7 mostra uma solução simples: as tarefas são processadas sequencialmente e as máquinas ficam frequentemente com *idle time*. A segunda solução apresentada na Figura 2.8 é a ótima.

[11] Ao sequenciar-se com recurso a uma heurística, a tarefa não poderá ser trocada por outra devido à verificação, agora, de uma qualquer condição. Todavia, no contexto prático, isto poderá acontecer em casos particulares onde há disponibilidade de recursos e tempos de *setup* reduzidos.

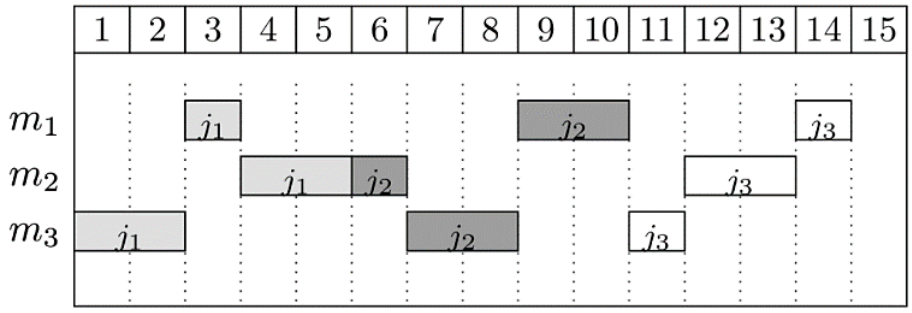

Figura 2.7 – Tarefas processadas sequencialmente

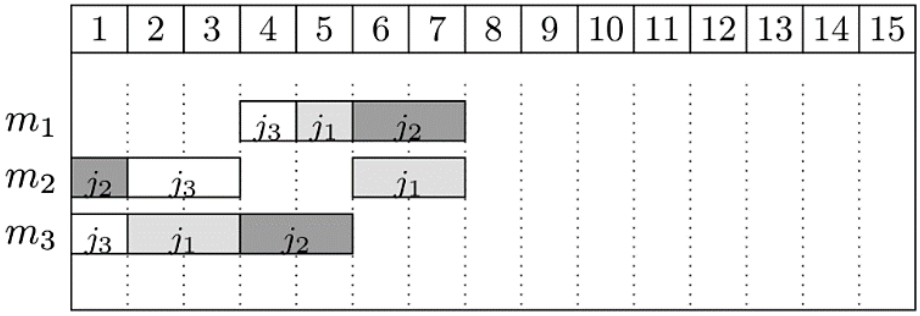

Figura 2.8 – Solução Ótima

Uma formulação MIP - *Mixed-Integer Linear Program* para o JSSP e as suas considerações, respetivas variáveis, restrições e função objetivo serão apresentadas de seguida (Santos & Toffolo, 2019).

Variáveis e considerações

Dados necessários:

- J conjunto de tarefas, $J = \{1, \ldots, n\}$
- M conjunto de máquinas, $M = \{1, \ldots, m\}$
- o_r^j máquina que processa a r-ésima tarefa j pela ordem $O^j = (o_1^j, o_2^j, \ldots, o_m^j)$
- p_{ij} tempo de processamento da tarefa j na máquina i, $p_{ij} \in \mathbb{Z}^+$

A solução deve respeitar as seguintes **restrições**:

- Todas as tarefas j são processadas pela sequência de máquinas definida por O^j.
- Cada máquina processa uma e só uma tarefa de cada vez.
- Quando uma máquina inicia uma tarefa, esta tem de ser completada sem interrupções.

As **variáveis de decisão** são definidas por:

- x_{ij} tempo de começo da tarefa $j \in J$ na máquina $i \in M$.
- y_{ijk} é 1, se a tarefa j precede a tarefa k na máquina i $(i \in M; j, k \in J, j \neq k)$ e 0, caso contrário

A **função objetivo** é calculada na variável auxiliar C correspondente ao *makespan*. O primeiro conjunto de restrições (2.1) são as restrições de precedência, que garantem que uma tarefa numa máquina só arranque após o processamento na anterior estar concluído. O

segundo e o terceiro conjuntos de restrições disjuntivas (2.2) e (2.3) garantem que apenas uma tarefa seja processada num determinado momento ou máquina. A constante M deve ser grande o suficiente para que estas restrições se verifiquem. Uma estimativa válida, mas fraca, para este valor pode ser a soma de todos os tempos de processamento. O quarto conjunto de restrições (2.4) garante que o valor referente ao $makespan$ seja calculado corretamente e as últimas restrições (2.5), (2.6) e (2.7) indicam domínios variáveis.

Restrições

$$x_{o_{rj}^{j}} \geq x_{o_{r-1j}^{j}} + p_{o_{r-1j}^{j}} \; \forall r \in \{2,\ldots,m\}, j \in J \tag{2.2}$$

$$x_{ij} \geq x_{ik} + p_{ik} - M.y_{ik} \forall j,k \in J, j \neq k, i \in M\, J \tag{2.3}$$

$$x_{ik} \geq x_{ij} + p_{ij} - M.(1 - y_{ikj}) \forall j,k \in J, j \neq k, i \in M \tag{2.4}$$

$$C \geq x_{o_{mj}^{j}} + p_{o_{mj}^{j}} \; \forall j \in J \tag{2.5}$$

$$x_{ij} \geq 0 \; \forall i \in J, i \in M \tag{2.6}$$

$$y_{ijk} \in \{0,1\} \forall j,k \in J, i \in M \tag{2.7}$$

$$C \geq 0 \tag{2.8}$$

Minimizar C**.**

2.3.2 - Heurísticas - COVERT

As abordagens heurísticas atualmente disponíveis para sequenciar tarefas podem ser classificadas como construtivas ou de melhoria. Uma abordagem heurística construtiva cria uma sequência de tarefas de forma que, uma vez construído o agendamento, este não possa ser revertido. Por outro lado, uma abordagem heurística de melhoria começa com qualquer sequência de tarefas e, em seguida, tenta melhorar a solução modificando a sequência de forma iterativa. A última abordagem tem mostrado ser superior em termos de desempenho. Diversas abordagens heurísticas construtivas para o problema de sequenciamento foram propostas por Armentano (1999) e Etiler (2004). Alguns algoritmos genéricos de solução exata também foram discutidos por Yeh e Allahverdi (2006).

Na Secção 4.5.2 é aplicado um algoritmo baseado no conceito COVERT[12]. Este conceito foi proposto em diversos estudos para problemas com apenas uma máquina. Entre esses estudos está o trabalho de Carroll (1965), que apresentou uma regra dinâmica chamada COVERT para uma única máquina e problemas de sequenciamento para minimizar o atraso total ponderado das tarefas. Morton e Rachamadugu (1983) introduziram uma regra chamada custo de atraso aparente (ATC) para o problema anterior. Mais tarde, Rachamadugu (1984) conduziu uma simulação para comparar a regra ATC com algoritmos baseados em COVERT. A experiência mostrou algumas situações em que o ATC resultou num fluxo melhor do que o COVERT. Vepsalainen e Morton (1987) chegaram à mesma conclusão quando compararam várias regras de sequenciamento baseadas nas de ATC e COVERT durante a resolução de problemas de sequenciamento. Os problemas tiveram em conta diversos prazos e cargas de trabalho.

[12] COVERT, heurística que se refere ao custo ao longo do tempo.

Concluíram, assim, que o ATC teve o melhor desempenho em todos estes casos e o COVERT teve o segundo melhor entre todas as outras regras consideradas. No entanto, em ambos os estudos, não foi avaliada a capacidade do COVERT em encontrar um agendamento viável para problemas de fluxo da produção. De facto, os estudos conduzidos por Baker (1984) promovem a eficácia das regras orientadas para as tarefas e as experiências preliminares de Ghassemi-Tari e Olfat (2004) sugerem, ainda, que COVERT é um conceito potencialmente promissor para o desenvolvimento de algoritmos heurísticos.

Existem também algumas outras classes de abordagens heurísticas, como o algoritmo de Campbell, Dudek e Smith (CDS)[13] (Johnson generalizado para $m > 2$ máquinas), pesquisa Tabu (Armentano & Ronconi, 1999), pesquisa Vizinha (Kim, 1993) e inteligência artificial (Lee, 2001).

2.3.3 - Moore-Hodgson

O algoritmo **Moore-Hodgson**, tal como as heurísticas SPT e EDD exploradas no Capítulo 4.2, é aplicado a uma classe de problemas com n tarefas processadas numa só máquina. Mesmo em problemas complexos de $m > 2$ máquinas, a sua utilização tem interesse por contemplar restrições existentes na prática, tais como o maior peso atribuído a um dado Cliente, ou uma máquina que apenas processa tarefas para este.

No entanto, ao contrário das outras heurísticas, a sua aplicação genérica não é tão direta. O algoritmo Moore-Hodgson pode ser aplicado a um problema com formulação do tipo *knapsack* (mochila), ilustrado na Figura 2.9. O problema da mochila remonta aos primeiros trabalhos do matemático Dantzig Tobias (1930) e refere-se ao típico problema de se guardar objetos sem se exceder a capacidade permitida. O problema da mochila é um problema de otimização combinatória, surge geralmente no contexto da alocação de recursos[14], onde se opta, por exemplo, entre um conjunto de tarefas sob uma restrição no tempo. O problema da mochila pode ser unidimensional, onde cada caixa teria um determinado valor ou multidimensional onde se poderia considerar tanto a massa como o volume das caixas.

[13] CDS: O método consiste em resolver "m-1" subproblemas com n tarefas e duas máquinas, utilizando o algoritmo de Johnson. O subproblema k consiste em considerar a soma dos tempos nas k primeiras máquinas e a soma dos tempos nas k últimas máquinas (k=1, 2, ... , m-1). A melhor sequência dentre as "m-1" geradas é considerada como solução do problema original.

[14] Alocação de Recursos, na gestão da produção, representa o sequenciamento de tarefas e dos recursos exigidos por estas. Tem em consideração a disponibilidade dos recursos e o tempo do projeto.

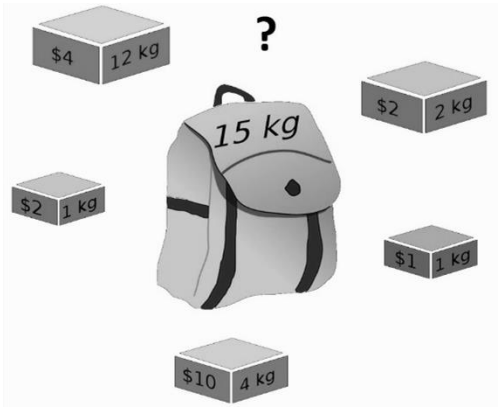

Figura 2.9 – Problema da mochila.

Suponha-se que haja n tarefas, cada uma com um tempo de processamento, data de partida, data de entrega ao Cliente e um peso particular atribuído. O objetivo é sequenciar as tarefas para processamento numa única máquina para que o peso total das tarefas atrasadas seja minimizado. O algoritmo de Moore-Hodgson resolve este problema em tempo $O(n\ log\ (n)\ )$[15] quando todas as datas de partida são iguais, desde que os tempos de processamento e os pesos das tarefas estejam de certa forma em concordância, Lawler (1994).

Procedimento

1. ordenar as tarefas em ordem crescente de data de entrega: dj ↑;
2. iniciar com a tarefa agendada $J_0 = \emptyset$, e λ = 0;
3. para $j = 1,\ldots,n$, se $\lambda + p_j \leq d_j$, então $J_j = J_{j-1} \cup \{j\}$; $\lambda = \lambda + p_j$; em caso contrário, deixar $j_{max} \in J_{j-1} \cup \{j\}$ ter o maior tempo de processamento; fazer $J_j = J_{j-1} \cup \{j\} \setminus \{j_{max}\}$; $\lambda = \lambda + p_j - p_{jmax}$
4. Sequenciar as tarefas em $= J_n$ pela ordem da data de entrega; sequenciar as restantes tarefas depois das em J_n por qualquer ordem.

[15] *O notation*, é uma notação matemática que descreve o limite de uma função quando o seu argumento tende para um determinado valor ou infinito. Na ciência da computação, é usada para classificar algoritmos de acordo com o aumento do tempo de execução ou de requisitos de espaço conforme o tamanho à entrada.

2.3.4 - GRASP

O GRASP, G*reedy Randomized Adaptive Search Procedure*, é uma meta-heurística que consiste em iterações a partir de sucessivas soluções *greedy* e melhorias por pesquisa local. Foi primeiro introduzido por Feo e Resende (1989). Um algoritmo *greedy* geralmente não produz uma solução ótima, mas produz máximos ou mínimos locais que se aproximam, no longo termo, de uma boa solução global para vários problemas.

Em particular, se um problema de otimização tem a estrutura de um matróide, então o algoritmo *greedy* apropriado irá resolvê-lo de forma ótima, (Papadimitriou & Steiglitz, 1982). Um matróide é uma estrutura matemática que generaliza a noção de independência linear de espaços vetoriais para conjuntos arbitrários. Outro exemplo da sua utilização seria na determinação do número mínimo de moedas a dar ao fazer o troco. De facto, a maioria dos sistemas monetários, incluindo o euro e o dólar americano, são casos especiais em que a estratégia *greedy* encontra sempre uma solução ótima.

Relativamente ao JSSP, o GRASP é uma abordagem particularmente viável para o reagendamento de problemas em ambientes dinâmicos (A Baykasoğlu, 2017). Na realidade, existem muitos eventos dinâmicos como a chegada de novas encomendas, avarias nas máquinas, mudanças nas datas de entrega, cancelamentos de pedidos, chegada de encomendas urgentes, entre outros, que limitam as abordagens de programação estática. Por outro lado, a decisão da aceitação/rejeição de um determinado pedido do Cliente deve ser integrada na decisão de programação para atender aos requisitos das datas de entrega, enquanto se executam ajustes de capacidade.

Assim, será proposta uma estratégia de reagendamento periódico que procura tornar a programação mais flexível e comprometer-se, simultaneamente, com um plano no longo termo. Além disso, a ferramenta será programada de modo a poder ser expandida e dinamizada no futuro por algoritmos como o GRASP.

2.3.5 - SA – Recozimento Simulado

SA, *Simulated Annealing* é um algoritmo *greedy* baseado no comportamento do metal em arrefecimento e é muito semelhante à pesquisa Tabu. O procedimento é otimizar uma função de energia (custo) procurando estocasticamente por mínimos em diferentes temperaturas através do método MCMC, *Markov Chain Monte Carlo*. A procura é estocástica, porque embora se aceite sempre um novo estado S′ com menor energia ($\Delta E < 0$), só se aceita um novo estado com maior energia ($\Delta E > 0$) com uma determinada probabilidade.

$$P(S \rightarrow S') = min\{1, e^{(-\Delta E/T)}\} \quad (2.9)$$

$$\Delta E = E(S') - E(S) \quad (2.10)$$

A distribuição de Boltzmann $P(c) = e^{(-\Delta E(c)/T)}/Z$ é usada para calcular as probabilidades para cada estado, onde T é uma "temperatura sintética" e Z representa a função partição.

- Se $T \to \infty$ tem-se uma distribuição uniforme onde todos os estados são igualmente prováveis.
- Se $T \to 0$ tem-se uma função delta de Dirac em torno do ótimo global.

Ao iniciar-se com um T elevado e diminuí-lo gradualmente, consegue-se uma amostra com estados suficientes e ao mesmo tempo estados energéticos mais elevados, que permitem escapar aos mínimos locais no percurso até ao ótimo global. Este processo é ilustrado na Figura 2.10:

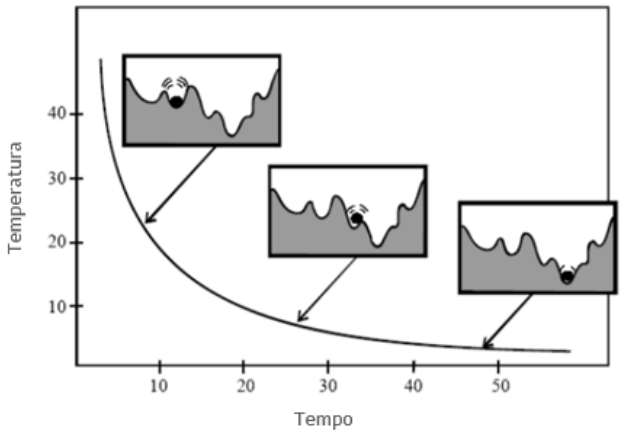

Figura 2.10 – Método SA.

Procedimento

- Fazer $s = s_0$
- Para $k = 0$ até k_{max} (exclusivo):
    - $T \leftarrow$ temperatura $((k+1)/k_{max})$
    - Escolher um vizinho aleatório, $s_{new} \leftarrow$ *neighbour*(s)
    - Se $P(E(s), \mathrm{E}(s_{new}), \mathrm{T})) \geq$ *random* $(0, 1)$:
        - $s \leftarrow s_{new}$
- Output: o estado final s

De modo a aplicar-se este método a um problema em particular, no nosso caso o JSSP é preciso especificar:

1. o espaço de estados
2. uma função objetivo energia E( )
3. um procedimento para gerar candidatos *neighbour*( )
4. uma função probabilística para a aceitação dos estados P( )
5. um agendamento T( )
6. uma "temperatura" inicial

Estas escolhas têm um impacto significativo na eficácia do método.

