Pontifícia Universidade Católica
do Rio de Janeiro

Luana Mesquita Carrilho

Formulação Bucket-Indexed: uma nova abordagem para resolver o problema de Programação de Máquinas Paralelas

Dissertação de Mestrado

Dissertação apresentada ao Programa de Pós–Graduação em Engenharia de Produção do Departamento de Engenharia Industrial da PUC-Rio, como requisito parcial para obtenção do grau de Mestre em Engenharia de Produção.

Orientador : Prof. Silvio Hamacher
Co–orientador: Prof. Tiago Coutinho Carneiro de Andrade

Rio de Janeiro
Agosto de 2019

Pontifícia Universidade Católica
do Rio de Janeiro

Luana Mesquita Carrilho

Formulação Bucket-Indexed: uma nova abordagem para resolver o problema de Programação de Máquinas Paralelas

Dissertação apresentada ao Programa de Pós–Graduação em Engenharia de Produção do Departamento de Engenharia Industrial do Centro Técnico Científico da PUC-Rio como requisito parcial para obtenção do grau de Mestre em Engenharia de Produção.

Prof. Silvio Hamacher
Orientador
Departamento de Engenharia Industrial — PUC-Rio

Prof. Tiago Coutinho Carneiro de Andrade
Co–orientador
PSR - Power Systems Research

Prof. Virgílio José Martins Ferreira Filho
UFRJ

Prof. Rafael Martinelli Pinto
PUC-Rio

Rio de Janeiro, 09 de agosto de 2019

Luana Mesquita Carrilho

Bacharela em Engenharia de Produção pela PUC-Rio, graduada em julho de 2017. Desde 2015 é pesquisadora no Instituto Tecgraf/PUC-Rio na área de Apoio à Decisão e Supply Chain atuando nas frentes de pesquisa de modelagem matemática para o problema de programação de sondas marítimas. Ingressou no mestrado acadêmico no Departamento de Engenharia Industrial (DEI) da PUC-Rio na área de concentração de Transporte e Logística em agosto de 2017. Atualmente, foi aceita no processo de seleção para o Programa de Doutorado em Engenharia Industrial (DEI) da PUC-Rio.

Ficha Catalográfica

Carrilho, Luana Mesquita

Formulação Bucket-Indexed: uma nova abordagem para resolver o problema de Programação de Máquinas Paralelas / Luana Mesquita Carrilho; orientador: Silvio Hamacher; co–orientador: Tiago Coutinho Carneiro de Andrade. — 2019.

60 f. : il. (color.); 30 cm

Dissertação (mestrado) - Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro, Departamento de Engenharia Industrial, 2019.

Inclui bibliografia.

1. Engenharia Industrial – Teses. 2. Programação de Máquinas. 3. Máquinas Paralelas Idênticas. 4. Formulação *Bucket-Indexed*. 5. Formulação *Time-Indexed*. 6. Programação Linear Inteira Mista. I. Hamacher, Silvio. II. Andrade, Tiago Coutinho Carneiro de. III. Pontifícia Universidade Católica do Rio de Janeiro. Departamento de Engenharia Industrial. IV. Título.

CDD: 658.5

Agradecimentos

À Deus por iluminar meus caminhos e orientar minhas escolhas para alcançar meus objetivos.

À Luciana Mesquita e Lula Carrilho pelo amor, paciência e dedicação, me fazendo acreditar no meu potencial.

À minha família pela minha educação, apoio e carinho.

Aos orientadores Silvio Hamacher e Tiago Andrade, pela ótima orientação, ensinamentos e pelas oportunidades concedidas durante o desenvolvimento da dissertação.

O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Código de Financiamento 001.

Ao CNPq, ao TECGRAF e à PUC-Rio, pelos auxílios concedidos e pelo ótimo ambiente de estudo sem os quais este trabalho não teria sido possível.

À Gabriela Ribas por compartilhar seus conhecimentos e me orientar profissionalmente.

Aos meus colegas de trabalho Janaina Marchesi, Luiza Fiorencio, Pedro Lobato, Iuri Martins, Victor Cunha, João Gabriel, Danuza e os demais pela disponibilidade em ajudar sempre que necessário e pelo excelente ambiente de trabalh proporcionado.

Aos colegas e professores da PUC-Rio pela conhecimento passado e que permitiram meu crescimento acadêmico.

Aos funcionários do Deparatamento de Engenharia Industrial da PUC-Rio pelo suporte dado ao longo do mestrado.

A todos aqueles que, de alguma forma, contribuíram para este trabalho.

Resumo

Carrilho, Luana Mesquita; Hamacher, Silvio; Andrade, Tiago Coutinho Carneiro de. **Formulação Bucket-Indexed: uma nova abordagem para resolver o problema de Programação de Máquinas Paralelas**. Rio de Janeiro, 2019. 60p. Dissertação de Mestrado — Departamento de Engenharia Industrial, Pontifícia Universidade Católica do Rio de Janeiro.

A programação de máquinas é um processo de tomada de decisão que desempenha um importante papel na maioria das indústrias de manufatura e serviços. Esta dissertação aborda o problema de programação de máquinas paralelas idênticas sem preempção, considerando características da programação de data de liberação e data limite para execução do início das tarefas, restrição de precedência entre pares de tarefas, elegibilidade e disponibilidade de máquinas. Para resolver este problema, uma formulação de programação linear inteira mista é proposta. O novo modelo, chamado de *bucket-indexed* (BI), particiona o horizonte de planejamento em períodos de tempos de mesmo tamanho (*buckets*). O tamanho dos *buckets* é um parâmetro que varia de acordo com a instância e influencia o porte do modelo, podendo assumir valores entre 1 e o menor tempo de processamento das tarefas. Quanto maior o tamanho do *bucket*, menor é o número de *buckets* criados e, consequentemente, menor o porte do modelo. A formulação proposta é testada em instâncias reais referentes ao problema de programação de sondas para construção de poços de petróleo de uma indústria brasileira de óleo e gás. A fim de avaliar os resultados obtidos pela formulação BI, a formulação clássica *time-indexed* (TI) foi também implementada para comparação dos tempos computacionais e qualidade da solução. Os resultados da formulação proposta apontam um melhor desempenho nas instâncias testadas, reduzindo o tempo computacional em todos os casos e resolvendo instâncias de grande porte não resolvidas pela formulação TI.

Palavras–chave

Programação de Máquinas; Máquinas Paralelas Idênticas; Formulação *Bucket-Indexed*; Formulação *Time-Indexed*; Programação Linear Inteira Mista.

Abstract

Carrilho, Luana Mesquita; Hamacher, Silvio (Advisor); Andrade, Tiago Coutinho Carneiro de (Co-Advisor). **Bucket-Indexed Formulation: a new approach to solve Parallel Machine Scheduling Problem**. Rio de Janeiro, 2019. 60p. Dissertação de Mestrado — Departamento de Engenharia Industrial, Pontifícia Universidade Católica do Rio de Janeiro.

Machine scheduling is a decision-making process that plays an important role in most manufacturing and service industries. This dissertation tackles a nonpreemptive identical parallel machine scheduling problem, considering release dates, deadlines, precedences, eligibility, and machine availability constraints. To solve this problem, a mixed-integer linear programming formulation is proposed. The new model, called bucket-indexed, partitions the planning horizon in periods of equal length (buckets). The bucket size is a parameter which varies according to instances and influences the model size, assuming values between 1 and the shortest processing time of jobs. The larger the bucket size, the smaller is the number of buckets created and, consequently, the smaller the model size. The proposed formulation is tested in real instances of the rig scheduling problem for a Brazilian oil and gas industry. To evaluate the results obtained by the BI formulation, the classical time-indexed (TI) formulation was also implemented for comparison of computational times and solution quality. The results of the proposed formulation highlight a better performance in all the tested instances, reducing computational time in all cases and solving large instances unsolvable by the TI formulation.

Keywords

Machine Scheduling; Identical Parallel Machines; Bucket-Indexed Formulation; Time-Indexed Formulation; Mixed-Integer Linear Programming.

Sumário

1
Introdução

A programação de máquinas é um processo de tomada de decisão que desempenha um papel importante na maioria das indústrias de manufatura e serviços, além de ser usada em muitos setores, como agricultura, petróleo e gás, hospitais e transportes (PINEDO, 2016). Esse processo lida com a alocação de tarefas nos recursos disponíveis, normalmente escassos e caros, em determinados períodos de tempo com o objetivo de encontrar a programação que otimiza alguma medida de desempenho.

De acordo com Behera (2012) e Kaabi e Harrath (2014), nos últimos 50 anos o interesse pela pesquisa de programação de máquinas tem crescido, especialmente em máquinas paralelas, proporcionando grandes avanços no desenvolvimento de técnicas que contribuem para o conhecimento teórico e permitem melhores soluções para problemas práticos.

A programação de máquinas paralelas pertence à ampla classe de problemas de otimização combinatória, muitos dos quais são conhecidos como NP-difíceis. A dificuldade dos problemas indica que o esforço computacional requerido para encontrar uma solução ótima aumenta exponencialmente com o porte da instância (MOKOTOFF, 2001).

Com propósito de ampliar a pesquisa sobre o problema de programação de máquinas usando métodos exatos, esta dissertação tem como principal objetivo propor uma formulação *bucket-indexed* para a programação de máquinas paralelas idênticas visando melhorar os tempos computacionais de outros métodos de solução.

Além disso, outro objetivo almejado neste trabalho diz respeito a aplicação prática da nova formulação em instâncias reais de uma indústria de óleo e gás brasileira para resolver o problema de programação de sondas marítimas para construção de poços. Atualmente essas instâncias ou não são resolvidas, ou levam grandes tempos computacionais para obter soluções ótimas.

As contribuições desta dissertação estão relacionadas aos aspectos teóricos e práticos do problema tratado. No âmbito do conhecimento teórico este trabalho estende a literatura de métodos exatos para resolver o problema de programação de máquinas paralelas idênticas através da generalização da formulação *bucket-indexed* proposta inicialmente por Boland et al. (2016).

No que tange a aplicação prática do problema, a nova formulação proposta aborda características de problemas reais da indústria como data

de liberação e data limite para início das tarefas, precedência entre tarefas, elegibilidade e disponibilidade de máquinas, além de incorporar regras de negócios necessárias para tratar adequadamente as instâncias reais testadas.

A fim de avaliar o desempenho da formulação *bucket-indexed* proposta quanto aos tempos computacionais e à qualidade da solução, a formulação clássica *time-indexed*, a qual é comumente usada para resolver problemas de programação, foi usada como *benchmarck* para instâncias reais testadas.

Essa dissertação está divida em 6 capítulos, considerando este capítulo introdutório. O Capítulo 2 descreve o problema de máquinas paralelas de acordo com Pinedo (2016) e as características a serem consideradas nos modelos implementados. O Capítulo 3 apresenta como a formulação *time-indexed* está sendo abordada na literatura e a versão dessa formulação implementada para tratar o problema descrito no Capítulo 2. O Capítulo 4 apresenta a nova formulação *bucket-indexed* proposta para resolver o problema de máquinas paralelas idênticas. O Capítulo 5 apresenta uma revisão da literatura para o problema de programação de sondas, explica as características dos problemas e das instâncias reais testadas e analisa os experimentos computacionais comparando as duas formulações previamente apresentadas. Finalmente, o Capítulo 6 contém as considerações do estudo e propõe sugestões de pesquisas futuras.

2
Descrição do Problema

Neste capítulo será apresentado o problema de programação de máquinas paralelas a ser tratado nesta dissertação, descrevendo o ambiente das máquinas, as características da programação e o critério de avaliação. É importante destacar que todos os dados são conhecidos previamente e são considerados no processo de otimização, caracterizando um problema de programação *offline* (PINEDO, 2016). Ao final deste capítulo, os parâmetros pertinentes para o problema tratado serão sintetizados em uma tabela.

