Carlos Alexandre Rodrigues

DIMENSIONAMENTO DE BUFFERS EM LINHAS DE PRODUÇÃO

UNIVERSIDADE D COIMBRA
1 2 9 0

1 2  9 0

UNIVERSIDADE D
COIMBRA

Carlos Alexandre Rodrigues

DIMENSIONAMENTO DE BUFFERS EM LINHAS DE PRODUÇÃO

Dissertação no âmbito do Mestrado em Engenharia e Gestão Industrial
orientada pelo Professor Doutor Samuel de Oliveira Moniz e apresentada ao
Departamento de Engenharia Mecânica da Faculdade de Ciências e Tecnologia da
Universidade de Coimbra.

Julho de 2019

FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE D COIMBRA

DEPARTAMENTO DE ENGENHARIA MECÂNICA

Dimensionamento de *buffers* em linhas de produção

Dissertação apresentada para a obtenção do grau de Mestre em Engenharia e Gestão Industrial

Sizing buffers in production lines

Autor
Carlos Alexandre Rodrigues

Orientadores
Professor Doutor Cristóvão Silva
Professor Doutor Samuel Moniz

Júri

Presidente	**Professor Doutor Pedro Mariano Simões Neto** **Professor Auxiliar da Universidade de Coimbra**
Vogais	**Investigadora Catarina Moreira Marques** **Investigadora do INESC TEC Porto** **Professor Doutor Samuel de Oliveira Moniz** **Professor Auxiliar da Universidade de Coimbra**
Orientador	**Professor Doutor Samuel de Oliveira Moniz** **Professor Auxiliar da Universidade de Coimbra**

Coimbra, julho, 2019

"Stay hungry, Stay foolish"

Steve Jobs

À minha família.

Agradecimentos

O desenvolvimento de uma dissertação de mestrado é culminar da minha formação académica, ou seja, é algo de uma extrema importância. Para chegar a este momento foi necessário muito apoio e colaboração por parte de todos os envolventes. Desta feita, quero expressar o reconhecimento a todos os que me acompanharam nesta fase da minha vida.

Em primeiro lugar gostaria de agradecer à minha família, mais concretamente à minha mãe Odete Rodrigues, à minha tia Elsa Silva e aos meus avós António e Maria Rodrigues, por todo o apoio e confiança depositada em mim.

Ao meu orientador, Professor Doutor Samuel Moniz, por me ter despertado em mim um interesse nas áreas de otimização e simulação. Para além disso, gostaria de agradecer todo o apoio e recetividade demonstrada pelo mesmo bem como o interesse nas minhas escolhas profissionais pós-universidade.

Ao Professor Mestre Pedro Coelho, por toda ajuda prestada de forma continuada seja na aplicação do modelo em CPLEX, seja no entendimento de certos conceitos.

A todos os meus amigos mais antigos e aos que Coimbra me deu quero deixar um outro agradecimento sentido, especialmente à Marta Carreira, Pedro Dias e Hildair Cuinala.

A todos, um sentido e sincero obrigado!

Resumo

A variabilidade que pode surgir numa linha de produção, bem como funcionamentos com diferentes taxas de produção são realidades com a qual a indústria tem de lidar. Uma solução eficaz para contornar estes tipos de problemas é a colocação de *buffer*s. Através da alocação de *buffer*s às máquinas é possível ainda um melhor controlo sobre tempos não produtivos, de processamento e de bloqueio, para além de permitir uma melhoria no *output*. Contudo, a alocação de um *buffer* não deve ser feita de forma casual. Deste modo, aplicação de um modelo de alocação de *buffer* torna-se um ponto interessante de estudo para que uma linha de produção consiga ser o maior desempenho possível.

A presente dissertação tem como objetivo a aplicação de um modelo conhecido otimização de programação linear inteira mista (MILP) para o dimensionamento e alocação de *buffer*s e aplicar o mesmo a um problema real, neste caso uma fábrica de produção de mobiliário.

Uma vez que o problema abordado é um sistema muito complexo com muitos processos envolvidos e muitos produtos a serem produzidos, sugere-se uma abordagem de decomposição. Para se conseguir aplicar o modelo foi criado um método de decomposição que envolve a agregação de dados dos produtos, máquinas e rotas e as famílias de produtos mais significativas para a fábrica. Esta abordagem permitiu-nos obter um escalonamento da produção eficaz, representado através de um diagrama de *Gantt*.

Palavras-chave: Escalonamento *job-shop*, Problema de alocação de *buffer*, *Blocking*, *No-wait*, Programação linear inteira mista, Otimização de processos.

Abstract

The variability that may arise in a production line as well as the operation with different production rates are realities that the industry must cope with. An effective solution to overcome these issues is the allocation of buffers. By allocating buffers to machines, it is possible to have a better control over idle, processing and blocking times, while also allowing for dead-end improvements. However, a buffer allocation should not be done casually. In this sense, the application of a buffer allocation model becomes an interesting topic to be explored so that a production line can be as efficient as possible.

The present dissertation aims to apply a model known as a Mixed Integer Linear Programming (MILP) for the sizing and allocation of buffers and apply it to a real problem, such as the case of a furniture production factory.

Since the problem addressed is a very complex system with many processes and products being produced, a decomposition approach is suggested. In order to apply the model, it has been developed a decomposition method which involves the aggregation of product data, machines and routes and the most important families of products for the industry. This approach allowed us to obtain an efficient production scheduling, represented by means of a Gantt diagram.

Keywords Job-shop scheduling, *Buffer* allocation problem, Blocking, No-wait, Mixed integer linear programming, Process optimization.

Índice

ÍNDICE DE FIGURAS

ÍNDICE DE TABELAS

1. INTRODUÇÃO

O sucesso no planeamento da produção está dependente de uma série de dificuldades, que vão desde a previsão de vendas à avaliação de diversos fatores externos, como a previsão do crescimento, a diversificação de produtos a serem produzidos, entre outros critérios. Embora com o avanço da tecnologia ao longo do tempo tenha sido possível aumentar a capacidade de produção, os desafios e as complexidades também aumentaram. Segundo Shimon (2006), para além das dificuldades acima referidas existem outras que estão relacionadas com a maior complexidade de trabalho, limitações causadas por maior interdependência, questões de integridade e confiança, maior necessidade de coordenação, cooperação e sincronização, desafios e falhas de comunicação, requisitos dos sistemas cada vez mais complexos, problemas de incompatibilidade e novos requisitos de formação de colaboradores que naturalmente leva a custos adicionais.

Uma linha de produção é um sistema no qual um conjunto de máquinas está conectado em série ou paralelo e separadas (ou não) por *buffer*s (ver Figura 1.1). Para evitar paragens de produção – por exemplo devido a diferença nos tempos de processamento, taxa de avarias, taxa de reparações e tempo entre avarias – existem posições na linha de produção onde se pode armazenar produtos semiacabados. Estas posições de armazenamento são chamadas de *buffer*. Um aumento no número e tamanho dos *buffer*s leva a uma redução do impacto da incerteza na linha, pois, desta forma haverá um maior controlo sobre a variabilidade e tempos mortos, de bloqueio e processamento.

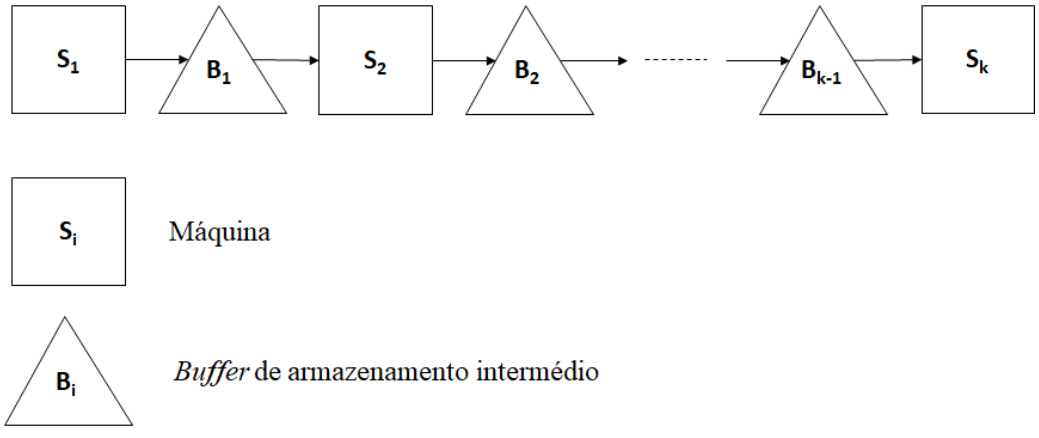

Figura 1.1. Linha de produção com K estações de trabalho e K-1 espaços de *buffer* (Zandieh, M. et al., 2017)

Para aumentar a eficiência de uma linha de produção e de todas as suas envolventes existem vários tipos de problemas de otimização. Os problemas de *Job-shop scheduling* (JSS) são problemas de otimização bem conhecidos, que relacionam um dado número de *jobs* (trabalhos) a um conjunto de máquinas. Cada trabalho consiste num conjunto de operações que necessitam de ser processadas numa determinada ordem, num conjunto de máquinas conhecidas e com um dado tempo de processamento também conhecido. Nenhuma máquina pode processar mais de uma operação ao mesmo tempo. O objetivo geralmente considerado nos problemas de JSS é a minimização do *makespan* (Louaqad e Kamach, 2016).

Contudo, este tipo de problemas apresenta elevada dificuldade do ponto de vista da otimização. Uma indicação da sua dificuldade remete para uma instância de dez *jobs* e dez máquinas, formulada pela primeira vez por Fisher e Thompson (1963) foi exatamente resolvida por Carlier e Pinson (1989), usando um algoritmo *branch-and-bound* em mais de cinco horas tempo de corrida.

O modelo clássico do JSS supõe que o espaço de armazenamento tem uma capacidade infinita (*infinite-buffer*) e as operações de transporte de uma máquina para outra são instantâneas. Contudo, em muitos casos práticos, o número de *buffer*s é limitado. Por outro lado, os *buffer*s podem ser caros ou inadequados por razões tecnológicas ou de processo.

As abordagens propostas na literatura fornecem informações valiosas sobre como dimensionar os *buffer*s. Os trabalhos focados em casos industriais destacam dificuldades relacionadas com a complexidade das restrições, o nível de detalhe do modelo desenvolvido, estendendo-se a uma configuração diferente das linhas de produção e diferentes condições de funcionamento (Demir et al., 2014).

O principal objetivo desta dissertação é desenvolver uma abordagem de otimização para encontrar os tamanhos ótimos dos *buffers* em problemas de larga escala. Desta forma, será estudada uma abordagem de otimização para resolver o problema de dimensionamento de *buffer*s e, após validação do algoritmo, executar e analisar como o mesmo se comporta num problema real.

Esta dissertação está organizada em cinco capítulos. No primeiro, é feito um enquadramento ao tema, abrangendo as motivações e objetivos propostos. O capítulo dois inclui o enquadramento teórico, no qual são definidos conceitos de *job-shop scheduling e*

flow-shop scheduling. O conceito de *buffer* e de alocação de *buffer* são igualmente apresentados. No capítulo três é descrita a metodologia utilizada para resolver o problema de alocação de *buffer*s. Deste modo é apresentada a descrição do problema, a formulação matemática do mesmo e os resultados obtidos que servem para efeitos de validação. O capítulo quatro diz respeito à implementação do algoritmo descrito no capítulo anterior num caso de estudo real de uma fábrica, testando a sua aplicabilidade e realizando a devida análise de resultados para comprovar os resultados obtidos. No capítulo cinco, e último, são apresentadas as considerações finais sobre o problema, bem como sugestões de trabalho futuro.

2. ENQUADRAMENTO TEÓRICO

Neste capítulo começa-se por abordar o problema de *job-shop scheduling* (JSS) ao identificar o seu interesse e propósito. No mesmo sentido será igualmente exposto o tema relativo à alocação de *buffer*s.