No artigo (Chakraborty & Bhowmik, 2013) encontra-se a solução ótima de um problema de referência de "*Resource Constrained Project Scheduling*" (Lawrence, 1984) através de SA. Este problema considera 10 tarefas em 5 máquinas. Os tempos de processamento associados encontram-se na tabela 2.2:

Tabela 2-2 — Problema LA01

Job	Machine (Processing Time)				
0	1 (21)	0 (53)	4 (95)	3 (55)	2 (34)
1	0 (21)	3 (52)	4 (16)	2 (26)	1 (71)
2	3 (39)	4 (98)	1 (42)	2 (31)	0 (12)
3	1 (77)	0 (55)	4 (79)	2 (66)	3 (77)
4	0 (83)	3 (34)	2 (64)	1 (19)	4 (37)
5	1 (54)	2 (43)	4 (79)	0 (92)	3 (62)
6	3 (69)	4 (77)	1 (87)	2 (87)	0 (93)
7	2 (38)	0 (60)	1 (41)	3 (24)	4 (83)
8	3 (17)	1 (49)	4 (25)	0 (44)	2 (98)
9	4 (77)	3 (79)	2 (43)	1 (75)	0 (96)

A temperatura inicial elevada escolhida foi de 200. Geraram-se 300 soluções de combinações que correspondem à cadeia de sequenciação das tarefas. A solução ótima tem um *makespan* de 667, presente em duas combinações: "9-6-8-3-4-1-0-5-2-7" e "3-6-8-9-4-1-0-5-2-7". A simulação das iterações que levaram ao *makespan* ótimo encontra-se ilustrada na Figura 2.10.

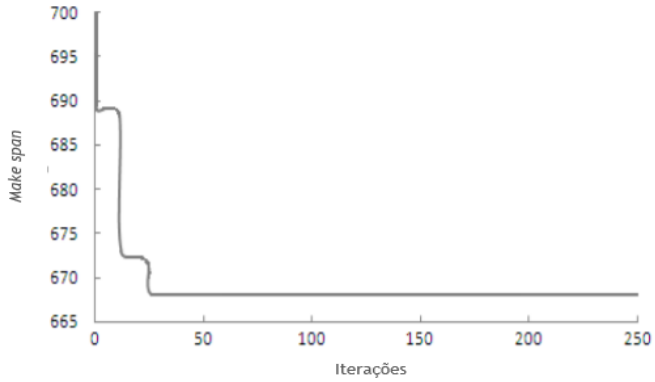

Figura 2.11 – Simulação do método SA para o problema LA01.

Capítulo 3

Encomendas e Produção

Neste capítulo apresentam-se as etapas de produção e explica-se o modo como a evolução das encomendas **em aberto** influencia o fluxo, ou seja, o tempo que uma tarefa gasta no processo. Neste contexto, identifica-se, ainda, o **gargalo** ou ponto de estrangulamento na produção, relacionando-o com o **equipamento**, a **mão-de-obra**, as **linhas de produção** e os **Clientes**.

3.1 - Planta da Fábrica

A HFA assembla e testa equipamento eletrónico e de telecomunicações. O mapa do chão de fábrica pode ser visualizado no Anexo A.

Existem sete linhas de produção para o conjunto de operações iniciais SMT - *Surface Mounted Technology*. Esta fase consiste em montar em superfície, com a inserção (posicionamento e soldadura) de componentes eletrónicos numa placa de circuito impresso (PCB). Frequentemente, há também a inspeção das faces "TOP" e "BOT".

A fase principal seguinte, THT - *Through-Hole Technology*, envolve o uso de condutores nos componentes que são inseridos por orifícios perfurados nas PCB e soldados. Em particular, a soldadura pode ser relativa a três processos distintos: soldadura manual, soldadura seletiva e soldadura por onda.

Por fim, a maioria das restantes operações são os testes, cortes, embalamento, envernizamento e montagem processados em etapas intermédias, ou numa fase posterior.

3.2 - Encomendas abertas

Uma encomenda aberta é aquela em que a quantidade encomendada pelo Cliente ainda não foi satisfeita/produzida na totalidade. As encomendas abertas podem permanecer no sistema, no limite, durante vários anos. Uma encomenda que subsiste um período de tempo significativamente maior que a data prevista é o resultado de, por exemplo, **não conformidades** no processo de produção. Embora o próprio processo admita uma taxa de

rejeição, podem surgir obstáculos nos processos iterativos de reparação representados de seguida. Estes obstáculos estão relacionados com o processo de fabrico e controlo de qualidade.

3.3 - Etapas da produção

É apresentado na Figura 3.1 um fluxograma das encomendas (abertas) e das etapas de produção. O processo de produção é destacado em pormenor como subprocesso das encomendas.

Relativamente à conceptualização das encomendas abertas, após a aceitação do pedido, é necessário definir-se a data prevista de entrega ao Cliente, ou múltiplas datas. O CRP – *Capability Resource Planning*, automatiza este processo, e procede-se à ordem de produção, após a sua confirmação. Note-se que este processo MTO - *Make to Order*[16] permite reduzir o inventário e aumentar a flexibilidade de resposta à grande variedade de produtos fabricados e SKU - S*tock Keeping Units*, existentes.

De outra forma, o mapeamento das etapas de produção permite identificar aqueles processos iterativos críticos. Estes processos excecionais são em primeiro, Reparação SMT, poderá demorar 2 semanas; o segundo, Reparação THT, até 3 meses; e o terceiro, o mais complexo, por não se conseguir identificar a causa do problema, levará, no extremo, a situações de anos. Este último é, por isso, a restrição, neste contexto, mais importante.

[16] MTO - *Make to Order*, começa apenas após um pedido do Cliente ser recebido. Permite maior flexibilidade, customização, redução de *stock* e desperdício em geral, mas traz um aumento de custos na produção e do tempo de espera. O MTO pode ser contrastado com o MTS - *Make to Stock*.

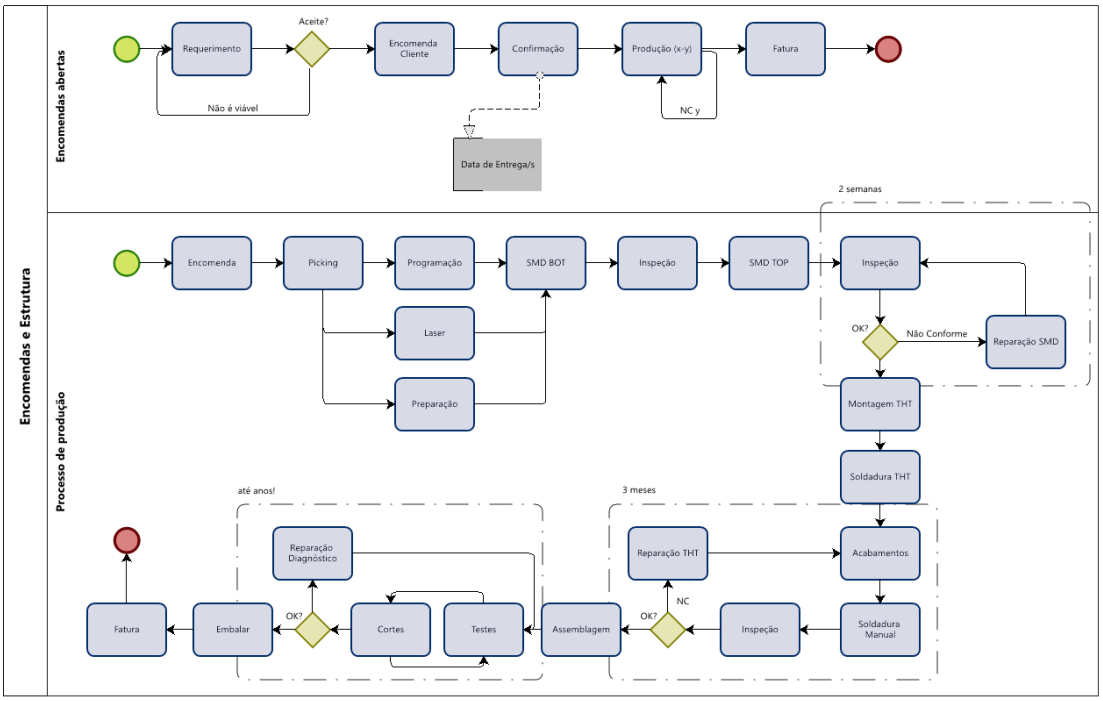

Figura 3.1 – Etapas de produção

3.4 - Matriz Carga

A recolha de dados referentes às encomendas abertas e às ordens de produção foi possível através da consulta do *software* da CentralGest.

As ordens de produção são relativas a um período recente e ao mesmo tempo extenso, últimos seis meses, para que manifestem o presente com variabilidade limitada. Estas identificam, para cada encomenda/referência, a sua respetiva cadência - quantidade produzida por unidade de tempo.

As encomendas correspondem às que estão em aberto até ao presente mais um período estabelecido em três semanas. Estas foram recolhidas diariamente de modo a observar-se a sua evolução de entrada e saída do sistema. As informações que facultam são a quantidade a produzir para cada referência resultante da encomenda do Cliente e a respetiva data de entrega.

Assim, o cruzamento de ambos os dados permite construir uma matriz carga cujos termos individuais são os tempos, em horas, de cada operação para cada referência em comum. Uma amostra de dez, de entre as cerca de cem referências da matriz, e respetivas operações principais são apresentadas na Tabela 3-1.

Conclui-se que, apenas na análise isolada das encomendas e da produção, o tempo de ocupação total maior, ou seja, o gargalo, corresponde ao processo de soldadura THT. Contudo, como se verá de seguida, a análise não está encerrada. Na verdade, é importante percorrer as operações intrínsecas ao gargalo e explicitar os pressupostos realizados.

Tabela 3-1 — Matriz Carga

	#Ref_P							
Carga/Operação[h]	57513220	57513573	538503011	57513360	57513681	57513683	…	**Σ [h]**
Embalar	0,03		14,33	425,90			…	1192
Cortar			14,33	70,98			…	541
Soldadura Manual							…	86
SMD - Top		1,89	10,75		1,45	0,56	…	287
SMD - Botton		1,70			2,29	0,83	…	102
Inspeção SMD TOP		0,28	35,82		0,13	0,04	…	526
Testar			21,49	425,90			…	1236
Envernizamento			35,82				…	440
Soldadura THT	0,22		39,44	283,93			…	1465
Oficina			75,42				…	402
Montagem DIP				354,92			…	1090
Programação				638,85			…	1296

3.5 - Pressupostos

Durante a realização do cálculo da matriz de carga através da análise dos dados, adotaram-se, ainda, as seguintes considerações:

- Utilizaram-se apenas os dados das operações cujos equipamentos associados são conhecidos.
- Atribuiu-se um peso, em termos de tempo, de 50% (carga distribuída igualmente) a cada uma das operações soldadura seletiva e soldadura por onda quando é indicada a utilização de ambas, mas não a cadência[17] da respetiva operação.
- Os dados apresentados foram recolhidos a 09/02/2021 e referem-se às encomendas abertas desde esta data até mais três semanas.
- Todos os dados estão em concordância com os consultados através da CentralGest.
- Os tempos do bloco “SMT” não incluem aqueles cujas linhas se destinam a Clientes específicos.

[17] Cadência, taxa de produção, medida por um determinado período de tempo de uma máquina, posto de trabalho ou qualquer sistema. Pode dizer respeito aos valores esperados ou aos valores observados.

3.6 - Gargalo

Apesar da maior carga ser referente ao processo de soldadura THT, o gargalo não se encontra completamente definido, porque este processo abrange mais do que uma operação e processos.

Na verdade, a engenharia da empresa HFA realiza uma simulação deste processo tendo em conta o custo unitário e a cadência da soldadura por onda, da soldadura seletiva e, ainda, do operador. Deste modo, procura minimizar o custo, escolhendo uma combinação ótima destas operações. No entanto, não é incomum ser necessário optar-se apenas pelo processo de soldadura seletiva, em virtude dos requisitos de qualidade. Este problema é descrito em pormenor na Secção 3.11.

Acrescendo à complexidade, há ainda encomendas de certos Clientes que estão restritas a uma única linha de produção ou a uma máquina distinta para a mesma operação. Assim sendo, a perceção do gargalo não será possível sem a descrição das máquinas e mão-de-obra, das linhas de produção e da identificação da influência dos principais Clientes.

3.7 - Equipamento

A atribuição das sete linhas aos equipamentos do bloco SMT, a identificação da maior parte dos equipamentos, e os respetivos custos por hora de funcionamento por hora estão representados na Tabela 3-2. Os custos por hora apresentados ao longo do Capítulo 3 encontram-se normalizados linearmente para o intervalo [0,1] através da expressão:

$$z = \frac{x - \min(x)}{[\max(x) - \min(x)]} \tag{3.1}$$

Tabela 3-2 — Custos hora de atividade por equipamento

Código Artigo	Nome Artigo	Custo/h
EQU0013	Linha 01	0,86
EQU0010	Linha 02	0,61
EQU0001	Linha 03	0,75
EQU0011	Linha 04	0,97
EQU0002	Linha 05	0,59
EQU0003	Linha 06	1,00
EQU0014	Linha 07	0,63
EQU0008 x 2	Soldadura por onda	0,39
EQU0005	Soldadura por onda ERSA	0,16
EQU0009 x 2	Soldadura seletiva	0,18
EQU0006	Soldadura seletiva VERSAFLO\	0,27
EQU0007	Inspeção SAKI	0,07
EQU0004	AOI - MARANTZ	0,07
EQU0016	ICT (TE)	0,00
EQU0012	Milling	0,04
EQU0015	Envernizar	0,07

O funcionamento das linhas quatro, cinco e seis está maioritariamente restrito a um Cliente de telecomunicações. Por outro lado, o equipamento seis é uma máquina distinta do equipamento nove e que realiza unicamente a operação de soldadura seletiva para um Cliente em específico do ramo automóvel. Por isso, e como se verificará nas próximas Secções, a nova matriz carga apresentada de seguida mostra uma elevada carga relativa aos equipamentos 2 (sendo a carga do equipamento 3 muito semelhante a este) e 11, mas que não tem uma influência decisiva para o gargalo.

3.8 - Matriz Carga - Equipamentos

Analisando os dados com as novas restrições em mente, é possível construir uma nova matriz carga para os equipamentos. O somatório dos tempos, em horas, para cada um dos equipamentos é representado na Tabela 3-3.

Tabela 3-3 — Matriz Carga – Equipamento

	SMD - Top & Bot						THT		Cliente Aut.
	EQU0013	EQU0010	EQU0001	EQU0002	EQU0011	EQU0014	EQU0009	EQU0008	EQU0006
Σ [h]	332	33	249	596	596	11	528	346	685

À primeira vista, o processo de soldadura THT, embora tenha uma carga significativa, pode parecer agora ultrapassado por operações como embalar e a programação presentes na Tabela 3-1. Contudo, é importante perceber-se que a última engloba, na verdade, diferentes etapas do processo de produção (e diversas máquinas), sendo realizada numa fase inicial, SMT, e numa fase final, durante os testes, por diferentes razões. Assim, este processo engloba tarefas como o carregamento de *firmware* e a verificação de, por exemplo, corrupção na memória *flash*. Por outro lado, a cadência de embalar por operário é relativamente elevada, sendo um processo facilmente ajustado desde que haja mão-de-obra disponível. Além disso, o processo de embalar tem também várias zonas alocadas, próprias para alguns Clientes em particular. De facto, no caso do Cliente do ramo automóvel, por exemplo, não só o processo de embalar, mas todo o processo relativo à assemblagem THT tem-lhe um local destinado.

Por último, é importante analisar-se em concreto as diferentes linhas de produção que constituem o processo SMT, que, no conjunto, possuiria a maior carga, para que possa ser comparado.

3.9 - Turnos

A HFA não tem todos os processos a funcionar nos mesmos turnos. Um turno tem uma duração de 8h, embora para efeitos práticos de produção, sejam consideradas 7,5h. Assim, o horário de um dia completo tem três turnos: entre as 05:00 e 13:00, 13:00 e 21:00, e 21:00 e 05:00. No contexto da pandemia Covid-19 foram introduzidos alguns desfasamentos pequenos não considerados.

Apresentam-se, na Tabela 3.4, os turnos praticados durante o período no qual se simulou o plano de produção. Note-se que os turnos estabelecidos poderão ser alterados e, por vezes, alguns processos/linhas não estão funcionais.

Tabela 3-4 — Turnos

	Turnos (Nº)		
	1	2	3
Linhas 1, 4 e 6			×
Linhas restantes		×	
Soldadura Onda		×	x
Soldadura Seletiva	×	×	x
Embalagem	×		
Outros	×	×	

Existem ainda desfasamentos dos turnos para além do SMT. A embalagem, por ter uma cadência relativamente elevada, apenas tem um turno. Já a diferença de turnos na soldadura é justificada pela existência de equipamentos distintos.

Na verdade, para além da soldadura seletiva exclusiva do Cliente mencionado, existem três equipamentos para esta operação e outros três para a soldadura por onda. Por exemplo, um dos equipamentos de soldadura por onda com maior capacidade é usado para circuitos impressos de maiores dimensões.

3.10 - Clientes

Os principais Clientes da empresa HFA, em termos de volume de negócios, são Clientes de telecomunicações e Clientes do ramo automóvel. Como se pode visualizar no cabeçalho da Tabela 3.1, as referências cujos números na terceira e quarta posições são ou “51” ou “85” correspondem a estes Clientes, respetivamente. Verifica-se que a maioria da carga de operações como a programação e testes é atribuída ao primeiro, enquanto a do processo de soldadura seletiva seria devido ao segundo. Além disso, relativamente ao processo SMT, as linhas 4, 5 e 6 são normalmente utilizadas para produtos do primeiro, e o mesmo acontece na linha dois com o segundo.

No Capítulo 4 são aplicadas algumas heurísticas para a realização dos planos de produção semanais. Em certos casos, como se verificará, o seu emprego imaculado levaria ao adiamento indeterminado das tarefas de carga superior. Os dois Clientes mencionados são os que têm maior carga e, ao mesmo tempo, os mais prioritários da empresa HFA.

Embora seja possível truncar as heurísticas, é crucial estabelecer um **sistema de etiquetas**[18], atribuindo-se a cor vermelha a estes Clientes. Assim, as suas encomendas

[18] Sistema de etiquetas, o sistema de etiquetas mencionado por Goldratt atribui a cor vermelha às tarefas de maior prioridade e a cor verde às de menor prioridade; é também realçada a importância da flexibilidade perante tal regra.

abertas irão ocupar continuamente as linhas dois, quatro e seis até serem faturadas na totalidade. Ao mesmo tempo, o sequenciamento incide sobre as outras tarefas nas restantes zonas até poder ser planeada uma agenda dinâmica global.

