Universidade do Minho
Escola de Engenharia

Vitor Duarte

Integração entre Modelos de Otimização e Ferramenta de Gerenciamento Visual para o Problema de Escalonamento em Máquinas de Produção em Lotes

Novembro de 2021

Universidade do Minho
Escola de Engenharia

Vitor Duarte

Integração entre Modelos de Otimização e Ferramenta de Gerenciamento Visual para o Problema de Escalonamento em Máquinas de Produção em Lotes

Dissertação de Mestrado em Engenharia Industrial

Trabalho efetuado sob a orientação do
Professor Doutor José António Vasconcelos Oliveira

Novembro de 2021

AGRADECIMENTOS

A ordem dos agradecimentos não significa nenhuma escala de importância.

Primeiramente gostava de agradecer aos meus pais pela oportunidade de continuar os meus estudos, um desejo pessoal que obteve todo suporte por parte deles.

Em segundo, gostava de ressaltar meu orientador, Professor José António Oliveira. Obrigado pelo apoio, horas despendidas e ensinamentos passados. Desde a cadeira de Otimização até nossa última reunião posso destacar que sempre pude aprender um pouco mais. Obrigado por aceitar as dificuldades provenientes de diferentes expressões do nosso vocabulário, sempre tentando contornar e conversar sobre possibilidades.

Meus sentimentos de gratidão por todo apoio nessa caminha que minha noiva Milena me deu. Todas as conversas, ideias, suporte, e mais, momentos que esteve comigo sempre que precisei, apoiando-me, pois, sabia do meu desejo e prazer em concluir esta Dissertação. Palavras não chegam. Amo-te.

À empresa ITZWOOD, Paulo Soares e Patrícia Soares. Obrigado pelos ensinamentos e oportunidade de verificar na prática processos e peculiaridades, possibilitando converter a situação em um estudo.

À Professora Doutora Liji Shen da Universidade de Dresden, que em um momento de dúvida sobre uma passagem do trabalho da mesma, respondeu meu email 10 minutos após enviá-lo. Parece pouco, mas gentilezas e atenção podem ser fatores decisivos para o desenvolvimento da edução e de um profissional. Obrigado.

Por fim, obrigado a todos os colegas e professores da Universidade do Minho que me aturaram, ensinaram e promoveram meu processo de aprendizagem.

DECLARAÇÃO DE INTEGRIDADE

Declaro ter atuado com integridade na elaboração do presente trabalho académico e confirmo que não recorri à prática de plágio nem a qualquer forma de utilização indevida ou falsificação de informações ou resultados em nenhuma das etapas conducente à sua elaboração.

Mais, declaro que conheço e que respeitei o Código de Conduta Ética da Universidade do Minho.

RESUMO

Integração entre Modelos de Otimização e Ferramenta de Gerenciamento Visual para o Problema de Escalonamento em Máquinas de Produção em Lotes

Setups são um dos principais causadores de desperdícios nos tempos de processamento em ambiente produtivo. É um tipo de atividade que corriqueiramente é citada como parte daquelas que não agregam valor ao processo de produção. Baseado em um estudo de caso visto em uma companhia produtora de móveis em madeira portuguesa, o presente estudo analisa um sistema onde *setups*, originados pela troca de processamento entre famílias em máquinas de produção em lotes, agravam os tempos de processamento dos artigos. Sendo um ambiente industrial que pratica o Sistema Kanban de produção, a utilização de kanbans físicos guia a produção. No entanto, em um dos estágios surge a necessidade da produção por lotes, visando eliminar ao máximo a ação dos tempos de *setups*. Sem nenhum processo previamente elaborado/padronizado no estágio a melhorar, o presente trabalho propõe a integração entre uma ferramenta *Lean* para gerenciamento visual do fluxo produtivo e modelos de otimização para realizar o escalonamento da produção. A partir da comparação com diversas ferramentas *Lean,* o *Rolling Kanban* é definido e apresentado como solução a utilizar para o controlo visual. O estágio de estudo é caracterizado como um ambiente composto por três máquinas em série (*Flow shop*) processadoras de lotes, nas quais a composição das famílias pode variar. Para o problema descrito, um modelo de programação linear mista é desenvolvido para a decisão de escalonamento da produção. Devido o modelo exato não comportar instâncias maiores do que dez tarefas, uma modificação da clássica NEH Heurística é proposta. Perante os modelos exatos e aproximados desenvolvidos, um exemplo é realizado propondo a integração do modelo junto à ferramenta *Rolling Kanban*, sintetizando as potencialidades de ganhos competitivos pela união dos métodos.

PALAVRAS-CHAVE

Rolling Kanban; Sistema Kanban; *Flow shop*; Modelo de Programação Linear Mista; NEH Heurística

ABSTRACT

An Integration Between Optimization Models and Lean Visual Control Tool Applied to Batch Processing Machines

Setups times are one of the most crucial causes of waste in processing times at the industrial environment. This type of activity is commonly mentioned as part of no add value's activities. Based on a case study seen in a Portuguese wood company, this current study analyses a system where setups times, originated due to changes between different batches of part of families, affect the makespan. As an industrial environment that practices the Kanban System of production, the use of physical kanbans guides production. However, in one of the stages there is a need for batch production, aiming to eliminate as much as possible the action of setup times. Without any previously elaborated/standardized process in the stage to be improved, this work proposes the integration between a Lean tool for visual management of the production flow and optimization models to carry out the production scheduling. Based on the comparison with different Lean tools, Rolling Kanban is defined and presented as a solution to be used to visual control. The study stage is characterized as an environment composed of three batch processing machines in series (*Flow shop*), in which the composition of the families can vary. For the problem described, a mixed linear programming model is developed for the production scheduling decision. Because the exact model does not support instances larger than ten jobs, a modification of the classic NEH Heuristic is proposed. Considering the exact and approximate models developed, an example is executed proposing the integration of the model with the Rolling Kanban tool, synthesizing the potential for competitive gains by combining the methods.

KEYWORDS

Rolling Kanban; Kanban System; *Flow shop*; Mixed Integer Linear Programming (MILP); NEH Heuristic

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABELAS

Lista de Abreviaturas, Siglas e Acrónimos

JIT – *Just in Time*

WIP – *Work In Process*

IO – Investigação Operacional

FIFO – *First-In-First-Out*

EDD – *Earliest Due Date*

SPT – *Shortest Processing Time*

SWPT – *Shortest Weight Processing Time*

APS – *Advanced Planning and Scheduling*

TI – Tecnologia da Informação

BPM – *Batch Processing Machine*

SD – Sequência Dependente

SI – Sequência Independente

1. INTRODUÇÃO

"*Lean Manufacturing*" ou simplesmente "*Lean*" é o termo que referencia uma filosofia para a gestão e aumento da eficiência dos processos (Van Der Krogt et al., 2010). Ligado diretamente com o termo "desperdício", a filosofia *Lean* vem sendo massivamente estudada ao longo dos anos, tendo como uma das principais justificativas o sucesso visto pela implementação na companhia automotiva "Toyota". Segundo Womack et al. (1990) o surgimento do *Lean* deve-se diretamente a companhia Toyota. Com justificativa para a afirmação dada devido que tal organização foi capaz de centralizar diversas técnicas eficientes para produções em massa, em que estas estavam anteriormente dispersas em outras empresas. Tal centralização propositou o início estruturado dos estudos da filosofia *Lean*, onde inclusive *Lean* e Sistema Toyota iniciaram como sinónimos. Todavia, a filosofia *Lean* possibilita a aplicação e surgimento de técnicas para específicas áreas, enquanto Sistema Toyota é apoiado exclusivamente nas técnicas vistas em tal organização.

Um dos subsistemas do Sistema Toyota de produção é o Sistema Kanban. O termo "Kanban" é definido como uma técnica para alcançar o *Just-in-Time* (JIT), pilar que sustenta o Sistema Toyota de produção. O sistema Kanban atua nos moldes de produções puxadas e ferramentas visuais para aumento da eficiência e redução de desperdícios nas empresas, em outras palavras, um melhor controlo produtivo (Monden, 2011). São diversas as situações de sucesso ao implementar os ditos "Kanbans" para controlo produtivo. Ao longo deste trabalho, exemplos seja de diferentes produções/ambientes como formas (físicas) que estes kanbans podem ter, foram identificadas e brevemente explicadas.

Mesmo com todos benefícios vistos com implementação do Sistema Kanban/ ferramentas *Lean*, ainda é encontrado uma certa dificuldade de implementar métodos quantitativos e/ou algoritmos matemáticos que possibilitariam ainda melhores resultados para o sistema (Van Der Krogt et al., 2010). "Complexidade" é o termo que justifica a não utilização métodos quantitativos e/ou algoritmos matemáticos. Alguns autores como Braglia et al. (2019) explicam que em muito dos casos os métodos de otimização/quantitativos fogem do dia-dia operacional. Entretanto, a junção de ferramentas *Lean* e estes métodos, quando bem implementados para o dia-dia operacional podem surgir como uma grande oportunidade para vantagens competitivas (Braglia et al., 2020).

No tocante aos métodos de otimização e modelos quantitativos em ambiente produtivo, a área do "Escalonamento" é uma das grandes áreas na comunidade de Investigação Operacional (IO) que fomenta estudos, principalmente em contexto de produção, desde meados de 1950 aos dias atuais

(Pinedo, 2016). Baseado, altas das vezes, em modelos matemáticos que visam a minimização do *tempo* de produção (*makespan*), a evolução do estudo do Escalonamento é sustentada pela implementação de modelos matemáticos e algoritmos que crescem com avaliação do contexto tecnológico. Logo, para produções com maior teor tecnológico, as técnicas de otimização para o escalonamento da produção representam uma oportunidade para o aumento efetivo da produção (Rossit et al., 2019a).

Como um dos elos entre Escalonamento e ferramentas *lean,* estão um dos principais agentes de desperdício e aumento do tempo da produção: os *setups. Setup* nada mais é que o tempo gasto entre o processamento de duas tarefas ou dois lotes. Os *Setups* são um dos principais exemplos de atividades que não atribuem valor, logo, encaixam-se no objetivo das técnicas *lean*, que buscam acabar com desperdícios. Ao mesmo tempo, são objetos de estudo para modelos de otimização para minimizar *makespan* (ou qualquer outro objetivo influenciado pelos *setups*) e ampliar os horizontes da utilização dos algoritmos e modelos desenvolvidos ao longo dos anos (C. Y. Cheng et al., 2021; Monden, 2011).

Assim, e sendo o estudo do Escalonamento e a filosofia *Lean* possibilitadores de ganhos operacionais, um dos objetivos deste trabalho é possibilitar e propor a integração entre métodos de otimização no contexto de produções que utilizam a filosofia *Lean.* Entretanto, algumas dificuldades podem, desde já, serem mencionadas: os estudos para Escalonamento muitas das vezes deram-se com foco teórico e; as companhias não são iguais a companhia Toyota, logo, uma série de adaptações para os diversos casos são necessárias.

A real motivação para este trabalho é originada por um caso prático visto em uma empresa portuguesa produtora de artigos em madeira, onde tal organização faz uso de kanbans físicos para controlo do fluxo produtivo. No entanto, um problema surge em um dos estágios produtivos, nomeadamente o estágio da pintura, no qual a produção por lotes faz-se necessária. Assim, uma ferramenta para controlo (físico e visual) desses kanbans e lotes precisa ser implementada. Juntamente com a ferramenta, a padronização do escalonamento produtivo faz-se necessária. Além de atualmente não existir um método definido para escalonar os kanbans na organização, caso aderido uma nova metodologia, esta deve-se fazer parte de um processo bem definido.

Para produções em lotes em meio ao Sistema Kanban, o presente trabalho por meio de uma estruturada pesquisa académica, propõe a ferramenta *Rolling Kanban*, trabalhada em Braglia et al. (2020), como solução para este caso. Em modos gerais, a ferramenta *Rolling Kanban* consiste em uma técnica para controlo visual em produções de lotes, onde os *setups* são altos pela troca de famílias dos artigos

(Braglia et al., 2020). Assim, um outro objetivo do trabalho é trazer mais um caso onde tal ferramenta pode ser utilizada, tendo em vista a escassez de trabalhos académicos que trabalham o tema.

Para o caso da padronização e utilização de um método que defina o escalonamento da organização, o presente trabalho tem também como objetivo adaptar e expandir o modelo matemático descrito em Shen e Gupta (2018), Atribuindo as modificações para o estudo de caso visto. No entanto, sendo os modelos exatos grandes utilizadores de recurso computacional, em certas situações os métodos exatos não são elencáveis para solução dos problemas. Especificamente para produções onde o número de tarefas ou famílias de artigos ou máquinas, estes modelos podem requerer um tempo de solução impossível para o dia-dia. Para contorno, o presente trabalho propõe o uso de métodos heurísticos, nomeadamente uma adaptação da NEH Heurística proposta em Nawaz et al. (1983).

Portanto, e sintetizando, o presente trabalho tem como objetivo geral explorar um caso visto em uma empresa produtora de artigos em madeira em meio a integrar métodos de otimização com uma ferramenta da filosofia *lean*.(*Rolling Kanban*). De modo a padronizar o processo, controlando-o com uso do *Rolling Kanban*, e, definindo um método para encontro do escalonamento dos lotes a produzir.

Para alcançar o objetivo geral, o trabalho sustenta-se em objetivos específicos, sendo estes:

- Estudar a literatura existente para escalonamento da produção em lotes e ferramentas de gerenciamento visual;
- Propor uma ferramenta de gestão visual capaz de orientar o processo produtivo do estudo de caso;
- Desenvolver um modelo de programação linear inteira que retrate a situação real e o(s) objetivo(s) a otimizar;
- Desenvolver e adaptar um algoritmo capaz de fornecer a sequência de ordens de trabalho a serem executadas pelas respetivas máquinas;
- Reestruturar o processo produtivo de maneira a integrar a ferramenta visual proposta e o algoritmo desenvolvido. Portanto, padronizar o processo produtivo do estágio em questão.

O restante do trabalho foi estruturado como segue. O Capítulo 2 concentrou-se na revisão da literatura, sendo fundamentalmente segmentado em JIT, Sistema Kanban, *Setups* no Sistema Kanban e técnicas para controlo visual (especificamente *Rolling Kanban*). A segunda parte do Capítulo 2 teve o Escalonamento como assunto central, destacando o respetivo papel em ambiente produtivo e os principais conceitos envolvidos. E por fim do capítulo, o ambiente *Flow shop* (contexto do estudo de caso)

e suas particularidades, nomeadamente para a existência dos *setups*, produção por lotes e o conceito de *Non-permutation Flow shop*. O Capítulo 3 caracterizou a empresa que motivou o estudo de caso. Assim, sector de atuação, estágios produtivos e enquadramento quando aos conceitos explicados no Capítulo 2 foram descritos para situar o leitor do problema a ser resolvido. O Capítulo 4 reuniu a metodologia utilizada neste trabalho. Foi explicado mais profundamente a ferramenta *Rolling Kanban*, assim como o modelo de programação linear desenvolvido, em especial as adaptações e extensões utilizadas a partir do trabalho de Shen e Gupta (2018). Finalizando o Capítulo 4, realiza-se uma explicação da NEH Heurística e a modificação proposta. Para o Capítulo 5, as instâncias utilizadas para testar modelo matemático são apresentadas. Logo em seguida, o solucionador é apresentado com respetiva razão de escolha. Seguindo, a justificação de uso da NEH Heurística é desenvolvida, baseado nos dados da organização estudada. Por fim do capítulo, o algoritmo e modelo matemático descritos são introduzidos no contexto do *Rolling Kanban*. No Capítulo 6, último do trabalho, centralizam-se as conclusões, recapitulação dos objetivos do trabalho e possíveis estudos futuros sobre a temática.

2. REVISÃO DA LITERATURA

Neste capítulo realiza-se uma revisão dos principais conceitos contidos nesta dissertação. O capítulo divide-se em três secções, tendo em vista os objetivos do trabalho. A primeira secção aborda o âmbito de fabricações *Just In Time*. Conceituou-se produção "puxada" (*pull systems*) e o sistema de produção orientado por cartões (*Kanban system*). Ambas as definições estão contidas ao denominado "Sistema Toyota de Produção". A partir do Sistema Kanban, uma subsecção é reservada para esclarecimento do modelo original. Reviu-se também outros "kanbans" adaptados a diferentes sistemas. Ainda no Sistema Toyota de Produção foram introduzidas ferramentas/técnicas de controlo visual do Sistema Kanban, especificamente para ambientes em que existam tempos de *setups.* Por fim, concluiu-se com uma breve síntese do funcionamento da ferramenta (que foi aplicada no estudo de caso) nomeada "*Rolling Kanban*". A segunda parte do capítulo destinou a tratar do assunto Escalonamento. Primeiramente, o conceito e papel do Escalonamento são esclarecidos. Posteriormente, classificações e particularidades são citadas, visando a compreensão das diversas possíveis situações a serem estudadas. Por fim, são citados métodos exatos e aproximados para solução de problemas de otimização combinatória, nomeadamente programação linear e heurísticas, em contexto de Escalonamento. A terceira e última secção trata do ambiente conhecido como *Flow shop*, realizando uma análise histórica do *Flow shop*; *Flow shops* com existência de *setups*; máquinas processadoras de lotes; e, termina com uma discussão sobre *Permutation Flow shops* e *Non-permutation Flow shop*s.

2.1 JIT e Sistema Kanban

Ao longo dos anos, a produtividade alcançada pelas companhias japonesas despertou o interesse de estudo entre diversos profissionais ao longo do mundo. A principal empresa relacionada ao contexto dessas produções é a companhia automotiva Toyota. Para mais, o nome associado por trás desse sucesso, e dito como criador do "Sistema Toyota de Produção", é o do engenheiro, e na época vice-presidente da empresa, Taiichi Ohno (Mitchell & Schonberger, 1983; Sugimori et al., 1977).

O Sistema Toyota de Produção tem como uma de suas bases, ou, um de seus sistemas internos a filosofia *"Just in time"* / JIT. O conceito de produções JIT pode ser definido como uma filosofia que exige a redução dos stocks intermédios (*Work In Process* (WIP)), auxiliando no melhoramento do processo e diminuição de sua variabilidade. A ideia chave dessa filosofia é produzir certos itens, em certas quantidades, em um certo tempo (Ohno, 1988; H. Wang & Hsu-Pin (Ben) Wang, 1991).

Entretanto, é difícil realizar o JIT para planeamentos da produção onde as operações são realizadas em simultâneo, e, portanto, o fluxo da quantidade produzida no estágio predecessor não é mantido (*Push Systems*). Em geral, não se aplica a produção empurrada (*push*) para ambientes de alta customização, devido desperdícios incontroláveis (*e.g.* erro na quantidade produzida) oriundos da alta variabilidade do sistema. Assim, pode-se afirmar que a filosofia JIT atua melhor em produções que o fluxo do material processado é ordenado pelo estágio de produção anterior (produção puxada/ *Pull Systems*), suportando melhor a ocorrência de variações no sistema (Monden, 2011).

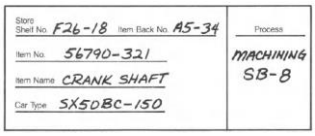
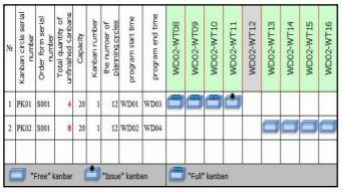
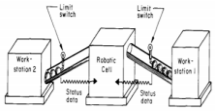
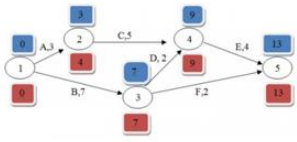
Dentro do Sistema Toyota de Produção define-se "Kanban" como uma das ferramentas para se conseguir produções JIT. Esse nome é atribuído a cartões (físicos ou digitais) que servem para autorizar o processamento, movimentação dos produtos não acabados (WIP) ou compra de material (matérias-primas e/ou insumos). Enquanto "Sistema Kanban", define o ambiente que se utiliza desses cartões para mover e controlar a produção. Em outras palavras, é um subsistema do Sistema de Produção Toyota (Sohal et al., 1989).

2.1.1 Particularidades do Sistema Kanban

O Sistema Kanban foi desenvolvido sob condições específicas, logo, é natural que existam dificuldades para sua implementação em ambientes que divergem do original. Entre as características divergentes, pode-se citar: longos tempos de preparação (*setup*), incerteza de fornecimento, operações não padronizadas, tempo de processamento instável, grandes flutuações de procura e ambientes com consideráveis distâncias físicas entre estágios produtivos (Ohno, 1982; Aggarwal, 1985). Entretanto, as características citadas acima são situações recorrentes nos mercados atuais, e assim, faz-se necessário, por parte das organizações, realizar adaptações visando sobrevivência (Van Veen-Dirks, 2005).

Diversos estudos de caso foram relatados usando o Sistema Kanban de maneira adaptada. Acerca destes estudos, desenvolveram-se modificações, seja ou na lógica produtiva ou no próprio kanban (cartão), consoantes a adaptação necessária. Lage Junior & Godinho Filho (2010) sintetizam casos de sistemas que aderem (ou não) à teoria do kanban original, variando suas configurações de acordo com o contexto empregado. Tais variações visam, dentre outras, as possíveis vantagens: a possibilidade de aderir procuras instáveis, facilidade para entrada de novos produtos, operar com diversos fornecedores, melhor balanceamento dos estágios de produção. A Tabela 1 representa uma síntese de aplicações reais, modificadas ou não, em que o Sistema Kanban foi inserido.

Tabela 1 – Exemplos de kanbans/Sistema Kanban

Nome da Abordagem	Vantagens	Aplicação real em	Comentários/Imagem
Kanban (ordem de produção)	**Redução dos stocks intermédios; fácil controlo visual**	**(Monden, 2011)**	
E-Kanban	**Possibilidade de atuar entre estágios produtivos (ou fornecedores) de grande distância física; visualização e controle em tempo-real.**	**(Mackerron et al., 2014);** **(Jin et al., 2011).**	
Sistema regenerativo de controle para produção puxada	**Atuação para sistema com alta variabilidade nos tempos de processamento; e com grande número de itens.**	**(Seidmann, 1988).**	
Job-shop Kanban	**Possibilita o uso do kanban para ambientes com diversos fluxos produtivos (Job-shop); e para ambientes de procura instável.**	**(Gravel & Price, 1988).**	**Diferentemente do original Sistema Kanban, nessa abordagem os cartões são destinados as operações e não aos produtos.**
Sistema Kanban modificado	**Atuação em ambientes com: grande quantidade de itens; máquinas com alto tempo de reparo.**	**(Otenti, 1992).**	**A linha de produção é dividida em equipes que controlam os respetivos inventários. Nessa abordagem um maior número de trabalhadores é necessário.**
Falso Sistema (kanban) de controle puxado	**Pode ser efetivamente utilizado em ambientes com gargalos produtivos.**	**(Hendrick, 1988).**	**Permite a abordagem empurrada (*push systems*) quando não é possível o sistema original kanban.**
Sistema Kanban baseado no método do caminho crítico	**Permite uma melhor coordenação do fluxo em estágios de montagem.**	**(Abdul-Nour et al., 1998).**	
Kanban com código de barra	**Efetivamente aplicado para fábricas com muitos fornecedores e procura dos produtos com instabilidade.**	**(Chaussé et al., 2000).**	

A primeira linha da Tabela 1 refere-se a um exemplo do kanban original e as subsequentes são compostas por exemplos com modificações. Apresentou-se, respetivamente, o nome da abordagem, as vantagens da aplicação, o trabalho fonte/referência e comentários ou uma imagem que permite prover uma noção da modificação. Como marco inicial da Tabela 1, define-se "kanban" como uma ferramenta do Sistema Kanban que visa alcançar a filosofia JIT (Monden, 2011). No exemplo em imagem da primeira linha da Tabela 1, tem-se o exemplo do kanban para produção. Esse cartão serve para desencadear a produção. O outro tipo de kanban inicialmente implantado no Sistema Kanban (na companhia Toyota) são aqueles que sinalizam a retirada de materiais nos stocks (em virtude da necessidade de material). As demais linhas da Tabela 1 destinam-se às variações (em reação da possibilidade de vantagens ou impossibilidade da original aplicação) de kanbans/Sistemas Kanbans.

- E-Kanban – Variação virtual do kanban original. Pode ser qualquer aplicação eletrónica que realize o fluxo da produção ou requisição de material. Mackerron et al. (2014) utilizam dessa abordagem em um estudo prático para requisição de material aos fornecedores. Como recente aplicação, Pekarcikova et al. (2020) desenvolvem um estudo de simulação para a mesma abordagem;

- Sistema regenerativo de controle para produção puxada – Proposto por Seidmann (1988), é uma aplicação automática das funções do kanban. Em outras palavras, o fluxo produtivo entre estágios desenrola-se automaticamente (controlo do WIP). A razão da automação nesse caso é tentar atuar em ambientes com alto grau de variabilidade/customização;

- Job-shop Kanban – Aplicado em Gravel e Price (1988), essa variação destinou-se para ambientes Job-shops, portanto, produção na qual diferentes fluxos produtivos são possíveis. Divergindo do kanban original, essa abordagem destina cada cartão para a operação e não ao produto;

- Sistema Kanban modificado – Desenvolvido em Otenti (1992). O autor propõe uma divisão do ambiente produtivo em várias equipas. Cada uma dessa com seus respetivos inventários. Essa variação de configuração permite uma rápida resposta para máquinas em que o reparo pode levar um elevado tempo;

- Falso Sistema (kanban) de controle puxado – A palavra "falso" é empregada por Hendrick (1988) para um sistema de produtos com baixo volume e alto custo. Nesse sistema a produção, quando permitida, utilizou-se da lógica empurrada (*push*);

- Sistema Kanban baseado no método do caminho crítico - Abdul-Nour et al. (1998) propuseram uma abordagem do kanban dependente do método do caminho crítico. Comumente relacionado à gestão de projetos;
- Kanban com código de barras – É o exemplo de kanban do qual se utiliza a empresa que destina o estudo de caso dessa dissertação. Permite fácil acesso às informações a produzir; possibilita uma melhor adaptação para procuras instáveis e rápido acesso às quantidades existentes em stocks.

Para uma última observação dos kanbans apresentados na Tabela 1, estas modificações podem ou não seguir as quatro principais funções da ferramenta kanban: 1 - Gerir uma visualização/sinal entre estágios; 2 - Coordenar interno ao estágio e na passagem para o estágio seguinte; 3 - Limitar o WIP; 4 - descentralizar o fluxo produtivo. Para os estudos referenciados na Tabela 1, os quatros primeiros (sem considerar o kanban original) seguem as funções do kanban original. Já os restantes modificam sua função em algum dos quatro pontos (Lage Junior & Godinho Filho, 2010). Entretanto, e demonstrando a versatilidade da ferramenta, o kanban utilizado no estudo de caso é um kanban com código de barra e segue as principais funções do sistema kanban original.

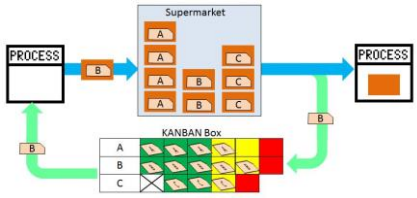
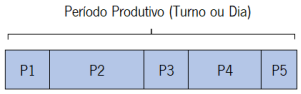
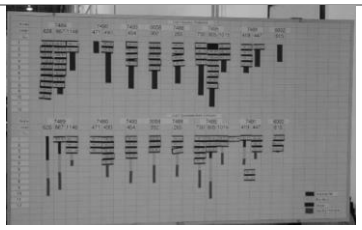
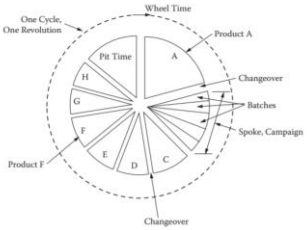
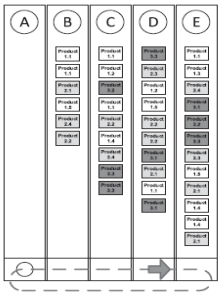
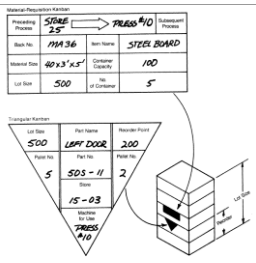
2.1.2 Kanban em função dos tempos de *setup*

Uma das principais características que faz o Sistema Kanban modificar-se é a existência de tempo de *setup* (Millstein & Martinich, 2014). Esses tempos são aqueles decorridos pelas trocas de processamento, início de operação ou finalização do dia produtivo. Generalizando, são tempos despendidos com preparação ou troca de processamento (Chris N. Potts & Kovalyov, 2000). Perante elevados tempos com *setups*, Millstein e Martinich (2014) contraindicam a utilização da utilização do Sistema Kanban. Os autores relatam que existirá um alto desperdício de tempo em caso de se seguir o Sistema Kanban para ambientes com altos *setups*. Em outras palavras, a frequência de ocorrência de *setups* será grande devido o Sistema Kanban atuar sobre um fluxo produtivo puxado, e, caso tais *setups* despenderem muito tempo ou custos, os desperdícios serão elevados.

Visando o controlo produtivo em Sistema Kanban com tempos de *setups*, algumas técnicas de gerenciamento visual do fluxo produtivo foram desenvolvidas para adaptação quanto a tempos de preparação ou troca de processamento. A Tabela 2 dissemina exemplos dessas "técnicas kanban" para controlo/gerenciamento visual da produção mesmo com a ocorrência de tempos com *setups*, indicando,

respetivamente, nome, característica do *setup*, o trabalho fonte/referência de conceituação e uma imagem para entendimento visual.

Tabela 2– Técnicas de gerenciamento visual para Sistemas Kanban com existência de tempos de *setup*

Técnica Kanban	**Característica do(s) *setup(s)***	**Referência**	**Imagem**
Kanban *box* / *Faxbox*/ Caixa de correio	Tempos não existentes ou negligenciáveis. Não dependem da sequência de processamento.	(Hirano, 2009; Monden, 2011).	
Pattern production	Baixo tempo com *setup* e depende da sequência de processamento.	(Seidman & Holloway, 2002).	
Lot-making board	Baixo tempo com *setup* e depende da sequência de processamento. Uma sequência fixa não pode ser empregada.	(Gross & McInnis, 2003; Smalley, 2009).	
Roda Kanban dos *setups* por famílias de produtos	Tempos de *setups* baixos entre troca para produção de itens de mesma família e tempos elevados para diferentes famílias.	(King & King, 2018)	
Rolling Kanban	Tempos de *setups* baixos entre troca para produção de itens de mesma família e tempos elevados para diferentes famílias. Uma produção cíclica não pode ser estabelecida.	(Braglia et al., 2020)	
Kanban triangular ou por sinal	Inevitáveis elevados tempos de *setup.*	(Monden, 2011; Smalley, 2009)	

É possível, a partir da Tabela 2, perceber que os tempos de *setup* caracterizados na segunda coluna crescem ao longo do quadro (cima para baixo). Uma breve síntese de cada técnica é desenvolvida a seguir.