2.1. *Job-shop scheduling*

O problema do JSS considera um certo número de *jobs* que necessitam de ser processados num conjunto de máquinas. A cada trabalho está associada uma determinada rota, ou seja, cada trabalho tem uma ordem de processamento que pode ser diferente de trabalho para trabalho (Liu et al., 2018). Este fator é o que distingue o JSS do *flow-shop scheduling* (FSS), onde todos os trabalhos são realizados na mesma ordem e, assim, por consequência resulta numa configuração das máquinas que promove a redução dos tempos não produtivos. A Figura 2.1 é uma representação gráfica da diferença de um *job-shop* e um *flow-shop*. Tendo em conta estas definições, podemos afirmar que para as situações de produção em massa a configuração mais indicada será o *flow-shop* e para os casos de customização a configuração mais indicada é o *job-shop*.

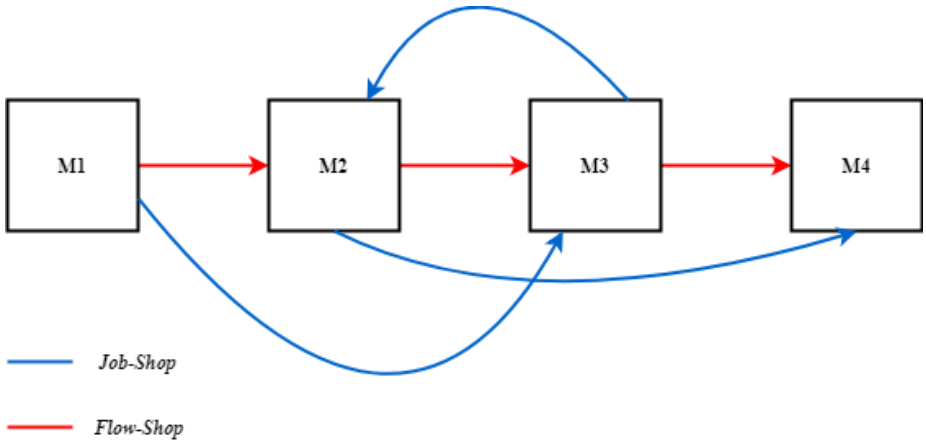

Figura 2.1. Situações de *job-shop* e *flow-shop*

Relativamente ao FSS desenvolveram-se vários trabalhos deste problema com restrições de *buffer*. As ideias de *no-wait*, *limited-buffer* e bloqueio em *buffer*s foram propostas por Leinsten (1990). O algoritmo exibe uma visão geral e sistemática de como

formular problemas de *flow-shop* com armazenamento de *buffer* limitado, bem como várias heurísticas para resolver esta classe de problemas. Fink e Voß (2003) desenvolveram vários algoritmos meta-heurísticos para resolver o FSS com restrições de *no-wait,* onde é considerada a aplicação de diferentes tipos de meta-heurísticas de um ponto de vista prático, para examinar o *trade-off* entre o tempo de execução e a qualidade da solução, bem como o conhecimento e os esforços necessários para implementar e calibrar os algoritmos. Grabowski e Pempra (2007) desenvolveram um algoritmo de pesquisa com o intuito de minimizar o *makespan* em problemas de *flow-shop* com restrições de bloqueio. Além disso, são utilizados os *multimoves,* que consistem na realização de vários movimentos simultaneamente numa única iteração e orientar o processo de pesquisa para áreas mais promissoras do espaço de soluções, onde boas soluções podem ser encontradas, permitindo acelerar a convergência do algoritmo. Um algoritmo híbrido baseado na evolução diferencial com restrições de *limited-buffer* em máquinas consecutivas para FSS foi feito por Qian et al. (2009). O algoritmo considera um mecanismo de evolução diferencial para pesquisar o espaço de soluções e também adota uma rotina de pesquisa local do problema para realizar uma exploração minuciosa das regiões. Ding et al. (2015) analisaram as propriedades de bloqueio do FSS e propuseram novas restrições (os autores consideram o tempo de início e de fim de uma máquina com tempo de bloqueio iguais) para o problema de *flow-shop* com bloqueio para minimizar o *makespan*. Um procedimento de remoção baseado nessas propriedades propostas é utilizado na fase de construção de um algoritmo para diminuir o número total de soluções a serem examinadas para encontrar um escalonamento ótimo. Considerando abordagens heurísticas, Aldowaisan e Ali (2015) desenvolveram um *simulated annealing* (SA) e um algoritmo genético (GA). Zhang et al. (2017), através da análise de um problema de *job-shop scheduling*, combinaram uma heurística e um algoritmo híbrido para resolver o FSS com um processador de lote e *buffer*s limitados e, desta forma, obtiveram um algoritmo eficiente com soluções de qualidade para problemas em larga escala.

O problema de JSS com restrições de *no-wait* e *no-buffer* (*blocking*) não teve o mesmo tipo de atenção por parte dos investigadores. Uma primeira introdução ao JSS com uma restrição de *no-wait* (NWJSS) e com *blocking* (BJSS) foi proposta por Hall e Sriskandarajah (1996), onde os autores analisaram a complexidade computacional de uma ampla variedade de problemas de escalonamento *no-wait* e *blocking*. Mascis e Pacciarelli

(2002) estudaram os problemas de BJSS e NWJSS por meio de um gráfico alternativo, que é uma extensão do gráfico disjuntivo clássico, mostrando que várias propriedades-chave, usadas para desenhar procedimentos heurísticos, não se sustentam nos casos de *blocking* e *no-wait*, enquanto algumas das ideias mais eficientes utilizadas para desenvolver algoritmos de *branch and bound* podem ser facilmente estendidas. Pacciarelli (2002) aplicou ainda o gráfico alternativo para modelar um problema complexo de escalonamento de uma fábrica que incorpora a restrição de *no-wait* e tempos de mudança independentes de sequência. Hauptman e Jovan (2004) estudaram um caso real de escalonamento da produção transformando-o num *job-shop* com restrições de *no-wait* e *no-buffer* e introduziram uma abordagem simples para evitar problemas de escalonamento. Um problema de JSS de agendamento de pacientes, com restrição de *no-wait* foi modelado e resolvido através de um algoritmo evolucionário por Chie et, al. (2008). Samarghandi e ElMekkawy (2013) desenvolveram um modelo matemático para resolver um problema com tempos de *set-up* dependentes das sequências e restrições de servidor único, através de JSS com restrição de *no-wait*. Além disso, um algoritmo genético é proposto para encontrar as soluções ótimas (ou quase ótimas). Pranzo e Pacciarelli (2016) desenvolveram um algoritmo iterativo para resolver variações do JSS com restrição de *no-buffer*. O algoritmo desenvolvido é uma meta-heurística baseada na repetição de uma fase de destruição, que remove parte da solução, e uma fase de construção, na qual uma nova solução é obtida. Louaqad e Kamach (2016) investigaram as situações de *no-wait* e *no-buffer* de JSS em células robóticas e desenvolveram um modelo MILP (*Mixed Integer Linear Programming*) para resolver este problema com até três robots, dez máquinas e dez *jobs*. Liu et al. (2018) investigaram, pela primeira vez, um problema com quatro restrições de *buffer,* CBJSS (*Combination of four Buffering constraints for JSS*) e desenvolveram um algoritmo eficiente. O problema CBJSS é importante para a modelação e análise de sistemas de escalonamento em sectores industriais nas áreas da química, alimentar, manufatura, ferrovia, saúde e aviação.

2.2. Alocação de *buffer*

A variabilidade que pode surgir nas linhas de produção, bem como diferentes taxas de produção, são realidades com que a indústria tem de lidar. Uma solução eficaz para lidar com estes tipos de problemas é a colocação de *buffer*s. Quanto à alocação de *buffer*s, a revisão da literatura pode ser categorizada em diferentes classes (Demir et al., 2014). Na

primeira categoria, o problema de alocação de *buffer* é classificado com base no facto da linha de produção ser determinística (Smith et al., 2010) ou não determinística (Sabuncuoglu et al., 2006). Revendo alguns dos estudos feitos, é percetível que alguns trabalhos estudaram o problema de alocação de *buffer* (BAP) em linhas de produção determinística enquanto outros estudaram BAP em linhas de produção não determinística (Demir et al., 2014). Além disso, a literatura pode ser classificada em termos da função objetivo. A maioria dos estudos da literatura lidou com o problema de alocação de *buffer* considerando objetivos de minimização de custos, maximização do lucro, maximização da taxa de produção ou minimização do tamanho total do *buffer*. Por outro lado, alguns trabalhos estudaram o problema de alocação de *buffer* multiobjectivo, no qual duas ou mais funções objetivo mencionadas anteriormente são consideradas simultaneamente. De ressaltar que, considerando múltiplas funções objetivo contraditórias, soluções mais realistas e aplicáveis podem ser obtidas.

Demir et al (2014) apresentam uma revisão da literatura abrangente. Este caracteriza as várias formas do BAP e os trabalhos desenvolvidos no período temporal de 1998 a 2012. Cerca de 100 estudos são incluídos na revisão (58 deles são sobre linhas de produção), e isso indica que o BAP tem sido um tema importante na última década.

Um algoritmo eficiente para o dimensionamento de *Buffer*s, que permite maximizar o lucro da linha de produção, sujeito a uma restrição da taxa de produção, através de técnicas de programação não-linear foi desenvolvido por Shi e Gershwin (2009). Neste modelo, tanto o custo do tamanho do *buffer* como o custo médio de stock foram considerados. Os resultados numéricos são obtidos tanto para linhas de produção curtas como para longas. A eficiência do algoritmo proposto é avaliada para uma linha de produção que possua até trinta máquinas. Massim, Y. et al (2010) também propuseram um algoritmo para a alocação ótima de um *buffer* em linhas de produção através da maximização das taxas de transferência e lucro. O algoritmo provou ser bastante eficiente tanto na velocidade de convergência, como na qualidade das soluções. Um procedimento multiobjectivo para BAP foi aplicado por Amiri e Mohtashami (2012). Este trabalho explorou o desenvolvimento do BAP especificamente. As funções de distribuição (normal, gama, Wiebull e uniforme) foram relaxadas para todos os parâmetros da linha de produção, ou seja, o tempo do processo, o tempo entre duas falhas e o tempo de reparo. A maximização da taxa de produção e minimização do tamanho total do *buffer* foram as funções objetivas do modelo, e um

algoritmo genérico híbrido foi utilizado para encontrar soluções não dominantes próximas. Nahas (2017) considerou o problema para determinar a melhor política de manutenção e alocação de *buffer* ideal que minimiza os custos totais do sistema sujeitos a um nível de *throughput*. Uma aproximação do tipo decomposição analítica é usada para estimar a taxa de transferência da linha de produção. O problema de desenho ótimo da política de manutenção e alocação de é formulado como um modelo de otimização, em que as variáveis de decisão são níveis de *buffer*. Renna, P. (2018) propôs, através de simulação, uma política para a alocação de *buffer*s com manutenção preventiva nas linhas de produção. Os resultados numéricos obtidos ilustram como a política proposta permite obter melhores resultados em termos de redução de custos totais e melhoria da taxa de produção. A abordagem proposta pode ser controlada por três parâmetros que influenciam principalmente os componentes do custo: posse, preventivo e corretivo. Portanto os decisores podem utilizar o ambiente de simulação desenvolvido e definir apenas três parâmetros para decidir quais custos reduzir, mantendo o mesmo nível de uma taxa de produção. Os decisores podem usar o modelo proposto para apoiar a tomada de decisão sobre os custos (devido ao nível do *buffer*) e a taxa de transferência a alcançar.

A revisão da literatura mostra que a maioria dos artigos tende a resolver versões muito agregadas do BAP, o que pode dificultar a aplicação prática das soluções propostas. Além disso, a probabilidade de falha de máquinas ou a linha de produção não determinística foi desconsiderada na maioria dos trabalhos. Para mais, quase todos os artigos lidaram com um único objetivo.