3.11 - Mão de Obra

A mão-de-obra no chão de fábrica é um aspeto importante, que influencia o gargalo e a definição de um *layout* eficiente. O custo por hora de um operador varia consoante o trabalho a que está destinado e pode ser consultado na Tabela 3-5. As operações listadas são todas aquelas que apenas requerem mão-de-obra, ou um operador associado a uma máquina. Neste último caso não são considerados os custos com as máquinas.

Tabela 3-5 — Mão-de-obra

Código Artigo	Nome Artigo	Custo/h
MOACB	Acabamentos THT	0,15
MOACBCEL	Acabamentos Célula	0,04
MOACBSMD	Acabamentos SMD	0,09
MOACBTESTE	Acabamentos do TESTE	0,09
MOASS	Assemblagem THT	0,15
MOASSCEL	Assemblagem Célula	0,04
MOASSSMD	Assemblagem do SMD	0,09
MOEMB	Embalagem	0,15
MOEMB1	Embalagem ONT7	0,09
MOEMBCEL	Embalagem Célula	0,04
MOFOR	Formatação	0,05
MOINSPMD	Inspeção do SMD	0,09
MOOFICINA	Oficina	0,15
MOSOLD	Soldadura (Manual)	0,00
MOSOLDCEL	Soldadura Célula	0,00
MOSOLDSC	Soldadura Seletiva Célula	0,00
MOSOLDSELC	Soldadura Máquina	0,00
MOTESTE	Teste PT	1,00
MOTESTECEL	Teste Célula	0,04
MOTESTEDIP	Teste DIP	0,15
MOTESTEGEN	Teste Genérico	0,50
MOTESTESMD	Teste SMD	0,09

A engenharia utiliza estes dados, juntamente com o preço estimado por hora das operações, de modo a estabelecer um compromisso entre a cadência e o custo por hora. Na Tabela 3-6 são apresentados os custos por hora do operador da assemblagem THT e de ambas as máquinas de soldadura com e sem operador.

Tabela 3-6 — Soldadura Seletiva vs Soldadura Onda

THT	Sem Operador		Com Operador	
Operador	Seletiva	Onda	Seletiva	Onda
0,39	0,00	0,33	0,67	1,00

Para um novo produto, sendo desconhecida a sua cadência nalguma das operações de soldadura, esta é estimada. Este procedimento é exequível tendo em conta o número de pinos e a sua disposição nas placas, a espessura, o comprimento e a complexidade destas. A cadência da soldadura seletiva é tão mais lenta quanto maior o número de pinos e a irregularidade da sua disposição na PCB. Note-se que a soldadura seletiva dedicada ao Cliente do ramo automóvel não é influenciada da mesma forma, já que, neste caso, o processo está automatizado para a produção de séries maiores. Já a cadência da soldadura por onda depende apenas do comprimento da placa. Isto porque os parâmetros de velocidade de translação e temperatura associados à máquina são aproximadamente constantes, podendo variar ligeiramente apenas devido à espessura ou à complexidade de determinados componentes. Assim, caso o gargalo esteja na assemblagem, para além do operador destinado à máquina poder ajudar, pode-se estabelecer o número mínimo de recursos humanos para que a restrição esteja sempre nesta operação.

O gargalo deve ser sempre a máquina. No entanto, para a otimização do *makespan*, esta situação nem sempre se verifica devido à limitação em termos de recursos humanos disponíveis. Esta realidade será confrontada na Secção 4.2.4 - Plano de Produção 3.

Os recursos humanos destinados à produção são classificados numa escala de 1 a 4 cujo significado é explicito na Figura 3.2. Esta informação é frequentemente atualizada para todas as operações.

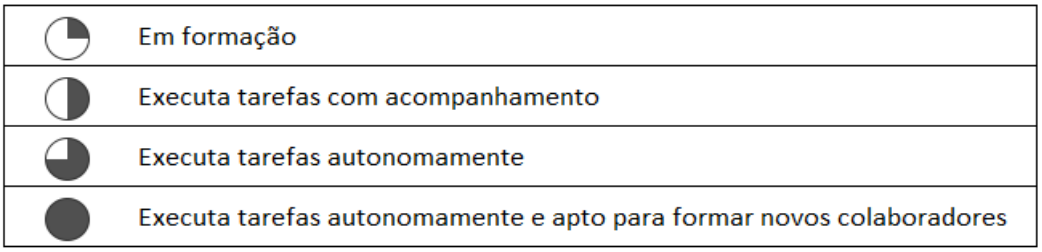

Figura 3.2 – Recursos Humanos – Formação

Na Secção 4.2.4 são explicados os obstáculos encontrados impostos por estas restrições. Posteriormente, as simulações mostraram que o número de recursos humanos não é o fator limitante em várias operações. Esta afirmação é justificada pela contratação a que a HFA tem recurso e que permite dar resposta aos picos de procura ao longo do ano. Não obstante, é programado um algoritmo para a otimização da alocação de recursos humanos descrito no Capítulo 5.

3.12 - Outros Dados

Os turnos discutidos na Secção 3.9 nem sempre refletem os tempos de partida ou conclusão das operações num determinado dia. Na verdade, algumas operações na fase inicial poderão ser iniciadas mais cedo. Outras operações, agora numa fase final, poderão ser processadas de forma independente para permitir a alocação de recursos humanos noutros locais.

Um exemplo real sucede-se com a referência "538503011R". A dinâmica e alocação destes recursos humanos à produção é esquematizado na Figura 3.3. Na Figura 3.4 observa-se que a cravação é processada mais cedo do que as restantes operações. Isto acontece porque a soldadura seletiva é o gargalo, criando-se imediatamente antes um *buffer* e uma corda que faz a comunicação desta à tarefa de cravação. Assim, o recurso humano destinado à soldadura seletiva nunca deixará de trabalhar nesta, não tendo que esperar pela transferência de placas entre as operações, que só ocorre em lotes com cerca de dezasseis de cada vez.

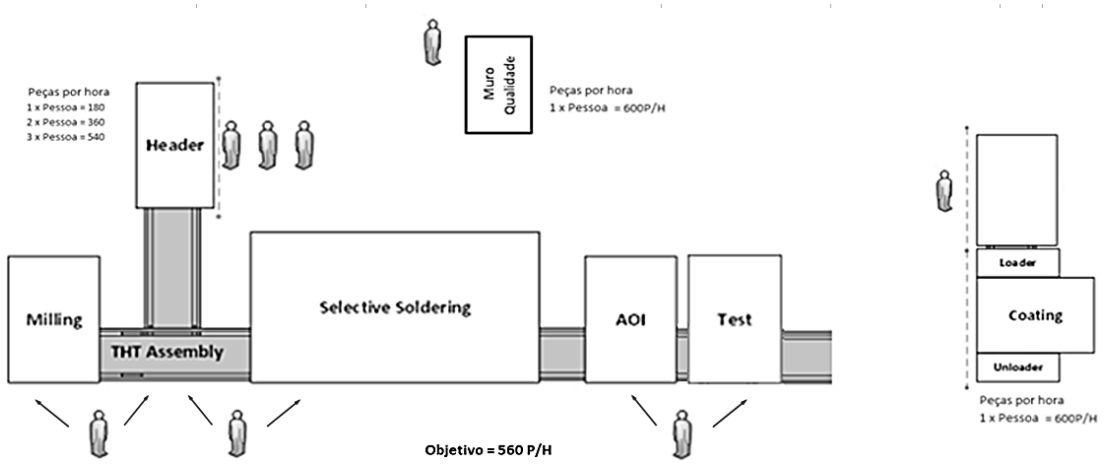

Figura 3.3 – Layout

A situação aqui apresentada teve influência na contabilização da mão-de-obra necessária durante a execução do plano de produção na Secção 4.2.4.

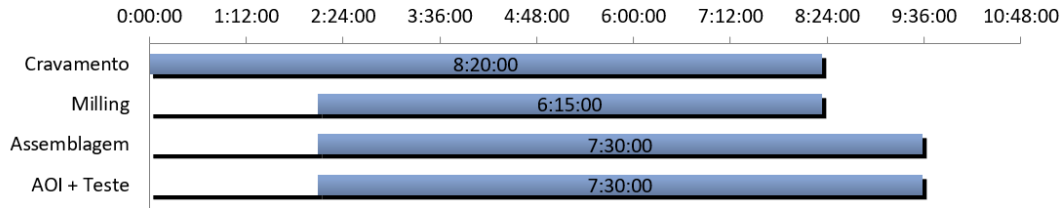

Figura 3.4 – Gantt

Outro importante tópico está relacionado com os dados relativos às cadências das operações de um qualquer produto nas diferentes linhas. Frequentemente, um produto que já foi encomendado no passado tem associada uma cadência para o conjunto "SMT", ou duas caso o "BOT" e o "TOP" sejam processados em linhas distintas. Acontece que, ou estes dados podem já estar desatualizados por não refletirem a tecnologia presente, ou é ideal agendar a tarefa noutra linha, se possível. No segundo caso, consegue-se converter estas cadências para outra linha. Assim, calcula-se uma estimativa para esta baseada nos pesos das setes linhas com os dados mais recentes. É importante referir que este último caso é de evitar uma vez que, como se verá no Capítulo 5, é um motivo da variabilidade entre os tempos planeados e os praticados. Esta diferença pode surgir ao converter-se a cadência inferior de uma determinada linha para outra de maior, mas não se considerar agora, o novo gargalo intrínseco ao processo, para a segunda linha.

Um exemplo para a conversão de uma cadência de "147" peças/hora registada na linha 1 é apresentado na Tabela 3-7. A mesma abordagem pode ser efetuada para qualquer linha.

Tabela 3-7 — Conversão da cadência na linha 1 para as outras linhas de produção

		Linha	**Qtd.**	**%**	**Conversão Linha**		
Linha 1	100%	Linha 2	42000	41%	147	Linha 2	60
		Linha 3	63000	61%		Linha 3	90
		Linha 4	114000	111%		Linha 4	163
		Linha 5	83100	81%		Linha 5	119
		Linha 6	198000	192%		Linha 6	283
		Linha 7	42000	41%		Linha 7	60

Por último, existe ainda uma "linha 8" destinada a protótipos, e onde poderão ser processadas algumas encomendas pequenas. No entanto, esta não foi considerada para o plano de produção.

Capítulo 4

Sequenciamento de tarefas

O fluxo de produção equivale a fluxo de caixa, e agendar está na raiz do processo. Agendar consiste num calendário para realizar atividades, utilizar recursos, ou alocar instalações Jacobs (2014). Neste capítulo, apresentam-se as heurísticas utilizadas a curto prazo e o desenvolvimento do plano de produção.

4.1 - Regras de prioridade - avaliação

As regras de prioridade são as regras usadas para obter uma sequência de tarefas. Isso pode ser muito simples, exigindo apenas que as tarefas sejam sequenciadas de acordo com uma parte dos dados, tais como tempo de processamento, data de entrega ou ordem de chegada. Outras regras, embora igualmente simples, podem exigir várias informações, tais como a regra da menor folga (*least-slack*) e a regra da razão crítica (*critical-ratio*). Outras ainda, como a regra de Johnson, discutida mais tarde, aplicam-se ao agendamento de tarefas numa sequência de máquinas e requerem um procedimento computacional para especificar a ordem de desempenho.

Os seguintes princípios de desempenho são usados para avaliar as regras de prioridade:

- Cumprir as datas de entrega do Cliente ou operações a jusante.
- Minimizar o tempo de fluxo.
- Minimizar o WIP.
- Minimizar o tempo inativo das máquinas ou trabalhadores.

4.2 - Problema com n tarefas e uma máquina

O agendamento estático envolve uma classe de problemas referidos como "n *job - one-machine problem*" ou simplesmente "n/1". A dificuldade teórica dos problemas aumenta à medida que são consideradas mais máquinas ao invés do processamento incremental de

tarefas. Na verdade, existem n tarefas e m máquinas, com $m >> 1$. Ao longo desta Secção, são explicadas as considerações e o método utilizado para a aplicação das heurísticas subsequentes.

4.2.1 - SPT

A heurística SPT - *Shortest Processing Time*, dá maior prioridade à tarefa que tem o menor tempo de processamento. Pode ser demonstrado matematicamente que esta regra resulta num tempo médio de fluxo ótimo, mantendo um bom desempenho em termos de atraso médio. Todavia, esta regra simples e eficaz tem as suas inconveniências. A principal delas é que as tarefas mais longas podem nunca ser iniciadas se continuarem a chegar tarefas curtas. Para evitar isso, pode-se utilizar a regra **SPT truncada**[19], na qual as tarefas à espera por um período de tempo especificado são automaticamente movidas para a frente da linha.

Como o problema real tem diferentes processos com uma sequência definida e um recurso a múltiplas máquinas, podem ser definidos gargalos em cada bloco de operações para uma dada referência. A regra dá, assim, prioridade ao menor tempo de entre o maior dos respetivos gargalos. Funciona, por isso, como uma regra de **MinMax**[20] do tempo das várias operações. Uma referência com um tempo de processamento total menor, mas com algum gargalo local maior, é mais restritiva, porque as operações não são independentes.

Na verdade, o planeamento da empresa HFA realiza dois planeamentos semanais, uma para o SMT e outra para as restantes operações, contendo as mesmas encomendas, mas de forma individual. Isto foi identificado como uma ineficiência, e resulta em obstáculos como a acumulação de *stock* intermédio e folgas elevadas ao longo de todo o processo, uma vez que o gargalo é a operação "THT" como se verificou no Capítulo 3. O agendamento proposto não seguirá a mesma lógica.

4.2.2 - Plano de Produção 1

A primeira tentativa de realizar o plano de produção foi baseada nesta regra. No entanto, houve alguns obstáculos não presentes que o tornaram inviável. Após um contacto direto adicional com o chão de fábrica e a comunicação entre os responsáveis pelo planeamento semanal, percebeu-se quais foram as restrições técnicas não consideradas. Algumas operações relativas ao SMT são processadas duas vezes pela linha ou linhas, sendo que na primeira se processa impreterivelmente o "BOT" seguido da segunda com o "TOP", fora raras exceções. Além disso, a operação "inspeção" pode ser realizada em ambas as etapas, ou apenas no final da última.

[19] SPT – Truncation. É necessário truncar-se o SPT para não se adiar indefinidamente as tarefas de maior carga, usando-se, para o efeito, o tempo máximo de atraso obtido com o EDD.

[20] MinMax É uma regra de decisão aplicada em vários campos; neste caso permite minimizar o tempo total de processamento, sabendo-se que estamos restringidos pelo maior dos gargalos

Por outro lado, as cadências teóricas registadas devem ser afetadas de um índice OEE - *Overall Equipment Effectiveness*. O **OEE**[21] associado a cada linha é indicado na Tabela 4-1. Este índice contempla a influência na cadência de vários fatores como o desgaste dos *nozzles*[22] das máquinas, que devem ser substituídos quando for observada uma diminuição na cadência efetiva, ou a avaria de componentes. Por último, surgem, situações imprevisíveis / aleatórias como o sucedido com o desmaio de um operário.

Tabela 4-1 — Índice OEE de referência por linha

Linhas	1	2	3	4	5	6	7
OEE	70%	70%	60%	80%	80%	80%	60%

4.2.3 - Plano de Produção 2

Após a realização de um segundo plano de produção, percebeu-se que não é possível estabelecer um valor médio de cerca de uma a duas horas para os **tempos de *setup***, uma vez que os produtos mais complexos, principalmente do Cliente de telecomunicações, podem chegar a pelo menos oito horas. Há também uma diferença significativa entre os tempos de *setup*, para a mesma referência, entre o "BOT" e o "TOP". Além disso, este tempo não depende só da referência ou da operação, mas também da linha onde é processado, por causa dos equipamentos distintos. Assim, foi necessário realizar-se algumas *queries* em SQL no *software* da CentralGest de modo a obter-se os dados necessários.

Adicionalmente, foram agendadas algumas encomendas que já não estavam em aberto, mas que continuavam a constar no ERP - *Enterprise Resource Planning* uma vez que ainda não tinham sido faturadas. O *stock* registado inclui tanto o stock em curso de fabrico, como o do armazém, sendo que este último contém produtos acabados, produtos resultantes de operações intermédias no SMT, THT e por reparar.

Contudo, apenas um obstáculo não foi solucionado definitivamente. Há várias encomendas pequenas resultantes de ordens de fabrico que não foram concluídas no passado. Destas, algumas também já não se encontram em aberto, porque foram **abatidas** e permanecem no sistema. As razões estão relacionadas com lacunas na organização e método no registo de dados e utilização do sistema de informação. Assim, este problema teve que ser analisado caso a caso.

No seguimento dos problemas anteriores, é importante referir ainda que não é possível conhecer-se qual o *stock* dos dois Clientes principais. Se as encomendas em aberto são registadas num documento específico, existe ainda um documento para os pedidos dos Clientes antes de serem confirmadas. Além disso, estas encomendas são faturadas

[21] OEE, medir o OEE é a prática recomendada de fabricação, identificando a percentagem em que esta é realmente produtiva.

[22] Nozzle: dispositivo projetado para controlar a direção ou as características do fluxo de um fluído.

componente a componente, sendo necessária uma forma automatizada para processar esta informação. Todavia, esta não é fundamental para o "SMT", já que o agendamento heurístico não incide sobre estes Clientes. Na verdade, enquanto as outras encomendas são processadas segundo um sistema *pull*, as encomendas destes Clientes são continuamente "empurradas" (*push*) para o mercado e as respetivas linhas estão permanentemente carregadas. Continua a ser necessário fazer-se a previsão das encomendas ao Cliente das telecomunicações de modo a saber-se a carga disponível no THT.