- Kanban *box* – É um exemplo para uma produção sem a existência de tempos para preparação. A técnica, denominada Kanban *box* armazena kanbans que chegam ao longo do tempo de maneira que a produção seguirá o fluxo da ordem de chegada. Portanto visualiza-se o *First-In-First-Out* (FIFO). Essa é a situação apropriada para se atingir o JIT (Braglia et al., 2020; Monden, 2011);
- *Pattern Production* – Técnica que leva em consideração produções dependentes da sequência de execução. Entretanto, é possível encontrar uma sequência ótima entre as ordens a produzir. Assim, estabelece-se uma sequência fixa onde os kanbans são ordenados de acordo como tal;
- *Lot-Making Board* – Aplicado a sistemas com dependência da sequência a produzir. Porém, diferente do ponto anterior, uma sequência fixa não pode ser seguida;
- Roda Kanban – Cada área da roda destina-se a um período de produção para uma família de produtos, acoplando também os tempos (elevados) de preparação existentes entre a troca de famílias. A roda gira em um sentido e aquela sequência é mantida para produção de todas as ordens;
- *Rolling Kanban* – Tem características de *setups* similares ao ponto anterior. No entanto, uma produção cíclica não pode ser desenvolvida em razão de procuras instáveis a curto prazo. Como solução, o *Rolling Kanban* desenvolve-se em um quadro no qual o operador monta a sequência produtiva de acordo com o horizonte estabelecido para execução. Essa técnica adapta-se à chegada constante de ordens;
- Kanban por sinal – Destina-se à produção por lotes. Entretanto, sempre decorrerá elevados tempos com *setups*, independente da sequência aderida.

O quinto ponto destaca a ferramenta *Rolling Kanban.* Os autores Braglia et al. (2020) resgatam a ferramenta a partir de um trabalho de consultoria da empresa italiana FESTO, descrido em Boyer (2004). Os autores justificam a importância da técnica devido a uma lacuna entre ferramentas existentes, especificamente uma que enquadre produções no qual a troca processamento decorra entre diferentes famílias. Com observação que tais trocas (entre diferentes famílias) acarretam altos tempos de *setup* (baixos tempos para troca entre produtos da mesma família).

Para essas produções, normalmente utiliza-se o conceito e aplicação dos lotes de produção, no qual uma máquina/recurso processa mais de um trabalho ao mesmo tempo (Chris N. Potts & Kovalyov, 2000). Nesse caso, o fluxo característico de sistemas puxados (primeira ordem a entrar no sistema, primeira a sair) deve ser quebrado para formação dos lotes. Em detrimento desse problema, o *Rolling Kanban* surgiu como uma ferramenta visual capaz de auxiliar o desencadeamento do Sistema Kanban para o processamento em lotes. Mais, com o pormenor da ocorrência de altos *setups* para troca de famílias e baixos para uma mesma família. O *Rolling Kanban* para além deste desencadeamento, promove também o escalonamento do processamento das ordens de produção (nesse contexto, kanbans), já que permite o realizador do processo visualizar as tarefas restantes e estrutura-las ao longo do período produtivo (Braglia et al., 2020).

2.1.3 Ferramenta de controlo visual – *Rolling Kanban*

A Figura 1 reúne dois exemplos de *Rolling Kanban.* Na Figura 1a representou-se a ideia inicial da configuração do quadro descrido em Boyer (2004). Já a Figura 1b revela o redesenho do trabalho de Boyer realizado por Braglia et al. (2020).

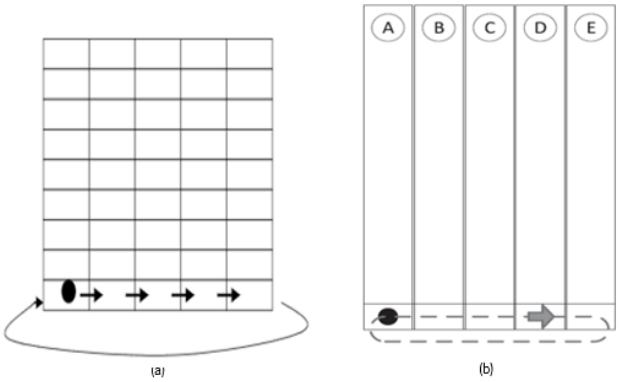


Figura 1 – Exemplos de Rolling Kanban; Figura 1a – Desenhado por Boyer (2004); Figura 1b – Desenhado por Braglia et al. (2020)

É relevante a menção que a fundamentação teórica por trás da ferramenta é escassa. Apenas os dois trabalhos mencionados formam o campo de trabalhos anteriores desenvolvidos sobre o *Rolling Kanban* (Braglia et al., 2020). Logo, como já mencionado, esta dissertação tem como um dos objetivos a expansão dos estudos e aplicações acerca da ferramenta. No restante desse subtópico uma breve explicação da ferramenta é realizada. Para além, uma aplicação prática com descrição pormenorizada é proposta no Capítulo 4.

A partir da Figura 2 é possível visualizar que alguns elementos são fundamentais para formação (física) da ferramenta. Inicialmente, um horizonte temporal precisa ser definido. Posteriormente esse horizonte é seccionado em períodos produtivos (dias, turnos, horas). Essas divisões são representadas pelas letras "A B C D E" da Figura 2b. Evidentemente, o horizonte temporal é a soma desses períodos. Por exemplo, se cada período representar um dia, o horizonte temporal (A+B+C+D+E) é igual a uma semana (5 dias) de planeamento da produção.

O início da produção é representado na Figura 2a. Cada ordem (kanban) foi disposta de acordo com a respetiva prioridade (*e.g.* kanbans com maior tempo de espera para processamento tem maior prioridade). Logo, itens alocados na coluna B são mais prioritários que itens em C e assim sucessivamente. O estágio atual da produção é evidenciado pelo objeto em branco na Figura 2a e preto na Figura 2b. Ao final de cada período produtivo, o objeto se desloca para direita (Figura 2a → Figura 2b). Em caso de chegada, durante o horizonte de planeamento, de novas ordens, os kanbans são postos a esquerda do objeto preto (Figura 2b), podendo haver atribuição a uma coluna que ainda falta ser produzida, caso haja capacidade produtiva.

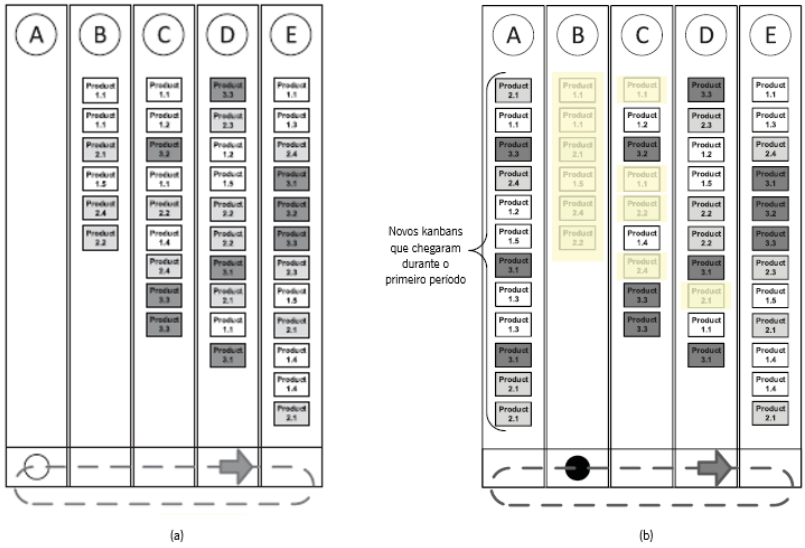


Figura 2 – Funcionamento Rolling Kanban; Figura 2a – Primeiro período produtivo (Braglia et al., 2020); Figura 2b – Segundo período produtivo (Braglia et al., 2020)

Sobre a Figura 2b, nota-se kanbans destacados. Estes foram os produzidos no primeiro período (A) e, portanto, eliminados do quadro (dando seguimento ao fluxo produtivo). A escolha de quais kanbans produzir (escalonamento das ordens) é realizada pelo operador. Seguindo alguns passos descritos em Boyer (2004).

1º passo: separar para realização dos kanbans mais urgentes;

2º passo: avaliar os outros kanbans (menos urgentes) que estão no quadro;

3º passo: montar uma sequência compatível com os kanbans mais urgentes e os outros kanbans de maneira a tentar otimizar *setups*. Realizar essa junção enquanto houver capacidade na estação de trabalho (máquina).

Destacando que no terceiro passo: (i) tentar juntar kanbans de mesma família; (ii) sequenciar dentro da família (lote) para o tempo de processamento ser mínimo (Boyer, 2004; Braglia et al., 2020).

Por fim, Braglia et al. (2020) detetam uma possibilidade de melhoria numa proposição de escalonamento tomando com base métodos de otimização. Tal proposta fundamenta o objetivo da dissertação e é aprofundada no Capítulo 4.

2.2 Escalonamento

"Escalonamento" foi o termo utilizado nessa dissertação para representar a tradução direta da palavra anglo-saxónica "*Scheduling*". Escalonamento vem sendo um assunto vastamente estudado ao longo dos anos. Alguns autores atribuem ao americano Henry Gantt o título de precursor nas atividades relacionadas ao escalonamento. Provavelmente tal título decorre devido ao enorme sucesso da ferramenta criada por Gantt, durante a primeira guerra mundial, o gráfico de Gantt. Embora a criação do gráfico de Gantt tenha se dado na década de 1910, apenas 40 anos depois os estudos científicos sobre escalonamento iniciaram desenvolvimento. Os problemas respetivos à escalonamento receberam uma enorme atenção dos pesquisadores e cientistas da área de Investigação Operacional, desencadeando estudos e publicações científicas desde meados da década de 50 até o presente tempo. Assim, com o passar dos anos, "*Scheduling*" formalizou-se como um dos grandes campos de estudos associado à comunidade de IO. Finalizando essa breve introdução, uma definição geral para Escalonamento é comumente vista: uma afetação dos recursos existentes no local analisado às tarefas/trabalhos que devem ser feitas durante um intervalo de tempo (Baker & Trietsch, 2009; Leung, 2004; Pinedo, 2016; C. N. Potts & Strusevich, 2009).

Durante o desenvolvimento dos estudos acerca do escalonamento, o ambiente industrial sempre foi o principal âmbito para entendimento e surgimento de novos problemas. Face à definição posta no parágrafo anterior, as fábricas e indústrias normalmente são constituídas por máquinas e trabalhadores (recursos) que processam ordens de produção (tarefas/trabalhos/atividades) em um determinado

período. Para manufatura, o foco do escalonamento será o processamento de itens para satisfação de um cliente (Framinan et al., 2014).

Essa pré-disposição dita dos estudos de escalonamento para com ambientes fabris pode ser justificada pelo contínuo desenvolvimento das indústrias e vasta variedade de configurações, processos e tecnologias. Pode-se historicamente evidenciar estudos determinísticos e estáticos que partem desde uma única máquina ou um conjunto de máquinas com processamento obrigatório (*Flow shop*) até áreas de grande interesse atual e que acompanham o desenvolvimento tecnológico como a Indústria 4.0 e Inteligência Artificial. Para os estudos primários decorridos na década de 50, destacam-se os famosos trabalhos de Jackson (1955) e o desenvolvimento de regra EDD (*Earliest Due Date*), Smith (1956) com as regras SPT (*Shortest Processing Time*) e SWPT (*Shortest Weight Processing Time*) e Johnson (1954) com a *regra de Johnson* para ambientes *Flow shop*. Já para escalonamento em contexto da Indústria 4.0 ver Rossit et al. (2019) e em contexto de Inteligência Artificial ver Laborie (2003).

Embora o ambiente de manufatura forneça um caráter prático e real aos estudos de escalonamento, não necessariamente o desenvolvimento científico desse campo deu-se com ênfase completa em casos ou instâncias reais. Pelo contrário, a teoria do escalonamento progrediu-se muitas das vezes tomando um exemplo de manufatura e realizando uma análise mais teórica da situação. Assim, em muitos dos casos, os métodos, ferramentas ou modelos utilizados como procedimentos de solução para uma certa situação não funcionariam no mundo real (Cowling & Johansson, 2002; Maccarthy & Liu, 1993; Stoop & Wiers, 1996). Nesse contexto entre escalonamento "real" e "teórico" é recorrente o exemplo em que os estudos mais teóricos tomam como base dados estáticos, quando na verdade os ambientes industriais tendem na imensa maioria das vezes a serem dinâmicos (Ouelhadj & Petrovic, 2009). Outra afirmação comum consiste na dificuldade de englobar todos os custos presentes nas operações, quando o objetivo está em obter a minimização dos custos (Framinan et al., 2014). Entretanto, a análise estática continua a ser importante. O uso estático dos dados pode ser justificado por razões de menor complexidade, descoberta de métodos de solução, comparações, encontro uma solução inicial ou análise da complexidade computacional de um certo problema. Para suportar as razões citadas, ver, respetivamente: Maccarthy e Liu (1993); Lu et al. (2020) e Rad et al. (2009); Ku e Beck (2016) e Lin et al. (2009); Floudas e Lin (2005) e Rudan et al. (2013); Ku e Beck (2016) e Lenstra et al. (1977).

Ainda em contexto de manufatura é fundamental estabelecer o papel do escalonamento na organização e tentar localizar em qual nível (estratégico, tático e operacional) a atividade pertence. Pinedo (2016) descreve o escalonamento como sendo uma atividade pertencente ao planeamento da produção, onde

desempenha um papel operacional (curto prazo). Em outras palavras, é uma atividade de todos os dias, em que o período temporal planeado pode ser desde semanas a dias ou até mesmo horas. Entretanto, o autor ressalva que as decisões tomadas em médio ou longo prazo devem ter em consideração a estratégia/método que compõe o escalonamento na organização. Assim como Pinedo, Framinan et al. (2014) e Framinan e Ruiz (2010) relatam que o escalonamento depende do plano agregado da produção, e mais, do período em que consistiu esse planeamento. Além do mais, o escalonamento segundo os autores está presente na atividade do nível mais baixo da planta da fábrica, ou seja, o "chão de fábrica". Por fim, os autores também afirmam que, tendo o planeamento como base, o papel do "escalonador" consiste em afetar tarefas às máquinas, criando a *sequência* de tarefas a processar nas respetivas máquinas durante aquele período produtivo. A Figura 3 fornece de maneira visual a definição dada por Framinan et al. (2014) e Framinan e Ruiz (2010), localizando a atividade de escalonamento dentro de um sistema APS (*Advanced Planning and Scheduling*).

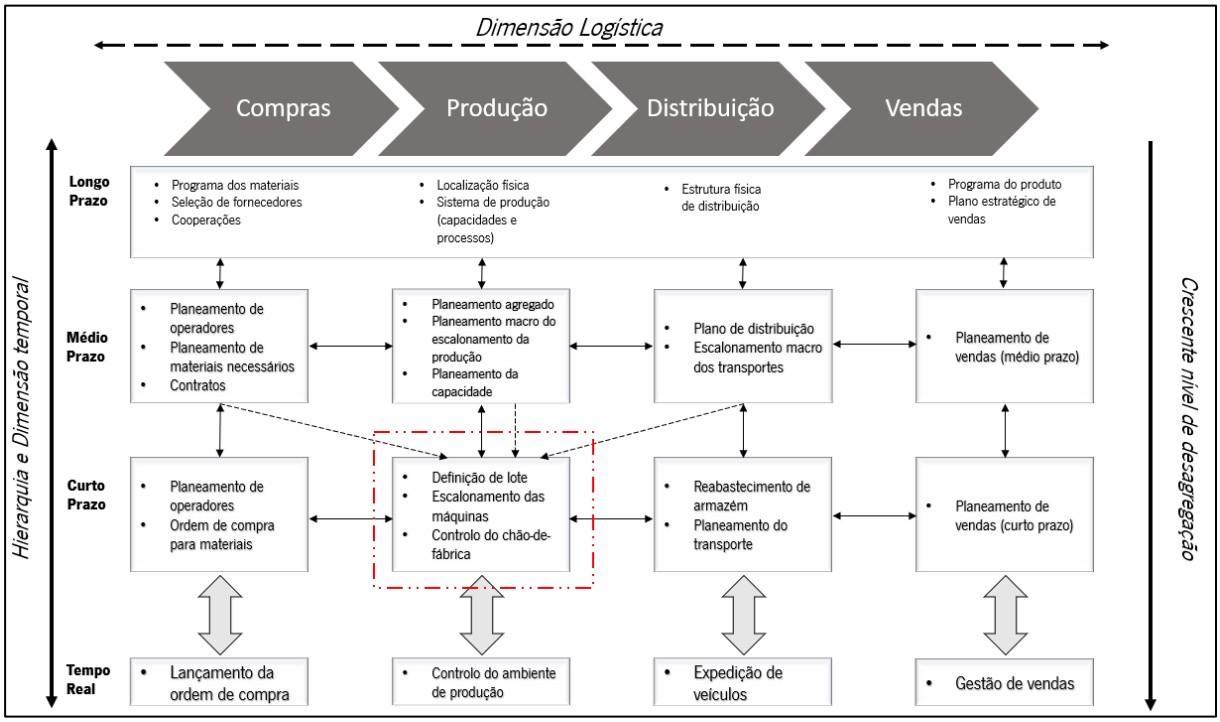


Figura 3 – Escalonamento num Sistema APS (Modificado e traduzido a partir de Framinan et al. (2014) e Fleischmann et al. (2008))

Sistemas APS são aqueles que, diferentemente do clássico planeamento de controlo da produção, incluem em uma perspetiva de Tecnologia da Informação (TI) a hierarquia das atividades do planeamento da produção (Eixo Y) e uma dimensão logística (Eixo X) (Framinan et al., 2014). A partir da Figura 3 nota–se a definição do papel do escalonamento posta no parágrafo anterior. No correspondente à hierarquia das atividades de produção, o escalonamento encontra-se no nível operacional (curto prazo),

porém a sua execução depende de dados oriundos do nível tático (médio prazo), que por sua vez interage com as atividades de longo prazo.

Os problemas de escalonamento em ambiente industrial são classificados por três fatores/campos. O primeiro dos fatores toma em conta os recursos (máquinas) existentes no ambiente, em outras palavras, o primeiro fator depende do *layout* fabril. O segundo fator ocorre em detrimento da operação realizada, ou seja, são peculiaridades daquele sistema para com o processamento das tarefas. O último critério é focado no critério de otimização, isso é, no objetivo que pretende-se melhorar na organização mediante ao escalonamento. Vários esquemas são propostos para representação da classificação por três fatores, dentre elas a proposta atribuída ao trabalho de Graham et al. (1979) que simboliza os três fatores pelos respetivos símbolos $\alpha|\ \beta|\ \gamma$ é a mais disseminada no âmbito científico. Entretanto o trabalho de Graham et al. (1979) trata-se de um aperfeiçoamento da classificação $\alpha|\ \beta|\ \gamma$ previamente abordada por Kan (1976). Cada respetivo campo ($\alpha|\ \beta|\ \gamma$) foi tratado no subtópico seguinte.

Para finalizar esta generalização do tema Escalonamento, algumas notações merecem esclarecimento para posterior entendimento nas questões analisadas. Todas as representações e definições gerais tiveram como base os livros e artigos de: Allahverdi et al. (2008); Brucker (2007); Framinan et al. (2014); Irani et al. (2007); Leung (2004); Pinedo (2016); C. N. Potts e Strusevich (2009); Chris N. Potts e Kovalyov (2000) e Rossit et al. (2018). Dado um problema de escalonamento, o conjunto $J = \{1, \dots, n\}$ representa um conjunto finito de n tarefas ou n trabalhos a processar. O conjunto finito $M = \{1, \dots, m\}$ engloba as m máquinas ou m recursos presentes no sistema. As operações vistas no sistema são descritas por O_{ij}, isto é, são todas as operações necessárias em que uma tarefa $j \in J$ deve ser processada numa máquina $i \in M$. p_{ij} simboliza o tempo de processamento da tarefa j na máquina i. As seguintes notações são associadas ao conjunto de tarefas J:

- r_j - Data que a tarefa j chega ao sistema. Logo, é a data mais cedo que essa tarefa estará disponível para processamento. Outra nomenclatura possível: *data de lançamento* de j;
- d_j - Data de vencimento/Data de entrega; data que é expectável a conclusão da tarefa. Uma observação pertinente: é permitido o término antecipado ou posterior, em ambos os casos podendo ou não haver uma penalidade pela antecipação ou atraso. Caso exista uma data máxima que a tarefa *deva* ser concluída atribui-se o termo *deadline* e símbolo $\overline{d_j}$ para esta condição;

- w_j - Representa o peso da tarefa j. Em outras palavras, caracteriza a importância ou custo atribuído para com aquela tarefa j, por exemplo: quanto maior tempo no sistema uma tarefa j_1 pode ter diferente custo de armazenamento que j_2.

2.2.1 Classificação em 3 campos - $\alpha|\ \beta|\ \gamma$

O campo α representa a configuração dos recursos (máquinas) no ambiente, ou seja, o *layout.* O campo α subdivide-se em α_1e α_2. α_2 retrata um número inteiro $K \in \mathbb{Z}^*$, que serve para quantificar a quantidade de máquinas existentes no *layout*. Já α_1 significa o tipo de *layout.* Para α_1, tem-se:

- Máquina única (1) – Em caso de só existir uma única máquina no sistema;
- Máquinas paralelas idênticas (P) – Ambientes com α_2 máquinas paralelas e iguais. O termo paralelo representa que uma tarefa j pode ser processada em qualquer uma das α_2 máquinas;
- Máquinas paralelas uniformes (Q) – São máquinas paralelas diferenciadas por suas respetivas velocidades. Logo, o tempo de processamento para um mesmo j difere (uniformemente) entre as diferentes α_2 máquinas;
- Máquinas paralelas não-relacionadas (R) – Máquinas também paralelas, mas que a velocidade não segue nenhuma relação entre as diferentes α_2 máquinas;
- *Flow shop* (F) – Ambiente de α_2 máquinas. Em que para todo $j \in J$ deve cumprir obrigatoriamente uma mesma sequência de máquinas (pré-estabelecida entre as α_2 máquinas);
- *Open shop* (O) – Ambiente também de $\alpha_2$máquinas, em que cada $j \in J$ deve passar por todas as máquinas, mas a sequência atribuída às tarefas pode ser alterada;
- *Job shop* (J) – Ambiente de α_2 máquinas, em que cada $j \in J$ tem sua própria necessidade (sequência) de máquinas, logo, pode, ou não, utilizar todas as máquinas existentes no sistema;
- Flexible *Flow shop* (FF_C) – Uma generalização do ambiente *Flow shop*, mas que contém c estágios com máquinas em paralelo;
- Flexible *Job shop* (FJ_C) – Uma generalização do ambiente *Job shop*, mas que contém c estágios com máquinas em paralelo;
- Mixed Shop (X) – É uma combinação entre os ambientes *Open shop* e *Job shop*;

O campo β indica as restrições existentes para o processamento das tarefas, por outras palavras, é como as operações O_{ij} ocorrem. O campo β agrega desde nenhuma atribuição até simultâneas representações. Em caso de nenhuma representação, isto significa que nenhuma particularidade nas operações deve ser levada em conta para entendimento da situação. Além do mais, a partir do momento

que regularmente novos problemas de Escalonamento são relatados no meio científico, o campo β pode conter diversas representações, tornando difícil relatar todas as restrições possíveis. Dentre as principais atribuições ao campo β, tem-se:

- Restrições de precedência ($prec$) – Indica operações O_{ij} que devem ser concluídas para que novas operações iniciem. Naturalmente, quando se trata de $\alpha_1 = F \vee O \vee J$ (ou variações) intrinsecamente $\beta = prec$, e, portanto, não precisa compor na classificação $\alpha|\,\beta|\,\gamma$. $\beta = chains, intree, outtree$ indica, respetivamente, uma relação de precedência em *Cadeia, Árvore dentro* (que significa 1 sucessor e vários predecessores)*, Árvore externa* (que significa vários sucessores e 1 predecessor). Para representação em grafo desses conceitos, ver Framinan et al. (2014, p. 77);
- *Preemptions* ($pmtn$) – Representa a possibilidade de um trabalho ser paralisado (portanto, não finalizado) e retomar posteriormente seu processamento na mesma ou noutra máquina;
- *Setups* ou custos com trocas (s_{ijk}) – Descreve a ocorrência de custos por trocas no processamento entre máquinas e/ou entre tarefas. Se ocorrer a existência de custos apenas pela troca entre os produtos $j \rightarrow k$ a representação pode ser dado apenas por s_{jk}. Se ocorrer custo diferentes para as m máquinas, o índice i é necessário. Caso a sequência de processamento não afete os custos, chama-se *sequência-independente*, e, portanto, $s_{jk} = s_{kj}$ entre as m máquinas. Os *setups* podem diferenciar-se também por *antecipatórios*, caso durante o processamento da tarefa na máquina predecessora possa-se *antecipar* a preparação (tempo de *setup*) na máquina sucessora, ou *não-antecipatório* caso contrário;
- *Permutation* ($prmu$) – É uma situação para ambientes *Flow shop*. Indica que a sequência estipulada para processamento deve-se manter invariante entre as máquinas. Para o caso de *Non-permutation* o campo β não precisa receber atribuições. Nota: a razão para a não realização da tradução direta deste termo se dá devido, na opinião do autor, a possibilidade de erro pelo significado que a palavra "permutação" (sinónimo: troca) induz na língua portuguesa;
- Pausas ($brkdwn$) – Da palavra de origem anglo-saxónica *Breakdown*, serve para representar quando uma máquina que não está disponível em todo o período de tempo estipulado. Pode indicar conhecidas quebras/avarias ou pausas para manutenção;
- *No-Idle* ($no - idle$) – Significa que não deve haver tempo de espera em qualquer máquina m (tempo ocioso). Por exemplo, não pode ocorrer pausas entre o processamento de duas tarefas.

Portanto, máquina é "ligada" apenas no momento em que de acordo com o escalonamento todas as tarefas possam ser executadas (naquela máquina) sem interrupções;

- *No-wait* (nwt) – Semelhante à *No-Idle*, nwt significa que uma tarefa j não deve *esperar* para iniciar seu processamento na máquina seguinte. Entretanto, em alguns casos um tempo máximo ou mínimo é permitido, e para estes casos $\beta = time\ lags$;
- Pulmão ($Buffer$ ou b) – Quando $\beta = Buffer = b$, isto expressa que existe entre máquinas uma restrição de espaço. Se a capacidade de armazenamento entre máquinas for infinita ($\beta = \infty$), a condição *No-Idle* deverá ser aplicada, e, a taxa de utilização das máquinas será máxima. Se a capacidade for finita para uma máquina $i \in M$, $\beta = b_i$. Se não houver qualquer local para armazenamento entre máquinas, então a condição a condição *No-Wait* deve ser aplicada ou $\beta = block$, que indica quando uma tarefa está finalizada na máquina predecessora porém bloqueada para início na máquina seguinte (que está em utilização);
- Lotes ($batch$) – Reproduz a existência de produção por lotes. Isto infere que mais de uma tarefa pode ser processada em uma máquina. Máquinas capazes de realizar a produção por lotes recebem a denominação de *Batch Processing Machine* (BPM). Essa representação pode variar entre *Serial Batching* ($s - batch$) e *Parallel Batching* ($p - batch$). *Serial Batching* traduz que o processamento das tarefas é realizado de maneira serial, ou seja, o tempo de processamento do lote é a soma dos tempos de processamento das tarefas presentes no lote. *Parallel Batching* representa uma BPM em que o tempo de processamento do lote é o maior tempo de processamento entre as tarefas presentes no lote, em outros termos, as tarefas são processadas em simultâneo (em paralelo). A ocorrência de lotes também pode diferir entre *lote disponível*, que significa que a transferência das tarefas contidas no lote deve ocorrer em conjunto (por exemplo, em paletes que carregam todas as ordens juntas), ou *tarefa disponível*, que é a situação em que uma tarefa finalizada no lote pode ser transferida para a máquina seguinte. Por fim, mais uma peculiaridade de produção por lotes, a denominação *lote inconsistente* quer dizer que os lotes podem diferentes configurações ao longo dos estágios produtivos. *Lotes inconsistentes* são essencialmente parte de um escalonamento *non-permutation*, já que sendo os lotes inconstantes, a sequência de processamento dos trabalhos tenderá a não ser fixa entre as máquinas (Shen & Gupta, 2018);
- Famílias ($fmls$) – Relata existência de famílias de produtos, que são produtos com características físicas similares ou com tecnologias de processamento semelhantes. Estendendo

o conceito de *setups* e lotes, s_{ijk} pode representar, por exemplo, o custo decorrido pela troca de processamento entre o lote de produção que continha a família j pelo lote de produção que contém a família k, na máquina i. Mais, *Tecnologia de grupo* e *Família de produtos incompatíveis* são dois termos presentes nesse contexto de Escalonamento. *Tecnologia de grupo* significa quando um grupo (lote) deve ser processado em conjunto, não podendo ser "repartido" em sub-lotes (*e.g. lot streamming*) (Cheng et al., 2000). *Família de produtos incompatíveis* infere que produtos de diferentes famílias não podem ser processados juntos em nenhuma circunstância (Mönch & Roob, 2018). Vale a ressalva que *Família de produtos incompatíveis* pode também ocorrer entre as máquinas, significando mudanças de famílias entre máquinas (Isenberg & Scholz-Reiter, 2013);

- Escalonamento Online ($online$) – Em geral, os problemas de escalonamento se dão de forma determinística/*offline* (M. Liu & Chu, 2012; Ouelhadj & Petrovic, 2009; Pruhs et al., 2004). Isto significa que todo o conhecimento acerca de tempos e disponibilidades é sabido antes do início do processamento. $\beta = online$ infere que as tarefas chegam ao longo do período de processamento, portanto, decisões em *tempo real* devem ser tomadas sobre essas tarefas.

Outras comuns afetações ao campo β são as já mencionadas r_j, d_j, $\overline{d}_j$ e w_j, representando, respetivamente, que o problema tem restrições relacionadas à data de lançamento, data de vencimento, *deadline* e custo/peso das tarefas $j \in J$. Demais atribuições ao campo β são normalmente autoexplicativas, caso não, cabe ao estudo em questão realizar a conceituação.

Como último campo da classificação $\alpha|\ \beta|\ \gamma$, a simbologia γ refere-se ao critério de otimização, logo, é a procura pela maximização ou minimização (ou ambas) de uma ou mais métricas. Brucker (2007) segmenta os tipos de critérios em duas classes. A primeira, chamada de *objetivos de gargalo*, estão objetivos preocupados com a(s) última(s) tarefa(s) a ser(em) concluída(s) (Equação 1). A segunda, são os *objetivos de soma*, em que se preocupam com a somatória de uma métrica/objetivo (Equação 2).

$$f_{max}(C) = \max\{\, f_j(C_j) |\ j\ \in J\} \tag{1}$$

$$\sum f_j\,(C) = \sum_{j=1}^{n} f_j\,(C_j) \tag{2}$$

Para ambas Equações, C_j representa o tempo de conclusão da tarefa j. Já f_j indica uma função custo associada à tarefa j, que pode ou não ser especificada. Como dois dos γ mais estudados, temos $\gamma = C_{max} = \max\{\, C_j \mid j\ \in J\}$ que quantifica o maior tempo de conclusão entre das tarefas em J, também nomeado como *makespan*, e $\gamma = \sum_{j=1}^{n} C_j$ (tempo total de percurso) que indica a somatória dos tempos

de conclusão da cada tarefa $j \in J$. Uma possível variação no tempo total de percurso é a inclusão de pesos w_j, portanto com $\gamma = \sum_{j=1}^{n} w_j C_j$. Para demais exemplos listados abaixo, todos podem seguir à objetivo de gargalo ou objetivos de soma.

