Universidade de São Paulo
Escola de Engenharia de São Carlos
Departamento de Engenharia de Produção
Programa de Pós-Graduação em Engenharia de Produção

**Caio Paziani Tomazella**

# Novos limitantes inferiores para o método *branch-and-bound* na solução de problemas *flowshop* permutacional

São Carlos - SP
2019

**Caio Paziani Tomazella**

# Novos limitantes inferiores para o método *branch-and-bound* na solução de problemas *flowshop* permutacional

Texto apresentado ao Programa de Engenharia de Produção da Escola de Engenharia de São Carlos como parte dos requisitos para a obtenção do título de Mestre em Engenharia de Produção.

Área de concentração: Processos e Gestão de Operações

Orientador: Prof. Dr. Marcelo Seido Nagano

São Carlos - SP
2019

AUTORIZO A REPRODUÇÃO TOTAL OU PARCIAL DESTE TRABALHO, POR QUALQUER MEIO CONVENCIONAL OU ELETRÔNICO, PARA FINS DE ESTUDO E PESQUISA, DESDE QUE CITADA A FONTE.

Ficha catalográfica elaborada pela Biblioteca Prof. Dr. Sérgio Rodrigues Fontes da EESC/USP com os dados inseridos pelo(a) autor(a).

Tomazella, Caio Paziani
T655n Novos limitantes inferiores para o método branch-and-bound na solução de problemas flowshop permutacional / Caio Paziani Tomazella; orientador Marcelo Seido Nagano. São Carlos, 2019.

Dissertação (Mestrado) - Programa de Pós-Graduação em Engenharia de Produção e Área de Concentração em Processos e Gestão de Operações -- Escola de Engenharia de São Carlos da Universidade de São Paulo, 2019.

1. scheduling. 2. flowshop permutacional. 3. branch-and-bound. 4. tempo de fluxo total. 5. atraso total. 6. setup dependente da sequência. I. Título.

Eduardo Graziosi Silva - CRB - 8/8907

# FOLHA DE JULGAMENTO

Candidato: Engenheiro **CAIO PAZIANI TOMAZELLA.**

Título da dissertação: "Novos limitantes inferiores para o método branch-and-bound na solução de problemas flowshop permutacional".

Data da defesa: 15/05/2019

**Comissão Julgadora:** **Resultado:**

Prof. Dr. **Marcelo Seido Nagano** Aprovado
**(Orientador)**
(Escola de Engenharia de São Carlos/EESC)

Prof. Dr. **Roberto Fernandes Tavares Neto** Aprovado
(Universidade Federal de São Carlos/UFSCar)

Prof. Dr. **José Francisco Ferreira Ribeiro** Aprovado
(Faculdade de Economia, Administração e Contabilidade de Ribeirão Preto/FEA/RP)

Coordenador do Programa de Pós-Graduação em Engenharia de Produção:
Prof. Dr. **Marcelo Seido Nagano**

Presidente da Comissão de Pós-Graduação:
Prof. Titular **Murilo Araujo Romero**

# Resumo

Tomazella, Caio Paziani **Novos limitantes inferiores para o método *branch-and-bound* na solução de problemas *flowshop* permutacional**. 90 p. Dissertação de mestrado – Escola de Engenharia de São Carlos, Universidade de São Paulo, 2019.

Em um contexto industrial, a programação da produção tem como objetivo alocar recursos para operações de forma a aumentar a eficiência operacional do processo de fabricação. Esta programação pode ser modelada na forma de problemas de sequenciamento de tarefas, que são resolvidos visando minimizar um determinado critério de desempenho. A aplicação de métodos exatos nestes problemas possibilita encontrar a solução ótima, tanto para aplicação direta como para a validação de métodos heurísticos e metaheurísticas. Entretanto, a literatura mostra que os métodos exatos, tanto a resolução do problema pela modelagem em programação linear-inteira mista como o *branch-and-bound*, têm sua aplicação restrita à problemas de menores tamanhos. O objetivo deste trabalho é propor novas formulações de limitantes inferiores para a aplicação do *branch-and-bound* em problemas de *flowshop* permutacional visando aumentar sua eficiência e aplicabilidade. Os limitantes propostos são avaliados em problemas de *flowshop* permutacional com tempos de *setup* dependente da sequência, tendo como critérios de desempenho o tempo de fluxo total e o atraso total. A avaliação da aplicabilidade de cada limitante é feita através do número de nós explorados e o tempo computacional gasto pelo *branch-and-bound* para resolver problemas de diversos tamanhos.

**Palavras-chave:** *scheduling*, *flowshop* permutacional, *branch-and-bound*, tempo de fluxo total, atraso total, *setup* dependente da sequência.

# Abstract

Tomazella, Caio Paziani **New lower bounds for the branch-and-bound method for solving permutation flowshop problems**. 90 p. Master Thesis – São Carlos School of Engineering, University of São Paulo, 2019.

In an industrial context, production sequencing aims at allocating resources for job processing while increasing manufacturing efficiency. This task can be modelled in the form of scheduling problems, which are solved by minimizing a pre-determined performance criterion. The use of exact methods allows the optimal solution to be found, which can be applied directly in the manufacturing shop or used to validate heuristic and metaheuristic methods. However, the literature shows that MILP and branch-and-bound, both exact methods, are restrained to small-sized scheduling problems. The aim of this project is to propose new lower bound formulations to be used in the branch-and-bound method for permutational flowshop probems, in order to extend its efficiency and applicability. The proposed bounds are tested in permutational flowshop problems with sequence dependent setup times, and using as performance criteria the total flow time and the total tardiness. The evaluation of each lower bounds applicability is done considering the number of explored nodes and the required computational time for the branch-and-bound to solve problem instances of different sizes.

**Keywords:** scheduling, permutation flowshop, branch-and-bound, total flow time, total tardiness, sequence dependent setup times.

# Lista de Ilustrações

Figura 1 Exemplo da aplicação do algoritmo *Branch-and-Bound* para a solução de um *PFSP* com 4 tarefas e 3 máquinas. Fonte: adaptado de Ignall e Schrage (1965) . . . . . 33

Figura 2 Ilustração da fórmula do cálculo do *makespan*. Fonte: Adaptado de Nagano e Moccellin (2002) . . . . . 49

Figura 3 Ilustração das propriedades estruturais entre as tarefas $u$ e $v$. Fonte: Adaptado de Nagano e Moccellin (2002) . . . . . 49

# Lista de Tabelas


Tabela 1 Descrição dos termos matemáticos usados no equacionamento dos problemas . . . . . 24

Tabela 2 Tempos de processamento da instância usada como exemplo . . . . . 32

Tabela 3 Publicações referentes ao uso do *Branch-and-Bound* (B&B) em Problema de *Flowshop* Parmutacional - *Permutation Flowshop Problem* (PFSP)s com 2 máquinas . . . . . 36

Tabela 4 Publicações referentes ao uso do B&B em PFSPs com 3 máquinas . . . 36

Tabela 5 Publicações referentes ao uso do B&B em PFSPs com $m$ máquinas . . 38

Tabela 6 Número de instâncias do banco de dados de Ronconi e Armentano (2001) para cada tamanho de problema . . . . . 61

Tabela 7 Faixa de distribuição dos tempos de *setup*, baseado nas distribuições de Ruiz, Maroto e Alcaraz (2005) . . . . . 61

Tabela 8 Parâmetros para geração de *due dates*, baseado nas distribuições de Ronconi e Armentano (2001) . . . . . 62

Tabela 9 Número de problemas resolvidos por $B\&B_{FT1}$ e $B\&B_{FT2}$ . . . . . 63

Tabela 10 Número médio de nós criados por $B\&B_{FT1}$ e $B\&B_{FT2}$ ao resolverem os problemas. . . . . 64

Tabela 11 Tempo médio (em segundos) gasto para o algoritmo por $B\&B_{FT1}$ e $B\&B_{FT2}$ para resolver os problemas . . . . . 65

Tabela 12 % de sucesso de $B\&B_{FT2}$ sobre $B\&B_{FT1}$ em relação ao tempo computacional gasto para resolver as instâncias . . . . . 66

Tabela 13 Número de problemas resolvidos por $B\&B_{TT1}$ e $B\&B_{TT2}$, divididos pelos grupos de distribuição de *setup* e *due date*. . . . . 67

Tabela 14 Número de problemas resolvidos por $B\&B_{TT1}$ e $B\&B_{TT2}$, divididos por número de tarefas e máquinas . . . . . 68

Tabela 15 Média de nós criados e tempo computacional (em segundos) gasto por $B\&B_{TT1}$ e $B\&B_{TT2}$ para resolver os problemas, divididos pelos grupos de distribuição de *setup* e *due date* . . . . . 69
Tabela 16 Média de nós criados e tempo computacional (em segundos) gasto por $B\&B_{TT1}$ e $B\&B_{TT2}$ para resolver os problemas, divididos por número de tarefas e máquinas . . . . . 70
Tabela 17 % de sucesso de $B\&B_{TT2}$ sobre $B\&B_{TT1}$ em relação ao tempo computacional gasto para resolver as instâncias, divididos entre os grupos de *setup* e *due date* . . . . . 70
Tabela 18 % de sucesso de $B\&B_{TT2}$ sobre $B\&B_{TT1}$ em relação ao tempo computacional gasto para resolver as instâncias, divididos entre os números de tarefas e máquinas . . . . . 71

# Lista de Abreviaturas

**ATSP** Problema do Caixeiro-Viajante Assimétrico - *Asymmetric Traveling Salesman Problem*

**B&B** *Branch-and-Bound*

**EDD** Menor *Due Date* - *Earliest Due Date*

**FIFO** *First In, First Out*

**LB** Limitante Inferior - *Lower Bound*

**LPT** Maior Tempo de Processamento - *Longest Processing Time*

**MILP** Programação Linear Inteira Mista - *Mixed Integer Linear Programming*

**NPS** Sequência Não Parcial - *Non-Partial Sequence*

**PFSP** Problema de *Flowshop* Parmutacional - *Permutation Flowshop Problem*

**PS** Sequência Parcial - *Partial Sequence*

**SPT** Menor Tempo de Processamento - *Shortest Processing Time*

**SRPT** Menor Tempo de Processamento Restante - *Shortest Remaining Processing Time*

**TFT** Tempo de Fluxo Total - *Total Flow Time*

**TT** Atraso Total - *Total Tardiness*

**UB** Limitante Superior - *Upper Bound*

# Sumário


**1 Introdução** **17**

1.1 Justificativa . . . . . 18

1.2 Objetivo Geral . . . . . 19

1.2.1 Objetivos Específicos . . . . . 19

1.3 Estrutura do trabalho . . . . . 20

**2 Definição do Problema** **21**

2.1 Restrições ($\beta$) . . . . . 21

2.2 Critérios de desempenho ($\gamma$) . . . . . 23

2.3 Formulação dos problemas estudados . . . . . 24

**3 Revisão da Literatura** **27**

3.1 O Método *Branch and Bound* . . . . . 27

3.1.1 Limitante Inferior . . . . . 28

3.1.2 Limitante Superior Inicial . . . . . 29

3.1.3 Regra de Busca . . . . . 29

3.1.4 Regra de Ramificação . . . . . 30

3.1.5 Regras de Dominância . . . . . 30

3.1.6 Critério de Parada . . . . . 31

3.1.7 Exemplo da Aplicação do B&B em um PFSP . . . . . 32

3.2 Problemas Estudados na Literatura . . . . . 33

3.3 Considerações . . . . . 38

**4 Limitantes Inferiores** **41**

4.1 Limitantes da Literatura . . . . . 41

4.1.1 Limitantes para $F_m|s^i_{j,l}|C_{max}$ . . . . . 42

4.1.2 Limitantes para $F_m||\sum C$ . . . . . 45

4.1.3 Limitantes para $F_m||\sum T$ . . . . . 47

4.2 Propriedades do PFSP . . . . . 48
4.2.1 A Propriedade $UBX$ . . . . . 50
4.2.2 A Propriedade $LBY$ . . . . . 50
4.2.3 As Propriedades para PFSPs com Tempos de *Setup* Dependentes da Sequência . . . . . 51
4.3 Novos Limitantes Propostos . . . . . 52
4.3.1 Limitantes Propostos para $F_m|s^i_{j,l}|\sum C$ . . . . . 52
4.3.2 Limitantes Propostos para $F_m|s^i_{j,l}|\sum T$ . . . . . 55
4.4 Considerações . . . . . 57

**5 Experimentação Computacional e Resultados 59**
5.1 Metodologia utilizada . . . . . 59
5.1.1 Instâncias . . . . . 60
5.1.2 Especificações computacionais . . . . . 62
5.2 Resultados para $F_m|s^i_{j,l}|\sum C$ . . . . . 62
5.3 Resultados para $F_m|s^i_{j,l}|\sum T$ . . . . . 65
5.4 Considerações . . . . . 68

**6 Conclusão 73**

**Referências Bibliográficas 77**

Capítulo 1

# Introdução

Problemas de sequenciamento de tarefas surgem da necessidade de buscar eficiência no processo produtivo, reduzindo o desperdício de recursos (eg. tempo de ociosidade de máquinas, quantidade de estoque em processamento) e aumentando a velocidade de resposta à variações de demanda (MACCARTHY; LIU, 1993). O sequenciamento é parte crucial da programação da produção, que toma decisões relativas à alocação de recursos para o processamento de tarefas dentro de um horizonte de planejamento, visando minimizar um ou mais critérios pré-determinados. Este processo de tomada de decisão é chamado de *scheduling* (PINEDO, 2008).

Dentro do contexto de uma indústria de manufatura, o sequenciamento da produção localiza-se entre os estágios de planejamento e produção, conectados pela troca constante de informações. A equipe de programação recebe da equipe de planejamento as datas de liberação e entregas de tarefas, disponibilidade de materiais e quantidade necessária a ser produzida; e da equipe de produção recebe atualizações em tempo real da operação das máquinas, como produtividade, paradas inesperadas e perda de material. Por sua vez, a equipe de programação envia para a equipe de planejamento as restrições do plano proposto (quantidade que não pode ser produzida no tempo desejado, itens que irão sofrer atraso, etc.), e para a equipe de produção a sequência detalhada de produção de cada item em cada máquina (PINEDO, 2008).

Técnicas de *scheduling* são aplicadas nos mais diversos tipos de ambientes produtivos, dentre eles o *flowshop*, que é caracterizado por um conjunto de máquinas dispostas em série, com as tarefas seguindo o mesmo roteiro de processamento. Neste caso, o espaço de armazenamento de tarefas em processamento pode ser considerado infinito ou limitado, levando à condição de bloqueio. Outras condições específicas também existem e são estudadas, como a não ociosidade de máquinas ou o tempo máximo que uma tarefa pode ser mantida em espera. O ambiente de *flowshop* em que um ou mais estágio consiste de um conjunto de máquinas paralelas leva o nome de *flowshop* flexível e também é considerado na literatura.

Na literatura, a propriedade permutacional do *flowshop*, que faz com que as máquinas

processem as tarefas na mesma ordem, é comumente considerada para possibilitar o estudo dos problemas, pois reduz exponencialmente o número de soluções possíveis. Um problema desta natureza é chamado de Permutational Flowshop Problem (PFSP), e, genericamente, possui complexidade *NP-Hard*, ou seja, não é possível utilizar um algoritmo com tempo de execução polinomial para obter-se uma solução ótima (MACCARTHY; LIU, 1993).

Sendo assim, dois conjuntos de métodos são utilizados para a obtenção de soluções para PFSPs: métodos heurísticos e métodos exatos. Os primeiros utilizam algoritmos de indexação, construção e/ou melhoria para obter uma solução sub-ótima em pouco tempo computacional. Já os métodos exatos buscam a solução ótima, porém devido ao seus altos tempos de processamento, a utilização destes métodos não é viável para problemas de grande porte. Para *scheduling*, duas classes de métodos exatos são principalmente utilizadas: formulação e solução de um Programação Linear Inteira Mista - *Mixed Integer Linear Programming* (MILP) e enumeração implícita através do B&B. Maccarthy e Liu (1993) mostram que o uso de métodos exatos fica limitado a problemas de pequeno porte, e com o B&B possuindo uma maior aplicabilidade que o MILP.

Considerando PFSPs, os únicos problemas conhecidos que podem ser resolvidos otimamente em tempo polinomial são mostrados por Johnson (1954) (minimizar a duração total da sequência em um *flowshop* de duas máquinas) e Saadani, Guinet e Moallaa (2003) (o mesmo problema anterior sem intervalos de espera na segunda máquina). Além disso, Pinedo (2008) e Wang et al. (2006) mostram casos particulares de outros problemas que podem ser resolvidos facilmente, porém dependem de relações de dominância e/ou igualdade entre os tempos de processamento das tarefas, não podendo ser aplicadas ao caso geral do problema.

Proposto primeiramente para a solução de MILPs (LAND; DOIG, 1960), o B&B foi adaptado para a solução de PFSPs por Ignall e Schrage (1965) e Lomnicki (1965). O método é uma enumeração implícita, que explora as soluções elegíveis por meio da criação de sequências parciais, e utiliza propriedades do problema para identificar quais não levam à uma solução ótima e diminuir assim o espaço de busca.

## 1.1 Justificativa

A importância do uso de *scheduling* em ambientes de produção se dá pelas vantagens obtidas ao sequenciar tarefas visando otimizar um determinado critério de avaliação. A minimização do Tempo de Fluxo Total - *Total Flow Time* (TFT) leva a custos menores de tarefas em processamento e respostas mais rápidas à variações de demanda (PINEDO, 2008; MACCARTHY; LIU, 1993). O uso de critérios relativos aos prazos de entrega tornou-se mais relevantes com a popularização de técnicas *lean*, e atrasos passaram a ser vistos como causas de perda que devem ser eliminadas. Consequências do atraso incluem perda de vendas, atraso de pagamentos e danos à reputação da marca (RONCONI; KAWAMURA,

2010), sendo assim importante o estudo de problemas que minimizam o Atraso Total - *Total Tardiness* (TT) de um ambiente produtivo.

Além das funções objetivas, uma restrição relevante abordada neste trabalho é o tempo de preparação de máquinas para o processamento de tarefas, chamado de *setup*. Allahverdi (2015) justifica que os tempos de *setup* devem ser levados em consideração pois são atividades que não agregam valor no produto final, e a minimização de sua duração implica na redução de perdas e aumento da produtividade e do aproveitamento de recursos. Apesar da relevância apontada por Allahverdi (2015), a revisão da literatura mostra que problemas com tempos de *setup* dependentes da sequência são raramente abordados através do método B&B.

As técnicas de *scheduling* permitem que a equipe responsável planeje a execução de atividades tendo em vista aprimorar índices que são inseridos nas funções objetivo. As características do ambiente produtivo são consideradas durante a modelagem, e os métodos podem ser adaptados para atender as especificidades do problema, estendendo a aplicabilidade para qualquer situação.

Neste contexto, uma solução ótima para o problema e, questão possui diferentes utilidades:

- ❑ Aplicação direta em uma situação prática.
- ❑ Avaliação do desempenho de métodos heurísticos e metaheurísticas.
- ❑ Validação de soluções existentes já aplicadas em ambientes reais.

O uso de um método exato, como o B&B, para encontrar a solução ótima é necessário por dois motivos: computacionalmente, os problemas possuem complexidade *NP-Hard* ou *NP-Complete*, não sendo possível resolvê-los em tempo polinomial; um PFSP com $n$ tarefas possui um número de $n!$ soluções factíveis, e assim, para problemas com $n > 10$ tarefas, a enumeração explícita do conjunto de soluções torna-se inviável.

## 1.2 Objetivo Geral

O objetivo deste trabalho é, baseando-se na literatura referente à aplicação do B&B para PFSP e nas propriedades estruturais do *flowshop*, propor novos Limitante Inferior - *Lower Bound* (LB)s para PFSPs com tempos de *setup* dependente, buscando aumentar a eficiência do método, e ampliando sua viabilidade para problemas de maior tamanho.

### 1.2.1 Objetivos Específicos

- ❑ Realizar uma extensa revisão de artigos contendo o uso de B&B para a solução de PFSPs, de forma a listar os problemas estudados e identificar as principais contribuições da literatura como um todo para o método.

- Estudar as propriedades estruturais dos PFSPs em questão, para, baseando-se nestas, desenvolver LBs de melhor desempenho em comparação aos existentes na literatura.
- Propor limitantes inferiores para os problemas de minimização tempo de fluxo total (TFT) e atraso total (TT)em PFSPs com tempos de *setup* dependentes da sequência.
- Experimentação computacional dos métodos propostos e comparação com os métodos encontrados na literatura, levando em consideração o tempo de execução do algoritmo e o número de nós explorados.

## 1.3 Estrutura do trabalho

A divisão deste trabalho é feita da seguinte forma:

- **Capítulo 2**: Definição do PFSP, suas principais características, os ambientes produtivos em que ocorre, e uma listagem das restrições e funções objetivo encontradas na literatura.
- **Capítulo 3**: Revisão da literatura, dividida em duas partes: a primeira sobre como o método B&B é aplicado em PFSPs, detalhando suas principais ferramentas; a segunda é uma extensa listagem das publicações contendo as aplicações, possibilitando uma visualização dos problemas mais abordados pela literatura.
- **Capítulo 4**: Descrição detalhada dos LBs relacionados aos problemas estudados neste trabalho; formulação dos LBs propostos com base nas propriedades dos problemas.
- **Capítulo 5**: Detalhamento da experimentação computacional, contendo a especificação do equipamento usado, geração de instâncias para testes, método de aplicação do B&B; análise dos resultados obtidos.
- **Capítulo 6**: Considerações finais e sugestões para trabalhos futuros.

Capítulo 2

# Definição do Problema

Em um PFSP, um conjunto de $n$ tarefas, $J = 1, 2, \ldots, n$ deve ser processado por um conjunto de $m$ máquinas, $M = 1, 2, \ldots, m$. Simplificadamente, o tamanho do PFSP é escrito da forma $n \times m$. As atividades que uma máquina executa em uma tarefa são chamadas de operações, e o tempo de processamento da operação da tarefa $j$ na máquina $i$ é dado por $p_{ij}$. Por ser um *flowshop*, toda tarefa segue o mesmo roteiro de fabricação, ou seja, segue a mesma sequência entre as máquinas. Devido à característica permutacional, todas as máquinas processam as tarefas na mesma ordem. Cada operação é processada uma vez e cada máquina possui capacidade para processar uma operação por vez. Assim que uma operação é iniciada, esta não pode ser interrompida (PINEDO, 2008).

Segundo a notação de Graham et al. (1979), um problema de sequenciamento de tarefas é apresentado na forma $\alpha|\beta|\gamma$, onde $\alpha$ representa o ambiente de máquinas, $\beta$ as características e restrições do ambiente e $\gamma$ o critério de desempenho. Os valores de $\alpha$, $\beta$ e $\gamma$ são extensivamente descritos por Pinedo (2008) e os presentes neste trabalho serão detalhados neste capítulo.

O ambiente *flowshop* é representado por $\alpha = F_m$, onde $m$ é o número de máquinas do sistema. Outros ambientes não presentes nesta revisão são: máquina única (1); máquinas em máquinas em paralelo ($P_m$); *flowshop* flexível ou híbrido ($\alpha = FF_m$), no qual um ou mais estágios são compostos por máquinas em paralelo.; *job shop* ($J_m$) em que as tarefas seguem roteiros fixos porém diferentes; e *open shop* ($O_m$), em que as tarefas devem ser processadas em diferentes máquinas porém sem uma ordem específica.

## 2.1 Restrições ($\beta$)

As características e restrições de um ambiente flowshop ($\beta$) mais comumente encontradas na literatura são:

- **Permutacional** (*prmu*): Já mencionada anteriormente, a característica permutacional faz com que as máquinas processem as tarefas na mesma ordem. Como

os problemas de *flowshop* abordados neste trabalho são todos permutacionais, esta característica é omitida no campo das restrições.

- ***Setup* Independente** ($s_j^i$): Para toda máquina, cada tarefa possui uma atividade de *setup* que deve ser executada antes de sua operação.

- ***Setup* Dependente** ($s_{j,l}^i$): Diferentemente do *setup* independente, para cada tarefa existe um tempo de preparação da máquina que varia de acordo com a tarefa anterior ou para inicialização da máquina no caso da primeira tarefa. Nestes dois casos o *setup* pode ser inicializado assim que a tarefa anterior é finalizada, não sendo necessário preceder imediatamente a tarefa seguinte.

- ***Due Dates*** ($d_j$): Por razões externas (ex. requisitos dos clientes) as tarefas possuem datas de entrega chamadas *due dates*, que, quando são consideradas no problema, entram no cálculo da função objetiva.

- ***Release Dates*** ($r_j$): Dentro de um ambiente de máquinas o momento em que a tarefa está disponível para processamento pode, por diversos motivos como prazos estipulados por fornecedores de matéria-prima, variar, e este instante é dado pela *release-date*. Quando todas as tarefas estão disponíveis no instante inicial, tem-se $r_j = 0$.

- **Bloqueio** (*block*): Em um *flowshop* convencional há a disponibilidade de se armazenar estoque intermediário infinitamente entre cada máquina. Nas situações em que este estoque tem capacidade nula (*buffer* zero) ou limitada, há a restrição de bloqueio. Neste cenário, a tarefa $j$ deve ser mantida na máquina $i$ após finalizada até que $i+1$ esteja disponível para processá-la, atrasando assim o início da operação de $j+1$ em $i$.