4.2.4 - Plano de Produção 3

O terceiro plano de produção teve em conta todas as questões que surgiram nas duas semanas anteriores e foram elaboradas, ainda, algumas diretrizes:

- As primeiras encomendas agendadas em cada linha devem ser todas operações com "BOT" ou operações "TOP" que não necessitem de "BOT", e que necessitem do processo "THT". (Permite aumentar o número de produtos em processo e diminuir o tempo inativo no THT.)
- O "BOT" e o "TOP" são normalmente agendados na mesma linha. (Embora possam ser processados em linhas diferentes para o SMT acabar primeiro e reduzir-se o tempo inativo no THT, não é viável prever-se com confiança um tempo semelhante de finalização dessas operações, face a todas a variáveis.)
- Algumas encomendas com datas de entrega distintas são agrupadas em apenas uma ordem de fabrico. (Permite reduzir os tempos de *setup* que por sua vez permitirão reduzir o atraso médio no longo termo; apenas é feito caso as encomendas sejam pequenas ou as datas de entrega sejam próximas.)
- Assume-se que a operação inspeção é sempre realizada depois do "TOP" para efeitos de contabilização do tempo. (Não há registos sobre como esta é efetuada para cada referência, embora esta consideração não afete o planeamento, visto o "BOT" e o "TOP" serem agendados na mesma linha de forma consecutiva.)

Durante este período, o contacto com a produção revelou também uma preocupação em saber o número de colaboradores que seriam necessários, discriminados por dia da semana. De facto, apesar de algumas operações terem, relativamente, uma elevada cadência, poderão ser uma restrição devido ao **número de colaboradores disponíveis**. Estas operações são: embalar, testar, cortar, soldadura manual e envernizar/colar.

Nas Tabela 4-2 e Tabela 4-3 são apresentadas, respetivamente, a carga e consequentemente o número de funcionários necessários para estas operações ao longo da semana.

Tabela 4-2 — Carga por operação

Carga/Operação [h]	segunda	terça	quarta	quinta	sexta
EMBALAR	15,80	31,12	10,56	4,17	15,42
CORTES	12,37	23,04	4,06	2,08	8,23
SOLDADURA MANUAL	16,48	26,21	0,00	0,00	0,00
TESTAR	6,64	11,11	0,00	0,00	0,00

Tabela 4-3 — Número de pessoas por operação

	segunda	terça	quarta	quinta	sexta
EMBALAR	2	4	2	1	3
CORTES	1	2	1	1	1
SOLDADURA MANUAL	1	2	0	0	0
TESTAR	1	1	0	0	0

A carga por operação foi calculada no final do agendamento. O número de pessoas necessárias é determinado através desta carga e consoante o número de turnos já conhecido associado a cada operação. É importante ter-se em conta, ainda, o facto de a carga não ser distribuída uniformemente ao longo de um dia qualquer. Além disso, como há operações cuja carga é mínima em comparação com o horário disponível pelo turno, é possível movimentar pessoas de outras operações para as concluir quando necessário. Esta tarefa normalmente está destinada a uma pessoa responsável por linha, que gere os recursos disponíveis em tempo real e com maior flexibilidade.

Conclui-se desta análise que, embora as cadências nestas outras operações sejam elevadas, elas poderão ser uma restrição devido ao limite de operadores disponíveis, bem como a sua qualificação para determinado processo. A formação dos recursos humanos nas diversas tarefas é fundamental, e nem sempre se verifica. Esta realidade refuta a situação ideal na qual se discute a minimização de custos e otimização do fluxo entre soldadura seletiva e soldadura onda na Secção 3.11.

Além disso, há ainda a variabilidade significativa verificada entre as cadências nos vários processos por operador. Um determinado colaborador pode ser mais eficaz em certas tarefas, ou mesmo ter desempenhos muito diferentes em dias consecutivos. As causas podem estar relacionadas com a maior experiência deste em determinadas operações, ou a motivação/condições de trabalho num determinado momento. Uma sugestão de solução passa por escolher os melhores operadores para cada tarefa e permitir que estes exponham os seus métodos e raciocínio com todos os outros.

4.2.5 - EDD

A heurística EDD, *Earliest Due Date*, corre primeiro as tarefas com as datas de entrega ao Cliente mais próximas, favorecendo uma importante questão logística, em troca de um menor desempenho no tempo médio de fluxo. Contudo, mesmo assim, os resultados mostram frequentemente um aumento do atraso médio, tendo apenas um menor atraso máximo

comparativamente à regra SPT pura. De facto, a heurística EDD é ótima quando o objetivo é minimizar o atraso máximo.

Uma das simulações realizadas para a comparação da heurística SPT com a heurística EDD incidiu sobre uma amostra de 25 encomendas. Estas encomendas foram escolhidas a partir da matriz carga na Tabela 3-1, considerando-se um período temporal no qual haveria uma grande dificuldade em cumprirem-se os prazos de entrega. Desta forma, consegue-se salientar as principais diferenças entre as heurísticas. Além disso, como estas heurísticas são aplicadas a problemas do tipo "n/1", considerou-se como tempo de ciclo o gargalo de cada encomenda e o processamento numa só máquina. Os resultados estão apresentados na Tabela 4-4

Tabela 4-4 — SPT vs EDD

	Tempo de Ciclo Médio [dias]	Qtd. de Encomendas Atrasadas	Desvio Padrão do Atraso [dias]	Atraso Máximo [dias]
SPT	0,46	12	3,90	14
EDD	0,67	11	3,84	14

O tempo de ciclo médio do EDD foi cerca de 45% superior comparativamente ao do SPT. O desvio padrão do atraso foi escolhido como parâmetro, uma vez que o atraso médio de todas as encomendas (incluindo as que cumprem com a data de entrega) é pouco intuitivo, mas, por outro lado, a quantidade de encomendas atrasadas é também distinta. Note-se que o atraso médio de todas as encomendas seria menor no caso do SPT. O EDD conseguiu ter menos uma encomenda atrasada e apresentou o mesmo atraso máximo quando arredondado.

Considerando que o EDD minimiza este último parâmetro, não mostrou um desempenho satisfatório. Contudo, no longo termo, embora o SPT leve a menos encomendas atrasadas, a diferença no atraso máximo será cada vez mais acentuada.

A heurística EDD assim como todas os pressupostos e informações obtidas com a aplicação da heurística SPT será utilizada como uma possível escolha no programa desenvolvido no Capítulo 5.

4.3 - Problema com *n* tarefas e duas máquinas

A próxima configuração em complexidade é o caso "$n/2$", em que m trabalhos devem ser processados em duas máquinas numa sequência comum. Como no caso "$n/1$", é uma abordagem que leva a uma solução ótima de acordo com um determinado critério. O objetivo desta abordagem, denominada regra de Johnson (Johnson, 1954), é minimizar o tempo de fluxo desde o início do primeiro trabalho até o fim do último.

4.3.1 - Johnson

A regra de Johnson consiste nas seguintes etapas:

- Listar o tempo de operação para cada tarefa em ambas as máquinas.
- Selecionar o menor tempo de operação.
- Se o tempo mais curto for na primeira máquina, fazer a tarefa primeiro; se estiver na segunda máquina, fazer a tarefa por último. Em caso de empate, fazer a tarefa da primeira máquina.
- Repetir as etapas dois e três para cada tarefa restante até que o sequenciamento seja concluído.

Assim, para a aplicação desta heurística, agruparam-se todas as operações das m máquinas em dois conjuntos (máquinas). A forma ideal de aplicar a regra de Johnson seria num caso cujo ponto médio da carga se situa entre a escolha das duas máquinas. Compararam-se duas situações: a primeira, Tabela 4-5 com máquina 1 (SMT) e máquina 2 (THT + Outros); e a segunda, Tabela 4-6 com máquina 1 (SMT + THT) e máquina 2 (Outros).

As colunas de 1 a 10 representam a ordem a sequenciar para cada tarefa, sendo destacado a verde a respetiva tarefa selecionada pelo algoritmo.

Tabela 4-5 — Máquina 1 (SMT) e Máquina 2 (THT + Outros)

Tarefa	Máquina 1	Máquina 2	1	2	3	4	5	6	7	8	9	10	
1	0,02	0,02	0	0	0	0	0	0	0	0	0	1	
2	0,03	0,04	0	0	0	0	1	0	0	0	0	0	
3	0,01	0,08	0	0	0	1	0	0	0	0	0	0	
4	0,05	0,19	0	0	0	0	0	1	0	0	0	0	
5	0,13	0,49	0	0	0	0	0	0	0	0	1	0	
6	0,49	3,17	0	0	0	0	0	0	1	0	0	0	
7	0,00	13,67	0	1	0	0	0	0	0	0	0	0	
8	0,00	44,05	0	0	1	0	0	0	0	0	0	0	
9	10,11	163,04	0	0	0	0	0	0	0	1	0	0	
10	0,00	1,35	1	0	0	0	0	0	0	0	0	0	
Máquina 1: tempo para a tarefa			0	0	0	0	0	0	1	11	11	11	3
												Makespan	
Máquina 2: tempo para a tarefa			1	15	59	59	59	59	63	226	226	226	99
													AFT
													51

	J10	***J7***	***J8***	***J3***	***J2***	***J4***	***J6***	***J9***	***J5***	***J1***
Máquina 1: tempo de partida	0,00	0,00	0,00	0,00	0,01	0,04	0,09	0,58	10,69	10,82
Máquina 1: tempo de conclusão	0,00	0,00	0,00	0,01	0,04	0,09	0,58	10,69	10,82	10,84
Máquina 2: tempo de partida	0,00	1,35	15,02	59,07	59,15	59,19	59,38	62,55	225,59	226,08
Máquina 2: tempo de conclusão	1,35	15,02	59,07	59,15	59,19	59,38	62,55	225,59	226,08	226,09

Tabela 4-6 — Máquina 1 (SMT + THT) e Máquina 2 (Outros)

Tarefa	Máquina 1	Máquina 2	1	2	3	4	5	6	7	8	9	10	
1	0,02	0,02	0	0	0	0	0	0	0	0	1	0	
2	0,06	0,01	0	0	0	0	0	0	0	0	0	1	
3	0,03	0,07	0	1	0	0	0	0	0	0	0	0	
4	0,09	0,15	0	0	1	0	0	0	0	0	0	0	
5	0,23	0,38	0	0	0	1	0	0	0	0	0	0	
6	0,49	3,17	0	0	0	0	1	0	0	0	0	0	
7	3,00	10,67	0	0	0	0	0	1	0	0	0	0	
8	35,71	8,33	0	0	0	0	0	0	0	1	0	0	
9	18,67	154,48	0	0	0	0	0	0	1	0	0	0	
10	0,00	1,35	1	0	0	0	0	0	0	0	0	0	
Máquina 1: tempo para a tarefa			0	0	0	0	1	4	23	58	58	58	20
												Makespan	
Máquina 2: tempo para a tarefa			1	1	2	2	5	16	177	185	185	185	76
													AFT
													48

	J10	***J3***	***J4***	***J5***	***J6***	***J7***	***J9***	***J8***	***J1***	***J2***
Máquina 1: tempo de partida	0,00	0,00	0,03	0,12	0,35	0,85	3,85	22,51	58,23	58,25
Máquina 1: tempo de conclusão	0,00	0,03	0,12	0,35	0,85	3,85	22,51	58,23	58,25	58,30
Máquina 2: tempo de partida	0,00	1,35	1,42	1,57	1,95	5,12	22,51	176,99	185,33	185,34
Máquina 2: tempo de conclusão	1,35	1,42	1,57	1,95	5,12	15,78	176,99	185,33	185,34	185,36

Destaca-se a laranja na Tabela 4-5 e Tabela 4-6 o *makespan* e o tempo médio de fluxo (AFT). Para as mesmas dez encomendas selecionadas, percebe-se que a escolha de um **ponto intermédio**[23] não é óbvia. Não só as operações "Outros" apresentam carga com elevada variabilidade, como as próprias encomendas relativas a um determinado período deslocam a posição ideal para este ponto de forma significativa. Assim, optou-se por não implementar esta heurística uma vez que, nesta situação, o SPT é simplesmente mais eficiente.

4.3.2 - Generalização - Johnson

Uma abordagem MIP foi usada para generalizar a regra de Johnson para $m > 2$ máquinas. Na Secção anterior, usou-se este modelo no Solver do Excel para um caso com duas máquinas, Kashipur (2021).

Variáveis

$\alpha_{it} = 1$ se a tarefa for processada no período t, 0 em caso contrário.

P_i^1: Tempo de processamento da i-ésima tarefa na máquina 1.

P_j^n: Tempo de processamento da i-ésima tarefa na máquina n.

$max(0, f_{N-1}(T_t) - f_N(T_{t-1})$: Tempo de espera da máquina N.

$f_1(T_t) = [\sum_{i=1}^{n} \alpha_{it} P_i^1 + f_1(T_{t-1})]$, tempo de fluxo cumulativo para a máquina 1.

[23] Ponto Intermédio: a eficácia da regra de Johnson depende da capacidade de se distribuir igualmente a carga por ambas as máquinas

$f_N(T_t) = [\sum_{i=1}^{n} \alpha_{it} P_i^N + f_N(T_{t-1}) + max(0, f_{N-1}(T_t) - f_N(T_{t-1})]$, tempo de fluxo cumulativo para a máquina n.

Restrições

α_{it} é uma variável binária.

$\sum_{t=1}^{n} \alpha_{it} = 1$, a tarefa n é agendada uma e só uma vez.

$\sum_{i=1}^{n} \alpha_{it} = 1$, não mais que uma tarefa pode ser agendada num determinado período.

Onde n é o número de tarefas e N o número de máquinas. A função objetivo depende do que se pretende otimizar. Assim são apresentadas as equações a minimizar para o tempo médio de fluxo (4.1), makespan (4.2) e tempo de espera entre máquinas (4.3).

$$Min[|\frac{1}{n}\sum_{i=1}^{n} f_n(T_t)\ |] \tag{4.1}$$

$$Min[\sum_{i=1}^{n} \alpha_{it} P_i^N + f_N(T_{t-1}) + max(0, f_{N-1}(T_t)] \tag{4.2}$$

$$Min[\sum_{i=1}^{n} \alpha_{it} P_i^1 + \sum_{i=1}^{T} max(0, f_{N-1}(T_t) - f_N(T_{t-1}) + (f_N(T_t - f_{N-1}(T_t))] \tag{4.3}$$

4.4 - Problema com *n* tarefas e *m* máquinas

No problema real há $n!$ sequências de tarefas possíveis para cada máquina e $n!^m$ agendamentos. Entre estas combinações, se considerarmos aquelas cuja mesma sequência de tarefas ocorre em cada máquina, teremos o caso da "permutação de agendamento". Neste caso, o esforço necessário para encontrar o melhor agendamento limita-se à procura por apenas $n!$ cenários. Nesta Secção, os algoritmos desenvolvidos consideram o caso da permutação em problemas de minimização de atrasos no fluxo.

Considerações

O problema generalizado é caracterizado pelas seguintes condições:

- Um número n de tarefas independentes está aberto no instante zero (encomendas abertas).
- Cada tarefa requer m operações, sendo a operação j processada na máquina j.
- Os tempos de *setup* são independentes da sequência e incluídos nos tempos de processamento.
- O tempo de processamento de cada operação é conhecido antecipadamente (matriz carga) e é determinístico.
- O processo é contínuo (todas as máquinas estão continuamente disponíveis).

- Quando uma operação é inicializada, procede sem interrupções (não são considerados eventuais problemas resultantes de não conformidades, por exemplo).

Estas condições englobam as suficientes para ser aceite como modelo básico do JSSP (Baker, Sequencing rules and due-date assignments in a job shop, 1984). No modelo generalizado, o total do atraso de um agendamento é definido pela soma dos atrasos individuais de cada tarefa, sendo o atraso de cada tarefa definido pela soma dos atrasos das operações individuais da mesma.

Variáveis

As seguintes notações são usadas ao longo da Secção "n/m":

P_{ij}; Tempo de processamento para a operação j da tarefa i.
d_{ij}: *Due date* da operação j para a tarefa i.
r_{ij}: Tempo total de processamento da tarefa i na máquina $1,2,\ldots,j-1$.
t_{ij}: Tempo de conclusão da tarefa de ordem j, anterior à i, na máquina j
C_{ij}: Tempo de conclusão da tarefa i na máquina j
x_{ij}: Tempo de arranque da operação de ordem k da tarefa i.
y_{ij}: Variável binária, sendo 1 quando a tarefa j precede a tarefa i, e 0 nos restantes casos.
T_{ik}: Atraso da operação de ordem k da tarefa i.
v_i: Valor esperado do atraso da tarefa i.
S_{ji}: Tempo de folga da tarefa i na máquina j.
TT: Atraso total; ou de S_j, $[TT(S_j)]$; ou de todas as tarefas na máquina j, $[TT_j]$.
$T_k(S)$: Atraso da tarefa k no agendamento S.
A_j: Conjunto ordenado de tarefas parcialmente sequenciadas na máquina j.
B: Conjunto não ordenado de tarefas (complemento de A_j).
S_j: Agendamento permutável, na ordem das tarefas na máquina j.

Função Objetivo

Se T_{ik} denotar o atraso da operação (máquina) de ordem k da tarefa i, então o valor T_{ik} é determinado através da seguinte expressão:

$$T_{ik} = max[0, (x_{ik} + p_{ik} - d_{ik})] \forall i = 1,2\ldots n, k = 1,2,\ldots m \quad (4.4)$$

E a função objetivo do modelo matemático vem:

$$min: TT = max \sum_{j=1}^{n} \sum_{k=1}^{m} T_{ik} \quad (4.5)$$

O tempo computacional para a obtenção da solução ótima num modelo de otimização combinatória aumenta exponencialmente com o aumento do número de variáveis de decisão. Uma vez que os problemas reais do JSSP são formulados como um modelo de otimização combinatória em grande escala, a única abordagem aplicável para a obtenção de uma solução é uma abordagem heurística (Lauff & Werner, 2004). Assim, será desenvolvido um algoritmo.

4.5 - COVERT modificado

Uma análise completa do problema generalizado indica que, para tarefas a sequenciar, é mais eficaz o uso de uma heurística que consiste em tempos de processamento de tarefas e nas suas datas de entrega associadas (Ghassemi-Tari & Olfat, 2008).