- Atraso (L_j) – Indica o atraso associado à tarefa j. Define-se atraso como: $L_j = C_j - d_j$. Nota-se que L_j pode ser tanto positivo quanto negativo;
- Atraso positivo (T_j) – Diferentemente de L_j, T_j pode apenas apresentar valores positivos. Portanto, $T_j = \max(C_j - d_j, 0) = \max(L_j, 0)$, e representa o quanto tardio está uma tarefa relativamente à data de entrega (d_j) estabelecida;
- Penalidade por atraso (U_j) – É uma penalidade em uma unidade pela ocorrência de atraso. Isto é: $U_j = \begin{cases} 1, & C_j > d_j \\ 0, & caso\ contrário \end{cases}$;
- Precocidade (E_j) – Quantifica o quão antes uma tarefa j foi finalizada antes de sua respetiva d_j. Em outras palavras, significa o quanto está antecipada uma tarefa relativamente à data de entrega (d_j) estabelecida. Logo: $E_j = \max(d_j - C_j, 0)$.

Diferentemente das demais, E_j é dita como uma função *não-regular* no estudo de Escalonamento. A razão está por E_j ser *não-crescente* em C_j, ou seja, é uma função que não é favorecida por um menor C_j. Esse tipo de função é mais recente no estudo de Escalonamento, e comum para ambientes que buscam baixos desvio no plano de produção (*e.g.* ambientes que buscam JIT) (Brucker, 2007; Pinedo, 2016). Para concluir o campo γ, o número de atribuições possíveis a este campo depende da quantidade de critérios de otimização, por exemplo se duas funções estão sendo avaliadas $\gamma = \gamma_1 \gamma_2$.

2.2.2 Métodos de solução para problemas de Escalonamento

Sendo o Escalonamento um problema de análise combinatória, os métodos para soluções desenvolvidos ao longo dos anos são basicamente divergidos entre métodos exatos e métodos aproximados. Em princípio, existindo um conjunto finito de soluções viáveis, qualquer algoritmo que abordasse todas as caraterísticas do problema encontraria a melhor resposta para o mesmo, e, portanto, se teria uma resposta *exata* para a situação. No entanto, uma complicação é vista quando o número de soluções possíveis é muito alto. Deste último caso, pode-se preferir uma resposta que se utiliza de menos recurso e resulte numa aproximação da melhor solução, sendo assim o método de obtenção desta resposta chamado de *aproximado*. Em IO, os métodos aproximados devem fornecer uma qualidade sobre a

aproximação. Em outras palavras, cabe ao profissional garantir que a resposta aproximada não se distancie muito da resposta ótima (Festa, 2014).

Alguns problemas de Escalonamento podem ser reduzidos a conhecidos problemas de otimização combinatória, logo resolvidos a partir de metodologias desenvolvidas para os mesmos (Brucker, 2007). A modelagem matemática foi massivamente implementada ao decorrer dos tempos para estes problemas. Mais, dentro da modelagem matemática, e buscando a solução *exata* do problema, a *Programação Linear* foi uma das abordagens mais comuns na busca por soluções. Programação linear é um técnica bastante conhecida dentro da comunidade de IO, e eficaz principalmente em problemas de pequeno e médio porte (Brucker, 2007; Pinedo, 2016). A relevância dessa técnica como fonte de solução pode ser evidenciada em livros base da teoria do Escalonamento. Tanto Brucker (2007) como Pinedo (2016) destinam parte de capítulos com preocupação de explicar a programação linear. Uma simples síntese de Programação Linear pode ser descrita da seguinte forma:

$$Maximizar\ ou\ Minimizar \rightarrow Função\ objetivo$$
$$Sujeito\ a\ (s.a.) \rightarrow Conjunto\ de\ restrições$$

Formalizando, a Programação Linear baseia-se em um objetivo que se deseja otimizar, nomeado de função objetivo; e um conjunto de restrições que delimitam aquele problema. No tocante ao objetivo do problema, este pode divergir entre um objetivo de maximização ou minimização. Basicamente em um objetivo de minimização (maximização) busca-se, dentro das soluções possíveis, o menor (maior) valor de uma função linear. Já o conjunto de restrições (também linear), é formado por equações ou inequações que relacionam características do problema com a variáveis existentes (Hiller & Lieberman, 2014). Exemplificando, tem-se:

$$Min\ Z(x) = c_1x_1 + \cdots + c_nx_n \tag{3}$$

s.a.

$$a_{11}x_1 + \cdots + a_{1n}x_n \geq b_1 \tag{4}$$

$$\vdots$$

$$a_{m1}x_1 + \cdots + a_{mn}x_n \geq b_m \tag{4.n}$$

$$x_j \geq 0, \qquad \forall\, j = 1,..,n \tag{5}$$

Pela Equação 3 pode-se constatar a função objetivo ($Z(x)$), que nesse caso é de minimização. A função depende das *variáveis de decisão*, nesse caso representas por (x_1 ... x_n). Como elemento restante da

Equação 3, $C = (c_1 \dots c_n)$ são parâmetros do problema, valores conhecidos e constantes. As Inequações 4 e 5 representam as restrições do problema. As n-Inequações 4 representam a relação (linear) entre as variáveis de decisão e outros parâmetros do problema (neste caso a matriz $A_{m\,x\,n} = (a_{11} \dots a_{mn})$ e o vetor $B = (b_1 \dots b_m)$). A Inequação 5 apenas dita que as variáveis de decisão não podem ser negativas. Sintetizando, busca-se em um problema com parâmetros C, $A_{m\,x\,n}$ e B, minimizar $Z(x)$, de forma que os valores encontrados para $(x_1 \dots x_n)$, ou seja, a solução, respeitem as restrições do problema (Inequações 4 e 5).

Caso as variáveis de um problema modelado em programação linear *devam* ser todas inteiras (restrição/imposição), tem-se o chamado *Programação Linear Inteira* ou *Programação Inteira*. Se for imposto que as variáveis apenas podem receber valores binários, ou seja, 0 ou 1, tem-se o caso de *Programação Linear Binária*. Em caso de um modelo de programação linear com algumas das variáveis restritas a serem inteiras, tem-se *Programação Linear Mista* (Brucker, 2007). Destaca-se o método *Simplex,* algoritmo de *Branch and Bound* e método de *programação dinâmica*, como importantes algoritmos desenvolvidos para resolver os problemas de programação linear (Festa, 2014; Hiller & Lieberman, 2014).

O uso da programação linear ainda é uma das abordagens mais utilizadas para resolver problemas de otimização combinatória, e consequentemente, problemas de escalonamento. A causa dessa utilização pode ser dada pelo rigor, flexibilidade e extensiva capacidade de modelagem da programação linear (Floudas & Lin, 2005). Entretanto, por grande parte dos problemas de escalonamento serem *NP-Hard*, a abordagem por programação linear é aconselhada apenas em problemas de tamanho médio ou pequeno (Reza Hejazi & Saghafian, 2005). Todavia, para solucionar problemas de escalonamento, ainda são diversos os exemplos do uso da programação linear/modelagem matemática/métodos exatos, até os atuais dias. Por exemplo, no artigo de revisão de Daniel Alejandro Rossit et al. (2018) para *Non-permutation Flow shop* das 74 situações avaliadas 12 utilizam procedimentos exatos; em Allahverdi (2015) para ambientes *Flow shop* com *setups* e famílias de tarefas, das 38 situações descritas 8 utilizam de métodos exatos; e no artigo de revisão para *Flow shops* Flexíveis (FF_C), Lee e Loong (2019) mostram que 8% dos artigos avaliados foram resolvidos por métodos exatos.

Uma das formas de implantar a programação linear é modelá-la em uma linguagem computacional (*e.g.* AMPL) e atribuir este modelo a um solucionador (*solver*) (*e.g.* CPLEX). Diversos solucionadores estão disponíveis gratuitamente. Um exemplo de serviço gratuito que fornece diversas possibilidades de resolvedores, em diversas linguagens, para diferentes modelagens (programação linear é uma das

possibilidades) é o NEOS SERVER. Tal serviço é hospedado pela Universidade de Wisconsin, Estados Unidos, e vem sendo utilizado em Universidade de todo o mundo, como Universidade de Klagenfurt, Austria, e Universidade do Minho, Portugal (Neos Server, 2021).

A partir do facto que grande parte dos problemas de Escalonamento são *NP-Hard*, os métodos aproximados compõem uma importante parcela nos métodos para resolução destes problemas. Potts e Strusevich (2009) em seu trabalho de descrição dos principais marcos do estudo do Escalonamento separam a quarta década desde o início do estudo dessa área como a "década dos métodos aproximados". Os autores definem "Heurística" como sendo um algoritmo aproximado que não se preocupa com avaliação dos piores casos (*worst-case*) ou comportamento do algoritmo.

Potts e Strusevich (2009) também separam tipos de Heurística em: 1 - Heurísticas construtiva, que são aquelas que partem de uma solução vazia ao passo que no decorrer do método a solução é construída. Alguns exemplos já foram citados no trabalho, são casos das Heurísticas SPT e EDD; 2 - Heurísticas de busca local, que são originadas pelo relaxamento do problema, possibilitam partir de uma solução inicial já estabelecida, uma série de passo que busca a melhoria desta solução. Vale destacar o conceito de "vizinhança" que nada mais é do que movimento permitidos visando encontrar uma melhor solução; 3 – Metaheurísticas, que nada mais são que Heurísticas de busca local que ao longo dos anos demonstraram, para situações genéricas, uma performance digna deste termo. Pode-se destacar alguns famosos métodos como Algoritmos genéticos, *Simulated Annealing* e Busca Tabu (*Tabu Search*).

2.3 Ambiente *Flow shop*

O ambiente *Flow shop*, que conceitua uma configuração na qual todas as tarefas devem seguir uma mesma sequência de máquinas para respetivo processamento, vem sendo massivamente estudado ao longo dos anos. A importância desse ambiente tem caráter tanto teórico como prático. Do ponto de vista teórico, o trabalho de Johnson (1954) marca o Escalonamento como um estudo de análise combinatória e área independente dentro da IO (C. N. Potts & Strusevich, 2009). Especificamente, o trabalho de Johnson (1954) tratou de um ambiente *Flow shop*, logo, pode-se apontar esse artigo como o estudo inicial para esse tipo de ambiente (Campbell et al., 1970; T. C.Edwin Cheng et al., 2000; Rabadi et al., 2019; Srikar & Ghosh, 1986; Tseng & Stafford, 2001). Johnson (1954) demonstrou que existe uma solução ótima para o escalonamento de tarefas em um ambiente com 2 ou 3 máquinas, em que a sequência de tarefas não muda entre máquinas (*permutation*/ $\beta = prmu$). A metodologia de resolução proposta por Johnson recebeu o nome do autor (regra de Johnson) e serve, até os atuais dias, para

estudos, discussões ou solução inicial em novos métodos (*e.g.* Li e Lu (2020); Rabadi et al. (2019); Wu et al. (2020); Zou et al. (2020)). Vistos de outra perspetiva, os estudos de caráter teórico acerca do *Flow shop* iniciam na década de 50 e perpetuam-se até os presentes dias, demonstrando sua relevância. Do ponto de vista prático, ambientes *Flow shop* destacam-se pelos inúmeros fabricos que necessariamente devem seguir uma sequência fixa de máquinas: células de produção robótica (Dawande et al., 2007); pintura automotiva (Salmasi et al., 2010); indústria de semicondutores (Celano et al., 2010); fabrico de eletrónicos (Gelogullari & Logendran, 2010); dentre outros exemplos como linhas de montagem, indústria metalúrgica, indústria química e indústria alimentícia (González-Neira et al., 2017). Portanto, do ponto de vista prático, a relevância dos estudos sobre *Flow shop* dá-se pela diversidade de setores e atividades que essa configuração atinge. Assim, unindo a teoria à prática, a importância dos estudos sobre ambientes *Flow shop* é justificada pela abrangência de atuação (diversas atividades/setores/fabricos) que naturalmente contribuirá para o desenvolvimento de novos problemas, métodos e/ou discussões.

2.3.1 *Flow shop* com *setups*

O ambiente *Flow shop* analisado por Johnson (1954), também conhecido como *Flow shop regular*, segue premissas que distanciam o problema das situações do mundo real (Gupta, 1979). Além da consideração sobre *permutation*/ $\beta = prmu$, o *Flow shop* regular não inclui tempos/custos com *setups*. Allahverdi e Soroush (2008) enaltecem o papel crucial dos *setups* nas operações, e comentam que mesmo com fundamental importância alguns trabalhos ignoram esses tempos/custos ou os incluem nos tempos de processamento. Os autores diferenciam *custos* e *tempos* com *setups*, onde custos com *setups* designam custos para configurar qualquer recurso usado antes de um novo início de operação; já tempo de *setup* atribui-se ao tempo despendido para preparar o recurso (máquina). Nesta dissertação o termo *setup* infere tanto aos custos com *setups* como os tempos com *setups*, comentando diferenças quando necessário.

Setups vem recebendo significativa atenção na literatura. Allahverdi et al. (1999), Allahverdi et al. (2008) e Allahverdi (2015) constituem uma série de três extensivos artigos de revisão para *setups* baseados na classificação $\alpha|\beta|\gamma$ e com respetiva atualização devido o passar dos anos. Os três artigos reúnem, cronologicamente, métodos para resolução deste tipo de problema, além de distinguir os *setups* pela incidência (ou não) de lotes com sequência-dependente ou sequência-independente. Cheng et al. (2000) seguem a mesma lógica dos trabalhos citados, entretanto com foco específico para ambientes *Flow shop*. Os autores comentam fundamentalmente sobre classificação, representação e complexidade computacional existente para *Flow shop* com *setups*. Cheng et al. (2000) destacam também uma

diferença entre tempos com *setups* e tempos de remoção, que seriam aqueles decorrentes pós processamento, como inspeções ou transferências de produto, podendo variar entre tarefa-dependente ou tarefa-independente. Alguns autores (*e,g,* Ríos-Mercado e Bard (2003)) evidenciam os *setups* para o primeiro processamento no recurso (s_{i0k}). Esta evidência dá-se por s_{i0k} não ser originado pela troca de tarefas, e a menção também é válida pela desconsideração desses *setups* em outros estudos (*e.g.* Srikar e Ghosh (1986)). Seguindo a mesma lógica de s_{i0k}, *setups* pós a última operação também podem ser evidenciados (s_{inn+1}), como uma preparação para o dia seguinte de produção ou uma última ação (exemplo, limpeza) necessária no dia produtivo. Outro pormenor a destacar entre *setups* é feito em Stefansdottir et al. (2017), no qual os autores realizam uma especificação sobre a atividade de *limpeza*, segregando-a dos demais *setups*. Os autores declaram possibilidades de ganhos quando atividades de limpeza são consideradas à parte dos comuns *setups*. Mais, Stefansdottir et al. (2017) também fazem uma completa elaboração das características gerais dos *setups*, sendo estas: separabilidade, substituibilidade, ponto de referência, flexibilidade, tarefa-dependência e lote-dependência. A Figura 4, com base nos trabalhos anteriormente citados, apresenta um esquema dos tipos, nomenclaturas e abreviações que sintetizam os diferentes *setups* para ambientes *Flow shop.*

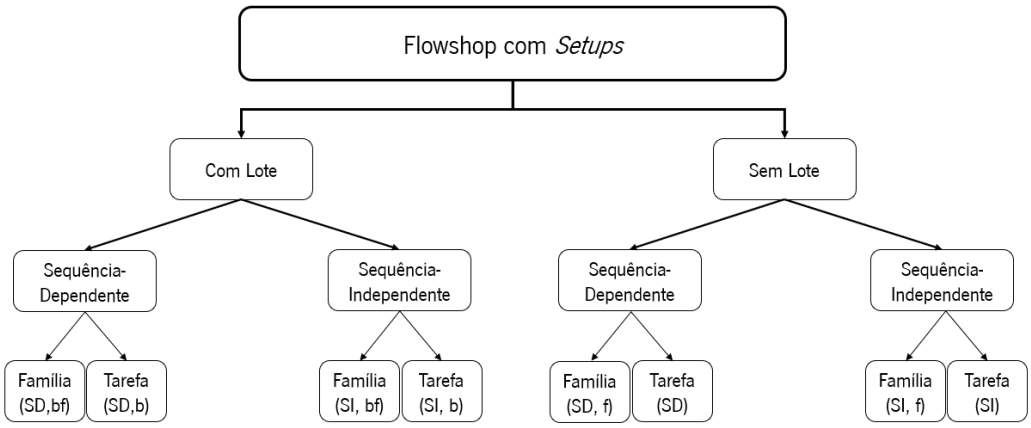


Figura 4 – Esquema *Flow shop* com *setups*

A Figura 4 inicia pela existência (ou não) do processamento por lotes (indicado pela letra "b") em ambientes *Flow shop*. O seguinte critério de segmentação dá-se pelo *setup* ser sequência–dependente (SD) ou sequência-independente (SI) entre tarefas ou famílias (indicado pela letra "f"). Portanto, e visto no último nível da Figura 4, para um *setup* em contexto de processamento por *lotes*, que *depende* da sequência em que as *famílias* de produtos são processadas, usou-se "SD,bf" como abreviação. Se famílias não existem no sistema, tem-se a abreviação "SD,b"; se os *setups* duma produção em lotes não dependem da sequência de processamento das famílias, tem-se "SI, bf". Em

igual caso, porém, apenas com tarefas, deu-se “SI, b”. Para a situação de não existência de lotes de produção e que o *setup* depende da sequência de processamento das famílias, tem-se “SD, f”, sem famílias “SD”. Os últimos dois casos destinam, respetivamente, os casos vistos em uma produção sem lotes onde os *setups* não dependem da sequência de processamento das famílias “SI, f” ou das tarefas “SI”. Uma importante observação é que a Figura 4 apresenta a possibilidade de haver famílias de produtos, mas o respetivo processamento destas famílias pode não ser necessariamente por lotes. Vale a lembrança de que a capacidade de realizar o processamento em lotes depende apenas do recurso (máquina), portanto, a existência de famílias de produtos independe se a máquina é processadora de lotes ou não.

Os *setups* podem também divergir entre *antecipatórios* e *não-antecipatórios*. Utilizando a característica de separabilidade descrita em Stefansdottir et al. (2017), *setups antecipatórios* e *não-antecipatórios* dependem da possibilidade de início do *setup* acontecer (máquina sucessora) antes do término da tarefa (portanto, antecipatório/separável) na máquina predecessora (Figura 5a); ou o *setup* ocorrer apenas (máquina sucessora) ao término da tarefa na máquina predecessora (portanto, não-antecipatório/inseparável) (Figura 5b).

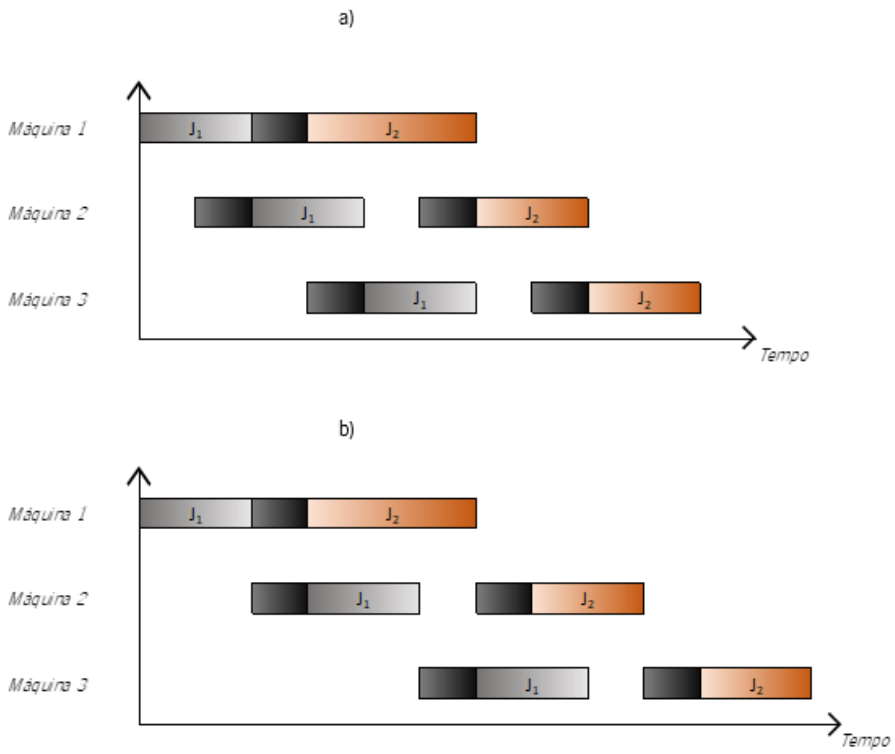


Figura 5 – Gráfico de Gantt para *setups* antecipatórios (5a) e não-antecipatórios (5b)

A Figura 5 é estruturada por 3 máquinas e 2 tarefas (J_1 e J_2) com respetivos *setups* (em preto) e tempos de processamento (cinza - J_1 e laranja - J_2). Pela Figura 5a pode constatar um diagrama de Gantt referenciando *setups* antecipatórios. Nota-se que é possível *antecipar* a ocorrência do *setup* em máquinas sucessoras, por exemplo, antes do término de J_1 na máquina 1 o *setup* de J_1 já é realizado na máquina 2. *Setups* antecipatórios são o que Stefansdottir et al. (2017) conceitualizam como a possibilidade de trabalhar *offline*. Já para o segundo diagrama de Gantt (Figura 5b), nota-se que o *setup* apenas ocorre quando a respetiva tarefa é finalizada na máquina predecessora. Por exemplo, o *setup* referente a J_1 na máquina 2 só ocorre quando J_1 é finalizado na máquina 1. Evidentemente, para dois ambientes *Flow shop* idênticos (quantidade de máquinas, tarefas e iguais tempos de processamento) que divergem entre *setups* antecipatórios e não-antecipatórios, o *makespan* do ambiente *Flow shop* para *setups* antecipatórios será menor.

Concluindo este subtópico, é fundamental comentar, mesmo que brevemente, acerca da complexidade computacional encontrada em problemas *Flow shop* com *setups*. Para um ambiente *Flow shop* regular (sem *setups*), com minimização do *makespan* e arbitrário número de máquinas ($F_m \mid\mid C_{max}$), sabe-se que esse problema é *NP-hard* em forte senso (Lawler et al., 1993). Para um também caso de arbitrariedade de máquinas, porém com a inclusão de *setup*, Garey e Johnson (1979) mostram que esse problema também é *NP-hard* em forte senso. Entretanto, Cheng et al. (2000) observam que dentro dos problemas *Flow shop* com *setups* existem situações polinomialmente resolvíveis (resolução do problema em tempo polinomial). Os autores destacam que problemas *Flow shop* com *setups* sequência-dependente e processamento em lotes (SD,bf e SD,b) são os mais complexos quando comparado entre todas situações da Figura 4. À razão de superior complexidade computacional, nestas situações tem-se para além de um problema de escalonamento *Flow shop* um problema de formação de lotes.

2.3.2 *Flow shop* com processamento de lotes

Implicitamente comentado no subtópico anterior, uma das motivações para realizar produções com lotes é a existência de *setups*. Mais do que existir *setups*, Chris N. Potts e Kovalyov (2000) generalizam que a produção por lotes ocorre devido qualquer oportunidade de ganho em efetividade. Normalmente, a preferência por produção em lotes será em virtude de um processamento mais rápido ou mais económico que o processamento individual das tarefas (Chris N. Potts & Kovalyov, 2000). Com menor generalização dos ganhos, Van Der Zee (2013) cita que a partir da produção de lotes atinge-se melhoramentos em métricas como tempo(s) de ciclo, redução de inventário para WIP, capacidade de resposta ao cliente e taxa de produtividade.

Idêntico à história do Escalonamento, o estudo da temática de processamento por lotes iniciou-se com uma máquina e posteriormente estendeu-se em situações mais complexas. Entretanto, por processamento em lotes ser uma capacidade intrínseca à máquina processadora de tarefas, os principais conceitos que regem as máquinas processadoras de lotes não dependem do *layout* em que as mesmas estão configuradas. Mais, deve-se ter em mente que quando se processa em lotes o sistema engloba mais de um problema, logo, em ambientes com mais de uma máquina existirão mais "subproblemas" e consequentemente maior complexidade. Assim, para sistemas com máquinas processadoras de lotes, tem-se: 1 - uma decisão do loteamento, que consiste no problema de formar os lotes (similar ao problema da mochila) e posteriormente, caso necessário, sequenciar, dentro de cada lote, as respetivas tarefas; 2 - uma decisão de escalonar os lotes (similar ao problema geral de afetação) na máquina de referência; e 3 - em caso da sequência poder ser alterada entre máquinas (*non-permutation*), realizar os pontos 1 e 2 em cada máquina. É evidente que o terceiro ponto só se refere aos ambientes com mais de um sector (*Flow shop*, *Open shop*, *Job shop* ou variantes) (Matin et al., 2017; Neufeld et al., 2016; Shen & Buscher, 2012; Shen & Gupta, 2018).

Existe uma variante à forma que o processamento do lote é executado. Imaginemos uma situação em que a formação dos lotes já está definida. Se o processamento de quaisquer desses lotes for feito de maneira que o recurso (máquina, operador, etc) incida individualmente sobre cada tarefa, temos a definição de *serial batch* ($\beta = s - batch$). Em outras palavras, o recurso processa as tarefas em série, portanto, o tempo de processamento do lote é o somatório de cada tempo de processamento das tarefas pertencentes ao lote (Figura 6a). Se o processamento de quaisquer dos lotes programados for feito de maneira que o recurso incide simultaneamente em todas as tarefas, tem-se a denominação *parallel batch* ($\beta = p - batch$). Em outras palavras, o recurso processa paralelamente as tarefas contidas no lote, assim, o tempo de processamento do lote é o maior tempo de processamento entre as tarefas pertencentes ao lote (Figura 6b) (Baptiste, 2000; Brucker, 2007; Brucker et al., 1998; Matin et al., 2017; Pinedo, 2016; C. N. Potts & Wassenhove, 1992; Chris N. Potts & Kovalyov, 2000).

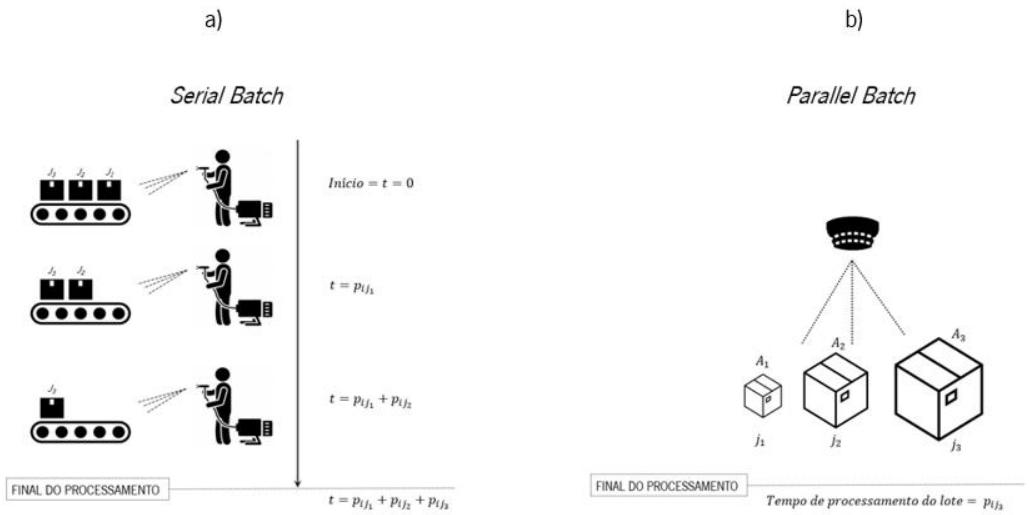


Figura 6 – *Serial Batch* (6a) *vs Parallel Batch* (6b)

A Figura 6 ilustra o processamento em lotes de forma serial e forma paralela. Para ambas, tem-se 3 tarefas (J_1, J_2 e J_3) e um recurso i. No processamento de lotes em série (Figura 6a) o recurso i processa, em uma sequência já estabelecida ($J_1 \rightarrow J_2 \rightarrow J_3$), uma tarefa de cada vez. Portanto, ao final do processamento em série o tempo de processamento do lote é a soma de cada tempo de processamento ($p_{ij_1} + p_{ij_2} + p_{ij_3}$). Já para o processamento de lotes feito paralelamente, o recurso i é capaz de processar simultaneamente todas as tarefas (portanto, não há sequência entre tarefas). Numa situação hipotética em que os tempos de processamento são diretamente proporcionais às áreas das tarefas/produtos ($A_1 < A_2 < A_3$), o tempo para processamento do lote, realizado de maneira paralela (Figura 6b), é o maior tempo de processamento entre as tarefas do lote. Nesse caso ilustrado, o maior tempo corresponde a tarefa de maior área (A_3), assim, o tempo de processamento do lote é p_{ij_3}.

Tanto *serial batch* quanto *parellel batch* são definidos numa suposição que o recurso tem capacidade física de processar todas as tarefas dos respetivos lotes, portanto, sendo um processamento *contínuo* do lote. Entretanto, existem situações no qual os lotes são formados e não existe capacidade física, quer seja da máquina ou da estação de trabalho, para processamento integral de algum dos lotes. Wang et al.(2020) inferem que essa seria uma terceira variante à forma de que o processamento ocorre. A denominação dessa situação foi realizada por Tang e Zhao (2008), precursores no que ficou conhecido como *semi-continuous batch* ($\beta = c - batch$). Neste caso de incapacidade física, uma quantidade máxima quantidade de tarefas, limitadas pela capacidade física, são processadas pelo maior tempo de processamento das tarefas do lote, e, ao término de uma das tarefas uma nova tarefa (pertencente ao mesmo lote) pode ser direcionada para seu processamento (Tang & Zhao, 2008). É evidente que a situação $\beta = c - batch$ só é possível se a operação suportar a dinâmica descrita (saída e entrada de

produtos, como exemplo uma esteira cíclica). Para impossibilidade de $\beta = c - batch$, deve-se dividir o lote incapaz de processamento em quantidades suportadas pela capacidade física.

Outra importante conceção para processamento de lotes está no modo que este é transferido. Quer seja para um cliente ou uma máquina seguinte (*Flow shop*, *Open shop*, *Job shop* ou variantes), o momento que o lote se torna disponível para o estágio seguinte diverge entre *lote disponível* ou *tarefa disponível*. Lote disponível quer dizer que o lote é transferido integralmente, portanto, a disponibilidade para seguinte processamento é a mesma entre todas as tarefas do lote. É exemplo dessa situação quando a transferência entre estágios é realizada por transportes, e torna-se inviável a transferência individual das tarefas. Tarefa disponível conceitua que a passagem do lote pode ser feita em partes, ou seja, quando uma tarefa, pertencente a um lote, é processada a mesma pode ser transferida (está disponível) para o estágio seguinte (Allahverdi et al., 2008; Chris N. Potts & Kovalyov, 2000; Shen & Gupta, 2018). Mais, mesmo com menor relevância prática, a situação de tarefa disponível é dominante entre a literatura existente sobre processamento em lotes (Shen & Gupta, 2018).