3. METODOLOGIA

Neste capítulo será apresentada a formulação do modelo de dimensionamento de *buffers*. Tendo como ponto de partida o modelo, CBJSS, desenvolvido por Liu et al (2018), iremos então apresentar a nossa formulação do problema e indicar quais as nossas contribuições para o modelo em questão bem como explicar todas as restrições do mesmo.

3.1. Formulação do problema

Num problema clássico de *job-shop* existem *n* trabalhos que são processados em *m* máquinas, tendo como finalidade a redução do tempo de conclusão de todos os trabalhos (*jobs)*. Em cada *job* existe um número de operações igual ao número de máquinas (*m*), em que a rota pode ser diferente de *job* para *job*. Uma máquina só pode processar uma operação de cada vez. Cada máquina tem associada uma restrição de *buffer*, seja por questões de serviço, processo, ou de segurança. Os tempos de transporte entre operações não são considerados. Para definir as restrições de *buffer* quatro soluções são consideradas, $b_i \in \{\varepsilon, \phi, \theta|\tau_i, \delta\}$ para cada máquina, ver Figura 3.1, Figura 3.2 e Figura 3.3. Caso $b_i = \varepsilon$, significa que estamos numa condição de *no-wait*, ou seja, cada *job* que seja concluído em M_i deve ser imediatamente transferido para a próxima máquina sem que exista qualquer atraso, por outras palavras estamos numa situação de processamento contínuo. Se $b_i = \phi$, encontramo-nos numa situação de *no-buffer*, o que implica que não existe armazenamento numa dada M_i após a conclusão de uma operação. Por sua vez, se $b_i = \theta|\tau_i$, estamos numa situação de *limited-buffer* o que significa que temos um número limitado de *buffer*s com $0 \leq \tau_i \leq m$. Relativamente à situação de *infinite-buffer*, $b_i = \delta$, é um conceito complicado de entender na medida em que estamos a modelar o dimensionamento de *buffers* e não podemos dizer que os mesmos têm capacidade ilimitada. Desta feita, optámos por não introduzir este parâmetro, pois é uma forma de o modelar, e considerarmos apenas as outras três situações.

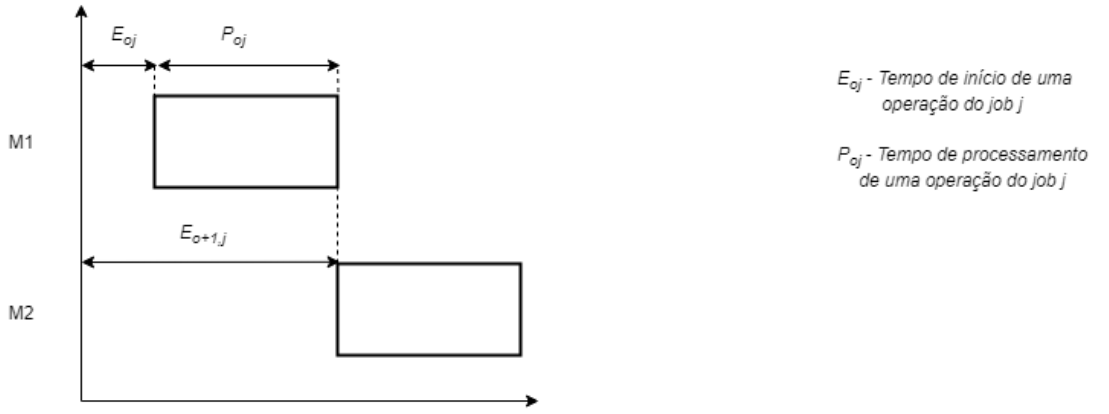

Figura 3.1. Situação de *no-wait*

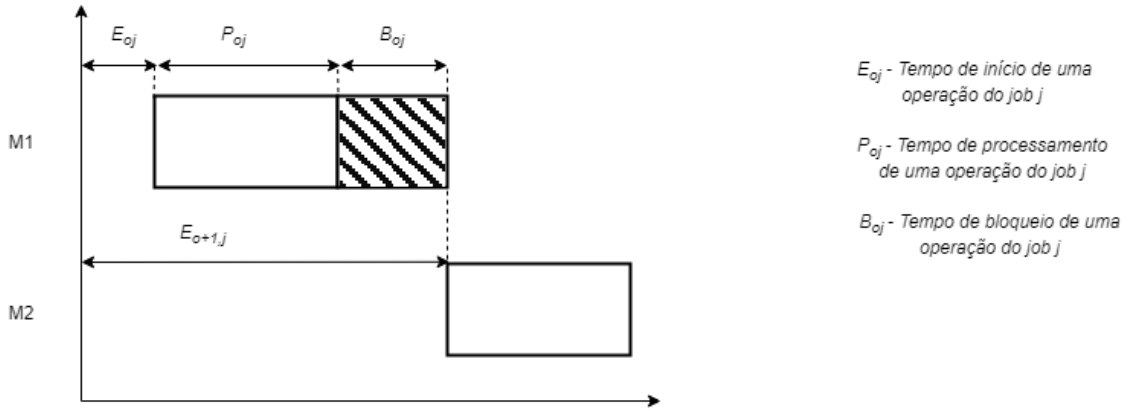

Figura 3.2. Situação de *no-buffer*

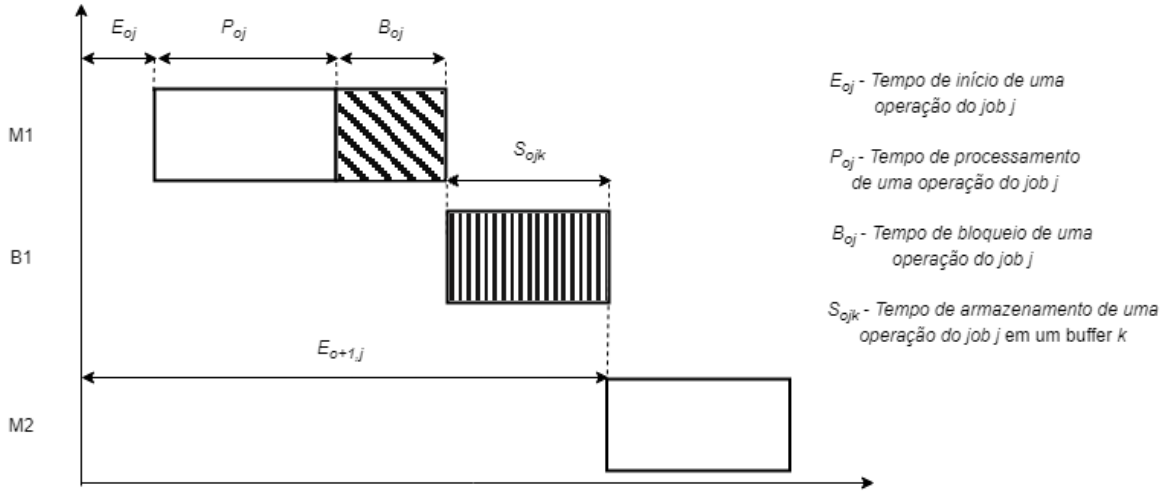

Figura 3.3. Situação de *limited-buffer*

3.1.1. Índices e parâmetros

Para além dos índices e parâmetros de *job-shop* apresentados em 3.1, existem outros que devemos considerar. Um acontecimento, A_{oj}, que diz respeito a uma dada associação entre uma operação, o, e um *job*, j. Numa lógica semelhante temos o tempo de processamento, P_{oj}. Como estamos numa situação com *buffers*, k, os mesmos têm um número dependendo da máquina a que estão associados, τ_i.

n	Número de *jobs*
m	Número de máquinas
j	*Job* índex, $j = 1,2, \dots, n$
J	Conjunto de *jobs*
π_j	Número de operações do *job j*
i	Índex da máquina, $i = 1,2, \dots, m$
M	Conjunto de máquinas
o	Índex da operação, $o = 1,2, \dots, m$
O^{NW}	Conjunto de operações com restrição de *buffer no-wait* ($b_i = \varepsilon$)
O^{NB}	Conjunto de operações com restrição de *buffer no-buffer* ($b_i = \phi$)
O^{LB}	Conjunto de operações com restrição de *buffer limited-buffer* ($b_i = \theta$)
O	Conjunto total de operações, $O = O^{NW} + O^{NB} + O^{LB}$
O^{π_j}	Conjunto de últimas operações de todos os *jobs*
A_{oj}	Acontecimento *o* do *job j*
P_{oj}	Tempo de processamento de A_{oj}
τ_i	Número do *buffer* associado a cada máquina quando $b_i = \theta$
k	Índex do *buffer*, $k = 1,2, \dots, \tau_i$
K	Conjunto de *buffer*s
H	Número grande constante positivo

3.1.2. Variáveis de decisão

O objetivo passa por minimizar o *makespan,* C_{max}. Para tal, devemos garantir que existe uma relação de precedência nas máquinas, $y_{ojo'j'i}$, e nos *buffers*, $w_{ojo'j'k}$.

Devemos ainda garantir que o tempo de início de um acontecimento, A_{oj}, não pode ser superior ao tempo de início do acontecimento seguinte do mesmo *job*, $A_{o+1,j}$. Outras variáveis como o tempo de saída de uma máquina, D_{oj}, ou tempo de saída de um *buffer*, L_{ojk}, variam de acordo com a restrição de *buffer* associada.

$y_{ojo'j'i}$	1, caso A_{oj} preceda $A_{o'j'}$ numa máquina e 0 em caso contrário
$w_{ojo'j'k}$	1, caso A_{oj} preceda $A_{o'j'}$ num *buffer* e 0 em caso contrário
E_{oj}	Tempo de início de A_{oj}
C_{oj}	Tempo de conclusão de A_{oj}, ou seja, $C_{oj} = E_{oj} + P_{oj}$
B_{oj}	Tempo de bloqueio de A_{oj}
D_{oj}	Tempo de saída de A_{oj}, ou seja, $D_{oj} = C_{oj} + B_{oj}$
S_{ojk}	Tempo de armazenamento de A_{oj} num *buffer*
L_{ojk}	Tempo de saída de A_{oj} num *buffer*, ou seja, $L_{ojk} = S_{ojk} + D_{oj}$
C_{max}	*Makespan*, ou seja, tempo máximo de conclusão de todas as tarefas

3.2. Formulação matemática

A função objetivo consiste em minimizar o *makespan*, ou seja, o tempo máximo de conclusão de todos os *jobs*.

$$C_{max} \tag{1}$$

Sujeito a:

$$C_{oj} = E_{oj} + P_{oj} \text{ , com } o \in O, j \in J \tag{2}$$

$$D_{oj} = C_{oj} + B_{oj}, \text{ com } o \in O, j \in J \tag{3}$$

$$L_{ojk} = S_{ojk} + D_{oj}, \text{ com } o \in O, j \in J, k \in K \tag{4}$$

As restrições (2) a (4) são utilizadas para diminuir o tamanho das restrições e assim facilitar a leitura das mesmas.

$$Hy_{ojo'j'i} + E_{oj} \geq D_{o'j'}, \text{com } o, o' \in O < \pi_j - 1, j, j' \in J | j \neq j' \quad (5)$$

$$H(1 - y_{ojo'j'i}) + E_{o'j'} \geq D_{oj}, \text{com } o, o' \in O < \pi_j - 1, j, j' \in J | j \neq j' \quad (6)$$

As restrições (5) e (6) servem para satisfazer as relações de precedência que existem num par de operações (i.e., A_{oj} e $A_{o'j'}$) associadas, cada uma delas, a um *job*, respetivamente (i.e. J_j e $J_{j'}$), e a uma máquina i pertencente ao conjunto de máquinas M.