- ***No-wait*** ($nwt$): Esta restrição impede a tarefa de passar pelo estoque intermediário, ou seja, assim que sua operação na máquina $i$ é finalizada, sua operação em $i+1$ deve ser iniciada imediatamente, não havendo tempo de espera entre máquinas (*job idle time*).

- **Tempo de Espera Limitado** ($w_j$): Define um tempo de espera que a tarefa aguarda entre duas operações. Se $w_j = 0$, ocorre a restrição *no-wait*, quando este tempo é limitado, tem-se $w_j = w_{max}$ e, caso haja um tempo de espera obrigatório entre duas operações, $w_j = w_{min}$.

- ***No-idle*** ($no - idle$): Com esta restrição o tempo de ociosidade de uma máquina entre o término de uma operação e o início da seguinte deve ser zero. Este cenário ocorre quando o custo para manter uma máquina disponível é elevado e/ou esta não pode ter sua operação interrompida. Assim, o início do processamento das tarefas são atrasados para que todo o conjunto $J$ possa ser processado sem interrupções.

- **Disponibilidade de Máquina Limitada** (*brkdwn*): Em situações reais, é comum que as máquinas de um ambiente não estejam disponíveis para funcionamento durante todo o horizonte de sequenciamento, devido a pausas agendadas Kubiak et al. (2002), manutenção preventiva ou falhas técnicas Allaoui e Artiba (2006).

- **Aprendizado e Deterioração** ($r^a$ **e** $\lambda$): Na literatura revisada, os casos encontrados estudam ambientes em que o aprendizado de máquinas/operadores faz com que o tempo de processamento seja reduzido de forma exponencial por um fator $r^a$. Por outro lado, há ambientes em que os tempos aumentam conforme as tarefas são deixadas para o final da sequência, e este efeito, chamado de deterioração, é representado pelo coeficiente $\lambda$.

- **Critério de desempenho** Esta restrição é usada quando o objetivo do problema é minimizar um critério sendo que a sequência obrigatoriamente atenda outro. Por exemplo, minimizar o tempo de fluxo ($\sum C$) em um PFSP de duas máquinas atendendo a condição de a solução possuir o *makespan* ($C_{max}$) ótimo (RAJENDRAN, 1992).

## 2.2 Critérios de desempenho ($\gamma$)

Por último, os critérios de desempenho, ou seja, os valores de $\gamma$, são:

- ***Makespan*** ($C_{max}$): Minimiza o instante de término do processamento da última tarefa, o que representa maximizar a eficiência do sistema produtivo.

- **TFT** ($\sum C$): Minimiza a soma do instante de término de todas as tarefas, o que representa minimizar a quantidade de tarefas em processamento ao longo do tempo e o *lead time* médio de produção.

- **Atraso** ($T_j$): No caso de as tarefas possuírem *due dates*, o atraso é calculado pela fórmula $T_j = max(0, C_j - d_j)$, e minimiza-lo representa reduzir a penalidade por não atender os prazos e maximizar a qualidade de atendimento ao cliente. PFSPs visando minimizar o atraso das tarefas fazem usando os critérios *Atraso Máximo* ($T_{max}$) ou *TT)* ($\sum T$).

- ***Lateness*** ($L_j$): Não possuindo uma tradução própria para português, a *lateness* é utilizada quando o atraso pode tomar valores negativos, havendo uma bonificação por entregar as tarefas antes do prazo. Neste caso, a formulação é $L_j = C_j - d_j$.

- ***Earliness*** ($E_j$): Usada no caso de haver penalidades por adiantamento da entrega em relação a um prazo. A fórmula para a *Earliness* é $E_j = max(0, d_j - C_j)$.

- **Número de Tarefas Atrasadas ($\sum U$)**: Esta função adota valores binários, com $U_j = 1$ se a tarefa $j$ sofre atraso e $U_j = 0$ caso contrário.

- **Peso Ponderado das Tarefas ($w$)**: Usado quando tarefas possuem diferentes prioridades de entrega, e isso é levado para a função objetiva na forma de pesos que são multiplicados ao valor do objetivo correspondente à tarefa ($\sum wC$ e $\sum wT$, por exemplo).

- **Múltiplos Critérios ($f(\alpha Crit_1 + \beta Crit_2 + ...)$)**: Esta função é a soma ponderada de diferentes critérios de desempenho, como por exemplo, *Makespan* e TFT ($\alpha C_{max} + \beta \sum C$, com $\alpha + \beta = 1$).

## 2.3 Formulação dos problemas estudados

Conhecendo-se a notação de Graham et al. (1979) para designar os PFSPs, é possível identificar os dois problemas abordados neste trabalho como $F_m|s^i_{jk}|\sum C$ (minimização do TFT) e $F_m|s^i_{jk}|\sum T$ (minimização do TT). A seguir é apresentado o equacionamento para a obtenção dos valores dos critérios de desempenho. A Tabela 1 apresenta a notação matemática utilizada para as fórmulas.

| Termo | Descrição |
|---|---|
| $p_{i,j}$ | Tempo de processamento da tarefa $j$ na máquina $i$ |
| $C_{i,j}$ | Tempo de finalização da operação da tarefa $j$ na máquina $i$ |
| $s^i_{j,l}$ | Tempo de *setup* da tarefa $j$ após o processamento de $l$ na máquina $i$ |
| $d_j$ | *Due date* da tarefa $j$ |

Tabela 1: Descrição dos termos matemáticos usados no equacionamento dos problemas

Dada uma sequência qualquer, com as Equações 1, 2, 3 e 4 é possível calcular recursivamente os tempos de finalização das tarefas na última máquina. Em seguida, usa-se a Equação 5 para calcular o TFT e a 6 para o TT.

$$C_{i,0} = 0 \quad \forall \quad 1 \leq i \leq m \tag{1}$$

$$C_{i,1} = \max\left(C_{i-1,1}, s^i_{0,1}\right) + pi,1 \quad \forall \quad 2 \leq i \leq m \tag{2}$$

$$C_{1,j} = C_{1,j-1} + s^1_{j-1,j} + p1,j \quad \forall \quad 1 \leq j \leq n \tag{3}$$

$$C_{i,j} = \max\left(C_{i-1,j}, C_{i,j-1} + s^i_{j-1,j}\right) + pi,j \quad \forall \quad 2 \leq i \leq n \quad \forall \quad 2 \leq j \leq n \tag{4}$$

$$\sum C = \sum_{j=1}^{n} C_{m,j} \tag{5}$$

$$\sum T = \sum_{j=1}^{n} (C_{m,j} - d_j)^+ \tag{6}$$

As equações apresentadas são utilizadas para as formulações desenvolvidas no Capítulo 4, e também são válidas para todos os problemas citados nos Capítulos 3 e 4.

Capítulo 3

# Revisão da Literatura

A revisão de literatura é dividida em duas partes: a primeira consiste em descrever o uso do método *Branch and Bound* para PFSPs, suas particularidades e principais componentes; em seguida, uma extensa revisão da literatura sobre a aplicação do método em problemas similares ou relacionados aos estudados neste trabalho é apresentada, mostrando em detalhes os cálculos dos LBs de cada artigo, bem como outras contribuições adicionais.

## 3.1 O Método *Branch and Bound*

O PFSP foi primeiramente estudado por Johnson (1954), que propôs um método para minimizar o *makespan* de maneira ótima em problemas com duas máquinas ($F_2|prmu|C_{max}$) e um caso particular com três ($F_3|prmu|C_{max}$) utilizando uma de regra de despacho. Entretanto, problemas generalizados com $m \geq 3$, são da classe *NP-hard* (PINEDO, 2008), sendo necessários métodos exatos como modelos de programação linear e *branch-and-bound* para resolvê-los.

O método B&B foi proposto por Land e Doig (1960) para a resolução de problemas de otimização com variáveis discretas, consistindo na divisão do problema original em subproblemas. Sua adaptação para a resolução de PFSPs iniciou-se com Ignall e Schrage (1965) e Lomnicki (1965), tratando o sequenciamento como um problema combinatorial e utilizando critérios próprios para ramificação e poda de nós. Desde então, estudos buscam aprimorar este método por meio de diferentes componentes, que serão descritos a seguir. Consequentemente, restrições adicionais foram exploradas: *flowshop* com bloqueio (RONCONI; ARMENTANO, 2001); *flowshop* com tempos de setup (RIOS-MERCADO; BARD, 1999); datas de entrega e liberação das tarefas (GRABOWSKI; SKUBALSKA; SMUTNICKI, 1983; TADEI et al., 1998). Além das restrições, os problemas também variam nos critérios de avaliação: tempo de fluxo (TFT) (BANSAL, 1977); atraso de tarefas (TT) (CHUNG; FLYNN; KIRCA, 2006) e combinação de mais de um objetivo (NAGAR; HERAGU; HADDOCK, 1995).

Resumidamente, o B&B consiste em enumerar todas as possíveis soluções por meio da criação de sequências parciais, tarefa por tarefa, formando uma árvore de nós que se ramifica até as soluções factíveis (sequências completas). Cada nó corresponde à uma sequência parcial e possui propriedades que permitem verificar se este nó pode ser eliminado, diminuindo a região de busca na árvore e o tempo computacional.

Através da revisão de literatura, foi identificado que o algoritmo B&B é formado por 6 diferentes componentes, que serão detalhadamente explicados nas subseções a seguir:

- **Limitante inferior**: menor valor possível do critério de desempenho com uma sequência parcial já fixada;
- **Limitante superior**: menor valor do critério de desempenho considerando as soluções conhecidas, ou seja, o maior valor ótimo do critério de desempenho até o momento;
- **Regra de busca**: critério usado para escolher qual sequência parcial será explorada em um passo do algoritmo;
- **Regra de ramificação**: critério usado para definir como as sequências parciais são geradas;
- **Regras de dominância**: critérios baseados em propriedades do problema que permitem verificar se uma sequência parcial não irá resultar em uma solução ótima, permitindo a exclusão desta da árvore de busca;
- **Critério de parada**: caso o B&B não encontre uma solução até certo ponto, o algoritmo é interrompido para a continuação dos testes em outras instâncias;

### 3.1.1 Limitante Inferior

O LB representa o menor valor da função objetiva que pode ser obtido a partir de uma Sequência Parcial - *Partial Sequence* (PS) e considerando as tarefas na Sequência Não Parcial - *Non-Partial Sequence* (NPS). A fórmula para o LB é obtida através de relaxações do problema original, fazendo com que, para um mesmo problema, existam diferentes possibilidades de se obter um limitante. Sendo assim, este componente é o que mais possui variações dentro da literatura.

Com base na relaxação feita, é possível obter uma estimativa para a função objetivo que, por ser a solução de uma relaxação do problema original, é obrigatoriamente menor (no caso de problemas de minimização) ou igual à solução ótima. Assim, a qualidade do limitante é medida pelo seu valor: em um problema de minimização, melhores formulações resultam em valores mais próximos do ótimo, ou seja, maiores (WOLSEY, 1998).

Outro critério de comparação é a complexidade computacional, já que limitantes complexos, mesmo resultando em valores mais altos, podem requerer um maior tempo computacional para seu cálculo, tornando seu uso inviável.

A relaxação mais utilizada para PFSP é a da capacidade das máquinas. Com exceção de uma ou duas máquinas do ambiente, a capacidade das demais passa a ser considerada infinita podendo processar até $n$ tarefas simultaneamente. Isso faz com que o problema passe a ser um caso de máquina única (IGNALL; SCHRAGE, 1965) ou *flowhsop* com duas máquinas (RIOS-MERCADO; BARD, 1999), e possível de ser resolvido em tempo polinomial. Lageweg, Lenstra e Kan (1978) descrevem os diferentes tipos de limitantes que podem ser obtidos através desta relaxação.

Alternativamente ao LB baseado na relaxação de capacidade das máquinas, há o limitante obtido considerando uma determinada tarefa como gargalo, com as operações intermediárias das demais (processadas entre as máquinas 2 e $m-1$) não interferindo no resultado final. Este limitante foi inicialmente proposto por McMahon e Burton (1967).

Por último, todo PFSP pode ser formulado através de um MILP, e uma relaxação deste também pode ser usada para cálculo de um limitante. Croce, Gupta e Tadei (2000) usam a relaxação linear de variáveis binárias para obter um limitante para o problema $F_2||\sum U$, enquanto Velde (1991) e Croce, Narayan e Tadei (1996) estudam diferentes limitantes calculados através da relaxação lagrangiana do MILP para o $F_2||\sum C$.

### 3.1.2 Limitante Superior Inicial

Em um problema de minimização, o Limitante Superior - *Upper Bound* (UB) representa a solução factível, dentre as conhecidas, que resulta no menor valor da função objetiva (LAND; DOIG, 1960). O uso de um UB permite ao algoritmo podar nós ativos que possuem um valor de limitante inferior maior, reduzindo o tamanho da árvore de busca. É comum a aplicação de uma heurística antes do início da execução do B&B, e alguns casos também aplicam em certos pontos durante a execução. Se uma solução encontrada durante a exploração da árvore possui um valor de função objetiva menor que o limitante superior inicial, este valor então é substituído.

Na literatura referente ao B&B, há casos em que autores aplicam uma heurística já existente na literatura para obter um UB (RONCONI; ARMENTANO, 2001), adaptam um método para a implementação (LEE; WU, 2004), ou desenvolvem uma heurística própria para o problema (CHUNG; FLYNN; KIRCA, 2002).

### 3.1.3 Regra de Busca

Esta regra define a estratégia que o método usa para escolher qual nó será ramificado dentre os nós ativos da árvore. A primeira a ser usada é a de *Melhor Limitante*, que procura pelo nó com menor limitante inferior em toda a árvore do problema, ou seja,

dentro do conjunto de todos os nós ativos. *Melhor Limitante* foi a regra usada inicialmente por Ignall e Schrage (1965) e Lomnicki (1965).

Já a regra de *Profundidade* procura, dentro do conjunto ativos de nós criados por último, pelo nó com menor limitante inferior. Dentro do algoritmo B&B, utilizar esta regra significa que o nó a ser ramificado sempre terá a maior sequência parcial entre todos os nós ativos. A regra de *Profundidade* encontra sequências completas já no início do algoritmo, possibilitando uma diminuição mais rápida do limitante superior. Potts (1980) foi o primeiro autor a utilizar esta regra, e após a década de 90 seu uso tornou-se predominante na literatura.

T'kindt, Croce e Esswein (2004) estudaram o desempenho de regras de busca em diferentes problemas de sequenciamento, dois de máquina única ($1|r_j|\sum C$ e $1||\sum wT$) e um PFSP ($F_2||\sum C$). Para o problema de *flowshop*, foi mostrado que as regras de *Melhor Limitante* e *Largura* (regra na qual o algoritmo seleciona nós mais próximos à origem da árvore, e não encontrada dentre os artigos revisados) são mais eficientes que a de *Profundidade*. Entretanto, quando uma regra de dominância é aplicada, a *Profundidade* mostra melhores resultados que as demais.

### 3.1.4 Regra de Ramificação

A regra de ramificação define como os subproblemas são criados a partir de um nó existente. Genericamente, em um problema de $n$ tarefas um nó ativo contendo uma sequência parcial de tamanho $r$ é ramificado em $n - r$ nós, cada um com uma sequência parcial de tamanho $r + 1$.

A regra de ramificação mais utilizada consiste em posicionar uma tarefa ainda não fixada na primeira posição seguinte a sequência parcial. Esta foi a primeira regra de ramificação a ser proposta com o método (IGNALL; SCHRAGE, 1965; LOMNICKI, 1965), sendo poucos os trabalhos que aplicam uma diferente.

Ladhari e Haouari (2005) usam uma ramificação alternada, na qual as tarefas são alternadamente posicionadas na primeira e última posições livres da sequência parcial. Potts (1980) e Drozdowski et al. (2012) apresentam regras adaptativas, onde uma condição pré-definida determina se a tarefa é posicionada na primeira ou na última posição livre. Por fim, Haouari e Ladhari (2003) fazem a ramificação de um nó em $r' < n - r$ nós, restringindo o B&B para a busca da solução ótima dentro da vizinhança de uma solução inicial.

### 3.1.5 Regras de Dominância

São regras que permitem a poda de nós através da análise de outras propriedades além dos valores dos limitantes. As regras (ou critérios) são particulares para cada variação

de problema e envolvem a comparação de tempos de processamento e de finalização das tarefas.

Antes do início da execução do método é possível estabelecer regras de dominância referente à precedência de tarefas. Genericamente, é matematicamente provado que, em determinado problema há uma solução ótima em que a tarefa $j$ precede a tarefa $l$. Assim, nós em que ocorre o contrário podem ser podados a fim de diminuir a quantidade de nós ativos que não levam à essa solução. Esta classe de regras é comumente usada devido à sua simplicidade computacional, e exemplos podem ser encontrados em Velde (1991) e Tadei et al. (1998).

Uma classe de regras também comum, porém mais específica do que as relações de precedência aparece em trabalhos como os de Croce, Narayan e Tadei (1996) e Chung, Flynn e Kirca (2006) . Estas regras dizem que, dada uma relação de precedência entre os tempos de processamento de $j$ e $l$ e uma sequência parcial qualquer $\sigma$ (não contendo o par de tarefas em questão), $\sigma j, l$ será dominante em relação à $\sigma l, j$. Estas duas regras podem ser aplicadas durante a etapa de ramificação do nó, evitando que nós com sequências dominadas sejam criados, ou seja, podados antes do cálculo do LB, poupando tempo computacional.

Apesar de estes dois tipos de regra de dominância serem os mais encontrados na literatura, a primeira regra a ser proposta compara dois nós, cada um com uma sequência parcial ($\sigma$ e $\sigma'$) contendo exatamente as mesmas tarefas em permutações diferentes. Neste caso, como as tarefas não agendadas são as mesmas, a parcela do LB referente à elas é inalterada, e a dominância de um nó é identificada através dos tempos de finalização da sequência parcial. Ignall e Schrage (1965) utilizaram a regra na primeira aplicação do método B&B, e Rios-Mercado e Bard (1999) e Chung, Flynn e Kirca (2002) desenvolveram também regras desta classe para seus respectivos problemas.

### 3.1.6 Critério de Parada

Para que o algoritmo não gaste um tempo indefinido até encontrar a solução ótima, toda aplicação computacional possui um critério de parada em que o B&B é interrompido, e a solução final é a com menor UB dentre as encontradas.

Em praticamente todos os trabalhos avaliados, o critério de parada é por tempo, com o algoritmo parando caso não encontre uma solução ótima em um tempo pré-determinado (variando de 30 a 60 minutos). Exceções foram encontradas em Chung, Flynn e Kirca (2002) e Chung, Flynn e Kirca (2006), que interrompem algoritmo caso o número de nós criados exceda $4.10^6$ e Madhushini e Rajendran (2012) usa um limite de $2.10^6$, mas também interrompe caso o tempo computacional alcance 60 minutos.

Ao contrário da regra de busca, não foi encontrado nenhum estudo mostrando qual a melhor escolha para um critério de parada de acordo com o problema estudado e o número de tarefas.

### 3.1.7 Exemplo da Aplicação do B&B em um PFSP

Nesta seção será apresentada uma aplicação numérica do método B&B em um PFSP de 4 tarefas e 3 máquinas visando minimizar o *makespan*. Este exemplo pode ser encontrado no artigo de Ignall e Schrage (1965) e mostra o uso de regras de busca e dominância.

Os tempos de processamento da instância são mostrados a seguir na tabela 2:

| Tarefa | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| Máquina 1 | 13 | 7 | 26 | 2 |
| Máquina 2 | 3 | 12 | 9 | 6 |
| Máquina 3 | 12 | 16 | 7 | 1 |

Tabela 2: Tempos de processamento da instância usada como exemplo

A árvore é iniciada criando-se um nó vazio, e seu limitante inferior é calculado considerando as 4 tarefas na NPS. Para este problema, o cálculo é feito usando a fórmula encontrada nos trabalhos de Ignall e Schrage (1965) e Lomnicki (1965).

Em seguida, o nó inicial é ramificado em 4, um para cada tarefa disponível para serem fixadas na sequência. Ao calcular-se o limitante para os nós criados, percebe-se que os nós 1 e 2 possuem o menor valor (55). Por conveniência, o nó 1 é escolhido para ser ramificado na sequência. Como a tarefa 1 já está sequenciada neste nó, 3 outros são criados a partir deste.

A próxima ramificação ocorre no nó 2, que contém a sequência parcial 2 e o menor limitante de todos os nós ativos da árvore, exemplificando o uso da regra *Melhor Limitante* (caso a regra de *Profundidade* fosse usada, o nó 5 seria ramificado).

Em seguida, é possível podar o nó 5 por este ser dominado pelo nó 8. A comparação é feita usando a regra proposta por Ignall e Schrage (1965), que compara as sequências com a mesmas tarefas em ordens diferentes, e a justificativa é encontrada no artigo. Desse modo, economiza-se tempo computacional, já que o nó 5 seria ramificado após o 8.

A terceira ramificação ocorre no nó 8. Como neste caso apenas 2 tarefas ainda podem ser sequenciadas, ambos os nós criados a partir desta ramificação são sequências completas, ou seja, soluções para o problema, e ao invés de um limitante inferior, os nós darão limitantes superiores ao problema. A sequência do nó 11 possui um *makespan* de 63 unidades de tempo, enquanto a do nó 12, 64. Por possuírem um limitante inferior maior que 63, os nós 3, 6, 7 são podados. Caso apenas uma solução ótima seja desejada, também pode-se podar os nós 4 e 10 por possuírem um valor igual a 63.

Por último, o nó 9 é ramificado, gerando uma solução com valor 62 (nó 13) e outra com 63 (nó 14). Assim, como não há mais nós a serem ramificados, a solução ótima do problema é a sequência $2, 3, 1, 4$ com um *makespan* de 62 unidades de tempo. A Figura 1 ilustra a árvore de busca deste problema.

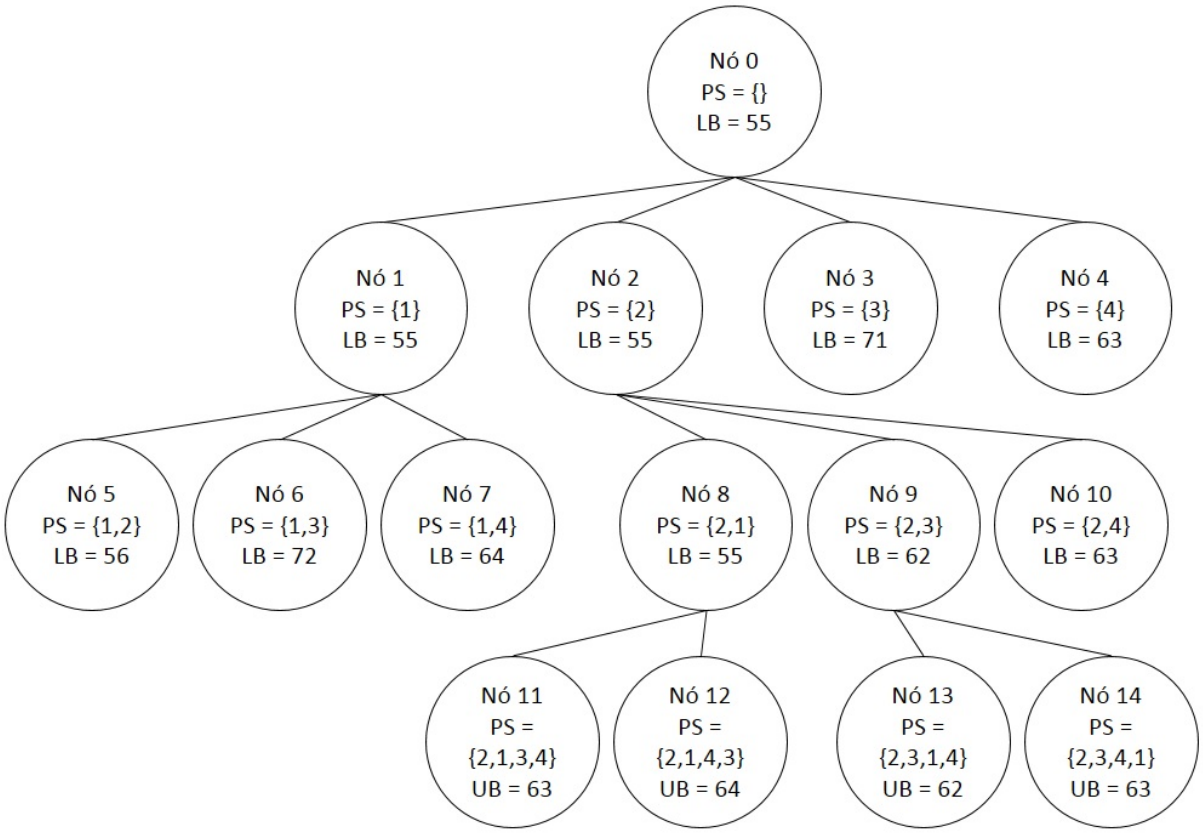


Figura 1: Exemplo da aplicação do algoritmo *Branch-and-Bound* para a solução de um *PFSP* com 4 tarefas e 3 máquinas. Fonte: adaptado de Ignall e Schrage (1965)

## 3.2 Problemas Estudados na Literatura

Nesta seção serão listadas as publicações referentes à aplicação do método B&B em PFSPs. Os artigos são divididos pelos seguintes critérios:

- ❑ Número de máquinas do ambiente *flowshop*;
- ❑ Critério de desempenho;
- ❑ Tipo de *setup*;
- ❑ Restrições do ambiente *flowshop*;

Estão inclusos neste revisão artigos que contenham: a proposta do o uso do B&B para a obtenção da solução ótima de um problema, com a formulação de LBs e regras de dominância; o uso do B&B em outros métodos (ex. metaheurísticas); estudo relacionados a aplicação computacional do método ou do desempenho de diferentes variações de outros componentes do B&B (ex. diferentes regras de busca). Artigos que abordam os

seguintes temas não foram inclusos nesta revisão, por estarem distantes do escopo dos problemas tratados neste trabalho: *flowshop* híbridos ou flexíveis; *flowshop* com montagem de produtos (*assemply*); *flowshop* com características não-permutacionais.