Como já foi dito, *setups* são um dos motivos para executar o processamento das tarefas em lotes. Um maior ou menor *setup* ocorrido entre trocas de processamento muita das vezes tem relação com a diferença entre a tarefa/lote finalizado e a tarefa/lote seguinte a processar. *Família* de produtos é o termo utilizado em certos ambientes/estudos para uma segregação, em virtude do grau de similaridade tecnologia ou da característica física, entre as tarefas/produtos processados no sistema. Normalmente, o *setup* existente entre a troca de processamento de tarefas da mesma família é baixo ou negligenciado, enquanto os *setups* ocorridos na troca de processamento de tarefas com diferentes famílias são sobretudo altos (Braglia et al., 2020; Liaee & Emmons, 1997; Chris N. Potts & Kovalyov, 2000; Webster & Baker, 1995). É fundamental entender que não necessariamente o número de lotes será o número de famílias de tarefas existentes. Caso isto aconteça, tem-se o conceito de *tecnologia de grupo*, ou seja, para um conjunto de famílias de tarefas $F = \{1, \dots, f\}$ com f famílias, processar-se-ão f lotes. Portanto, em processamento por tecnologia de grupo não existe a ocorrência de sub-lotes (*e.g. lot streamming*) numa família, isto é, tarefas de iguais famílias e diferentes lotes (Baker & Trietsch, 2009; Cheng et al., 2000; Neufeld et al., 2016; Shen & Gupta, 2018; Van Der Zee, 2013). Mais, formar-se lotes depende das vantagens (custos, tempos, etc) em processar tarefas juntas. No entanto, por tratar de uma série de combinações a analisar, pode existir em situações que os lotes são compostos por apenas uma tarefa, ou até por tarefas de diferentes famílias. Em alguns casos reais é impossível processar dois produtos de diferentes famílias em um mesmo lote, para estes casos denomina-se o problema com o termo *famílias de tarefa incompatíveis* (X. L. Li et al., 2019; Mönch & Roob, 2018; Uzsoy, 1995).

Formalizado a composição de um lote (parágrafo anterior), esta pode, ou não, ser a mesma entre todos os estágios a processar. Basicamente, problemas com lotes *inconsistentes* são aqueles que a composição do lote diverge entre duas ou mais máquinas existentes no sistema. Este fenômeno pode ocorrer devido ao fator económico de ter, ou não, um produto em um lote em uma certa máquina (*e.g.* Isenberg e Scholz-Reiter (2013)); pela configuração dos *setups* (inicial, entre processamentos e final) entre máquinas (*e.g.* Shen e Gupta (2018)); pela capacidade física da máquina (Matin et al., 2017); ou, por existir diferentes famílias entre máquinas (*e.g.* Isenberg e Scholz-Reiter (2013)). A Figura 7 ilustra uma comparação do que é um escalonamento em ambiente *Flow shop* com lotes, processamento em série e lote disponível, divergindo, respetivamente, por ser em tecnologia de grupo, não ser em tecnologia de grupo (sublotes) e lotes inconsistentes.

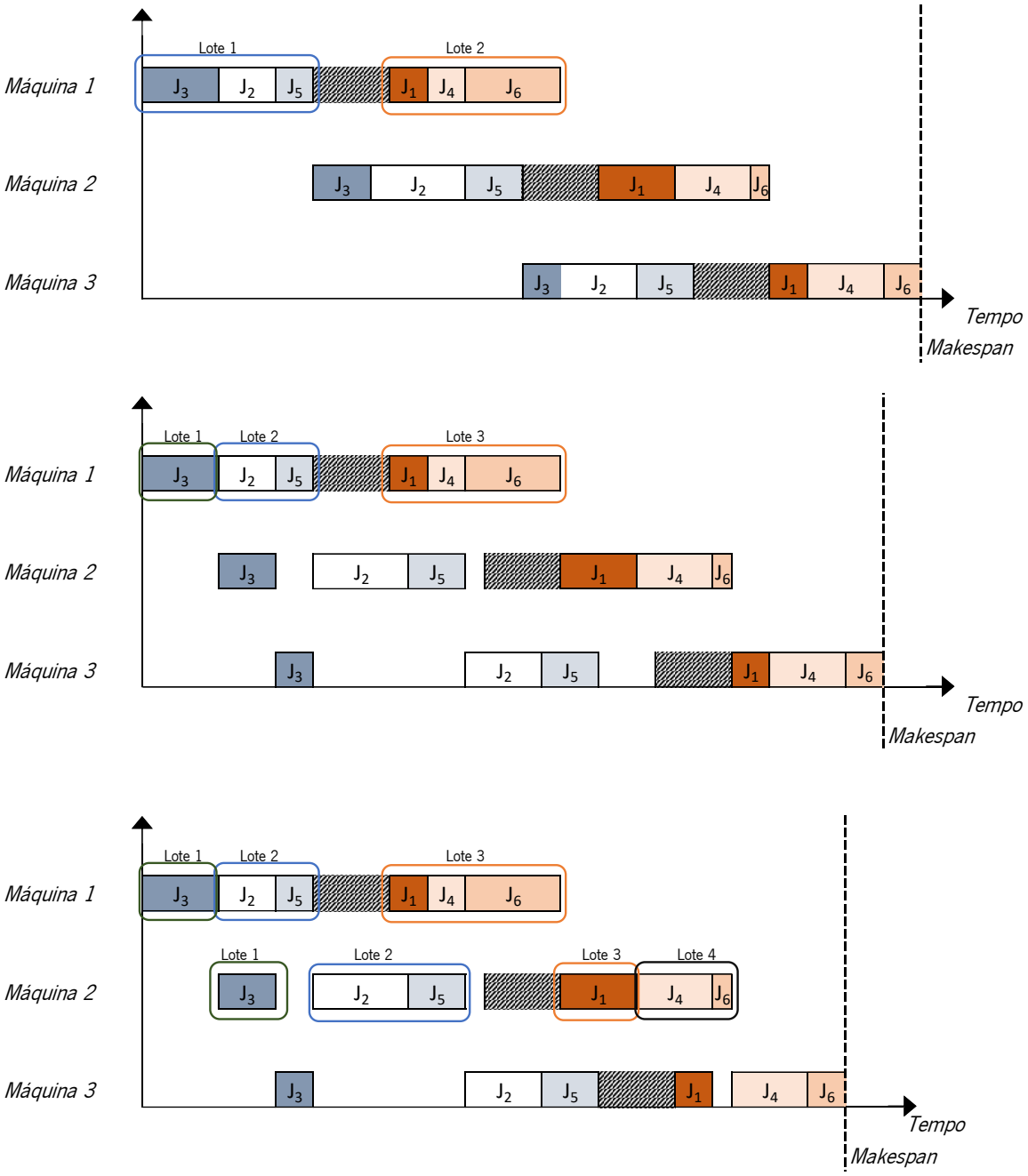


Figura 7 – Gráfico de Gantt para escalonamento com tecnologia de grupo; sem tecnologia de grupo; e com lotes inconsistentes

A Figura 7 idealiza uma situação com 3 máquinas, 6 tarefas e respetivos tempos de processamento e *setups* (entre famílias). O conjunto de família é representado por $F = \{f_1, f_2\}$ (2 famílias), com as seguintes tarefas pertencentes as famílias: $f_1 = \{J_2, J_3, J_5\}$ e $f_2 = \{J_1, J_4, J_6\}$. Partindo do topo até baixo, tem-se as respetivas situações: escalonamento com lotes e tecnologia de grupo; escalonamento com lotes sem tecnologia de grupo; e escalonamento com lotes inconsistentes, todas para um ambiente *Flow shop*. No primeiro gráfico note-se que o número de lotes é o mesmo da quantidade de famílias no sistema (2 lotes; 2 famílias), logo a família é inteiramente processada dentro dos respetivos lotes. A

constituição dos lotes para a primeira situação é a mesma para todas 3 máquinas, portanto tem-se o exemplo tem um lote consistente. Para a segunda situação, *Flow shop* com processamento em lotes sem tecnologia de grupo, vê-se a existência de sublotes, como exemplo $f_1 = \{J_2, J_3, J_5\}$ que se subdivide em 2 lotes (Lote 1 e Lote 2). Em termos práticos, isso pode ser explicado por não existir nenhum benefício em processar todos os itens de f_1 juntos. Por fim, no último gráfico de Gantt é possível constatar uma situação com lotes inconsistentes, evidenciado na troca entre primeira e segunda máquina. Na máquina 1 J_1, J_4 e J_6 são processados juntos no "Lote 3", já na máquina 2 vê-se uma diferente configuração onde J_1 não se processa no mesmo lote que J_4 e J_6. Semelhante com a explicação dada para a segunda situação, a inconsistência na terceira situação pode ser explicada pois na máquina 2 não existe benefício em processar J_1 junto a J_4 e J_6. Outra forma de evidenciar uma inconsistência nos lotes é quantificar o número de lotes *com tarefas* em cada máquina. Se o número não for o mesmo em todas as máquinas, como por exemplo na Figura 7 (último gráfico) em que na máquina 1 tem-se 3 lotes e na máquina 2 existem 4 lotes, automaticamente tem-se uma inconsistência de lotes. A enfase em lotes *com tarefa* foi dada pois é possível afirmar que o número de lotes é invariável nas máquinas, basta, tomando o mesmo exemplo, atribuir um 4º lote à máquina 1, porém, esse lote é fictício, já que nenhuma tarefa pertence ao "Lote 4".

2.3.3 *Non-permutation Flow shop*

Desde Johnson (1954) que ambientes *Flow shop* com $\beta = prmu$ são massivamente estudados na literatura. Seguido do trabalho de Johnson, Wagner (1959) desenvolve um modelo de Programação Linear Inteira (PLI) para *Permutation Flow shop*. Tal formulação foi usada como base para diversos trabalhos nos anos que se seguiram, como exemplo modelos de PLI para *Permutation Flow shop* e *setup* (Jr e Tseng (1990); Ríos-Mercado e Bard (2003); Srikar e Ghosh (1986); e Tseng e Stafford (2001)). O *Permutation Flow shop* também foi objeto para desenvolvimento outros importantes métodos de solução como a Heurística NEH (Nawaz et al. (1983)) e heurísticas que tomam a heurística NEH como base (*e.g.* W. Liu et al. (2017)).

Permutation Flow shop trata-se de uma especificação dentro dos ambientes *Flow shop*. Essa restrição de manter a mesma sequência de processamento das tarefas entre máquina pode não beneficiar situações reais (Meng & Xu, 2020; Daniel A. Rossit et al., 2021). Por outro lado, em contexto atual de crescente customização, desenvolvimento tecnológico e montagem tardia, a troca da sequência de processamento entre máquinas (*Non-permutation Flow shop*) pode ser benéfico ao sistema (Daniel Alejandro Rossit et al., 2019a).

A razão de preferência por *Permutation Flow shop* pode ser averiguada pelo número de possíveis escalonamentos. Enquanto existem $(n!)^m$ escalonamentos para uma situação de *Non-permutation Flow shop*, existem $n!$ escalonamentos para *Permutation Flow shop* (Rossi & Lanzetta, 2014). Outra justificativa pode decorrer da prova que existe um escalonamento ótimo para *Flow shops* regulares com 2 ou 3 máquinas (Johnson, 1954; Pinedo, 2016). Mais, não tratar o ambiente *Flow shop* de maneira geral (*Non-permutation Flow shop*) pode ocorrer também pela impossibilidade de troca da sequência entre estágio (J. Schaller, 2012). Uma troca da sequência só se é possível caso exista um mecanismo de armazenamento entre estágio (pulmões), onde possa-se "segurar" os itens e permitir que outras tarefas "passem" à frente. Se $\beta \rightarrow b = 0$, implicitamente tem-se um ambiente *Permutation Flow shop*. Outras restrições que tornam o ambiente *Permutation Flow shop* são $\beta = block\ e\ \beta = no - wait$ (Daniel Alejandro Rossit et al., 2018). Em linhas gerais, apesar de existir situações em que o *Permutation Flow shop* apresente resposta ótima, a forma geral de um ambiente *Flow shop* (*Non-permutation Flow shop*) domina o *Permutation Flow shop* para mais de 3 máquinas no ambiente (Figura 8) (Chris N. Potts et al., 1991; Pugazhendhi et al., 2003; D. Rossit et al., 2016). Em outras palavras, *Permutation Flow shop* é uma subclasse do *Non-permutation Flow shop* (Chris N. Potts et al., 1991).

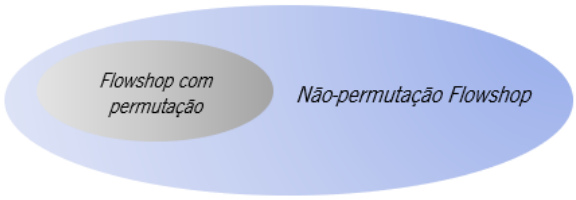


Figura 8 – Espaço solução *Permutation Flow shop* e *Non-permutation Flow shop* (traduzido de D. A. Rossit et al. (2018))

Dos principais aspetos que podem prejudicar a análise por negligenciar ambientes *non-permutation Flow shop*, pode-se citar que: 1 – para ambientes com 4 ou mais máquinas o *Permutation Flow shop* pode não ser ótimo (Koulamas, 1998); 2 – para objetivos diferentes à minimização do *makespan*, principalmente no tocante de datas de vencimento (d_j), Lin e Ying (2009) e Lin et al. (2009) mostram que a diferença de ganho em eficiência é em média 10% superior a favor de ambientes *Non-permutation Flow shop*, podendo atingir 30%; 3 – a natureza da operação contrapor-se à *Permutation Flow shop*, como casos de um *Flow shop* com diferentes famílias entre máquina (*e.g.* Isenberg e Scholz-Reiter (2013)) ou *Flow shop* com *falta* de operações (*missing operations*) (Henneberg & Neufeld, 2016; Rossit et al., 2021).

Com o desenvolvimento tecnológico e ganhos que acompanham a evolução do mercado, os problemas de *Non-permutation Flow shop* vem despertando interesse de estudo (Rossit et al., 2018). Rossit et al. (2018) em seu artigo de revisão apenas à *Non-permutation Flow shop* relatam que 65% dos artigos selecionados para compor seu trabalho foram desenvolvidos depois de 2006, portanto, demonstrando-se esta uma temática recente. Outra forma de fortalecer "*Non-permutation Flow shop*" como uma temática recente é vista ao filtrar por palavras-chaves, título e resumo na base de dados *Scopus* a partir das palavras "*Non-permutation flow**" ou "*Non permutation flow**". Dos 52 resultados obtidos pela pesquisa, apenas 2 são anteriores ao ano de 2005 (Scopus, 2021).

Por fim, *Non-permutation Flow shop* mescla-se com outros assuntos já aqui comentados, tais como processamento por lotes e existência de *setups*. No mesmo sentido que a sequência de processamento das tarefas ser ou não alterada nos estágios, a sequência de processamento dos *lotes* pode ou não ser alterada. Ng e Kovalyov (2007) apresentam um relevante estudo na temática de *Permutation Flow shop* vs *Non-permutation Flow shop* com o processamento em série dos lotes e *setups* dependentes. Os autores provam que para o problema $F|\, s-batch,\ s_{ijk}|\, C_{max}$, independente dos *setups* serem antecipatórios ou não, e independente dos lotes serem consistentes ou não, existe um escalonamento ótimo *com* o conceito de *permutation*/ $\beta = prmu$ para 1, 2 ou 3 máquinas.

3. Estudo de Caso: ItzWood – Soluções Tecnológicas

Este capítulo descreve as características da empresa que origina o problema de estudo. O setor de atuação, os estágios produtivos, o fluxo de produção e os problemas vistos são tratados na primeira parte do capítulo. Na segunda parte é concatenado as classificações abordadas na revisão da literatura com a situações real, e assim, formula-se o sistema e seus componentes que embasam o restante da dissertação.

3.1 ItzWood – Soluções Tecnológicas, LDA

A empresa contida na presente dissertação tem como nome "ItzWood – Soluções Tecnológicas, LDA". A empresa pratica a manufatura de artigos sustentados pela matéria-prima madeira, presando pela inovação tecnológica e alta qualidade para com qualquer móvel/artigo que surja de uns dos materiais mais nobres, belos e sustentáveis – a madeira (Itzwood, 2021). A companhia está localizada na região norte de Portugal, mais especificamente na cidade de Paços de Ferreira. A cidade está inserida num polo reconhecido pelas atividades têxteis, couro e madeira. Desta última afirmação, a Figura 9 segrega respetivamente, a indústria da madeira e cortiça (com exclusão do mobiliário) (Figura 9a) e de mobiliário e colchões (Figura 9b) em Portugal por "NUT 2" (divisão do território português em sete regiões). Sendo evidente o domínio territorial da região Norte nessa indústria.

Figura 9 – Segregação do setor de atuação da empresa por NUT 2; Figura 3a - Indústria da madeira e cortiça; Figura 3b - Indústria mobiliária e colchões (Direção-Geral das Atividades Económicas, 2021)

3.1.1 Estágios e Fluxos Produtivos

A ItzWood é composta por cinco estágios produtivos, sendo estes: marcenaria, pintura, acabamento, expedição e estofos. Em cada estágio pode também existir subdivisões. Isto é o caso dos estágios

marcenaria e pintura, com as subdivisões: pré-corte, CNC, trabalho manual e aros (para marcenaria); primário, lixagem e acabamento (para pintura). A Tabela 3 sintetiza a numerização (códigos) utilizada nesta dissertação para atribuir cada estágio e respetiva subdivisão.

Tabela 3– Numeração dos estágios produtivos

Código	Estágio	Subdivisão
1	Marcenaria	
11		*Pré-corte*
12		*CNC*
13		*Trabalho manual*
14		*Aros*
2	Pintura	
21		*Primária*
22		*Lixagem*
23		*Acabamento*
3	Acabamento	-
4	Expedição	-
5	Estofos	-

De acordo com a Tabela 3, pode-se definir uma representação do fluxo produtivo como uma união entre os códigos, por exemplo: 11 - 12 - 13 - 14 - 2 - 3 - 4. Algumas observações devem ser comentadas: atualmente o estágio da marcenaria (1) é apenas indicativo, portanto, se o fluxo produtivo iniciar pela marcenaria (imensa maioria dos fluxos existentes) receberá um número entre 11-14; atualmente o estágio da pintura ainda não é operacionalmente segregado, portanto, apenas existem fluxos que contém o código 2 e nenhum com valores entre 21-23. A implementação da correção dessa segunda observação (utilizar os códigos entre 21-23) é um dos procedimentos futuros prioritários na empresa.

Para proporcionar uma ideia espacial do fluxo produtivo, a Figura 10 localiza cada estágio e subdivisão (pelo código) e simula um fluxo produtivo em que todos os estágios são acionados. Tal esquema estruturou-se a partir do layout da empresa.

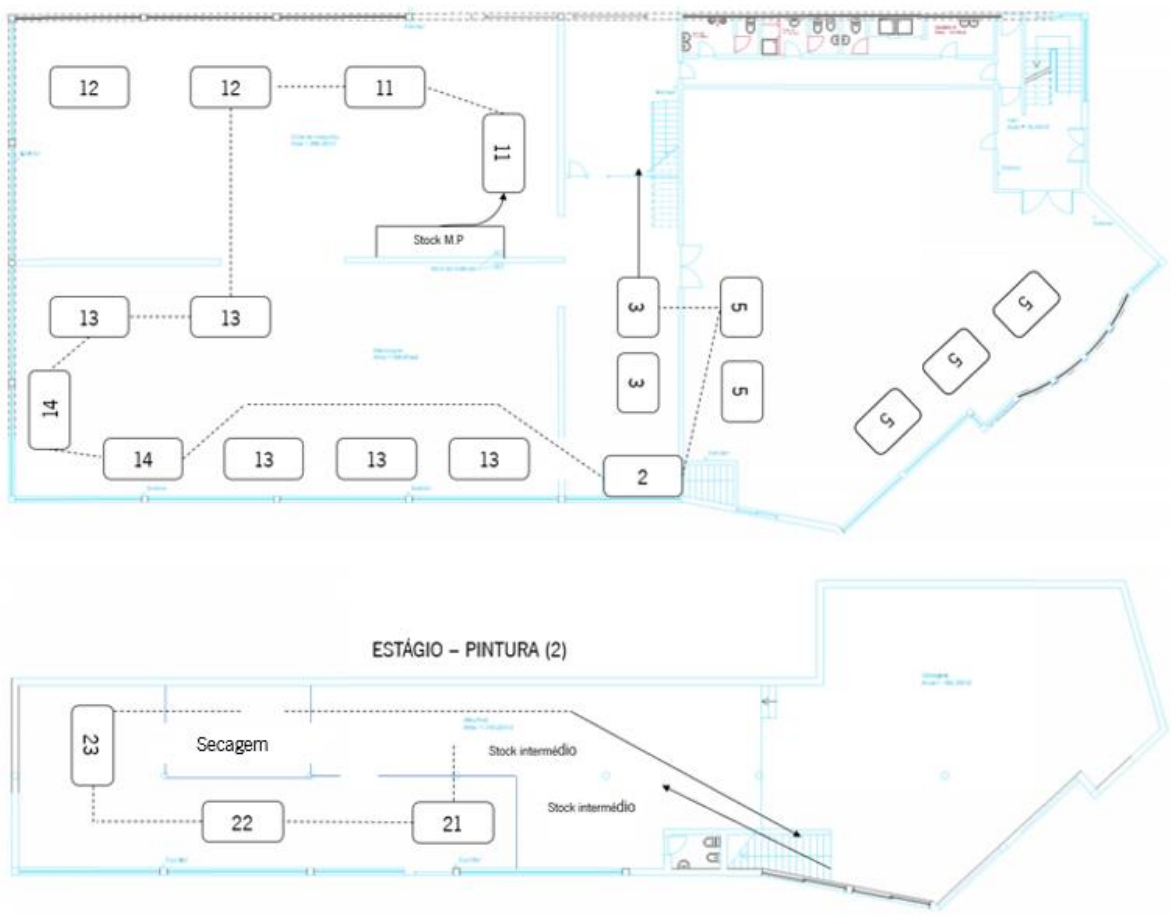


Figura 10 – Exemplo de fluxo produtivo

É percebível a partir da Figura 10 a existência de dois níveis na empresa. O estágio da pintura é realizado no nível subsolo (cave). Os demais processos são feitos ao nível do solo. A Figura 10 exemplifica um fluxo que apenas algumas máquinas de cada estágio foram necessárias. Por isso não se vê linhas penetrando em certas máquinas. Outra observação se dá caso mais de um kanban estivesse a ser processado. Para este exemplo, o funcionamento ocorreria em paralelo, logo, mais de um fluxo deveria ser indicado. A Figura 10 apenas os stocks intermédios da pintura foram representados. Todavia, entre toda troca de estágio existe espaços determinados para stocks intermédios.

Quanto às ordens de produção, o subtópico seguinte aborda o esclarecimento do funcionamento quando é recebida uma ordem de produção.

3.1.2 Ordem de Produção - Kanban

Ao ser recebido um pedido para fabricação (por parte do cliente), as seguintes ações são tomadas antes de iniciar o processo produtivo: elaboração do projeto e orçamentação e o envio para validação do cliente. Caso validado, a ordem de produção será gerada. Nos casos em que o produto já tenha sido pedido e

produzido para o mesmo cliente, a validação do projeto não é necessária, decorrendo a imediata possibilidade de geração da ordem de produção e início da produção. Posto a ordem em produção, pedidos customizados carregam 25 dias úteis de prazo e pedidos padrões 15 dias úteis.

A empresa utiliza cartões (kanbans - ordens de produção) como uma forma de guiar a produção. Para além do controlo visual, as informações e pormenores do produto estão contidas no kanban (por meio do respetivo código de barras) para facilitar o operador durante a operação. Cada kanban também informa a quantidade que deve ser produzida, assim como, a quantidade a se produzir naquele estágio. Por exemplo, para se produzir X artigos pode ser necessário no pré-corte (11) Y cortes diferentes. Esses Y cortes podem seguir para o trabalho manual (13) e serem unidos (colados) dando forma a X peças. Todas nestas informações são acessíveis via kanban.

A utilização dos kanbans é entendida como uma mais-valia pela empresa. Conforme foi reportado no Capítulo 2, o sistema kanban promove diversas vantagens. No âmbito da ItzWood esses ganhos são bem observados. Dentre os principais, destacam-se:

- Diminuição dos stocks intermédios: como cada kanban contém a quantidade de material necessária, o conhecimento da quantidade a produzir é de fácil acesso. Antes do processamento o operador verifica (no sistema interno) se aquela quantidade está presente em stock, utilizando só a quantidade demandada. Podendo também, em caso de falta de produto, requisitar o(s) item(s);
- Antecipação da falta de material: o estágio seguinte de produção também pode facilmente obter as informações do kanban e requisitar itens;
- Rápido rastreio da ordem de produção;
- Fácil controlo do sistema produtivo.

A passagem entre estágios dá-se por colocar o kanban num "sequenciador" presente no estágio seguinte. Entende-se por sequenciador uma caixa como a representada na Figura 11.

Figura 11 - Sequenciador

É possível visualizar três locais onde o kanban pode ser depositado. O primeiro local (de cima para baixo), pintado de vermelho, destina-se para kanbans com "problema", por exemplo falta de material. O segundo é destinado para kanbans "via verde" aonde são depositados aqueles com máxima prioridade. Em certos momentos, o gestor pode resgatar uma ordem da pilha de kanbans a produzir (terceiro deposito) e afetá-lo ao posto "via verde" – prioritário. O último local, como já mencionado, é destinado à sequência natural das ordens de produção, sendo o próximo item a adentrar no estágio o localizado mais abaixo. Na Figura 11 pode-se observar uma pilha de kanbans no último nível, logo, o kanban mais abaixo desta pilha é o próximo a entrar em estágio de produção.

Para mudanças de estágio, o operador do estágio atual deposita o kanban no terceiro posto (em preto) do sequenciador do estágio seguinte. Um exemplo com mesma estética e funcionalidade são as caixas de correio descritas em Hirano (2009).

Entre os códigos 11 – 14 (pré-corte; CNC; trabalho manual; aros) o fluxo peça por peça (*one-piece-flow*) é razoavelmente conservado. Os tempos de processamento destes estágios seguem valores similares e estáveis, assim como baixos e consistentes tempos com *setups* para as trocas de kanbans. Portanto, naturalmente o primeiro item posto no estágio 11 será (na maioria das vezes) o primeiro item a sair do estágio 14 (obedecendo o *First-In-First-Out*, FIFO). A quebra do FIFO pode ocorrer para kanbans com diferentes fluxos. Por exemplo, considere que uma encomenda tem sua ordem de produção indicada por um kanban denominado como "A" e outra encomenda (completamente diferente da vista no kanban A) tem sua ordem de produção indicada pelo kanban denominado com "B". O kanban A tem um fluxo específico para respetivo fabrico descrito pelos estágios 11 - 12 - 13 - 2. Já o kanban B tem um fluxo para respetivo fabrico descrito pelos estágios 11 - 12 - 13 - 14 - 2. Portanto, mesmo em caso da produção do kanban A ser iniciada antes da produção do kanban B, o kanban B pode chegar antes no estágio da pintura (2) devido seu fluxo necessitar de um estágio a menos. Ainda neste exemplo, enquanto o kanban A pode estar em processamento no estágio "aros" o kanban B pode ter terminado seu

processamento no estágio “trabalho manual” e seguir fluxo para o estágio da pintura, e, portanto, a regra FIFO sendo quebrada. Este é um exemplo descrito no Capítulo 2 subsecção 2.3.3 para *Flow shop* com falta de operações.

3.1.3 Estágio Produtivo - Pintura

Contrapondo a filosofia JIT para a abordagem por kanbans (Sistema Kanban), a produção por lotes caminha em passos opostos à preservação da política FIFO. Mesmo sabendo das vantagens inerentes à adesão do Sistema Kanban, existem favorecimentos na produção por lotes que podem se sobressair quando comparados às vantagens do Sistema Kanban. Assim, certas produções necessitam adaptar a produção por lotes ao funcionamento do Sistema Kanban (Savsar, 1997; Sivakumar & Shahabudeen, 2009). Algumas técnicas que auxiliam essas adaptações foram descritas na Tabela 2 (Roda Kanban; *Rolling Kanban*; Kanban triangular).

No contexto da empresa estudada, o estágio produtivo destinado à pintura (código - 2) realiza produção por lotes. Assim, por seguir o Sistema Kanban e em um dos estágios ocorrer produção por lotes, são necessárias adaptações. O estágio da pintura é composto por três máquinas, todas com capacidade de suportar mais de um kanban por processamento (produção por lotes). As máquinas 1 e 3 são responsáveis para aplicação da pintura propriamente dita e realização do acabamento. Já na máquina 2 realiza-se o processo de lixagem. Uma descrição de cada máquina e respetivo processo é feita a seguir.

- Máquina 1 (M1) – Pintura Primária (21): Consiste em um processo de pintura mais fundamental. Níveis de detalhamento são menores caso apenas essa máquina for utilizada. Os tipos de acabamento proporcionados pelo subprocesso são “baratos” (valor monetário). E as cores usadas são “básicas” (preto e branco);
- Máquina 2 (M2) – Lixagem (22): Necessária para todo fluxo que continuará no estágio da pintura pós pintura primária. É uma operação de apoio. Visa promover futura qualidade na fixação do acabamento e/ou tinta. Em outras palavras, é uma “limpeza” necessária para posterior pintura e/ou acabamento. Vale a ressalva que essa atividade retém um tempo maior de processamento que as outras duas;
- Máquina 3 (M3) – Pintura Acabamento (23): Consiste em um processo de extremo detalhamento à peça. Diversos acabamentos e cores podem ser empregados nesta máquina. Os efeitos produzidos na peça são de alta qualidade, e, portanto, o material consumido por esta máquina é “caro” (valor monetário).

As vantagens de se produzir por lotes (de kanbans) no estágio da pintura se dá por: 1 – pela redução de gastos com material. Dado que existe uma quantidade mínima de material para funcionamento da máquina, o material não consumido será desperdiçado após o processamento, caso o próximo kanban dispor outra configuração (pintura ou acabamento); 2 – pela redução de tempos necessários para trocar as configurações da máquina. Se decorrer uma troca entre cores, a máquina precisará ser limpa. O mesmo acontece em caso de diferente acabamento, com tempo ligeiramente menor do que uma troca de cores.

Claramente, a operacionalidade do estágio da pintura não funcionaria com as ferramentas destinadas à abordagem kanban. O sequenciador perder sua função, já que o fluxo FIFO não mais pode ser seguido. Mais, caso seja empregado o sequenciador, a visibilidade do operador para com elaboração dos lotes seria dificultada, pois sempre que fosse preciso elaborar o lote a produzir o operador teria de decidir a composição do lote conforme o acumulado de kanbans (terceiro posto do sequenciador). Esta decisão poderia requerer tempo e resultar em erros no reposicionamento no sequenciador, devido a poder haver uma grande quantidade de kanbans à espera de processamento. Portanto, pode-se afirma que o sequenciador no estágio da pintura é inefetivo.

Um dos problemas da atual operação consiste no procedimento de receção dos produtos e kanbans no estágio da pintura. Atualmente este procedimento não é padronizado. Problemas são vistos por esta não-padronização, principalmente devido ao fato de não existir um espaço específico para transmissão dos produtos (e respetivo kanban). Logo, o operador que leva os produtos até a pintura deposita em qualquer espaço físico disponível o produto e o kanban. Assim, quando se acumula diversos artigos (e kanbans) a visibilidade do operador é afetada pela dispersão em que os mesmos se encontram, não sendo possível o conhecimento, com absoluta certeza, dos itens que estão disponíveis para processamento. Como consequência, uma melhor composição dos lotes ou escalonamento de kanbans poderá não ser realizada.

Quanto a composição dos lotes, atualmente, essa fundamenta-se na similaridade tecnológica entre os kanbans, buscando adiar a necessidade com longos *setups* (produtos com tecnologias diferentes). Entende-se por “tecnologia”, aplicada no estágio da pintura, a cor e acabamento que serão atribuídos a um item. Uma escala de preferência entre a execução seguida de dois kanbans pode ser sistematizada na Figura 12.