4.5.1 - COVERT original

O COVERT original representa o custo de atraso esperado por unidade de tempo de processamento iminente, ou custo ao longo do tempo (Baker, Sequencing rules and due-date assignments in a job shop, 1984). O sequenciamento da tarefa i para a operação j com folga nula ou negativa é projetada para ter um atraso esperado de v_i e prioridade dada pelo índice v_i/p_{ij}. Se, por outro lado, a folga exceder alguma estimativa generosa do "*worst case*" para o tempo de espera, o custo esperado de atraso é definido como zero. O tempo de espera do pior caso serve como referência do agendamento (com visão no futuro) do custo esperado de atraso sobre a folga S_{ij}:

$$COVERT_y(t) = \frac{v_i}{p_{ij}} . \frac{k.[\sum_{q=j}^{m_i} W_{iq} - (S_{ij})^+]^+}{\sum_{q=j}^{m_i} W_{iq}} \tag{4.6}$$

Onde W_{iq}é o tempo de espera esperado para a operação seguinte q, e k é um fator que ajusta este tempo para o "*worst case*", por exemplo, através de uma distribuição normal com intervalo de confiança a 99%. O valor de k pode ser determinado experimentalmente.

4.5.2 - Algoritmo proposto

O algoritmo começa por determinar, através do índice COVERTM, uma ordem dos trabalhos na máquina 1. Em seguida, considera-se as máquinas de 2 a m e, usando o mesmo índice, determina-se a ordem para cada máquina individual. Através deste procedimento, identificam-se m diferentes ordens das tarefas. Assim, utiliza-se cada pedido individual para determinar uma agenda de permutação de n tarefas na máquina m, obtendo-se m diferentes combinações. Para cada agenda de permutação, determinamos o valor do atraso total e, entre esses m agendamentos, seleciona-se aquele com o menor atraso total.

$$COVERTM_{ij} = \frac{1}{p_{ij}} \frac{[k.b.p_{ij} - (d_{ij} - max[t_{ij}, r_{ij}])^+]^+}{k.b.p_{ij}} \tag{4.7}$$

O melhor valor para as constantes b e k é sugerido ser 2, segundo Baker (1984). Além disso, este índice teve o melhor desempenho entre os quatro simulados por Olfat (2008), numa amostra reduzida. A sua implementação vai na continuidade da aplicação desta heurística a curto prazo.

As etapas do algoritmo "CO" proposto podem ser apresentadas como se segue:

1 Seja $j = 1$, e $r_{ij} = 0\ para\ i = 1,2,\dots,n\ e\ j = 1,2,\dots,m.$

2 Seja $TT_j = 0, t_{ij} = 0, C_{ij} = 0, A_j = \phi\ e\ B = \{1,2,\dots,n\}.$

3 Calcular CO_i com:

$$CO_i = max\{COVERTM_{ij}(t_{ij})\}.$$

4 Seja i a tarefa associada a CO_i, remover i de B e colocá-lo na última posição de A_j

5 Seja $C_{ij} = max\{t_{ij}, r_{ij}\} + p_{ij}$, e $TT_j = TT_j + max\{0, (C_{ij} - d_{ij})\}$. Se B estiver vazio, ir para etapa 6; em caso contrário, deixar $t_{ij} = C_{ij}$, e ir para a etapa 3.

6 Definir o agendamento S_j, ao sequenciar as tarefas pela ordem em A_j.

7 Seja $TT(S_j)$ definido como o atraso total de S_j, calcular o valor de $TT(S_j)$. Se $j = m$, fazer $TT(S_k) = min\{TT(S_j)\}$, selecionar S_k como agenda final, e depois parar. De outro modo, fazer $j = j + 1$, e $r_{ij} = \sum_{k=1}^{j-1} p_{ik}$ para $i = 1,2,\dots m$, depois ir para a etapa 2.

Capítulo 5

Programa e Resultados

Neste capítulo é elaborado um programa que automatiza o plano de produção através de diferentes heurísticas e com base em todas as considerações do Capítulo 4, acrescendo as apresentadas de seguida.

5.1 - Novas Considerações

- As operações "BOT" e "TOP" são agendadas na mesma linha, à exceção das linhas 4, 5 e 6. (O "BOT" tem de ser realizado em primeiro lugar, e há alguma acumulação de stock deste no fim da linha antes de ser processado o "TOP". Há perdas resultantes de matéria-prima ao ser processado em linhas distintas, entre outras desvantagens já mencionadas. Assim, só compensa agendar estas operações em linhas separadas para grandes encomendas, onde o processo passa a funcionar continuamente. Note-se que esta afirmação não tem em consideração a possibilidade de custos superiores.)

- Quando existem três turnos disponíveis para uma determinada linha, o primeiro turno de segunda-feira, ou seja, o primeiro dia da semana, é considerado como sendo o turno da noite de domingo. Além disso, tem-se em consideração, ainda, o tempo de pausa para almoço.

5.2 - Complexidade do Programa

O tempo de execução do programa desenvolvido depende principalmente do algoritmo de sequenciamento. O algoritmo usado foi o *Timsort*[24]. A sua complexidade para os diferentes casos encontra-se na tabela 5.1.

[24] O programa foi desenvolvido no Google Colaboratory e é executado num GPU da Google

Timsort é um algoritmo de sequenciamento híbrido, derivado de *merge sort* e *insertion sort*, pensado de modo a ter um bom desempenho em diferentes tipos de dados do mundo real. Foi implementado por Tim Peters em 2002 para uso na linguagem de programação Python (Virtanen, Gommers, Oliphant, & al, 2020). O algoritmo encontra as subsequências dos dados que já estão ordenados e usa-os para sequenciar os restantes com mais eficiência. Isto é realizado ao juntar várias execuções até serem verificados determinados critérios. Cada execução tem um tamanho mínimo, que se baseia no tamanho do *input* e é definida no início do algoritmo. Se uma execução for menor do que esse tamanho mínimo, o *insertion sort* permite correr mais elementos até o tamanho mínimo ser alcançado.

Tabela 5.1 — *Big* O *Notation - Timsort*.

Worst-case performance	O(n log n)
Best-case performance	O(n)
Average performance	O(n log n)
Worst-case space complexity	O(n)

A complexidade do pior caso só foi provada mais tarde por Auger (2018). O melhor caso acontece quando os elementos de entrada já se encontram ordenados, tornando-se num algoritmo de sequenciamento adaptável (Chandramouli & Goldstein, 2014).

5.3 - Simulação 1

A primeira simulação realizou-se em comparação com o plano de produção da HFA da semana 16, 17 e 18 do ano de 2021. O objetivo é minimizar o *makespan* e o parâmetro a avaliar é o tempo de ciclo médio por encomenda. Os resultados aqui apresentados são relativos à semana 18, que foi aquela que apresentou a menor variação deste parâmetro em relação ao plano realizado pelo departamento de planeamento da HFA.

A heurística utilizada dá prioridade às encomendas com o menor tempo de processamento entre os gargalos das operações SMT e THT.

A execução do algoritmo resulta num plano de produção com as mesmas 27 encomendas da HFA. Estas encomendas foram escolhidas previamente de modo a se poderem comparar objetivamente ambos os planos. No entanto, a ideia por trás do planeamento proposto é, por ordem:

1. Correr o MRP.
2. Selecionar todas as encomendas com datas de entrega previstas nas próximas 3 semanas.
3. Correr o algoritmo (CRP) baseado na heurística sugerida.

Esta metodologia levaria a uma melhoria do *makespan* e do cumprimento das datas de entrega dos Clientes no longo termo. Os seus resultados serão discutidos na Secção 5.4.

Considerações

1. Os tempos são referentes às operações SMT e THT, não se considerando as restantes operações de carga inferior e que não são o gargalo global na produção. Isto inclui a soldadura do Cliente do ramo automóvel, embora o seu tempo de processamento no SMT da linha 2 seja contabilizado.
2. A carga é distribuída indiferentemente pelas 3 máquinas de soldadura onda e pelas 3 máquinas de soldadura seletiva. Para isso, considera-se apenas uma máquina que processa a operação soldadura com uma cadência 6 vezes superior.
3. Começa-se por sequenciar uma encomenda com um elevado tempo de processamento. Esta encomenda é relativa à semana anterior (semana 17). Esta decisão resulta da necessidade de se truncarem algumas encomendas de modo a não serem adiadas indefinidamente e conseguir-se, assim, respeitar a restrição com as datas do Cliente. Além disso, devido à consideração anterior, as encomendas sequenciadas inicialmente verão o seu fluxo prejudicado. Na prática, esta situação seria contornada ao colocar-se a encomenda de maior tempo de processamento numa máquina, estando as outras livres para as restantes encomendas. Contudo, esta simulação é a mais afetada comparativamente ao plano da HFA, pelo que os resultados representarão um minorante da melhoria no *maskespan*.
4. O algoritmo que calcula o tempo de ciclo médio por encomenda, entre outros parâmetros, é preemptivo[25]. Além disso, embora a ordem das encomendas esteja sequenciada de forma distinta, este algoritmo irá sequenciar as operações no THT segundo a heurística do SPT, o que levará, novamente, a uma melhoria do *makespan* do plano da HFA que não se verificaria na prática.

Resultados

A comparação da distribuição normal dos tempos de ciclo por encomenda em horas entre os planos pode ser observada na Figura 5.1.

[25] Em computação, preemptividade é o ato de interromper temporariamente uma tarefa, sem precisar da sua cooperação, com a intenção de a retomar posteriormente.

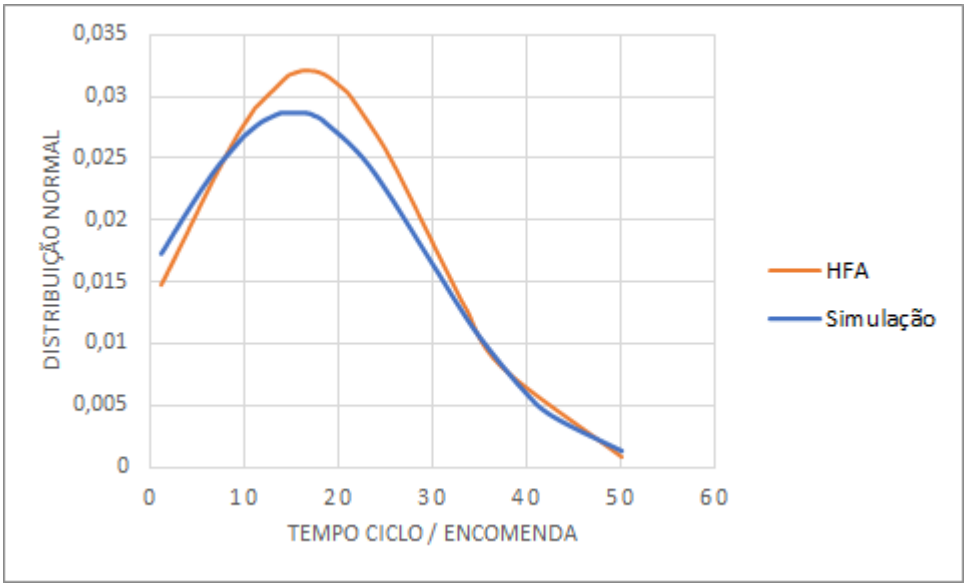

Figura 5.1 – Simulação 1 vs Plano HFA semana 18

A simulação apresenta um valor médio de 15,1h com um desvio padrão de 13,9h, enquanto o plano de produção da HFA apresenta um valor médio de 16,6h com um desvio padrão de 12,4h. Estes valores, evidenciados na Figura 5.2, representam um desempenho superior em cerca de 10% relativamente ao tempo de ciclo médio do plano simulado.

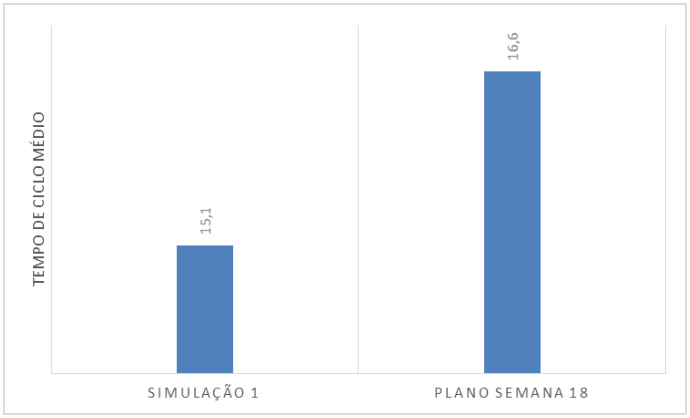

Figura 5.2 – Comparação do tempo de ciclo médio

Outros parâmetros a considerar são o *waiting time* (WT), que representa o tempo médio que uma encomenda espera para a sua operação ser inicializada no "THT", após ter sido concluída a respetiva operação no SMT; e o *turnaround time* (TAT) que, para a mesma situação, representa agora o tempo até à operação no "THT" ser finalizada, incluindo o WT.

A simulação revelou um WT e TAT médios por encomenda de 1,6h e 3,5h, respetivamente, contra 1,5h e 3,3h por parte do plano de produção da HFA. O desempenho

ligeiramente inferior para estes parâmetros por parte do plano simulado confirma a afirmação resultante da terceira consideração na Secção anterior. De facto, começar-se o sequenciamento por uma encomenda com um elevado tempo de processamento tem um impacto negativo, mais negativo no plano simulado, uma vez que haverá, de seguida, várias encomendas menores em WIP paradas. Contudo, esta situação reforça a eficácia da heurística utilizada, já que demonstrou uma melhoria significativa do tempo de ciclo médio e, consequentemente, do *makespan* num curto espaço temporal e numa situação desfavorável.

5.4 - Simulação 2

De modo a colocar-se em prática a ideia do planeamento é necessário, para além das etapas já mencionadas, otimizar-se a alocação de recursos humanos. Durante a primeira simulação, esta questão não foi um obstáculo, porque a escolha das encomendas selecionadas pelo plano da HFA, e consequentemente as da simulação, já garantia a existência de recursos humanos suficientes para cada uma das operações.

Após correr o MRP e selecionar as encomendas a sequenciar, calcula-se a carga total do conjunto das encomendas para cada operação e divide-se pelo tempo correspondente dos turnos ao longo da semana. Se o resultado ultrapassar o número de recursos humanos disponíveis para quaisquer operações, o programa segue um procedimento decidindo quais as encomendas que não serão viáveis para a semana a planear.

O procedimento toma esta decisão através do compromisso entre dois critérios: maximizar a utilização dos recursos humanos disponíveis; e maximizar o número de encomendas a planear para a semana em questão.

Como há funcionários aptos a realizar mais que uma operação, mas não é necessário especificar-se qual irá realizar determinada tarefa, são impostas restrições quer a nível do número de funcionários disponíveis para uma dada tarefa, quer ao nível total de funcionários na HFA.

Além disso, há casos em particular onde é necessário atribuir mais funcionários para o gargalo ser a máquina. Por exemplo, se o tempo de assemblagem de uma encomenda for pelo menos 25% superior ao respetivo tempo de soldadura seletiva, aloca-se mais um funcionário a esta última operação, que é a dependente da cadência da máquina. Os “25%” foram usados como critério (podendo este ser alterado) após ter sido comparada a melhoria na cadência com o custo acrescentado.

Assim, o programa indica o número de recursos humanos necessários para cada operação e escolhe o conjunto de encomendas ideal a sequenciar de acordo com a heurística selecionada.

Resultados

Utilizando a mesma heurística da Secção anterior, simulou-se o planeamento proposto já descrito e comparou-se com o planeamento realizado pela HFA. Os resultados são referentes à semana 21, embora a simulação também sequencie encomendas para as próximas semanas, tendo em conta a matéria-prima disponível.

Como o *makespan* de ambos é fixo numa semana, o parâmetro a avaliar é o número de encomendas realizadas. Na Figura 5.3 é apresentado o número de encomendas por linha para cada um dos planos.

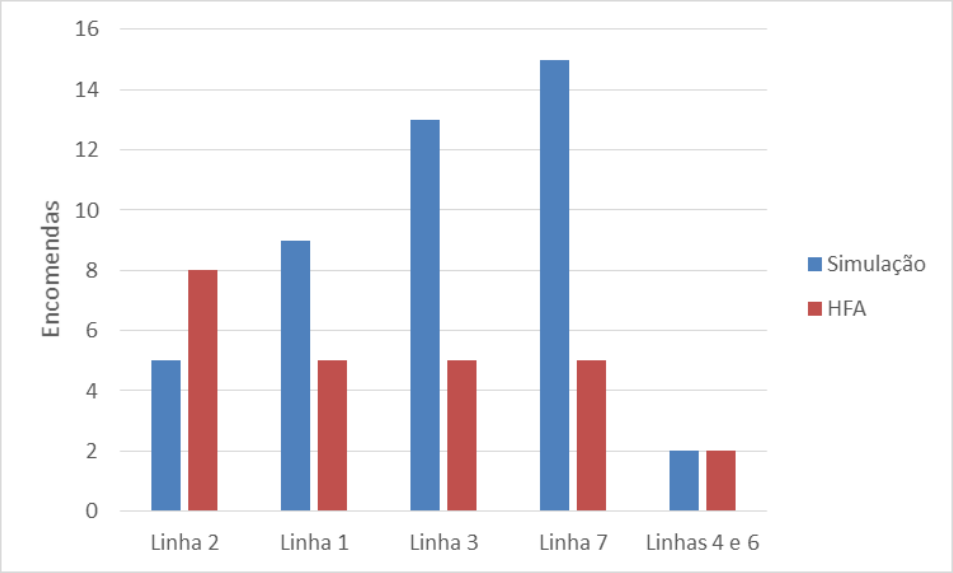

Figura 5.3 – Simulação 2 vs Plano HFA semana 21

Durante esta semana, a linha 5 não estava funcional. Note-se também que as encomendas realizadas nas linhas 4 e 6 são as mesmas, devido à restrição da data do Cliente de telecomunicações. O total de encomendas realizadas pela simulação foi de 44 em comparação com as 25 pela HFA

Capítulo 6

Conclusão

Neste capítulo apresentam-se as conclusões em relação aos resultados obtidos e os objetivos que foram cumpridos. De seguida referem-se as próximas etapas deste projeto e, na secção final, são resumidas as considerações finais.