O intuito da programação é a elaboração de cronogramas de execução de tarefas em máquinas num horizonte de planejamento limitado, previamente estipulado, visando atender objetivos específicos (CHENG; SIN, 1990). Segundo Lustosa et al. (2008), o problema de programação de máquinas determina os instantes em que cada tarefa inicia e termina na máquina que a executará, baseando-se na designação e no sequenciamento propostos.

Seja o conjunto de tarefas J, as principais características de uma tarefa $j \in J$ abordadas nesta pesquisa de acordo com Pinedo (2016) são:

• Tempo de processamento ($p_j \in \mathbb{Z}^+$) representa a duração da tarefa j em dias;

• Data de liberação ($r_j \in \mathbb{Z}^+$) define o menor instante de tempo no qual a tarefa j pode iniciar sua execução;

• Data limite ($\bar{d}_j \in \mathbb{Z}^+$) define o maior instante de tempo no qual a tarefa j pode iniciar sua execução.

Baseado na classificação de Graham et al. (1979), o problema de programação pode ser definido em três aspectos: ambiente das máquinas, características da programação e o critério de avaliação. A programação em questão trata do problema de máquinas paralelas idênticas, o qual é a generalização do problema de máquinas únicas. Neste ambiente, um conjunto de máquinas idênticas M são disponibilizadas em paralelo e é preciso definir em qual máquina a tarefa será executada. É importante destacar que os tempos de processamento das tarefas são os mesmos em qualquer máquina que a execute, por isso são chamadas de idênticas.

As características da programação refletem restrições a serem cumpridas a fim de criar cronogramas viáveis. Neste contexto, serão detalhadas as características que descrevem o problema tratado de acordo com Pinedo (2016):

• Data de liberação (r_j) e data limite ($\bar{d}_j$): Todas as tarefas têm uma janela de início a ser respeitada, ou seja, cada tarefa j deve iniciar sua execução a partir da data de liberação até a data limite, não sendo permitido atrasos.

• Preempção indica que não é necessário manter uma tarefa na máquina, uma vez iniciada, até sua conclusão. Neste caso, o programador tem a possibilidade de interromper o processamento da tarefa a qualquer momento e colocar outra tarefa para execução. Quando a tarefa interrompida é retomada, apenas o tempo de processamento restante é executado. No problema tratado, as tarefas devem ser executadas sem-preempção, i.e., uma vez iniciada a tarefa não pode ser interrompida.

• Restrições de precedência indicam que uma ou mais tarefas devem terminar antes que outra tarefa, a elas relacionadas, possa iniciar. A restrição de precedência é contemplada entre as tarefas $j, \hat{j}$ indicada pelo parâmetro $prec_{j,\hat{j}} \in \{0, 1\}$ que vale 1 caso a tarefa j preceda a tarefa $\hat{j}$. Os inícios das tarefas $j, \hat{j}$ devem acontecer em instantes distintos, respeitando uma distância temporal dada pelo parâmetro $dif_{j,\hat{j}} \in \mathbb{Z}^+$ em dias. Caso a tarefa $\hat{j}$ possa iniciar imediatamente após o término da tarefa j, o parâmetro $dif_{j,\hat{j}}$ é igual ao tempo de processamento da tarefa j, isto é, $dif_{j,\hat{j}} = p_j$.

• Avaria nas máquinas implicam que uma máquina pode ou não estar continuamente disponível. Os períodos em que uma máquina não está disponível são assumidos como conhecidos e fixos, devido a manutenção programada. Seja o conjunto de disponibilidade A_m, cada disponibilidade $a \in A_m$ pertence a uma máquina m e seus inícios e fins podem ser representados por $[s_a, f_a]$, onde $s_a, f_a \in \mathbb{Z}^+$ e $s_a \leq f_a \ \forall a \in A_m$. Cada máquina m pode ter mais de uma disponibilidade e não podem ser sobrepostas, significando que para duas disponibilidades a e $(a+1)$ consecutivas, o fim da disponibilidade a deve terminar estritamente antes do início da disponibilidade $a+1$, isto é, $f_a < s_{a+1}$.

• Elegibilidade de máquinas representada pelo conjunto $M_j \subseteq M$ indica que nem todas as máquinas m são capazes de executar a tarefa j. Logo, para cada tarefa j existe pelo menos uma máquina $m \in M_j$ capaz de executá-la.

Segundo Pinedo (2016), o objetivo a ser minimizado é sempre uma função dos tempos de conclusão das tarefas e esses dependem de um cronograma. Por vezes o objetivo pode ser uma função das datas desejo, como por exemplo métricas de atraso.

O problema abordado trata as restrições de datas de maneira rígida, i.e., não permitindo atrasos. Portanto, para avaliar a programação, a função objetivo escolhida visa encontrar cronogramas que minimizem a soma dos tempos de conclusão das tarefas executadas em máquinas idênticas elegíveis e disponíveis.

O problema de programação de máquinas paralelas descrito não pode ser resolvido em um tempo eficiente, chamado de tempo polinomial, e portanto é classificado como um problema NP-difícil (PINEDO, 2016).

2.1 Considerações

Os conjuntos e parâmetros necessários para representar as características do problema de programação de máquinas paralelas descrito anteriormente estão resumidos na Tabela 2.1.

Tabela 2.1: Conjuntos e Parâmetros

Conjuntos	
J	Conjunto de tarefas
M	Conjunto de máquinas
$M_j \subseteq M$	Subconjunto de máquinas elegíveis para atender a tarefa j
A_m	Conjunto de disponibilidades
Parâmetros	
$p_j \in \mathbb{Z}^+$	Tempo de processamento da tarefa j
$r_j \in \mathbb{Z}^+$	Data de liberação da tarefa j
$d_j \in \mathbb{Z}^+$	Data limite da tarefa j
$prec_{j,\hat{j}} \in \{0,1\}$	1, quando tarefa j precede a tarefa $\hat{j}$; 0, caso contrário
$dif_{j,\hat{j}} \in \mathbb{Z}^+$	Diferença mínima entre os inícios das tarefas j e $\hat{j}$, caso $prec_{j,\hat{j}} = 1$
$s_a \in \mathbb{Z}^+$	Início da disponibilidade a
$f_a \in \mathbb{Z}^+$	Fim da disponibilidade a

É importante salientar que para um cronograma ser considerado viável ele deve cumprir as seguintes restrições:

- Cada tarefa só pode ser executada em uma máquina, uma única vez;
- Uma vez iniciada a tarefa, esta não pode ser interrompida (sem preempção);
- Os tempos de processamento das tarefas são conhecidos e independem da máquina e da programação, o que caracteriza um problema determinístico de máquinas idênticas;
- As tarefas devem respeitar as datas de liberação e as datas limite;
- A restrição de precedência entre tarefas deve ser respeitada em todos os casos;
- Uma máquina não pode executar mais de uma tarefa por vez;
- A restrição de elegibilidade das máquinas deve ser respeitada em todos os casos;
- As tarefas devem ser executadas sem preempção durante uma janela de disponibilidade.

A seguir, os Capítulos 3 e 4 apresentam as formulações desenvolvidas para resolver o problema descrito, a fim de compará-las em termos de tempo computacional e qualidade da solução.

3
Formulação *Time-Indexed*

Este capítulo mostra uma breve revisão da literatura sobre a formulação *time-indexed* aplicada ao problema de programação de máquinas e em seguida descreve o modelo de programação linear inteira desenvolvido para o problema de máquinas paralelas descrito no Capítulo 2.

3.1
Revisão da Literatura

A formulação clássica *time-indexed* (TI), apresentada por Dyer e Wolsey (1990), é um método exato capaz de encontrar a solução ótima para problemas combinatórios (NOCEDAL; WRIGHT, 2006). Contudo, essa formulação possui desvantagens em relação ao porte do modelo, o qual está relacionado ao tamanho da instância.

O modelo TI é motivado pela necessidade de programar um conjunto de tarefas (J) em máquinas (M) disponíveis durante um horizonte de planejamento (T). Assume-se que os dados são inteiros e que o horizonte de planejamento é discretizado em períodos. O tamanho do horizonte T é ao menos a soma dos tempos de processamento das tarefas e cresce de maneira pseudopolinomial com o tamanho da instância. Em alguns casos, a instância pode ser grande o suficiente para que o tempo necessário para resolver apenas a relaxação linear do problema seja intratável (VAN DEN AKKER et al., 1999; BAPTISTE; SADYKOV, 2009). Apesar disso, a maioria dos problemas complexos de planejamento da produção são modelados com essa formulação devido a simplicidade oriunda da modelagem de variáveis indexadas no tempo (BOLAND et al., 2016).

Diversos estudos passaram a ser realizados, a fim de melhorar os tempos de solução para problemas usando a formulação TI. Sousa e Wolsey (1992) e Van den Akker et al. (1999) foram os primeiros a mostrar que, apesar da formulação criar modelos de grande porte, sua relaxação linear é forte em relação a outras alternativas de programação linear inteira mista e fornece limites inferiores melhores, levando a geração de algoritmos de planos de corte mais robustos.

Van den Akker et al. (2000) foram os primeiros a utilizar geração de colunas para resolver o problema de programação de uma máquina, minimizando a soma dos tempos de conclusão sujeito a datas de liberação, a partir da de-

composição *Dantzig-Wolfe* aplicada a formulação TI. Motivados por problemas reais da indústria, Cordone et al. (2017) também utilizaram o método de geração de colunas para atingir resultados ótimos em um problema programação de máquina única com restrições de data limite.

No início dos anos 2000, importantes pesquisas foram feitas usando técnicas de combinação de programação inteira e programação de restrições (constraint programming), possibilitando a resolução de aplicações reais as quais não podiam ser resolvidas pelas duas abordagens separadamente (EDIS; OZKARAHAN, 2011). Seguindo o viés de aplicações de técnicas combinadas, Özpeynirci (2015) propôs o uso de uma heurística para calcular o menor tamanho de horizonte de planejamento que seja suficiente para realizar a programação das tarefas em máquinas paralelas gerando o menor porte de modelo possível.

Li (2017) e Jäger (2018) estudaram algoritmos aproximativos baseados na formulação TI para o problema clássico de programação de máquinas paralelas minimizando a soma dos tempos de conclusão ponderados. Li (2017) tratou restrições de precedência entre tarefas e Jäger (2018) incorporou as restrições de data de liberação e permitiu atrasos nas precedências, obtendo algoritmos mais genéricos que Li (2017). Esses novos algoritmos melhoram os resultados do estado da arte.

Em suma, a maioria dos trabalhos encontrados na literatura abordam a formulação *time-indexed* como base para desenvolver planos de corte, geração de colunas e algoritmos aproximativos, devido a relaxação linear forte desta formulação. Esses algoritmos obtiveram melhor desempenho em relação aos tempos computacionais quando comparado a métodos de solução simples, como a resolução da formulação TI.

Em relação às características dos problemas de programação de máquinas, pode ser observado que os artigos tratam apenas de restrições de precedência, data de liberação e data limite. Nessa dissertação, a formulação TI proposta irá considerar, simultaneamente, as restrições de precedência, data de liberação e data limite, além de incluir as restrições de elegibilidade e disponibilidade de máquinas. Nesta abordagem não serão consideradas técnicas para melhorar a eficiência dessa formulação.