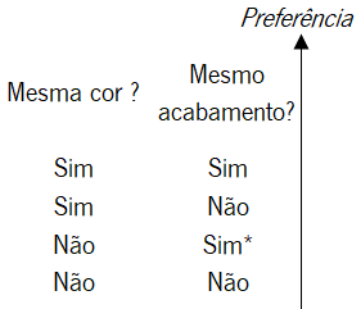


Figura 12 – Escala de preferência entre duas ordens a produzir (estágio – pintura)

Portanto, de acordo com a Figura 12 se existir na fila para processamento na máquina 1 ou na máquina 3 um kanban para pintar com a mesma cor (do kanban em processamento) e mesmo acabamento, a preferência desse kanban ser o próximo a processar é máxima (1ª combinação). Ao contrário (4ª combinação), se nenhum kanban em espera para processamento coincidir com tinta e acabamento, um novo lote será processado (com alto tempo de *setup*). Uma observação é importante para a 3ª combinação (*) (mesmo acabamento e diferente cor), se esse kanban precisar também pintar essa escala de preferência é quebrada. Ela se torna igual a última na escala de preferência. Motivo dado, pois caso necessite pintar, toda a máquina precisa ser reconfigurada (lavada) e o acabamento será perdido. Essa observação é válida devido a máquina 1 processar kanbans que necessitam apenas realizar um acabamento, logo, existe a possibilidade da 3ª combinação. Para simplificação, nesse trabalho um lote construiu-se levando em conta a família de cor, e dentro do lote, a sequência de processamento é liderada pela semelhança dos acabamentos.

3.2 Enquadramento do Estudo de Caso

Antes de enquadrar o estudo de caso na classificação $\alpha|\beta|\gamma$, é necessária uma descrição aprofundada do processo da pintura. Feito, cada campo da classificação pode receber, com clareza, uma respetiva atribuição.

A máquina 1 (M1) estabelece o início de todo o processo. Em outras palavras, qualquer ordem de produção que entra no estágio da pintura realizará a pintura primária (M1). Atualmente, o operador responsável por M1 é o realizador dos lotes de produção da mesma. Este operador leva em consideração os fatores tecnológicos (pintura e acabamento) descritos na Figura 12. Quando definido o lote a processar, o operador por meio do código de barras do kanban regista no sistema o início daquela(s) ordem(s). Para a máquina 3 (M3), o processo funciona idêntico à M1. A diferença é que a pilha de kanbans em M1 é originária dos estágios predecessores, enquanto em M3 a pilha estabelece-se pelo que foi processado a partir de M1. De maneira similar à M1, o operador responsável por M3 também é

o agente que constrói os lotes. Quanto a M2, a sequência de processamento é dependente da próxima máquina, M3 na maioria dos casos ou M1 em caso de recirculação.

É possível aperceber-se de uma falta de integração entre M1 e M3 quanto a elaboração dos respetivos lotes. Ganhos como menor acúmulo de material entre os estágios, redução do *makespan* e gastos desnecessários com tinta ou acabamento poderiam ser atingidos caso a elaboração dos lotes, e consequente escalonamento, não fosse feita de maneira individual. Basicamente, a atual operação ignora o formato *Flow shop* do ambiente, considerando cada máquina como uma única, portanto, escalonando as respetivas operações tendo em conta fatores exclusivos às máquinas.

Por fim dessa descrição, dois pormenores são relevantes: 1 - Sendo realizado M1 ou M3 é necessário o procedimento de secagem. Para algumas pinturas esse tempo é elevado, logo, naturalmente o estágio da pintura (código - 2) seja o gargalo no fluxo geral da produção. 2 - Excluindo o fator de secagem (descrito no primeiro pormenor), M2 é baseado em um trabalho manual de peça por peça, portanto, o tempo de processamento em M2 é maior que em M1 e M3. Entre M1 e M3 a comparação entre tempos é baseado nos *setups*. Assim, por M3 tem maior número de acabamentos, na maioria das vezes seu tempo total de processamento será maior que em M1.

Desprezando fluxos não usuais, um Kanban realizará o fluxo orientado por M1 → M2 → M3. Unindo este fato com a falta de interação descrita acima, o presente estudo focou exclusivamente neste fluxo. Considerou-se, portanto, que a estruturação do escalonamento do estágio e definição do procedimento geral, consistem em uma oportunidade de melhoria para a empresa estudada. Logo, a proposição de uma lógica de escalonamento e sua respetiva operacionalização, resumem os objetivos da dissertação.

3.2.1 Classificação $\alpha|\beta|\gamma$ - Pintura

Para o parâmetro $\alpha = \alpha_1\alpha_2$ relativo às máquinas do ambiente, temos: $\alpha_2 = 3$ por se tratar de três máquinas (M1, M2 e M3); e $\alpha_1 = F$, ambiente *Flow shop*, pois, qualquer que seja a ordem (Kanban), é visto um mesmo fluxo de processamento (M1 - M2 - M3).

Já o parâmetro β recebe as indicações de famílias ($fmls$); lotes ($s - batch$); sequência dependente (s_{ijk}). As famílias nesse caso são caracterizadas por famílias de cores. A execução do lote é feita em série ($s - batch$), portanto, com tempo de processamento descrito pela soma dos tempos de processamento de cada item. O conceito de "Lote disponível" também foi empregado, logo, todo o lote se desloca junto para a próxima máquina/tempos de conclusão das tarefas iguais para as mesmas que compõe o lote. Os tempos de *setup* dependem da sequência em que os lotes são feitos. Existe um tempo

específico para mudar entre uma família j para uma k. Outro ponto tange quanto a existência de uma data de lançamento "interna" (r_j). Tal data é guiada pelo processo de secagem (pós M1 ou M3), assim, um lote só poderá iniciar seu processamento na máquina $k+1$ após finalizar seu processamento em k com acréscimo do tempo necessário para secagem. O campo β não necessariamente precisa receber o indicativo de r_j, pois obviamente para todo ambiente *Flow shop* existirá uma data de lançamento orientada pela relação de precedência entre máquinas. Finalizando, vale a ressalva deste estudo ser um caso de *Non-permutation Flow shop*. Sendo possível a troca da sequência de lotes a processar por máquinas, evidentemente existe capacidade para "stockar" tarefas semiacabadas. No caso apresentado, o aguardo entre máquinas ocorre na maioria do tempo pela atividade de secagem. Pelo pequeno tamanho (área) dos artigos, considerou-se a capacidade de armazenamento nesses espaços como ilimitada, já que nenhum problema com espaço para secagem (ou qualquer outro aguardo) foi antes registado. Mais, a troca de sequência no processamento das tarefas *deve* ser feita. Motivo este pela existência de diferentes quantidades de famílias por máquinas. Na máquina 1 existem duas famílias (branco e preto), já na máquina 3 o número de famílias é superior a dez. Face à diferente configuração de famílias por máquina, tem-se a situação de lote inconsistente e famílias de tarefas incompatíveis.

Por fim, no campo γ o objetivo de otimização abordado foi a minimização do *makespan.* Assim, sintetiza-se para os três filtros: $F3|Fmls, s-batch, s_{ijk}|C_{max}$.

3.2.2 Nota: complexidade computacional do estudo

Nenhuma prova para com a complexidade computacional do problema descrito foi feita no presente estudo. Informalmente, compreende-se por teoria da complexidade computacional a avaliação do recurso (tempo) necessário por parte da máquina para atribuição de uma solução a partir do algoritmo fornecido. Sendo avaliado se o algoritmo é capaz (ou não) de fornecer soluções em tempo polinomial (C. N. Potts & Strusevich, 2009). A prova da complexidade computacional é uma atividade complexa, evidenciada por diversos trabalhos que tem como objetivo apenas a obtenção da mesma (*e.g.* Garey et al. (1976) e Ng e Kovalyov (2007). O conhecimento da complexidade computacional é de fundamental para construção do método de solução do problema. Mais, a informação sobre a complexidade computacional de problemas básico de escalonamento favorece conclusões sobre o status computacional de extensões daquele problema (C. N. Potts & Strusevich, 2009).

Mesmo com nenhuma prova propriamente feita, alguns artigos auxiliam para pensar que o problema descrito é *NP-Hard* em forte senso. Primeiramente, diz-se que um problema é *NP-Hard* caso este não for

mais “fácil” que nenhum outro problema pertencente à classe *NP.* A classe *NP* reúne problemas que podem ser resolvidos em tempo polinomial a partir de algoritmos não-determinísticos. Em um problema que é provado como *NP-Hard*, assume-se que é altamente improvável a existência de um algoritmo capaz de resolver o problema em tempo polinomial. Já termo “forte senso” caracteriza os problemas que permanecem *NP-Hard* sob codificação unária (C. N. Potts & Strusevich, 2009).

Na tentativa de situar o problema em questão ($F3|Fmls, s-batch, s_{ijk}|C_{max}$), alguns estudos podem ser citados. Para o problema de *Flow shop* regular como duas máquinas é provado como *NP-Hard* em forte senso em (Garey et al., 1976); Strusevich e Zwaneveld (1994) analisam um problema *Flow shop* com sequência dependência e concluem o status de *NP-Hard* em forte senso (sem necessidade do mesmo ser *Permutation Flow shop*) ao problema; T. C.E. Cheng et al. (2000) provam que um problema em ambiente *Flow shop* com máquinas processadoras de lotes (em especial *serial-batch*) é *NP-Hard* em forte senso; Por fim, Ng e Kovalyov (2007) obtiveram solução em tempo polinomial para *Flow shops* com processamento em lotes (*serial-batch*) e sequência dependente, entretanto, seu estudo não contemplou incompatibilidade de famílias, fato aumentaria a similaridade com o problema visto na presente dissertação.

O trabalho em questão não busca aprofundamento nas questões tocantes à teoria da complexidade computacional. Todavia, torna-se fundamental a tentativa de localizar o problema nesse quesito. A partir dos exemplos citados (problema mais básicos que o problema aqui tratado) pode-se concluir que o problema analisado nesta dissertação *tende* a ser *NP-Hard* em forte senso.

4. METODOLOGIA

Nos capítulos anteriores os conceitos necessários de esclarecimentos foram apresentados, assim como a descrição do estudo de caso e enquadramento do mesmo para a temática de Escalonamento. Para o capítulo em questão, o percurso metodológico é iniciado pela modelação matemática da situação de maneira genérica. Portanto, visando replicação e melhor entendimento, o modelo inicialmente apresentado não inclui algumas particularidades da empresa estudada. O passo seguinte é a adaptação/inclusão de algumas inequações/variáveis e parâmetros ao modelo previamente proposto, comentando os motivos das respetivas modificações em vigência das especificações da empresa. Toda representação matemática dos dois primeiros passos é feita de acordo com a estrutura apresentada no Capítulo 2 para modelos de *Programação Linear Mista*. Logo, a fase inicial da Metodologia modela o problema visando o entendimento e mais importante, a resolução por um método *exato*. Dando continuidade, o seguinte passo é a apresentação de um método aproximado para resolução do problema (Heurística), comentado, quando necessário, aonde as particularidades da empresa são incluídas nos passos para realização do procedimento. Por fim, na tentativa de estabelecer uma estrutura que facilite a operacionalização dos métodos descritos, a ferramenta *Rolling Kanban* é incluída no estudo de caso.

4.1 Modelo de Programação Linear Mista

Até onde conhecido, Shen e Gupta (2018) é o primeiro trabalho a propor um modelo de programação linear mista que objetiva a geração do escalonamento no contexto de: (1) *Flow shop*; (2) produção por lotes; (3) lote disponível; (4) família de produtos; (5) *setups* dependentes; e (6) lotes inconsistentes/ *Non-permutation Flow shop*. Para *famílias de produtos* os autores deixam implícito o conceito de incompatibilidade de tarefas/tarefas incompatíveis. Em seguimento o modelo proposto pelos autores:

$\underline{Parâmetros, Índices\ e\ Conjuntos}$:

n: número de tarefas;

j: índice para tarefas;

m: número de máquinas;

k: índice para máquinas;

f: índice e número de famílias;

b': número de possíveis lotes;

b: índice do lote;

N: conjunto de tarefas. $N = \{1, \dots, n\}$;

M: conjunto de máquinas. $M = \{1, \dots, m\}$;

F: conjunto de famílias. $F = \{1, \dots, f\}$;

B: conjunto de lotes. $B = \{1, \dots, b'\}$;

p_{jk}: denota o tempo de processamento de uma tarefa $j \in N$ na máquina $k \in M$;

B_{bk}: denota o b-néssimo ($b \in B$) lote processado em $k \in M$. Exemplo: B_{11} representa o primeiro lote feito na máquina 1;

s_{fgk}: denota o tempo gasto pela troca de processamento de um lote com tarefa(s) da família $f \in F \cup \{0\}$ para a família $g \in F$ na máquina $k \in M$. $s_{ggk} = 0$ e s_{0gk} representa o tempo necessário para iniciar o período produtivo a partir da família $g \in F$ em $k \in M$;

β_{jf}: parâmetro que recebe o valor de 1 caso a tarefa $j \in N$ pertença à família $f \in F$; 0 caso contrário;

$bigM$: um número grande.

Variáveis de decisão:

x_{jbk}: variável binária que recebe o valor 1 caso a tarefa $j \in N$ for agrupada ao lote $b \in B$ na máquina $k \in M$, 0 caso contrário;

T_{bk}: quantifica o tempo do início do lote B_{bk};

C_{jk}: quantifica o tempo de conclusão da tarefa $j \in N$ na máquina $k \in M$;

C_{max}: quantifica o makespan.

Modelo:

minimizar C_{max} (6)

sujeito a:

$$\sum_{b=1}^{B} x_{jbk} = 1 \,, \forall\, j \in N;\ k \in M \quad (7)$$

$$x_{jbk} \leq \beta_{jf} - \beta_{if} - x_{ibk} + 2 \,, \forall\, j \in N;\ k \in M; i \in N;\ b \in B; f \in F \quad (8)$$

$$\sum_{j=1}^{n} x_{j1k} \geq 1 \,, \forall\, k \in M \quad (9)$$

$$T_{1k} \geq s_{0gk} \,,\ \forall\, k \in M;\ B_{1k} \subseteq g \quad (10)$$

$$T_{ck} \geq T_{bk} + s_{fgk} + \sum_{j=1}^{n} x_{jbk} \cdot p_{jk} \,,$$
$$\forall b = 1, \dots b' - 1;\ c = 2, \dots, b';\ c > b; k \in M;\ B_{bk} \subseteq f; B_{ck} \subseteq g \quad (11)$$

$$C_{jk} \geq T_{bk} + \sum_{i=1}^{n} x_{ibk} \cdot p_{ik} - (1 - x_{jbk}) \cdot bigM \,,\ \forall\, j \in N; b \in B; k \in M \quad (12)$$

$$T_{b(k+1)} \geq C_{jk} - (1 - x_{jb(k+1)}) \cdot bigM \,,\ \forall\, j \in N; b \in B; k = 1, \dots, m-1 \quad (13)$$

$$C_{max} \geq T_{b'm} + \sum_{j=1}^{n} x_{jb'm} \cdot p_{jm} \quad (14)$$

O conjunto de restrições (7) garante que qualquer tarefa, em qualquer máquina, só seja agrupada a apenas um único lote. O conjunto de restrições (8) é relativo à impossibilidade da existência de um lote que comporta tarefas de diferentes famílias. Isto é: $\beta_{jf} - \beta_{if} - x_{ibk} + 2 = 0$, portanto $x_{jbk} = 0$. O conjunto de restrições (9) designa que qualquer B_{1k} tenha que conter pelo menos uma tarefa. As restrições (10) garante que o tempo de início do primeiro lote de qualquer máquina seja pelo menos o valor do *setup* despendido pela família g para iniciar aquela máquina. Obviamente, para $k = 1$ têm-se $T_{11} = s_{0g1}$ para $B_{11} \subseteq g$. O conjunto de restrições (11) também quantifica o tempo de início de um lote, mas garante que não haja sobreposição dos lotes. Em outras palavras, que um lote B_{bk}não inicie seu processamento antes de um lote B_{ck}, onde $c > b$. O conjunto de restrições (12) é relativo à premissa de lote disponibilidade, logo, garante que os trabalhos dispostos em um lote tenham o mesmo tempo de conclusão. Os conjuntos de restrições (13) e (14) definem o ambiente *Flow shop*. Para o conjunto (13) têm-se que o b-néssimo só inicie seu processamento na máquina sucessora $(k + 1)$ quando decorrer a conclusão das respetivas tarefas (que compõe $B_{b(k+1)}$) na máquina predecessora (k). Por fim, a restrição 14 quantifica o makespan para que C_{max} seja o valor referente ao início do último lote na última acrescido do tempo de processamento deste lote.

4.1.1 Adaptação do modelo de programação linear para o estudo de caso

Alguns pressupostos do modelo descrito acima necessitam ser comentados para que faça sentido as modificações propostas devido o estudo de caso. São eles: (1) o modelo acima toma em conta um horizonte temporal operacional *contínuo*, ou seja, não existe nenhuma atividade que gere interrupção no processamento das tarefas; (2) o *setup* entre tarefas de mesma família é zero $s_{ggk} = 0$; (3) as famílias são as mesmas para todas as máquinas; (4) o modelo acima considera um período produtivo contínuo, logo, é desconsiderado horas não-trabalhadas.

A característica de continuidade das operações contrapõe a existência do processo de secagem descrito para o estudo de caso. O procedimento, necessário, de secagem funciona com uma interrupção no processamento. Os gráficos de Gantt descritos respetivamente pela Figura 13a e Figura 13b mostram o efeito da inclusão da interrupção no horizonte temporal.

Figura 13a

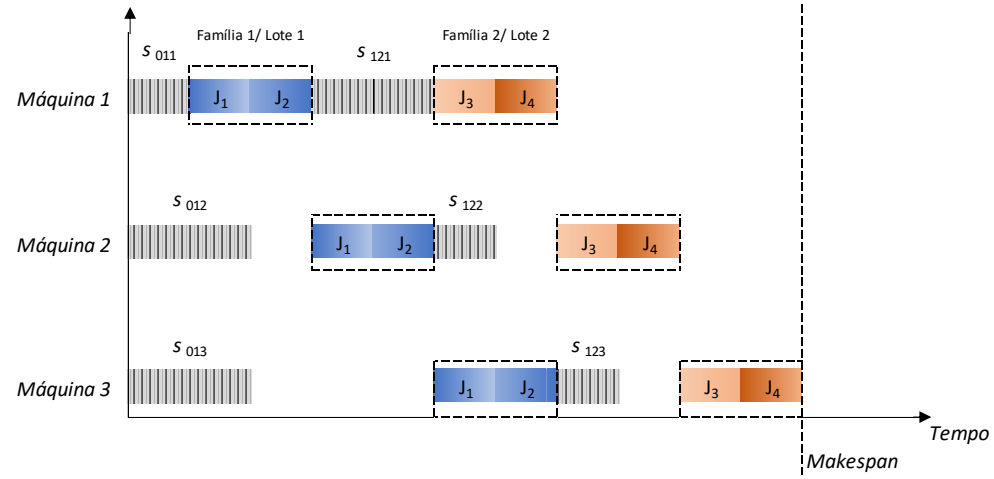


Figura 13b

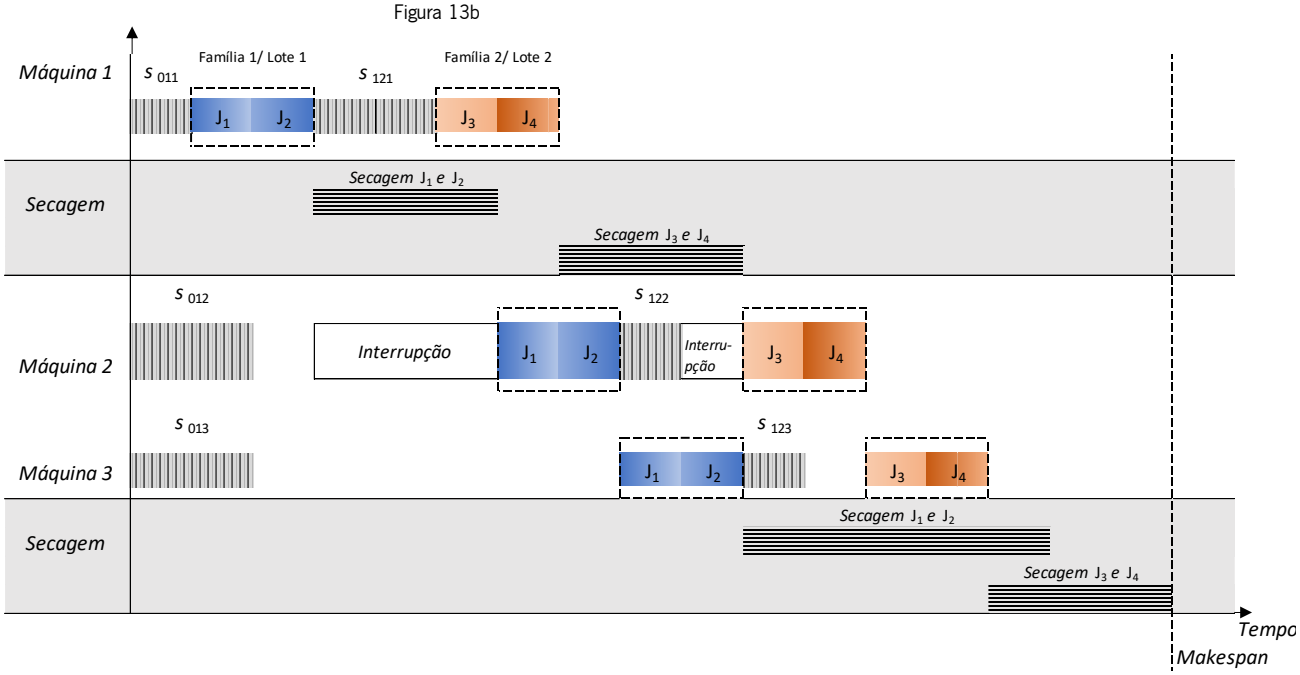


Figura 13 – Continuidade temporal das tarefas/máquinas

A partir da Figura 13b pode-se perceber que o aumento do makespan pela inclusão do procedimento de secagem. O termo "interrupção" foi utilizado para caracterizar uma descontinuidade das operações. Durante o período de secagem as máquinas estão "disponíveis", porém ociosas. Diferentemente dos problemas clássicos de Escalonamento, esse intervalo de não-produção é um reflexo de um procedimento intermédio, e não uma espera de finalização na máquina predecessora.

O efeito da existência do procedimento de secagem sobre o modelo de programação linear visto acima sugere duas inclusões (parâmetro e variável) e modificações nas restrições. A primeira inclusão é a do tempo de secagem sec_{jk}(parâmetro), e, portanto, o início de um lote na máquina sucessora deve ser

maior ou igual à conclusão das tarefas do respetivo lote na máquina predecessora *mais* o tempo de secagem entre a máquina predecessora e sucessora. Logo, podemos afirmar que existe uma *data de lançamento* como resultado do procedimento intermédio. Assim, r_{jk} será uma nova variável incluída (segunda inclusão) no modelo que representa a disponibilidade da tarefa $j \in N$ na máquina $k \in M$; formalizando, sec_{jk} é tempo de secagem específico da tarefa $j \in N$ *pós* a máquina $k \in M$. Assim, forma-se a seguinte relação (restrição):

$$r_{j(k+1)} = C_{jk} + sec_{jk}, \forall\, j \in N;\ k = 1, \dots, m-1 \tag{15}$$

Portanto, a disponibilidade de uma tarefa $j \in N$ na máquina sucessora $k+1$ é igual ao tempo de conclusão da respetiva tarefa na máquina predecessora k acrescido do tempo de secagem necessário pós processamento de j em k. Como neste caso específico o procedimento de secagem também existe pós a última máquina (m), assim makespan também deve ser modificado. Logo, pode-se escrever a quantificação do makespan para o caso como:

$$C_{max} \geq C_{jm} + sec_{jm}, \forall\, j \in N \tag{16}$$

Além da quantificação da disponibilidade e novo makespan, têm-se a limitação do início de um lote (que deve ser maior ou igual à disponibilidade), portanto:

$$T_{bk} \geq r_{jk} - (1 - x_{jb(k+1)}) \cdot bigM, \forall\, j \in N;\ k \in M;\ b \in B \tag{17}$$

Sendo o conjunto de restrições (17) a garantia de que um lote apenas inicie processamento de acordo com a disponibilidade.

Dando seguimento as alterações por existência de procedimentos intermédios, o conjunto restrições (12) restringe o tempo de conclusão de uma tarefa para com o tempo de conclusão do seu respetivo lote. A relação criada funciona até a última máquina, pois como o tempo de conclusão pode ser *maior ou igual* ao respetivo lote da tarefa nada limita a quantificação certa do tempo de conclusão na última máquina. Para facilitar o entendimento, dar-se um exemplo:

Cenário: $k = m;\ n = 2;\quad j = 1 \in F = 1;\ x_{11m} = 1\ ;\quad j = 2 \in F = 2;\ x_{22m} = 1;$
$p_{1m} = 10; p_{2m} = 10; s_{12m} = 10; T_{1m} = 200.$

Portanto, evidentemente a tarefa $j = 1$ será finalizada antes de $j = 2$, já que $j = 1$ está no primeiro lote da última máquina. Segundo o conjunto de restrições (12) têm-se a situação:

$C_{1m} \geq 200 + 1 \cdot 10 - (1 - 1) \cdot bigM \therefore C_{1m} \geq 210$,

logo: $T_{2m} = 210 + 10 = 220$,

assim, $C_{2m} \geq 220 + 1 \cdot 10 - (1 - 1) \cdot bigM \therefore C_{2m} \geq 230$. Como o objetivo é de minimização tem-se $C_{2m} = C_{max} = 230$. Entretanto é visto que nada impede a quantificação de C_{1m} entre $230 \leq C_{1m} > 210$. No modelo inicial proposto essa correção é feita pelo conjunto de restrições (14), como agora a quantificação do makespan foi substituída pelo conjunto (16) uma nova relação deve ser estabelecida para quantificação correta dos tempos de conclusão na última máquina. Sendo esta:

$$T_{bk} + \sum_{i=1}^{n} x_{ibk} \cdot p_{ik} \geq C_{jk} - (1 - x_{jbk}) \cdot bigM\ ,\ \forall\, j \in N; b \in\ B; k \in M \qquad (12.1)$$

Podendo concluir que o conjunto de restrições (18) serve como um complemento do conjunto de restrições (12).

A partir do estudo de caso descrito no Capítulo 3, nota-se que as famílias variam entre estágios (similar Isenberg e Scholz-Reiter (2013)). Assim, uma modificação necessária é apenas a inclusão de um índice no parâmetro β_{jf}. Portanto o novo parâmetro β_{jfk} é 1 caso a tarefa $j \in N$ pertença à família $f \in F$ na máquina $k \in M$; 0 caso contrário. Evidentemente, o conjunto de restrições (8) que faz presente esse parâmetro é alterado. Agora sendo:

$$x_{jbk} \leq \beta_{jfk} - \beta_{ifk} - x_{ibk} + 2\ ,\ \forall\, j \in N;\ k \in M; i \in N;\ b \in B; f \in F \qquad (8.1)$$

Outro ponto relevante para esse conjunto de restrições (8.1) se dá por uma sugestão de melhoria aqui proposta. Seja essas restrições apenas relevantes para o caso de restringir que diferentes famílias habitem o mesmo lote, então tal condição pode ser simbolizada por: $Se\ \beta_{jfk} + \beta_{ifk} = 1 \rightarrow \beta_{jfk} + \beta_{ifk} \geq x_{jbk} + x_{ibk}$. Ou seja, quando as famílias forem diferentes ($\beta_{jfk} + \beta_{ifk} = 1$) então o máximo de $x_{jbk} + x_{ibk}$ é 1. Assim, o conjunto de restrições para esse tipo de situação poderia ser modelado na forma de implicação, porém, feito isso o modelo poderia ter uma performance abaixo do desejado. Uma outra forma é modelar da forma que o conjunto (8.1) sugere. Para essa forma a relação é capaz de proibir a seguinte situação: $\beta_{jfk} = 0;\ \beta_{ifk} = 1; x_{ibk} = 1$ forçando x_{jbk} ser 0. Entretanto uma falha é vista para a situação com: $\beta_{jfk} = 1;\ \beta_{ifk} = 0; x_{ibk} = 1$, possibilitando x_{jbk} ser 1 (combinação que não deveria ser possível). Para proibir essa combinação, a seguinte relação é proposta:

$$x_{jbk} \leq \beta_{ifk} - \beta_{jfk} - x_{ibk} + 2\ ,\ \forall\, j \in N;\ k \in M; i \in N;\ b \in B; f \in F \qquad (8.2)$$

Como pode ser observado, o conjunto de restrições (8.2) não seria capaz de proibir $x_{jbk} = 1$ para a combinação $\beta_{jfk} = 0;\ \beta_{ifk} = 1; x_{ibk} = 1$. Porém, quando em conjunto com as restrições (8.1) ambas combinações elencadas acima tornam-se impossíveis.

Até o presente momento, com as modificações sugeridas pode-se formalizar um modelo de programação linear para o escalonamento em um ambiente *Flow shop* com famílias de produtos, *setups* dependentes e máquinas processadoras de lotes (*serial batch*). Tendo as premissas de lote disponibilidade, processos intermédios e diferentes famílias por máquinas (estágios) sido introduzidas ao modelo.

Outras dois modificações ainda podem ser feitas ao modelo, trazendo um caráter mais realístico ao modelo. A primeira é uma possibilidade de dividir um lote e processá-los em sequência. Essa situação foi ilustrada na Figura 7 (segundo e terceiro gráfico). Basicamente, se não houver benefício, seja este de tempo ou financeiro, o lote pode ser dividido e processado em sequência. Note que há uma diferença para a situação descrita em Shen e Gupta (2018), onde o lote pode ser dividido, porém, sendo o *setup* para processamento de uma mesma família zero ($s_{ggk} = 0$) não faz sentido a divisão do lote e processamento seguido dos lotes dividido. Aqui no estudo, abre-se esta possibilidade, já que os *setups* entre troca de uma mesma família é diferente de zero ($s_{ggk} > 0$). O reflexo desse adento apenas sugere uma modificação nos conjuntos do modelo e seguida remodelação destes nas restrições (11).

A última ressalva sugere, talvez, a característica mais realista a introduzir no modelo. Na imensa maioria dos problemas de Escalonamento é considerado um horizonte temporal contínuo. Ou seja, os recursos estão disponíveis o tempo inteiro. Isto na verdade, na maioria dos ambientes, é uma simplificação. Por exemplo, trabalhadores realizam suas ações apenas durante os respetivos turnos de serviço, enquanto os problemas clássicos de escalonamento irá considerar uma disponibilidade temporal contínua. Assim, para o estudo de caso e respetiva adaptação, considerou-se os períodos que não se pode processar ordens. Isso implica diretamente na *janela temporal* da máquina 1, máquina 2 e máquina 3, em que a cada intervalo de tempo pré-definido as operações não poderão ser afetadas às máquinas. Entretanto, o mesmo não acontece com o procedimento de secagem, em que neste pode ser feito uso de *horas não-produtivas*. A Figura 14 ilustra as diferentes disponibilidades para Máquina 1, Máquina 2, Máquina 3 e respetivos procedimentos de secagem ao longo de um exemplo para 8 dias.