$$C_{oj} \leq E_{o+1,j}, \text{com } o \in O < \pi_j - 1, j \in J \quad (7)$$

A restrição (7) obriga a que o tempo de conclusão de uma operação A_{oj} seja menor ou igual que o tempo de início da operação seguinte do mesmo job $A_{o+1,j}$.

$$C_o \leq C_{max}, \text{com } o \in O^{\pi_j}, j \in J \quad (8)$$

$$B_o = 0, \text{com } o \in O^{\pi_j}, j \in J \quad (9)$$

A restrição (8) serve para introduzir o C_{max} no modelo e garante que, o tempo de conclusão da última operação de cada *job*, deve ser menor ou igual ao *makespan*. Por sua vez, a restrição (9) indica que não deve existir tempo de bloqueio na última operação de cada *job*, uma vez que terminada a última operação a mesma é tratada como se abandonasse o modelo.

$$C_{oj} = E_{o+1,j}, \text{com } o \in O^{NW} < \pi_j - 1, j \in J \quad (10)$$

$$B_{oj} = 0, \text{com } o \in O^{NW} < \pi_j - 1, j \in J \quad (11)$$

As restrições (10) e (11) estabelecem a restrição de *no-wait* excluindo a última operação de cada *job*. Nesta situação, restrição (10), o tempo de conclusão de uma operação A_{oj} deve ser igual ao tempo de início da operação seguinte do mesmo *job* $A_{o+1,j}$. Por sua vez, a restrição (11) estabelece o tempo de bloqueio da operação A_{oj} como nulo, pois estamos numa situação de *no-wait*.

$$D_{oj} = E_{o+1,j}, \text{com } o \in O^{NB} < \pi_j - 1, j \in J \quad (12)$$

A restrição (12) define a situação de *no-buffer* excluindo a última operação de cada *job*. Neste caso, após a conclusão de uma operação A_{oj}, uma dada máquina pode ficar bloqueada devido à inexistência de *buffer*s associados e, como tal, o tempo de bloqueio não pode ser nulo.

$$L_{ojk} = E_{o+1,j}, \text{com } o \in O < \pi_j - 1, j, j' \in J, k \in K \quad (13)$$

$$Hw_{ojo'j'k} + D_{oj} \geq L_{o'j'k}, \text{com } o, o' \in O^{LB} < \pi_j - 1, j, j' \in J | j \neq j', k \in K \quad (14)$$

$$H(1 - w_{ojo'j'k}) + D_{o'j'} \geq L_{ojk}, \text{com } o, o' \in O^{LB} < \pi_j - 1, j, j' \in J | j \neq j', k \in K \quad (15)$$

As restrições (13) a (15) definem a restrição de *limited-buffer* excluindo a última operação de cada *job*. Na restrição (13), semelhante à restrição (12) é adicionado o tempo de armazenamento no *buffer* k, que não deve ser nulo. Para além disso, só um *buffer* de uma dada máquina pode armazenar uma só operação A_{oj} de cada vez. As restrições (14) e (15) implicam que os *buffer*s intermédios de capacidade limitada (a capacidade de um *buffer* é proporcional à capacidade de processamento da máquina à qual está agregado) podem ser tratados como máquinas *dummy* que poderão não ser utilizadas. Caso o tempo de armazenamento de uma operação A_{oj} for diferente de zero, significa que o *buffer* está a ser utilizado pela operação A_{oj}.

$$E_{oj}, B_{oj}, D_{oj}, S_{ojk} \geq 0 \quad (16)$$

$$y_{ojo'j'}, w_{ojo'j'k} \in \{0,1\} \quad (17)$$

As restrições (16) e (17) declaram as situações de positividade e binária, respetivamente.

3.3. Resultados obtidos

Os dados da literatura clássica oferecem somente valores para o JSS clássico, sendo esses valores denominados de instâncias de S. Lawrence (1984). Tratando-se de um

problema de otimização, é dada somente atenção às soluções ótimas do problema, neste caso as instâncias de cinco máquinas e dez *jobs* (Liu et al., 2018). Para o conjunto de dados em questão o a alocação de *buffer*s, baseada na alocação realizada por Liu et al. (2018). é definida por:

- Na máquina um temos restrição de *no-wait*.
- Na máquina dois temos restrição de *no-buffer*.
- Restantes três máquinas com um *buffer* cada.

Os resultados foram validados com os da literatura, através do limite mínimo de *makespan* na literatura para as instâncias de S. Lawrence (1984), e com um diagrama de *Gantt*. O modelo foi resolvido com o IBM ILOG-CPLEX 12.8, num computador com Intel Core i7 a 2.4 GHz e 8GB RAM. Os resultados contemplam para cada instância o valor de *makespan* para um *job.shop* clássico, o valor de *makespan* paro CBJSS e o respetivo tempo de computação. As soluções obtidas através da implementação do modelo de CBJSS são apresentados na Tabela 3.1.

Tabela 3.1. Resultados obtidos

Instâncias	n*m	*Makespan* JSS	*Makespan* com *buffers*	Tempo (s) com *buffers*
CBJSS - LA01	10*5	666	727	46.9
CBJSS - LA02	10*5	655	723	25.2
CBJSS - LA03	10*5	597	670	40.5
CBJSS - LA05	10*5	590	671	58.7
CBJSS - LA05	10*5	593	620	120.12

Um problema de *job-shop scheduling* clássico é um problema de escalonamento simplificado, define somente a precedência das operações não tendo em conta tempos de bloqueio, por exemplo. Assim, como o modelo considera quatro restrições de *buffer* o resultado de *makespan* para o CBJSS é maior que o de um *job-shop scheduling* clássico.

Focando na instância CBJSS – LA01, para efeitos de validação, é apresentado o diagrama de *Gantt* (Figura 3.4) onde podemos observar a alocação de cada *job* às respetivas máquinas e de onde retiramos o valor de 727 para o *makespan*.

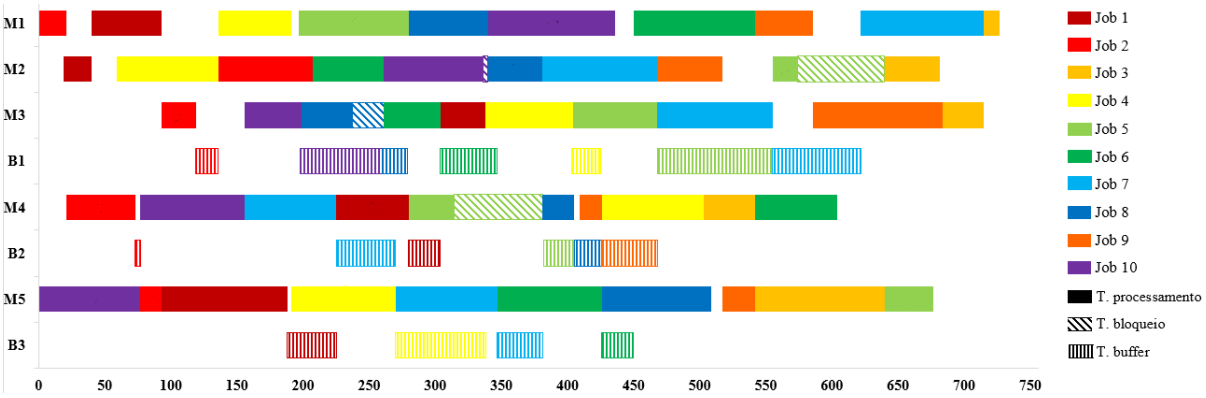

Figura 3.4. Diagrama de *Gantt* com solução ótima para CBJSS - LA01

Continuando com o foco na mesma instância foi testado o impacto que cada uma das restrições de *buffer* tem no problema, sendo os resultados apresentados na Tabela 3.2.

Tabela 3.2. Impacto das restrições de *buffer* no modelo

LA01						
Tipo de *buffer*	*Makespan*	Tempo (s)	Variáveis binárias	Variáveis contínuas	Restrições	GAP
CBJSS	727	46.9	618	249	1467	0%
NWJSS	971	5.6	450	249	1114	0%
NBJSS	793	18.1	450	201	1090	0%
LBJSS	672	64.3	732	281	1704	0%

Em todas as variações foi sempre encontrada a solução ótima e, como tal, o valor do erro de integralidade (*gap*) é sempre nulo. Os resultados do *makespan* foram os esperados, ou seja, sabendo que temos um *buffer* disponível em cada máquina (LBJSS) o tempo de conclusão deve ser menor que qualquer outra combinação dado que a máquina fica livre libertando-a para outros *jobs*. Contudo, um *buffer* acarreta a si sempre custos de posse e, portanto, a colocação de *buffer* em todas as máquinas pode não ser a solução mais viável. A situação de *no-wait* (NWJSS) era expetável ser o maior valor de *makespan*, pois ao limitarmos uma máquina ao tempo de conclusão de um *job* estamos a aumentar os tempos mortos existentes no conjunto das operações. Em suma, a situação mais interessante é a de quatro situação de *buffer* (CBJSS), uma vez que a diferença de *makespan* entre ela e a situação de *buffer* em todas as máquinas é pouca e os custos de posse serão certamente

inferiores. Para tornar mais visível o que foi dito é apresentada a **Error! Reference source not found.** e Figura 3.6 que ilustram o que foi dito.

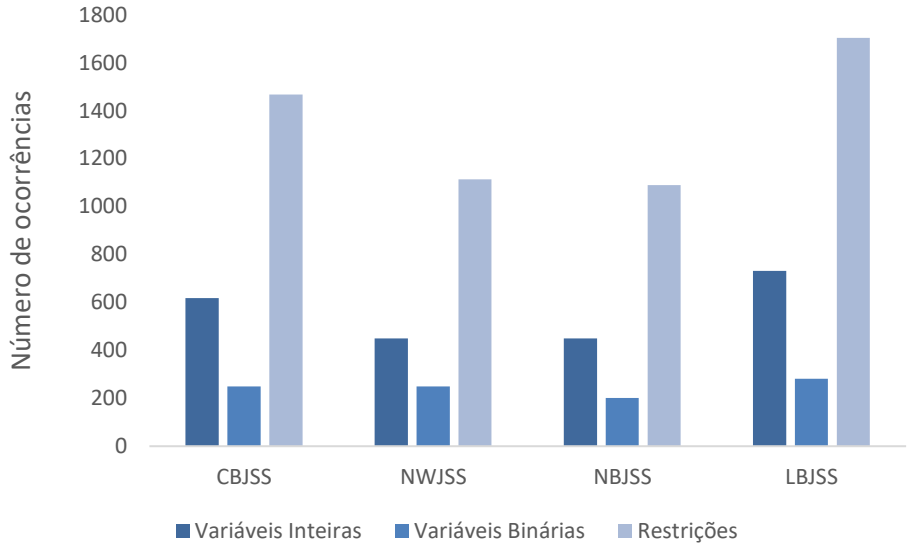

Figura 3.5. Variação das variáveis e restrições com o tipo de *buffer*

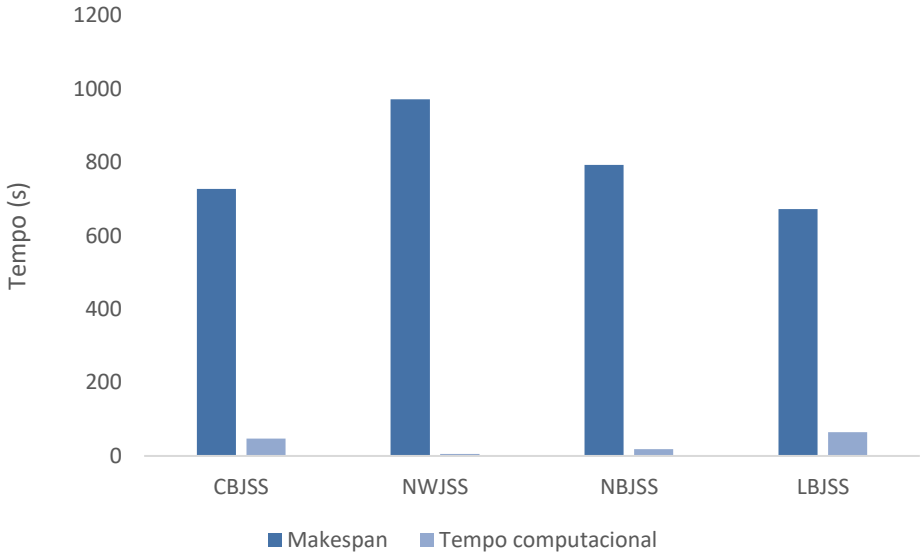

Figura 3.6. Variação do *makespan* e tempo computacional com o tipo de *buffer*

4. CASO DE ESTUDO

Neste capítulo será apresentada a aplicação do modelo a um caso de estudo real. Será ainda realizada a análise e discussão de resultados.