3.2 Modelo proposto

O problema abordado está descrito no Capítulo 2, onde um conjunto de tarefas J deve ser programadas em máquinas elegíveis M_j em uma dada disponibilidade A_m durante um horizonte de planejamento T. O valor de T

pode ser obtido pela equação (3-1) onde T é igual ao valor máximo obtido pela soma da data limite com o tempo de processamento gerando um horizonte de planejamento suficiente para programação.

$$T = \max_{j \in J} (\bar{d}_j + p_j) \tag{3-1}$$

Na formulação *time-indexed* os possíveis períodos para início da tarefa são facilmente modelados, por se tratar de variáveis indexadas em t. No problema tratado é necessário que cada tarefa j respeite a data de liberação r_j e a data limite $\bar{d}_j$. Além disso, cada tarefa j só pode ser programada em uma disponibilidade a, a partir do início s_a até o último período em que a tarefa j possa iniciar cumprindo o fim da disponibilidade $f_a - p_j + 1$. Para garantir essas restrições da programação e a elegibilidade das máquinas, define-se um parâmetro $existeX_{m,j,t} \in \{0,1\}$ que vale 1 quando a tarefa j pode iniciar na máquina m no período t. O procedimento para cálculo desse parâmetro é apresentado no Algoritmo 1.

Algoritmo 1: CÁLCULO $existeX_{m,j,t}$

Entrada: $J, M_j, A_m, T, p_j, r_j, \bar{d}_j, s_a, f_a$
Saída: $existeX_{m,j,t}$

```
início
    para cada j ∈ J, m ∈ M_j, a ∈ A_m e t ∈ [1, T − p_j + 1] faça
        se t ≥ r_j e t ≤ d̄_j e t ≥ s_a e t ≤ f_a − p_j + 1 então
            existeX_{m,j,t} = 1
        fim
    fim
fim
retorna existeX_{m,j,t}
```

O modelo TI descrito a seguir é formulado com apenas a variável de decisão $x_{m,j,t} \in \{0,1\}$, a qual assume o valor 1 quando a tarefa j inicia na máquina m no período t.

$$\min \sum_{m \in M_j} \sum_{j \in J} \sum_{t \in [1,T]} x_{m,j,t}(t + p_j - 1) \tag{3-2}$$

$$\sum_{m \in M_j} \sum_{t \in [1,T]} x_{m,j,t} = 1 \quad \forall j \in J \tag{3-3}$$

$$\sum_{j \in J} \sum_{\hat{t} \in [t-p_j+1,t]} x_{m,j,\hat{t}} \leq 1 \quad \forall m \in M_j, \ \forall t \in [1,T] \tag{3-4}$$

$$\sum_{m \in M_{\hat{j}}} \sum_{t \in [1,T]} t.x_{m,\hat{j},t} - \sum_{m \in M_j} \sum_{t \in [1,T]} t.x_{m,j,t} \geq dif_{j,\hat{j}} \ \ \forall (j,\hat{j}) \in J \mid prec_{j,\hat{j}} = 1 \tag{3-5}$$

$$x_{m,j,t} \in \{0,1\} \ \ \forall m \in M_j, \ \forall j \in J, \ \forall t \in [1,T] \mid existeX_{m,j,t} = 1 \tag{3-6}$$

A função objetivo (3-2) visa minimizar a soma dos tempos de conclusão das tarefas. As restrições (3-3) indicam que cada tarefa j inicia exatamente uma vez e, portanto, é executada sem preempção em uma máquina. As restrições (3-4) garantem que no máximo uma tarefa é executada por vez na máquina m, impedindo que haja sobreposição de tarefas na mesma. As restrições (3-5) indicam que a precedência entre as tarefas $j, \hat{j}$ é respeitada, considerando pelo menos a distância mínima $dif_{j,\hat{j}}$ entre seus inícios. As restrições (3-6) impõem a integralidade das variáveis $x_{m,j,t}$ e asseguram que só serão geradas caso o parâmetro $existeX_{m,j,t} = 1$, isto é, respeitando as datas de liberação, as datas limites, a elegibilidade e a disponibilidade das máquinas.

A seguir o Capítulo 4 irá apresentar a formulação *bucket-indexed* proposta para abordar o problema de programação de máquinas paralelas.

4
Formulação *Bucket-Indexed*

Este capítulo tem como principal objetivo apresentar o modelo de programação linear inteira mista *bucket-indexed* (BI) proposto para resolver o problema de máquinas paralelas descrito no Capítulo 2, explicando os conceitos que fundamentam essa formulação e as contribuições desta dissertação. Antes disso, uma breve revisão da literatura irá mostrar como essa formulação já foi abordada para o problema de programação de máquinas.

4.1
Revisão da Literatura

A recente formulação *bucket-indexed* (BI) proposta por Boland et al. (2016) é um modelo de programação linear inteira mista baseado em uma discretização do horizonte de planejamento em períodos de tempo mais longos chamados de *buckets*. Essa ideia já foi aplicada a problemas de fluxo de rede dinâmico, do caixeiro viajante com janela de tempo e de planejamento de transporte (BOLAND et al., 2016).

Neste trabalho, Boland et al. (2016) apresentam pela primeira vez a formulação BI para o problema de programação de tarefas sem preempção em uma máquina, mostrando as versões do modelo com restrições de data de liberação, data desejo e data limite. Instâncias geradas foram comparadas com o modelo TI e na maioria dos casos a formulação proposta teve um desempenho superior em relação aos tempos de solução. Uma observação é feita em relação ao modelo proposto: quanto maior o tamanho dos *buckets* mais fraca é a relaxação linear e menores são os tempos para encontrá-la. Por outro lado, quanto menor o tamanho dos *buckets* a relaxação linear é mais forte, mas os tempos para encontrá-la são maiores.

Inspirados nessa nova formulação, Riedler et al. (2017) e Riedler (2018) propuseram a formulação BI para resolver o problema real de programação de pacientes em um centro de tratamento de câncer austríaco. Nessas abordagens, a formulação BI é resolvida iterativamente e o tamanho do *bucket* é refinado com heurísticas. Esses algoritmos, comparados com formulações tradicionais e metaheurísticas, mostraram excelente desempenho.

Nota-se que na literatura este modelo ainda é pouco abordado, porém apresentou bom desempenho se comparado com a formulação TI e metaheurísticas. Por isso, essa dissertação tem como objetivo propor a formulação BI

para o problema de programação de máquinas paralelas descrito no Capítulo 2, considerando restrições de precedência, data de liberação, data limite, elegibilidade e disponibilidade de máquinas.

4.2 Contribuições da dissertação

Essa dissertação estende a literatura de métodos exatos para resolver o problema de programação de máquinas, contribuindo com uma formulação *bucket-indexed* para máquinas paralelas.

A formulação proposta é uma extensão do trabalho de Boland et al. (2016) e aborda características de problemas reais da indústria como precedência entre tarefas, data de liberação, data limite, elegibilidade e disponibilidade de máquinas.

Pontualmente as contribuições deste trabalho incluem:

• Generalizar a formulação original de máquina única para máquinas paralelas elegíveis;

• Restrições de precedência entre pares de tarefas $j, \hat{j}$, garantindo uma distância mínima entre os inícios das tarefas;

• Restrições que indicam que uma tarefa só pode ser executada em uma disponibilidade, respeitando os inícios e fins;

• Restrições que limitam as variáveis contínuas em relação aos inícios e fins das disponibilidades;

• Novas condições de geração de variáveis para impedir a geração de variáveis fora da disponibilidade da máquina.

A seguir será apresentado o modelo proposto, explicando a ideias da formulação e cada restrição incorporada no modelo.

4.3 Modelo proposto

O modelo *bucket-indexed* proposto tem o objetivo de programar um conjunto de tarefas J em máquinas elegíveis M_j em uma dada disponibilidade A_m durante um horizonte de planejamento B, sendo B o número de *buckets* em que o horizonte foi discretizado. O *bucket* pode ser entendido como uma partição "grosseira" do horizonte de planejamento ou uma agregação de períodos de tempo. A Figura 4.1 mostra um exemplo de discretização do horizonte de planejamento em 15 períodos e como seria a discretização em *buckets* de tamanho $\Delta = 5$. O tamanho do *bucket* (Δ), por definição, deve ser menor ou igual ao menor tempo de processamento das tarefas de uma instância, ou seja, $1 \leq \Delta \leq \min_{j \in J} p_j$.

Figura 4.1: Horizonte de planejamento discretizado em períodos e *buckets*

De acordo com Boland et al. (2016), o fator de sucesso desta formulação é que para cada tarefa j há duas partes distintas em um *bucket* b, representadas pelo conjunto $k_j \in K = \{0, 1\}$, que vale 0 caso a tarefa j inicie na primeira parte e 1 na segunda parte. As duas partes são contíguas e a união delas representa um *bucket*. O índice k_j portanto é essencial para identificar o número de *buckets* que uma tarefa irá ocupar, além de informar as frações de *buckets* que estão sendo utilizadas para executá-la. Esses conceitos serão vistos mais adiante ao longo do capítulo.

Para ilustrar o índice k_j, a Figura 4.2 apresenta o *bucket* divido em duas partes por uma linha tracejada, onde a primeira parte indica $k_j = 0$ e a segunda parte $k_j = 1$.

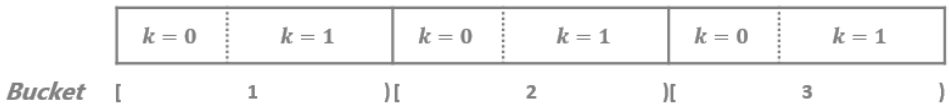

Figura 4.2: As duas partes de um *bucket*

Para cálculo do k_j duas informações são necessárias: o início e o tempo de processamento da tarefa. Supondo que a tarefa j inicie sua execução no período s_j, então ela iniciará sua execução no *bucket* $S_j \in \mathbb{Z}^+$, onde $S_j = \left\lfloor \frac{s_j - 1}{\Delta} \right\rfloor + 1$ e consome a fração $\sigma_j = S_j - \frac{s_j - 1}{\Delta}$ do *bucket* S_j, sendo $\sigma_j \in [0, 1]$ apresentado na Figura 4.3.

O tempo de processamento da tarefa j vale p_j períodos e $P_j = \left\lfloor \frac{p_j}{\Delta} \right\rfloor + 1$ *buckets*, isso significa que serão necessários, no mínimo, $P_j \in \mathbb{Z}^+$ *buckets* para executar a tarefa j. A fração necessária para execução da tarefa é p_j/Δ, logo existe uma fração $\pi_j \in [0, 1]$ que vale $\pi_j = P_j - \frac{p_j}{\Delta}$ e representa a fração

remanescente dos P_j *buckets* ocupados. A Figura 4.3 também mostra as frações calculadas p_j/Δ e π_j.

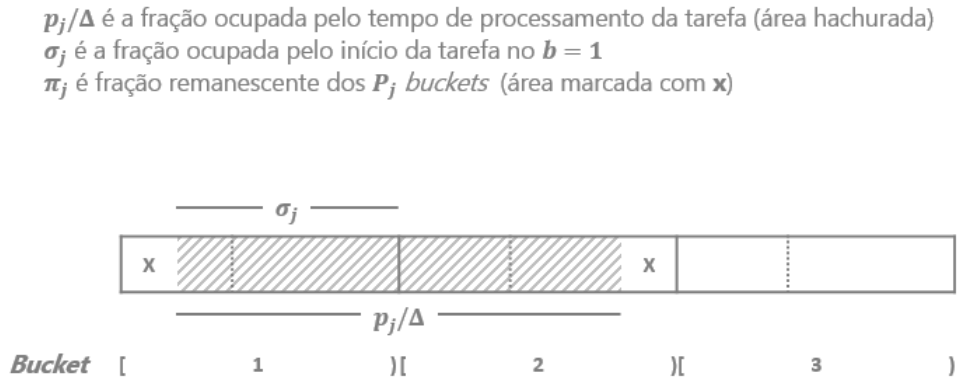

Figura 4.3: Exemplo representando as frações $p_j/\Delta, \sigma_j, \pi_j$

Com os valores de σ_j e π_j é possível calcular o valor de k_j através da expressão $1 - k_j < \sigma_j + \pi_j \leq 2 - k_j$, provada por Boland et al. (2016) a partir da proposição que indica que a tarefa j dura $P_j + k_j$ *buckets* para terminar sua execução. As Figuras 4.4 e 4.5 apresentam um exemplo numérico para elucidar as duas situações. Assume-se que a tarefa j tem o tempo de processamento $p_j = 8$ e o tamanho do *bucket* é $\Delta = 5$. Os parâmetros P_j e π_j valem, respectivamente, 2 e 0,4.