6.1 - Conclusões

Através do planeamento proposto conseguiu-se, para o mesmo espaço temporal de uma semana, restrições e seleção de encomendas com datas de entrega urgentes, um aumento em 75% do número de encomendas realizadas pelo planeamento da HFA.

Embora nas semanas subsequentes seja de esperar que este desempenho não se mantenha, uma vez que em parte esta melhoria resulta da escolha de encomendas de carga menor, foi mostrado na primeira simulação que, para as mesmas encomendas e numa situação desfavorável, esta continua a melhorar em cerca de 10% o tempo de ciclo médio por encomenda.

Foi cumprido o objetivo da TOC: ao aumentar-se o fluxo da produção, ter-se-á um crescimento equivalente em termos de fluxo de caixa. Em relação à filosofia *Lean* e a uma das importantes restrições do problema: o cumprimento das datas de entrega e consequente satisfação do Cliente, espera-se também conseguir um progresso significativo no longo termo.

Além disso, ao sequenciar-se antecipadamente as encomendas para as próximas semanas, e ao automatizar-se dinamicamente o planeamento da produção, é possível realizar-se uma análise mais eficaz da matéria-prima a encomendar, das ordens de fabrico a submeter e das datas previstas com os Clientes.

6.2 - Recomendações futuras

As próximas etapas passam por melhorias em questões logísticas ao nível do armazém e, em particular, na ponderação de variáveis como o custo de posse de *stock* vs o custo de *setup*. A otimização e o planeamento dinâmico relativamente a estas variáveis são feitos através de algoritmos de dimensionamento de lotes.

Além disso, discute-se um Modelo baseado em Sistemas Adaptativos Complexos (CAS) diversas perspetivas para tornar os sistemas de *machine learning* (ML) mais justos.

6.2.1 - Dimensionamento de lotes

O modelo de dimensionamento de lote dinâmico é uma generalização do modelo da quantidade económica de encomenda (EOQ) que leva em consideração que a procura pelo produto varia ao longo do tempo (Wagner & Whitin, 1958).

Uma boa solução pode ser encontrada através da formulação em programação inteira que se segue e resolvida pelo algoritmo Simplex, Kashipur (2021).

Variáveis a considerar

S_t: custo de set-up
I_t: custo de posse de *stock* no período t
D_t: procura no período t
X_t: quantidade produzida no período t
Y_t: variável binária igual a 1 se e só se $X_t > 0$
M: número "muito grande"
E_t: *stock* terminado no período t transportado para o próximo período
P: custo de produção por unidade

Função Objetivo

$$Min: \sum_{t=1}^{n} [P.X_t + I_t.E_t + Y_t.S_t] \tag{6.1}$$

Onde o primeiro termo é o custo de produção, o segundo o custo de posse de *stock* e o terceiro o custo de *setup*.

Restrições

$M.Y_t >= X_t$	não há custo de *setup* num período em que a produção é nula
$X_t >= 0$	a quantidade produzida não pode ser negativa
$I_t >= 0$	o stock final não pode ser negativo
$X_t = INT$	a quantidade produzida deve ser um número inteiro
$Y_t = BINARY$	a variável Y_t é binária

O algoritmo de Wagner e Whitin encontra uma solução ótima por programação dinâmica.
Começar com $t^* = 1$:

1. Considerar os períodos a ordenar t^{**}, $t^{**} = 1, 2, \cdots, t^{*}$ e satisfazer as procuras d_t, $t = t^{**}, t^{**}+1, \cdots, t^{*}$ por esta ordem
2. Adicionar $H(X_{t^{**}})\, S_{t^{**}} + i_{t^{**}} I_{t^{**}}$ aos custos ótimos para os períodos de 1 a $t^{**}-1$ determinados na última iteração do algoritmo
3. Destas t^{*}alternativas, selecionar a combinação com o custo mínimo para os períodos de 1 a t^{*}
4. Seguir para o período $t^{*}+1$ (ou parar se $t^{*} = N$)

Onde $H()$ é a função de Heaviside. Contudo, este método é demasiado complexo computacionalmente.

A heurística de Silver-Meal, criada por Edward Silver e Harlan Meal, é uma variante da EOQ que funciona como aproximação do algoritmo de Wagner-Whitin. É um método progressivo que requer a determinação do custo médio por período em função do número de períodos a abranger (Tersine, 1988).

Sendo $C(n)$ a média entre o custo de posse e o custo de *setup* por período t nos próximos n períodos tem-se como procedimento geral:

$$C(j) = \frac{S + ID_2 + 2ID_3 + \cdots + (j-1)ID_j}{j} \qquad (6.2)$$

A procura pelo n ótimo continua até $C(n) > C(n-1)$. Quando $C(j) > C(j-1)$, fazer $D_1 + D_2 + \cdots + D_{j-1}$ e iterar-se novamente a partir do período j.

6.2.2 - *Machine Learning*

O uso de algoritmos baseados em Machine Learning (ML) é, frequentemente, objeto de diversas discussões nos últimos anos. Embora seja promissor na resolução de problemas como o JSSP, a sua eficácia atual num sistema aberto com dados estocásticos e em tempo limitado é sobrestimada. Nesta Secção, não se investiga os fundamentos de matemática e da ciência da computação. Apresentam-se, sim, os conceitos atuais de um Modelo baseado em Sistemas Adaptativos Complexos (CAS) (Oneto & Navarin, 2020). Os tópicos a ser investigados no futuro são a Formação da Teoria Causal (CCTF) que poderá incorporar, através de Sistemas Dinâmicos baseados na Comunidade (CBSD), diversas perspetivas para tornar os sistemas ML mais justos.

Modelo baseado em Sistemas Adaptativos Complexos (CAS)

Os sistemas CAS[26] têm origens na Teoria Geral dos Sistemas (Von Bertalanffy, 1950), que surgiu na década de 1950 como uma abordagem interdisciplinar coesa para estudar sistemas

[26] CAS é uma disciplina ampla e profunda com muitos aspetos não abordados nesta tese. Por exemplo, auto-organização, comportamento caótico, comportamento *fat-tailed* e escalas em relações *power law* - elementos-chave do CAS - não são necessários para a introdução dos seus elementos em CBSD

em todos os campos da ciência. Estes sistemas adaptativos são complexos na medida em que os seus componentes não são interligados numa rede causal, e o comportamento do sistema não pode ser previsto apenas com base no comportamento destes; e são adaptáveis no sentido de que se adaptam às mudanças no seu ambiente por mutação ou auto-organização das suas estruturas internas.

Tipos de elementos-chave de CAS

Taxonomia é a disciplina biológica que define os grupos de organismos com base em características comuns e atribui um nome e nota para cada um. Os grupos podem ser agregados para formar um supergrupo de maior pontuação, criando uma classificação hierárquica. Os grupos criados por este processo são referidos como taxa.

Distinguir estes tipos permite desenvolver uma representação mais rica para o contexto societal que, por exemplo, separa objetivos de **agentes**, dos mecanismos de **preceitos**, e resultados manifestados como **artefactos**. No entanto, é importante perceber que estas distinções nem sempre são rígidas, uma vez que, em alguns casos, há incorporação de propriedades de mais de um tipo de elemento.

Modelo Taxonómico do Contexto Societal

Desde a sua introdução na década de 1960, os CAS têm sido utilizados para modelar sistemas sociais como redes de cadeia de abastecimento (Thomas Y Choi, 2001).

A Figura 6.1 (Matin Jr. & al, 2020) descreve a relação taxonómica entre os principais tipos de elementos que compõem o contexto social a partir de uma perspetiva dos CAS. Os preceitos conduzem e restringem o comportamento dos agentes e são refletidos nos artefactos que resultam desse comportamento. Por sua vez, os preceitos são influenciados pelos artefactos aos quais são expostos, resultando em ciclos de feedback que contribuem para a natureza dinâmica e complexa do contexto social (Sterman, 2000).

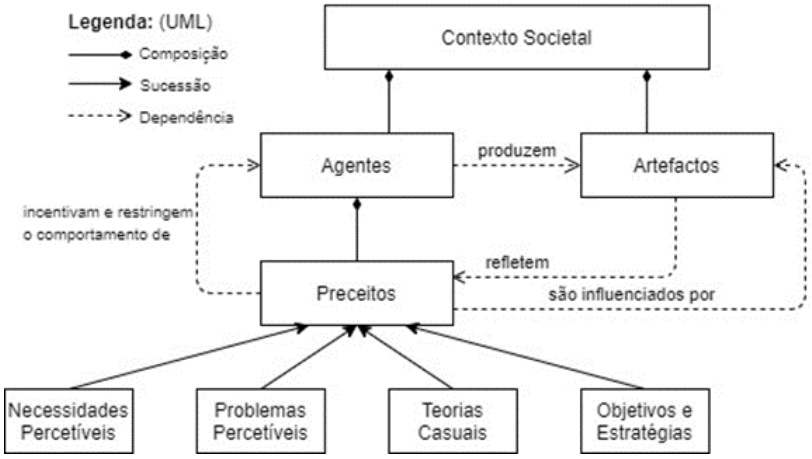

Figura 6.1 – Modelo Taxonómico – Perspetiva dos CAS

Qualquer abordagem para incluir o contexto social nos sistemas de ML deve ser centrada na identificação e representação de preceitos humanos. Em particular, os modelos base do processo de decisão humana daqueles que financiam, constroem, utilizam e são afetados pelos produtos dos sistemas de ML devem ser considerados como endógenos ao próprio processo. Por exemplo, ignorar-se o papel da decisão humana (como os de juízes, réus e as suas famílias) num tribunal, levaria a resultados injustos.

Surgem quatro preceitos-chave que são essenciais para se compreender o contexto social que cerca os sistemas/produtos de ML: *Perceived Needs, Perceived Problems, Casual Theories e Goals & Strategies*. Na realidade, estes são interdependentes, sobrepostos e mutuamente influentes. Além disso, são apoiados e influenciados pelos valores, emoções, preconceitos e estereótipos mantidos pelo agente.

6.3 - Considerações finais

A dissertação realizada é a primeira etapa de um projeto estimado para dois anos. As próximas etapas introduzirão ferramentas estocásticas e a análise do lucro e do risco associado.

Os algoritmos de dimensionamento de lotes apresentados contribuirão para a previsão da procura determinística discreta ao calcular as quantidades ótimas por encomenda com consequente diminuição dos custos operacionais.

A análise do risco só será possível através de uma análise estatística à dispersão de múltiplas variáveis com recurso a algoritmos complexos. Não obstante, chama-se a atenção que quaisquer algoritmos não deverão substituir experiência e o contexto humano (Bishop, 2006).

Referências

A Baykasoğlu, F. K. (2017). Solving comprehensive dynamic job shop scheduling problem by using a GRASP-based approach. *International Journal of Production Research , 55*(11), 3308-3325.

Ahmadi, J. H., Ahmadi, R. H., Dasu, S., & Tang, C. S. (1992). Batching and scheduling jobson batch and discrete processors. *Operations research, 40*(4), 750–763.

Armentano, V. A., & Ronconi, D. P. (1999). Tabu search for total tardiness minimization in flowshop scheduling problems. *Computers & operations research, 26*(3), 219–235.

Auger, N., Jugé, V., Nicaud, C., & Pivoteau, C. (2018). On the Worst-Case Complexity of TimSort. *arXiv:1805.08612*.

Baker, K. R. (1984). Sequencing rules and due-date assignments in a job shop. *Management science, 30*(9), 1093–1104.

Baker, K. R., & Scudder, G. D. (1990). Sequencing with earliness and tardiness penalties: a review. *Operations Research, 38*, 22-36.

Bishop, C. (2006). *Pattern Recognition and Machine Learning*. New York: Springer-Verlag.

Carroll, D. C. (1965). *Heuristic sequencing of single and multiple component jobs*. MIT. School of Industrial Management; Sloan School of Management: Massachusetts Institute of Technology.

Chakraborty, S., & Bhowmik, S. (2013). Job shop scheduling using simulated annealing. *First International Conference on Computation and Communication Advancement*, 69-73.

Chandramouli, B., & Goldstein, J. (2014). Patience is a Virtue: Revisiting Merge and Sort on Modern Processors. *International Conference on Management of Data* . ACM SIGMOD.

Dias, J. (2005). *Logística Glogal e Macrologística*. Lisboa: Edições Sílabo.

Etiler, O., Toklu, B., Atak, M., & Wilson, J. (2004). A genetic algorithm for flow shop scheduling problems. *Journal of the Operational Research Society, 55*(8), 830–835.

Faaland, B., & Schmitt, T. (1987). Scheduling tasks with due dates in a fabrication/assembly process. *Operations Research, 35*(3), 378–388.

Feo, T. A., & Resende, M. G. (1989). A probabilistic heuristic for a computationally difficult set covering problem. *Operations Research Letters, 8*(2), 67-71.

Ghassemi-Tari, F., & Olfat, L. (2004). Two covert based algorithms for solving the generalized flow shop problems. *Proceedings of the 34th International Conference on Computers and Industrial Engineering*, *34*.

Ghassemi-Tari, F., & Olfat, L. (2008). Covert based algorithms for solving the generalized tardiness flow shop problems. *Journal of Industrial and Systems Engineering*.

Goldratt, E. M. (1984). *The Goal*. North River Press.

Goldratt, E. M. (1997). *Critical Chain*. North River Press.

Hall. (1991). Earliness-tardiness scheduling problems, i: weighted deviation of completion times about a common due date. *Operations Research, 39*(5), 836–846.

Jacobs, F. R., Chase, R. B., & Lummus, R. R. (2014). *Operations and supply chain management*. New York: McGraw-Hill/Irwin .

Johnson, S. M. (1954). Optimal two- and three-stage production schedules with setup times included. *Naval Research Logistics Quarterly*, 61-68.

Kanet, J. J. (1981). Minimizing the average deviation of job completion times about a common due date. *Naval Research Logistics Quarterly, 28*(4), 643–651.

Karimi, I. A. (1992). Optimal cycle times in multistage serial systems with set-up and inventory costs. *Management science, 38*(10), 1467–1481.

Kashipur, I. (10 de 2 de 2021). *The IIM Kashipur Blog*. Obtido em 10 de 4 de 2021, de https://iimkashipurblog.in/page/3/

Kim, Y.-D. (1993). Heuristics for flowshop scheduling problems minimizing mean tardiness. *Journal ofthe Operational Research Society, 44*(1), 19–28.

Lauff, V., & Werner, F. (2004). On the complexity and some properties of multi-stage scheduling problems with earliness and tardiness penalties. *Computers & Operations Research, 31*(3), 317–345.

Lawler, E. L. (1994). Knapsack-like scheduling problems, the Moore-Hodgson algorithm and the 'tower of sets' property. *Mathematical and Computer Modelling, 20*(2), 91–106.

Lawrence, S. (1984). Resource Constrained Project Scheduling: An Experimental Investigation of Heuristic Scheduling Techniques. Graduate School of Industrial Administration, Carnegie-Mellon University, Pittsburgh, Pennsylvania.

Lee, I. (2001). Artificial intelligence search methods for multi-machine two-stage scheduling with due date penalty, inventory, and machining costs. *Computers & Operations Research, 28*(9), 835–852.

Matin Jr., D., & al, e. (2020). Extending the machine learning abstraction boundary: A Complex systems approach to incorporate societal context. *Computer Science* , arXiv:2006.09663 .

Morton, T. E., & Rachamadugu, R. V. (1983). Myopic heuristics for the single machine weighted tardiness problem. *Adaptive Agents and Intelligent Robotics*.

Mosheiov, G. (2003). Scheduling unit processing time jobs on an m-machine flow-shop. *Journal of the Operational Research Society, 54*(4), 437–441.

Oneto, L., & Navarin, N. (2020). Fairness in Machine Learning. *Recent Trends in Learning From Data* (pp. 155-196). Springer.

Papadimitriou, C., & Steiglitz, K. (1982). *Combinatorial optimization: algorithms and complexity*. United States: Prentice-Hall.

Pisinger, D. (2003). Where are the hard knapsack problems? *Technical Report, Department of Computer Science, University of Copenhagen*.

Rachamadugu, R. (1984). Myopic heuristic in open shop scheduling. *Proceedings of the 14 Annual Pittsburgh Conference on Modeling & Simulation, 14*, pp. 1245–1250. Pittsburgh.

Santos, H. G., & Toffolo, T. A. (2019). Tutorial de desenvolvimento de métodos de programação linear inteira mista em python usando o pacote python-mip. *Pesquisa Operacional para o Desenvolvimento, 11*(3), 127–138.

Sterman, J. D. (2000). *Business dynamics: Systems thinking and modeling for acomplex world*. McGraw-Hill.

Sundararaghavan, & Ahmed, M. U. (1984). Minimizing the sum of absolute lateness in single-machineand multimachine scheduling. *Naval Research Logistics, 31*(2), 325–333.

Tersine, R. J. (1988). *Instructor's Manual to Principles of Inventory and Materials Management*. North-Holland.

Thomas Y Choi, K. J. (2001). Supply networks and complex adaptive systems: control versus emergence. *Journal of operations management, 19*(3), 351–366.

Tobias, D., & John, M. (1930). *Number: The language of science*.

Vepsalainen, A. P., & Morton, T. E. (1987). Priority rules for job shops with weighted tardiness costs. *Management science, 33*(8), 1035–104.

Virtanen, P., Gommers, R., Oliphant, T. E., & al, e. (2020). SciPy 1.0: fundamental algorithms for scientific computing in Python. *Nature Methods, 17*(3), 261–272.

Von Bertalanffy, L. (1950). An outline of general system theory. *British Journal for the Philosophy of science*.

Wagner, H. M., & Whitin, T. M. (1958). Dynamic version of the economic lot size model. *Management science, 5*(1), 89-96.

Wantuck, K. (1983). *The japanese approach to productivity*. FabTech International.

Womack, J. P., Jones, D. T., & Roos, D. (1990). *The machine that changed the world: The story of lean production*. New York: Rawson Associates.