4.1. Análise e tratamento de dados

Tratando-se de uma empresa que produz em grande escala, a quantidade de dados recolhidos é significativamente maior em comparação com a quantidade de dados considerada nas instâncias de teste. Assim, primeiro passo foi analisar e tratar os dados para os poder aplicar no modelo.

Tabela 4.1. Descrição dos dados

Produtos produzidos	310
Famílias de produtos	10
Máquinas utilizadas	26
Número de rotas	27
Total procura anual [Unidades]	6760361

A empresa tem a capacidade de produzir 310 produtos. Relativamente às máquinas existem 24 máquinas, sendo que duas delas existem em duplicado. Existem 27 rotas que, de forma geral, são exclusivas das famílias, à exceção de duas rotas que conseguem satisfazer as rotas de duas famílias de produtos. Cada produto tem a si associado uma família de um total de dez famílias, estando elas apresentadas na Tabela 4.2.

Tabela 4.2. Descrição das famílias de produtos e respetiva quantidade dos mesmos

Famílias de Produtos	Nº de Produtos
FA	21
FB	21
FC	3
FD	63
FE	52
FF	40
FG	33
FH	104
FI	8
FJ	9

Tendo um total de 310 produtos produzidos e um total de 26 máquinas é percetível que a complexidade dos processos é enorme. Para ilustrar essa complexidade do problema foi desenhado um diagrama de *Sankey* (ver Figura 4.1), usando a ferramenta *Power BI*. Através de uma breve análise da Figura 4.1, temos a perfeita noção da complexidade dos fluxos de rotas que existem na empresa. Um outro objetivo da criação deste diagrama, para além do já referido, foi tentar identificar uma macro-rota que pudesse ser utilizada diretamente modelo.

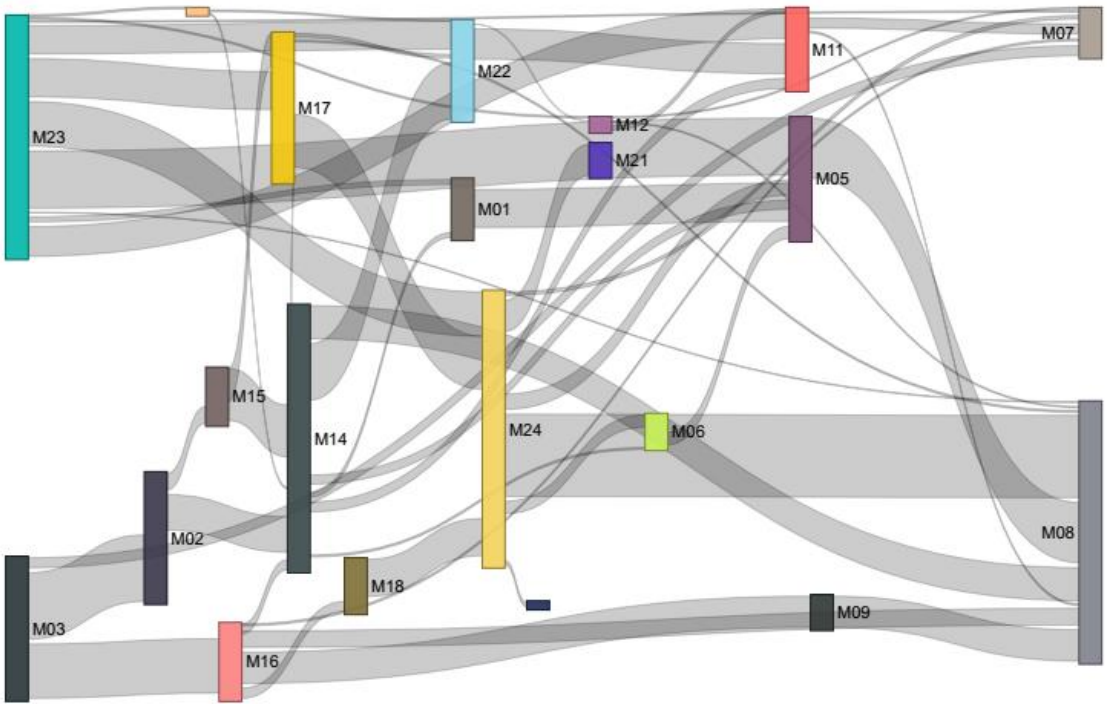

Figura 4.1. Diagrama de *Sankey* para todas as máquinas

4.1.1. Identificação do gargalo do sistema

A identificação do estrangulamento do sistema é fundamental, pois permite-nos saber onde está a limitação do nosso processo e atuar sobre o mesmo para melhorar a eficiência da linha de produção, ver Figura 4.2.

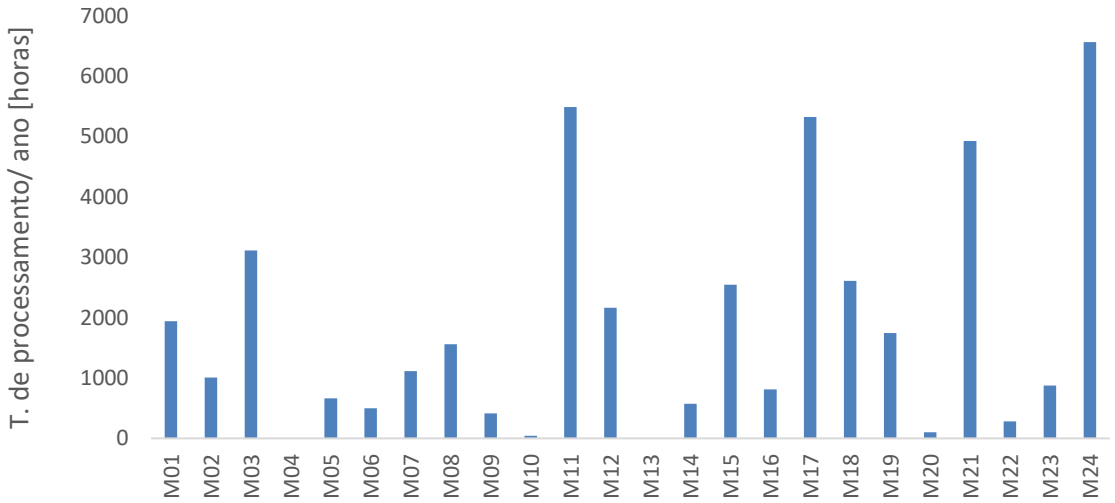

Figura 4.2. Estrangulamento na linha de produção

Por análise da Figura 4.2 torna-se evidente que a máquina que mais está a condicionar o processo é a máquina 24 (M24) e, desta forma, o modelo de otimização terá de ser aplicado a uma rota onde a esta esteja presente. Otimizando o tempo de processamento nesta máquina teremos melhorias em todo o processo.

4.1.2. Análise ABC

O chamado princípio de *Pareto* indica que, para muitos fenômenos, o seguinte é válido: 80% dos efeitos são alcançados por 20% das causas. O princípio remonta a *Vilfredo Pareto* (economista italiano), que percebeu que 80% da propriedade pertencia a 20% da população italiana. Passando este princípio para a nossa área de estudo, podemos afirmar que 20% dos produtos correspondem a 80% do volume de produção.

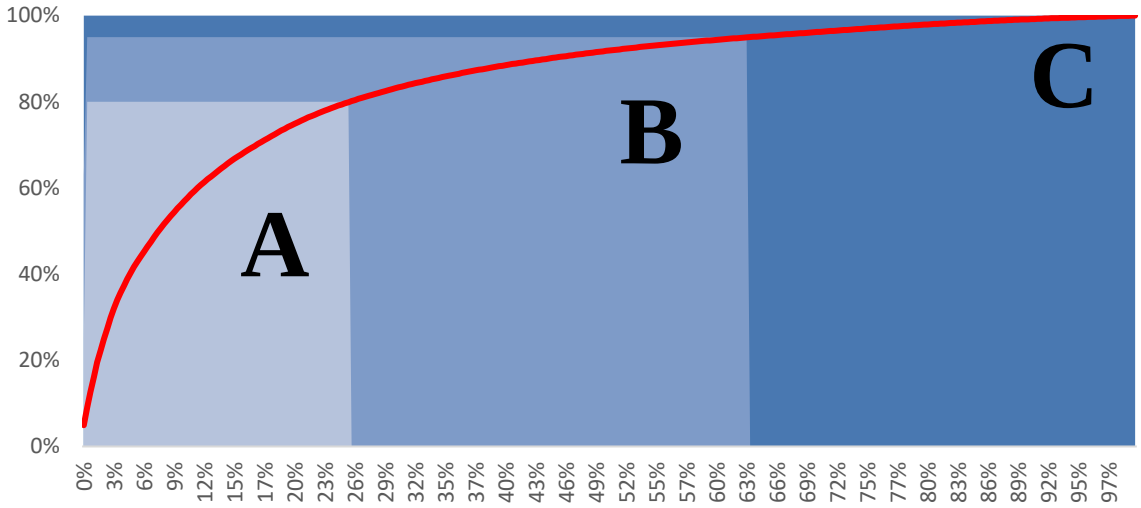

Figura 4.3. Curva ABC para os produtos

A Figura 4.3 relaciona os produtos com a venda dos mesmos. Como conseguimos observar cerca de 25% dos produtos são do tipo A, correspondendo a 80% do volume de produção, o que nos permite concluir um nicho de produtos tem um peso considerável naquilo que é o volume de vendas da empresa. Contudo, devido à complexidade do caso de estudo tornou-se necessário realizar novamente a análise, mas agora para as máquinas e para as rotas. A finalidade era encontrar uma combinação de vários "A" das respetivas análises, para termos um conjunto de dados que melhor representem a complexidade do problema.

4.1.3. Combinações

Uma análise ABC é, de facto, uma ferramenta muito interessante pela sua simplicidade de nos dar a entender a importância de um produto para uma empresa. Assim, foi feita uma análise ABC para as rotas e máquinas, para além da já referida análise aos produtos. Tendo esses valores, fomos analisar por família de produtos quais seriam as famílias com um maior número de A's da análise ABC, ou seja, a família de produtos que tivesse uma maior número de produtos do tipo A, maior número de máquinas do tipo A e maior número de rotas do tipo A, para assim termos um conjunto de dados mais específico à qual pudéssemos aplicar o modelo em questão. Para tornar a aplicação do modelo mais realista foi ainda tido em conta o número de rotas, ou seja, tendo em nossa posse um modelo válido para uma situação de *job-shop scheduling* teríamos todo interesse em abordar o

problema nesta vertente e não numa vertente de *flow-shop scheduling*. Assim sendo, o caso a aplicar deveria ter mais que uma rota. Neste caso de estudo cada família possui no máximo uma rota do tipo A, logo todas as famílias de produtos que não possuíam uma rota do tipo A foram excluídas.

4.1.3.1. Máquinas

Para realizarmos a análise ABC às máquinas foi necessário primeiro que tudo saber quantas vezes uma dada máquina era necessária para satisfazer a produção de cada produto. Na Figura 4.4 temos a análise ABC para todas as máquinas.