A divisão da listagem dos artigos é feita nas tabelas a seguir: a Tabela 3 contém os artigos referentes aos PFSPs com 2 máquinas ($\alpha = F_2$); a Tabela 4, aos com $\alpha = F_3$; e por última, a Tabela 5 lista os artigos com $\alpha = F_m$.

| Critério | Setup | Restrições | Referências |
|---|---|---|---|
| $C_{max}$ | - | $r_j$ | Tadei et al. (1998), Cheng, Steiner e Stephenson (2002) |
| $C_{max}$ | - | $block$, $\lambda$ | Lee et al. (2010) |
| $C_{max}$ | - | $nwt$, $r_j$, $brkdwn$ | Chihaoui et al. (2011) |
| $C_{max}$ | - | $nwt$, $brkdwn$ | Labidi et al. (2018) |
| $C_{max}$ | - | $w_j$ | Yang e Chern (1995), Bouquard e Lenté (2006), Joo e Kim (2009), Moukrim, Rebaine e Serairi (2014) |
| $C_{max}$ | - | $brkdwn$ | Kubiak et al. (2002), Liao e Tsai (2009), Hnaien, Yalaoui e Mhadhbi (2015) |
| $C_{max}$ | - | $r^a$ | Cheng et al. (2013) |
| $C_{max}$ | - | $\lambda$ | Wagneur e Sriskandarajah (1993) |
| $C_{max}$ | - | $\lambda$, $r^a$ | Wang et al. (2012) |
| $C_{max}$ | - | $Conflict$ | Tellache e Boudhar (2017) |
| $C_{max}$ | - | $Prec.$ | McMahon e Lim (1993), Gim, Curry e Deuermeyer (1994) |
| $C_{max}$ | - | $Buffer$ | Agnetis, Rossi e Gristina (1998), Kononov et al. (2012) |
| $C_{max}$ | - | $Buffer$, $Prec.$ | Lin, Hong e Lin (2009) |
| $C_{max}$ | $s_j^i$ | - | Yang (2002) |
| $C_{max}$ | $s_{j,l}^i$ | $w_j$ | An, Kim e Choi (2016) |
| $\sum C$ | - | - | Ignall e Schrage (1965), Velde (1991), Croce, Narayan e Tadei (1996), Pan e Wu (1996), Croce, Ghirardi e Tadei (2002), Akkan e Karabati (2004), T'kindt, Croce e Esswein (2004), Lin e Wu (2005), Hoogeveen, Norden e Velde (2006), Haouari e Kharbeche (2013) |
| $\sum C$ | - | $r_j$ | Rakrouki et al. (2017) |
| $\sum C$ | - | $w_j$ | Msakni et al. (2016) |
| $\sum C$ | - | $no-idle$ | Narain e Bagga (2005) |

| Critério | Setup | Restrições | Referências |
|---|---|---|---|
| $\sum C$ | - | $r^a$ | Lee e Wu (2004), Li et al. (2011), Wu et al. (2012), Shiau et al. (2015) |
| $\sum C$ | - | $\lambda$ | Wang et al. (2006), Wu e Lee (2006), Wang e Liu (2009b), Ng et al. (2010) |
| $\sum C$ | - | $\lambda$, $r^a$ | Wang e Liu (2009a) |
| $\sum C$ | - | $\lambda$, $min(C_{max})$ | Cheng et al. (2014) |
| $\sum C$ | - | $min(C_{max})$ | Rajendran (1992), T'kindt, Gupta e Billaut (2003) |
| $\sum C$ | - | $min(C_2)$ | Yanai e Fujie (2006) |
| $\sum C$ | $s_j^i$ | - | Allahverdi (2000), Wang e Cheng (2005), Gharbi et al. (2010), Gharbi et al. (2013), Detienne, Sadykov e Tanaka (2016) |
| $\sum C$ | $s_j^i$ | $nwt$ | Aldowaisan (2001), Su e Lee (2008) |
| $\sum wC$ | - | $\lambda$ | Yang e Wang (2011), Wang e Wang (2013b) |
| $L_{max}$ | - | $r_j$ | Haouari e Ladhari (2000), Cheng, Steiner e Stephenson (2002) |
| $L_{max}$ | $s_j^i$ | - | Dileepan e Sen (1991) |
| $L_{max}$ | $s_j^i$ | $nwt$ | Fondrevelle, Allahverdi e Oulamara (2005) |
| $T_{max}$ | - | $r^a$ | Wu, Lee e Wang (2007) |
| $\sum T$ | - | - | Sen, Dileepan e Gupia (1989), Kim (1993), Pan e Fan (1997), Pan, Chen e Chao (2002), Schaller (2005), Haouari e Kharbeche (2013) |
| $\sum T$ | - | $brkdwn$ | Lee e Kim (2017) |
| $\sum T$ | - | $\lambda$ | Bank et al. (2012) |
| $\sum T$ | - | $Prec.$ | Cheng et al. (2017) |
| $\sum U$ | - | - | Croce, Gupta e Tadei (2000) |
| $\sum U$ | - | $r_j$ | Ardakan, Hakimian e Rezvan (2014) |
| $\sum wU$ | - | - | Bulfin e M'Hallah (2003) |
| $\sum Y$ | - | - | Lin, Lin e Lee (2006) |
| $f(C_{max}, \sum C)$ | - | - | Nagar, Heragu e Haddock (1995), Nagar, Heragu e Haddock (1996), Sivrikaya-Serifoglu e Ulusoy (1998), Sayin e Karabati (1999), Yeh (1999), Yeh (2001), Lin e Wu (2006) |
| $f(C_{max}, \sum C)$ | - | $nwt$ | Allahverdi e Aldowaisan (2002) |
| $f(C_{max}, \sum C)$ | - | $\lambda$ | Cheng et al. (2015) |
| $f(C_{max}, L_{max})$ | - | $nwt$ | Allahverdi e Aldowaisan (2004) |

| Critério | Setup | Restrições | Referências |
|---|---|---|---|
| $f(C_{max}, T_{max})$ | - | $r^a$ | Chen, Wu e Lee (2006) |
| $f(C_{max}, \sum T)$ | - | - | Lee e Wu (2001) |
| $f(C_{max}, L_{max})$ | - | - | Toktas, Azizoglu e Koksalan (2004) |
| $f(C_{max}, \sum C, L_{max})$ | - | - | Allahverdi (2001) |
| $T_{max} + E_{max}$ | - | - | Moslehi et al. (2009) |
| $\sum T + \sum E$ | - | - | Yeung, Oguz e Cheng (2004) |
| $f(T_{max}, D)$ | - | - | Mazdeh e Rostami (2014) |

Tabela 3: Publicações referentes ao uso do B&B em PFSPs com 2 máquinas

| Critério | Setup | Restrições | Referências |
|---|---|---|---|
| $C_{max}$ | - | - | Ignall e Schrage (1965), Lomnicki (1965), McMahon e Burton (1967), Tsujimura et al. (1993), Temiz e Erol (2004) |
| $C_{max}$ | - | $r_j$ | Cheng, Steiner e Stephenson (2001) |
| $C_{max}$ | - | $w_j$ | Kim e Lee (2017) |
| $C_{max}$ | - | $no - idle$ | Narain e Bagga (2003) |
| $C_{max}$ | - | $r^a$ | Wang, Wei e Lu (2016) |
| $C_{max}$ | - | $\lambda$ | Wang et al. (2010), Wang e Wang (2013a), Jafari et al. (2016), Jafari et al. (2017) |
| $\sum C$ | $s_j^i$ | - | Allahverdi e Al-Anzi (2006) |
| $f(C_{max}, \sum C)$ | - | - | Yeh e Allahverdi (2004) |

Tabela 4: Publicações referentes ao uso do B&B em PFSPs com 3 máquinas

| Critério | Setup | Restrições | Referências |
|---|---|---|---|
| $C_{max}$ | - | - | Brown e Lomnicki (1966), Nabeshima (1967), Panwalkar e Khan (1975), Bestwick e Hastings (1976), Lageweg, Lenstra e Kan (1978), Potts (1980), Carlier e Rebai (1996), Cheng, Kise e Matsumoto (1997), Cheng, Kise e Karuno (1997), Companys (1999), Cheng e Janiak (2000), Balasubramanian e Grossmann (2002), Haouari e Ladhari (2003), Ladhari e Haouari (2005), Companys e Mateo (2007), Soylu, Kirca e Azizoglu (2007), Drozdowski et al. (2012), Madhushini e Rajendran (2012), Gharbi e Mahjoubi (2013), Ritt (2016) |
| $C_{max}$ | - | *block* | Suhami e Mah (1981), Ronconi e Armentano (2001), Ronconi (2005), Companys e Mateo (2007), Sanches, Takano e Nagano (2016), Toumi et al. (2017b) |
| $C_{max}$ | - | $r^a$ | Chung e Tong (2011), Wu et al. (2015) |
| $C_{max}$ | - | $w_j$ | Fondrevelle, Oulamara e Portmann (2006) |
| $C_{max}$ | - | $min(\sum C)$ | Bagga e Bhambani (2002) |
| $C_{max}$ | $s^i_j$ | - | Schaller (2001), Kurihara et al. (2009) |
| $C_{max}$ | $s^i_{j,l}$ | - | Rios-Mercado e Bard (1999), Schaller, Gupta e Vakharia (2000), Tang e Huang (2007) |
| $C_{max}$ | $s^i_{j,l}$ | *block* | Takano e Nagano (2017) |
| $\sum C$ | - | - | Bansal (1977), Ahmadi e Bagchi (1990), Karabati e Kouvelis (1993), Gowrishankar, Rajendran e Srinivasan (2001), Chung, Flynn e Kirca (2002), Gharbi e Mahjoubi (2013), Ren et al. (2017) |
| $\sum C$ | - | $r_j$ | Bai et al. (2017) |
| $\sum C$ | - | *block* | Moslehi e Khorasanian (2013) |
| $L_{max}$ | - | $r_j$ | Grabowski, Skubalska e Smutnicki (1983), Haouari e Ladhari (2007) |
| $L_{max}$ | - | $w_j$ | Fondrevelle et al. (2009) |
| $L_{max}$ | - | $r^a$ | He (2016) |
| $\sum T$ | - | - | Kim (1995), Gowrishankar, Rajendran e Srinivasan (2001), Chung, Flynn e Kirca (2006) |
| $\sum T$ | - | *block* | Ronconi e Armentano (2001), Toumi et al. (2017a) |

| Critério | Setup | Restrições | Referências |
|---|---|---|---|
| $\sum T$ | - | $r^a$ | Lee e Chung (2013) |
| $\sum T$ | - | $\lambda$ | Lee, Yeh e Chung (2014) |
| $\sum U$ | - | - | Hariri e Potts (1989) |
| $f(C_{max}, \sum C)$ | - | $r^a$ | Chung e Tong (2012), Wang e Zhang (2015) |
| $f(C_{max}, \sum C)$ | $s^i_{j,l}$ | - | Hendizadeh, ElMekkawy e Wang (2007) |
| $\sum(wC, wT, wE)$ | - | - | Madhushini, Rajendran e Deepa (2009), Madhushini e Rajendran (2011) |

Tabela 5: Publicações referentes ao uso do B&B em PFSPs com $m$ máquinas

## 3.3 Considerações

Primeiramente, pode-se observar que as publicações referentes à aplicação do B&B em PFSPs seguem um padrão:

- Definição e formulação do problema;
- Estudo de propriedades do PFSP e proposta de regras de dominância;
- Cálculo de um ou mais LBs;
- Descrição do algoritmo: heurística usada para encontrar o UB, regras de ramificação e busca;
- Experimentação computacional;

Dentre os artigos revisados, foi notado que a motivação por trás do estudo do método B&B se dá por dois fatores. O primeiro é a superioridade do B&B em relação a outros métodos exatos (MACCARTHY; LIU, 1993), ou seja, é o método que chega em soluções ótimas com menor tempo computacional. O segundo é o uso destas soluções para validar o desempenho de métodos heurísticos. Esta análise é feito calculando o erro das soluções das heurísticas relativo às soluções ótimas de instâncias que o B&B é capaz de resolver dentro de um tempo factível, e, na maioria dos casos, este estudo é seguido de uma comparação dos resultados heurísticos entre si para instâncias de grande porte.

Em relação aos tipos de problemas estudados, foi percebida uma grande segmentação, ou seja, o método B&B foi proposto para diversos problemas únicos. Na Tabelas 3 e 4 (problemas com $m = 2$ e $m = 3$, respectivamente) foram listados no total 59 diferentes PFSPs, com 42 (71%) deles presentes uma única publicação. Percebe-se também uma diversidade nas restrições e critérios de desempenho abordados.

Já os PFSPs com $m \geq 3$, listados na tabela 4, apresentam uma menor variedade, com 22 combinações únicas, e 12 (55%) sendo estudados apenas uma vez. Também pode-se observar um foco maior nos critérios de desempenho e uma menor variedade de restrições no cambo $\beta$ para esta classe de problemas.

Esta tendência faz com que a literatura referente ao B&B possua áreas defasadas entre si, com novos métodos e formulações que foram aplicadas para um e não para outra.

Por último, existe uma quantidade baixa de publicações estudando a aplicação do B&B em PFSPs com tempos de *setup* (11 dos 81 PFSPs, ou seja, 14%). Allahverdi (2015), em sua terceira revisão completa sobre esta restrição, destaca a importância de seu estudo e mostra uma extensa variedade de métodos heurísticos e metaheurísticos para a obtenção de soluções de alta qualidade, porém a quantidade de artigos referentes ao B&B também é baixa se comparada ao todo.

Em relação ao B&B em si, foram encontrados poucos estudos em relação ao desempenho das variação de componentes do método. Dentre os exemplos estão T'kindt, Croce e Esswein (2004) (desempenho de diferentes regras de busca), Potts (1980) (proposta de uma nova regra de ramificação) e Ladhari e Haouari (2005) (uso de LBs de diferentes complexidades de acordo com o tamanho da sequência parcial do nó).

Tendo em vista estas observações, este trabalho irá abordar a aplicação do método B&B para PFSPs do tipo $F_m$ com tempos de *setup* dependentes da sequência, e minimizando dois dos critérios de desempenho mais relevantes encontrados na literatura: TFT ($\sum C$) e TT ($\sum T$). Estes problemas foram escolhidos pois, apesar de não terem sido abordados dentro da literatura referente ao B&B, suas variações menos complexas possuem uma base literária consolidada, a qual é possível ser usada como um ponto de partida para este trabalho. A motivação para o estudo dos tempos de *setup* dependente da sequência nestes problemas decorre da baixa quantidade de publicações referentes à esta característica.

Os LBs propostos serão calculados a partir da relaxação da capacidade de máquinas, transformando o PFSP em problemas de máquina única, solucionáveis em tempo polinomial. Esta classe de LBs é a mais encontrada na literatura, com a complexidade computacional variando entre $O(mn)$ e $O(mn^2)$.

Concluindo, a seguir estão listados os problemas para os quais serão propostos novos LBs, selecionados a partir de lacunas encontradas na literatura revisada, escritos na notação de Graham et al. (1979), são $F_m|s^i_{jk}|\sum C$ e $F_m|s^i_{jk}|\sum T$. O próximo capítulo contém a descrição detalhada dos LBs contidos nos artigos listados anteriormente, e que são referentes a estes dois problemas, além das formulações propostas.

Capítulo 4

# Limitantes Inferiores

O objetivo desta seção é propor LBs de alto desempenho para diferentes PFSPs, e comparando com LBs presentes na literatura com características semelhantes. A seguir é apresentada a notação utilizada neste trabalho. Como cada autor utiliza uma notação própria em seus artigos, a fórmula de cada LB existente é reescrita usando esta para possibilitar uma comparação direta com os novos. Os primeiros termos já foram descritos na Tabela 1 do Capítulo 2, quando introduzidos no trabalho pela primeira vez.

- $p_{i,j}$ - Tempo de processamento da tarefa $j$ na máquina $i$;
- $C_{i,j}$ - Tempo de finalização da operação da tarefa $j$ na máquina $i$;
- $q_i$ - Tempo de finalização da PS na máquina $i$;
- $s^i_{j,l}$ - Tempo de *setup* da tarefa $j$ após o processamento de $k$ na máquina $i$;
- $d_j$ - *Due date* da tarefa $j$;
- $J$ - Conjunto total de tarefas;
- $J'$ - Conjunto de tarefas da NPS;
- $T$ - Conjunto das tarefas da NPS mais a última tarefa da sequência parcial;
- $\sigma$ - Uma sequência parcial contendo tarefas de $J$, de tamanho $s < n$;
- $|PS|$ - Cardinalidade (número de tarefas) da PS;
- $|NPS|$ - Cardinalidade da NPS;

## 4.1 Limitantes da Literatura

Os LBs selecinados para estudo são os seguintes:

- $F_m|s^i_{j,l}|C_{max}$: Rios-Mercado e Bard (1999), Tang e Huang (2007) e Takano e Nagano (2017) (este último contendo a restrição de bloqueio);
- $F_m||\sum C$: Bansal (1977), Ahmadi e Bagchi (1990), Chung, Flynn e Kirca (2002) e Madhushini, Rajendran e Deepa (2009);
- $F_m||\sum T$: Kim (1995) e Chung, Flynn e Kirca (2006);

Estes LBs foram escolhidos por possuírem como característica em comum a relaxação feita para os seus cálculos. Nestes casos a capacidade das máquinas, com exceção de uma, é considerada infinita, transformando o PFSP em um problema de máquina única. Para cada máquina gargalo um valor de LB é calculado, e ao final, o maior valor dentre os $m$ obtidos é usado como LB.

Dentre os LBs relacionados porém não inclusos nesta seção estão:

- LB de Schaller, Gupta e Vakharia (2000) para $F_m|s^i_{j,l}|C_{max}$: o problema considera tempos de *setup* para famílias de tarefas.
- LB de Hendizadeh, ElMekkawy e Wang (2007) para $F_m|s^i_{j,l}|f(C_{max}, \sum C)$: o problema considera tempos de *setup* para famílias de tarefas.
- LB de Karabati e Kouvelis (1993) para $F_m||\sum C$: o LB usa a relaxação Lagrangiana da formulação MILP do problema.
- LBs de Gowrishankar, Rajendran e Srinivasan (2001) e Ren et al. (2017) para $F_m||\sum C$: o problema considera funções não lineares do critério de desempenho.
- LB de Gowrishankar, Rajendran e Srinivasan (2001) para $F_m||\sum T$: o problema considera funções não lineares do critério de desempenho.

LBs para os critérios de desempenho $\sum C$ e $\sum T$ com restrições como aprendizado (LEE; CHUNG, 2013) ou bloqueio (MOSLEHI; KHORASANIAN, 2013) também não foram inclusos nesta seção pois as particularidades do problema levam à formulações que fogem do escopo dos PFSPs estudados neste trabalho, ou são adaptações de LBs existentes para os casos específicos em questão.

### 4.1.1 Limitantes para $F_m|s^i_{j,l}|C_{max}$

Para um melhor entendimento da formulação dos LBs apresentados nesta seção, é necessário primeiramente mostrar o desenvolvimento do primeiro LB encontrado na literatura, proposto por Ignall e Schrage (1965) para o problema com 3 máquinas, e adaptado por Brown e Lomnicki (1966) para o caso genérico com $F_m$.

Sendo uma máquina $k$ ($1 \leq k \leq m$) o gargalo, calcula-se, para as tarefas em $J'$, o tempo total de processamento em $k$ e a menor soma dos tempos nas máquinas seguintes

de uma única tarefa. Esta soma é chamada de cauda, e pode ser calculada durante a etapa de pré-processamento, economizando tempo de processamento. Estes termos são então tomados somando-se ao tempo de finalização da PS, $q_k$.

$$LB_{BL} = \max_{1 \leq k \leq m} (q_k + \sum_{j \in J'} p_{k,j} + \min_{j \in J'} \sum_{i=k+1}^{m} p_{i,j}) \tag{7}$$

Ignall e Schrage (1965) complementaram o LB com um cálculo mais preciso do termo $q_k$, que considera também os tempos de processamento nas máquinas anteriores à $k$, ou seja, o início do tempo de processamento das tarefas de $J'$ em $k$ só sera iniciado após $q_k$ ou o menor tempo possível de finalização de uma tarefa de $J'$ em $k-1$, considerando todas as máquinas entre 1 e $k-1$. Esta melhoria é calculada através de (8)

$$q'_k = \max_{1 \leq i \leq k} (q_i + \min_{j' \in J'} (\sum_{i'=i}^{k-1} p_{i',j'})) \tag{8}$$

#### 4.1.1.1 Limitante de Rios-Mercado e Bard (1999)

Rios-Mercado e Bard (1999) propuseram dois LBs para o problema $F_m|s^k_{l,j}|C_{max}$, o primeiro utilizando uma máquina como gargalo ($LB_{RB}$), e o segundo utilizando um par ($LB_{RB2}$). Apesar de os autores terem mostrado que $LB_{RB2}$ dominou $LB_{RB}$ em termos de valores, $LB_{RB}$ foi mais eficiente quanto aplicado no algoritmo B&B por ser menos complexo computacionalmente.

Considerando uma máquina como gargalo, o LB é calculado através de problemas do tipo $1|s^k_{l,j}|C_{max}$. Este problema pode é escrito como um Problema do Caixeiro-Viajante Assimétrico - *Asymmetric Traveling Salesman Problem* (ATSP), com os tempos de *setup* sendo as distâncias entre as cidades. Como o ATSP é $NP$-hard, Rios-Mercado e Bard (1999) usaram um LB para este problema, dado pela solução do problema de designação obtido quando as restrições de sub-rotas do ATSP são relaxadas ($S_k(J')$). Entretanto, os autores não especificaram o método usado para resolver este problema. A formulação para $LB_{RB}$ é similar a $LB_{BL}$, e inclui os tempos de *setup* através do termo $S_k(J')$.

$$LB_{RB} = \max_{1 \leq k \leq m} \left( q''_k + \sum_{j \in J'} p_{kj} + S_k(J') + \min_{j \in J'} \sum_{i=k+1}^{m} p_{i,j} \right) \tag{9}$$

Com $q''_k$ sendo calculado da mesma maneira que $q'_k$, porém com os tempos de *setup*:

$$q''_k = \max_{1 \leq i \leq k} \left( q_i + \min_{j \in J'} \left( s^i_{s,j} + \sum_{i'=i}^{k-1} p_{i',j} \right) \right) \tag{10}$$

#### 4.1.1.2 Limitante de Tang e Huang (2007)

Tang e Huang (2007) modelaram um ambiente de produção de aço na forma de um problema do tipo $F_m|s^k_{l,j}|C_{max}$, e propuseram um algoritmo B&B usando uma formulação

de LB semelhante a Rios-Mercado e Bard (1999). Os autores armazenaram os tempos de *setup* em uma matriz $(n-s+1) \times (n-s+1)$ de distâncias do ATSP, e usaram um método de transformação para calcular um LB para este problema.