Yeh, W.-C., & Allahverdi, A. (2006). A branch-and-bound algorithm for three-machine flowshop scheduling problem to minimize total completion time with separate setup times. *European Journal of Operational Research , 169*, 767-780.

ANEXO A

Planta do chão de fábrica

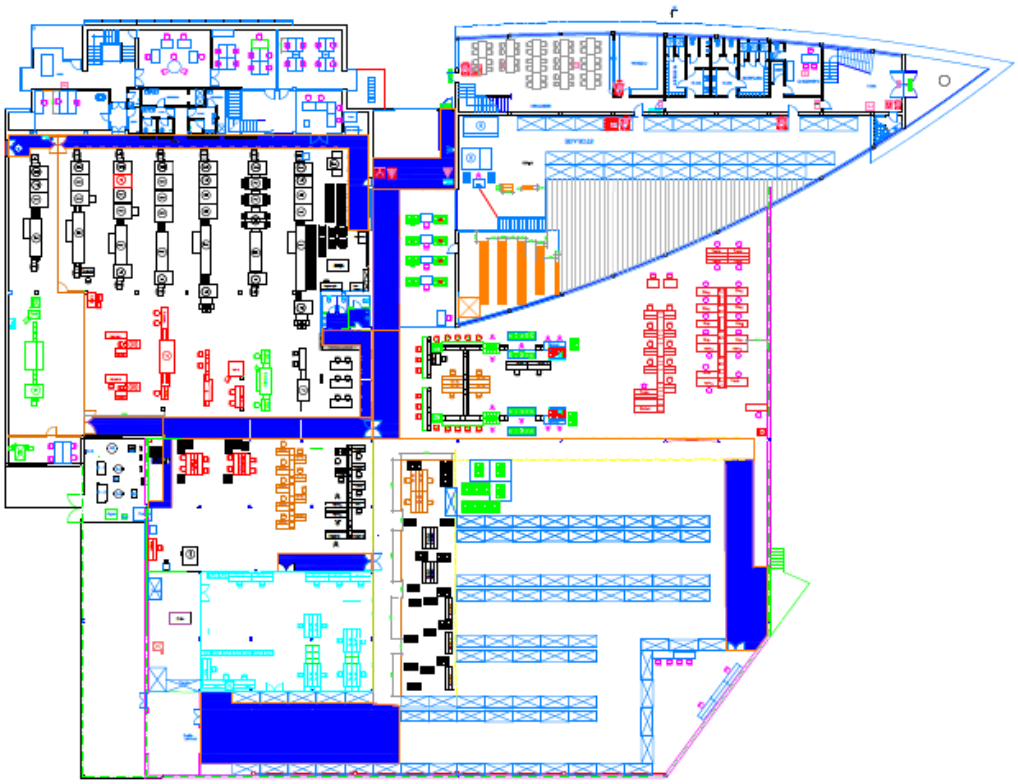

Figura Anexo A.1 – Visão geral

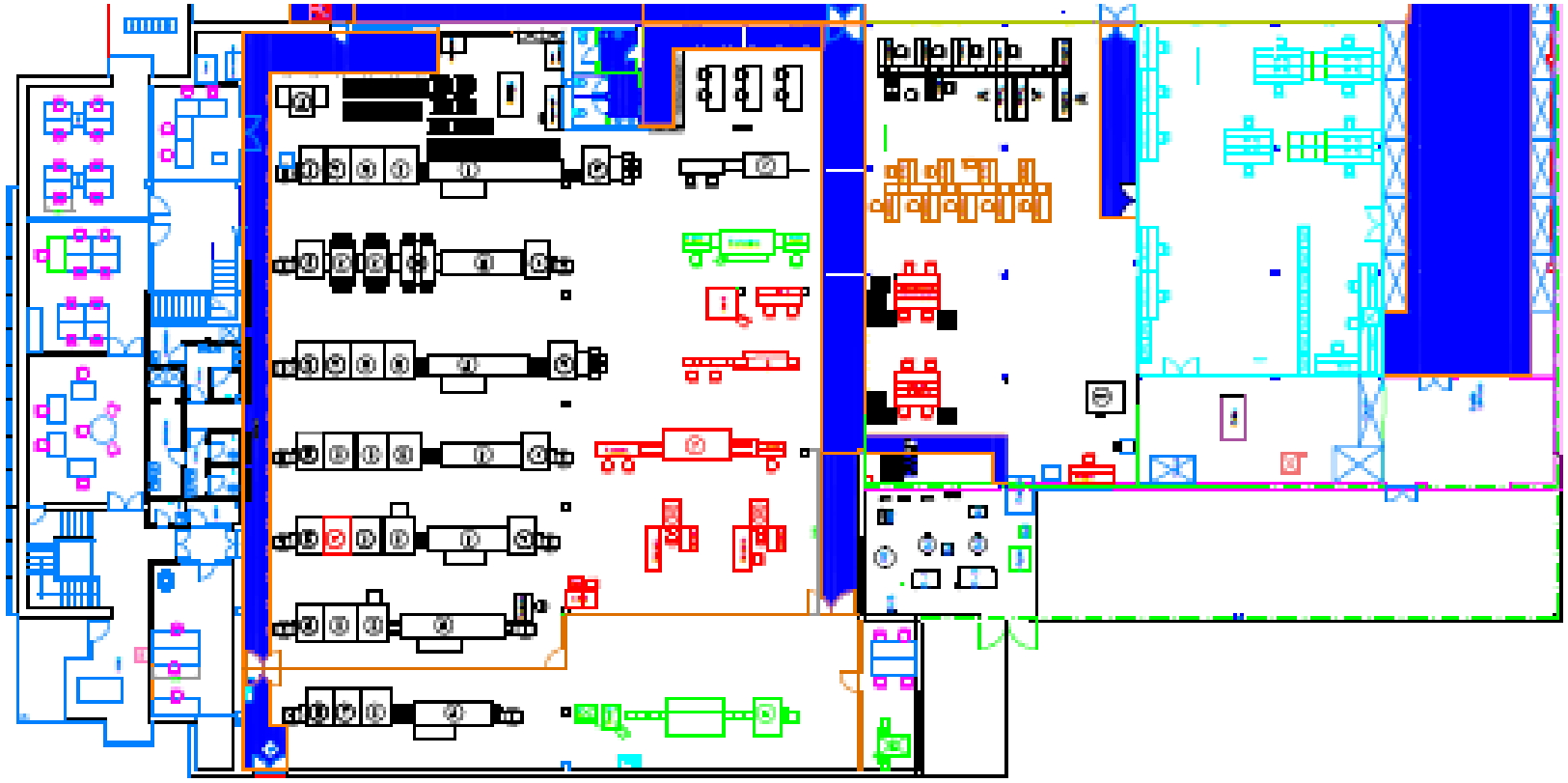

Figura Anexo A.2 –Linhas de Produção

ANEXO B

VSM – *Value Stream Mapping*

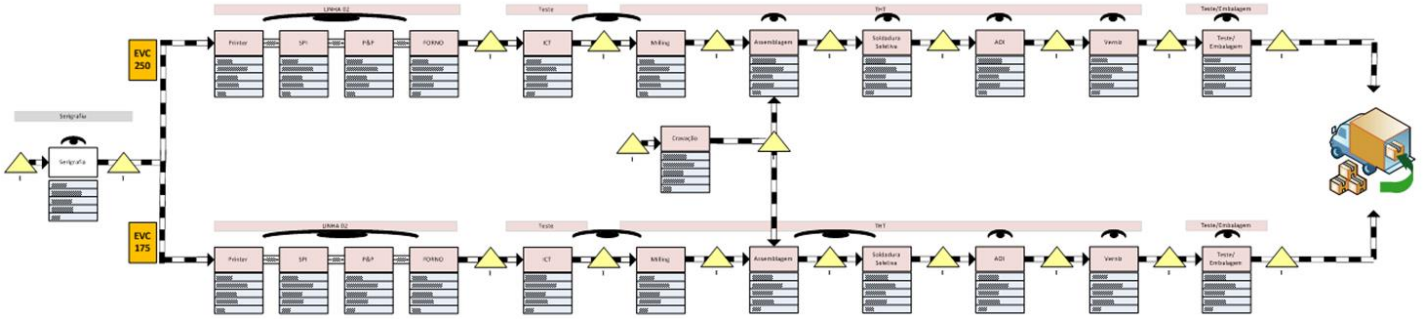

Figura Anexo B.1 –Linha 2

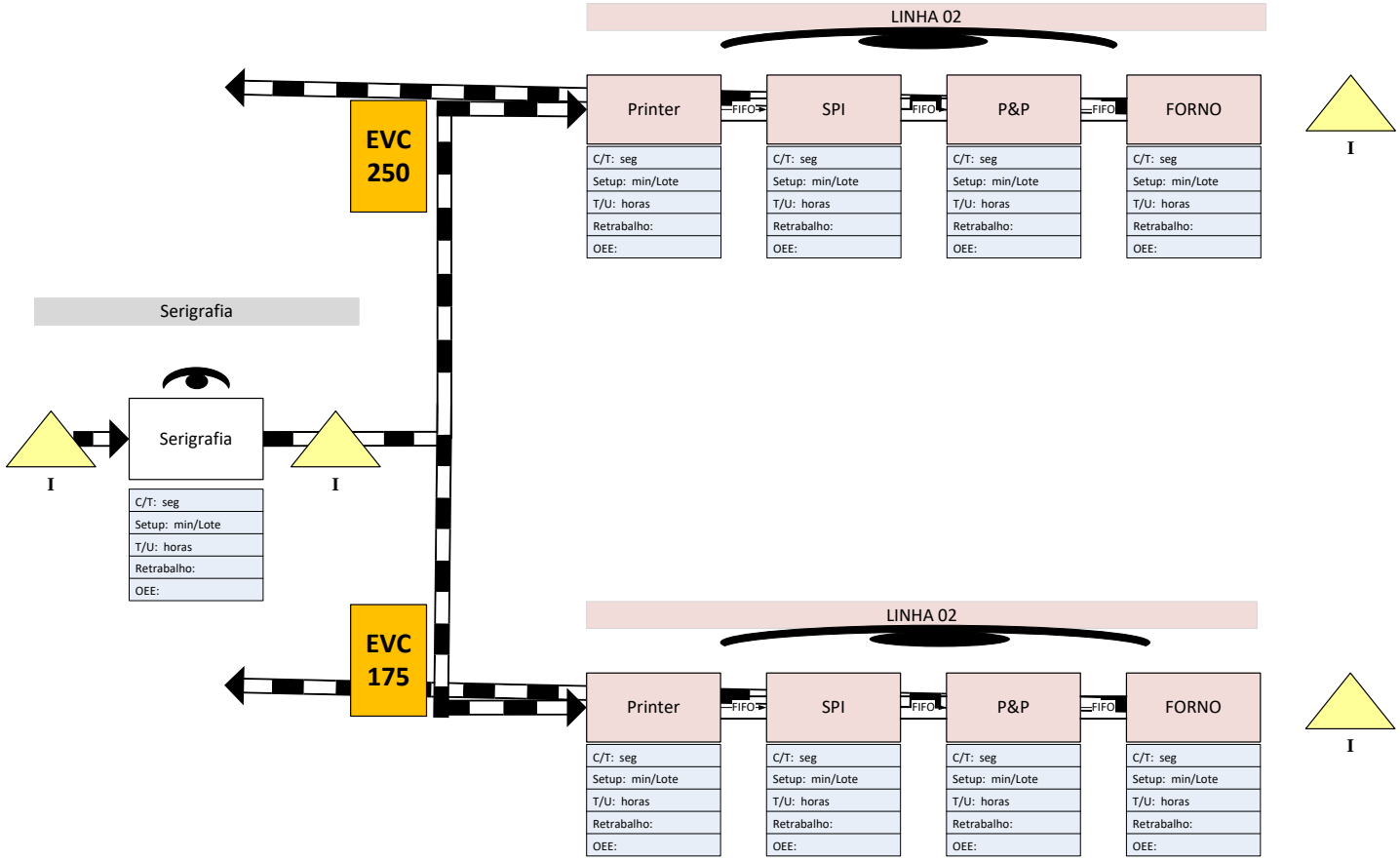

Figura Anexo B.2 – Linha 2 – 1/3

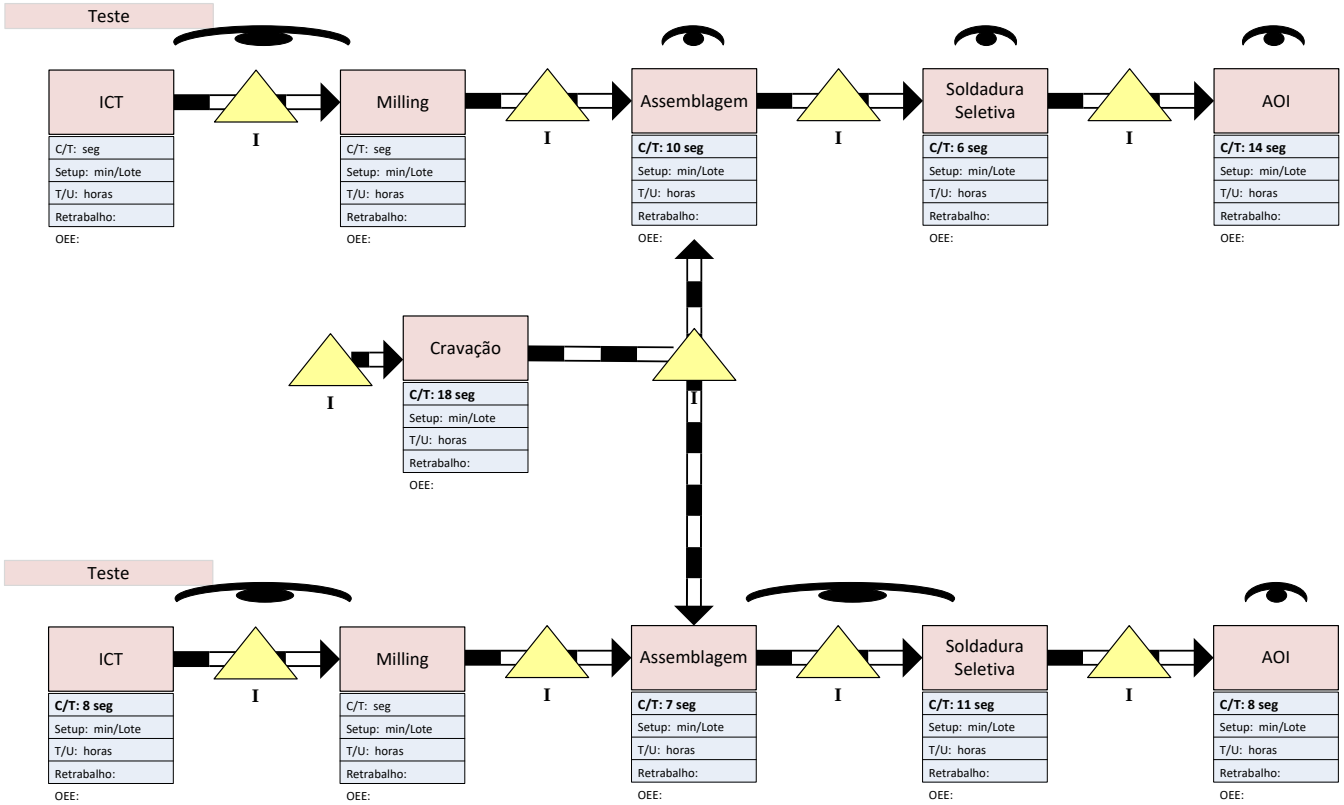

Figura Anexo B.3 – Linha 2 – 2/3

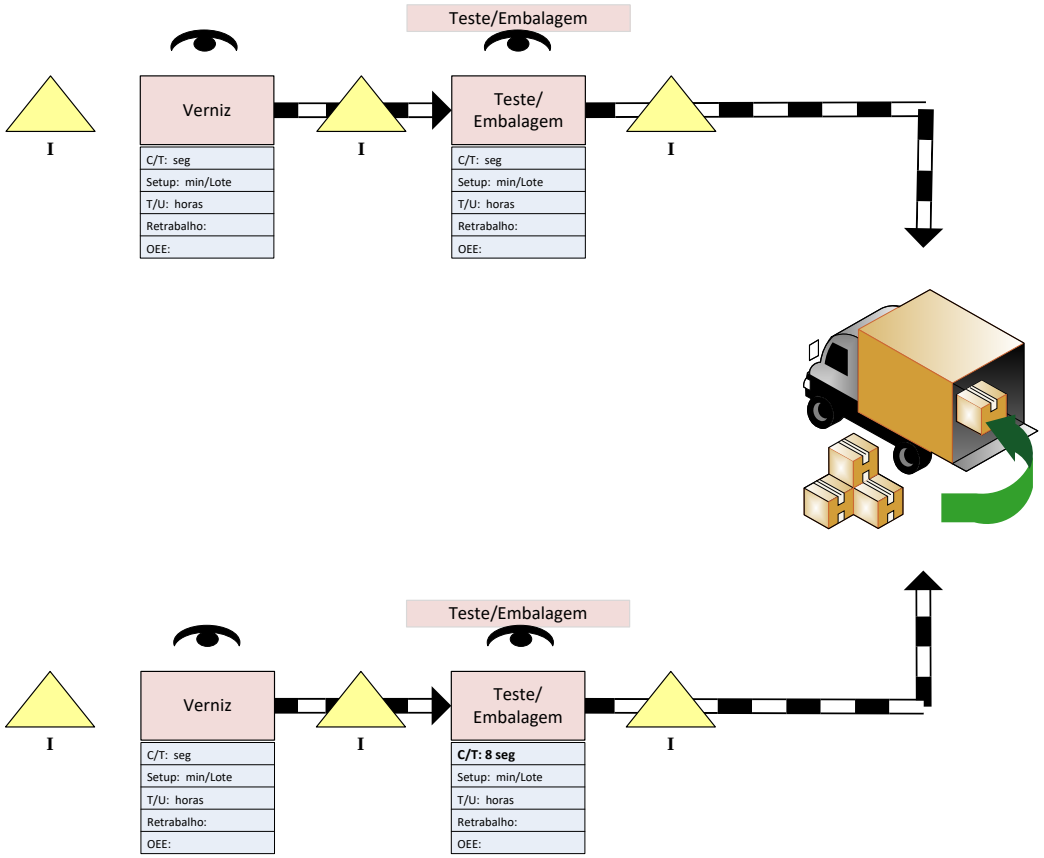

Figura Anexo B.4 – Linha 2 – 3/3

ANEXO C

Código fonte do programa desenvolvido

Diagrama de Sequência

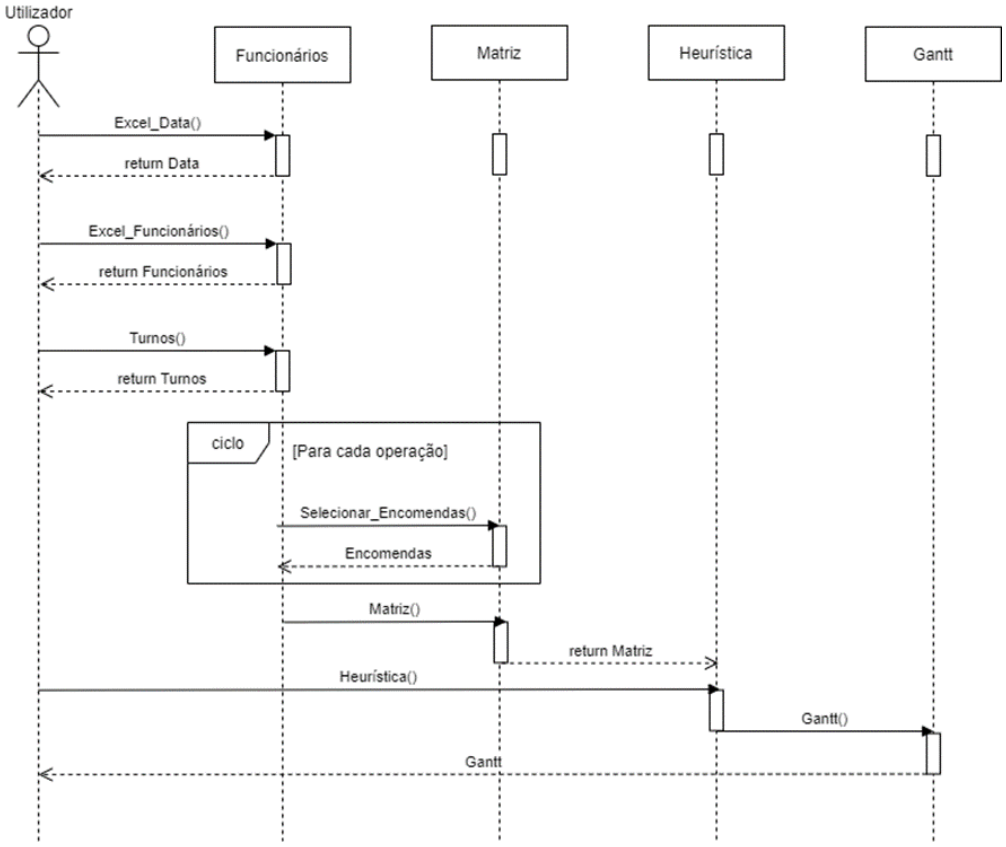

Figura Anexo C.1 – Diagrama de sequência

```
#* início do programa Planeamento da produção e alocação de RH

#*************************
#     Ler Excel Data
#*************************
import numpy as np
from openpyxl import Workbook
from openpyxl import load_workbook
import pandas as pd
from google.colab import files

uploaded = files.upload()

for fn in uploaded.keys():
   print('User uploaded file "{name}" with length {length} bytes'.format(
       name=fn, length=len(uploaded[fn])))

df = pd.read_excel ('Data11 - plano tese.xls')

m = 8 #6 linhas (excluindo l.4) + Onda e Seletiva

REF = df['#REF']
EDD = df['Data']
Welding = df['Welding']
Time_THT = df['Time THT']
Programaçao = df['Programaçao']
Testes = df['Testes']
Embalagem = df['Embalagem']
Soldadura_Manual = df['Soldadura Manual']
Assemblagem = df['Assemblagem']

#*************************
#   Ler Excel Pessoas
#*************************
uploaded = files.upload()

for fn in uploaded.keys():
   print('User uploaded file "{name}" with length {length} bytes'.format(
       name=fn, length=len(uploaded[fn])))

wb = load_workbook("Pessoas.xlsx", data_only=True)
sheet3=wb['Pessoas']
P_Onda = sheet3.cell(row=1, column=1).value
P_Seletiva = sheet3.cell(row=1, column=2).value
P_Soladadura_Manual = sheet3.cell(row=1, column=6).value
P_THT = sheet3.cell(row=1, column=7).value
P_Programaçao = sheet3.cell(row=1, column=3).value
```

Código fonte do programa desenvolvido

```
P_Testes = sheet3.cell(row=1, column=4).value
P_Embalagem = sheet3.cell(row=1, column=5).value

#************************
#         Turnos
#************************
aux_index = (2, 1, 3, 6, 7)
vec_turnos = np.zeros([len(aux_index)])
j=0

#***********************************
#  Restrição: Pessoas >= Necessidades
#***********************************
import math

aux_sum = 0 #necessidade total
aux_max = []
cut = 0 #índice da encomenda a cortar
aux_value_cut = [] #necessidade para cada encomenda a cortar
drop = []
i_master = np.zeros([len(Welding)])
turno = 7.5*5 #horas de um turno por semana
P_Assemblagem_n = 0

for i in range(len(Welding)):
  i_master[i] = i
  drop.append(i)

#Embalagem
for i in i_master:
    aux_sum += Embalagem[i]

while aux_sum/turno > P_Embalagem:

  for i in i_master:
      aux_max.append(Embalagem[i])
      if Embalagem[i]/turno > aux_sum/turno - P_Embalagem:
        aux_value_cut.append(Embalagem[i])

  if aux_value_cut == []:
    aux_sum -= max(aux_max)
    cut = df[Embalagem == max(aux_max)].index.values
  if aux_value_cut != []:
    aux_sum -= min(aux_value_cut)
    cut = df[Embalagem == max(aux_value_cut)].index.values

  for j in range(len(cut)):
    i_master = i_master[i_master != cut[j]]
  aux_max = []
```

```
  aux_value_cut = []

aux_sum = 0

#Testes
for i in i_master:
    aux_sum += Testes[i]

while aux_sum/turno > P_Testes:

  for i in i_master:
      aux_max.append(Testes[i])
      if Testes[i]/turno > aux_sum/turno - P_Testes:
        aux_value_cut.append(Testes[i])

  if aux_value_cut == []:
    aux_sum -= max(aux_max)
    cut = df[Testes == max(aux_max)].index.values
  if aux_value_cut != []:
    aux_sum -= min(aux_value_cut)
    cut = df[Testes == max(aux_value_cut)].index.values

  for j in range(len(cut)):
    i_master = i_master[i_master != cut[j]]
  aux_max = []
  aux_value_cut = []

aux_sum = 0

#Programaçao
for i in i_master:
    aux_sum += Programaçao[i]

while aux_sum/turno > P_Programaçao:

  for i in i_master:
      aux_max.append(Programaçao[i])
      if Programaçao[i]/turno > aux_sum/turno - P_Programaçao:
        aux_value_cut.append(Programaçao[i])

  if aux_value_cut == []:
    aux_sum -= max(aux_max)
    cut = df[Programaçao == max(aux_max)].index.values
  if aux_value_cut != []:
    aux_sum -= min(aux_value_cut)
    cut = df[Programaçao == max(aux_value_cut)].index.values

  for j in range(len(cut)):
    i_master = i_master[i_master != cut[j]]
```

Código fonte do programa desenvolvido

```
  aux_max = []
  aux_value_cut = []

aux_sum = 0

#Onda
for i in i_master:
  if Welding[i] == 8 and str(REF[i])[:4] != "5751":
    aux_sum += Time_THT[i]

while aux_sum/turno > P_Onda:

  for i in i_master:
    if Welding[i] == 8 and str(REF[i])[:4] != "5751":
      aux_max.append(Time_THT[i])
      if Time_THT[i]/turno > aux_sum/turno - P_Onda:
        aux_value_cut.append(Time_THT[i])

  if aux_value_cut == []:
    aux_sum -= max(aux_max)
    cut = df[Time_THT == max(aux_max)].index.values
  if aux_value_cut != []:
    aux_sum -= min(aux_value_cut)
    cut = df[Time_THT == max(aux_value_cut)].index.values


  for j in range(len(cut)):
    i_master = i_master[i_master != cut[j]]
  aux_max = []
  aux_value_cut = []

P_Onda_n = math.ceil(aux_sum/turno)
aux_sum = 0

#Seletiva
for i in i_master:
  if Welding[i] == 9 and str(REF[i])[:4] != "5751":
    aux_sum += Time_THT[i]
    if Welding[i] < 0.75*Assemblagem[i]: #Se o tempo na Assemblagem for >> (~25%) que a Seletiva, é ne
cessária uma pessoa extra
      P_Assemblagem_n += 1

while aux_sum/turno > P_Seletiva:

  for i in i_master:
    if Welding[i] == 9 and str(REF[i])[:4] != "5751":
      aux_max.append(Time_THT[i])
      if Time_THT[i]/turno > aux_sum/turno - P_Seletiva:
        aux_value_cut.append(Time_THT[i])
```

```
  if aux_value_cut == []:
    aux_sum -= max(aux_max)
    cut = df[Time_THT == max(aux_max)].index.values
  if aux_value_cut != []:
    aux_sum -= min(aux_value_cut)
    cut = df[Time_THT == max(aux_value_cut)].index.values

    for j in range(len(cut)):
    i_master = i_master[i_master != cut[j]]
  aux_max = []
  aux_value_cut = []