Na primeira situação apresentada na Figura 4.4, a tarefa j tem início no período $s_j = 2$, ou seja, no *bucket* $S_j = 1$, consumindo $\sigma_j = 0{,}8$. Dado que $\sigma_j + \pi_j = 0{,}8 + 0{,}4 = 1{,}2$ o valor de $k_j = 0$. Neste caso, a tarefa irá terminar sua execução no *bucket* $S_j + P_j + k_j - 1 = 1 + 2 + 0 - 1 = 2$.

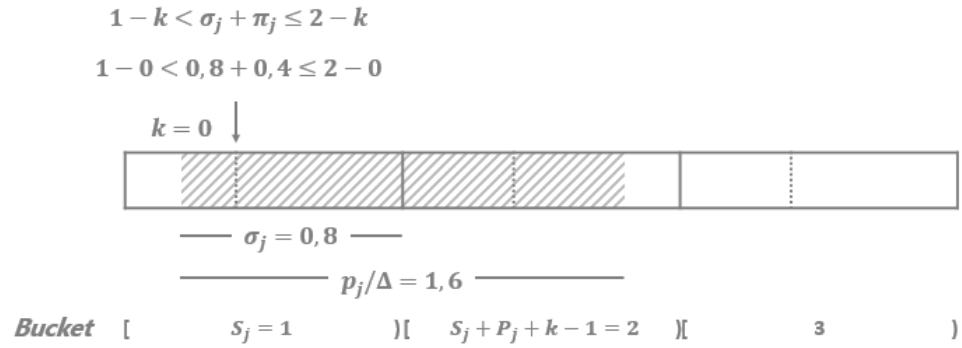

Figura 4.4: Início da tarefa j em $k_j = 0$

Na segunda situação, a tarefa j tem início no período $s_j = 5$, ou seja, no *bucket* $S_j = 1$, consumindo $\sigma_j = 0{,}2$. Dado que $\sigma_j + \pi_j = 0{,}2 + 0{,}4 = 0{,}6$ o valor de $k_j = 1$. Assim, na Figura 4.5, é possível observar que a tarefa irá terminar sua execução no *bucket* $S_j + P_j + k_j - 1 = 1 + 2 + 1 - 1 = 3$, concluindo que a duração em *buckets* da tarefa j é $P_j + k_j$ e depende da parte do *bucket* em que a tarefa iniciou.

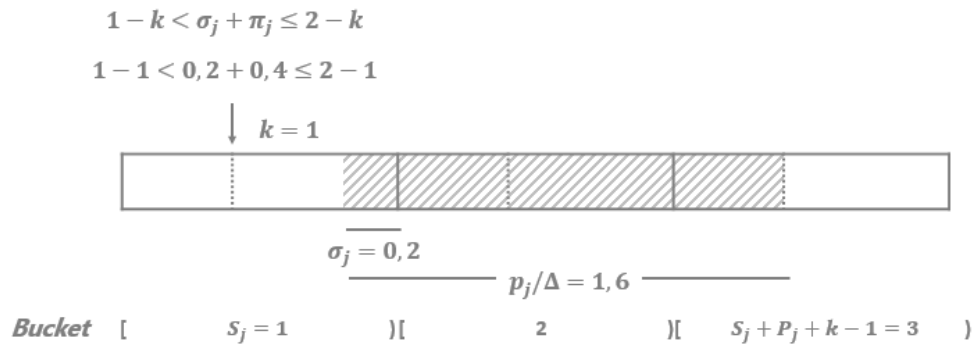

Figura 4.5: Início da tarefa j em $k_j = 1$

A partir dessa explicação inicial é possível introduzir o modelo *bucket-indexed*. As seções a seguir apresentam o modelo base para programação de máquinas paralelas elegíveis seguido das restrições de precedência, data de liberação, data limite e disponibilidade de máquinas que são adicionadas ao modelo base para tratar o problema descrito no Capítulo 2. O Apêndice A apresenta o modelo completo proposto por essa dissertação.

4.3.1 Modelo Base

O modelo base para programação de máquinas paralelas elegíveis é definido a partir de duas variáveis de decisão: a variável binária $z_{m,j,b,k} \in \{0, 1\}$ e a variável contínua $u_{m,j,b,k} \in [0, 1]$. Como as variáveis são definidas para cada tarefa j, o índice k_j, por simplificação, pode ser escrito simplesmente como k. A variável $z_{m,j,b,k}$ indica que a tarefa j é executada pela máquina m e inicia sua execução na parte k do *bucket* b, levando $b + P_j + k - 1$ *buckets* para ser concluída. A variável contínua $u_{m,j,b,k} \in [0, 1]$ representa a fração do *bucket* b ocupada pelo início da tarefa j na máquina m quando a variável $z_{m,j,b,k} = 1$, ou seja, o valor de σ_j apresentado anteriormente. Para facilitar a explicação da formulação BI, a variável auxiliar contínua $v_{m,j,b+P_j+k-1,k} \in [0, 1]$

será adicionada ao modelo representando a fração *bucket* final $b + P_j + k - 1$ ocupada pela tarefa j na máquina m.

Para ilustrar a interpretação das variáveis, a Figura 4.6 apresenta o exemplo anterior, no qual a variável $z_{m,j,1,1}$ assume valor 1 no *bucket* $b = 1$ na parte $k = 1$ e a fração ocupada pela tarefa j é $u_{m,j,1,1} = 0{,}2$. A variável $v_{m,j,3,1}$, por sua vez, assume valor 0,4 no *bucket* $b = 3$. Pela Figura 4.6 nota-se que a tarefa j, além de ocupar frações dos *buckets* 1 e 3, também ocupa por inteiro o *bucket* 2. Somando as frações de u, v e 1 (correspondente ao *bucket* 2) obtem-se o valor 1,6 que representa, como esperado, a fração de *buckets* do seu tempo de processamento, $P_j - \pi_j = 2 - 0{,}4 = 1{,}6$.

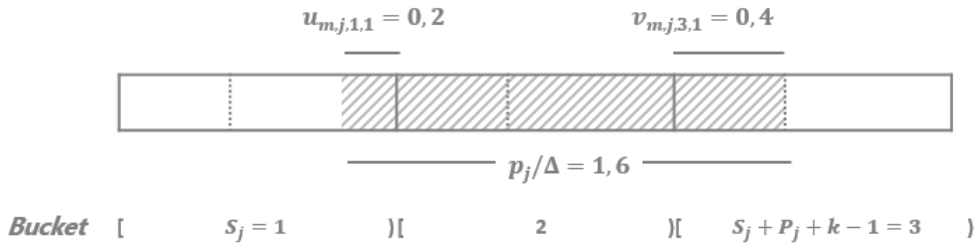

Figura 4.6: Variáveis $u_{m,j,b,k}$ e $v_{m,j,b+P_j+k-1,k}$

As equações (4-1) são responsáveis por fazer a relação entre as variáveis do modelo e permitir a omissão da variável $v_{m,j,b,k}$. A soma das variáveis $u_{m,j,b,k}$ e $v_{m,j,b+P_j+k-1,k}$ representam, respectivamente, as frações do início da tarefa no *bucket* inicial e do fim da tarefa no *bucket* final. Essa soma depende da parte do *bucket* em que a tarefa é iniciada. Se a tarefa j iniciar em $k = 0$ ela levará P_j *buckets* para ser executada e a soma das variáveis é igual a $2 - \pi_j$, se a tarefa j inciar em $k = 1$ ela levará $P_j + 1$ *buckets* para ser executada e a soma das variáveis é igual a $1 - \pi_j$.

$$u_{m,j,b,k} + v_{m,j,b+P_j+k-1,k} = (2 - k - \pi_j) z_{m,j,b,k} \quad \forall m \in M_j,\ \forall j \in J,\ \forall b \in [1, B],\ \forall k \in K \tag{4-1}$$

Assim como na formulação *time-indexed*, a geração de variáveis também deve ser restringida nesse modelo. As variáveis $z_{m,j,b,k}$ e $u_{m,j,b,k}$ são geradas apenas para os *buckets* em que a tarefa j possa iniciar, enquanto as variáveis $v_{m,j,b,k}$ apenas para os *buckets* em que a tarefa possa finalizar. Além disso, não são necessárias as variáveis $z_{m,j,b,1}$,$u_{m,j,b,1}$ e $v_{m,j,b+P_j,1}$ quando $\pi_j = 1$, pois a

partir da relação $1 - k < u_{m,j,b,k} + \pi_j \leq 2 - k$, se $\pi_j = 1$, $u_{m,j,b,k} + \pi_j > 1$ e $k = 0$ sempre.

Para garantir essas restrições da programação e a elegibilidade das máquinas, definem-se os parâmetros $existeZeU_{m,j,b,k} \in \{0,1\}$, que vale 1 quando a tarefa j pode iniciar na máquina m no *bucket* b, e $existeV_{m,j,b,k} \in \{0,1\}$, que vale 1 quando a tarefa j pode finalizar na máquina m no *bucket* b. O procedimento para cálculo desses parâmetros é apresentado nos Algoritmos 2 e 3.

Algoritmo 2: CÁLCULO $existeZeU_{m,j,b,k}$

Entrada: J, M_j, B, K, P_j, π_j
Saída: $existeZeU_{m,j,b,k}$

```
1 início
2   para cada j ∈ J, m ∈ M_j, k ∈ K = {0,1} e
      b ∈ [1, B − P_j − k + 1] faça
3     se π_j < 1 ou k = 0 então
4       existeZeU_{m,j,b,k} = 1
5     fim
6   fim
7 fim
8 retorna existeZeU_{m,j,b,k}
```

Algoritmo 3: CÁLCULO $existeV_{m,j,b,k}$

Entrada: J, M_j, B, K, P_j, π_j
Saída: $existeV_{m,j,b,k}$

```
1 início
2   para cada j ∈ J, m ∈ M_j, k ∈ K = {0,1} e b ∈ [P_j + k, B] faça
3     se π_j < 1 ou k = 0 então
4       existeV_{m,j,b,k} = 1
5     fim
6   fim
7 fim
8 retorna existeV_{m,j,b,k}
```

A variável contínua $u_{m,j,b,k}$ também deve ser limitada através de restrições de limite inferior e superior para que recebam valores de frações condizentes com o valor de k assumido. As restrições (4-2) e (4-3) estabelecem, respec-

tivamente, esses limites inferiores e superiores das variáveis considerando os domínios válidos quando $k = 0$ e $k = 1$.

$$\left((1-k)(1-\pi_j) + \frac{1}{\Delta}\right) z_{m,j,b,k} \leq u_{m,j,b,k} \;\; \forall m \in M_j, \; \forall j \in J, \; \forall b \in [1,B], \; \forall k \in K \tag{4-2}$$

$$u_{m,j,b,k} \leq (1 - k\pi_j) z_{m,j,b,k} \forall m \in M_j, \;\; \forall j \in J, \;\; \forall b \in [1,B], \;\; \forall k \in K \tag{4-3}$$

Quando $k = 0$ a tarefa inicia na primeira parte do *bucket*, logo a fração de $u_{m,j,b,k}$ é no máximo 1 e no mínimo $(1 - \pi_j) + 1/\Delta$. Se $u_{m,j,b,0} = 1$ a tarefa j iniciou na máquina m no primeiro instante do *bucket* b. Se $u_{m,j,b,0} = (1 - \pi_j) + 1/\Delta$ a tarefa j iniciou na máquina m no último instante do *bucket* b em que $k = 0$. A Figura 4.7 ilustra as duas situações possíveis para $k = 0$.