Seja $T$ o conjunto de tarefas contendo a última tarefa da PS e as tarefas da NPS. Para cada máquina $k$, a matriz é gerada com índices $c_{lj}$: se $l \in T$ e $j \in J'$, $c_{ij} = s^k_{l,j}$; se $l \in T$ e $j = s$, $c_{i,j} = 0$; e para todo $i = j$, $c_{i,j} = \infty$

$$TZ_k(J') = \sum_{l \in T} c_{l,j'} + \sum_{j \in T} \left( \min_{i \in T} \left( c_{l,j} - c_{l,j'} \right) \right) \tag{11}$$

Em que:

$$c_{l,j'} = \min_{j \in T} s^k_{l,j} \tag{12}$$

A fórmula final para o LB de Tang e Huang (2007) é então:

$$LB_{TH} = \max_{1 \leq k \leq m} \left( q_k + \sum_{j \in J'} p_{k,j} + TZ_k(J') + \min_{j \in J'} \sum_{i=k+1}^{m} p_{i,j} \right) \tag{13}$$

#### 4.1.1.3 Limitante de Takano e Nagano (2017)

Takano e Nagano (2017) formulou quatro LBs para o *makespan* em um PFSP com tempos de *setup* dependentes da sequência e bloqueio. Neste trabalho, o LB mais eficiente destes quatro, $LB_{BTN}$, é revisado. Os autores usaram a relaxação da capacidade de máquinas e a propriedade $LBB$, que é um LB para o tempo de espera das tarefas que é causado pela restrição de bloqueio. Estes termos são detalhados na publicação original. A fórmula de $LB_{BTN}$ é mostrada na Equação 14.

$$LB_{BTN} = \max_{1 \leq k \leq m} \left( q_k + SLBB^k_{J'} + \sum_{j \in J'} p_{k,j} + Ss^k_{J'} + ST^k_{J'} \right) \tag{14}$$

Com:

$$ST^k_{J'} = \min_{j \in J'} \left( \min_{\substack{j' \in J' \\ j' \neq j}} \left( \sum_{i=k+1}^{m} LBB^k_{j',j} \right) + \sum_{i=k+1}^{m} p_{i,j} \right) \tag{15}$$

Em $LB_{BTN}$, os tempos de *setup* são computados no termo $S_{s^k_{J'}}$, o qual é uma soma dos mínimos tempos de *setup* que seguem cada uma das tarefas em $T$. Já que não há uma operação de *setup* após a última tarefa da sequência completa, o máximo destes valores mínimos, excluíndo o referente à $s$, é subtraído. A fórmula para calcular $S_{s^k_{J'}}$ é mostrada na Equação 16.

$$Ss^k_{J'} = \sum_{l \in T} \left( \min_{j \in J'} s^k_{l,j} \right) - \max_{l \in J'} \left( \min_{\substack{j \in J' \\ l \neq j}} s^k_{l,j} \right) \tag{16}$$

### 4.1.2 Limitantes para $F_m||\sum C$

#### 4.1.2.1 Limitante de Bansal (1977)

O primeiro LB para $F_m||\sum C$ foi proposto por Bansal (1977), usando a relaxação de capacidade de máquinas, transformando o PFSP em $m$ problemas do tipo $1||\sum C$. Este problema é resolvido de maneira ótima pela regra Menor Tempo de Processamento - *Shortest Processing Time* (SPT) (SMITH, 1956). Seja $p_{k,[j]}$ a $j$-ésima tarefa da sequência SPT na máquina $k$, o LB de Bansal (1977), para qualquer sequência parcial $\sigma$, é dado pela Equação 17.

$$LB_B = \sum_{j=1}^{s} C_m, j + \min_{1\leq k\leq m}\left((n-s)q'_k + \sum_{j=1}^{n-s}\left((n-s-j+1)p_{k,[j]} + \sum_{i=k+1}^{m} p_{i,[j]}\right)\right) \quad (17)$$

#### 4.1.2.2 Limitante de Ahmadi e Bagchi (1990)

Ahmadi e Bagchi (1990) usaram a mesma relaxação, porém consideraram problemas de máquina única do tipo $1|r_j|\sum C$, com as *release dates* das tarefas sendo:

$$r_{kj} = \max_{1\leq i\leq k}\left(q'_k + \sum_{i'=i}^{k-1} p_{i',j}\right) \quad (18)$$

Apesar de o problema $1|r_j|\sum C$ é $NP$-hard, é possível obter uma solução em tempo polinomial usando a regra Menor Tempo de Processamento Restante - *Shortest Remaining Processing Time* (SRPT) quando preempções são permitidas. A fórmula final do LB é dada pela Equação 19, na qual $C'(j, r_{j,k})$ é o TFT da sequência preemptiva em $k$. É importante observar que, quando $k = 1$, $r_{1,j} = 0$, e a regra SPT pode ser aplicada diretamente.

$$LB_{AB} = \sum_{j=1}^{s} C_j + \min_{1\leq k\leq m}\left(C'(j, r_{j,k}) + \sum_{j=1}^{n-s}\sum_{i=k+1}^{m} p_{i,[j]}\right) \quad (19)$$

Ahmadi e Bagchi (1990) mostraram que $LB_{AB}$ dominou $LB_B$ provando matematicamente e através da experimentação em nós raízes.

#### 4.1.2.3 Limitante de Chung, Flynn e Kirca (2002)

Chung, Flynn e Kirca (2002) aprimorou a formulaçao de Bansal (1977) considerando o tempo de finalização das tarefas em $k-1$ quando $k$ é a máquina gargalo.

Seja $R_{k,[j]}$ um LB para o tempo de finalização da $j$-ésima tarefa da sequência SPT na máquina $k$, calculado por $R_{k,[j]} = q'_k + \sum_{t=1}^{j} p_{k,[j]}$, com $R_{k,[0]} = q'_k$. Um LB para o início

de $[j]$ em $k$ é dado por $E_{k,[j]} = \max(R_{k,[j-1]}, R_{k-1,[j]})$. Para qualquer sequência parcial $\sigma$, o LB para o TFT é:

$$LB_{TFT-CFK} = \sum_{j=1}^{s} C_{m,j} + \max_{1 \leq k \leq m} \left( \sum_{j=1}^{n-s} \left( E_{k,[j]} + \sum_{i=k}^{m} p_{i,[j]} \right) \right) \tag{20}$$

Os autores também mostraram que $LB_{CFK}$ dominou tanto $LB_B$ e $LB_{AB}$:

- Ao substituir $E_{k,[j]} + p_{k,[j]}$ por $R_{k,[j]}$ em $LB_{CFK}$ (Equação 20), a mesma formulação de $LB_B$ (Equação 17) é obtida. Como, por definição, $E_{k,[j]} + p_{k,[j]} \geq R_{k,[j]}$, $LB_{CFK} \geq LB_B$.

- Através de experimentação, foi observado que $LB_{CFK}$ é mais eficiente que $LB_{AB}$, podando mais nós quando aplicado no algoritmo B&B.

#### 4.1.2.4 Limitante de Madhushini, Rajendran e Deepa (2009)

Madhushini, Rajendran e Deepa (2009) propuseram um cálculo de LB para a finalização de uma tarefa $j \in J'$, quando colocada em cada possível posição da NPS. Este método permitiu o cálculo do LB de uma sequência parcial para três diferentes critérios, TFT, TT e Adiantamento Total (*Total Earliness*). No artigo citado, os autores consideraram a versão com peso de cada critério, bem como a soma dos mesmos.

Dentro do algoritmo B&B, o LB é aplicado da seguinte maneira:

- Passo 1: para cada $j \in J'$, o LB to tempo de finalização de $j$ em $m$ é calculado para cada posição que $j$ pode ocupar na NPS. Neste caso, $R_{m,[r],j}$, em que $j$ ocupa a posição $r + 1$ da NPS.

- Passo 2: os valores são alocados em uma matriz de tamanho $|NPS| \times |NPS|$, representando um problema de designação.

- Passo 3: como o problema de designação é resolvido em tempo polinomial, sua solução ótima é somada ao critério de desempenho de $\sigma$, totalizando um LB para a sequência parcial.

Este LB mostrou-se altamente complexo, tanto para sua implementação como execução, e não foi encontrado um modo efetivo de incluir tempos de *setup* dependente nesta formulação. Por este motivo, optou-se por não usar este LB como base para os propostos neste trabalho.

Na sequência, Madhushini e Rajendran (2011) estenderam o uso deste LB para diferentes critérios de desempenho envolvendo tempos de fluxo e atrasos, porém com nenhuma mudança na formulação.

#### 4.1.2.5 Limitante de Gharbi e Mahjoubi (2013)

Gharbi e Mahjoubi (2013) adaptou o LB de Croce e T'kindt (2003) para o problema com $m$ máquinas. A formulação foi originalmente proposta para o problema $1|r_j|\sum C$, permitindo preemptividade e usando a regra SRPT. Com uma sequência ordenada, era feito o cálculo de índices para estimar um mínimo aumento no TFT desta sequência no caso de remoção da preemptividade, e este valor é adicionado ao LB final.

A adaptação foi feita considerando $m$ problemas de máquina única com *release dates* e *tempos de entrega*. Assim, a formulação de Gharbi e Mahjoubi (2013) pode ser identificada como uma versão aprimorada de $LB_{AB}$ (AHMADI; BAGCHI, 1990).

### 4.1.3 Limitantes para $F_m||\sum T$

#### 4.1.3.1 Limitante de Kim (1995)

O primeiro LB para $F_m||\sum T$ foi proposto por Kim (1995), usando as relaxações de capacidade de máquina (IGNALL; SCHRAGE, 1965; LOMNICKI, 1965) e uma regra ramificação adicionando tarefas à sequência parcial de trás para frente.

Em cada máquina $k$, as tarefas da NPS são ordenadas em SPT, e as *due dates*, em Menor *Due Date - Earliest Due Date* (EDD). Para cada $j$-ésima tarefa da NPS, é calculado um LB para o atraso considerando a soma dos tempos na SPT em $k$ até $j$, um tempo mínimo para o início do processamento da SPT em $k$ e um tempo mínimo de finalização das tarefas na máquina seguinte. Como estes cálculos resultam em um vetor de valores não-decrescentes, o autor mostrou que, subtraindo as *due dates* em EDD pelos seus termos correspondentes, é obtido um LB para o atraso total da sequência.

Para as tarefas da PS, são calculados LBs para o tempo de liberação da máquina, e com esses valores é possível estimar um atraso para as tarefas já sequenciadas usando as mesmas fórmulas recursivas de uma sequência convencional. O LB para a liberação da máquina $k$ é dado por:

$$\max_{1\leq i\leq k}\Big(min_{j\in J'}\Big(\sum_{i'=1}^{i-1} p_{i',j}\Big)+\sum_{j\in J'} p_{i,j}+min_{j\in J'}\Big(\sum_{i'=i+1}^{k} p_{i',j}\Big)\Big) \tag{21}$$

O tempo de finalização de uma tarefa da PS, considerando que as máquinas sejam liberadas nos tempos calculados acima, é dado por $C'_{m,j}$. Assim, $LB_K$ é calculado usando a seguinte formulação:

$$LB_K = \sum_{j=1}^{s}\Bigg(\max_{1\leq k\leq m}\Big(min_{j'\in J'}\Big(\sum_{i=1}^{k-1} p_{i,j'}\Big)+\sum_{j'=1}^{j} p_{k,[j']}+min_{j'\in J'}\Big(\sum_{i=k+1}^{m} p_{i,j'}\Big)-d_{[j-s]}\Big)^{+}\Bigg)+ \sum_{j=s+1}^{n}(C'_{m,j}-d_j)^{+} \tag{22}$$

#### 4.1.3.2 Limitante de Chung, Flynn e Kirca (2006)

Este LB usa como base o de Chung, Flynn e Kirca (2002), ordenando as tarefas de $J'$ por SPT em $k$ e fazendo os mesmos cálculos de $E_{k,t}$ para cada uma. Separadamente, para cada $t \in J'$ é calculado um termo $h_{k,t}$, a diferença entre a *due date* de $t$ e a sua cauda.

$$h_{k,t} = d_t - \sum_{i=k}^{m} p_{i,t} \tag{23}$$

Estes termos também são ordenados de forma não-decrescente e, seguindo a notação usada pelo autor, $h_{k,[t]}$ é o $t$-ésimo menor valor de $h_{k,t}$. Por fim, o atraso da PS é $TT(PS) = \sum_{j=1}^{s}(C_{m,j} - d_j, 0)^+$. A fórmula final do limitante é mostrada em 24.

$$LB_{TT-CFK} = \sum_{j=1}^{s} (C_{m,j} - {d_j}^+ + \max_{1 \leq k \leq m} \left( \sum_{t=s+1}^{n} \left( E_{k,[t]} - h_{k,[t-s]} \right)^+ \right) \tag{24}$$

Os autores mostram que o valor de $LB_{TT-CFK}$ é um limitante inferior. A sequência $(E_{k,[t]} - h_{k,[t-s]})$ de $t = 1, 2, \ldots, n - s$ pode ser escrita genericamente como $(a_t - b_t)$. Se $a_t$ e $b_j$ estiverem ordenados em ordem não-decrescente, $(a_t - b_t)$ também estará. A prova desta relação matemática é mostrada em detalhes por Kim (1995).

Além disso, foi também mostrado que $LB_{TT-CFK}$ dominou $LB_K$, no caso deste ser aplicado para sequências parciais construídas da primeira posição para a última.

## 4.2 Propriedades do PFSP

Esta seção tem como objetivo estudar as propriedades estruturais do PFSP com o objetivo de formular um termo para ser aplicado nos LBs, de forma a torná-los mais eficientes em relação ao valor total.

Em um PFSP convencional, o *makespan* de uma sequência conhecida pode ser calculado pelo uso da fórmula (25).

$$C_{max} = \sum_{j=1}^{n} p_{1,j} + \sum_{k=2}^{m} p_{k,n} + \sum_{k=1}^{m-1} Y_n^k \tag{25}$$

Em que $Y_n^k$ é o tempo que uma tarefa $v$ espera entre o fim de sua operação em uma máquina $k$ e seu início em $k + 1$. A Figura 2 ilustra este intervalo de tempo dentro do sequenciamento. É importante notar que (25) pode ser usada para o cálculo de qualquer $C_{i,j}$ apenas mudando os termos da fórmula.

Na Figura 3 é possível observar a relação entre duas tarefas subsequentes, $u$ e $v$. Além de $Y_v^k$, o outro termo usado é $X_{u,v}^k$, que representa o tempo de espera da máquina entre duas operações subsequentes.

Pelas propriedades conhecidas do PFSP convencional, o início do processamento de uma operação se dá imediatamente após: **1)** o fim do processamento da mesma tarefa

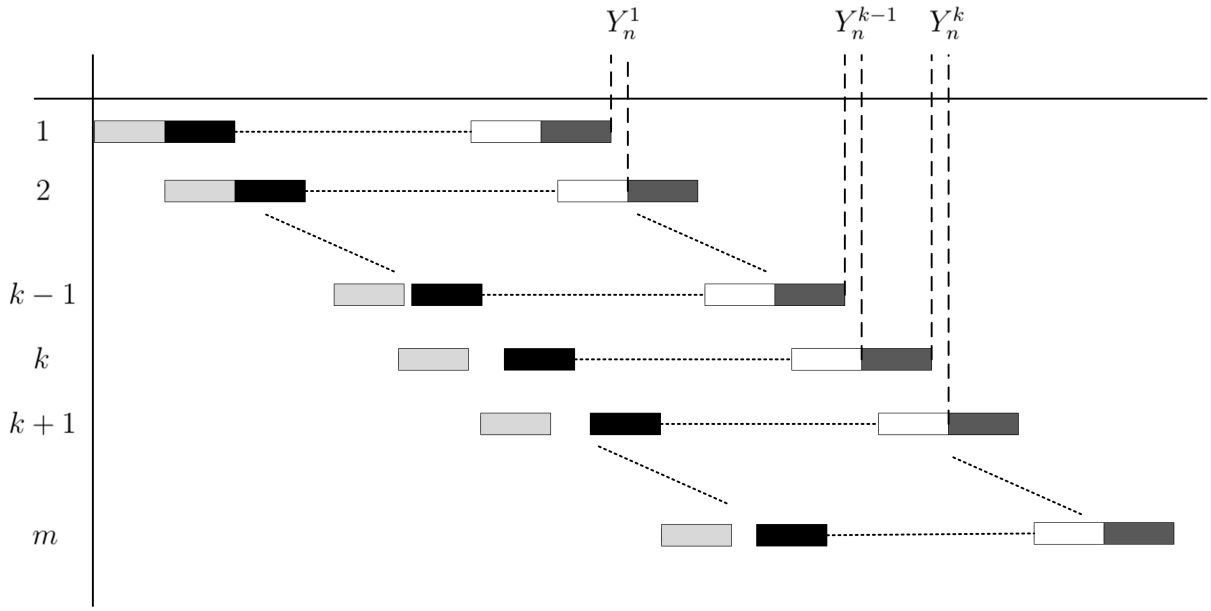


Figura 2: Ilustração da fórmula do cálculo do *makespan*. Fonte: Adaptado de Nagano e Moccellin (2002)

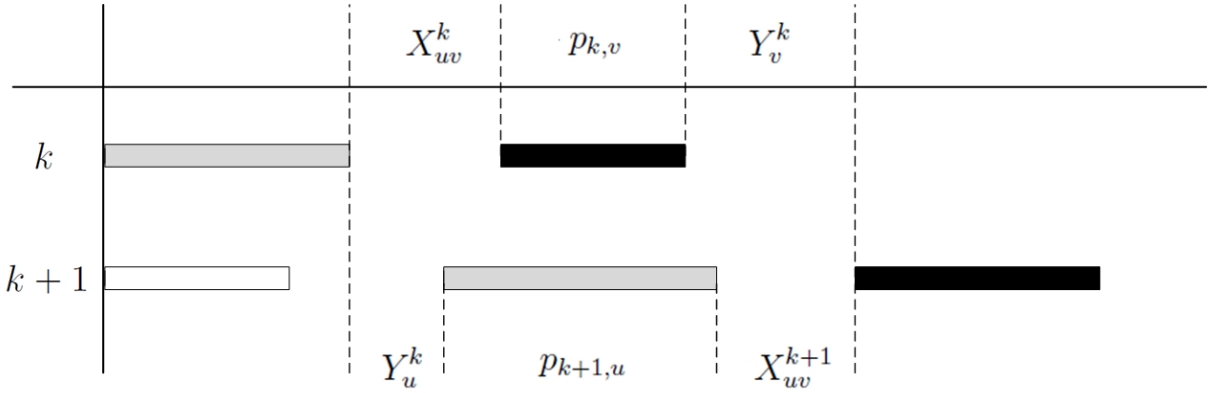


Figura 3: Ilustração das propriedades estruturais entre as tarefas $u$ e $v$. Fonte: Adaptado de Nagano e Moccellin (2002)

na máquina anterior; **2)** o fim do processamento da tarefa anterior na mesma máquina. Assim, tem-se a seguinte restrição:

$$\begin{cases} Se \quad Y_v^k > 0, \quad X_{u,v}^{k+1} = 0 \\ Se \quad X_{u,v}^{k+1} > 0, \quad Y_v^k = 0 \end{cases} \tag{26}$$

E também que:

$$X_{u,v}^k + p_{k,v} + Y_v^k = Y_u^k + p_{k+1,u} + X_{u,v}^{k+1} \tag{27}$$

Como $Y_u^k > 0$, pode-se eliminar este termo transformando (27) na desigualdade:

$$X_{u,v}^{k+1} - Y_v^k \leq X_{u,v}^k + p_{k,v} - p_{k+1,u} \tag{28}$$

Durante o cálculo de LBs para sequências parciais de nós ativos, apenas o primeiro termo de (25) com é obtido com exatidão, pois a ordem das tarefas na NPS e a tarefa corresponde à $n$ não são definidas. Pelas fórmulas de LBs mostradas anteriormente, nota-se que o os autores buscam valores mínimos para o segundo termo, enquanto o terceiro ($\sum Y_n^k$) é desconsiderado. Esta seção tem como objetivo encontrar limitantes para os dois termos ($X_{u,v}^{k+1}$ e $Y_v^k$), que em seguida serão aplicados na formulação de novos LBs de sequências parciais.

### 4.2.1 A Propriedade $UBX$

Primeiramente proposto por Moccellin (1995), um UB para o valor de $X_{u,v}^{k+1}$, $UBX_{u,v}^{k+1}$, é calculado a partir da relação em (26) aplicada à (28).

Quando $X_{u,v}^{k+1} > 0$:

$$X_{u,v}^{k+1} \leq X_{u,v}^{k} + p_{k,v} - p_{k+1,u} \tag{29}$$

Levando à seguinte fórmula recursiva:

$$UBX_{u,v}^{k+1} = (UBX_{u,v}^{k} + p_{k,v} - p_{k+1,u})^{+} \tag{30}$$

Com $UBX_{u,v}^{1} = 0$ como condição de contorno, pois não há tempo de ociosidade entre operações na primeira máquina.

### 4.2.2 A Propriedade $LBY$

A propriedade $LBY$ foi proposta por Nagano e Moccellin (2002) como parte do índice de uma heurística construtiva para o problema $F_m||C_{max}$, e permite, usando apenas um par de tarefas, calcular um LB para $Y_v^k$, ou seja, $LBY_{u,v}^k$.

Semelhantemente à dedução anterior, quando $Y_v^k > 0$:

$$Y_v^k \geq p_{k+1,u} - X_{u,v}^{k} - p_{k,v} \tag{31}$$

E um LB para $Y_{u,v}^k$ é:

$$LBY_{u,v}^{k} = (p_{k+1,u} - p_{k,v} - UBX_{u,v}^{k})^{+} \tag{32}$$

Desse modo, a propriedade $LBY$ torna-se útil para a formulação de LBs da seguinte forma: para qualquer tarefa $j \in J'$ na NPS, é possível encontrar um mínimo para tempo que $j$ deve esperar para iniciar seu processamento em $k$ após esta ser finalizada em $k-1$, $\min_{\substack{j' \in J' \\ j' \neq j}} (LBY_{j',j}^{k})$. Este valor é independente de qualquer sequenciamento prévio, e assim pode ser incorporado aos valores das caudas nos LBs, independentemente do critério de desempenho do problema.

Entretanto, duas observações devem ser feitas sobre esta aplicação do $LBY$:

- No cálculo de um LB para o *makespan* apenas a cauda da última tarefa é somada, logo a busca do mínimo $LBY$ é feita apenas no conjunto conjunto J'; já para critérios que consideram os tempos de finalização de todas as tarefas, como TFT e TT, esta busca deve incluir a última tarefa da PS, $s$, pois esta obrigatoriamente precederá uma tarefa não alocada.

- O valor de $LBY^k_{u,v}$ pode ser calculado no estágio de pré-processamento, já que, como mencionado, independe da sequência parcial. Entretanto, a busca do menor $LBY$ em $J'$ ou $J' \cup s$ adiciona um grau $n$ de complexidade computacional ao processamento do LB. Desse modo, é importante levar em consideração o *trade-off* entre custo computacional e valor numérico do LB, assim como observado por Rios-Mercado e Bard (1999).

Os LBs propostos neste trabalho possuem como principal característica a inclusão dos valores de $LBY^k_{u,v}$ nas caudas das tarefas, e os resultados são analisados de forma a verificar o tanto ganho de eficiência para a resolução de problemas quanto o *trade-off* entre tempo computacional e valor.

### 4.2.3 As Propriedades para PFSPs com Tempos de *Setup* Dependentes da Sequência

A propriedade $UBX$ pode ser estendida para problemas com a restrição $s^i_{j,l}$ e calculada de forma similar ao que é feito no PFSP convencional. Para esta aplicação, os tempos de *setup* são tratados como tarefas próprias, que seguem a ordenação entre as tarefas porém não obedecem as relações de precedência (ou seja, o *setup* entre duas tarefas na máquina $k+1$ pode ser iniciado antes do *setup* entre as mesmas tarefas na máquina $k$ ser finalizado). (MOCCELLIN; NAGANO, 2007)

Assim, para um par de tarefas subsequentes $u$ e $v$ e as atividades de *setup* $s^k_{u,v}$, a igualdade (27) pode ser replicada para os pares $u$-$s^k_{u,v}$ (33) e $s^k_{u,v}$-$v$ (34).

$$X^k_{u,s_{u,v}} + s^k_{u,v} + Y^k_{s_{u,v}} = Y^k_u + p_{k+1,u} + X^{k+1}_{u,s_{u,v}} \tag{33}$$

$$X^k_{s_{u,v},v} + p_{k,v} + Y^k_v = Y^k_{s_{u,v}} + s^{k+1}_{uv} + X^{k+1}_{s_{u,v},v} \tag{34}$$

Como as atividades de *setup* podem ser iniciadas assim que a operação da tarefa anterior termina, tem-se $X^k_{u,s_{u,v}} = X^{k+1}_{u,s_{u,v}} = 0$. Isolando o termo $Y^k_{s_{u,v}}$ em (33):

$$Y^k_{s_{u,v}} = Y^k_u + p_{k+1,u} - s^k_{u,v} \tag{35}$$

Neste caso, $Y^k_{s_{u,v}}$ pode assumir valores negativos devido ao fato de não haver precedência entre as operações de *setup*. Substituindo (35) em (34) e reescrevendo em função de $X^{k+1}_{s_{u,v},v}$ e $Y^k_v$:

$$X^{k+1}_{s_{u,v},v} - Y^k_v = X^k_{s_{u,v},v} + (p_{k,v} + s^k_{u,v}) - (p_{k+1,u} + s^{k+1}_{u,v}) - Y^k_u \tag{36}$$

Como $Y^k_u \geq 0$ e $UBX^k_{s_{u,v},v} \geq X^k_{s_{u,v},v}$:

$$X^{k+1}_{s_{u,v},v} - Y^k_v \leq UBX^k_{s_{u,v},v} + (p_{k,v} + s^k_{u,v}) - (p_{k+1,u} + s^{k+1}_{u,v}) \tag{37}$$

Semelhantemente ao que foi feito no problema sem tempos de *setup*, ao se aplicar as restrições de (26) em (37), obtém-se:

$$UBX^{k+1}_{s_{u,v},v} = (UBX^k_{s_{u,v},v} + (p_{k,v} + s^k_{u,v}) - (p_{k+1,u} + s^{k+1}_{u,v}))^+ \tag{38}$$

$$LBY^k_{u,v} = ((p_{k+1,u} + s^{k+1}_{u,v}) - (p_{k,v} + s^k_{u,v}) - UBX^k_{s_{u,v},v})^+ \tag{39}$$

Assim, a formulação de $LBY^k_{u,v}$ em (39) pode ser usada de forma semelhante à (32) para PFSPs com tempos de *setup* dependente.