P_Seletiva_n = math.ceil(aux_sum/turno)
aux_sum = 0

#Soldadura Manual
for i in i_master:
    aux_sum += Soldadura_Manual[i]

while aux_sum/turno > P_Soladadura_Manual:

  for i in i_master:
      aux_max.append(Soldadura_Manual[i])
      if Soldadura_Manual[i]/turno > aux_sum/turno - P_Soladadura_Manual:
        aux_value_cut.append(Soldadura_Manual[i])

  if aux_value_cut == []:
    aux_sum -= max(aux_max)
    cut = df[Soldadura_Manual == max(aux_max)].index.values
  if aux_value_cut != []:
    aux_sum -= min(aux_value_cut)
    cut = df[Soldadura_Manual == max(aux_value_cut)].index.values

    for j in range(len(cut)):
    i_master = i_master[i_master != cut[j]]
  aux_max = []
  aux_value_cut = []

P_Soldadura_Manual_n = math.ceil(aux_sum/turno)

P_THT_n = 2*P_Onda_n + P_Seletiva_n + P_Soldadura_Manual_n + P_Assemblagem_n

aux_Embalagem = 0
aux_Programaçao = 0
aux_Testes = 0
for i in i_master:
  aux_Embalagem += Embalagem[i]
  aux_Programaçao += Programaçao[i]
```

```
  aux_Testes += Testes[i]
P_Embalagem_n = math.ceil(aux_Embalagem/turno)
P_Programaçao_n = math.ceil(aux_Programaçao/turno)
P_Testes_n = math.ceil(aux_Testes/turno)

n = len(i_master)

#************************************
# Criar matriz SMD [times], THT [times2]
#************************************
times = np.zeros([n, m-2])
times2 = np.zeros([n, m-2])
sum_times = np.zeros(m-2)
SPT = np.zeros([n, m-2])

k = 0
for i in i_master:
  for j in range(m-4):
    if j == df['Line'][i] - 1:
      times[k][j] = df['Time SMD'][i]
      times2[k][j] = df['Time THT'][i]
      SPT[k][j] = max(times[k][j], times2[k][j])
  for j in range(m-4, m-2):
    if j == df['Line'][i] - 2:
      times[k][j] = df['Time SMD'][i]
      times2[k][j] = df['Time THT'][i]
      SPT[k][j] = max(times[k][j], times2[k][j])
  k += 1


#************************************
# Redimensionar Vetores
#************************************

print(REF)
REF_m = []
EDD_m = []
Welding_m = []
for i in i_master:
  REF_m.append(REF[i])
  EDD_m.append(EDD[i])
  Welding_m.append(Welding[i])


#************************************
# Criar matriz SPT (SMD + THT) [machine]
#************************************

def sort_spt(tempos, ref, edd, welding, tempos2, spt):
```

```
 memória_REF = ref
 memória_Data = edd
 memória_Welding = welding
 machine1 = np.zeros([m-2, n, 5]) #(m - 2) -> nº de linhas; n - > nº de referências; 5 -
 > número de parâmetros
 for i in range(m-2):
   ref = memória_REF
   edd = memória_Data
   welding = memória_Welding
   spt[:,i], tempos[:,i], tempos2[:,i], ref, edd, welding = (list(t) for t in zip(*sorted(zip(spt[:,i], tempos[:,i], ref, e
dd, tempos2[:,i], welding))))
   aux2 = np.c_[ref, tempos[:,i], edd, tempos2[:,i], welding]
   machine1[i] = aux2
 return (machine1)

#************************
# Criar matriz SPT (SMD) [machine]
#************************
def sort_spt_SMD(tempos, ref, edd, welding, tempos2):
 memória_REF = ref
 memória_Data = edd
 memória_Welding = welding
 machine1 = np.zeros([m-2, n, 5])
 for i in range(m-2):
   ref = memória_REF
   edd = memória_Data
   welding = memória_Welding
   tempos[:,i], tempos2[:,i], ref, edd, welding = (list(t) for t in zip(*sorted(zip(tempos[:,i], ref, edd, tempos2[:,i]
, welding))))
   aux2 = np.c_[ref, tempos[:,i], edd, tempos2[:,i], welding]
   machine1[i] = aux2
 return (machine1)
#************************
# Criar matriz EDD [machine]
#************************
def sort_edd(tempos, ref, edd, welding, tempos2):
 memória_REF = ref
 memória_Data = edd
 memória_Welding = welding
 machine1 = np.zeros([m-2, n, 5])
 for i in range(m-2):
   ref = memória_REF
   edd = memória_Data
   welding = memória_Welding
   edd, tempos[:,i], ref, tempos2[:,i], welding = (list(t) for t in zip(*sorted(zip(edd, tempos[:,i], ref, tempos2[:,i]
, welding))))
   aux2 = np.c_[ref, tempos[:,i], edd, tempos2[:,i], welding]
   machine1[i] = aux2
 return (machine1)
```

```
metodo="3"
#Expandir programa para outras heurísticas
#metodo=input("1 - Min Max (SMD); 2 - EDD; 3 - Min Max (SMD + THT)")
if metodo == "1":
 machine=sort_spt_SMD(times, REF, EDD, Welding, times2)
 x=3
 y=2
if metodo == "2":
 machine=sort_edd(times, REF, EDD, Welding, times2)
 x=0
 y=3
if metodo == "3":
 machine=sort_spt(times, REF_m, EDD_m, Welding_m, times2, SPT)
 x=3
 y=2

##FIM************************
```

```
# JSSP [n, m]
model = Model('JSSP')

c = model.add_var(name="C")
x = [[model.add_var(name='x({},{})'.format(j+1, i+1))
      for i in range(m)] for j in range(n)]
y = [[[model.add_var(var_type=BINARY, name='y({},{},{})'.format(j+1, k+1, i+1))
       for i in range(m)] for k in range(n)] for j in range(n)]

model.objective = c

for (j, i) in product(range(n), range(1, m)):
    model += x[j][machines[j][i]] - x[j][machines[j][i-1]] >= \
        times[j][machines[j][i-1]]

for (j, k) in product(range(n), range(n)):
    if k != j:
        for i in range(m):
            model += x[j][i] - x[k][i] + M*y[j][k][i] >= times[k][i]
            model += -x[j][i] + x[k][i] - M*y[j][k][i] >= times[j][i] - M

for j in range(n):
    model += c - x[j][machines[j][m - 1]] >= times[j][machines[j][m - 1]]

model.optimize()

print("Time: ", c.x)
for (j, i) in product(range(n), range(m)):
  if times[j][i] != 0:
    print("Job %d starts on Machine %d at Time %g " % (j+1, i+1, x[j][i].x))
```

Código fonte do programa desenvolvido

```
# SPT preemptivo (CT, WT, TAT)

n = int(input('Enter no of processes: '))
bt = [0] * (n + 1)
at = [0] * (n + 1)
abt = [0] * (n + 1)
for i in range(n):
  abt[i] = int(input('Enter the burst time for process {} : '.format(i + 1)))
  at[i] = int(input('Enter the arrival time for process {} : '.format(i + 1)))
  bt[i] = [abt[i], at[i], i]

bt.pop(-1)
sumbt = 0
i = 0
ll = []
for i in range(0, sum(abt)):
  l = [j for j in bt  if j[1] <= i]
  l.sort(key=lambda x: x[0])
  bt[bt.index(l[0])][0] -= 1
  for k in bt:
    if k[0] == 0:
      t = bt.pop(bt.index(k))
      ll.append([k, i + 1])
ct = [0] * (n + 1)
tat = [0] * (n + 1)
wt = [0] * (n + 1)
for i in ll:
  ct[i[0][2]] = i[1]

for i in range(len(ct)):
  tat[i] = ct[i] - at[i]
  wt[i] = tat[i] - abt[i]
ct.pop(-1)
wt.pop(-1)
tat.pop(-1)
abt.pop(-1)
at.pop(-1)
for i in range(len(ct)):
  print("{}\t{}\t{}\t{}\t{}\n".format(abt[i], at[i], ct[i], tat[i], wt[i]))
print('Average Waiting Time = ', sum(wt)/len(wt))
print('Average Turnaround Time = ', sum(tat)/len(tat))
print('Average Cycle Time = ', sum(ct)/len(ct))
```