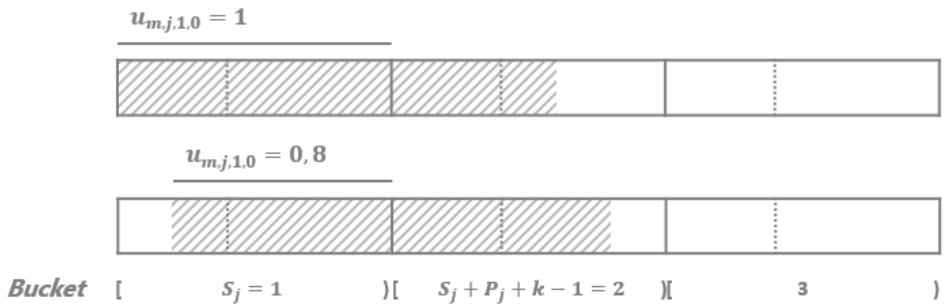

Figura 4.7: Limite da variável $u_{m,j,b,0}$ em $k = 0$

Quando $k = 1$ a tarefa inicia na segunda parte do *bucekt*, logo a fração de $u_{m,j,b,k}$ é no máximo $1 - \pi_j$ e no mínimo $1/\Delta$. Se $u_{m,j,b,1} = 1 - \pi_j$ a tarefa j iniciou na máquina m no primeiro instante do *bucket* b em que $k = 1$. Se $u_{m,j,b,1} = 1/\Delta$ a tarefa j iniciou na máquina m no último instante do *bucket* b. A Figura ilustra as três situações possíveis para $k = 1$.

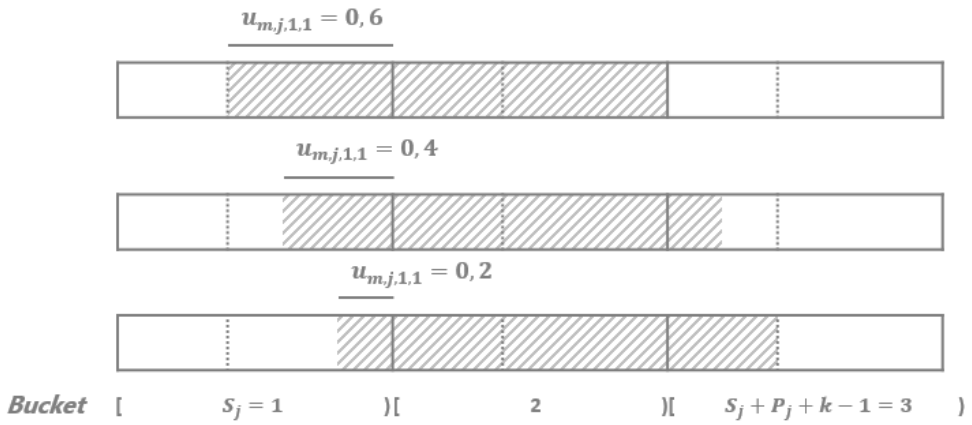

Figura 4.8: Limite da variável $u_{m,j,b,1}$ em $k = 1$

Outra forma de visualizar os limites da variável $u_{m,j,b,k}$ é através da relação já apresentada, $1 - k < u_{m,j,b,k} + \pi_j \leq 2 - k$, substituindo σ_j por $u_{m,j,b,k}$. A Tabela 4.1 apresenta para cada valor de $u_{m,j,b,k}$ possível o valor assumido por k.

Tabela 4.1: Relação $u_{m,j,b,k}, \pi_j ek$

$u_{m,j,b,k}$	π_j	$u_{m,j,b,k} + \pi_j$	k
1,0	0,4	1,4	0
0,8	0,4	1,2	0
0,6	0,4	1,0	1
0,4	0,4	0,8	1
0,2	0,4	0,6	1

Boland et al. (2016) assumem três hipóteses que também serão incorporadas no modelo dessa dissertação:

1. No máximo uma tarefa pode iniciar em um *bucket*.

   Essa hipótese é garantida pela restrição (4-4) em que a soma das variáveis $z_{m,j,b,k}$, as quais indicam que a tarefa j inicia no *bucket* b, seja no máximo 1 para todo *bucket* b. A soma das variáveis $z_{m,j,b,k}$ é zero quando nenhuma tarefa inicia no *bucket* b. A interpretação desta restrição é análoga a restrição (3-4) da formulação TI.

$$\sum_{j \in J} \sum_{k \in K} \sum_{\hat{b} \in [b - P_j + 2, b]} z_{m,j,\hat{b},k} \leq 1 \quad \forall m \in M_j, \ \forall b \in [1, B] \tag{4-4}$$

No exemplo da Figura 4.9, duas tarefas $J = \{1, 2\}$ com tempos de processamento $p_1 = p_2 = 8$ são programadas na máquina m e começam no início do *bucket* $b = 1$. Neste caso, as variáveis $z_{m,1,1,0}$ e $z_{m,2,1,1}$ assumem valor 1 e a restrição (4-4) para máquina m e *bucket* $b = 1$ é violada, pois a soma das variáveis z em questão é igual 2, impedindo que duas tarefas iniciem no mesmo *bucket*.

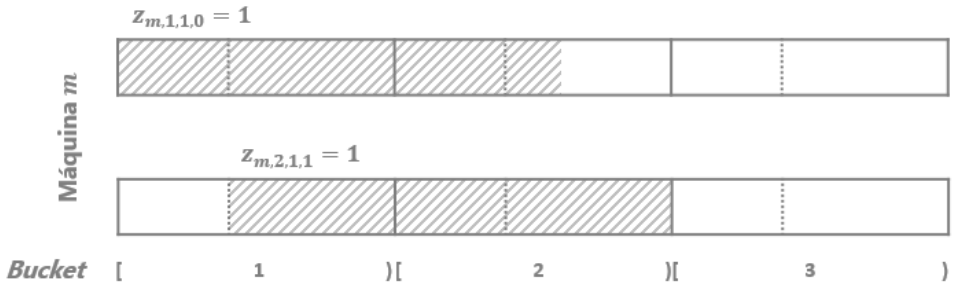

Figura 4.9: Exemplo de sobreposição de tarefas que iniciam no mesmo *bucket*

2. No máximo duas tarefas podem ser executadas em um *bucket*, mas não no mesmo instante de tempo.

   Essa hipótese garante que não haja sobreposição entre tarefas em um mesmo *bucket* b e está representada pela restrição (4-5).

$$\sum_{j \in J} \sum_{k \in K} \left(u_{m,j,b,k} + v_{m,j,b,k} + \sum_{\hat{b} \in [b - P_j + 2, b-1]} z_{m,j,\hat{b},k} \right) \leq 1 \;\; \forall m \in M_j, \; \forall b \in [1, B] \tag{4-5}$$

Na Figura 4.10, as mesmas duas tarefas $J = \{1, 2\}$ são programadas na máquina m e começam, respectivamente, no início do *bucket* $b = 1$ e no início do *bucket* $b = 2$. As variáveis $z_{m,1,1,0}$ e $z_{m,2,2,0}$ assumem valor 1, porém, nesta situação as restrições (4-4) para máquina m nos *buckets* $b = 1$ e $b = 2$ não acusam sobreposição de tarefas e, portanto uma solução viável permitiria tarefas sobrepostas no *bucket* $b = 2$. Com a restrição (4-5) para máquina m no $b = 2$ é possível indicar que o final da tarefa 1 está sobreposta com o início da tarefa 2, pois a soma das

frações $v_{m,1,2,0} = 0{,}6$ e $u_{m,2,2,0} = 1$ é maior que 1, violando a restrição e mostrando que a solução é na verdade inviável.

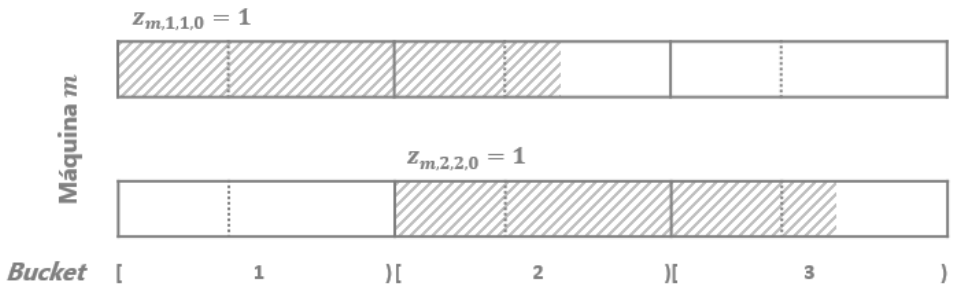

Figura 4.10: Exemplo de sobreposição de tarefas que iniciam em *buckets* diferentes

3. Todas as tarefas levam no mínimo dois *buckets* para serem executadas.

   O tamanho do *bucket* varia entre os valores $1 \leq \Delta \leq \min_{j \in J} p_j$ e, consequentemente a fração p_j/Δ varia no intervalo $1 \leq p_j/\Delta \leq p_j$ assumindo os valores $p_j/\Delta = 1$ quando $\Delta = p_j$ e $p_j/\Delta = p_j$ quando $\Delta = 1$.

   Para o cálculo de $P_j = \left\lfloor \frac{p_j}{\Delta} \right\rfloor + 1$:

   - Se $\Delta = p_j$ então $P_j = \left\lfloor \frac{p_j}{p_j} \right\rfloor + 1 = 1 + 1 = 2$
   - Se $\Delta = 1$ então $P_j = \left\lfloor \frac{p_j}{1} \right\rfloor + 1 = p_j + 1$

   Logo, mesmo que $p_j = 1$, o valor de $P_j \geq 2$. Como o número de *buckets* usados para execução da tarefa j é $P_j + k$ sendo $k \in \{0, 1\}$, o menor valor obtido por $P_j + k \geq 2$.