## 4.3 Novos Limitantes Propostos

Nesta seção serão propostas novas formulações de LBs para os seguintes problemas:

- $F_m|s^i_{j,l}|\sum C$
- $F_m|s^i_{j,l}|\sum T$

### 4.3.1 Limitantes Propostos para $F_m|s^i_{j,l}|\sum C$

Os LBs apresentados nesta seção são baseados na formulação de Chung, Flynn e Kirca (2002) ($LB_{CFK}$), já que este foi visto como o mais apropriado para a adaptação para o problema $F_m|s^k_{j,l}|\sum C$. A principal contribuição de $LB_{CFK}$ é o cálculo dos termos $E_{k,[j]}$, que neste trabalho foram adaptados para contabilizar os tempos de *setup*. Neste caso, as tarefas em $J'$ também são ordenadas em SPT, e uma abordagem similar a de Takano e Nagano (2017) foi usada para a adaptação:

- **Passo 1:** os tempos mínimos de *setup* após cada tarefa em $T$, quando sucedidas pelas tarefas de $J'$, são encontrados.
- **Passo 2:** $s^k_{[0]}$ é o valor referente à tarefa na posição $s$, e os demais são classificados em ordem não-decrescente, com o $j$-ésimo termo da sequência sendo dado por $s^k_{[j]}$, com $1 \leq j \leq n - s$.

- **Passo 3:** para cada máquina $k$, o LB para o tempo de finalização da operação de *setup* que sucede a $[j]$-ésima tarefa da sequência SPT é calculado como $R'_{k,[j]} = R'_{k,[j-1]} + p_{k,[j]} + s^k_{[j]}$, com $R'_{k[0]} = q_k + s^k_{[0]}$.

- **Passo 4:** o LB para o tempo de início da tareda $[j]$ em $k$ é calculado usando $E'_{k,[j]} = \max(R'_{k,[j-1]}, R'_{k-1,[j-1]} + p_{k-1,[j]})$.

Usando as formulações dos passos 3 e 4, é garantido que $E'_{k[j]}$ inclua $j$ operações de *setup*, com a primeira sendo o *setup* de saída da sequência parcial $\sigma$. Ao se computar os menores tempos de *setup* e ordenálos em SPT, um LB para o TFT em $k$ é encontrado ao se alternar as duas sequências entre si (tempos de processamento e *setup*).

A segunda contribuição para as formulações é através a propriedade $LBY$, que pode ser aplicada de duas maneiras, como mostrado nas subseções seguintes:

#### 4.3.1.1 $LB_{FT1}$

A primeira formulação, $LB_{FT1}$, para cada máquina $1 \leq k \leq m$ a soma de:

- O tempo de fluxo das tarefas em $\sigma$.

- Os LBs dos tempos de início das tarefas de $J'$ em $k$ ($E'_{k,[j]}$).

- As caudas das tarefas de $J'$ de $k$ para $m$ ($\sum_{i=k}^{m} p_{i,[j]}$).

- Os LBs dos tempos de espera de cada tarefa em $J'$ entre as máquinas $k$ e $m$, através dos termos $LBY$. Como uma tarefa $j \in J'$ sucede imediatamente outra tarefa $j' \in T$, este valor é obtido pela menor soma $\sum_{i=k}^{m} LBY^i_{j',j}$.

A Equação 40 mostra a formulação completa de $LB_{FT1}$.

$$LB_{FT1} = \sum_{j=1}^{s} C_{m,j} + \max_{1 \leq k \leq m} \sum_{j=1}^{n-s} \left( E'_{k,[j]} + \sum_{i=k}^{m} p_{i,[j]} + \min_{\substack{j' \in T \\ j' \neq j}} \left( \sum_{i=k}^{m-1} LBY^k_{j',[j]} \right) \right) \tag{40}$$

Visando reduzir a complexidade dos cálculos, as caudas de cada tarefa ($\sum_{i=k}^{m} p_{i,[j]}$) são calculadas apenas uma vez durante a etapa de pré-processamento. Além disso, o LB utiliza os valores de $q_k$ ao invés de $q''_k$. Durante testes preliminares, foi observado que o valor ganho com a aplicação de $q''_k$ não permitia a pode de nós o suficiente para compensar o esforço adicional para calcular os termos.

De forma similar as caudas, todos os valores e somatórios de $LBY$são calculados e armazenados durante o pré-processamento. Entretanto, como o conjunto $T$ é único para cada nó da árvore, as buscas pelos menores valores são feitas $m \times |J'|$ vezes cada vez que $LB_{FT1}$ é calculado. Assim, excluindo-se as buscas pelos menores tempos de *setup* e as ordenações em SPT, $LB_{FT1}$ possui uma complexidade computacional $O(mn^2)$.

#### 4.3.1.2 $LB_{FT2}$

Uma alternativa para reduzir a complexidade computacional do uso do $LBY$, um segundo LB é proposto, eliminando a necessidade da busca pelos mínimos termos de $LBY$ em cada nó da árvore. Também durante o pré-processamento, a etapa de busca dos menores termos de $LBY$ feita em $LB_{FT1}$ é aplicada para toda tarefa em relação ao conjunto $J$. Assim, os seguintes valores são armazenados:

$$minLBY_j^k = \min_{\substack{j' \in J \\ j' \neq j}} \left( \sum_{i=k}^{m-1} LBY_{j',j}^k \right) \tag{41}$$

$LB_{FT2}$ é então calculado através da Equação 42.

$$LB_{FT2} = \sum_{j=1}^{s} C_{m,j} + \max_{1 \leq k \leq m} \sum_{j=1}^{n-s} \left( E'_{k[j]} + \sum_{i=k}^{m} p_{i,[j]} + minLBY_{[j]}^k \right) \tag{42}$$

A formulação acima (excluindo as buscas pelos mínimos *setups* e as ordenações SPT) possui uma complexidade computacional de $O(mn)$, já que a adição dos termos $minLBY$ requerem uma única operação cada. Entretanto, os valores somados desta maneira são sempre iguais ou menores que em $LB_{FT1}$. Para cada nó com $|PS| > 2$, $T$ é um subconjunto de $J$ ($T \subset J$), e o mínimo termo $LBY$ encontrado para $j' \in J$ pode estar designado para uma tarefa não contida em $T$. Desse modo, a Inequação 43 é válida.

$$\min_{\substack{j' \in T \\ j' \neq j}} \sum_{i=k}^{m-1} LBY_{j',j}^k \leq \min_{\substack{j' \in J \\ j' \neq j}} \sum_{i=k}^{m-1} LBY_{j',j}^k \tag{43}$$

Concluindo, em termos de valor total, ocorre a dominância $LB_{FT1} \geq LB_{TN2}$. Por outro lado, a complexidade de $LB_{FT1}$ é $O(mn^2)$, enquanto a de $LB_{FT2}$ é $O(mn)$. O *trade-off* entre o ganho de valor e o tempo computacional entre $LB_{FT1}$ e $LB_{FT2}$ é analisado no Capítulo 5, determinando qual das formulações é mais eficiente para o uso no algoritmo B&B.

#### 4.3.1.3 $LB_{FT1}$ e $LB_{FT2}$ são LBs válidos para o problema $F_m|s_{j,lk}^i|\sum C$

Seja $\pi^*(\sigma)$ a solução ótima local para um conjunto de sequências iniciadas com $\sigma$. Uma formulação completa do TFT para $\pi^*(\sigma)$, válida para todo $1 \leq k \leq m$, pode ser escrita usando os termos da Seção 4.2, $X_{u,v}^k$ and $Y_{u,v}^k$), como mostrado na Equação 44.

$$TFT = \sum_{j=1}^{n} C_{m,j} = \sum_{j=1}^{n} \left( \sum_{l=1}^{j} \left( s_{l-1,l}^k + X_{s_{l-1,l}l}^k + p_{kl} \right) + \sum_{i=k+1}^{m} \left( Y_{s_{j-1,j},j}^{k-1} + p_{kj} \right) \right) \tag{44}$$

A Equação 44 pode ser reescrita de modo similar às formulações dos LBs separando-se as tarefas em $\sigma$.

$$TFT = \sum_{j=1}^{s} C_{m,j} + \sum_{j=s+1}^{n} \left( q_k + \sum_{l=s+1}^{j} \left( s_{l-1,l}^{k} + X_{s_{l-1,l},l}^{k} + p_{k,l} \right) + \sum_{i=k}^{m-1} Y_{s_{j-1,j},j}^{k} + \sum_{i=k+1}^{m} p_{k,j} \right) \tag{45}$$

Em cada nó da árvore do B&B, a ordem das tarefas em $J'$ não é conhecida. Sabendo disso, é possível provar que $LB_{FT1}$, e consequentemente $LB_{FT2}$, são limitantes do TFT de $\pi^*(\sigma)$ comparando-se as Equações 40 e 45. Para isso, três partes das formulações são comparadas:

- O início da tarefa na [j]-ésima posição:

$$E'_{k,[j-s]} \leq q_k + \sum_{l=s+1}^{j-1} \left( s_{l-1,l}^{k} + X_{s_{l-1,l},l}^{k} + p_{k,l} \right) + \left( s_{j-1,j}^{k} + X_{s_{j-1,j},j}^{k} \right) \tag{46}$$

- A soma dos tempos de espera das tarefas em $J'$, da máquina $k$ até $m$.

$$\sum_{j=1}^{n-s} \left( \min_{\substack{j' \in T \\ j' \neq j}} \left( \sum_{i=k}^{m-1} LBY_{j',[j]}^{k} \right) \right) \leq \sum_{j=s+1}^{n} \left( \sum_{i=k}^{m-1} Y_{s_{j-1,j},j}^{k} \right) \tag{47}$$

- A soma das caudas das tarefas em $J'$. Como estes valores não dependem da sequência, os somatórios são iguais.

$$\sum_{j=1}^{n-s} \left( \sum_{i=k}^{m} p_{i,[j]} \right) = \sum_{j=s+1}^{n} \left( \sum_{i=k}^{m} p_{k,j} \right) \tag{48}$$

Substituindo 46, 47 and 48 em 45:

$$TFT \geq \sum_{j=1}^{s} C_{m,j} + \sum_{j=s+1}^{n} \left( E'_{k,[j-s]} + \sum_{i=k}^{m} p_{i,[j-s]} + \min_{\substack{j' \in T \\ j' \neq j}} \left( \sum_{i=k}^{m-1} LBY_{j',[j-s]}^{k} \right) \right) \tag{49}$$

O termo do lado direito da Equação 49 é a formulação de $LB_{FT1}$ ao se considerar $k$ como gargalo. Assim, $LB_{FT1}$ é um LB válido para o TFT deste ótimo local, e, devido à dominância já mostrada anteriormente, $LB_{TN2}$ também é.

### 4.3.2 Limitantes Propostos para $F_m|s_{j,l}^{i}|\sum T$

Para adaptar o LB de Chung, Flynn e Kirca (2006) para $F_m|s_{jk}^{i}|\sum T$, foram feitas as mesmas alterações nos cálculos de $R_{k,[j]}$ e $E_{k,[j]}$ para computar os tempos de *setup*. Dois LBs para TT são propostos, utilizando a mesma lógica apresentada na seção anterior.

#### 4.3.2.1 $LB_{TT1}$

A inclusão da propriedade $LBY$ na formulação ocorre no momento do cálculo dos termos $h^1_{k,j}$, e é feita antes da ordenação destes em valores não-decrescentes. Utilizado a primeira abordagem, o cálculo de $h^1_{k,j}$ para $LB_{TT1}$ é:

$$h^1_{k,j} = d_j - \sum_{i=k}^{m} p_{i,j} - \min_{\substack{j' \in T \\ j' \neq j}} (\sum_{i=k}^{m-1} LBY^i_{j',j}) \tag{50}$$

E a formulação final de $LB_{TT1}$ é apresentada na Equação .

$$LB_{TT1} = \sum_{j=1}^{s} \left( C_{m,j} - d_j \right)^+ + \max_{1 \leq k \leq m} (\sum_{j=1}^{n-s} \left( E'_{k,[j]} - h^1_{k,[j]} \right)^+ ) \tag{51}$$

#### 4.3.2.2 $LB_{TT2}$

Já com a segunda abordagem, para $LB_{TT2}$, o termo $h^2_{k,j}$ é calculado com o conjunto de busca para $LBY$ sendo $J$ ao invés de $T$:

$$h^2_{kj} = d_j - \sum_{i=k}^{m} p_{i,j} - minLBY^k_j \tag{52}$$

Assim, o LB é calculado como mostrado na Equação 53

$$LB_{TT2} = \sum_{j=1}^{s} \left( C_{m,j} - d_j \right)^+ + \max_{1 \leq k \leq m} (\sum_{j=1}^{n-s} \left( E'_{k,[j]} - h^2_{k,[j]} \right)^+ ) \tag{53}$$

Como os termos $LBY$ não se alteram em relação à seção anterior, a inequação 43 mantém-se válida, e por isso a dominância $LB_{TT1} > LB_{TT2}$ também ocorre.

#### 4.3.2.3 $LB_{TT1}$ e $LB_{TT2}$ são LBs válidos para o problema $F_m|s^i_{jk}|\sum T$

De forma semelhante ao que foi feito na Subseção 4.3.1.3, deve-se mostrar que $LB_{TT1}$ e, por consequência, $LB_{TT2}$ são valores menores ou iguais ao Atraso Total de uma sequência.

Seja $\pi^*(\sigma)$ um ótimo local para o problema $F_m|s^i_{jk}|\sum T$, considerando todas as sequências iniciadas or $\sigma$. O atraso da tarefa na tarefa que ocupa a posição $j > s$ é:

$$TT(j^{\pi^*(\sigma)}) = \left( q_k + \sum_{l=s+1}^{j} \left( s^k_{l-1,l} + X^k_{s_{l-1,l},l} + p_{kl} \right) + \sum_{i=k}^{m-1} Y^k_{s_{j-1,j},j} + \sum_{i=k+1}^{m} p_{k,j} - d_j \right)^+ \tag{54}$$

A expressão acima pode ser simplificada aplicando as Substituições 55 e 56.

$$A_{k,j} = \left( q_k + \sum_{l=s+1}^{j-1} \left( s^k_{l-1,l} + X^k_{s_{l-1,l},l} + p_{k,l} \right) + \left( s^k_{j-1,j} + X^k_{s_{j-1,j},j} \right) \right) \tag{55}$$

$$H_{k,j} = \sum_{i=k}^{m-1} Y^k_{s_{j-1,j},j} + \sum_{i=k}^{m} p_{k,j} \tag{56}$$

$$TT(j^{\pi^*(\sigma)}) = \left(A_{k,j} - \left(d_j - H_{k,j}\right)\right)^+ \tag{57}$$

Para a tarefa na posição $j > s$, por definição, $E'_{k,[j]} \leq A_{k,j}$. Já para qualquer tarefa $j \in J'$, $h^1_{k,j} \geq d_j - H_{k,j}$. Como as ordenações não-decrescentes de $E'_{k,j}$ e $h^1_{k,j}$ são independentes, a Inequação 58 é valida para qualquer tarefa $j \in J'$.

$$TT(j^{\pi^*(\sigma)}) \geq \left(E'_{k,[j]} - h^1_{k,[j]})\right)^+ \tag{58}$$

Como último passo para mostrar a validade dos LBs, é necessário provar que as ordenações não-decrescentes minimizam o termo $\left(E'_{k,[j]} - h^1_{k,[j]})\right)^+$. Esta prova foi feita por Kim (1995) e é detalhada a seguir.

Duas tarefas são definidas, $i$ e $j$, com tempos de finalização $C_i \leq C_j$ e *due dates* $d_i < d_j$. O TT deste par é dado por $T = (C_i - d_i)^+ + (C_j - d_j)^+$. Considerando um atraso com as *due dates* trocadas, $T' = (C_i - d_j)^+ + (C_j - d_i)^+$, o LB é validado mostrando que $T \leq T'$ em todos os possíveis casos.

- **Caso 1:** $C_j \geq C_i \geq d_j > d_i$. $T = T' = C_i - d_i + C_j - d_j$.
- **Caso 2:** $C_j \geq d_j \geq C_i \geq d_i$. $T = C_i - d_i + C_j - d_j$ e $T' = C_j - d_i$, ou seja, $T = C_i - d_j + T'$. Como $C_i - d_j \leq 0$, $T \leq T'$.
- **Caso 3:** $C_j \geq d_j > d_i \geq C_i$. $T = C_j - d_j$ e $T' = C_j - d_i$. Como $d_j > d_i$, $T < T'$.
- **Caso 4:** $d_j \geq C_j \geq C_i \geq d_i$. $T = C_i - d_i$ e $T' = C_j - d_i$. Como $C_i \leq C_j$, $T \leq T'$.
- **Caso 5:** $d_j \geq C_j \geq d_i \geq C_i$. $T = 0$ e $T' = C_j - d_i$, logo $T \leq T'$.
- **Caso 6:** $d_j > d_i \geq C_j \geq C_i$. $T = T' = 0$

## 4.4 Considerações

Na primeira parte deste capítulo, foi feita uma revisão de LBs encontrados na literatura que possuem relação com os problemas estudados neste trabalho (Seção 4.1) e de uma propriedade estrutural do PFSP que permite o cálculo de um LB para o tempo de espera de uma tarefa entre máquinas (Seção 4.2). Na segunda parte (4.3), foram apresentados dois LBs para o problema $F_m|s^i_{j,l}|\sum C$ ($LB_{FT1}$ e $LB_{FT2}$) e dois para o problema $F_m|s^i_{j,l}|\sum T$ ($LB_{TT1}$ e $LB_{TT2}$).

Os LBs propostos são baseados nos conceitos apresentados por Chung, Flynn e Kirca (2002) e Chung, Flynn e Kirca (2006), e os tempos de *setup* são incorporados nas formulações utilizando uma abordagem baseada no trabalho de Takano e Nagano (2017). Além disso, a propriedade $LBY$, inicialmente proposta por Nagano e Moccellin (2002),

é utilizada para aprimorar o valor obtido com as formulações. Dois métodos de aplicação desta propriedade são propostos, cada um com uma complexidade computacional e uma qualidade de solução. No capítulo seguinte, as quatro formulações propostas são analisadas.

Capítulo 5

# Experimentação Computacional e Resultados

O desempenho de um LB é analisado considerando-se a eficiência do B&B quando este é utilizado e também seu valor absoluto em relação à solução ótima. Para isso os testes com a aplicação das formulações nos algoritmos serão extensivamente estudados para analisar estes dois fatores, bem como a eficiência do método para resolver problemas com diferentes tamanhos e distribuições de tempos de *setup*.

## 5.1 Metodologia utilizada

Nesta série de testes, o B&B é aplicado em um conjunto de problemas de diversos tamanhos, com um limite máximo de 3.600 segundos para encontrar uma solução ótima usando as técnicas apresentadas no Capítulo 3. Para cada problema resolvido, são computados dois valores: o número total de nós explorados e o tempo computacional gasto. Os principais objetivos destes testes são verificar a viabilidade do uso dos LBs no algoritmo B&B e analisar o *trade-off* entre complexidade computacional e valor do LB a partir do desempenho do algoritmo.

Devido às formulações dos LBs, a criação de nós é feita adicionando-se uma tarefa por vez na sequência parcial para cada nó criado. Um UB inicial é encontrado antes da execução do B&B através da heurística $NEH$. Como o objetivo dos testes é avaliar o desempenho dos LBs durante a execução do B&B, regras de dominância para os problemas não são utilizadas.

No que se refere às regras de busca, duas observações puderam ser feitas sobre as regras de busca:

- Utilizando-se a regra de *Melhor Limitante*, o B&B é capaz de resolver um número maior de instâncias do que com a Regra de *Profundidade*, explorando um número menor de nós e, na maioria dos casos, com um tempo computacional menor. Entre-

tanto, em instâncias maiores, o LB não é forte o suficiente para podar um número alto de nós, causando dois problemas: falta de memória RAM devido ao alto número de nós ativos, interrompendo a execução do programa; gasto excessivo de tempo computacional para buscar o nó de menor LB em toda a árvore, problema que ainda persistiu após melhorias na estrutura de armazenamento. Matematicamente, o número de nós ativos pode atingir uma quantidade da ordem de $n!$.

- A regra de *Profundidade* permite que o B&B mantenha um número de nós ativos menor que $\frac{n^2}{2}$, não havendo assim problemas com memória. Além disso, o algoritmo explora uma quantidade de nós maior, pois esta regra permite uma estrutura de armazenamento de nós simples. Entretanto, foi percebida uma queda brusca no desempenho em termos de número de problemas resolvidos a partir de uma certa quantidade de tarefas, o que não ocorreu com a regra *Melhor Limitante*.

Para este trabalho, foi escolhida a regra de *Profundidade* pois, durante testes preliminares com a regra de *Melhor Limitante* ocorreram os problemas de falta de memória, interrompendo a execução dos algoritmos. Foi testada também uma regra *Híbrida*, a qual ramifica por *Melhor Limitante* até a busca atingir um nó com $\frac{n}{2}$ tarefas sequenciadas, que era então ramificado por *Profundidade*, retornando à árvore no máximo um único nó folha. Entretanto, esta ideia foi descartada pois os problemas de memória persistiram.

Como notação, os algoritmos B&B usando $LB_{TF1}$ e $LB_{TF2}$ são identificados como $B\&B_{TF1}$ e $B\&B_{TF2}$, respectivamente. Do mesmo modo, os algoritmos usando $LB_{TT1}$ e $LB_{TT2}$ são $B\&B_{TT1}$ e $B\&B_{TT2}$.

### 5.1.1 Instâncias

Na experimentação foi usado o conjunto de instâncias gerado por Ronconi e Armentano (2001), sendo originalmente usado para *flowshop* com bloqueio. Entretanto, esta base de dados se mostrou ideal para este trabalho por conter uma quantidade suficiente de problemas de variados tamanhos, com $10 \leq n \leq 20$ e $2 \leq m \leq 10$, e 20 instâncias únicas para cada combinação, totalizando 540 problemas. Os tempos de processamento das tarefas não números inteiros gerados aleatoriamente, seguindo uma distribuição uniforme no intervalo $[1, 99]$. A Tabela 6 mostra as combinações de $n$ e $m$ do conjunto de problemas.

#### 5.1.1.1 Geração dos tempos de *setup*

Para um estudo mais detalhado do desempenho dos LBs propostos em relação aos tempos de *setup*, estes tempos são gerados em 4 diferentes distribuição, conforme feito por Ruiz, Maroto e Alcaraz (2005). Em cada faixa de distribuição, os tempos de *setup* são gerados aleatoriamente entre 1 e um valor correspondente a uma porcentagem do máximo

| $m$ | 10 | 12 | 14 | 16 | 18 | 20 |
|---|---|---|---|---|---|---|
| 2 | 20 | 20 | 20 | 20 | 20 | 20 |
| 3 | 20 | 20 | 20 | 20 | 20 | 20 |
| 4 | 20 | 20 | 20 | 20 | 20 | 20 |
| 5 | 20 | 20 | 20 | - | - | - |
| 7 | 20 | 20 | 20 | - | - | - |
| 10 | 20 | 20 | 20 | - | - | - |

Tabela 6: Número de instâncias do banco de dados de Ronconi e Armentano (2001) para cada tamanho de problema

tempo de processamento (99). A Tabela 7 mostra as diferentes faixas de distribuição dos tempos de *setup* e a porcentagem considerada.

| Grupo | % | Mínimo | Máximo |
|---|---|---|---|
| 1 | 10 | 1 | 9 |
| 2 | 50 | 1 | 49 |
| 3 | 100 | 1 | 99 |
| 4 | 125 | 1 | 124 |

Tabela 7: Faixa de distribuição dos tempos de *setup*, baseado nas distribuições de Ruiz, Maroto e Alcaraz (2005)

#### 5.1.1.2 Geração das *due dates*

A geração de *due dates* segue o mesmo procedimento usado por Ronconi e Armentano (2001), em que dois parâmetros independentes e um termo referente aos tempos de processamento são usados para definir a faixa de distribuição das datas:

- $P$ **- Limitante inferior do *makespan* do conjunto de tarefas:** usado como referência para valor médio da faixa de distribuição, é calculado a partir de uma fórmula similar ao LB de Brown e Lomnicki (1966), como mostra a Equação 59. Esta formulação foi adaptada de forma a incluir também os tempos de *setup*.

$$P = \max\left(\max_{1\leq k\leq m}\left(\min_{j}\sum_{i=1}^{k-1} p_{ij} + \sum_{j=1}^{n} p_{kj} + S_k + \min_{j}\sum_{i=k+1}^{m} p_{ij}\right), \max_{j}\sum_{k=1}^{m} p_{kj}\right) \quad (59)$$

Com:

$$S_k = \frac{\sum_{j\in J}\left(\sum_{\substack{j'\in 0,J\\ j'\neq j}}\left(s^k_{j'j}\right)\right)}{n^2} \quad (60)$$

- $T$ **- Fator de rigorosidade dos prazos de entrega:** este parâmetro desloca a média da faixa de distribuição das *due dates*. Quando maior seu valor, menor esta média e mais rigorosos os prazos de entrega.

- $R$ **- Fator de distribuição dos prazos de entrega:** este parâmetro determina o tamanho do intervalo de distribuição das *due dates*. Quanto maior seu valor, maior a faixa na qual são gerados os prazos.