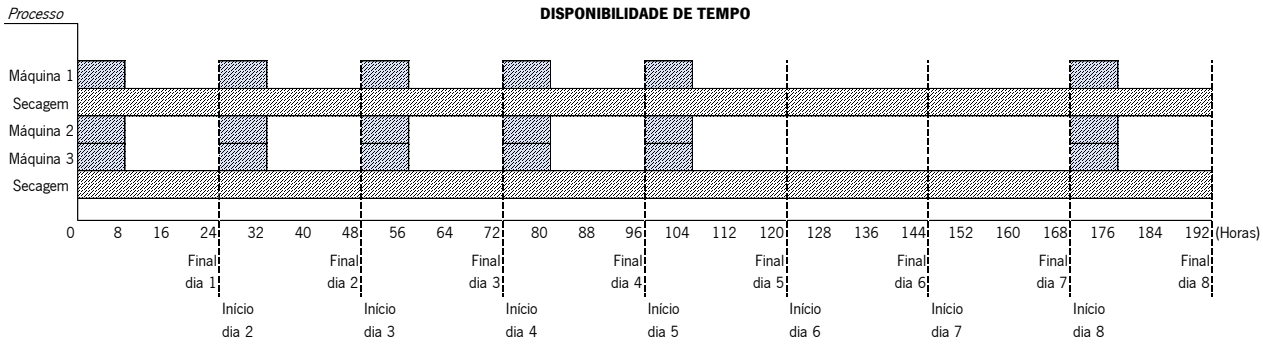


Figura 14 – Disponibilidade de tempo semanal

A Figura 14 simula uma semana produtiva, seu sucessivo fim de semana e início de semana produtiva seguinte. Como visto, uma tarefa $j \in N$ não pode ser executada em uma determinada máquina $k \in M$ em uma janela temporal posterior às 8 horas produtivas do respetivo dia até o início do próximo dia produtivo. Por outro lado, o processo de secagem está sempre disponível. Assim, uma tarefa finalizada na máquina 1 ou máquina 3 e movida para secagem pode fazer uso integral do recurso temporal. A partir dessa particularidade entre procedimentos, o modelo matemático deve ser capaz de não permitir uma execução de uma tarefa $j \in N$ em uma máquina $k \in M$ durante horas não-produtivas. Uma solução para inserção dessa condição é a inclusão de tarefas fictícias. Essas tarefas devem ter o respetivo tempo de conclusão igual ao final de um dado dia e duração para processamento de 16 horas (para semana produtiva). Uma representação de tal proposta é exemplificada na Figura 15.

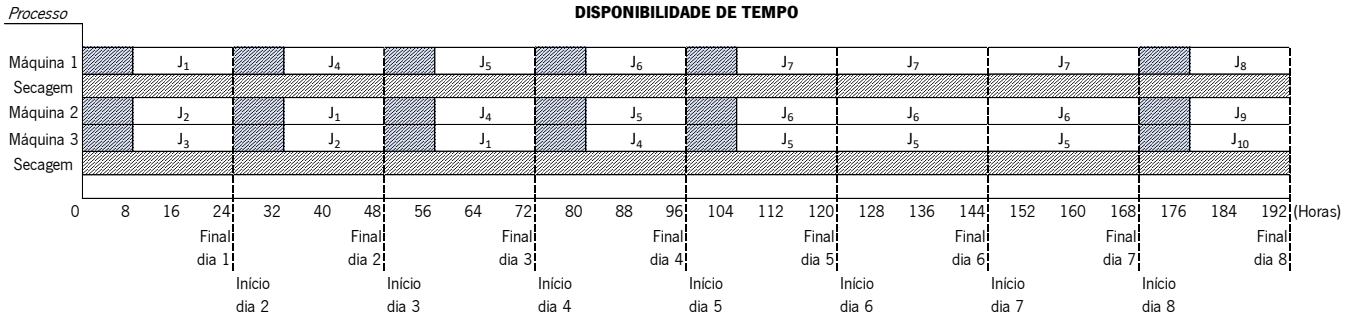


Figura 15 – Escalonamento de tarefas fictícias ao longo do tempo

As tarefas fictícias $j_1, \dots, j_{10}$ portanto devem ser restritas quando ao tempo de conclusão em cada máquina. Além do mais, essas tarefas devem ser processadas individualmente, ou seja, o lote que contém uma dessas tarefas é composto apenas pela respetiva tarefa. Para que a composição do lote seja de uma tarefa basta indicar uma família diferente para cada tarefa fictícias.

Uma importante observação vista na Figura 15, se dá para algumas tarefas que iniciam seu processamento pela máquina 2 ou máquina 3. Tal comportamento não pode ser permitido em ambientes

Flow shop. Para contorno desta situação, designa-se os respetivos tempos de processamento e *setups* existentes como nulos. Claramente, para um escalonamento de um grande número de tarefas, um algoritmo deve ser desenvolvido para rápido criação e preenchimento no tempo das tarefas fictícias.

Para concentrar as modificações, com exclusão do pormenor descrito na Figura 14 e Figura 15 (que requer, para melhor reprodução, inclusão de um procedimento heurístico), têm-se o novo modelo matemático:

$\underline{Parâmetros, Índices\ e\ Conjuntos}$:

n: número de tarefas;

j: índice para tarefas;

m: número de máquinas;

k: índice para máquinas;

f: índice e número de famílias;

g: índice de famílias;

b: índice e número de lotes;

N: conjunto de tarefas. $N = \{1, \dots, n\}$;

M: conjunto de máquinas. $M = \{1, \dots, m\}$;

$F\{k \in M\}$: conjunto de famílias na máquina $k \in M$. $F\{k\} = \{1, \dots, f\}$;

B: conjunto de lotes. $B = \{1, \dots, b\}$;

$J'\{k \in M, g \in F\{k\}\}$: conjunto de tarefas $j \in N$ que pertencem a uma família $g \in F\{k\}$ em $k \in M$;

p_{jk}: denota o tempo de processamento de uma tarefa $j \in N$ na máquina $k \in M$;

B_{bk}: denota o b-néssimo ($b \in B$) lote processado em $k \in M$. Exemplo: B_{11} representa o primeiro lote feito na máquina 1;

s_{fgk}: denota o tempo gasto pela troca de processamento de um lote com tarefa(s) da família $f \in F\{k\} \cup \{0\}$ para a família $g \in F\{k\}$ na máquina $k \in M$. s_{0gk} representa o tempo necessário para iniciar o período produtivo a partir da família $g \in F\{k\}$ em $k \in M$;

β_{jfk}: parâmetro que recebe o valor de 1 caso a tarefa $j \in N$ pertença à família $f \in F\{k\}$ na máquina $k \in M$; 0 caso contrário;

sec_{jk}: parâmetro para o procedimento de secagem. Tempo despendido para secar a tarefa $j \in N$ pós processamento na máquina $k \in M$;

$bigM$: um número grande.

Variáveis de decisão:

x_{jbk}: variável binária que recebe o valor 1 caso a tarefa $j \in N$ for agrupada ao lote $b \in B$ na máquina $k \in M$, 0 caso contrário;

T_{bk}: quantifica o tempo do início do lote B_{bk};

C_{jk}: quantifica o tempo de conclusão da tarefa $j \in N$ na máquina $k \in M$;

r_{jk}: data de lançamento da tarefa $j \in N$ na máquina $k \in M$;

C_{max}: quantifica o makespan.

Modelo:

minimizar C_{max} (6)

sujeito a:

$$\sum_{b=1}^{B} x_{jbk} = 1 \,, \forall\, j \in N;\ k \in M \quad (7)$$

$$x_{jbk} \leq \beta_{jfk} - \beta_{ifk} - x_{ibk} + 2 \,, \forall\, j \in N;\ k \in M; i \in N;\ b \in B; f \in F\{k\} \quad (8.1)$$

$$x_{jbk} \leq \beta_{ifk} - \beta_{jfk} - x_{ibk} + 2 \,, \forall\, j \in N;\ k \in M; i \in N;\ b \in B; f \in F\{k\} \quad (8.2)$$

$$\sum_{j=1}^{n} x_{j1k} \geq 1 \,, \forall\, k \in M \quad (9)$$

$$T_{1k} \geq s_{0gk} \cdot x_{j1k} \,, \forall\, k \in M; g \in F\{k\}; j \in J'\{k,g\} \quad (10.1)$$

$$T_{ck} \geq T_{bk} + s_{fgk} \cdot x_{jbk} + \sum_{j' \in J'\{k,f\}} x_{j'bk} \cdot p_{jk} - (1 - x_{ibk}) \cdot bigM \,,$$
$$\forall\, b = 1, \dots b-1;\ c = 2, \dots, b;\ c > b; k \in M; j \in J'\{k,f\}; i \in J'\{k,g\}; i \neq j;\ f \in F\{k\}; g \in F\{k\} \quad (11.1)$$

$$C_{jk} \geq T_{bk} + \sum_{i=1}^{n} x_{ibk} \cdot p_{ik} - (1 - x_{jbk}) \cdot bigM \,,\ \forall\, j \in N; b \in B; k \in M \quad (12)$$

$$T_{bk} + \sum_{i=1}^{n} x_{ibk} \cdot p_{ik} \geq C_{jk} - (1 - x_{jbk}) \cdot bigM \,,\ \forall\, j \in N; b \in B; k \in M \quad (12.1)$$

$$T_{b(k+1)} \geq C_{jk} - (1 - x_{jb(k+1)}) \cdot bigM \,,\ \forall\, j \in N; b \in B; k = 1, \dots, m-1 \quad (13)$$

$$r_{j(k+1)} = C_{jk} + sec_{jk}, \forall\, j \in N;\ k = 1, \dots, m-1 \quad (15)$$

$$T_{bk} \geq r_{jk} - (1 - x_{jb(k+1)}) \cdot bigM, \forall\, j \in N;\ k \in M;\ b \in B \quad (17)$$

$$C_{max} \geq C_{jm} + sec_{jm}, \forall\, j \in N \quad (16)$$

A partir da nova estruturação do modelo, alguns comentários devem ser feitos. Assim como explicado para o conjunto de restrições (8.1), (8.2) e (12.1), o conjunto de restrições (10.1) e (11.1) representam apenas alterações dos conjuntos (10) e (11) para a nova estrutura dos índices e conjuntos do modelo.

Além disso, a simbologia para os *setups* (s_{fgk}) acompanha nestas restrições da variável x_{jbk}, representando a existência do *setup* apenas quando $x_{jbk} = 1$.

4.2 Heurística Proposta – NEH Modificada

Ao passo do esclarecimento (Capítulo 2) que problemas de escalonamento podem ser resolvidos por métodos aproximados, o presente trabalho fez uso de uma heurística, nomeadamente uma modificação da NEH heurística, para solucionar o problema descrito. A NEH heurística é talvez a principal heurística no contexto de ambientes *Flow shop*. Obviamente, por se tratar de um método aproximado, a solução entregue pela NEH heurística não garante a solução ótima, mas diversos autores (*e.g.* Dong et al. (2008) e Kalczynski e Kamburowski (2008)) elegem a NEH heurística como o melhor algoritmo construtivo para este tipo de ambiente (*Flow shop*) (W. Liu et al., 2017). No entanto, a NEH heurística foi desenvolvida para o já definido *Flow shop* regular, o que faz necessário modificações no algoritmo para receber e solucionar problemas *Flow shop* "não-regulares".

No que toca o estudo de caso, o conceito de famílias e consequentes *setups* são as principais modificações a serem inclusos no método. A começar pelo tratamento de famílias, a modificação da NEH heurística proposta aqui preocupa-se que o resultado do método seja o escalonamento destas famílias, utilizando o conceito de *tecnologia de grupo*. O escalonamento dentro da família não é de necessária preocupação. Este não precisa ser abordado já que se trata de um caso de *lote disponibilidade*. Como já pode ser avaliado, os conceitos de lotes inconsistentes e *Non-permutation Flow shop* não existe para o método, podendo, ou não, piorar o desvio da resposta com a resposta ótima. Pela inclusão dos *setups*, o trabalho tomou como base o conceito de tempo efetivo de processamento por máquina explicado em Schaller et al. (2000). Com o adento que foi necessária uma adaptação para cálculo do tempo efetivo de processamento, justificado por tarefas mudarem de famílias nas diferentes máquinas. O Algoritmo 1 norteia os passos para execução do método. As equações 18 e 19 suplementam o Algoritmo 1. Para maior exemplificação dos passos descritos no Algoritmo 1, o Apêndice A demonstra o pseudocódigo em Python 3.9.1

Algoritmo 1	NEH-Modificada
Passo 1	Calcular o *setup* médio da família f ϵ F na máquina j (Equação 18)
Passo 2	Calcular o tempo efetivo de processamento de cada família na máquina j (Equação 19)
Passo 3	Classifique as famílias de acordo com os maiores tempos efetivos de processamento
Passo 4	Classificar as duas famílias com maiores tempos efetivos de processamento e verificar qual sequência entre elas resulta em um menor *makespan.* Defina a sequência parcial como $\delta = (\delta_1, \delta_2)$ e faça i = 2
Passo 5	Para o restante das famílias inclua a seguinte família e desenvolva i+1 sequências parciais. Verifique qual das sequências resulta em um menor *makespan*, tornando a mesma a sequência parcial para i+1 famílias
Passo 6	Verifique se i < K, se sim, faça i = i+1 e retorne ao passo 5; caso contrário o escalonamento foi definido com o respetivo *makespan*

$$\bar{s}_{jf} = (\sum_{x=1}^{K} s_{fxj})/K \quad (18)$$

$$E_{fj} = \bar{s}_{jf} + \sum_{x \in N_f} p_{xj} \quad (19)$$

A adaptação necessária que diverge do dito em Schaller et al. (2000) consiste em ter em consideração que as tarefas pertencentes as famílias não são fixas ao longo das máquinas. Ou seja, o cálculo para o *setup* médio por máquina (Equação 18) não pode ser seguido como em Schaller et al. (2000). Para contorno da situação, o presente trabalho propôs que a somatória ocorra em função de uma matriz tarefa x família exemplificada abaixo.

1	0	0	0	0	0
1	0	0	0	1	0
1	0	0	0	0	1
1	0	0	0	1	0
0	1	0	0	0	0
0	1	0	0	0	0
1	0	1	0	0	0
0	1	0	0	0	0
1	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	0	0
0	1	0	1	0	0

Na situação exemplificada, 12 tarefas pertencem a 6 famílias. Em um cenário "padrão" a somatória das linhas seria 1, representando que cada tarefa tem apenas uma família. Nesta adaptação uma tarefa pode "trocar" de família durante o processo produtivo. O exemplo é dado por uma situação com 3 máquinas em que na primeira máquina existem 2 famílias e 6 famílias nas restantes máquinas. As duas famílias da primeira máquina podem ser consideradas como origem, e são representadas pelas primeiras duas colunas da matriz acima. Na linha dois tem o primeiro exemplo de tarefa com "duas famílias". Para a tarefa 2 (linha dois) é visto que esta pertence as famílias 1 e 5, ou seja, na primeira máquina esta tarefa pertence à família 1 e nas restantes máquinas à família 5. Da mesma forma dita para a linha 2, as linhas 3,4,7,9,10 e 12 são exemplos similares.

A partir da matriz explicada no parágrafo anterior, o funcionamento da NEH heurística, nomeadamente passo 4, tenderá a ter famílias de "origem" como as duas selecionadas para compor a primeira sequência parcial. No entanto, como o restante do método baseia-se na inserção de tarefas nas possíveis posições, duas famílias de "origem" dificilmente serão adjacentes (nas duas primeiras posições a processar), demonstrando coerência na adaptação. Evidentemente que para o cálculo do *makespan* a matriz exemplificada não pode ser utilizada, assim, portanto, considera-se apenas as famílias "finais" das tarefas, e não as de "origem", portanto, uma nova matriz é vista com soma de linhas igual a 1.

Outra nota importante que diverge do descrito em Schaller et al. (2000) está na equação 18. No que representa o *setup* de mudança de família s_{fxj} $(f \rightarrow x)$ em Schaller et al. (2000) aparece como $s_{xfj}(x \rightarrow f)$. Essa mudança é representativa para o cálculo do tempo efetivo de processamento (equação 19) e consequente conclusão no passo 4. A mudança proposta aqui é justificada devido o objetivo de *construir* uma solução (a partir da primeira máquina), logo, não faz sentido avaliar o *setup* médio de uma família despendido entre famílias *anteriores* a esta família $x \rightarrow f$. Sendo então, necessário avaliar *setups* para famílias que sucedem aquela $f \rightarrow x$. Sumarizando, a Equação 18 fundamental para os passos de 1-4 atua sob a lógica da matriz descrita, tendo o tempo efetivo de processamento (Equação 19) como resultado da Equação 18 acrescido do tempo de processamento de todas as tarefas que compõe a família avaliada ($\sum_{x \in N_f} p_{xj}$).

Uma importante nota sobre a utilização da NEH-Modificada se dá em virtude da mesma ser idealizada para a situação de estudo descrita. Onde, deve-se ter atenção que o nível de customização aumento ao longo das máquinas. Outras palavras, a quantidade de família na máquina 1 é menor que na máquina 3. Portanto, para replicação da NEH-Modificada é necessário encontrar a máquina com maior quantidade

de famílias e utilizar os passos ditos acima. Outra importante questão está para uma realação existente entre as famílias (origem x finais), em caso das famílias, ao trocarem de máquinas, serem completamente independentes das máquinas anteriores, ocorre a impossibilidade de replicação da Heurística descrita.

4.3 Utilização da Ferramenta *Rolling Kanban*

A utilização da ferramenta de controlo visual *Rolling Kanban* é justificada neste trabalho por alguns aspetos: 1- sendo o estudo de caso aplicado a uma empresa aderente ao Sistema Kanban e paralelamente praticante de produções em lotes, faz-se necessário ferramentas/técnicas que auxiliem esta adaptação; 2- face a já utilização de uma ferramenta para alocar os kanbans de produção (sequenciador – Figura 11), primeiramente a antiga ferramenta deve ser descontinuada, posteriormente encontrar uma nova ferramenta para manter a filosofia do Sistema Kanban visto na empresa; 3- o *Rolling Kanban* apesar de pouco visto na literatura mostrou-se, pelo estudo de Braglia et al. (2020), facilmente adaptável e possibilitador que bons resultados.

A metodologia por trás do *Rolling Kanban* diverge sutilmente nos únicos dois trabalhos que a ferramenta foi imposta. Em Boyer (2004), vê-se uma abordagem onde previamente não existe nenhuma tarefa para ser sequenciada, logo, os decisores aguardam, de acordo com um *lead time* máximo definido, para atribuir um escalonamento para produção das famílias. Já em Braglia et al. (2020) vê-se a ferramenta *Rolling Kanban* a atuar sob uma produção que inicialmente já existem tarefas a serem escalonadas. Em ambos os casos o horizonte temporal é seccionado em períodos produtivos (dias). Entretanto, como em Braglia et al. (2020) já existem tarefas a serem produzidas, com somatória do tempo de processamento maior que os períodos produtivos, ou seja maior que um dia, já se pode tentar definir um escalonamento da produção desde o instante inicial. Para Boyer (2004) o escalonamento não precisa de uma definição imediata, sendo possível por exemplo dias produtivos sem processamento.

Para a empresa em estudo e para o autor desta dissertação, dias produtivos sem processamento de produtos é uma situação que não deve ocorrer. Uma justificação para tal ação deveria ser seguida de um custeio muito eficaz da produção juntamente com um contexto produtivo de baixa incerteza e baixa variação nos processos. Assim, apoiado no estudo prático de Braglia et al. (2020) e opiniões de especialistas, a metodologia vista em Braglia et al. (2020) é mais coerente com a realidade encontrada nas organizações.

Tomando como base o trabalho de Braglia et al. (2020), podemos sintetizar a metodologia *Rolling Kanban* como um algoritmo/fluxograma (Figura 16). Vale a ressalva que Braglia et al. (2020) já adaptou o processo do *Rolling Kanban* visto em Boyer (2004), portanto, atualizando a ferramenta de acordo com reais necessidades.

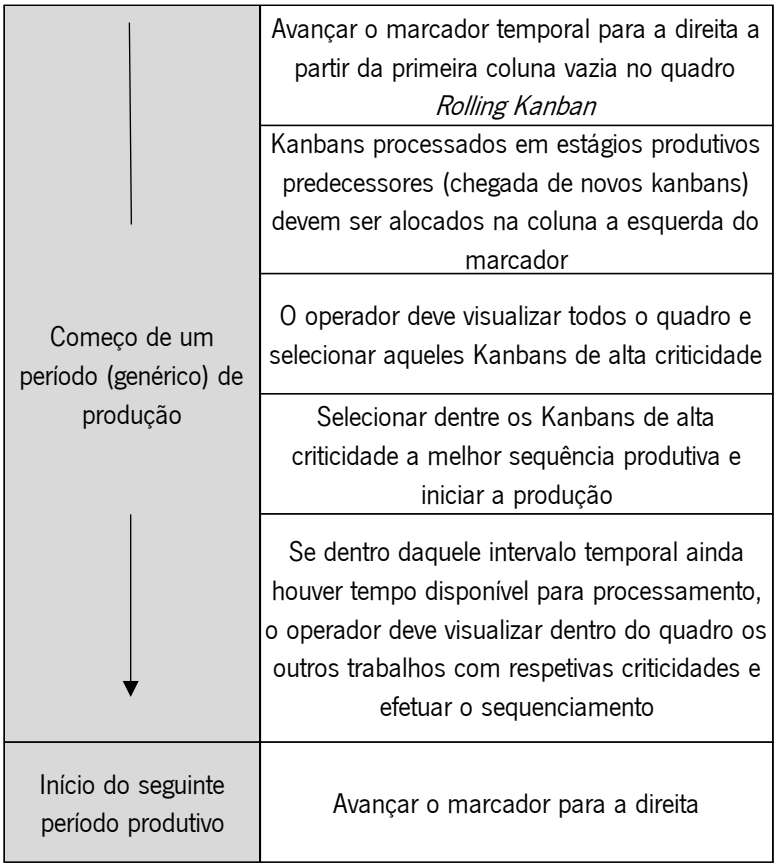


Figura 16 – Etapas *Rolling Kanban* (adaptado de Braglia et al. (2020))

A partir da Figura 16 deve-se perceber algumas particularidades produtivas do ambiente estudado por Braglia et al. (2020). Iniciando pelo 4° processo (separar pela criticidade), a palavra "criticidade" obviamente deve ser de acordo com o ambiente estudado. Em Braglia et al. (2020) um kanban mais crítico é aquele com stock insuficiente (da família) para todo o horizonte produtivo. Logo, parte da fam+ilia é processada (adiantando a produção) e um posterior pedido de abastecimento é feito.

No estudo de caso proposto o principal fator estudado é o *makespan*. Sendo esta a métrica utilizada para condicionar o modelo matemático e heurística previamente descritos, a utilização do *Rolling Kanban* deve ter em consideração uma adaptação quanto o *makespan*. Escusado seria dizer que outros objetivos poderiam ser influenciadores na utilização da técnica. No entanto, como a ferramenta aqui é *auxiliar* a

decisão de escalonamento, a mesma deve ser completamente influenciada pela estruturação dos métodos de otimização. Face ao *makespan*, uma adaptação da Figura 16 é proposta (Figura 17).

Começo de um período (genérico) de produção	Estruturar os intervalos temporais que melhor caracterização um horizonte produtivo para o contexto produtivo
	Avançar o marcador temporal para a direita a partir da primeira coluna vazia no quadro *Rolling Kanban* (início da utilização fisíca do quadro)
	Executar o modelo matemático **ou** Heurística para colher o escalonamento do horizonte produtivo
	Selecionar, a partir da resposta do método de otimização, os kanbans que irão ser processados naquela secção do horizonte temporal e iniciar a respetiva produção
	Kanbans processados em estágios produtivos predecessores (chegada de novos kanbans) devem ser alocados na coluna a esquerda do marcador
	Se dentro daquela secção temporal ainda houver tempo disponível para processamento, o operador deve executar novamente o método de otimização e continuar o processamento a partir da resposta do método
Início do seguinte período produtivo	Avançar o marcador para a direita

Figura 17 – Etapas *Rolling Kanban* adaptado ao estudo atual

Alguns pontos a destacar pelo novo processo proposto: 1- como para o *makespan* não existe fatores que implicam em kanbans mais importantes que outros, o 4º passo da Figura 17 foi retirado e o 5º passo da mesma foi reestruturado; 2- é possível ver uma utilização dinâmica do método de otimização, já que kanbans podem chegam ao logo do tempo. Logo, alguns pormenores devem ser executados com certa cautela; e 3- tanto o modelo exato como o heurístico podem ser utilizados. A respetiva utilização dependerá da quantidade de tarefas a serem escalonadas (tópico discutido no Capítulo 5).

Quanto ao segundo ponto destacado no parágrafo anterior, alguns cenários são possíveis em ambientes dinâmicos e consequentes respostas do modelo matemático devem ser interpretadas. Relembrando que

toda o processo funciona sob a definição de um horizonte planeado de produção e respetiva segmentação do mesmo. A utilização do método de otimização deve ser feita tendo em conta daquela secção do horizonte de produção. Entretanto, com a chegada de novas tarefas dois importantes pontos devem ser incluídos no modelo: 1- a partir da execução do método observar se ainda é possível a inclusão de tarefas respeitando o horizonte temporal definido, portanto, na chegada de novas tarefas e posterior execução do método de otimização, se a resposta ultrapassar o horizonte produtivo os kanbans inclusos devem ser retirados (de acordo com o último a chegar) e processados no seguinte horizonte produtivo (próxima semana); 2- a execução do modelo deve-se ter a cautela de analisar se ainda existe tempo para aquele segmento de tempo, se sim, o modelo (no caso do exato) deve ser executado como qualquer segmento de tempo qualquer, entretanto deve-se ter consideração a família recém processada e incluir este *setup* como o s_{00k} do modelo. Um exemplo com tempos fictícios é realizado no Capítulo 5.

Finalizando o descritivo da adaptação do *Rolling Kanban*, vale ressaltar algumas dificuldades de implementação citadas em Braglia et al. (2020) que são eliminadas com a integração do método de otimização. Em Braglia et al. (2020) faz-se necessário uma utilização de uma ferramenta visual para controlo do tempo de processamento e respetiva escolha do escalonamento. Claramente, caso aplicado o método de otimização esta etapa deixa de ser necessária. Outro ponto, válido apenas para o caso de *lote disponível*, é a não necessidade de encontrar uma sequência dentro da família (lote), todavia, vale reforçar a importância da identificação entre *lote disponível* e *tarefa disponível*.

5. RESULTADOS E DISCUSSÕES

O capítulo em questão tratou da utilização prática dos modelos descritos no capítulo anterior. Iniciado pela apresentação dos dados (instâncias) utilizados para percorrer os experimentos computacionais, o capítulo desenrola-se com o seguimento dos experimentos, passando tanto pelo modelo exato quanto modelo aproximado. Para cada experimento, as respetivas discussões acerca dos resultados foram realizadas, se atentando principalmente a métrica do tempo computacional necessário para encontro de uma solução. Além dos próprios experimentos, a justificativa de se estudar o modelo aproximado é feita tendo como base dados reais do estudo de caso. Por fim do capítulo, o enquadramento do método de otimização exemplificado é feito para com a ferramenta de controlo visual *Rolling Kanban*, contemplando o principal objetivo da dissertação.

5.1 Instâncias Iniciais – Experimentos Computacionais

Após a elaboração do método exato para solução do problema em questão, faz-se necessário observar o comportamento do mesmo para diferentes instâncias. Antes de qualquer análise mais profunda, uma simples observação pode ser feita a partir de dados fictícios. A partir desta, pode-se aperceber-se de como o modelo comporta-se, seja com o aumento de variáveis ou com respetivo reflexo no tempo computacional para encontro da solução. A Tabela 4 caracteriza os elementos que compõe as onze instâncias testadas. Os números observados na Tabela 4 foram pensados para possibilitar ao leitor o entendimento visual dos números, já que caso fosse considerado instâncias mais elaboradas no tocante ao número de famílias por máquina, a Tabela 4 cresceria horizontalmente de maneira a impossibilitar a ilustração dos números. Posteriormente, a Tabela 5 preocupou-se com apresentar os resultados encontrados após atribuição do modelo exato e respetivas instâncias ao NEOS SERVER (introduzido no Capítulo 2), realizados (modelo + instância) na linguagem AMPL (exemplo: Apêndice B).

Tabela 4 – Instâncias Iniciais

Instância Nome	n	m	b	F1	F2	F3
VD01	2	2	2	2	2	-
VD02	3	2	2	2	2	-
VD03	3	2	3	2	2	-
VD04	4	2	4	2	2	-
VD05	4	3	3	2	3	3
VD06	4	3	4	2	3	3
VD07	6	3	6	2	3	3
VD08	8	3	8	2	4	4
VD09	8	3	6	2	4	4
VD10	10	3	10	2	4	4
VD11	12	3	12	2	6	6

A Tabela 4 reúne em cada coluna (com exclusão do nome da instância) seis parâmetros que estão contidos no modelo descrito no Capítulo 4. Seguindo o descritivo do modelo, “n” representa o número de tarefas, “m” o número de máquinas, “b” o número de lotes permitidos, “F1” a quantidade de famílias que existem na máquina 1, “F2” a quantidade de famílias que existem na máquina 2 e “F3” a quantidade famílias que existem na máquina 3. Como pode ser observado, entre as primeiras cinco instâncias pequenas nuances são vistas. Uma variação é clara quanto ao número de tarefas e máquinas. Entretanto, um pormenor estudado também foi a reação do modelo quando *não* necessariamente o número máximo de lotes fosse o número de tarefas (*e.g.* instâncias VD08 e VD09). Para além desses seis parâmetros, estão em falta o tempo de processamento das tarefas, tempos de *setups* e tempos de secagem, descritos a seguir: para os tempos de processamentos atribuiu-se um valor uniforme entre 20 minutos e 30 minutos; para *setups* atribuiu-se valores uniformes entre 10 minutos e 25 minutos para a máquina 1 e valores uniformes entre 15 minutos e 30 minutos para as máquinas 2 e 3; e para os tempos com secagem usou-se valores uniformes entre 100 minutos e 200 minutos.

Tabela 5 – Resultados das instâncias iniciais

	Gurobi		CPLEX					
Instância Nome	Tempo Total	Gap	Tempo Total	Gap	Quantidade Restrições	Não-zeros	Variáveis Lineares	Variáveis Binárias
VD01	00:00:01	0%	00:00:03	0%	62	176	13	8
VD02	00:00:01	0%	00:00:01	0%	106	336	17	12
VD03	00:00:02	0%	00:00:02	0%	167	573	19	18
VD04	00:00:02	0%	00:00:02	0%	394	1580	25	32
VD05	00:00:02	0%	00:00:15	0%	447	1548	34	36
VD06	00:00:01	0%	00:00:02	0%	655	2380	37	48
VD07	00:00:13	0%	00:00:13	0%	2571	11352	55	108
VD08	01:20:31	0%	08:02:02	Máx Tempo Permitido	7531	35864	73	192
VD09	00:01:43	0%	00:01:17	0,006025%	4659	21216	67	144
VD10	08:02:07	Máx Tempo Permitido	08:02:03	Máx Tempo Permitido	17103	98480	91	300
VD11	08:02:10	Máx Tempo Permitido	08:02:01	Máx Tempo Permitido	35175	199440	109	432
VD10*	08:00:00	31,60%	08:00:00	27,99%	17103	98480	91	300
VD11*	08:00:00	44,30%	08:00:00	41,69%	35175	199440	109	432

** Instância com restrição de 28800 segundos para encontro da solução.*

Para a Tabela 5 as seguintes segregações podem ser feitas: 1- a tabela encontra-se divida entre resultados fixos relativos às instâncias; 2- resultados variantes do solucionador utilizado. Entre os solucionadores testados têm-se o GUROBI e o CPLEX. Como pode-se apercebe-se pela Tabela 5, o solucionador GUROBI para instâncias entre VD01 e VD07 foi relativamente melhor no tocante ao tempo computacional. Para a instância VD09, o CPLEX apresentou um tempo computacional menor, no entanto, o resultado foi obtido com um GAP (desvio da resposta) maior que zero. Para as seguintes instâncias (VD10 e VD11), no qual possuem dez ou mais tarefas, o tempo regulamentado como máximo pelo NEOS SERVER foi ultrapassado. Para encontrar uma resposta mesmo assim (denominada solução incumbente), estabeleceu-se um tempo máximo (tempo no qual o solucionador para de procurar resposta) de 8 horas (288000 segundos), e foi possível observar certa similaridade no GAP para ambos solucionadores. Em destaque na Tabela 5, a instância VD08 foi o principal ponto de diferença entre os solucionadores, sendo a mesma destacada na Tabela 5. Para esta instância, o GUROBI necessitou aproximadamente de 1 hora e 20 minutos, enquanto para o CPLEX o máximo de tempo determinado pelo NEOS SERVER foi excedido. Para comprovar essa observação, mais *dez experimentos* foram realizados com o CPLEX e a instância VD08, no qual *todos* excederam o máximo de tempo. Como última observação relevante, fazendo uma junção entra a Tabela 4 e Tabela 5, a partir das instâncias VD08 e VD09, pode-se notar-se que a diminuição da possibilidade do número de lotes influencia fortemente no tempo computacional.