A função objetivo e as restrições a seguir definem o modelo base da formulação BI, considerando as hipóteses apresentadas, a fim de garantir a viabilidade da programação.

$$\min \sum_{m \in M_j} \sum_{j \in J} \sum_{b \in [1,B]} \sum_{k \in K} ((b + P_j + k - 2) z_{m,j,b,k} + v_{m,j,b+P_j+k-1,k}) \Delta \quad (4\text{-}6)$$

$$u_{m,j,b,k}+v_{m,j,b+P_j+k-1,k} = (2-k-\pi_j)z_{m,j,b,k} \ \forall m \in M_j, \ \forall j \in J, \ \forall b \in [1,B], \ \forall k \in K \tag{4-1}$$

$$\left((1-k)(1-\pi_j)+\frac{1}{\Delta}\right) z_{m,j,b,k} \leq u_{m,j,b,k} \ \forall m \in M_j, \ \forall j \in J, \ \forall b \in [1,B], \ \forall k \in K \tag{4-2}$$

$$u_{m,j,b,k} \leq (1-k\pi_j)z_{m,j,b,k} \forall m \in M_j, \ \forall j \in J, \ \forall b \in [1,B], \ \forall k \in K \tag{4-3}$$

$$\sum_{j \in J} \sum_{k \in K} \sum_{\hat{b} \in [b-P_j+2,b]} z_{m,j,\hat{b},k} \leq 1 \ \ \forall m \in M_j, \ \forall b \in [1,B] \tag{4-4}$$

$$\sum_{j \in J} \sum_{k \in K} \left(u_{m,j,b,k} + v_{m,j,b,k} + \sum_{\hat{b} \in [b-P_j+2,b-1]} z_{m,j,\hat{b},k} \right) \leq 1 \ \ \forall m \in M_j, \ \forall b \in [1,B] \tag{4-5}$$

$$\sum_{m \in M_j} \sum_{b \in [1,B]} \sum_{k \in K} z_{m,j,b,k} = 1 \ \ \forall j \in J \tag{4-7}$$

$$z_{m,j,b,k} \in \{0,1\} \ \ \forall m \in M_j, \ \forall j \in J, \ \forall b \in [1,B], \ \forall k \in K \mid existeZeU_{m,j,b,k} = 1 \tag{4-8}$$

$$u_{m,j,b,k} \in [0,1] \ \ \forall m \in M_j, \ \forall j \in J, \ \forall b \in [1,B], \ \forall k \in K \mid existeZeU_{m,j,b,k} = 1 \tag{4-9}$$

$$v_{m,j,b,k} \in [0,1] \ \ \forall m \in M_j, \ \forall j \in J, \ \forall b \in [1,B], \ \forall k \in K \mid existeV_{m,j,b,k} = 1$$

(4-10)

A função objetivo (4-6) visa minimizar a soma dos tempos de conclusão das tarefas. As equações (4-1) são responsáveis por fazer a relação entre as variáveis do modelo e permitir a omissão da variável $v_{m,j,b,k}$. As restrições (4-2) e (4-3) estabelecem, respectivamente, esses limites inferiores e superiores das variáveis considerando os domínios válidos quando $k = 0$ e $k = 1$. As restrições (4-4) indicam que no máximo 1 tarefa pode iniciar no *bucket* b e na máquina m sem que haja sobreposição. As restrições (4-5) garantem que as tarefas executadas na máquina m no *bucket* b não sejam sobrepostas. As restrições (4-7) indicam que cada tarefa j inicia exatamente uma vez e, portanto, é executada sem preempção em uma máquina. As restrições (4-8) - (4-10) impõem o domínio das variáveis binárias e contínuas.

4.3.2 Precedência entre tarefas

Para tratar a precedência descrita no Capítulo 2, uma das contribuições dessa dissertação, é adicionar ao modelo base restrições que garantem a relação de precedência entre tarefas $j, \hat{j}$, mantendo a distância mínima $dif_{j,\hat{j}}$ entre seus inícios.

Para isso, é necessário calcular os instantes de tempo em que as tarefas j e $\hat{j}$ iniciaram sua execução nos *buckets*. Os inícios são calculados de acordo com a equação (4-11).

$$inicio_j = \sum_{minM_j} \sum_{b\in[1,B]} \sum_{k\in K} ((b-1)z_{m,j,b,k} - u_{m,j,b,k})\Delta + \Delta + 1 \tag{4-11}$$

A partir dos inícios das tarefas calculados, as restrições (4-12) são simplesmente descritas pela subtração dos inícios de j e $\hat{j}$ garantindo a distância mínima de $dif_{j,\hat{j}}$.

$$inicio_j - inicio_{\hat{j}} \geq dif_{j,\hat{j}} \forall (j,\hat{j}) \in J \mid prec_{j,\hat{j}} = 1 \tag{4-12}$$

Vale observar que as variáveis $inicio_j, inicio_{\hat{j}}$ só foram usadas para simplificar a leitura dessa restrição, as restrições (4-13) garantem a integralidade das variáveis.

$$inicio_j \in \mathbb{Z}^+ \ \forall j \in J \tag{4-13}$$

4.3.3 Data de Liberação e Data Limite

Nessa formulação também são consideradas as restrições para início da execução das tarefas, impostas pelas datas de liberação (r_j) e datas limite ($\bar{d}_j$), limitam os valores assumidos pela variável contínua $u_{m,j,b,k}$, respectivamente, nos *buckets* $R_j \in \mathbb{Z}^+$ e $\bar{D}_j \in \mathbb{Z}^+$.

A restrição de data de liberação acontece no *bucket* $R_j = \left\lfloor \frac{r_j - 1}{\Delta} \right\rfloor + 1$ onde a fração $\rho_j \in [0,1]$ tem valor $\rho_j = R_j - \frac{r_j - 1}{\Delta}$ e representa a fração máxima do *bucket* R_j que pode ser usada para executar o início da tarefa j, ou seja, um limite superior para variável $u_{m,j,b,k}$. As restrições (4-14) e (4-15) são criadas a fim de garantir o cumprimento da data de liberação.

$$u_{m,j,R_j,0} \leq \rho_j z_{m,j,R_j,0} \quad \forall m,\ \forall j \mid 1 - \pi_j < \rho_j \leq 1 \tag{4-14}$$

$$u_{m,j,R_j,1} \leq \rho_j z_{m,j,R_j,1} \quad \forall m,\ \forall j \mid \rho_j \leq 1 - \pi_j \tag{4-15}$$

As restrições (4-14) indicam o limite superior de $u_{m,j,R_j,0}$ quando $k = 0$, e por isso só são geradas para $\rho_j \mid 1 - \pi_j < \rho_j \leq 1$. As restrições (4-15) indicam o limite superior de $u_{m,j,R_j,1}$ quando $k = 1$ e são geradas para o complemento $\rho_j \mid \rho_j \leq 1 - \pi_j$.

Continuando com o exemplo em que a tarefa j tem tempo de processamento $p_j = 8$ e o tamanho do *bucket* é $\Delta = 5$. Suponha que a tarefa j tem data de liberação igual a $r_j = 2$, logo $R_j = 1$ e $\rho_j = 0{,}8$. Nesse exemplo, no *bucket* 1 a tarefa j pode ocupar no máximo a fração de 0,8 e portanto $u_{m,j,b,k} \leq 0{,}8$. Qualquer valor assumido por $u_{m,j,b,k} > 0{,}8$, viola a data de liberação $r_j = 2$. A Figura 4.11 apresenta o exemplo.

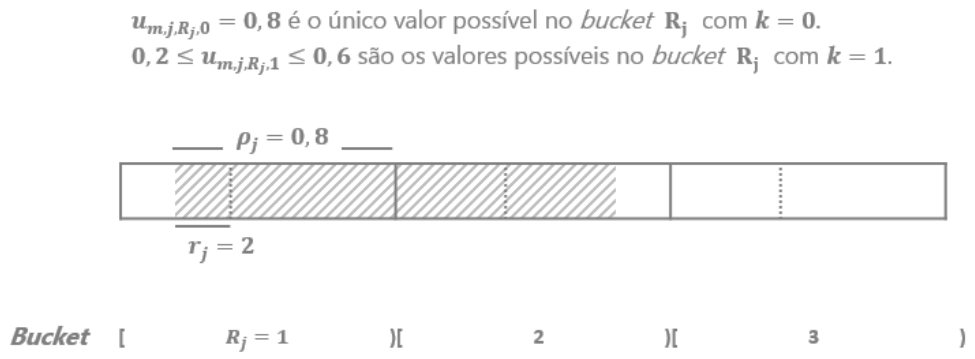

Figura 4.11: Exemplo de data de liberação da tarefa j

Analogamente, a restrição de data de limite acontece no *bucket* $\bar{D}_j = \left\lfloor \frac{\bar{d}_j - 1}{\Delta} \right\rfloor + 1$ onde a fração $\bar{\delta}_j \in [0,1]$, $\bar{\delta}_j = \bar{D}_j - \frac{\bar{d}_j - 1}{\Delta}$ representa a fração mínima do *bucket* $\bar{D}_j$ que pode ser usada para executar o início da tarefa j, ou seja, um limite inferior para variável $u_{m,j,b,k}$. As restrições (4-16) e (4-17) são criadas a fim de garantir o cumprimento da data limite.

$$\bar{\delta}_j z_{m,j,\bar{D}_j,0} \leq u_{m,j,\bar{D}_j,0} \quad \forall m,\ \forall j \mid 1 - \pi_j < \bar{\delta}_j \leq 1 \tag{4-16}$$

$$\bar{\delta}_j z_{m,j,\bar{D}_j,1} \leq u_{m,j,\bar{D}_j,1} \quad \forall m,\ \forall j \mid \bar{\delta}_j \leq 1 - \pi_j \tag{4-17}$$

As restrições (4-14) indicam o limite inferior de $u_{m,j,\bar{D}_j,0}$ quando $k = 0$, e por isso só são geradas para $\bar{\delta}_j \mid 1 - \pi_j < \bar{\delta}_j \leq 1$. As restrições (4-15) indicam o limite inferior de $u_{m,j,\bar{D},1}$ quando $k = 1$ e são geradas para o complemento $\bar{\delta}_j \mid \bar{\delta}_j \leq 1 - \pi_j$.

Suponha que a tarefa j tem data de liberação igual $\bar{d}_j = 8$, logo $\bar{D}_j = 2$ e $\bar{\delta}_j = 0{,}6$. Nesse exemplo, no *bucket* 2 a tarefa j pode ocupar no mínimo a fração de 0,6 e portanto $u_{m,j,b,k} \geq 0{,}6$. Qualquer valor assumido por $u_{m,j,b,k} < 0{,}6$, viola a data limite $\bar{d}_j = 8$. A Figura 4.12 apresenta o exemplo.

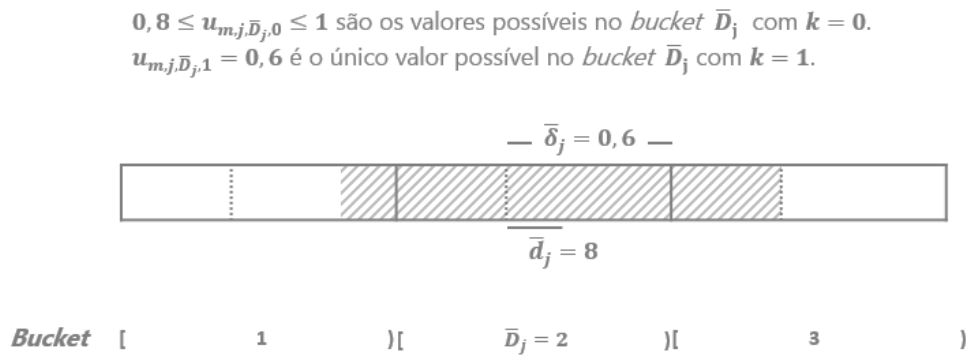

Figura 4.12: Exemplo de data limite da tarefa j

Ao incluir as restrições de data de liberação e data limite ao modelo, são necessárias alterações nas condições de geração das variáveis como pode ser observado nos Algoritmos 4 e 5.