Com estes termos definidos, as *due dates* são geradas aleatoriamente em distribuição uniforme na faixa de valores inteiros entre $P(1 - T - \frac{R}{2})$ e $P(1 - T + \frac{R}{2})$. Este trabalho utiliza quatro combinações diferentes de valores de $T$ e $R$, semelhantemente ao que foi feito por Ronconi e Armentano (2001). A Tabela 8 mostra os valores de cada grupo.

| Grupo | $T$ | $R$ |
|---|---|---|
| 1 | 0,2 | 0,6 |
| 2 | 0,2 | 1,2 |
| 3 | 0,4 | 0,6 |
| 4 | 0,4 | 1,2 |

Tabela 8: Parâmetros para geração de *due dates*, baseado nas distribuições de Ronconi e Armentano (2001)

### 5.1.2 Especificações computacionais

Todos os experimentos descritos nesta seção foram escritos em C++ e executados em um computador físico Intel® Core™ i5-2500K com 3.30GHz, 4GB de RAM e Windows 7.

## 5.2 Resultados para $F_m|s^i_{j,l}|\sum C$

A Tabela 9 mostra o número de problemas resolvidos por $B\&B_{FT1}$ e $B\&B_{FT2}$. Todas as instâncias com 16 ou menos tarefas foram otimamente resolvidas dentro do tempo limite estipulado, porém notou-se uma drástica queda na eficiência do algoritmo para $n = 20, m = 4$. No total, o $B\&B_{FT1}$ resolveu 2.057 (95.8%) problemas, enquanto $B\&B_{FT2}$, 2.080 (96.3%). Isso mostra que, mesmo com $LB_{TN1}$ dominando $LB_{TN2}$ em termos de valor, a complexidade computacional tem um efeito mais significativo na eficiência do método do que a diferença da soma dos termos $LBY$.

Nas análises seguintes serão considerados apenas os problemas resolvidos por ambos os algoritmos. Como todos os problemas resolvidos por $B\&B_{FT1}$ também foram por $B\&B_{FT2}$, o número de instâncias consideradas é o número de problemas resolvidos por $B\&B_{FT1}$ mostrado na Tabela 9.

| $n$ | $m$ | $B\&B_{FT1}$ | $B\&B_{FT2}$ | $B\&B_{FT1}$ | $B\&B_{FT2}$ |
|---|---|---|---|---|---|
| | | *Setup* 1 | | *Setup* 2 | |
| 10 | 2 - 10 | 120 | 120 | 120 | 120 |
| 12 | 2 - 10 | 120 | 120 | 120 | 120 |
| 14 | 2 - 10 | 120 | 120 | 120 | 120 |
| 16 | 2 - 4 | 60 | 60 | 60 | 60 |
| 18 | 2 | 20 | 20 | 20 | 20 |
| | 3 | 20 | 20 | 20 | 20 |
| | 4 | 20 | 20 | 19 | 20 |
| 20 | 2 | 20 | 20 | 20 | 20 |
| | 3 | 17 | 17 | 10 | 14 |
| | 4 | 8 | 10 | 3 | 5 |
| Total | | 525 | 527 | 512 | 519 |
| | | *Setup* 3 | | *Setup* 4 | |
| 10 | 2 - 10 | 120 | 120 | 120 | 120 |
| 12 | 2 - 10 | 120 | 120 | 120 | 120 |
| 14 | 2 - 10 | 120 | 120 | 120 | 120 |
| 16 | 2 - 4 | 60 | 60 | 60 | 60 |
| 18 | 2 | 20 | 20 | 20 | 20 |
| | 3 | 20 | 20 | 20 | 20 |
| | 4 | 18 | 19 | 19 | 20 |
| 20 | 2 | 20 | 20 | 20 | 20 |
| | 3 | 12 | 15 | 9 | 13 |
| | 4 | 1 | 3 | 1 | 4 |
| Total | | 511 | 517 | 509 | 517 |

Tabela 9: Número de problemas resolvidos por $B\&B_{FT1}$ e $B\&B_{FT2}$

A primeira análise referente aos problemas que foram otimamente resolvidos é relacionada à quantidade de nós explorados. a Tabela 10 mostra a média do número de nós criados pelo algoritmo B&B para encontrar a solução ótima dentro de cada tamanho de problemas. No geral, $B\&B_{FT1}$ explorou um total de nós aproximadamente 0.33% a menos do que $B\&B_{FT2}$. A maior diferença ocorreu no Grupo de *Setup* 1, com uma diferença média de 0.90%. Além disso, foi observado que esta diferença aumentou com o número de máquinas, chegando a 5.00% em grupos de problemas específicos.

Considerando os valores absolutos, a observação mais significante é o alto aumento de nós explorados entre os Grupos de *Setup* 1 e 2. Isto mostra que a duração relativa dos tempos de *setup* possui um impacto direto na eficiência dos LBs.

Os tempos médios gastos pelo B&B para resolver os problemas é mostrado na Tabela 11. Como o tempo computacional é proporcional ao número de nós explorados, é notado um padrão semelhante ao da Tabela 10, com os valores mais altos sendo referentes às instâncias com mais tarefas e maiores tempos de *setup*.

Apesar de $LB_{FT1}$ explorar menos nós, o tempo computacional do B&B usando este LB

| $n$ | $m$ | *Setup* 1 | | *Setup* 2 | | *Setup* 3 | | *Setup* 4 | |
|---|---|---|---|---|---|---|---|---|---|
| | | $B\&B_{FT1}$ | $B\&B_{FT2}$ | $B\&B_{FT1}$ | $B\&B_{FT2}$ | $B\&B_{FT1}$ | $B\&B_{FT2}$ | $B\&B_{FT1}$ | $B\&B_{FT2}$ |
| 10 | 2 | 1.165 | 1.167 | 2.572 | 2.579 | 2.938 | 2.941 | 2.555 | 2.556 |
| | 3 | 2.731 | 2.762 | 4.751 | 4.772 | 6.452 | 6.461 | 8.185 | 8.205 |
| | 4 | 3.277 | 3.364 | 7.653 | 7.718 | 10.585 | 10.632 | 10.422 | 10.460 |
| | 5 | 6.018 | 6.261 | 10.736 | 10.930 | 12.053 | 12.149 | 15.909 | 16.018 |
| | 7 | 3.299 | 3.500 | 10.503 | 10.671 | 16.241 | 16.386 | 22.130 | 22.292 |
| | 10 | 4.692 | 4.988 | 11.691 | 11.961 | 21.985 | 22.378 | 26.226 | 26.594 |
| 12 | 2 | 14.442 | 14.445 | 24.214 | 24.230 | 32.327 | 32.338 | 22.973 | 22.981 |
| | 3 | 7.350 | 7.388 | 30.795 | 30.859 | 44.615 | 44.679 | 53.245 | 53.411 |
| | 4 | 37.107 | 37.917 | 70.904 | 71.689 | 89.827 | 90.294 | 107.852 | 108.281 |
| | 5 | 22.995 | 23.673 | 72.069 | 72.824 | 132.494 | 133.396 | 147.242 | 148.128 |
| | 7 | 36.863 | 40.524 | 155.038 | 160.838 | 237.412 | 240.161 | 295.551 | 298.332 |
| | 10 | 48.740 | 50.958 | 172.323 | 175.263 | 337.358 | 341.026 | 439.279 | 446.127 |
| 14 | 2 | 74.074 | 74.075 | 162.171 | 162.214 | 128.571 | 128.599 | 167.628 | 167.661 |
| | 3 | 220.059 | 220.667 | 438.798 | 440.152 | 424.721 | 424.979 | 386.389 | 386.736 |
| | 4 | 407.233 | 428.354 | 900.564 | 911.083 | 1.085.543 | 1.092.355 | 1.197.709 | 1.201.598 |
| | 5 | 469.769 | 499.296 | 965.190 | 983.850 | 1.308.318 | 1.313.396 | 2.364.706 | 2.375.022 |
| | 7 | 543.877 | 566.004 | 2.092.986 | 2.123.739 | 3.698.846 | 3.723.013 | 3.801.886 | 3.819.592 |
| | 10 | 730.578 | 780.931 | 2.868.482 | 2.925.756 | 5.982.499 | 6.032.540 | 8.937.493 | 8.989.007 |
| 16 | 2 | 542.224 | 542.353 | 726.988 | 727.265 | 1.104.059 | 1.104.324 | 964.984 | 965.107 |
| | 3 | 1.619.646 | 1.639.121 | 5.640.054 | 5.651.690 | 4.287.755 | 4.294.534 | 3.007.060 | 3.010.457 |
| | 4 | 1.013.075 | 1.038.496 | 4.901.405 | 4.925.458 | 7.572.693 | 7.595.020 | 13.326.189 | 13.355.007 |
| 18 | 2 | 9.524.059 | 9.524.166 | 9.409.643 | 9.411.463 | 6.442.345 | 6.443.429 | 6.520.227 | 6.520.610 |
| | 3 | 11.423.910 | 11.485.665 | 58.669.013 | 58.800.570 | 46.910.396 | 46.959.810 | 33.331.115 | 33.343.623 |
| | 4 | 27.132.387 | 27.675.408 | 87.470.310 | 88.019.422 | 94.106.769 | 94.381.152 | 118.307.521 | 118.584.807 |
| 20 | 2 | 64.232.446 | 64.234.247 | 58.087.647 | 58.097.839 | 45.753.596 | 45.755.313 | 49.925.606 | 49.927.241 |
| | 3 | 78.610.008 | 79.037.972 | 138.460.625 | 138.876.813 | 169.692.689 | 169.756.803 | 197.053.633 | 197.095.214 |
| | 4 | 73.308.648 | 75.329.654 | 116.502.056 | 116.575.415 | 229.038.282 | 229.135.861 | 77.124.590 | 77.127.250 |

Tabela 10: Número médio de nós criados por $B\&B_{FT1}$ e $B\&B_{FT2}$ ao resolverem os problemas.

foi na média 35% maior do que o B&B usando $LB_{FT2}$, considerando somente os problemas resolvidos pelos dois algoritmos. O número de máquinas também teve um impacto na diferença de tempo, aumentando-a diretamente.

Assim, o *trade-off* entre o valor numérico do LB e o tempo computacional requerido para seu cálculo favorece $LB_{FT2}$, o qual apresenta uma menor complexidade, apesar de um valor final menor.

Por fim, para confirmar esta superioridade do $B\&B_{FT2}$ em relação ao tempo, foi calculada a porcentagem de sucesso deste método em relação a $B\&B_{FT1}$. Esta porcentagem equivale ao número de problemas em que $B\&B_{FT2}$ resolveu em um tempo menor que $B\&B_{FT1}$, dividido pelo número total. Como, após verificação, em todas as 2.057 instâncias $B\&B_{FT2}$ teve um tempo menor ou igual a $B\&B_{FT1}$, os casos em que os tempos foram iguais não foram considerados como sucesso.

A Tabela 12 mostra as porcentagens divididas por tamanho de instância e grupos de *setup*. No geral, a porcentagem de sucesso de $B\&B_{FT2}$ foi de 52%, com a grande maioria destes casos em instâncias de maior tamanho. Foi observada também uma ocorrência mais frequente de casos de sucesso em instâncias de pequeno e médio tamanho com o aumento médio dos tempos de *setup*. Entretanto, o baixo número de amostras não permite uma análise detalhada sobre esta tendência.

| $n$ | $m$ | *Setup* 1 | | *Setup* 2 | | *Setup* 3 | | *Setup* 4 | |
|---|---|---|---|---|---|---|---|---|---|
| | | $B\&B_{FT1}$ | $B\&B_{FT2}$ | $B\&B_{FT1}$ | $B\&B_{FT2}$ | $B\&B_{FT1}$ | $B\&B_{FT2}$ | $B\&B_{FT1}$ | $B\&B_{FT2}$ |
| 10 | 2 | - | - | - | - | - | - | - | - |
| | 3 | - | - | - | - | - | - | - | - |
| | 4 | - | - | - | - | - | - | - | - |
| | 5 | - | - | - | - | - | - | - | - |
| | 7 | - | - | - | - | - | - | - | - |
| | 10 | - | - | - | - | - | - | - | - |
| 12 | 2 | - | - | - | - | - | - | - | - |
| | 3 | - | - | - | - | - | - | - | - |
| | 4 | - | - | - | - | - | - | 1 | - |
| | 5 | - | - | - | - | 1 | 1 | 1 | 1 |
| | 7 | - | - | 1 | 1 | 2 | 1 | 3 | 2 |
| | 10 | 1 | - | 2 | 1 | 4 | 3 | 5 | 3 |
| 14 | 2 | - | - | 1 | - | - | - | 1 | - |
| | 3 | 1 | 1 | 2 | 1 | 2 | 1 | 2 | 1 |
| | 4 | 3 | 2 | 6 | 4 | 8 | 5 | 9 | 5 |
| | 5 | 4 | 3 | 8 | 5 | 12 | 7 | 20 | 12 |
| | 7 | 6 | 4 | 24 | 15 | 42 | 26 | 45 | 27 |
| | 10 | 12 | 8 | 45 | 27 | 96 | 57 | 141 | 82 |
| 16 | 2 | 2 | 2 | 3 | 2 | 5 | 3 | 4 | 3 |
| | 3 | 10 | 7 | 36 | 24 | 29 | 19 | 22 | 14 |
| | 4 | 8 | 5 | 43 | 27 | 68 | 43 | 123 | 76 |
| 18 | 2 | 46 | 33 | 50 | 36 | 35 | 25 | 36 | 26 |
| | 3 | 90 | 59 | 462 | 305 | 399 | 261 | 292 | 187 |
| | 4 | 286 | 181 | 934 | 592 | 1,059 | 662 | 1.386 | 849 |
| 20 | 2 | 370 | 263 | 368 | 265 | 306 | 219 | 335 | 234 |
| | 3 | 754 | 488 | 1.400 | 917 | 1.790 | 1.155 | 2.202 | 1.391 |
| | 4 | 1.012 | 671 | 1.713 | 1.104 | 3.416 | 2.117 | 1.331 | 806 |

Tabela 11: Tempo médio (em segundos) gasto para o algoritmo por $B\&B_{FT1}$ e $B\&B_{FT2}$ para resolver os problemas

É importante observar que o cálculo foi feito com os tempos arredondados para segundos, pois a máquina usada para os experimentos é sujeita à oscilações de desempenho, e uma análise usando os tempos em milissegundos pode apresentar resultados incorretos devido a estas oscilações. O estudo da porcnetagem de sucesso foi feito somente em relação ao tempo computacional pois este é o critério com mais importância em aplicações práticas e pelo fato de a diferença entre o número de nós, como mostrado previamente na Tabela 10.

## 5.3 Resultados para $F_m|s^i_{j,l}|\sum T$

Primeiramente, devido ao alto número de instâncias criadas para a experimentação com TT ($540 \times 4 \times 4 = 8640$), os resultados nesta seção serão apresentados de duas maneiras, uma de acordo com os tempos de *setup* e *due date*, e a outra de acordo com o número de tarefas e máquinas.

As Tabelas 13 e Tabela 14 mostram os números de instâncias resolvidas por $B\&B_{TT1}$ e

| *Setup* | $n$ | $m$ | | | | | |
|---|---|---|---|---|---|---|---|
| | | 2 | 3 | 4 | 5 | 7 | 10 |
| 1 | 10 | 0% | 0% | 0% | 0% | 0% | 0% |
| | 12 | 0% | 0% | 5% | 5% | 15% | 45% |
| | 14 | 10% | 20% | 45% | 65% | 90% | 100% |
| | 16 | 35% | 85% | 90% | | | |
| | 18 | 95% | 100% | 100% | | | |
| | 20 | 95% | 100% | 100% | | | |
| 2 | 10 | 0% | 0% | 0% | 0% | 0% | 0% |
| | 12 | 0% | 0% | 20% | 40% | 40% | 85% |
| | 14 | 15% | 55% | 95% | 100% | 100% | 100% |
| | 16 | 65% | 95% | 100% | | | |
| | 18 | 100% | 100% | 100% | | | |
| | 20 | 100% | 100% | 100% | | | |
| 3 | 10 | 0% | 0% | 0% | 0% | 0% | 5% |
| | 12 | 0% | 0% | 15% | 40% | 75% | 100% |
| | 14 | 10% | 70% | 100% | 100% | 100% | 100% |
| | 16 | 90% | 100% | 100% | | | |
| | 18 | 100% | 100% | 100% | | | |
| | 20 | 100% | 100% | 100% | | | |
| 4 | 10 | 0% | 0% | 0% | 0% | 0% | 5% |
| | 12 | 0% | 5% | 35% | 55% | 70% | 95% |
| | 14 | 15% | 45% | 100% | 100% | 100% | 100% |
| | 16 | 70% | 100% | 100% | | | |
| | 18 | 100% | 100% | 100% | | | |
| | 20 | 100% | 100% | 100% | | | |

Tabela 12: % de sucesso de $B\&B_{FT2}$ sobre $B\&B_{FT1}$ em relação ao tempo computacional gasto para resolver as instâncias

$B\&B_{TT2}$ com estas duas divisões. No geral $B\&B_{TT1}$ foi capaz de resolver 7.755 instâncias (89.8%) dentro do limite de 1 hora, enquanto $B\&B_{TT2}$ resolveu 7.903 (91.6%). Apesar da quantidade relativa de problemas solucionados ser menor que o observado com TFT, a diferença entre os dois métodos ficou mais evidente neste caso. Todos os problemas resolvidos por $B\&B_{TT1}$ também foram resolvidos por $B\&B_{TT2}$.

Pela Tabela 13 nota-se uma diferença no desempenho do B&B conforme a alteração dos parâmetros das instâncias. Neste caso, a ocorrência de maiores tempos de *setup* causou um leve aumento no número de problemas resolvidos, principalmente entre os grupos 2 e 3. Também observou-se um melhor desempenho para $R = 1, 2$ (grupos 2 e 4) e $T = 0, 4$ (grupos 3 e 4). Isso permite concluir que o B&B foi mais eficiente para resolver instâncias com maior ocorrência e maiores valores de atrasos nas tarefas.

De fato, uma análise de regressão mostrou que os parâmetros $R$ e $T$ possuem uma influência significativa no número de problemas resolvidos, com o $R$ tendo um peso maior. Esta análise também mostrou que a distribuição dos tempos de *setup* não influenciou tanto

o desempenho do algoritmo quanto os parâmetros de *due date*.

Fazendo-se um estudo passo-a-passo da execução do algoritmo em uma instância com *due dates* relaxadas e tempos de *setup* curtos (grupos 1 para os dois parâmetros), foi observado que os LBs de diversos nós eram iguais a zero, já que a solução ótima muitas vezes era próxima de zero, o que dificultava a poda de nós por meio do UB. Um efeito contrário foi observado em uma instância dos dois grupos 4: com *due dates* mais rígidas os LBs assumiam valores mais altos, e mais nós eram podados.

| *Setup* | *Due date* | $B\&B_{TT1}$ | $B\&B_{TT2}$ |
|---|---|---|---|
| 1 | 1 | 425 | 429 |
| | 2 | 497 | 499 |
| | 3 | 476 | 482 |
| | 4 | 526 | 527 |
| Total | | 1.924 | 1.937 |
| 2 | 1 | 441 | 453 |
| | 2 | 514 | 522 |
| | 3 | 476 | 488 |
| | 4 | 495 | 504 |
| Total | | 1.926 | 1.967 |
| 3 | 1 | 467 | 476 |
| | 2 | 491 | 499 |
| | 3 | 492 | 509 |
| | 4 | 502 | 514 |
| Total | | 1.952 | 1.998 |
| 4 | 1 | 468 | 483 |
| | 2 | 483 | 501 |
| | 3 | 499 | 507 |
| | 4 | 503 | 510 |
| Total | | 1.953 | 2.001 |
| Total Geral | | 7.755 | 7.903 |

Tabela 13: Número de problemas resolvidos por $B\&B_{TT1}$ e $B\&B_{TT2}$, divididos pelos grupos de distribuição de *setup* e *due date*.

Ao se analisar a quantidade de problemas solucionados por cada B&B em respeito ao tamanho das instâncias, observa-se uma tendência similar ao ocorrido com o TFT. A Tabela 14 mostra que todos os problemas com até 14 tarefas foram resolvidos pelos dois métodos, e o desempenho de ambos caiu de acordo com o aumento de tarefas e máquinas.

As Tabelas 15 e 16 mostram a média de nós explorados e tempo computacional gasto por cada B&B com as instâncias resolvidas. As tabelas mostram os resultados agrupados por grupos de *setup* e *due date* (Tabela 15) e tamanho (Tabela 16). Novamente foram considerados apenas os dados referentes às instâncias resolvidas por ambos os B&Bs, ou seja, resolvidas por $B\&B_{TT1}$.

| $n$ | $m$ | $B\&B_{TT1}$ | $B\&B_{TT2}$ |
|---|---|---|---|
| 10 | 2 - 10 | 1920 | 1920 |
| 12 | 2 - 10 | 1920 | 1920 |
| 14 | 2 - 10 | 1920 | 1920 |
| 16 | 2 | 312 | 313 |
| | 3 | 316 | 316 |
| | 4 | 319 | 319 |
| 18 | 2 | 279 | 286 |
| | 3 | 278 | 292 |
| | 4 | 217 | 264 |
| 20 | 2 | 152 | 190 |
| | 3 | 78 | 110 |
| | 4 | 44 | 53 |
| Total Geral | | 7755 | 7903 |

Tabela 14: Número de problemas resolvidos por $B\&B_{TT1}$ e $B\&B_{TT2}$, divididos por número de tarefas e máquinas

Em nenhum dos cenários mostrados a diferença de nós entre os dois métodos foi maior que 1.8%, com a média geral sendo 0.15%. Isto mostra que, apesar da diferença na complexidade computacional entre os dois LBs, a diferença entre o valor final dos dois é significantemente baixa.

Por outro lado, a diferença ente o tempo computacional gasto pelos B&Bs é de aproximadamente 40%, maior do que entre $LB_{FT1}$ e $LB_{FT2}$. Assim, semelhantemente ao problema anterior, a formulação menos complexa ($LB_{TT2}$) é mais eficiente quando aplicada no B&B, pois a diferença entre valores finais (tendo que $LB_{TT1}$ matematicamente domina $LB_{TT2}$), não é alta o suficiente para podar nós adicionais e compensar o tempo adicional requerido para o cálculo do LB.

Por fim, o mesmo teste de porcentagem de sucesso é feito para $B\&B_{TT1}$ e $B\&B_{TT2}$ em relação ao tempo computacional, e as porcentagens são agrupadas também pelos parâmetros de *setup* e *due date* (Tabela 17) e tamanho dos problemas (Tabela 18).

$B\&B_{TT2}$ apresentou uma porcentagem de sucesso de 56% em relação à $B\&B_{TT1}$, com maiores ocorrências em instâncias com maiores tempos de *setup* e número de tarefas. Novamente, mostra-se que a quantidade de nós podados a mais por $LB_{TT1}$ não é o suficiente para compensar o tempo adicional necessário para seu cálculo, principalmente em instâncias em que o B&B requer a exploração de um alto número de nós.

## 5.4 Considerações

Nesta seção foi realizada uma experimentação para analisar o desempenho de cada um dos LBs dentro do algoritmo B&B. Todos os experimentos foram realizados nas mes-

| *Setup* | *Due date* | Nós Criados | | Tempo (s) | |
|---|---|---|---|---|---|
| | | $B\&B_{TT1}$ | $B\&B_{TT2}$ | $B\&B_{TT1}$ | $B\&B_{TT2}$ |
| 1 | 1 | 33.304.122 | 33.412.121 | 148 | 93 |
| | 2 | 29.329.989 | 29.341.531 | 56 | 37 |
| | 3 | 19.002.396 | 19.053.135 | 173 | 108 |
| | 4 | 7.884.258 | 7.909.154 | 84 | 53 |
| Média | | 21.789.770 | 21.835.966 | 113 | 71 |
| 2 | 1 | 21.629.723 | 21.659.297 | 157 | 99 |
| | 2 | 14.808.804 | 14.828.289 | 119 | 73 |
| | 3 | 13.472.467 | 13.505.333 | 137 | 79 |
| | 4 | 14.532.147 | 14.569.135 | 153 | 89 |
| Média | | 15.969.232 | 15.998.832 | 141 | 85 |
| 3 | 1 | 19.381.796 | 19.398.391 | 189 | 116 |
| | 2 | 15.863.207 | 15.876.613 | 169 | 97 |
| | 3 | 14.013.123 | 14.030.752 | 155 | 90 |
| | 4 | 15.563.023 | 15.579.393 | 160 | 96 |
| Média | | 16.161.489 | 16.177.485 | 168 | 99 |
| 4 | 1 | 14.655.521 | 14.665.580 | 149 | 91 |
| | 2 | 14.008.299 | 14.018.148 | 150 | 86 |
| | 3 | 15.469.572 | 15.481.996 | 179 | 108 |
| | 4 | 14.735.226 | 14.749.611 | 176 | 106 |
| Média | | 14.723.977 | 14.735.703 | 164 | 98 |
| Média Geral | | 17.148.087 | 17.173.879 | 147 | 88 |

Tabela 15: Média de nós criados e tempo computacional (em segundos) gasto por $B\&B_{TT1}$ e $B\&B_{TT2}$ para resolver os problemas, divididos pelos grupos de distribuição de *setup* e *due date*

mas condições, tanto em relação às especificações de *hardware* como aos componentes do algoritmo.