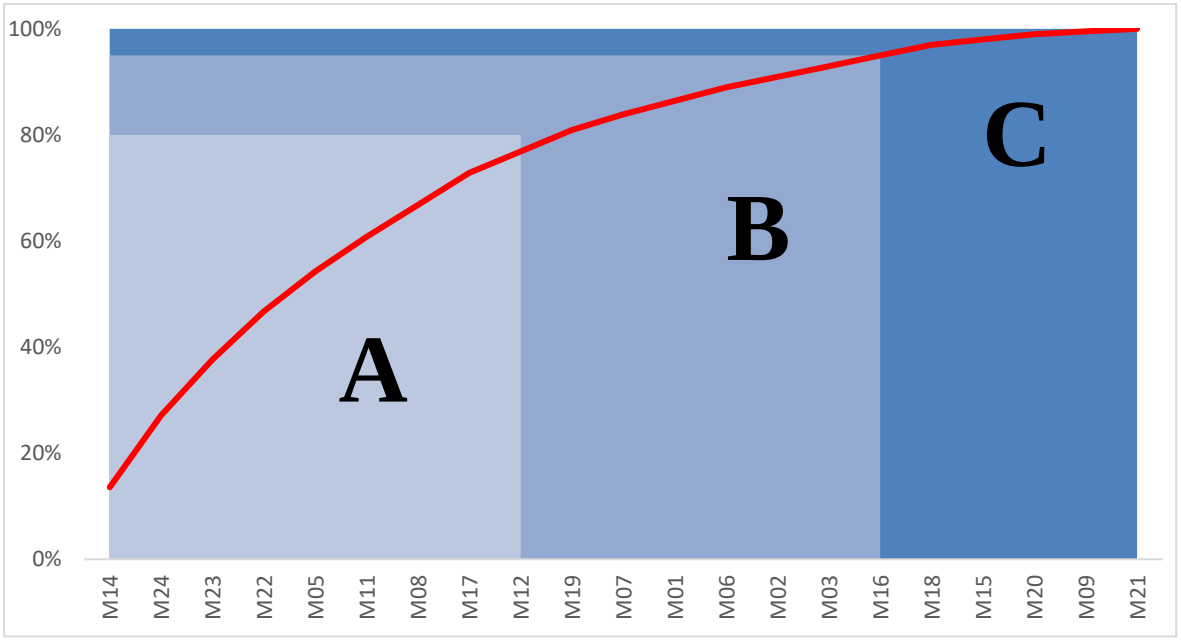

Figura 4.4. Análise ABC para todas as máquinas

Como podemos ver na Figura 4.4, temos cerca de nove máquinas do tipo A, de onde destacamos a presença da máquina 24 (M24) que, como já referido, é o gargalo do processo. É ainda interessante perceber que cerca de 43% das máquinas tem uma elevada importância para empresa, o que nos permite observar que a distribuição das máquinas não é a melhor, ou seja, a empresa está dependente apenas de várias máquinas para que consiga satisfazer a maior parte da sua procura.

4.1.3.2. Rotas

Uma abordagem semelhante à da análise ABC para as máquinas foi tida em conta para a análise ABC das rotas. Por outras palavras, estando na posse dos valores da

procura fomos observar a percentagem da procura à qual correspondia cada uma das rotas individualmente.

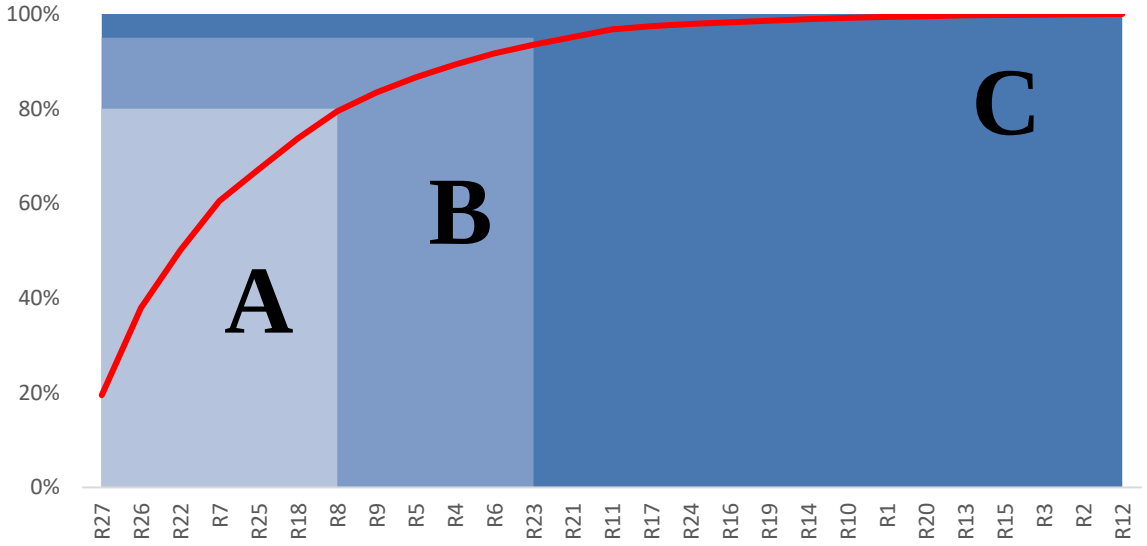

Figura 4.5. Análise ABC para todas as rotas

Na Figura 4.5 é percetível que temos sete rotas extremamente importantes para a empresa (rotas do tipo A), mais concretamente, essas rotas correspondem exatamente a 26% das rotas. É interessante, e expectável, que todas as rotas do tipo A (da análise ABC) tenham na sua constituição a máquina 24 (M24), a máquina gargalo do sistema.

4.1.3.3. Análise global dos dados

Por fim, foi necessário criar o nosso mini caso de estudo, digamos assim. Foi decidido verificar por família de produto número de produtos, máquinas e rotas do tipo A (da análise ABC) e fazer um somatório destes três valores. Contudo teria de haver uma coerência de máquinas, ou seja, verificar se as famílias resultantes desse somatório possuíam rotas que tivessem um núcleo semelhante de máquinas. Deste modo, verificámos as duas famílias com os valores mais elevados, uma vez que existiam duas rotas diferentes com um número de máquinas semelhantes elevado. Escolhemos duas famílias em vez de uma, para o problema ser considerado um problema de *job-shop scheduling* em vez de um problema de *flow-shop scheduling*. Assim sendo, as famílias selecionadas foram FE e FH.

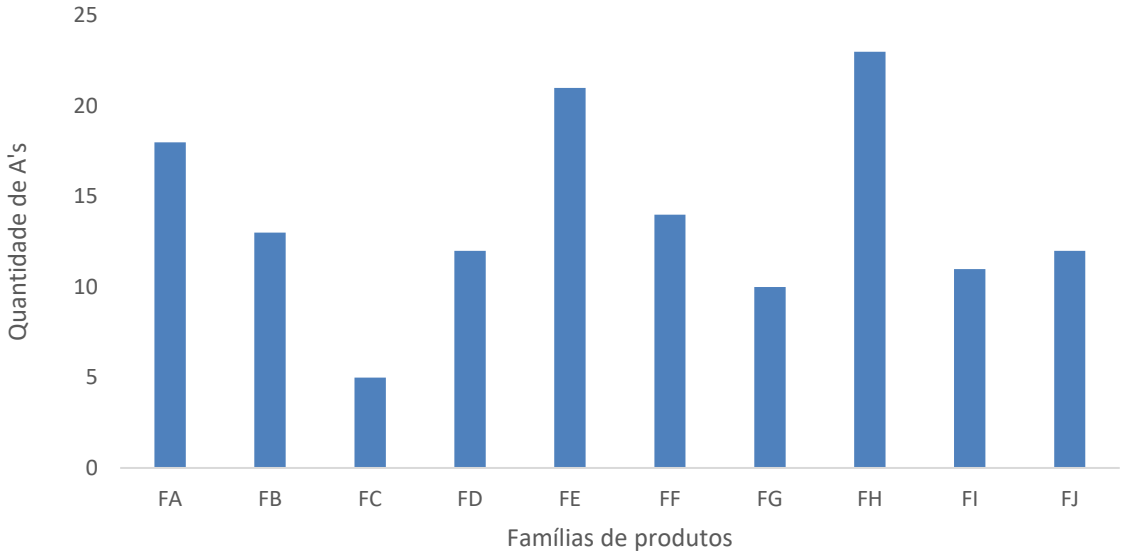

Figura 4.6. Combinações do tipo A

Uma vez mais, através da ferramenta *Power BI*, foi realizado um diagrama de *Sankey* agora considerando apenas as rotas do tipo A das famílias escolhidas. Como podemos ver na Figura 4.7 já temos uma situação mais limpa, ou seja, sem tantos cruzamentos entre fluxos. Desta forma, a aplicação do modelo ao problema já se torna mais adequada, visto que reduzimos drasticamente o número de produtos e também o número de máquinas.

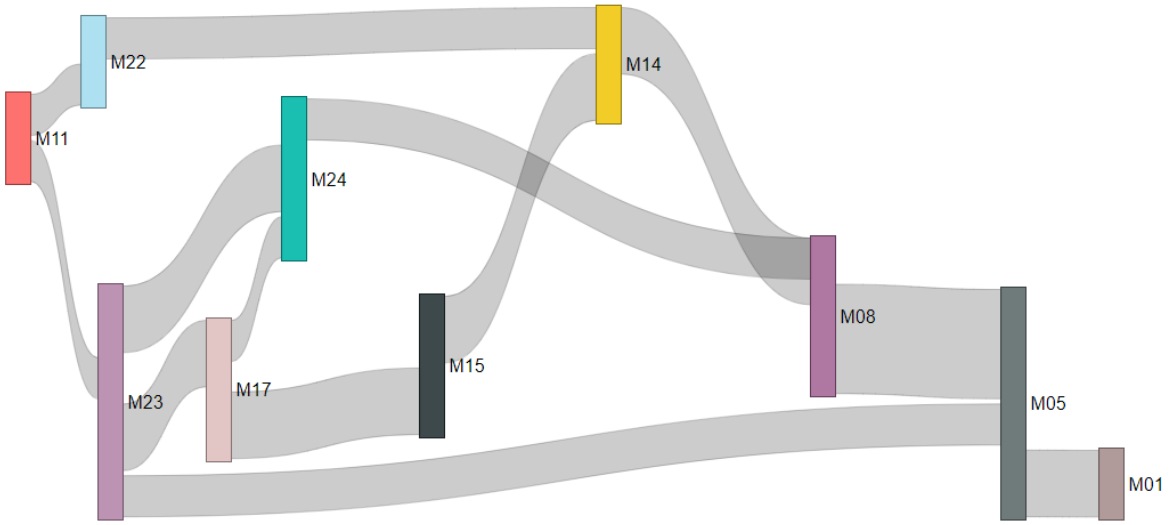

Figura 4.7. Diagrama de *Sankey* para FE e FH

4.1.3.4. Jobs considerados

Era necessário encontrar uma combinação de *jobs* que correspondesse a ambas as famílias de produtos. Então fomos calcular a importância que cada produto tinha na sua família, ou seja, a percentagem de vendas de cada produto do tipo A (da análise ABC) por FE e FH representados na Figura 4.8 e Figura 4.9 respetivamente.