Algoritmo 4: CÁLCULO $existeZeU_{m,j,b,k}$

Entrada: $J, M_j, B, K, P_j, \pi_j, R_j, \rho_j, \bar{D}_j, \bar{\delta}_j$
Saída: $existeZeU_{m,j,b,k}$

```
início
    para cada j ∈ J, m ∈ M_j, k ∈ K = {0,1} e
     b ∈ [max(1, R_j), min(B − P_j − k + 1, D̄_j)] faça
        se (π_j < 1 ou k = 0) e
        (1 − π_j < ρ_j ≤ 1 ou k = 1 ou b ≠ R_j) e
        (δ̄_j ≤ 1 − π_j ou k = 0 ou b ≠ D̄_j) então
            existeZeU_{m,j,b,k} = 1
        fim
    fim
fim
retorna existeZeU_{m,j,b,k}
```

Algoritmo 5: CÁLCULO $existeV_{m,j,b,k}$

Entrada: $J, M_j, B, K, P_j, \pi_j, R_j, \rho_j, \bar{D}_j, \bar{\delta}_j$

Saída: $existeV_{m,j,b,k}$

```
1  início
2      para cada j ∈ J, m ∈ M_j, k ∈ K = {0,1} e
         b ∈ [max(P_j + k, R_j + P_j + k − 1), min(B, D̄_j + P_j + k − 1)] faça
3          se (π_j < 1 ou k = 0) e
4          (1 − π_j < ρ_j ≤ 1 ou k = 1 ou b ≠ R_j + P_j + k − 1) e
5          (δ̄_j ≤ 1 − π_j ou k = 0 ou b ≠ D̄_j + P_j + k − 1) então
6              existeV_{m,j,b,k} = 1
7          fim
8      fim
9  fim
10 retorna existeV_{m,j,b,k}
```

As novas condições são relacionadas aos valores de ρ_j e $\bar{\delta}_j$. A primeira condição indica que se a fração máxima $\rho_j \leq 1 - \pi_j$ as variáveis $z_{m,j,R_j,0}$, $u_{m,j,R_j,0}$, $v_{m,j,R_j+P_j+k-1,0}$ são iguais a zero, isso porque, assumindo $\rho_j = 1 - \pi_j$ e $u_{m,j,R_j,k} = \rho_j$, pela relação $1-k < 1-\pi_j+\pi_j \leq 2-k,\ k = 1$, e para qualquer valor de $\rho_j < 1 - \pi_j$, a soma $u_{m,j,R_j,k} + \pi_j$ sempre indicará o valor de $k = 1$.

A segunda condição indica que se a fração mínima $1 - \pi_j < \bar{\delta}_j \leq 1$ as variáveis $z_{m,j,\bar{D}_j,1}$, $u_{m,j,\bar{D}_j,1}$, $v_{m,j,\bar{D}_j+P_j+k-1,1}$ são iguais a zero. De forma similar a análise anterior, pela relação de k, qualquer valor $\bar{\delta}_j > 1 - \pi_j,\ k = 0$.

4.3.4 Disponibilidade de Máquinas

As máquinas abordadas nesse problema, além de elegíveis devem estar disponíveis para execução da tarefa. Essas disponibilidades, assim como descrito no Capítulo 2, podem não ser contínuas. Para considerar essa característica do problema a variável $w_{a,j} \in \{0, 1\}$ que vale 1 caso a tarefa j inicie durante a disponibilidade a.

Assim como as restrições bases de programação, é necessário indicar que para toda tarefa j a variável $w_{a,j}$ assume o valor 1, somente uma vez como apresentado nas restrições (4-18). Além disso, as restrições (4-19) indicam que a variável $w_{a,j}$ só vale 1, se a variável $z_{m,j,b,k}$ vale 1 para máquina m tal que a disponibilidade $a \in A_m$. As restrições (4-20) definem o domínio das variáveis wa, j.

$$\sum_{m \in M_j} \sum_{a \in A_m} w_{a,j} = 1 \ \forall j \in J \tag{4-18}$$

$$\sum_{b \in [1,B]} \sum_{k \in K} z_{m,j,b,k} = \sum_{a \in A_m} w_{a,j} \ \forall m \in M_j \ \forall j \in J \tag{4-19}$$

$$w_{a,j} \in \{0,1\} \ \forall m \in M_j, \ \forall a \in A_m, \ \forall j \in J \tag{4-20}$$

Assim como as restrições de data de liberação e data limite citadas anteriormente, as disponibilidades das máquinas também influenciam nos limites das variáveis $u_{m,j,b,k}$. Para isso, é necessário identificar os *buckets* $S_a, F_{a,j} \in \mathbb{Z}^+$ que representam, em *buckets*, o início da disponibilidade a e o início mais tarde para que a tarefa j respeite o fim f_a. Portanto, $S_a = \left\lfloor \frac{s_a - 1}{\Delta} \right\rfloor + 1$ e $F_{a,j} = \left\lfloor \frac{f_a - p_j}{\Delta} \right\rfloor + 1$.

A máxima fração do *bucket* S_a que pode ser ocupada executando qualquer tarefa será de $\theta_a = S_a - \frac{s_a - 1}{\Delta}$ e limita superiormente a variável $u_{m,j,b,k}$ e a fração mínima que pode ser ocupada para cada tarefa j na disponibilidade a é $\phi_{a,j} = F_{a,j} - \frac{f_a - p_j}{\Delta}$ e limita inferiormente a variável $u_{m,j,b,k}$. As restrições (4-21)-(4-24) seguem a mesma estrutura das restrições (4-14)-(4-17).

$$u_{m,j,S_a,0} \leq \theta_a z_{m,j,S_a,0} \ \ \forall m, \forall j, \forall a | a \in A_m \wedge 1 - \pi_j < \theta_a \leq 1 \tag{4-21}$$

$$u_{m,j,S_a,1} \leq \theta_a z_{m,j,S_a,1} \ \ \forall m, \forall j, \forall a | a \in A_m \wedge \theta_a \leq 1 - \pi_j \tag{4-22}$$

$$\phi_{a,j} z_{m,j,F_{a,j},0} \leq u_{m,j,F_{a,j},0} \ \ \forall m, \forall j, \forall a | a \in A_m \wedge 1 - \pi_j < \phi_{a,j} \leq 1 \tag{4-23}$$

$$\phi_{a,j} z_{m,j,F_{a,j},1} \leq u_{m,j,F_{a,j},1} \ \ \forall m, \forall j, \forall a | a \in A_m \wedge \phi_{a,j} \leq 1 - \pi_j \tag{4-24}$$

Novamente são necessárias alterações nas condições de geração das variáveis para incorporar as restrições de disponibilidade das máquinas. As condições adicionadas são semelhantes às alterações feitas para incluir data de liberação e data limite, apresentadas nos Algoritmos 6 e 7, porém vale ressaltar

que ao incluir essas restrições os parâmetros $existeZeU_{m,j,b,k,a}$ e $existeV_{m,j,b,k,a}$ recebem o índice a de disponibilidade.

Algoritmo 6: CÁLCULO $existeZeU_{m,j,b,k,a}$

Entrada: $J, M_j, B, K, A, P_j, \pi_j, R_j, \rho_j, \bar{D}_j, \bar{\delta}_j, S_a, \theta_a, F_a, \phi_{a,j}$
Saída: $existeZeU_{m,j,b,k,a}$

início
  para *cada* $j \in J$, $m \in M_j$, $k \in K = \{0,1\}$, $a \in A_m$ e $b \in [max(S_a, R_j), min(F_{a,j}, \bar{D}_j)]$ **faça**
    se $(\pi_j < 1\ ou\ k = 0) e$
    $(1 - \pi_j < \rho_j \leq 1\ ou\ k = 1\ ou\ b \neq R_j)\ e$
    $(\bar{\delta}_j \leq 1 - \pi_j\ ou\ k = 0\ ou\ b \neq \bar{D}_j)\ e$
    $(1 - \pi_j < \theta_a \leq 1\ ou\ k = 1\ ou\ b \neq S_a)\ e$
    $(\phi_{a,j} \leq 1 - \pi_j\ ou\ k = 0\ ou\ b \neq F_{a,j})$ **então**
      $existeZeU_{m,j,b,k,a} = 1$
    fim
  fim
fim
retorna $existeZeU_{m,j,b,k,a}$

Algoritmo 7: CÁLCULO $existeV_{m,j,b,k,a}$

Entrada: $J, M_j, B, K, A, P_j, \pi_j, R_j, \rho_j, \bar{D}_j, \bar{\delta}_j, S_a, \theta_a, F_a, \phi_{a,j}$
Saída: $existeV_{m,j,b,k,a}$

início
  para *cada* $j \in J$, $m \in M_j$, $k \in K = \{0,1\}$, $a \in A_m$ e $b \in [max(S_a + P_j + k - 1, R_j + P_j + k - 1), min(F_{a,j} + P_j + k - 1, \bar{D}_j + P_j + k - 1)]$ **faça**
    se $(\pi_j < 1\ ou\ k = 0)\ e$
    $(1 - \pi_j < \rho_j \leq 1\ ou\ k = 1\ ou\ b \neq R_j + P_j + k - 1)\ e$
    $(\bar{\delta}_j \leq 1 - \pi_j\ ou\ k = 0\ ou\ b \neq \bar{D}_j + P_j + k - 1)\ e$
    $(1 - \pi_j < \theta_a \leq 1\ ou\ k = 1\ ou\ b \neq S_a + P_j + k - 1)\ e$
    $(\phi_{a,j} \leq 1 - \pi_j\ ou\ k = 0\ ou\ b \neq F_{a,j} + P_j + k - 1)$ **então**
      $existeV_{m,j,b,k,a} = 1$
    fim
  fim
fim
retorna $existeV_{m,j,b,k,a}$

A primeira condição adicionada ao Algoritmos 6 e 7 indica que se a fração máxima $\theta_a \leq 1 - \pi_j$ as variáveis $z_{m,j,S_a,0}$, $u_{m,j,S_a,0}$, $v_{m,j,S_a+P_j+k-1,0}$ são iguais a zero, isso porque, assumindo $\theta_a = 1 - \pi_j$ e $u_{m,j,S_a,k} = \theta_a$, pela relação $1 - k < 1 - \pi_j + \pi_j \leq 2 - k,\ k = 1$, e para qualquer valor de $\theta_a < 1 - \pi_j$, a soma $u_{m,j,S_a,k} + \pi_j$ sempre indicará o valor de $k = 1$.

A segunda condição indica que se a fração mínima $1 - \pi_j < \phi_{a,j} \leq 1$ as variáveis $z_{m,j,F_{a,j},1}$, $u_{m,j,F_{a,j},1}$, $v_{m,j,F_{a,j}+P_j+k-1,1}$ são iguais a zero. De forma similar a análise anterior, pela relação de k, qualquer valor $\phi_{a,j} > 1 - \pi_j,\ k = 0$.

Os novos domínios das variáveis $z_{m,j,b,k}, u_{m,j,b,k}$ e $v_{m,j,b,k}$ são definidos pelas restrições (4-25)-(4-27) a seguir.

$$z_{m,j,b,k} \in \{0,1\}\ \forall m \in M_j,\ \forall j \in J,\ \forall b \in [1,B],\ \forall k \in K \mid existeZeU_{m,j,b,k,a} = 1 \tag{4-25}$$

$$u_{m,j,b,k} \in [0,1]\ \forall m \in M_j,\ \forall j \in J,\ \forall b \in [1,B],\ \forall k \in K \mid existeZeU_{m,j,b,k,a} = 1 \tag{4-26}$$

$$v_{m,j,b,k} \in [0,1]\ \forall m \in M_j,\ \forall j \in J,\ \forall b \in [1,B],\ \forall k \in K \mid existeV_{m,j,b,k,a} = 1 \tag{4-27}$$

Por fim, o modelo de programação linear inteira mista proposto nessa dissertação se resume nas restrições (4-1)-(4-27), também expostas no Apêndice A. Essa formulação resolve o problema descrito no Capítulo 2, estendendo a formulação *bucket-indexed* de Boland et al. (2016).

No próximo capítulo será apresentado o estudo de caso feito com instâncias de uma indústria brasileira de óleo e gás para o problema de programação de sondas para construção de poços de petróleo, tema este que motivou o desenvolvimento da nova formulação.