O banco de dados de Ronconi e Armentano (2001) foi usado como base para criar os problemas aos quais o B&B foi aplicado. Usando parâmetros encontrados na literatura, tempos de *setup* e *due dates* foram geradas em diferentes distribuições para testar variações de desempenho em relação a estes dois fatores.

Os resultados da experimentação mostraram que os LBs podem ser aplicados com sucesso para resolver ambos os problemas estudados, $F_m|s_{j,l}^k|\sum C$ e $F_m|s_{j,l}^k|\sum T$. Entretanto, existe um *trade-off* evidente entre as formulações propostas, com os $LB$s mais simples computacionalmente ($LB_{FT2}$ e $LB_{TT2}$) são mais eficientes que seus respectivos pares ($LB_{FT1}$ e $LB_{TT1}$), que possuem um valor final maior ou igual, porém uma complexidade computacional maior. Esta diferença, no geral, torna-se mais evidente ao se analisar os dados relativos ao tempo computacional gasto pelos algoritmos, com ambos $B\&B_{FT2}$ e $B\&B_{TT2}$ resolvendo problemas mais rapidamente, mesmo explorando uma quantidade de nós marginalmente maior, principalmente nas instâncias com tempos de *setup* mais

| $n$ | $m$ | Nós Criados | | Tempo (s) | |
|---|---|---|---|---|---|
| | | $B\&B_{TT1}$ | $B\&B_{TT2}$ | $B\&B_{TT1}$ | $B\&B_{TT2}$ |
| 10 | 2 | 6.785 | 6.789 | - | - |
| | 3 | 7.512 | 7.537 | - | - |
| | 4 | 10.991 | 11.063 | - | - |
| | 5 | 14.084 | 14.269 | - | - |
| | 7 | 14.374 | 14.514 | - | - |
| | 10 | 17.694 | 18.009 | 0 | - |
| 12 | 2 | 133.604 | 133.649 | 0 | 0 |
| | 3 | 79.500 | 79.623 | 0 | 0 |
| | 4 | 116.737 | 117.522 | 1 | 0 |
| | 5 | 140.640 | 141.400 | 1 | 1 |
| | 7 | 188.564 | 191.945 | 2 | 1 |
| | 10 | 240.041 | 243.201 | 3 | 2 |
| 14 | 2 | 4.057.341 | 4.059.885 | 9 | 6 |
| | 3 | 1.032.356 | 1.034.458 | 6 | 4 |
| | 4 | 1.330.733 | 1.340.696 | 10 | 6 |
| | 5 | 1.684.233 | 1.714.088 | 16 | 10 |
| | 7 | 2.600.052 | 2.620.577 | 34 | 21 |
| | 10 | 4.745.445 | 4.790.475 | 83 | 52 |
| 16 | 2 | 31.155.165 | 31.158.797 | 103 | 66 |
| | 3 | 30.869.760 | 30.971.324 | 158 | 99 |
| | 4 | 15.975.516 | 16.044.558 | 143 | 87 |
| 18 | 2 | 86.260.992 | 86.275.414 | 465 | 285 |
| | 3 | 94.682.518 | 94.755.961 | 850 | 511 |
| | 4 | 96.296.639 | 96.625.031 | 1.278 | 761 |
| 20 | 2 | 133.875.382 | 133.896.582 | 1.113 | 657 |
| | 3 | 117.314.808 | 117.339.636 | 1.357 | 804 |
| | 4 | 53.927.390 | 54.059.692 | 842 | 517 |
| Média Geral | | 17.148.087 | 17.173.879 | 147 | 88 |

Tabela 16: Média de nós criados e tempo computacional (em segundos) gasto por $B\&B_{TT1}$ e $B\&B_{TT2}$ para resolver os problemas, divididos por número de tarefas e máquinas

| *Setup* | *Due Date* | | | |
|---|---|---|---|---|
| | 1 | 2 | 3 | 4 |
| 1 | 51% | 40% | 50% | 45% |
| 2 | 58% | 52% | 57% | 56% |
| 3 | 63% | 64% | 60% | 59% |
| 4 | 62% | 62% | 62% | 62% |

Tabela 17: % de sucesso de $B\&B_{TT2}$ sobre $B\&B_{TT1}$ em relação ao tempo computacional gasto para resolver as instâncias, divididos entre os grupos de *setup* e *due date*

longos e número de tarefas e máquinas maiores.

Uma última observação relevante foi a maior eficiência do método $B\&B_{TT}$ (usando ambos os LBs) para resolver problemas com *due dates* mais rígidas. Em instâncias em

| $n$ | $m$ | | | | | |
|---|---|---|---|---|---|---|
| | 2 | 3 | 4 | 5 | 7 | 10 |
| 10 | 0% | 0% | 0% | 0% | 0% | 2% |
| 12 | 7% | 19% | 33% | 40% | 53% | 77% |
| 14 | 73% | 82% | 89% | 92% | 96% | 98% |
| 16 | 95% | 95% | 97% | | | |
| 18 | 97% | 98% | 99% | | | |
| 20 | 99% | 99% | 100% | | | |

Tabela 18: % de sucesso de $B\&B_{TT2}$ sobre $B\&B_{TT1}$ em relação ao tempo computacional gasto para resolver as instâncias, divididos entre os números de tarefas e máquinas

que atrasos ocorrem com mais frequência e possuem valores significantemente altos, o algorítimo é capaz de podar mais nós do que em instâncias em que a maioria das tarefas podem ser entregues antes do prazo.

Capítulo 6

# Conclusão

Este trabalho teve como objetivo propor novos LBs para TFT e TT em problemas de *flowshop* com tempos de *setup* dependentes da sequência. Estes LBs são utilizados como principais componentes do método B&B, que obtém soluções ótimas para os problemas em questão. Duas formulações foram propostas para cada caso, buscando maximizar a eficiência do algorítimo.

Uma extensa revisão literária foi feita referente à aplicação do método B&B em PFSPs, com o objetivo de analisar os seguintes pontos: os tipos de problemas abordados; a justificativa para a aplicação do B&B; e a forma como o algoritmo é usado. A análise, feita no Capítulo 3 mostrou que o B&B é o método exato mais eficiente atualmente, superando a resolução de problemas diretamente através do MILP, e também que os autores aplicam o algoritmo de uma forma sistemática, focando em regras de dominância e cálculo de LBs fortes para aumentar a eficiência do método.

Em relação aos problemas estudados, notou-se uma vasta exploração de diferentes combinações de restrições e critérios de desempenho, principalmente em problemas com 2 máquinas. Foram encontrados também buscando analisar o desempenho de diferentes técnicas e componentes do B&B, como tipos de ramificação e regras de busca. A lacuna de conhecimento mais evidente na revisão foi a pequena quantidade de artigos abordando problemas com tempos de *setup*, com destaque para o caso de *setup* dependente, presente em apenas 6 dos artigos revisados.

Esta lacuna motivou a definição do escopo deste trabalho para os dois problemas em questão, já que não foram encontradas formulações de LBs para ambos os problemas, $F_m|s^k_{j,l}|\sum C$ e $F_m|s^k_{j,l}|\sum T$, com os tempos de *setup* não relacionados à famílias de produtos.

No Capítulo 4 foram estudados em detalhe LBs de problemas relacionados, tendo como objetivo encontrar formulações para servirem de base para as propostas neste trabalho. Este estudo foi dividido em 3 partes: a primeira incluiu LBs para problemas com *setup* dependente da sequência, a segunda, LBs para o TFT e a terceira, para o TT. Em todos os casos foram considerados apenas problemas com um número generalizado de $m$

máquinas.

Com base nos conceitos identificados na revisão e no desenvolvimento da propriedade $LBY$ (um LB para o tempo de espera de uma tarefa entre máquinas), foram propostos dois LBs para cada um dos problemas. As formulações foram feitas usando os princípios de Chung, Flynn e Kirca (2002) e Chung, Flynn e Kirca (2006), com a adição dos termos referentes aos tempos de *setup* de maneira similar ao feito por Takano e Nagano (2017). A diferença entre as formulações surgiu da aplicação do termo referente ao $LBY$ nas caudas das tarefas.

A primeira formulação necessita de uma etapa de busca para este termo exclusiva para cada nó, elevando a complexidade computacional do cálculo, porém resultando em um valor mais elevado. A segunda não realiza esta etapa de busca, com a única adição do termo $LBY$ sendo na etapa de pré-processamento, simplificando o cálculo com o custo da resultar em um valor menor. Devido a este *trade-off* entre complexidade e valor, a experimentação computacional for usada também para poder definir qual formulação é mais eficiente ao ser aplicada no algoritmo B&B.

A experimentação foi feita usando um banco de dados inicial de 540 tarefas, modificado para incluir várias combinações de distribuições de tempos de *setup* e *due dates*. Os resultados mostraram que a segunda proposta de LBs é mais eficiente que a primeira em todos os casos, devido à menor complexidade computacional. Os critérios utilizados foram tempo de processamento necessário para o B&B resolver os problemas, quantidade de nós explorados. Considerando todos os casos nos dois critérios de desempenho, $B\&B_{FT2}$ e $B\&B_{TT2}$ resolveram os problemas em menor tempo, e explorando uma quantidade de nós apenas marginalmente maior que $B\&B_{FT1}$ e $B\&B_{TT1}$ respectivamente.

Outra observação relevante em relação aos resultados foi a maior facilidade do B&B em resolver problemas com *due dates* mais rígidas, ou seja, em que o prazo de entrega das tarefas é fator determinante para o sequenciamento como um todo.

Este trabalho contribui para a literatura ao propor um método exato para dois problemas com grande relevância para aplicações práticas, e que ainda não foram, até a data atual, abordados desta maneira. As formulações propostas mostram uma maneira viável de inclusão dos tempos de *setup dependente* nas formulações de LBs para TFT e TT, e também incluem um termo referente ao tempo de espera das tarefas entre máquinas.

Por fim, são destacados dois pontos como sugestões para trabalhos futuros:

- Novas propostas para os termos referentes aos tempos de *setup*: foi observada uma queda de desempenho do B&B para instâncias com tempos de *setup* mais longos, decorrente da formulação que escolhe sempre o menor valor entre um conjunto. Uma formulação usando conceitos de ATSP, similar ao trabalho de Rios-Mercado e Bard (1999), adaptado para TFT e TT pode ser viável dependendo de sua complexidade computacional.

- Maior uso de pré-processamento: grande parte do tempo requerido para o cálculo dos limitantes decorre dos cálculos dos termos dos tempos de de *setup*. O uso de técnicas computacionais para fazer buscas e ordenações já na etapa de pré-processamento e manter os resultados para diferentes conjuntos aumentará a eficiência dos algoritmos consideravelmente. O mesmo pode ser feito para o termo $LBY$ em $LB_{TFT1}$ e $LB_{TT1}$.

# Referências Bibliográficas


AGNETIS, A.; ROSSI, F.; GRISTINA, G. An exact algorithm for the batch sequencing problem in a two-machine flow shop with limited buffer. **Naval Research Logistics**, 1998. v. 45, n. 2, p. 141–164, 1998.

AHMADI, R. H.; BAGCHI, U. Improved lower bounds for minimizing the sum of completion times of n jobs over m machines in a flow shop. **European Journal of Operational Research**, 1990. v. 44, n. 3, p. 331–336, 1990.

AKKAN, C.; KARABATI, S. The two-machine flowshop total completion time problem: Improved lower bounds and a branch-and-bound algorithm. **European Journal of Operational Research**, 2004. v. 159, n. 2, p. 420 – 429, 2004.

ALDOWAISAN, T. A new heuristic and dominance relations for no-wait flowshops with setups. **Computers & Operations Research**, 2001. v. 28, n. 6, p. 563 – 584, 2001.

ALLAHVERDI, A. Minimizing mean flowtime in a two-machine flowshop with sequence-independent setup times. **Computers & Operations Research**, 2000. v. 27, n. 2, p. 111–127, 2000.

ALLAHVERDI, A. The tricriteria two-machine flowshop scheduling problem. **International Transactions in Operational Research**, 2001. Blackwell Publishers Ltd, v. 8, n. 4, p. 403–425, 2001.

ALLAHVERDI, A. The third comprehensive survey on scheduling problems with setup times/costs. **European Journal of Operational Research**, 2015. v. 246, n. 2, p. 345 – 378, 2015.

ALLAHVERDI, A.; AL-ANZI, F. S. A branch-and-bound algorithm for three-machine flowshop scheduling problem to minimize total completion time with separate setup times. **European Journal of Operational Research**, 2006. v. 169, n. 3, p. 767–780, 2006.

ALLAHVERDI, A.; ALDOWAISAN, T. No-wait flowshops with bicriteria of makespan and total completion time. **The Journal of the Operational Research Society**, 2002. Palgrave Macmillan Journals, v. 53, n. 9, p. 1004–1015, 2002.

ALLAHVERDI, A.; ALDOWAISAN, T. No-wait flowshops with bicriteria of makespan and maximum lateness. **European Journal of Operational Research**, 2004. v. 152, n. 1, p. 132 – 147, 2004.

ALLAOUI, H.; ARTIBA, A. Scheduling two-stage hybrid flow shop with availability constraints. **Computers & Operations Research**, 2006. v. 33, n. 5, p. 1399 – 1419, 2006.

AN, Y.-J.; KIM, Y.-D.; CHOI, S.-W. Minimizing makespan in a two-machine flowshop with a limited waiting time constraint and sequence-dependent setup times. **Computers & Operations Research**, 2016. v. 71, p. 127 – 136, 2016.

ARDAKAN, M. A.; HAKIMIAN, A.; REZVAN, M. T. A branch-and-bound algorithm for minimising the number of tardy jobs in a two-machine flow-shop problem with release dates. **International Journal of Computer Integrated Manufacturing**, 2014. v. 27, n. 6, p. 519–528, 2014.

BAGGA, P. C.; BHAMBANI, A. Bicriteria in special flowshop problems. **Indian Journal of Pure and Applied Mathematics**, 2002. v. 33, n. 4, p. 435–442, 2002.

BAI, D. et al. Permutation flow shop scheduling problem to minimize nonlinear objective function with release dates. **Comput. Ind. Eng.**, 2017. v. 112, n. C, p. 336–347, 2017.

BALASUBRAMANIAN, J.; GROSSMANN, I. A novel branch and bound algorithm for scheduling flowshop plants with uncertain processing times. **Computers & Chemical Engineering**, 2002. v. 26, n. 1, p. 41 – 57, 2002.

BANK, M. et al. Two-machine flow shop total tardiness scheduling problem with deteriorating jobs. **Applied Mathematical Modelling**, 2012. v. 36, n. 11, p. 5418 – 5426, 2012.

BANSAL, S. P. Minimizing the sum of completion times of n jobs over m machines in a flowshop: A branch and bound approach. **A I I E Transactions**, 1977. v. 9, n. 3, p. 306–311, 1977.

BESTWICK, P. F.; HASTINGS, N. A. A new bound for machine scheduling. **Operational Research Quarterly**, 1976. v. 27, n. 2, p. 479–487, 1976.

BOUQUARD, J.-L.; LENTÉ, C. Two-machine flow shop scheduling problems with minimal and maximal delays. **4OR**, 2006. v. 4, n. 1, p. 15–28, Mar 2006.

BROWN, A. P. G.; LOMNICKI, Z. A. Some applications of the "branch-and-bound"algorithm to the machine scheduling problem. **OR**, 1966. Operational Research Society, v. 17, n. 2, p. 173–186, 1966.

BULFIN, R.; M'HALLAH, R. Minimizing the weighted number of tardy jobs on a two-machine flow shop. **Computers & Operations Research**, 2003. v. 30, n. 12, p. 1887 – 1900, 2003.

CARLIER, J.; REBAI, I. Two branch and bound algorithms for the permutation flow shop problem. **European Journal of Operational Research**, 1996. v. 90, n. 2, p. 238 – 251, 1996.

CHEN, P.; WU, C. C.; LEE, W. C. A bi-criteria two-machine flowshop scheduling problem with a learning effect. **The Journal of the Operational Research Society**, 2006. Palgrave Macmillan Journals, v. 57, n. 9, p. 1113–1125, 2006.

CHENG, J.; KISE, H.; MATSUMOTO, H. A branch-and-bound algorithm with fuzzy inference for a permutation flowshop scheduling problem. **European Journal of Operational Research**, 1997. v. 96, n. 3, p. 578 – 590, 1997.

CHENG, J.; STEINER, G.; STEPHENSON, P. A computational study with a new algorithm for the three-machine permutation flow-shop problem with release times. **European Journal of Operational Research**, 2001. v. 130, n. 3, p. 559 – 575, 2001.

CHENG, J.; STEINER, G.; STEPHENSON, P. Fast algorithms to minimize the makespan or maximum lateness in the two-machine flow shop with release times. **Journal of Scheduling**, 2002. v. 5, n. 1, p. 71–92, 2002.

CHENG, J.-L.; KISE, H.; KARUNO, Y. Optimal scheduling for an automated m-machine flowshop. **Journal of the Operations Research Society of Japan**, 1997. v. 40, n. 3, p. 356–372, 1997.

CHENG, M. et al. Bicriteria hierarchical optimization of two-machine flow shop scheduling problem with time-dependent deteriorating jobs. **European Journal of Operational Research**, 2014. v. 234, n. 3, p. 650 – 657, 2014.

CHENG, M. et al. Two-machine flow shop scheduling with deteriorating jobs: minimizing the weighted sum of makespan and total completion time. **Journal of the Operational Research Society**, 2015. v. 66, n. 5, p. 709–719, 2015.

CHENG, S.-R. et al. A two-machine flowshop scheduling problem with precedence constraint on two jobs. **Soft Computing**, 2017. v. 21, n. 8, p. 2091–2103, 2017.

CHENG, T.; JANIAK, A. A permutation flow-shop scheduling problem with convex models of operation processing times. **Annals of Operations Research**, 2000. v. 96, p. 39–60, 2000.

CHENG, T. et al. Two-machine flowshop scheduling with a truncated learning function to minimize the makespan. **International Journal of Production Economics**, 2013. v. 141, n. 1, p. 79 – 86, 2013.

CHIHAOUI, F. B. et al. No-wait scheduling of a two-machine flow-shop to minimise the makespan under non-availability constraints and different release dates. **International Journal of Production Research**, 2011. v. 49, n. 21, p. 6273–6286, 2011.

CHUNG, C. S.; FLYNN, J.; KIRCA, O. A branch and bound algorithm to minimize the total flow time for m-machine permutation flowshop problems. **International Journal of Production Economics**, 2002. v. 79, n. 3, p. 185–196, 2002.

CHUNG, C. S.; FLYNN, J.; KIRCA, O. A branch and bound algorithm to minimize the total tardiness for m-machine permutation flowshop problems. **European Journal of Operational Research**, 2006. v. 174, n. 1, p. 1–10, 2006.

CHUNG, Y.-H.; TONG, L.-I. Makespan minimization for m-machine permutation flowshop scheduling problem with learning considerations. **The International Journal of Advanced Manufacturing Technology**, 2011. v. 56, n. 1, p. 355–367, Sep 2011.

CHUNG, Y. H.; TONG, L. I. Bi-criteria minimization for the permutation flowshop scheduling problem with machine-based learning effects. **Computers and Industrial Engineering**, 2012. v. 63, n. 1, p. 303–312, 2012.

COMPANYS, R. Note on an improved branch-and-bound algorithm to solve n/m/p/fmax problems. **Top**, 1999. v. 7, n. 1, p. 25–31, Jun 1999.

COMPANYS, R.; MATEO, M. Different behaviour of a double branch-and-bound algorithm on fm|prmu|cmax and fm|block|cmax problems. **Computers & Operations Research**, 2007. v. 34, n. 4, p. 938 – 953, 2007.

CROCE, F. D.; GHIRARDI, M.; TADEI, R. An improved branch-and-bound algorithm for the two machine total completion time flow shop problem. **European Journal of Operational Research**, 2002. v. 139, n. 2, p. 293 – 301, 2002.

CROCE, F. D.; GUPTA, J. N.; TADEI, R. Minimizing tardy jobs in a flowshop with common due date. **European Journal of Operational Research**, 2000. v. 120, n. 2, p. 375 – 381, 2000.

CROCE, F. D.; NARAYAN, V.; TADEI, R. The two-machine total completion time flow shop problem. **European Journal of Operational Research**, 1996. v. 90, n. 2, p. 227 – 237, 1996.

CROCE, F. D.; T'KINDT, V. Improving the preemptive bound for the one-machine dynamic total completion time scheduling problem. **Operations Research Letters**, 2003. v. 31, n. 2, p. 142 – 148, 2003.

DETIENNE, B.; SADYKOV, R.; TANAKA, S. The two-machine flowshop total completion time problem: Branch-and-bound algorithms based on network-flow formulation. **European Journal of Operational Research**, 2016. v. 252, n. 3, p. 750 – 760, 2016.

DILEEPAN, P.; SEN, T. Job lateness in a two-machine flowshop with setup times separated. **Computers & Operations Research**, 1991. v. 18, n. 6, p. 549 – 556, 1991.

DROZDOWSKI, M. et al. Grid branch-and-bound for permutation flowshop. In: _____. **Parallel Processing and Applied Mathematics: 9th International Conference, PPAM 2011, Torun, Poland, September 11-14, 2011. Revised Selected Papers, Part II**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012. p. 21–30. ISBN 978-3-642-31500-8.

FONDREVELLE, J.; ALLAHVERDI, A.; OULAMARA, A. Two-machine, no-wait flowshop scheduling problem to minimize maximum lateness with separate set-up and removal times. 2005. v. 8, p. 165–174, 01 2005.

FONDREVELLE, J.; OULAMARA, A.; PORTMANN, M. C. Permutation flowshop scheduling problems with maximal and minimal time lags. **Computers and Operations Research**, 2006. v. 33, n. 6, p. 1540–1556, 2006.

FONDREVELLE, J. et al. Permutation flowshops with exact time lags to minimize maximum lateness. **International Journal of Production Research**, 2009. Taylor & Francis, v. 47, n. 23, p. 6759–6775, out. 2009.

GHARBI, A. et al. Polynomial lower bounds for the two-machine flowshop problem with sequence-independent setup times. **Electronic Notes in Discrete Mathematics**, 2010. v. 36, p. 1089 – 1096, 2010.

GHARBI, A. et al. The two-machine flowshop scheduling problem with sequence-independent setup times: New lower bounding strategies. **European Journal of Operational Research**, 2013. v. 231, n. 1, p. 69 – 78, 2013.

GHARBI, A.; MAHJOUBI, A. New lower bounds for flow shop scheduling. **International Journal of Humanities and Management Sciences**, 2013. v. 1, n. 1, p. 28–32, 2013.

GIM, B.; CURRY, G. L.; DEUERMEYER, B. L. Two-machine flow-shop sequencing with sparse precedence constraints. **Comput. Ind. Eng.**, 1994. v. 26, n. 1, p. 173–180, jan. 1994.

GOWRISHANKAR, K.; RAJENDRAN, C.; SRINIVASAN, G. Flow shop scheduling algorithms for minimizing the completion time variance and the sum of squares of completion time deviations from a common due date. **European Journal of Operational Research**, 2001. v. 132, n. 3, p. 643–665, 2001.

GRABOWSKI, J.; SKUBALSKA, E.; SMUTNICKI, C. On flow shop scheduling with release and due dates to minimize maximum lateness. **The Journal of the Operational Research Society**, 1983. Palgrave Macmillan Journals, v. 34, n. 7, p. 615–620, 1983.

GRAHAM, R. et al. Optimization and approximation in deterministic sequencing and scheduling: a survey. **Annals of Discrete Mathematics**, 1979. v. 5, p. 287 – 326, 1979.

HAOUARI, M.; KHARBECHE, M. An assignment-based lower bound for a class of two-machine flow shop problems. **Computers & Operations Research**, 2013. v. 40, n. 7, p. 1693 – 1699, 2013.

HAOUARI, M.; LADHARI, T. Minimising maximum lateness in a two-machine flowshop. **The Journal of the Operational Research Society**, 2000. Palgrave Macmillan Journals, v. 51, n. 9, p. 1100–1106, 2000.

HAOUARI, M.; LADHARI, T. A branch-and-bound-based local search method for the flow shop problem. **The Journal of the Operational Research Society**, 2003. v. 54, n. 10, p. 1076–1084, 2003.

HAOUARI, M.; LADHARI, T. Minimizing maximum lateness in a flow shop subject to release dates. **The Journal of the Operational Research Society**, 2007. Palgrave Macmillan Journals, v. 58, n. 1, p. 62–72, 2007.

HARIRI, A.; POTTS, C. A branch and bound algorithm to minimize the number of late jobs in a permutation flow-shop. **European Journal of Operational Research**, 1989. v. 38, n. 2, p. 228 – 237, 1989.

HE, H. Minimization of maximum lateness in an m-machine permutation flow shop with a general exponential learning effect. **Computers and Industrial Engineering**, 2016. v. 97, p. 73–83, 2016.

HENDIZADEH, S. H.; ELMEKKAWY, T. Y.; WANG, G. G. Bi-criteria scheduling of a flowshop manufacturing cell with sequence dependent setup times. **European Journal of Industrial Engineering**, 2007. v. 1, n. 4, p. 391–413, 2007.