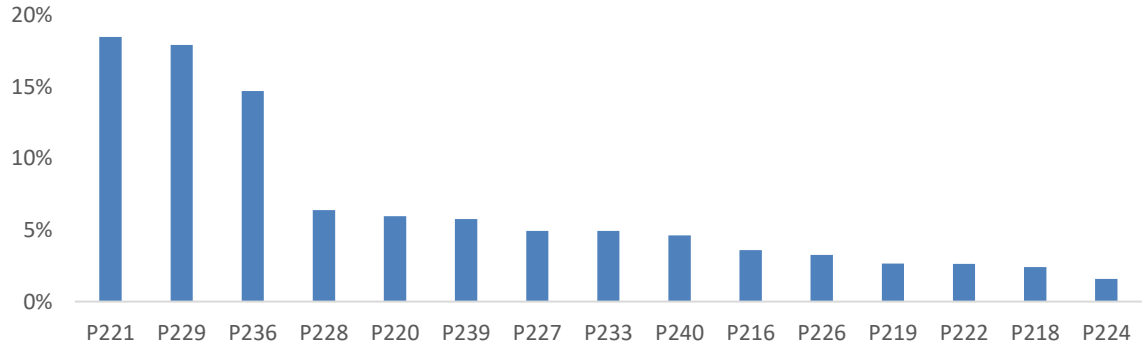

Figura 4.8. Vendas de cada produto do tipo A por FE

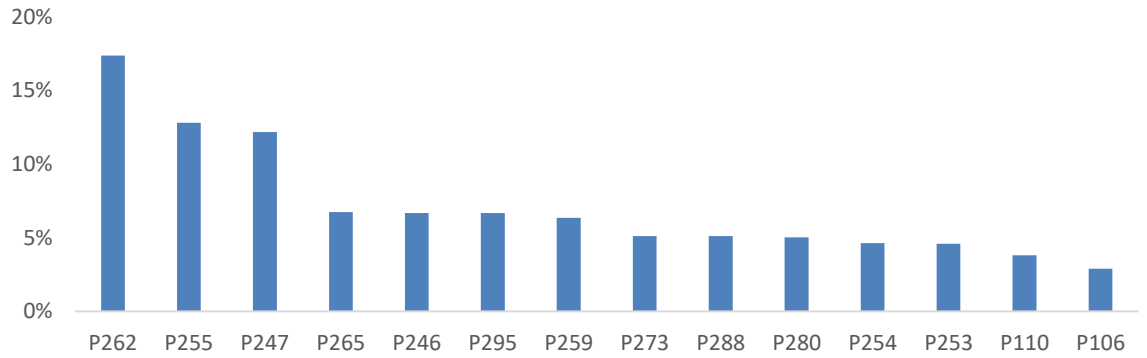

Figura 4.9. Vendas de cada produto do tipo A por FH

Numa lógia semelhante, juntámos todos estes produtos e ordenamos os mesmos pela percentagem de importância e a partir desses valores escolhemos os produtos de maior importância que nos permitiram obter a Figura 4.10.

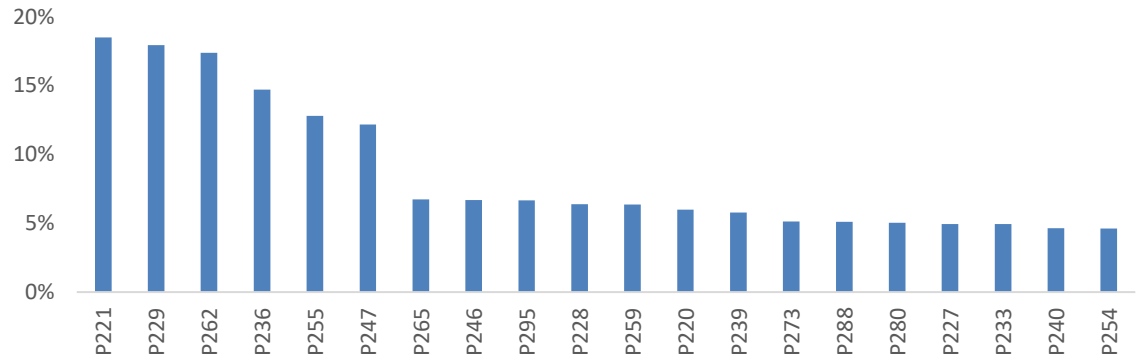

Figura 4.10. Combinação de vendas de cada produto do tipo A por FE com vendas de cada produto do tipo A por FH

Através desta combinação obtivemos os *jobs* que serviram de *input* do caso de estudo. Estes dez produtos são considerados como os produtos mais importantes para a empresa.

4.1.4. Dimensionamento dos lotes de produção

De forma geral nenhuma empresa faz o seu planeamento com um pensamento produto a produto, mas sim tendo conta lotes de produtos. Nesta ótica considerámos que os lotes de produção já estavam pré-definidos.

Contudo, ao considerarmos o produto como um conjunto de lotes existe uma clara desvantagem. O modelo em questão considera que o produto só vai para outra máquina ou *buffer* quando este termina a sua operação na máquina anterior. Ora, se estamos a falar de lotes isto não é totalmente verdade, pois um lote não se desloca para a máquina ou *buffer* seguinte quando o lote de produtos está terminado, mas sim quando os produtos correspondentes ao lote estão concluídos. Nesta fase, podemos concluir, que a aplicação de um modelo de dimensionamento de lotes em simbiose com o modelo desenvolvido seria uma abordagem interessante a perseguir no futuro.

4.1.5. Aplicação ao caso de estudo

Após todo o tratamento de dados seguiu-se a aplicação e corrida do modelo. Assim sendo, inicialmente considerou-se apenas cinco máquinas e dez *jobs*, dado que foi com esta tipologia que o modelo foi encontrou uma solução ótima. Numa fase seguinte,

adicionamos todas as máquinas que as rotas consideravam, para termos uma situação mais real, e, assim, todos os valores (*makespan*, tempo de computação, variáveis binárias e variáveis contínuas) subiram consideravelmente. Os valores das corridas estão representados na Tabela 4.3.

Tabela 4.3. Aplicação do modelo a ao caso de estudo

n*m	*Makespan*	Tempo (s)	Variáveis binárias	Variáveis contínuas
10*5	316	50.1	582	241
10*10	352	196.56	966	419
11*10	353	309.27	1194	455
12*10	368	4314.35	1416	495

Tendo os dados de *input* escolhidos e tratando-se de um problema de escalonamento, desenhámos novamente um diagrama de *Gantt*, mas agora com os dados reais do caso de estudo. O diagrama em questão está representado na Figura 4.11.

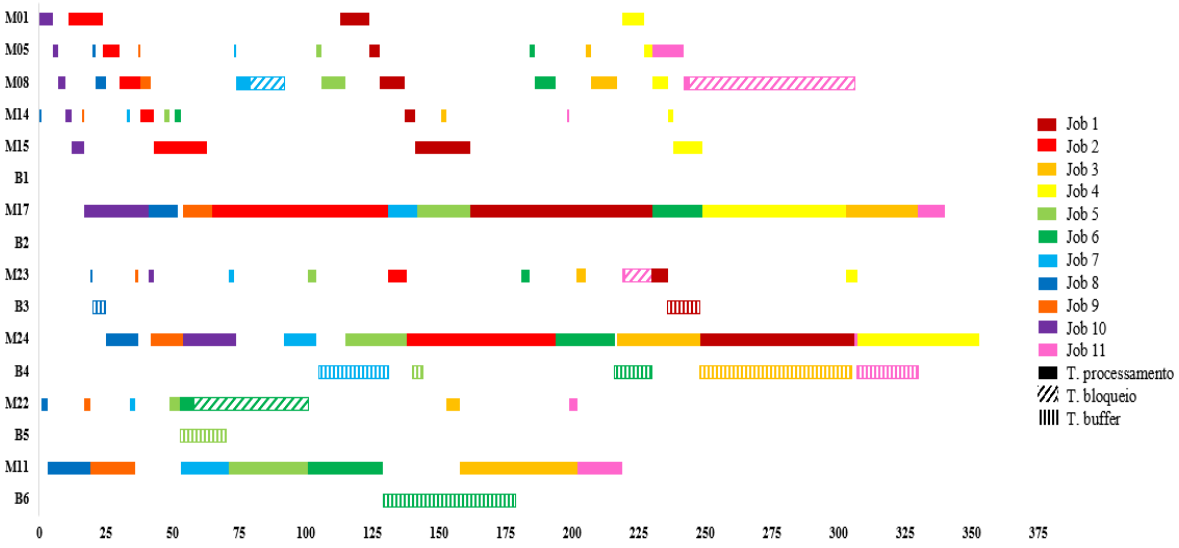

Figura 4.11. Diagrama de *Gantt* para o caso de estudo

4.2. Discussão de resultados

O diagrama de *Gantt* presente na Figura 4.11 permite-nos concluir que as máquinas 17 (M17) e 24 (M24) são aquelas que condicionam o processo a nível temporal e que a máquina 24 (M24) é o gargalo neste caso (e também é o gargalo do processo). A nível

dos *buffer*s, como é percetível, existem *buffer*s que nem se quer são utilizados, logo tornou-se interessante perceber a taxa de utilização dos *buffer*s, que poderemos observar na Figura 4.12.

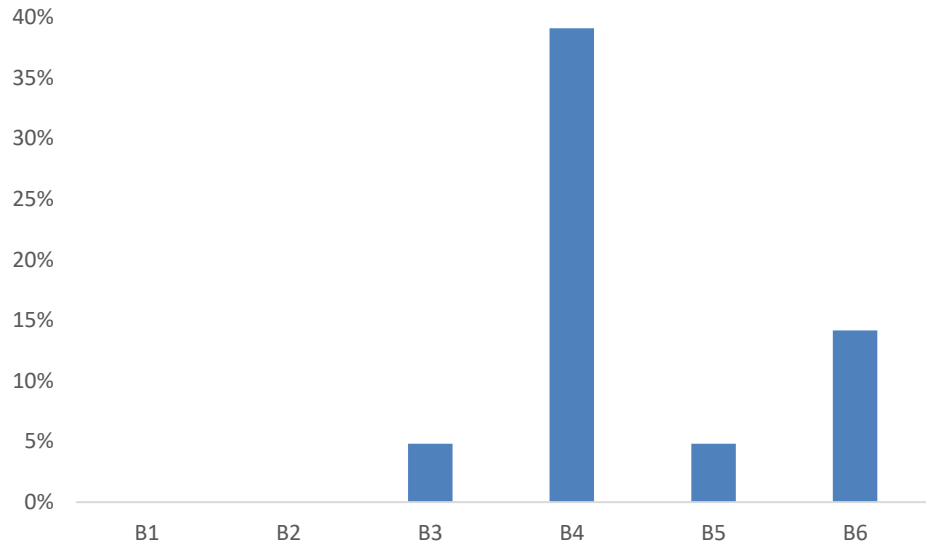

Figura 4.12. Taxa de utilização dos *buffer*s

O *buffer* quatro (B4) é o *buffer* mais utilizado, pois está associado à máquina 24 (M24) que é o gargalo do sistema. Este facto é interessante, pois é um valor bastante atrativo, na medida em que é altamente justificável um *buffer* nesta situação, ou seja, se o *buffer* quatro (B4) não existisse o *makespan* seria efetivamente maior. A taxa de utilização dos *buffer*s um (B1) e dois (B2) é nula, para esta situação, o que pode levar à reflexão sobre a sua existência. Contudo, estamos a trabalhar apenas com onze *jobs* e com duas rotas, quando na realidade para essas rotas o número de *jobs* é maior. Numa situação ideal (em que todos os *jobs* fossem considerados), a taxa de utilização dos *buffer*s um (B1) e dois (B2) muito provavelmente deixaria de ser nula.

O tempo de bloqueio, ou tempo não produtivo, é outro dos fatores que deve ser analisado. Na mesma ótica dos tempos de *buffer* analisámos os tempos de bloqueio, para assim perceber o seu impacto no processo, como podemos observar na Figura 4.13.