5.2 Relação: estudo de caso x modelo matemático

Após os testes efetuados com as instâncias inicias fica claro que o modelo matemático proposto só comporta instâncias pequenas. A Figura 18 representa o comportamento no quesito tempo para cada uma das instâncias, com o adento da quantidade de restrições no passar das instâncias.

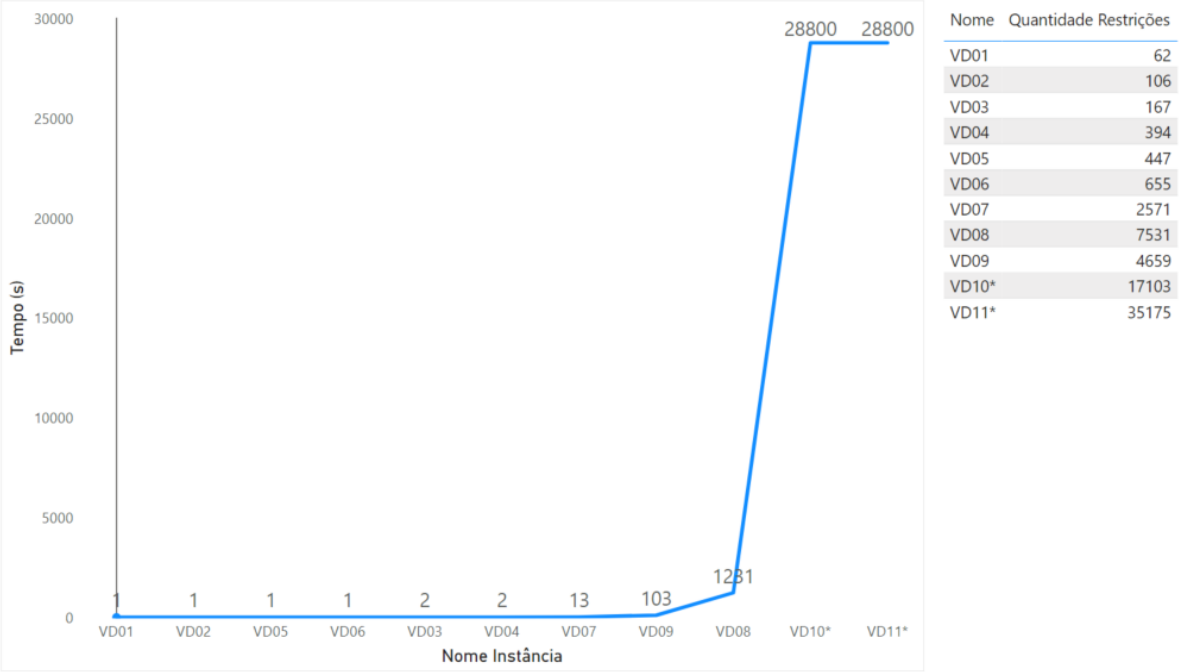


Figura 18 – Tempo para solução e quantidade de restrições por instância

A partir de instâncias com dez tarefas, em consonância com a Figura 18, pode ver-se que o modelo não é capaz fornecer uma resposta em 8 horas (28800 segundos) e, portanto, a respetiva aplicação em alguns cenários pode ser completamente inviável. Seja por não fornecer uma resposta ou fornecer uma resposta (solução incumbente) de baixa qualidade. Vale mais uma vez lembrar que as instâncias inicias foram simplificadas para ser possível a elaboração da Tabela 4 e facilitar a ilustração para o leitor. As instâncias, mesmo com menos de dez tarefas, poderiam ser mais complexas com um maior número de máquinas e divergência de famílias por máquinas, crescendo horizontalmente o modelo da Tabela 4.

Mesmo que a partir de um determinado número de tarefas, máquinas ou famílias o modelo mostre-se inviável, pode existir cenários que onde o número de tarefas, máquinas ou famílias seja baixo, e, portanto, o modelo seja uma solução (ótima) para solucionar o problema de escalonamento. Dado o impasse, é necessário verificar o histórico das ocorrências na empresa estudada, especificamente no estágio produtivo da pintura. Assim, antes de mais, como mencionado que diversos fluxos existem na organização (Capítulo 3 – página 53), deve-se ter em conta os principais fluxos existentes e observar

quais estágios antecedem o estágio da pintura nestes fluxos. Em outras palavras, ter em consideração o estágio que *origina* a procura evidenciada no estágio da pintura. A Figura 19 sumariza a quantidade de processamento de produtos para cada um dos fluxos vistos na imagem (eixo Y).

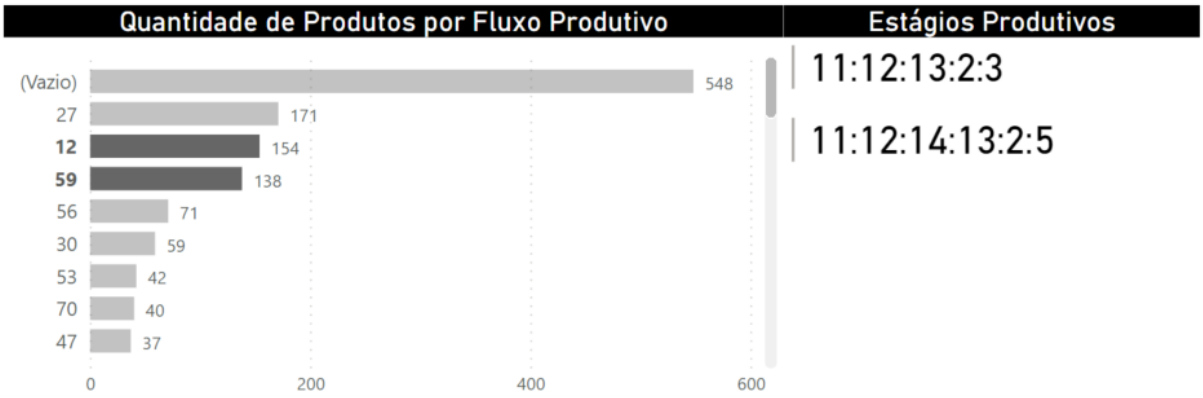


Figura 19 – Contagem por fluxo produtivo

Pela Figura 19 é visto que 548 processos para fabrico de produtos não tem um fluxo definido. Logo, considerar esses fluxos seria uma ação de incerteza para o respetivo estudo, sendo estes, por opção, excluídos do estudo. A maior ocorrência está no fluxo 27, porém, este reside apenas no estágio 3 (Acabamento) – não ocorrendo necessariamente um processamento (nem entrada no estágio 2), sendo este também excluído do estudo. Após as duas maiores incidências, têm-se os fluxos 12 e 59 (destacados na Figura 19), com o número de processamento similares. Os dois fluxos são descritos, respetivamente o fluxo 12 e 59, no lado direito da Figura 19, compondo assim os estágios 11-12-13-2-3 para o fluxo 12 e 12-14-13-2-5 para o fluxo 59. Sendo também possível observar que para ambos os fluxos o estágio que antecede o estágio da pintura é o estágio 13 (trabalho manual). Assim, para saber a real necessidade, e poder, ou não, empregar o modelo exato no estágio da pintura, deve-se avaliar a quantidade de kanbans que saem do estágio 13 e se direcionam para o estágio da pintura. Para tal visualização, a Figura 20 sintetiza no período de março de 2020 e janeiro de 2021 quanto a quantidade de kanbans que processados no estágio 13, portanto, que seguiram o fluxo produtivo para o estágio da pintura.

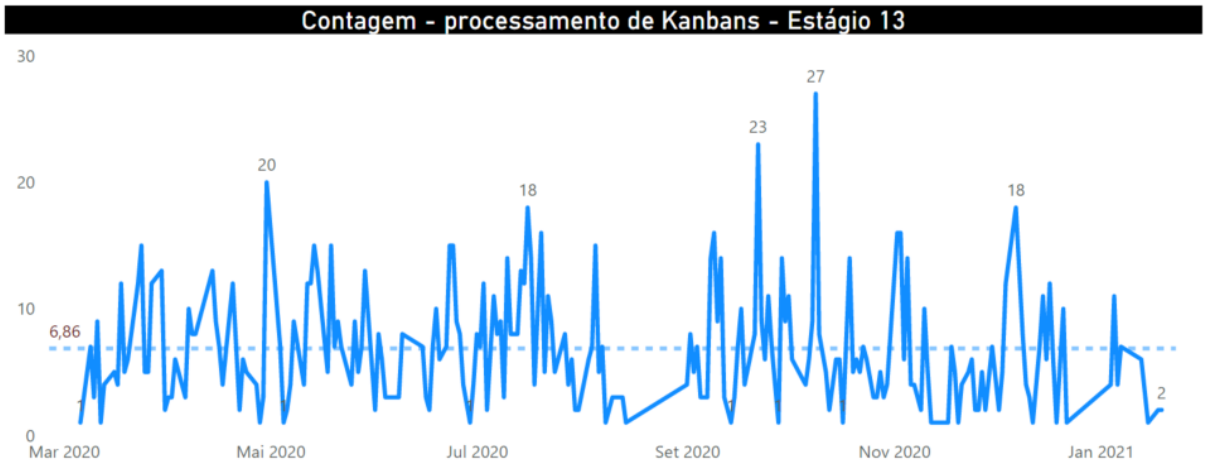


Figura 20 – Contagens de kanbans estágio 13

Pode-se, pela Figura 20, observar a existência de uma média (linha constante no eixo y) de 6,86 saídas de kanbans. Assim, usando apenas a média como fator de decisão, o modelo exato deveria ser empregado, seja que para as instâncias testadas (n = 8) o modelo ainda fornece resposta ótima em tempo aceitável. Porém, também pode ser visualizado diversos valores *acima* de 10 tarefas e um valor máximo de 27 saídas. Sendo, portanto, necessário recorrer a um método mais rápido e eficiente computacionalmente para encontro de um escalonamento. Em outra palavras, a utilização de uma heurística (ou qualquer modelo aproximado) é completamente apropriado, devido a Figura 20 apresentar diversos valores acima de 10 tarefas.

5.3 Experimentos Computacionais – NEH-Modificada

Para avaliação da NEH-Modificada realizou-se um estudo do desempenho, nomeadamente o tempo computacional para encontro da solução, da heurística NEH-modificada (explicada no Capítulo 4). O estudo deu-se para instâncias de tamanho igual a instância VD11 (maior instância avaliado sob a ótica do modelo matemático), possibilitando de certa forma comparar com o modelo matemático e instâncias de grandes tamanhos. O resultado do tempo computacional da NEH-Modificada foi esquematizado na Tabela 6, enquanto a Figura 21 ilustrou graficamente os resultados da Tabela 6. Mais, o estudo deu-se com desenvolvimento do algoritmo na linguagem computacional Python em sua versão 3.9.1, fazendo, principalmente, uso da biblioteca Numpy. A máquina utilizada para os experimentos foi um Intel-Core i7 2.80GHz 16GB RAM

Tabela 6 – Resultados NEH-Modificada

Tarefas (n)	Máquinas (m)	Tempo (s)
12	3	0,00192118
25	3	0,00299668
27	3	0,00300217
50	3	0,00499153
75	3	0,00501299
150	3	0,00904441
300	3	0,00992346
500	3	0,01304746
500	4	0,01440692
1000	4	0,02590775

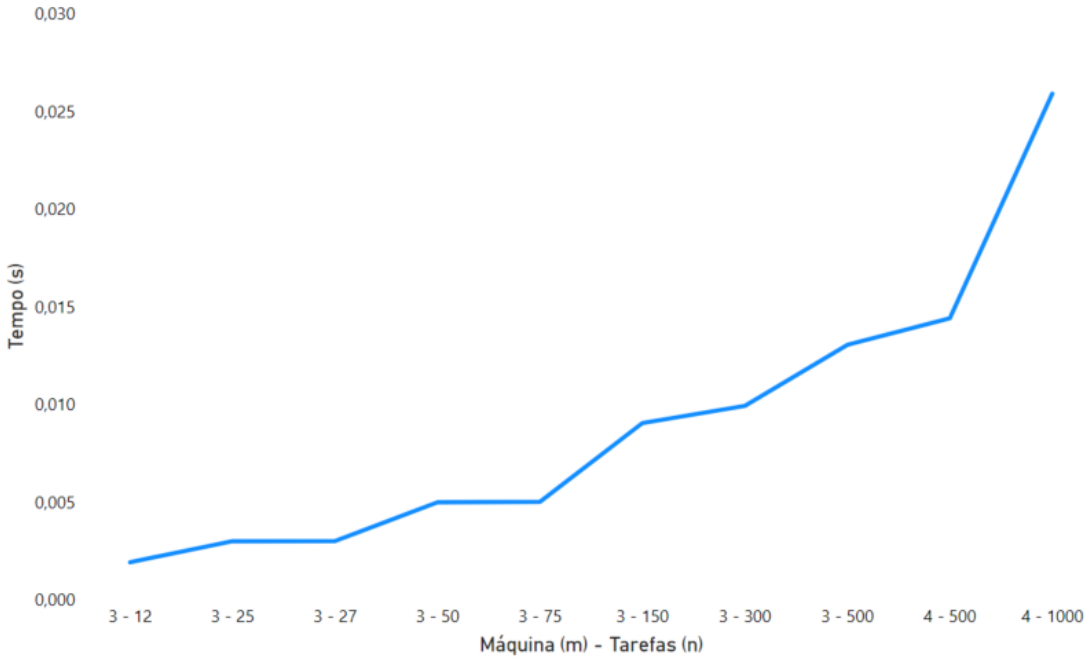


Figura 21 – Tempo de solução por instâncias NEH-Modificada

Ao avaliar os resultados da Tabela 6, pode-se perceber que a heurística apresentou tempos computacional na ordem 10^{-3} e 10^{-2} segundos, sendo, portanto, aplicacional no tocante do tempo necessário para o encontro de uma solução. Para a Figura 21, nota-se relevância visual ao adicionar uma máquina na instância com 500 tarefas, evidenciando que um aumento das máquinas pode ser um estudo relevante, devido o impacto em segundos no acréscimo de uma máquina no problema.

Os resultados da Tabela 6 e Figura 21 fundamental uma questão importante do estudo. Inicialmente, deve-se destacar a potencialidade do uso de modelo aproximados por motivo do baixo custo computacional para encontro da solução. No trabalho de Koulamas (1998) a NEH Heurística em sua forma original necessitou de $4\,x\,10^{-2}$ segundos para obter uma solução em uma instância de 5

máquinas e 200 tarefas. No presente estudo, um modelo com maiores adaptações e peculiaridades consome menos de $1\ x\ 10^{-1}$ segundos para obter uma solução em uma instância de 3 máquinas e 300 tarefas. Evidenciando que, passado 20 anos os modelos aproximados melhoraram sua performance (possivelmente consoante a evolução tecnologia) e continuam a consumir pouco recurso temporal para obter soluções (para o problema de escalonamento *Flow shop*) independente do fato do tamanho das instâncias.

Após explicação da NEH-modificada e respetiva adequação quanto ao algoritmo, fica claro que uma comparação direta com o modelo matemático só seria possível com adaptação do mesmo, devido fatores como secagem não foram incluídas na heurística. Assim, definindo um *gap* da resposta ótima e resposta da heurística como a percentagem sob a resposta ótima que resulta da diferença entre o valor da heurística e modelo matemático (Equação 20), a Tabela 7 concentrou tais resultados de forma a compará-los. Vale a ressalva que as instâncias foram nomeadas igualmente, com quantidade de tarefas, máquinas e famílias iguais, porém, nestas existem as modificações citadas acima (exclusão do tempo de secagem).

$$Gap\ (\%) = \frac{Valor\ Apriximado - Valor\ ótimo}{Valor\ ótimo}\ x\ 100 \tag{20}$$

Tabela 7 – Modelo Exato vs Modelo Heurístico

Instância Nome	MILP (F.O)	NEH (F.O)	GAP(%)
VD01	139	139	0%
VD02	170	170	0%
VD03	179	182	2%
VD04	104	129	24%
VD05	293	293	0%
VD06	266	293	10%
VD07	231	233	1%
VD08	260	279	7%
VD09	260	279	7%
VD10*	315	351	11%
VD11*	412	402	-2%

** Instâncias cujo os resultados do modelo exato representam soluções incumbentes*

A partir da Tabela 7, pode-se notar-se que o maior desvio visto foi de 24% (instância VD04). Mais, ocorreu para três instância resultados iguais ao proposto pelo modelo exato. Outro ponto de extrema relevância é dado por existir uma situação onde a heurística superou o resultado encontrado pelo modelo

matemático (instância VD11). Para esta instância e instância VD10 foi preciso, assim como visto na Tabela 5, recorrer ao limite de tempo do NEOS SERVER, obtendo-se, portanto, as respetivas soluções incumbentes (ou seja, a melhor solução encontrada até o momento em que o processo de otimização é interrompido). Obviamente, estas soluções podem não ser ainda a solução ótima (no caso da instância VD11 pode-se afirmar que não é a solução ótima), justificando um resultado de pior qualidade por parte do método exato, em outra perspetiva, justificando o fato do modelo heurístico apresentar uma resposta de melhor qualidade.

A comparação descrita na Tabela 7 serve também para expor a tendência para o problema estudado ser *Np-Hard*. O abrupto crescimento exponencial do tempo de solução entre as instâncias VD09 e VD10 fortalece a teoria à custa deste repentino de crescimento ser característico da classe *Np-Hard*. Logo, pondo em causa que o uso dos modelos aproximados, em paralelismo com um estudo que sustenta a qualidade do modelo, deve ser sempre proposto para contornar a ineficiência e ineficácia dos modelos exatos.

5.4 Enquadramento – *Rolling Kanban*

Nesta secção abortou o enquadramento, de maneira descritiva, da ferramenta visual *Rolling Kanban*. Para tal, considerou-se a instância (sem tempo de secagem) VD08 (2x). Portanto, a situação descritiva conta com 16 tarefas com planeamento produtivo para 1 semana. O quadro *Rolling Kanban* idealizado conta com 5 segmentados desse horizonte (5 dias), descritos como “A” “B” “C” “D” e “E”. Todos os valores para tempos nas instâncias são considerados em minutos.

Inicialmente apenas a instância VD08 se encontra para processamento no início do período genérico idealizado (Figura 22). Assim, para início, 8 tarefas devem ser escalonadas. De acordo com o que foi visto, para essa instância o modelo exato pode ser utilizado. Como resposta obteve um escalonamento com função objetivo de 260 (minutos) e última família processada a família 1.

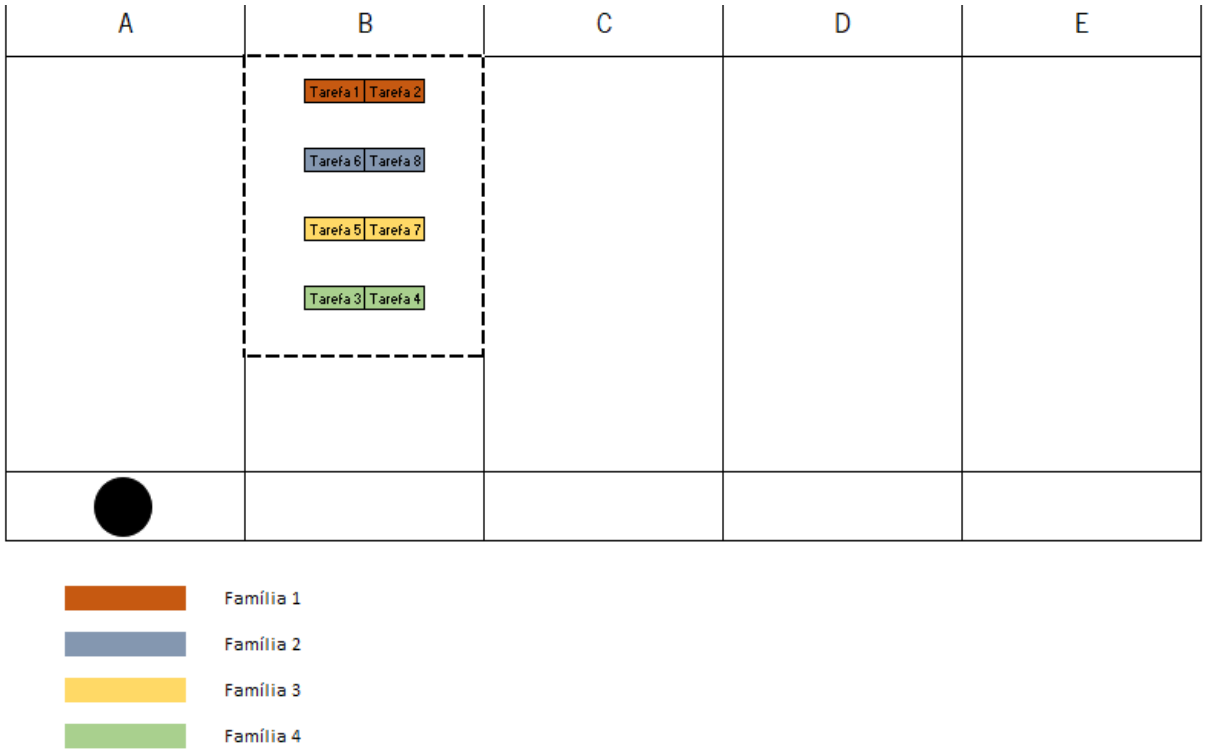


Figura 22 – Exemplo *Rolling Kanban*

Face a situação simulada, têm-se que durante esse primeiro processamento mais 8 tarefas idênticas chegaram dos estágios predecessores (Figura 23). A primeira resposta que deve-se ter é se essas 8 tarefas podem ser feitas no restante do horizonte produtivo descrito. Se uma semana é composta por 2400 minutos, obviamente que é possível o processamento dessas 8 tarefas. No entanto, nota-se ao processar novamente essas 8 tarefas, a somatória decorrida será no mínimo 520 minutos, logo, mais que a secção "B" permite (1 dia produtivo – 480 minutos). Nesse caso a sequência deve ser seguida na mesma e continuada no dia seguinte. Um ponto importante aqui é perceber que a família 1 finalizou a primeira produção, assim existirá um *setup* a acontecer, pode-se nesses casos considerar então que $s_{0f1} > 0$ e a família 0 é a família 1 na máquina 1. Neste caso específico caso a sequência se mantém, com o adento que a função objetivo foi de 290 devido o *setup* mencionado ($s_{0f1} = 30$). Assim, é possível observar que este segundo sequenciamento deverá ser finalizado no primeiro dia analisado e continuado no dia posterior. Consequentemente pelo que consiste o *Rolling Kanban*, o marcador irá para "C", a coluna "B" fica vazia (no início do dia) e após um total de 550 minutos ("A" + parte de "B") uma nova resposta pelo modelo deve ser executada.

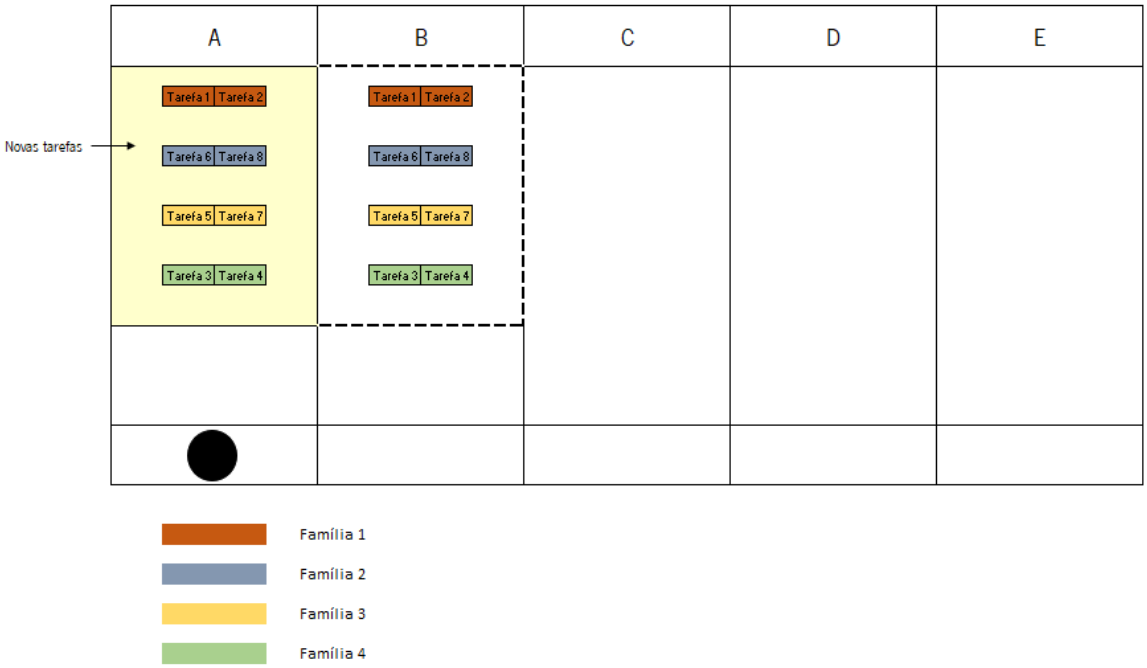


Figura 23 – Escala de preferência entre duas ordens a produzir (estágio – pintura)

Finalizando a adaptação do modelo para a ferramenta de controlo visual, caso mais de 10 trabalhos necessitem ser sequenciados, deve-se ser feito uso do método heurístico, e, caso visualize a possibilidade de cumprimento no horizonte temporal estabelecido (uma semana) os "novos" trabalhos devem ser retirados do modelo respeitando aqueles que a mais tempo estão à espera do respetivo processamento.

6. CONCLUSÕES

Produção *Lean*, Sistema Kanban, Kanban são filosofias vastamente implementados pelas organizações nos tempos atuais. Visando a redução do desperdício e sendo capaz de abordar conceitos qualitativos e quantitativos durante a respetiva utilização, o universo de estudo para estes temas pode ser classificado com atual e em constante desenvolvimento. Entretanto, como visto neste estudo, o *Rolling Kanban* é uma poderosa ferramenta para estas áreas de estudos, mas que pouco foi comentada em contexto académico. Partindo deste ponto, a presente dissertação cumpre com um dos seus objetivos que consiste em trazer mais uma possibilidade de implementação, mais, exemplifica a forma mais básica de utilização da ferramenta. Todavia, as ferramentas *lean* tendem a serem implementadas em contextos cada vez mais complexos. No caso de estudo trazido pelo trabalho, um dilema existe devido a empresa em questão usar do Sistema Kanban, porém um dos estágios não pode seguir à risca o Sistema idealizado na companhia Toyota. Assim, o trabalho desenvolveu como portar-se diante da adaptação, e como fazer uso da ferramenta.

Não só para o estudo de caso preocupou-se o presente trabalho. É possível perceber que uma extensa revisão da literatura foi realizada no tocante a Produção *Lean*, Sistema Kanban e respetivas ferramentas de controlo visual. Promovendo, portanto, outras possibilidades de replicação para um tema que visivelmente colabora para controlo e efetividade produtiva.

Do mesmo sentido, porém, para Escalonamento, uma extensa revisão dos principais temas e diferentes situações foi feita. Se preocupando principalmente com o ambiente *Flow shop*, o estudo trouxe importantes conceitos que ajudam a área de Escalonamento na tentativa de maior utilização prática. Abordou-se também exemplos de métodos para resolução de tais problemas.

Para os métodos de resolução, na opinião do autor, este ponto consiste na maior contribuição científica do trabalho. Especialmente para o modelo de programação linear mista, já que o desenvolvimento do modelo final engloba temáticas que podem ser facilmente vistas em outras situações. Mais, a adaptação do modelo trouxe à tona importantes conceitos debatidos somente em um único trabalho anterior, e que desde então não foram desenvolvidos por outros autores.

Outro ponto a destacar foi provar por dados reais a necessidades em certos casos de modelos aproximados. A NEH-Modificada foi desenvolvida neste caso especial e entra para lista de casos de extensão do modelo original. A partir desta extensão, o objetivo geral foi atingido, quando, modelo exato e/ou aproximado enquadraram-se na ferramenta *Rolling Kanban*. A união da ferramenta e modelo foi

exemplificado a partir de uma situação simples e hipotética, mas que serve de oportunidade de estudos futuros, seja com utilização de instâncias reais, seja para um estudo mais extenso quanto ao número de instâncias. Com a importante ressalva que o estudo possibilitou ultrapassar algumas dificuldades vistas em trabalhos anteriores com o tópico *Rolling Kanban*, como a possibilidade de ultrapassar grandes tempos gastos para escolha do escalonamento.

Mais uma oportunidade de estudo futuro pode ser relada para o modelo exato/modelo aproximado. Primeiramente uma junção de uma heurística que contemple a situação descrita na Figura 14 e Figura 15 com o modelo exato proposto, e, portanto, atribuindo todos os aspetos reais do estudo de caso. Como também a adaptação do modelo para metaheurística, promovendo a possibilidade de estudos comparativos com o modelo proposto em Shen e Gupta (2018), fazendo uso das estruturas de vizinhança propostas em Shen e Gupta (2018) e Shen et al. (2014). Logo, surge-se mais uma oportunidade de estudo para novas elaborações de vizinhança para escalonamento de famílias de produtos, produção por lotes, sequência dependentes e lotes inconsistentes.

Relativamente ao modelo matemático, a partir dos experimentos computacionais pôde-se notar que o problema estudado tende a ser *NP-Hard*, devido respetiva ineficiência por parte do modelo em encontrar uma solução ótima, para o contexto de instâncias relativamente pequenas (10 tarefas e 3 máquinas), em 28800 segundos. A partir deste fato, mostrou-se como algoritmos aproximados, com a enfase de constante evolução tecnológica, devem ser utilizados. Justificando a afirmação pela elaboração do algoritmo aproximado do presente estudo em Python 3.9.1, com encontro da solução para uma instância com 1000 tarefas e 4 máquinas em aproximadamente $2{,}6\ x\ 10^{-2}$ segundos.

REFERÊNCIAS BIBLIOGRÁFICAS

Abdul-Nour, G., Lambert, S., & Drolet, J. (1998). Adaptation of JIT phylosophy and kanban technique to a small-sized manufacturing firm; a project management approach. *Computers and Industrial Engineering*. https://doi.org/10.1016/s0360-8352(98)00123-5

Aggarwal, S. C. (1985). Mrp, Jit, Opt, Fms? *Harvard Business Review*.