HNAIEN, F.; YALAOUI, F.; MHADHBI, A. Makespan minimization on a two-machine flowshop with an availability constraint on the first machine. **International Journal of Production Economics**, 2015. v. 164, p. 95 – 104, 2015.

HOOGEVEEN, H.; NORDEN, L. van; VELDE, S. van de. Lower bounds for minimizing total completion time in a two-machine flow shop. **Journal of Scheduling**, 2006. v. 9, n. 6, p. 559–568, 2006.

IGNALL, E.; SCHRAGE, L. Application of the branch and bound technique to some flow-shop scheduling problems. **Operations Research**, 1965. v. 13, n. 3, p. 400–412, 1965.

JAFARI, A.-A. et al. A note on minimizing makespan in three machine flow shop with deteriorating jobs. **Computers & Operations Research**, 2016. v. 72, p. 93 – 96, 2016.

JAFARI, A.-A. et al. A note on on three-machine flow shop scheduling with deteriorating jobs. **International Journal of Production Economics**, 2017. v. 191, p. 250 – 252, 2017.

JOHNSON, S. M. Optimal two and three stage production schedules with set-up time included. **Naval Research Logistics Quarterly**, 1954. v. 1, n. 1, p. 61–68, 1954.

JOO, B.-J.; KIM, Y.-D. A branch-and-bound algorithm for a two-machine flowshop scheduling problem with limited waiting time constraints. **Journal of the Operational Research Society**, 2009. v. 60, n. 4, p. 572–582, Apr 2009.

KARABATI, S.; KOUVELIS, P. The permutation flow shop problem with sum-of-completion times performance criterion. **Naval Research Logistics (NRL)**, 1993. Wiley Subscription Services, Inc., A Wiley Company, v. 40, n. 6, p. 843–862, 1993.

KIM, H.-J.; LEE, J.-H. A branch and bound algorithm for three-machine flow shop with overlapping waiting time constraints. **IFAC-PapersOnLine**, 2017. v. 50, n. 1, p. 1101 – 1105, 2017.

KIM, Y.-D. A new branch and bound algorithm for minimizing mean tardiness in two-machine flowshops. **Computers & Operations Research**, 1993. v. 20, n. 4, p. 391 – 401, 1993.

KIM, Y.-D. Minimizing total tardiness in permutation flowshops. **European Journal of Operational Research**, 1995. v. 85, n. 3, p. 541 – 555, 1995.

KONONOV, A. et al. Quantity-based buffer-constrained two-machine flowshop problem: active and passive prefetch models for multimedia applications. **Journal of Scheduling**, 2012. v. 15, n. 4, p. 487–497, 2012.

KUBIAK, W. et al. Two-machine flow shops with limited machine availability. **European Journal of Operational Research**, 2002. v. 136, n. 3, p. 528 – 540, 2002.

KURIHARA, K. et al. Flow shop scheduling for separation model of set-up and net process based on Branch-and-Bound method. **Computers and Industrial Engineering**, 2009. v. 57, n. 2, p. 550–562, 2009.

LABIDI, M. et al. A computational study of the two-machine no-wait flow shop scheduling problem subject to unequal release dates and non-availability constraints. **IEEE Access**, 2018. v. 6, p. 16294–16304, 2018.

LADHARI, T.; HAOUARI, M. A computational study of the permutation flow shop problem based on a tight lower bound. **Computers & Operations Research**, 2005. v. 32, n. 7, p. 1831–1847, 2005.

LAGEWEG, B. J.; LENSTRA, J. K.; KAN, A. H. G. R. A general bounding scheme for the permutation flow-shop problem. **Operations Research**, 1978. v. 26, n. 1, p. 53–67, 1978.

LAND, A. H.; DOIG, A. G. An automatic method of solving discrete programming problems. **Econometrica**, 1960. [Wiley, Econometric Society], v. 28, n. 3, p. 497–520, 1960.

LEE, J.-Y.; KIM, Y.-D. Minimizing total tardiness in a two-machine flowshop scheduling problem with availability constraint on the first machine. **Computers & Industrial Engineering**, 2017. v. 114, p. 22 – 30, 2017.

LEE, W. C.; CHUNG, Y. H. Permutation flowshop scheduling to minimize the total tardiness with learning effects. **International Journal of Production Economics**, 2013. v. 141, n. 1, p. 327–334, 2013.

LEE, W.-C. et al. A two-machine flowshop scheduling problem with deteriorating jobs and blocking. **International Journal of Production Economics**, 2010. v. 124, n. 1, p. 188 – 197, 2010.

LEE, W.-C.; WU, C.-C. Minimizing the total flow time and the tardiness in a two-machine flow shop. **International Journal of Systems Science**, 2001. v. 32, n. 3, p. 365–373, 2001.

LEE, W.-C.; WU, C.-C. Minimizing total completion time in a two-machine flowshop with a learning effect. **International Journal of Production Economics**, 2004. v. 88, n. 1, p. 85 – 93, 2004.

LEE, W. C.; YEH, W. C.; CHUNG, Y. H. Total tardiness minimization in permutation flowshop with deterioration consideration. **Applied Mathematical Modelling**, 2014. v. 38, n. 13, p. 3081–3092, 2014.

LI, D.-C. et al. Two-machine flowshop scheduling with truncated learning to minimize the total completion time. **Computers & Industrial Engineering**, 2011. v. 61, n. 3, p. 655 – 662, 2011.

LIAO, L.-M.; TSAI, C.-H. Heuristic algorithms for two-machine flowshop with availability constraints. **Computers & Industrial Engineering**, 2009. v. 56, n. 1, p. 306 – 311, 2009.

LIN, B. M. T.; LIN, F. C.; LEE, R. C. T. Two-machine flow-shop scheduling to minimize total late work. **Engineering Optimization**, 2006. v. 38, n. 4, p. 501–509, 2006.

LIN, B. M. T.; WU, J. M. A simple lower bound for total completion time minimization in a two-machine flowshop. **Asia-Pacific Journal of Operational Research**, 2005. v. 22, n. 03, p. 391–407, 2005.

LIN, B. M. T.; WU, J. M. Bicriteria scheduling in a two-machine permutation flowshop. **International Journal of Production Research**, 2006. v. 44, n. 12, p. 2299–2312, 2006.

LIN, F.-C.; HONG, J.-S.; LIN, B. M. A two-machine flowshop problem with processing time-dependent buffer constraints. an application in multimedia presentations. **Computers & Operations Research**, 2009. v. 36, n. 4, p. 1158 – 1175, 2009.

LOMNICKI, Z. A. Branch-and-bound algorithm for the exact solution of the three-machine scheduling problem. **Operational Research Society**, 1965. v. 16, n. 1, p. 89–100, 1965.

MACCARTHY, B. L.; LIU, J. Addressing the gap in scheduling research: a review of optimization and heuristic methods in production scheduling. **International Journal of Production Research**, 1993. v. 31, n. 1, p. 59–79, 1993.

MADHUSHINI, N.; RAJENDRAN, C. Branch-and-bound algorithms for scheduling in an m-machine permutation flowshop with a single objective and with multiple objectives. **European J. of Industrial Engineering**, 2011. v. 5, n. 4, p. 361, 2011.

MADHUSHINI, N.; RAJENDRAN, C. A bottleneck-assignment based branch-and-bound algorithm to minimize the makespan in an m-machine permutation flowshop. **Technology Operation Management**, 2012. v. 3, n. 1, p. 1–10, Dec 2012.

MADHUSHINI, N.; RAJENDRAN, C.; DEEPA, Y. Branch-and-bound algorithms for scheduling in permutation flowshops to minimize the sum of weighted flowtime/sum of weighted tardiness/sum of weighted flowtime and weighted tardiness/sum of weighted flowtime, weighted tardiness and weighted earliness of jobs. **The Journal of the Operational Research Society**, 2009. Palgrave Macmillan Journals, v. 60, n. 7, p. 991–1004, 2009.

MAZDEH, M. M.; ROSTAMI, M. A branch-and-bound algorithm for two-machine flow-shop scheduling problems with batch delivery costs. **International Journal of Systems Science: Operations & Logistics**, 2014. v. 1, n. 2, p. 94–104, 2014.

MCMAHON, G. B.; BURTON, P. G. Flow-shop scheduling with the branch-and-bound method. **Operations Research**, 1967. v. 15, n. 3, p. 473–481, 1967.

MCMAHON, G. B.; LIM, C.-J. The two-machine flow shop problem with arbitrary precedence relations. **European Journal of Operational Research**, 1993. v. 64, n. 2, p. 249 – 257, 1993. Project Management anf Scheduling.

MOCCELLIN, J. a. V.; NAGANO, M. S. Uma propriedade estrutural do problema de programação da produção flow shop permutacional com tempos de setup. **Pesquisa Operacional**, 2007. scielo, v. 27, p. 487 – 515, 00 2007.

MOCCELLIN, J. V. A new heuristic method for the permutation flow shop scheduling problem. **The Journal of the Operational Research Society**, 1995. v. 46, n. 7, p. 883–886, 1995.

MOSLEHI, G.; KHORASANIAN, D. Optimizing blocking flow shop scheduling problem with total completion time criterion. **Computers & Operations Research**, 2013. v. 40, n. 7, p. 1874 – 1883, 2013.

MOSLEHI, G. et al. Two-machine flow shop scheduling to minimize the sum of maximum earliness and tardiness. **International Journal of Production Economics**, 2009. v. 122, n. 2, p. 763 – 773, 2009.

MOUKRIM, A.; REBAINE, D.; SERAIRI, M. A branch and bound algorithm for the two-machine flowshop problem with unit-time operations and time delays. **RAIRO Operations Research**, 2014. v. 48, n. 2, p. 235 – 254, 2014.

MSAKNI, M. K. et al. Minimizing the total completion time in a two-machine flowshop problem with time delays. **Engineering Optimization**, 2016. v. 48, n. 7, p. 1164–1181, 2016.

NABESHIMA, I. On the bound of makespan and its application in m machine-schedule problem. **Journal of the Operations Research Society of Japan**, 1967. v. 9, n. 3 & 4, p. 98 – 136, 1967.

NAGANO, M. S.; MOCCELLIN, J. V. A high quality solution constructive heuristic for flow shop sequencing. **The Journal of the Operational Research Society**, 2002. v. 53, n. 12, p. 1374–1379, 2002.

NAGAR, A.; HERAGU, S. S.; HADDOCK, J. A branch-and-bound approach for a two-machine flowshop scheduling problem. **The Journal of the Operational Research Society**, 1995. Palgrave Macmillan Journals, v. 46, n. 6, p. 721–734, 1995.

NAGAR, A.; HERAGU, S. S.; HADDOCK, J. A combined branch-and-bound and genetic algorithm based approach for a flowshop scheduling problem. **Annals of Operations Research**, 1996. v. 63, n. 3, p. 397–414, Jun 1996.

NARAIN, L.; BAGGA, P. C. Minimizing total elapsed time subject to zero total idle time of machines in n x 3 flowshop problem. **Indian Journal of Pure and Applied Mathematics**, 2003. v. 34, n. 2, p. 219–228, 2003.

NARAIN, L.; BAGGA, P. C. Flowshop no idle scheduling to minimise the mean flowtime. **The ANZIAM Journal**, 2005. Cambridge University Press, v. 47, n. 2, p. 265–275, 2005.

NG, C. et al. A branch-and-bound algorithm for solving a two-machine flow shop problem with deteriorating jobs. **Computers & Operations Research**, 2010. v. 37, n. 1, p. 83 – 90, 2010.

PAN, C.-H.; WU, C.-C. An asymptotic two-phase algorithm to minimize total flow time for a two-machine flowshop. **International Journal of Systems Science**, 1996. v. 27, n. 10, p. 925–930, 1996.

PAN, J. C.-H.; CHEN, J.-S.; CHAO, C.-M. Minimizing tardiness in a two-machine flow-shop. **Computers & Operations Research**, 2002. v. 29, n. 7, p. 869 – 885, 2002.

PAN, J. C.-H.; FAN, E.-T. Two-machine flowshop scheduling to minimize total tardiness. **International Journal of Systems Science**, 1997. v. 28, n. 4, p. 405–414, 1997.

PANWALKAR, S. S.; KHAN, A. W. An improved branch and bound procedure for n x m flow shop problems. **Naval Research Logistics Quarterly**, 1975. v. 22, n. 4, p. 787–790, 1975.

PINEDO, M. L. **Scheduling: Theory, Algorithms, and Systems**. 3rd. ed. [S.l.]: Springer Publishing Company, Incorporated, 2008. ISBN 0387789340, 9780387789347.

POTTS, C. N. An adaptive branching rule for the permutation flow-shop problem. **European Journal of Operational Research**, 1980. v. 5, n. 1, p. 19–25, 1980.

RAJENDRAN, C. Two-stage flowshop scheduling problem with bicriteria. **The Journal of the Operational Research Society**, 1992. Palgrave Macmillan Journals, v. 43, n. 9, p. 871–884, 1992.

RAKROUKI, M. A. et al. A branch-and-bound algorithm for the two-machine total completion time flowshop problem subject to release dates. **Operational Research**, 2017. 2017.

REN, T. et al. Permutation flow-shop scheduling problem to optimize a quadratic objective function. **Engineering Optimization**, 2017. v. 49, n. 9, p. 1589–1603, 2017.

RIOS-MERCADO, R. Z.; BARD, J. A branch-and-bound algorithm for permutation flow shops with sequence-dependent setup times. **IIE Transactions**, 1999. v. 31, n. 8, p. 721–731, 1999.

RITT, M. A branch-and-bound algorithm with cyclic best-first search for the permutation flow shop scheduling problem. In: **2016 IEEE International Conference on Automation Science and Engineering (CASE)**. [S.l.: s.n.], 2016. p. 872–877.

RONCONI, D. P. A branch-and-bound algorithm to minimize the makespan in a flowshop with blocking. **Annals of Operations Research**, 2005. v. 138, n. 1, p. 53–65, 2005.

RONCONI, D. P.; ARMENTANO, V. A. Lower bounding schemes for flowshops with blocking in-process. **Journal of the Operational Research Society**, 2001. v. 52, n. 11, p. 1289–1297, 2001.

RONCONI, D. P.; KAWAMURA, M. S. The single machine earliness and tardiness scheduling problem: lower bounds and a branch-and-bound algorithm. **Computational and Applied Mathematics**, 2010. scielo, v. 29, p. 107 – 124, 06 2010.

RUIZ, R.; MAROTO, C.; ALCARAZ, J. Solving the flowshop scheduling problem with sequence dependent setup times using advanced metaheuristics. **European Journal of Operational Research**, 2005. v. 165, n. 1, p. 34 – 54, 2005.

SAADANI, N. E. H.; GUINET, A.; MOALLAA, M. Three stage no-idle flow-shops. **Computers & Industrial Engineering**, 2003. v. 44, n. 3, p. 425–434, 2003.

SANCHES, F. B.; TAKANO, M. I.; NAGANO, M. S. Evaluation of heuristics for a branch and bound algorithm to minimize the makespan in a flowshop with blocking. **Acta Scientiarum**, 2016. v. 38, n. 3, p. 321–326, 2016.

SAYIN, S.; KARABATI, S. Theory and methodology a bicriteria approach to the two-machine flow shop scheduling problem. **European Journal of Operational Research**, 1999. v. 113, n. 2, p. 435 – 449, 1999.

SCHALLER, J. A new lower bound for the flow shop group scheduling problem. **Computers & Industrial Engineering**, 2001. v. 41, n. 2, p. 151–161, 2001.

SCHALLER, J. Note on minimizing total tardiness in a two-machine flowshop. **Computers & Operations Research**, 2005. v. 32, n. 12, p. 3273 – 3281, 2005.

SCHALLER, J. E.; GUPTA, J. N.; VAKHARIA, A. J. Scheduling a flowline manufacturing cell with sequence dependent family setup times. **European Journal of Operational Research**, 2000. v. 125, n. 2, p. 324 – 339, 2000.

SEN, T.; DILEEPAN, P.; GUPIA, J. N. The two-machine flowshop scheduling problem with total tardiness. **Computers & Operations Research**, 1989. v. 16, n. 4, p. 333 – 340, 1989.

SHIAU, Y.-R. et al. Two-agent two-machine flowshop scheduling with learning effects to minimize the total completion time. **Computers & Industrial Engineering**, 2015. v. 87, p. 580 – 589, 2015.

SIVRIKAYA-SERIFOGLU, F.; ULUSOY, G. A bicriteria two-machine permutation flowshop problem. **European Journal of Operational Research**, 1998. v. 107, n. 2, p. 414 – 430, 1998.

SMITH, W. E. Various optimizers for single-stage production. **Naval Research Logistics Quarterly**, 1956. Wiley Subscription Services, Inc., A Wiley Company, v. 3, n. 1-2, p. 59–66, 1956.

SOYLU, B.; KIRCA, .; AZIZOGLU, M. Flow shop-sequencing problem with synchronous transfers and makespan minimization. **International Journal of Production Research**, 2007. v. 45, n. 15, p. 3311–3331, 2007.

SU, L.-H.; LEE, Y.-Y. The two-machine flowshop no-wait scheduling problem with a single server to minimize the total completion time. **Computers & Operations Research**, 2008. v. 35, n. 9, p. 2952 – 2963, 2008.

SUHAMI, I.; MAH, R. S. An implicit enumeration scheme for the flowshop problem with no intermediate storage. **Computers & Chemical Engineering**, 1981. v. 5, n. 2, p. 83 – 91, 1981.

TADEI, R. et al. Minimising makespan in the two-machine flow-shop with release times. **The Journal of the Operational Research Society**, 1998. v. 49, n. 1, p. 77–85, 1998.

TAKANO, M. I.; NAGANO, M. S. A branch-and-bound method to minimize the makespan in a permutation flow shop with blocking and setup times. **Cogent Engineering**, 2017. Cogent, v. 4, n. 1, 2017.

TANG, L.; HUANG, L. Optimal and near-optimal algorithms to rolling batch scheduling for seamless steel tube production. **International Journal of Production Economics**, 2007. v. 105, n. 2, p. 357 – 371, 2007.

TELLACHE, N. E. H.; BOUDHAR, M. Two-machine flow shop problem with unit-time operations and conflict graph. **International Journal of Production Research**, 2017. v. 55, n. 6, p. 1664–1679, 2017.

TEMIZ, I.; EROL, S. Fuzzy branch-and-bound algorithm for flow shop scheduling. **Journal of Intelligent Manufacturing**, 2004. v. 15, n. 4, p. 449–454, Aug 2004.

T'KINDT, V.; CROCE, F. D.; ESSWEIN, C. Revisiting branch and bound search strategies for machine scheduling problems. **Journal of Scheduling**, 2004. v. 7, n. 6, p. 429–440, Nov 2004.

T'KINDT, V.; GUPTA, J. N.; BILLAUT, J.-C. Two-machine flowshop scheduling with a secondary criterion. **Computers & Operations Research**, 2003. v. 30, n. 4, p. 505 – 526, 2003.

TOKTAS, B.; AZIZOGLU, M.; KOKSALAN, S. K. Two-machine flow shop scheduling with two criteria: Maximum earliness and makespan. **European Journal of Operational Research**, 2004. v. 157, n. 2, p. 286 – 295, 2004.

TOUMI, S. et al. Branch and bound algorithm for solving blocking flowshop scheduling problem with total tardiness and total weighted tardiness criteria. **International Journal of Operational Research**, 2017. v. 30, n. 4, p. 441–459, 2017.

TOUMI, S. et al. Branch-and-bound algorithm for solving blocking flowshop scheduling problems with makespan criterion. **International Journal of Mathematics in Operational Research**, 2017. v. 10, n. 1, p. 34, 2017.

TSUJIMURA, Y. et al. An effective method for solving flow shop scheduling problems with fuzzy processing times. **Computers & Industrial Engineering**, 1993. v. 25, n. 1, p. 239 – 242, 1993.

VELDE, S. L. van de. Minimizing the sum of the job completion times in the two-machine flow shop by lagrangian relaxation. **Ann. Oper. Res.**, 1991. J. C. Baltzer AG, Science Publishers, Red Bank, NJ, USA, v. 26, n. 1-4, p. 257–268, jan. 1991.

WAGNEUR, E.; SRISKANDARAJAH, C. The two-machine permutation flow shop with state-dependent processing times. **Naval Research Logistics (NRL)**, 1993. v. 40, n. 5, p. 697–717, 1993.

WANG, J.-B. et al. Minimizing makespan in a two-machine flow shop with effects of deterioration and learning. **Optimization Letters**, 2012. v. 6, n. 7, p. 1393–1409, 2012.

WANG, J.-B.; LIU, L.-L. Two-machine flow shop problem with effects of deterioration and learning. **Computers & Industrial Engineering**, 2009. v. 57, n. 3, p. 1114 – 1121, 2009.

WANG, J.-B.; LIU, L.-L. Two-machine flow shop scheduling with linear decreasing job deterioration. **Computers & Industrial Engineering**, 2009. v. 56, n. 4, p. 1487–1493, 2009.

WANG, J.-B. et al. Minimizing total completion time in a two-machine flow shop with deteriorating jobs. **Applied Mathematics and Computation**, 2006. v. 180, n. 1, p. 185 – 193, 2006.

WANG, J.-B.; WANG, M.-Z. Minimizing makespan in three-machine flow shops with deteriorating jobs. **Computers & Operations Research**, 2013. v. 40, n. 2, p. 547 – 557, 2013.

WANG, J.-B.; WANG, M.-Z. Solution algorithms for the total weighted completion time minimization flow shop scheduling with decreasing linear deterioration. **The International Journal of Advanced Manufacturing Technology**, 2013. v. 67, n. 1, p. 243–253, 2013.

WANG, J. J.; ZHANG, B. H. Permutation flowshop problems with bi-criterion makespan and total completion time objective and position-weighted learning effects. **Computers and Operations Research**, 2015. v. 58, p. 24–31, 2015.

WANG, L. et al. On three-machine flow shop scheduling with deteriorating jobs. **International Journal of Production Economics**, 2010. v. 125, n. 1, p. 185 – 189, 2010.

WANG, X.; CHENG, T. Two-machine flowshop scheduling with job class setups to minimize total flowtime. **Computers & Operations Research**, 2005. v. 32, n. 11, p. 2751 – 2770, 2005.

WANG, Z.; WEI, C.-M.; LU, Y.-Y. Permutation flow shop problem with shortening job processing times. **Asia-Pacific Journal of Operational Research**, 2016. v. 33, n. 04, p. 1650032, 2016.

WOLSEY, L. A. **Integer programming**. New York, NY, USA: Wiley-Interscience, 1998.

WU, C.-C.; LEE, W.-C. Two-machine flowshop scheduling to minimize mean flow time under linear deterioration. **International Journal of Production Economics**, 2006. v. 103, n. 2, p. 572 – 584, 2006.

WU, C.-C.; LEE, W.-C.; WANG, W.-C. A two-machine flowshop maximum tardiness scheduling problem with a learning effect. **The International Journal of Advanced Manufacturing Technology**, 2007. v. 31, n. 7, p. 743–750, 2007.

WU, C.-C. et al. A two-machine flowshop scheduling problem with a truncated sum of processing-times-based learning function. **Applied Mathematical Modelling**, 2012. v. 36, n. 10, p. 5001 – 5014, 2012.

WU, W. H. et al. A heuristic-based genetic algorithm for the two-machine flowshop scheduling with learning consideration. **Journal of Manufacturing Systems**, 2015. v. 35, p. 223–233, 2015.

YANAI, S.; FUJIE, T. A three-machine permutation flow-shop problem with minimum makespan on the second machine. **The Journal of the Operational Research Society**, 2006. Palgrave Macmillan Journals, v. 57, n. 4, p. 460–468, 2006.

YANG, D.-L.; CHERN, M.-S. A two-machine flowshop sequencing problem with limited waiting time constraints. **Computers & Industrial Engineering**, 1995. v. 28, n. 1, p. 63 – 70, 1995.

YANG, S.-H.; WANG, J.-B. Minimizing total weighted completion time in a two-machine flow shop scheduling under simple linear deterioration. **Applied Mathematics and Computation**, 2011. v. 217, n. 9, p. 4819 – 4826, 2011.

YANG, W. H. Group scheduling in a two-stage flowshop. **The Journal of the Operational Research Society**, 2002. Palgrave Macmillan Journals, v. 53, n. 12, p. 1367–1373, 2002.

YEH, W.-C. A new branch-and-bound approach for the n/2/flowshop/n/2/flowshop/$\alpha f + \beta c_{max}$ flowshop scheduling problem. **Computers & Operations Research**, 1999. v. 26, n. 13, p. 1293 – 1310, 1999.

YEH, W.-C. An efficient branch-and-bound algorithm for the two-machine bicriteria flowshop scheduling problem. **Journal of Manufacturing Systems**, 2001. v. 20, n. 2, p. 113 – 123, 2001.

YEH, W.-C.; ALLAHVERDI, A. A branch-and-bound algorithm for the three-machine flowshop scheduling problem with bicriteria of makespan andtotal flowtime. **International Transactions in Operational Research**, 2004. Blackwell Publishing, v. 11, n. 3, p. 323–339, 2004.

YEUNG, W.; OGUZ, C.; CHENG, T. E. Two-stage flowshop earliness and tardiness machine scheduling involving a common due window. **International Journal of Production Economics**, 2004. v. 90, n. 3, p. 421 – 434, 2004. Production Control and Scheduling.