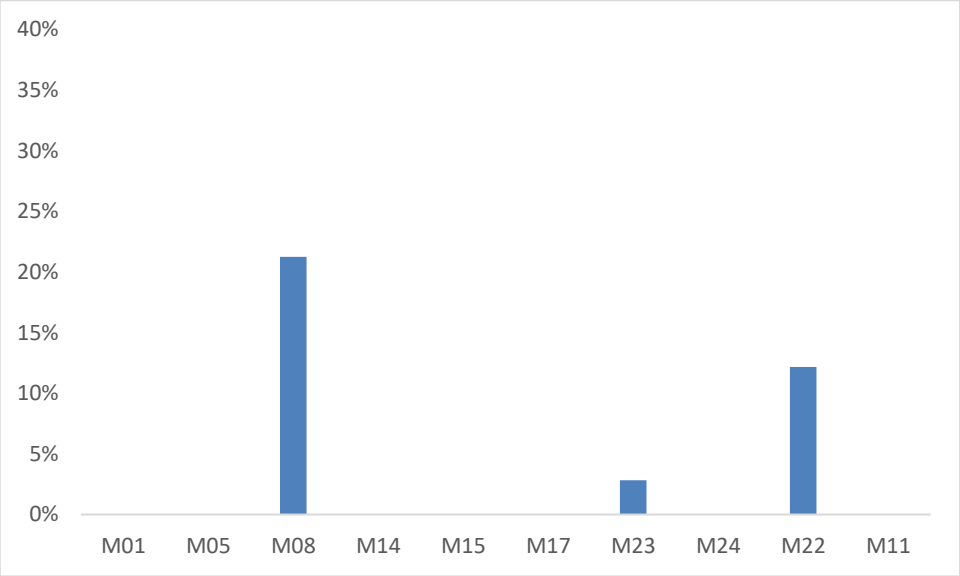

Figura 4.13. Taxa de tempo de bloqueio

Como podemos observar pela Figura 4.13 o tempo de bloqueio é praticamente nulo na maioria das máquinas, este fator deve-se essencialmente à existência de *buffer*s e a restrição de *no-buffer*. Contudo, a máquina 8 (M08) possui um tempo de bloqueio relativamente alto em comparação com as restantes, mas neste caso concreto parte do tempo deve ser ignorado. Ao termos em atenção a Figura 4.11, conseguimos observar que este valor percentual elevado se deve ao último *job* (*Job* 11) que passa na respetiva máquina e esse tempo não tem qualquer impacto no *makespan*, uma vez que é inferior ao mesmo. A máquina 22 (M22) também possuí um tempo de bloqueio considerável que se deve a dois fatores. Primeiro os *buffer*s só possuem capacidade para um *job* e neste caso o *buffer* associado à máquina (B5) está ocupado, em segundo lugar o *job* em questão (*Job* 6) pretende deslocar-se para a máquina 24 (M24) que está ocupada. Em suma, seria mais interessante uma análise detalhada ao tempo de bloqueio da máquina 22 (M22) relativamente ao da máquina 24 (M24), dado que este tempo de bloqueio está diretamente relacionado com o dimensionamento e alocação dos *buffer*s.

5. CONSIDERAÇÕES FINAIS

Esta dissertação teve como objetivo desenvolver uma abordagem de otimização para encontrar tamanhos ótimos de *buffer* em linhas de produção. Nesse sentido, um modelo de otimização foi validado e aplicado num caso de estudo real de uma empresa de fabrico de móveis. Para a aplicação do modelo, foi necessário e fundamental o estudo de um solver comercial - o IBM ILOG-CPLEX 12.8.

A abordagem metodológica proposta dividiu o problema *core* em vários problemas de menor dimensão, de forma a que os sub-problemas possam ser resolvidos diretamente pelo modelo de otimização. O modelo foi aplicado em apenas duas famílias de produtos, a duas rotas (uma de cada família), a onze *jobs* e dez máquinas. As soluções permitiram dimensionar os *buffers* e, de forma geral, o escalonamento de produção. A aplicação do modelo a um caso de estudo foi efetuada como sucesso, pois os resultados verificados na literatura foram aqui verificados.

Como trabalhos futuros sugere-se a aplicação de um modelo de alocação de *buffers*, dado que a alocação das restrições de *buffer* teve como uma base uma ideia lógica e não uma base científica. Para além de um modelo de alocação de *buffers*, um modelo de loteamento poderia também ser aplicado, uma vez que a questão do loteamento é assumida como definida. Seira ainda interessante considerar no modelo os tempos de transporte das operações entre máquinas e perceber a influência destes mesmos tempos no *makespan*. Uma outra abordagem, interessante em diversas maneiras, seria criar um modelo de simulação que representasse o trabalho feito, seria ótimo para retirar algumas conclusões adicionais e ainda ter uma representação real do caso de estudo.

REFERÊNCIAS BIBLIOGRÁFICAS

Aldowaisan, T.e Allahverdi, A. (2015), "No-Wait Flowshops to Minimize Total Tardiness with Setup Times", Intelligent Control and Automation, 06(01), pp. 38–44.

Amiri, M. e Mohtashami, A. (2012), "Buffer allocation in unreliable production lines based on design of experiments, simulation, and genetic algorithm", The International Journal of Advanced Manufacturing Technology, 62(1–4), pp. 371–383.

Carlier, J. e Pinson, E. (1989), "An Algorithm for Solving the Job-Shop Problem", Management Science, 35, pp. 164-176

Demir, L., Tunali, S. e Eliiyi, D. T. (2014)., "The state of the art on buffer allocation problem", A comprehensive survey. Journal of Intelligent Manufacturing, 25(3), pp. 371–392.

Ding, J.-Y., Song, S., Gupta, J. N. D., Wang, C., Zhang, R. e Wu, C. (2016), "New block properties for flowshop scheduling with blocking and their application in an iterated greedy algorithm", International Journal of Production Research, 54(16), pp. 4759–4772.

Fink, A. e Voß, S. (2003), "Solving the continuous flow-shop scheduling problem by metaheuristics", European Journal of Operational Research, 151(2), pp. 400–414.

Fisher, H. e Thompson, G.L. (1963), "Probabilistic Learning Combinations of Local Job-Shop Scheduling Rules", Prentice-Hall, Englewood Cliffs, pp. 225-251.

Grabowski, J. e Pempera, J. (2007), "The permutation flow shop problem with blocking", A tabu search approach. Omega, 35(3), pp. 302–311.

Hall, N. G. e Sriskandarajah, C. (1996), "A Survey of Machine Scheduling Problems with Blocking and No-Wait in Process", Operations Research, 44(3), pp. 510–525.

Hauptman, B. e Jovan, V. (2004), "An approach to process production reactive scheduling", ISA Transactions, 43(2), pp. 305–318.

Lawrence. S. (1984), "Resource Constrained Project Scheduling: An Experimental Investigation of Heuristic Scheduling Techniques (Supplement)", Carnegie Mellon University.

Leisten, R. (1990), "Flowshop sequencing problems with limited buffer storage", International Journal of Production Research, 28(11), pp. 2085–2100.

Liu, S. Q., Kozan, E., Masoud, M., Zhang, Y. e Chan, F. T. S. (2018), "Job shop scheduling with a combination of four buffering constraints", International Journal of Production Research, 56(9), pp. 3274–3293.

Louaqad, S. e Kamach, O. (2016), "Mixed Integer Linear Programs for Blocking and No Wait Job Shop Scheduling Problems in Robotic cells", International Journal of Computer Applications, 153(10), pp. 1–7.

Mascis, A. e Pacciarelli, D. (2002), "Job-shop scheduling with blocking and no-wait constraints", European Journal of Operational Research, 143(3), pp. 498–517.

Massim, Y., Yalaoui, F., Amodeo, L., Chatelet, E. e Zeblah, A. (2010), "Efficient combined immune-decomposition algorithm for optimal buffer allocation in production lines for throughput and profit maximization", Computers and Operations Research, 37(4), pp. 611–620.

Nahas, N. (2017), "Buffer allocation and preventive maintenance optimization in unreliable production lines", Journal of Intelligent Manufacturing, 28(1), pp. 85–93.

Nof, S. Y. (2006), "Collaborative e-work and e-manufacturing: Challenges for production and logistics managers", Journal of Intelligent Manufacturing, 17(6), pp. 689–701.

Pacciarelli, D. (2002), "Alternative graph formulation for solving complex factory-scheduling problems", International Journal of Production Research, 40(15), pp. 3641–3653.

Pranzo, M. e Pacciarelli, D. (2016), "An iterated greedy metaheuristic for the blocking job shop scheduling problem", Journal of Heuristics, 22(4), pp. 587–611.

Qian, B., Wang, L., Huang, D., Wang, W. e Wang, X. (2009), "An effective hybrid DE-based algorithm for multi-objective flow shop scheduling with limited buffers", Computers and Operations Research, 36(1), pp. 209–233.

Renna, P. (2018), "Adaptive policy of buffer allocation and preventive maintenance actions in unreliable production lines", Journal of Industrial Engineering International.

Sabuncuoglu, I., Erel, E. e Gocgun, Y. (2006), "Analysis of Serial Production Lines: Characterisation Study and a New Heuristic Procedure for Optimal Buffer Allocation", International Journal of Production Research, 44(13), pp. 2499-2523.

Samarghandi, H. e ElMekkawy, T. Y. (2013), "Two-machine, no-wait job shop problem with separable setup times and single-server constraints", The International Journal of Advanced Manufacturing Technology, 65(1–4), pp. 295–308.

Shi, C. e Gershwin, S. B. (2009), "An efficient buffer design algorithm for production line profit maximization", International Journal of Production Economics, 122(2), pp.725–740.

Smith, J. M. e Cruz, F. R. B. (2005), "The buffer allocation problem for general finite buffer queuing networks", IIE Transactions, 37(4), pp. 343–365

Zandieh, M., Joreir-Ahmadi, M. N. e Fadaei-Rafsanjani, A. (2017), "Buffer allocation problem and preventive maintenance planning in non-homogenous unreliable production lines", The International Journal of Advanced Manufacturing Technology, 91(5–8), pp. 2581–2593.

Zhang, C., Shi, Z., Huang, Z., Wu, Y. e Shi, L. (2017), "Flow shop scheduling with a batch processor and limited buffer", International Journal of Production Research, 55(11), pp. 3217–3233.

ANEXO A – MÁQUINAS PERTENCENTES A CADA UMA DAS ROTAS

Rotas	Máquinas								
R27	M01	M05	M08	M14	M15	M17	M23	M24	
R26	M14	M02	M03	M16	M09	M08	M24	M21	
R22	M14	M22	M11	M23	M05	M08	M24	M17	
R7	M14	M15	M02	M03	M16	M08	M24	M18	
R25	M02	M03	M07	M11	M14	M16	M18	M24	
R18	M14	M22	M23	M05	M06	M24	M17		
R8	M14	M22	M23	M05	M24	M17			
R9	M14	M01	M23	M05	M24	M17			
R5	M05	M14	M17	M22	M23	M24			
R4	M01	M05	M14	M17	M23	M24			
R6	M14	M02	M03	M16	M07	M24	M18		
R23	M14	M22	M11	M23	M05	M08	M24	M18	
R21	M14	M22	M11	M23	M05	M06	M24	M17	
R11	M14	M22	M11	M12	M08	M24	M19		
R17	M14	M22	M12	M11	M23	M08	M24	M19	
R24	M14	M22	M11	M23	M05	M06	M24	M17	M08
R16	M14	M22	M12	M11	M23	M07	M24	M19	
R19	M14	M22	M23	M05	M08	M24	M17		
R14	M14	M22	M23	M11	M12	M08	M24	M19	
R10	M14	M22	M11	M12	M07	M24	M19		
R1	M05	M14	M20	M22	M23	M24			
R20	M14	M22	M23	M05	M06	M24	M17	M08	
R13	M14	M22	M23	M12	M11	M08	M24	M19	
R15	M14	M22	M23	M11	M12	M07	M24	M19	
R3	M01	M05	M06	M14	M17	M23	M24		
R2	M01	M05	M14	M20	M23	M24			
R12	M14	M22	M23	M12	M11	M07	M24	M19	