Allahverdi, A. (2015). The third comprehensive survey on scheduling problems with setup times/costs. In *European Journal of Operational Research*. https://doi.org/10.1016/j.ejor.2015.04.004

Allahverdi, A., Gupta, J. N. D., & Aldowaisan, T. (1999). A review of scheduling research involving setup considerations. *Omega*. https://doi.org/10.1016/S0305-0483(98)00042-5

Allahverdi, A., Ng, C. T., Cheng, T. C. E., & Kovalyov, M. Y. (2008). A survey of scheduling problems with setup times or costs. *European Journal of Operational Research*. https://doi.org/10.1016/j.ejor.2006.06.060

Allahverdi, A., & Soroush, H. M. (2008). The significance of reducing setup times/setup costs. *European Journal of Operational Research*. https://doi.org/10.1016/j.ejor.2006.09.010

Baker, K. R., & Trietsch, D. (2009). Principles of Sequencing and Scheduling. In *Principles of Sequencing and Scheduling*. https://doi.org/10.1002/9780470451793

Baptiste, P. (2000). Batching identical jobs. *Mathematical Methods of Operations Research*. https://doi.org/10.1007/s001860000088

Blazewicz, J., Ecker, K., Pesch, E., Schmidt, G., & Weglarz, J. (2007). Handbook on Scheduling From Theory to Applications. In *Knowledge Management*.

Boyer, M. (2004). II Rolling Kanban (In Italian). In *Quaderni di Management,* Issue 5. Milan E.G.V Edizioni Srl.

Braglia, M., Castellano, D., Gallo, M., & Romagnoli, G. (2019). A visual planning solution to streamline the processes of hybrid cross-dockings. *Production Planning and Control*. https://doi.org/10.1080/09537287.2018.1520317

Braglia, M., Gabbrielli, R., & Marrazzini, L. (2020). Rolling Kanban: a new visual tool to schedule family batch manufacturing processes with kanban. *International Journal of Production Research*. https://doi.org/10.1080/00207543.2019.1639224

Brucker, P. (2007). Scheduling algorithms. In *Scheduling Algorithms*. https://doi.org/10.1007/978-3-540-69516-5

Brucker, P., Gladky, A., Hoogeveen, H., Kovalyov, M. Y., Potts, C. N., Tautenhahn, T., & Van De Velde, S. L. (1998). Scheduling a batching machine. *Journal of Scheduling*. https://doi.org/10.1002/(SICI)1099-1425(199806)1:1<31::AID-JOS4>3.0.CO;2-R

Campbell, H. G., Dudek, R. A., & Smith, M. L. (1970). A Heuristic Algorithm for the n Job, m Machine Sequencing Problem . *Management Science*. https://doi.org/10.1287/mnsc.16.10.b630

Celano, G., Costa, A., & Fichera, S. (2010). Constrained scheduling of the inspection activities on semiconductor wafers grouped in families with sequence-dependent set-up times. *International Journal of Advanced Manufacturing Technology*. https://doi.org/10.1007/s00170-009-2112-x

Chaussé, S., Landry, S., Pasin, F., & Fortier, S. (2000). Anatomy of a kanban: A case study. *Production and Inventory Management Journal*.

Cheng, C. Y., Pourhejazy, P., Ying, K. C., & Liao, Y. H. (2021). New benchmark algorithms for No-wait *Flowshop* Group Scheduling Problem with Sequence-Dependent Setup Times. *Applied Soft Computing*. https://doi.org/10.1016/j.asoc.2021.107705

Cheng, T. C.E., Lin, B. M. T., & Toker, A. (2000). Makespan minimization in the two-machine *Flowshop* batch scheduling problem. *Naval Research Logistics*. https://doi.org/10.1002/(SICI)1520-6750(200003)47:2<128::AID-NAV4>3.0.CO;2-#

Cheng, T. C.Edwin, Gupta, J. N. D., & Wang, G. (2000). A review of *Flowshop* scheduling research with setup times. In *Production and Operations Management*. https://doi.org/10.1111/j.1937-5956.2000.tb00137.x

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

Dong, X., Huang, H., & Chen, P. (2008). An improved NEH-based heuristic for the permutation *Flowshop* problem. *Computers and Operations Research*. https://doi.org/10.1016/j.cor.2007.05.005

Festa, P. (2014). A brief introduction to exact, approximation, and heuristic algorithms for solving hard combinatorial optimization problems. *International Conference on Transparent Optical Networks*. https://doi.org/10.1109/ICTON.2014.6876285

Fleischmann, B., Meyr, H., & Wagner, M. (2008). Advanced planning. In *Supply Chain Management and Advanced Planning (Fourth Edition): Concepts, Models, Software, and Case Studies*. https://doi.org/10.1007/978-3-540-74512-9_5

Floudas, C. A., & Lin, X. (2005). Mixed integer linear programming in process scheduling: Modeling, algorithms, and applications. *Annals of Operations Research*. https://doi.org/10.1007/s10479-005-3446-x

Framinan, J. M., Leisten, R., & Ruiz García, R. (2014). Manufacturing Scheduling Systems. In *Manufacturing Scheduling Systems*. https://doi.org/10.1007/978-1-4471-6272-8

Framinan, J. M., & Ruiz, R. (2010). Architecture of manufacturing scheduling systems: Literature review and an integrated proposal. In *European Journal of Operational Research*. https://doi.org/10.1016/j.ejor.2009.09.026

Garey, M. R., & Johnson, D. S. (1979). Computers and Intractability: A Guide to the Theory of NP-Completeness (Series of Books in the Mathematical Sciences). *Computers and Intractability*.

Garey, M. R., Johnson, D. S., & Sethi, R. (1976). COMPLEXITY OF *FLOWSHOP* AND JOB SHOP SCHEDULING. *Mathematics of Operations Research*. https://doi.org/10.1287/moor.1.2.117

Gelogullari, C. A., & Logendran, R. (2010). Group-scheduling problems in electronics manufacturing. *Journal of Scheduling*. https://doi.org/10.1007/s10951-009-0147-3

González-Neira, E. M., Montoya-Torres, J. R., & Barrera, D. (2017). Flow-shop scheduling problem under uncertainties: Review and trends. In *International Journal of Industrial Engineering Computations*. https://doi.org/10.5267/j.ijiec.2017.2.001

Graham, R. L., Lawler, E. L., Lenstra, J. K., & Kan, A. H. G. R. (1979). Optimization and approximation in deterministic sequencing and scheduling: A survey. *Annals of Discrete Mathematics*. https://doi.org/10.1016/S0167-5060(08)70356-X

Gravel, M., & Price, W. L. (1988). Using the kanban in a Job shop environment. *International Journal of Production Research*. https://doi.org/10.1080/00207548808947921

Gross, J., & McInnis, K. (2003). Kanban made simple: demystifying and applying Toyota's legendary manufacturing process. In *New york: Amacom*.

Gupta, J. N. D. (1979). A review of *Flowshop* scheduling research. In *Disaggregation*. https://doi.org/10.1007/978-94-015-7636-9_23

Hendrick, T. E. (1988). "FAKE PULL" IN A KANBAN ENVIRONMENT: ACCEPTABLE TRADE-OFF OR VIOLATION OF PRINCIPLE? . *Prod Invent Manage J*.

Henneberg, M., & Neufeld, J. S. (2016). A constructive algorithm and a simulated annealing approach for solving *Flowshop* problems with missing operations. *International Journal of Production Research*. https://doi.org/10.1080/00207543.2015.1082670

Hiller, F. S., & Lieberman, G. J. (2014). Introduction - Operations Research. In *McGraw-Hill Higher Education*.

Hirano, H. (2009). JIT Implementation Manual Vol 3: Flow Manufacturing – Multi-Process Operations and Kanban. In *Vol. 1*.

Irani, S. A., Subramanian, S., & Allam, Y. S. (2007). Introduction to Cellular Manufacturing Systems. In *Handbook of Cellular Manufacturing Systems*. https://doi.org/10.1002/9780470172476.ch

Isenberg, M.-A., & Scholz-Reiter, B. (2013). *The Multiple Batch Processing Machine Problem with Stage Specific Incompatible Job Families*. https://doi.org/10.1007/978-3-642-35966-8_9

ITZWOOD Technological Solutions. (2021). *About Us*. https://www.itzwood.com/

Jin, Q., Pan, X. T., & Zhang, Z. (2011). On solving JIT production problems for small batch orders based on E-kanban visualization. *Proceedings - 3rd International Conference on Measuring Technology and Mechatronics Automation, ICMTMA 2011*. https://doi.org/10.1109/ICMTMA.2011.473

Johnson, S. M. (1954). Optimal two- and three-stage production schedules with setup times included. *Naval Research Logistics Quarterly*. https://doi.org/10.1002/nav.3800010110

Jr, E. F. S., & Tseng, F. T. (1990). On the Srikar-Ghosh MILP model for the iVx M SDST *Flowshop* problem. *International Journal of Production Research*. https://doi.org/10.1080/00207549008942836

Kalczynski, P. J., & Kamburowski, J. (2008). An improved NEH heuristic to minimize makespan in permutation *Flowshop*s. *Computers and Operations Research*. https://doi.org/10.1016/j.cor.2007.01.020

Kan, A. H. G. R. (1976). Machine Scheduling Problems. In *Machine Scheduling Problems*. https://doi.org/10.1007/978-1-4613-4383-7

King, P. L., & King, J. S. (2018). The Product Wheel Handbook: Creating Balanced Flow in High-Mix Process Operations. In *The Product Wheel Handbook: Creating Balanced Flow in High-Mix Process Operations*. https://doi.org/10.1201/b14568

Koulamas, C. (1998). A new constructive heuristic for the *Flowshop* scheduling problem. *European Journal of Operational Research*. https://doi.org/10.1016/S0377-2217(97)00027-1

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

Laborie, P. (2003). Algorithms for propagating resource constraints in AI planning and scheduling: Existing approaches and new results. *Artificial Intelligence*. https://doi.org/10.1016/S0004-3702(02)00362-4

Lage Junior, M., & Godinho Filho, M. (2010). Variations of the kanban system: Literature review and classification. In *International Journal of Production Economics*. https://doi.org/10.1016/j.ijpe.2010.01.009

Lawler, E. L., Lenstra, J. K., Rinnooy Kan, A. H. G., & Shmoys, D. B. (1993). Sequencing and scheduling: Algorithms and complexity. In *Handbooks in Operations Research and Management Science*. https://doi.org/10.1016/S0927-0507(05)80189-6

Lee, T. S., & Loong, Y. T. (2019). A review of scheduling problem and resolution methods in flexible *Flowshop*. In *International Journal of Industrial Engineering Computations*. https://doi.org/10.5267/j.ijiec.2018.4.001

Lenstra, J. K., Rinnooy Kan, A. H. G., & Brucker, P. (1977). Complexity of machine scheduling problems. *Annals of Discrete Mathematics*. https://doi.org/10.1016/S0167-5060(08)70743-X

Leung, J. Y. T. (2004). Handbook of scheduling: Algorithms, models, and performance analysis. In *Handbook of Scheduling: Algorithms, Models, and Performance Analysis*.

Li, D., & Lu, X. (2020). Two-machine *Flowshop* scheduling with an operator non-availability period to minimize makespan. *Journal of Combinatorial Optimization*. https://doi.org/10.1007/s10878-020-00548-6

Li, X. L., Li, Y. P., & Huang, Y. L. (2019). Heuristics and lower bound for minimizing maximum lateness on a batch processing machine with incompatible job families. *Computers and Operations Research*. https://doi.org/10.1016/j.cor.2019.02.012

Liaee, M. M., & Emmons, H. (1997). Scheduling families of jobs with setup times. *International Journal of Production Economics*. https://doi.org/10.1016/S0925-5273(96)00105-3

Lin, S. W., & Ying, K. C. (2009). Applying a hybrid simulated annealing and tabu search approach to non-permutation *Flowshop* scheduling problems. *International Journal of Production Research*. https://doi.org/10.1080/00207540701484939

Lin, Shih Wei, Ying, K. C., & Lee, Z. J. (2009). Metaheuristics for scheduling a non-permutation flowline manufacturing cell with sequence dependent family setup times. *Computers and Operations Research*. https://doi.org/10.1016/j.cor.2007.12.010

Liu, M., & Chu, C. (2012). OPTIMAL SEMI-ONLINE ALGORITHMS for m-BATCH-MACHINE *FLOWSHOP* SCHEDULING. *Discrete Mathematics, Algorithms and Applications*. https://doi.org/10.1142/S1793830912500516

Liu, W., Jin, Y., & Price, M. (2017). A new improved NEH heuristic for permutation *Flowshop* scheduling problems. *International Journal of Production Economics*. https://doi.org/10.1016/j.ijpe.2017.06.026

Lu, S., Pei, J., Liu, X., Qian, X., Mladenovic, N., & Pardalos, P. M. (2020). Less is more: variable neighborhood search for integrated production and assembly in smart manufacturing. *Journal of Scheduling*. https://doi.org/10.1007/s10951-019-00619-5

Maccarthy, B. L., & Liu, J. (1993). Addressing the gap in scheduling research: A review of optimization and heuristic methods in production scheduling. *International Journal of Production Research*. https://doi.org/10.1080/00207549308956713

Mackerron, G., Kumar, M., Kumar, V., & Esain, A. (2014). Supplier replenishment policy using e-Kanban: A framework for successful implementation. *Production Planning and Control*. https://doi.org/10.1080/09537287.2013.782950

Matin, H. N. Z., Salmasi, N., & Shahvari, O. (2017). Makespan minimization in *Flowshop* batch processing problem with different batch compositions on machines. *International Journal of Production Economics*. https://doi.org/10.1016/j.ijpe.2017.09.015

Meng, Q., & Xu, X. (2020). Solving scheduling problems for a non-permutation assembly *Flowshop*. *IEEE International Conference on Automation Science and Engineering*. https://doi.org/10.1109/CASE48305.2020.9217021

Millstein, M. A., & Martinich, J. S. (2014). Takt Time Grouping: Implementing kanban-flow manufacturing in an unbalanced, high variation cycle-time process with moving constraints. *International Journal of Production Research*. https://doi.org/10.1080/00207543.2014.910621

Mitchell, P. S., & Schonberger, R. J. (1983). Japanese Manufacturing Techniques: Nine Hidden Lessons in Simplicity. *The Academy of Management Review*. https://doi.org/10.2307/257841

Mönch, L., & Roob, S. (2018). A matheuristic framework for batch machine scheduling problems with incompatible job families and regular sum objective. *Applied Soft Computing Journal*. https://doi.org/10.1016/j.asoc.2017.10.028

Monden, Y. (2011). Toyota Production System : An Integrated Approach to Just-In-Time, 4th Edition [Internet]. In *Toyota Production System: An Integrated Approach to Just-In-Time*.

Nawaz, M., Enscore, E. E., & Ham, I. (1983). A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem. *Omega*. https://doi.org/10.1016/0305-0483(83)90088-9

NEOS SERVER Optimization Org. (2021). NEOS SERVER. https://neo-server.org/neos/

Neufeld, J. S., Gupta, J. N. D., & Buscher, U. (2016). A comprehensive review of *Flowshop* group scheduling literature. In *Computers and Operations Research*. https://doi.org/10.1016/j.cor.2015.12.006

Ng, C. T., & Kovalyov, M. Y. (2007). Batching and scheduling in a multi-machine *Flowshop*. *Journal of Scheduling*. https://doi.org/10.1007/s10951-007-0041-9

Ohno, T. (1982). How the Toyota Production System was Created. *Japanese Economic Studies*. https://doi.org/10.2753/jes1097-203x100483

Ohno, T. (1988). Toyota Production System Summary. In *Toyota Production System: Beyond Large-Scale Production*.

Otenti, S. (1992). A modified Kanban system in a semiconductor manufacturing environment. *IEEE/SEMI Advanced Semiconductor Manufacturing Conference and Workshop*. https://doi.org/10.1109/asmc.1991.167380

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

Pekarcikova, M., Trebuna, P., Kliment, M., & Rosocha, L. (2020). Material flow optimization through e-kanban system simulation. *International Journal of Simulation Modelling*. https://doi.org/10.2507/IJSIMM19-2-513

Pinedo, M. L. (2016). Scheduling: Theory, algorithms, and systems, fifth edition. In *Scheduling: Theory, Algorithms, and Systems, Fifth Edition*. https://doi.org/10.1007/978-3-319-26580-3

Potts, C. N., & Strusevich, V. A. (2009). Fifty years of scheduling: A survey of milestones. *Journal of the Operational Research Society*. https://doi.org/10.1057/jors.2009.2

Potts, C. N., & Wassenhove, L. N. (1992). Integrating scheduling with batching and lot-sizing: A algorithms

and complexity. *Journal of the Operational Research Society*. https://doi.org/10.1057/jors.1992.66

Potts, Chris N., & Kovalyov, M. Y. (2000). Scheduling with batching: a review. In *European Journal of Operational Research*. https://doi.org/10.1016/S0377-2217(99)00153-8

Potts, Chris N., Shmoys, D. B., & Williamson, D. P. (1991). Permutation vs. non-permutation *Flowshop* schedules. *Operations Research Letters*. https://doi.org/10.1016/0167-6377(91)90014-G

Pruhs, K., Sgall, J., & Torng, E. (2004). Online scheduling. In *Handbook of Scheduling: Algorithms, Models, and Performance Analysis*. https://doi.org/10.1201/9780429428890-20

Pugazhendhi, S., Thiagarajan, S., Rajendran, C., & Anantharaman, N. (2003). Performance enhancement by using non-permutation schedules in flowline-based manufacturing systems. *Computers and Industrial Engineering*. https://doi.org/10.1016/S0360-8352(02)00189-4

Rabadi, G., Msakni, M. K., Rodriguez-Velasquez, E., & Alvarez-Bermudez, W. (2019). New characteristics of optimal solutions for the two-machine *Flowshop* problem with unlimited buffers. *Journal of the Operational Research Society*. https://doi.org/10.1080/01605682.2018.1475114

Rad, S. F., Ruiz, R., & Boroojerdian, N. (2009). New high performing heuristics for minimizing makespan in permutation *Flowshop*s. *Omega*. https://doi.org/10.1016/j.omega.2007.02.002

Reza Hejazi, S., & Saghafian, S. (2005). *Flowshop*-scheduling problems with makespan criterion: A review. In *International Journal of Production Research*. https://doi.org/10.1080/0020754050056417

Ríos-Mercado, R. Z., & Bard, J. F. (2003). The *Flowshop* Scheduling Polyhedron with Setup Times. *Journal of Combinatorial Optimization*. https://doi.org/10.1023/A:1027372722187

Rossi, A., & Lanzetta, M. (2014). Native metaheuristics for non-permutation *Flowshop* scheduling. *Journal of Intelligent Manufacturing*. https://doi.org/10.1007/s10845-012-0724-8

Rossit, D., Tohmé, F., Frutos, M., Bard, J., & Broz, D. (2016). A non-permutation *Flowshop* scheduling problem with lot streaming: A mathematical model. *International Journal of Industrial Engineering Computations*. https://doi.org/10.5267/j.ijiec.2015.11.004

Rossit, Daniel A., Vásquez, Ó. C., Tohmé, F., Frutos, M., & Safe, M. D. (2021). A combinatorial analysis of the permutation and non-permutation *Flowshop* scheduling problems. *European Journal of Operational Research*. https://doi.org/10.1016/j.ejor.2019.07.055

Rossit, Daniel Alejandro, Tohmé, F., & Frutos, M. (2018). The Non-Permutation Flow-Shop scheduling problem: A literature review. In *Omega (United Kingdom)*. https://doi.org/10.1016/j.omega.2017.05.010

Rossit, Daniel Alejandro, Tohmé, F., & Frutos, M. (2019a). An Industry 4.0 approach to assembly line resequencing. *International Journal of Advanced Manufacturing Technology*. https://doi.org/10.1007/s00170-019-03804-0

Rossit, Daniel Alejandro, Tohmé, F., & Frutos, M. (2019b). Industry 4.0: Smart Scheduling. *International Journal of Production Research*. https://doi.org/10.1080/00207543.2018.1504248

Rossit, Daniel Alejandro, Toncovich, A., Rossit, D. G., & Nesmachnow, S. (2021). Solving a *Flowshop* scheduling problem with missing operations in an Industry 4.0 production environment. *Journal of Project Management*. https://doi.org/10.5267/j.jpm.2020.10.001

Rudan, J., Kersbergen, B., Van Den Boom, T., & Hangos, K. (2013). Performance analysis of MILP based

model predictive control algorithms for dynamic railway scheduling. *2013 European Control Conference, ECC 2013*. https://doi.org/10.23919/ecc.2013.6669393

Salmasi, N., Logendran, R., & Skandari, M. R. (2010). Total flow time minimization in a *Flowshop* sequence-dependent group scheduling problem. *Computers and Operations Research*. https://doi.org/10.1016/j.cor.2009.04.013

Savsar, M. (1997). Simulation analysis of a pull-push system for an electronic assembly line. *International Journal of Production Economics*. https://doi.org/10.1016/S0925-5273(97)00055-8

Schaller, J. (2012). Scheduling a permutation *Flowshop* with family setups to minimise total tardiness. *International Journal of Production Research*. https://doi.org/10.1080/00207543.2011.575094

Schaller, J. E., Gupta, J. N. D., & Vakharia, A. J. (2000). Scheduling a flowline manufacturing cell with sequence dependent family setup times. *European Journal of Operational Research*. https://doi.org/10.1016/S0377-2217(99)00387-2

Seidman, T. I., & Holloway, L. E. (2002). Stability of pull production control methods for systems with significant setups. *IEEE Transactions on Automatic Control*. https://doi.org/10.1109/TAC.2002.803531

Seidmann, A. (1988). Regenerative pull (Kanban) production control policies. *European Journal of Operational Research*. https://doi.org/10.1016/0377-2217(88)90230-5

Shen, L., & Buscher, U. (2012). Solving the serial batching problem in Job shop manufacturing systems. *European Journal of Operational Research*. https://doi.org/10.1016/j.ejor.2012.03.001

Shen, L., & Gupta, J. N. D. (2018). Family scheduling with batch availability in *Flowshop*s to minimize makespan. *Journal of Scheduling*. https://doi.org/10.1007/s10951-017-0529-x

Shen, L., Gupta, J. N. D., & Buscher, U. (2014). *Flowshop* batching and scheduling with sequence-dependent setup times. *Journal of Scheduling*. https://doi.org/10.1007/s10951-014-0369-x

Sivakumar, G. D., & Shahabudeen, P. (2009). Algorithms for the design of a multi-stage adaptive kanban system. *International Journal of Production Research*. https://doi.org/10.1080/00207540802302071

Smalley, A. (2009). Connecting Assembly with Batch Process Via Basic Pull System. *Management Science/Operation Research*.

Sohal, A. S., Keller, A. Z., & Fouad, R. H. (1989). A Review of Literature Relating to JIT. *International Journal of Operations & Production Management*. https://doi.org/10.1108/eum0000000001228

Srikar, B. N., & Ghosh, S. (1986). A milp model for the n-job, m-stage *Flowshop* with sequence dependent set-up times. *International Journal of Production Research*. https://doi.org/10.1080/00207548608919815

Stefansdottir, B., Grunow, M., & Akkerman, R. (2017). Classifying and modeling setups and cleanings in lot sizing and scheduling. *European Journal of Operational Research*. https://doi.org/10.1016/j.ejor.2017.03.023

Stoop, P. P. M., & Wiers, V. C. S. (1996). The complexity of scheduling in practice. In *International Journal of Operations and Production Management*. https://doi.org/10.1108/01443579610130682

Strusevich, V. A., & Zwaneveld, C. M. (1994). On non-permutation solutions to some two machine *Flowshop* scheduling problems. *ZOR Zeitschrift Für Operations Research Mathematical Methods of Opeartions Research*. https://doi.org/10.1007/BF01435460

Sugimori, Y., Kusunoki, K., Cho, F., & Uchikawa, S. (1977). Toyota production system and kanban system materialization of just-in-time and respect-for-human system. *International Journal of Production Research*. https://doi.org/10.1080/00207547708943149

Tang, L., & Zhao, Y. (2008). Scheduling a single semi-continuous batching machine. *Omega*. https://doi.org/10.1016/j.omega.2007.11.003

Throughput Optimization in Robotic Cells. (2007). In *Throughput Optimization in Robotic Cells*. https://doi.org/10.1007/0-387-70988-6

Tseng, F. T., & Stafford, E. F. (2001). Two MILP models for the N × M SDST *Flowshop* sequencing problem. *International Journal of Production Research*. https://doi.org/10.1080/00207540010029433

Uzsoy, R. (1995). Scheduling batch processing machines with incompatible job families. *International Journal of Production Research*. https://doi.org/10.1080/00207549508904839

Van Der Krogt, R., Geraghty, J., Salman, M. R., & Little, J. (2010). On supporting Lean methodologies using constraint-based scheduling. *Journal of Scheduling*. https://doi.org/10.1007/s10951-009-0144-6

Van Der Zee, D. J. (2013). Family based dispatching with batch availability. *International Journal of Production Research*. https://doi.org/10.1080/00207543.2012.756590

Van Veen-Dirks, P. (2005). Management control and the production environment: A review. *International Journal of Production Economics*. https://doi.org/10.1016/j.ijpe.2004.06.026

Wagner, H. M. (1959). An integer linear-programming model for machine scheduling. *Naval Research Logistics Quarterly*. https://doi.org/10.1002/nav.3800060205

Wang, H., & Hsu-Pin (Ben) Wang. (1991). Optimum number of kanbans between two adjacent workstations in a JIT system. *International Journal of Production Economics*. https://doi.org/10.1016/0925-5273(91)90093-9

Wang, J. Q., Fan, G. Q., & Liu, Z. (2020). Mixed batch scheduling on identical machines. *Journal of Scheduling*. https://doi.org/10.1007/s10951-019-00623-9

Webster, S., & Baker, K. R. (1995). Scheduling groups of jobs on a single machine. *Operations Research*. https://doi.org/10.1287/opre.43.4.692

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

Wu, C. C., Gupta, J. N. D., Cheng, S. R., Lin, B. M. T., Yip, S. H., & Lin, W. C. (2020). Robust scheduling for a two-stage assembly shop with scenario-dependent processing times. *International Journal of Production Research*. https://doi.org/10.1080/00207543.2020.1778208

Zou, Y., Wang, D., Lin, W. C., Chen, J. Y., Yu, P. W., Wu, W. H., Chao, Y. P., & Wu, C. C. (2020). Two-stage three-machine assembly scheduling problem with sum-of-processing-times-based learning effect. *Soft Computing*. https://doi.org/10.1007/s00500-019-04301-y

APÊNDICE A – PSEUDOCÓDIGO NEH-MODIFICADA

Pseudocódigo (Python) - NEH-Modificada

```
import numpy as np
import time

n = tarefas
m = maquinas
k = familias #type(k) = list
K = len(k)

def makespan(sequencia, batch_time, setups) #calculo do makespan

def batch_time(matriz_1_0_familia_tarefa, tempos_processamento) #calculo do tempo para processamento do lote
def setup_medio(setups) #media do setup da familia por maquina

a = batch_time(matriz_1_0_familia_tarefa, tempos_processamento)
b = setup_medio(setups)
c = a+b
a1 = batch_time(matriz_1_0_familia_tarefa_mod, tempos_processamento) #matriz modificada (igual pagina 62 do
trabalho)

#encontrar as 2 familias com maior tempo efetivo de processamento
schedule = []
for i in range(K):
    schedule.append(0)
for f in range(K):
    schedule[f] = sum(c[:,f])
j=[]
schedule1 = schedule.copy()
schedule1.sort(reverse = True)
for i in schedule1:
    rodada = 0
    for p, k in enumerate(schedule):
        if i == k:
            if rodada ==0:
                j.append(p)
                rodada = rodada + 1
            else:
                rodada =0
                continue
# se empate
```

```
while sum(j)<sum(k):
    for p, i in enumerate(j):
        b= j.copy()
        b.remove(i)
        if i in b:
            j[p] = i+1

#further
delta = j.copy()
delta = [j[0], j[1]]

#first schedule:

first_make = makespan(delta, a1, Setups)
if makespan(delta[::-1], a1, Setups_fam) < first_make:
    delta = delta[::-1]

fase = 2
while fase < K:
    o = j[fase]
    for i in range(fase+1):
        if i ==0:
            delta.insert(i, o)
            delta1 = delta.copy()
            makespan(delta, a1, Setups_fam)
            resp = [delta1, makespan(delta, a1, Setups_fam)]
            delta.remove(o)
        else:
            delta.insert(i, o)
            delta1 = delta.copy()
            if makespan(delta, a1, Setups_fam) < resp[1]:
                resp = [delta1, makespan(delta, a1, Setups_fam)]
                delta.remove(o)
            else:
                delta.remove(o)
    delta = resp[0]
    fase = fase+1
print(resp)


print('Tempo computacional: {} segundos'.format(time.time() - t1))
```

APÊNDICE B – FORMULAÇÃO AMPL

```
param n_jobs;
set Jobs := {1..n_jobs};
param n_machines;
set M:={1..n_machines};
param n_familias{k in M};
set F{k in M};
set Fo{k in M};
set J{k in M, g in F[k]};
param beta{k in M, g in F[k], j in Jobs};
param s{k in M, f in Fo[k], g in F[k]};
param time{j in Jobs, k in M};
param sec{j in Jobs, k in M};
param n_batch;
param bigM := 9999;
set B := 1..n_batch;
var x{j in Jobs, k in M, b in B}, binary;
var T{k in M, b in B}, >=0;
var C{j in Jobs, k in M}, >=0;
var r{j in Jobs, k in M}, >=0;
var Cmax, >=0;

minimize obj: Cmax;
s.t. rest1{j in Jobs, k in M}: sum{b in B}x[j,k,b]=1;
s.t. rest2{i in Jobs, j in Jobs, k in M, b in B,f in F[k]}:  x[j,k,b]<= beta[k,f,j] - beta[k,f,i] - x[i,k,b] + 2;
s.t. rest22{i in Jobs, j in Jobs, k in M, b in B,f in F[k]}: x[j,k,b]<= beta[k,f,i] - beta[k,f,j] - x[i,k,b] + 2;
s.t. rest3{k in M}: sum{j in Jobs}x[j,k,1] >=1;
s.t. rest4{k in M, g in F[k], j in J[k,g]}: T[k,1] >= s[k,0,g]*x[j,k,1];
s.t. rest5{b in 1..n_batch-1, c in 2..n_batch, k in M, g in F[k], f in F[k], i in J[k,g], j in J[k,f]: c>b and i<>j}: T[k,c]>= T[k,b] + s[k,f,g]*x[j,k,b]
+ sum{j1 in J[k,f]}x[j1,k,b]*time[j1,k] - (1 - x[i,k,c])*bigM;
s.t. rest6{j in Jobs, k in M, b in B}: C[j,k] >= T[k,b] + sum{i in Jobs}time[i,k]*x[i,k,b] - (1 - x[j,k,b])*bigM;
s.t. rest11{j in Jobs,k in M, b in B}: T[k,b] + sum{i in Jobs}x[i,k,b]*time[i,k] >= C[j,k] - (1 - x[j,k,b])*bigM;
s.t. rest7{j in Jobs, k in 1..n_machines-1, b in B}: T[k+1,b] >= C[j,k] - (1 - x[j,k+1,b])*bigM;
s.t. rest9{j in Jobs, k in 1..n_machines-1}: r[j,k+1] = C[j,k]+sec[j,k];
s.t. rest10{j in Jobs, k in M, b in B}: T[k,b]>=r[j,k] - (1 - x[j,k,b])*bigM;
s.t. rest8{j in Jobs}: Cmax >= C[j,n_machines] + sec[j,n_machines];
```