5
Aplicações e Resultados

Este capítulo aborda o estudo de caso sobre o problema de programação de sondas, apresentando uma breve revisão da literatura sobre o tema e descrevendo as instâncias reais de indústria brasileira de óleo e gás que serão usadas para realizar os testes da nova formulação. Por fim, este capítulo apresenta os resultados obtidos pela formulação *bucket-indexed* proposta e a comparação com a formulação *time-indexed*.

5.1
Problema de Programação de Sondas

O problema de programação de sondas consiste em um conjunto de poços de petróleo, com tarefas associadas, que devem ser programadas em um conjunto de sondas disponíveis. A programação deve considerar restrições de data de atendimento para as tarefas e precedência entre tarefas do mesmo poço e entre poços, para garantir a execução correta do poço e a produção de óleo prevista, além de restrições técnicas e contratuais da sonda e sua disponibilidade.

O problema pode ser classificado de acordo com quatro características, são elas:

• O tipo de serviço a ser realizado no poço: tarefas de construção de poços, as quais são responsáveis por preparar o poço desde a perfuração até a completação, garantindo que o poço esteja pronto para ser interligado; tarefas de *workover*, as quais são essenciais para a manutenção da produção de óleo e gás pelos poços, podendo influenciar diretamente a curva de óleo estimada.

• O ambiente em que se encontra o campo petrolífero, terrestre ou marítimo, sendo o último com operações mais complexas, caras e arriscadas.

• As características técnicas e contratuais das sondas, que indicam se uma frota é homogênea, ou seja, todas as sondas possuem as mesmas características e qualquer uma delas pode executar qualquer tarefa, ou uma frota heterogênea, na qual existe um subconjunto de sondas que pode executar um subconjunto de tarefas, respeitando as necessidades técnicas e contratuais exigidas pelas mesmas.

• A abordagem utilizada para solucionar o problema: programação, geralmente usada quando o tempo de deslocamento entre poços é desprezível

e roteamento, quando os deslocamentos das sondas se tornam significativos e impactam na solução fornecida.

As instâncias reais a serem testadas usando a formulação proposta nesta dissertação referem-se ao problema de programação de sondas heterogêneas para construção de poços marítimos. A seguir, uma breve revisão da literatura mostra como o problema de programação de sondas está sendo estudado pelos pesquisadores.

5.2 Revisão da Literatura

Irgens et al. (2008) apontam que o planejamento e programação de uma frota de sondas são uma das principais atividades da indústria do setor de petróleo, devido aos altos custos associados a aquisição e operação das sondas, além da complexidade que envolve o sequenciamento de poços.

O problema de programação de sondas data desde a década de 60, quando Aronofsky e Williams (1962) abordaram o problema de forma simplificada, desenvolvendo dois modelos lineares para resolver o problema com o objetivo de maximizar o fluxo de caixa acumulado. Brennan et al. (1977) trataram o problema de programação de sondas de *workover* propondo duas técnicas aproximativas.

Devido à baixa capacidade computacional, as aplicações a problemas reais era limitada. Só a partir dos anos 90, com os avanços computacionais, as pesquisas foram retomadas. Eagle (1996) e Iyer et al. (1998) apresentaram, respectivamente, uma metaheurística *Simulated Annealing (SA)* e um modelo de programação linear inteira mista multiperíodo com técnicas de *Branch & Bound*, para solucionar a programação de sondas visando maximizar o valor presente líquido.

Nos anos 2000, houve um crescente aumento do número de artigos publicados tratando o problema de roteamento de sondas *workover* com o objetivo de minimizar os custos de sondas e a perda da produção de óleo. Paiva et al. (2000) desenvolveram uma metaheurítica *Simulated Annealing (SA)* e Aloise et al. (2002) testaram o uso da combinação entre a heurística Colônia de Formigas com *Path-Relinking*, obtendo as melhores soluções para as instâncias testadas se comparado com Algoritmo Genético e *Greedy Randomized Adaptive Search Procedure (GRASP).*

Costa e Ferreira Filho (2004) e Costa e Filho (2005) desenvolveram, respectivamente, Heurística de Máxima Prioridade Tricritério (HMPT) e Heurística de Montagem Dinâmica (HMD). As heurísticas foram testadas em 300 instâncias reais e a HMD apresentou os melhores resultados em todos os casos,

provando ser um bom algoritmo para resolver instâncias de pequeno e grande porte. O Algoritmo Genético (GA) proposto por Alves e Filho (2006) também foi comparado com as heurísticas de Costa e Ferreira Filho (2004) e Costa e Filho (2005), obtendo resultados competitivos em menor tempo computacional para instâncias de grande porte de uma frota de sondas homogêneas.

Aloise et al. (2006) desenvolveram um heurística *Variable Neighborhood Search (VNS)* para resolver o problema de programação de uma frota heterogênea de sondas de *workover* visando minimizar a perda de produção de óleo. O método foi testado em instâncias reais da bacia de Potiguar no nordeste brasileiro variando entre 100-200 poços e 10 sondas. Como resultado do trabalho, a heurística foi implementada na Petrobras, companhia responsável pelas instâncias.

Os artigos citados até então deram enfoque ao problema de programação de sondas *workover* terrestres. Só a partir de 2007, o número de de publicações aplicadas a outros tipos de sondas aumentaram. Neste ano, Litvak et al. (2007) desenvolveram uma metodologia para modelagem de reservatório *top-down* usando algoritmos genéticos para o desenvolvimento do campo de petróleo marítimo incluindo decisões de programação de sondas de perfuração e completação para uma frota homogênea de sondas. No ano seguinte, Onwunalu et al. (2008) aprimoraram os algoritmos de Litvak et al. (2007) através de procedimentos estatísticos que usam técnicas de *cluster* para construir uma correlação entre a função objetivo e os atributos, diminuindo o tempo de execução.

Irgens e Lavenue (2007) e Irgens et al. (2008) abordaram o problema de programação de sondas heterogêneas através de um algoritmo de Busca Local Estocástica com os objetivos de maximizar a produção de petróleo e minimizar os custos de transporte, considerando precedência dos poços e janelas de tempo. Além disso, os autores desenvolveram aplicações que fornecem visualização em tempo real, comparações de cronogramas e aprimoram as análises de indicadores. O mesmo problema também foi tratado por Gupta e Grossmann (2012) usando um modelo de programação inteira não-linear e não-convexa multiperíodo para maximizar o valor presente líquido.

Falex (2009) propôs um Algoritmo Genético para o problema de programação de uma frota heterogênea de sondas de perfuração visando minimizar os custos contratuais de sondas e de perda de produção associados ao atraso.Gonçalves (2009) também desenvolveu um Algoritmo Genético para o mesmo problema, porém o autor introduziu restrições complexas, como leis ambientais e regulatórias, e custos de deslocamento de sondas.

Retomando as abordagens do problema de programação de sondas *wor-*

kover, Douro e Lorenzoni (2009) testaram um Algoritmo Genético utilizando a técnica de melhoria 2-opt para frota de sondas homogêneas. No mesmo ano, Pacheco et al. (2009) implementaram uma heurística denominada *Bubble Swap* para minimizar a perda de produção de óleo associada à manutenção do poço. Os algoritmos propostos encontraram melhores soluções superando resultados encontrados na literatura como Costa e Ferreira Filho (2004), Costa e Filho (2005) e Alves e Filho (2006). Alguns anos depois, Pacheco et al. (2012) desenvolveram um GRASP híbrido com *Path-Relinking*, o qual se tornou um método confiável, obtendo melhores soluções do que Pacheco et al. (2009).

Diferente dos trabalhos citados, Al Gharbi (2011) também aborda o problema de programação e roteamento de sondas homogêneas para perfuração de poços terrestres. O autor analisou diversas heurísticas construtivas utilizando o algoritmo de Dijkstra e desenvolveu uma específica para um caso real. O algoritmo proposto resultou em um número maior de dias de operação, mas com 30% e 11,5% de melhoria no total de distância percorrida e custo total, respectivamente.

A fim de melhorar soluções para o problema de programação de *workover*, Ribeiro et al. (2011) propõem uma heurística baseada em *Simulated Annealing (SA)* usando três tipos de vizinhanças diferentes. A primeira vizinhança permite reordenar poços na mesma sonda, a segunda insere poços em outras sondas e a terceira efetua troca de poços entre sondas. A estratégia de vizinhança é escolhida aleatoriamente a cada iteração, gerando soluções boas em baixo tempo computacional. Ribeiro et al. (2011) trataram instâncias que variam entre 25-125 poços e 2-10 sondas. Este método também melhorou as soluções da literatura obtidas por outros métodos, como GRASP, Algoritmo Genético e *Scatter Search*.

No ano seguinte, Ribeiro et al. (2012a) apresentaram duas novas metaheurísticas *Clustering Search* e *Adaptive Large Neighborhood Search (ALNS)* e a comparam com o algoritmo *Iterated Local Search (ILS)*. Para todas as instâncias testadas, a metaheurística ALNS superou os outros algoritmos. No segundo artigo, Ribeiro et al. (2012b) mostraram a primeira abordagem exata para o problema de roteamento de uma frota heterogêneas de sondas *workover*. O algoritmo *Branch-Price-and-Cut (BPC)* revela sucesso resolvendo problemas de tamanhos reais em um tempo computacional satisfatório.

Além das abordagens heurísticas e de programação linear inteira, Bassi et al. (2012) foram os primeiros a utilizar técnicas de simulação-otimização para o problema de programação de sondas heterogêneas marítimas para tarefas de *workover*, onde o tempo de serviço é incerto. O objetivo é minimizar os custos de oportunidade sujeitos a restrições operacionais, obtendo soluções viáveis.

Em 2014 mais estudos sobre *workover* foram publicados por Ribeiro et al. (2014), Marques et al. (2014) e Bissoli et al. (2016). Ribeiro et al. (2014) analisam quatro heurísticas para uma frota heterogênea de sondas terrestres: a VNS de Aloise et al. (2006), o BPC de Ribeiro et al. (2012b), a ALNS Ribeiro et al. (2012a) e um Algoritmo Genético Híbrido (AGH). Todos os modelos consideram premissas e restrições realistas do problema. Os resultados do BPC, do ALNS e do AGH foram superiores ao VNS, com desvios de mais de 9% dos outros métodos. O algoritmo BPC encontrou boas soluções em pouco tempo em relação aos outros métodos, mas a qualidade das soluções são inferiores ao ALNS e ao AGH. O AGH foi o único método que encontrou todas as melhores soluções. Análises de sensibilidade foram feitas para detectar mudanças de desempenho devido à calibração de algoritmos.

Diferente dos artigos que tratam o problema de *workover* citados anteriormente, Marques et al. (2014) apresentaram um sistema de apoio à decisão que utiliza um modelo de programação linear inteira mista para dimensionar e programar uma frota homogênea de sondas de *workover* marítimas com o objetivo de minimizar o número total de sondas e maximizar sua utilização. O modelo proposto foi testado em três cenários diferentes e apresentou resultados excepcionais.

Bissoli et al. (2016) desenvolveu um algortimo *Adaptive Large Neighborhood Search* (ALNS) baseado em Ribeiro et al. (2012a) com uma função bi-objetiva para minimizar a perda de produção de petróleo e o número de sondas terrestres. Ao final do trablho é feita uma análise do *trade-off* entre perda de produção de petróleo e custos de afretamento de sondas.

Assim como Marques et al. (2014), Monemi et al. (2015) trataram o problema de programação de uma frota heterogênea de *workover* através de um novo modelo de programação linear inteira mista, baseado em formulações *arc-time indexed* e duas técnicas de solução: *Branch-Price-and-Cut* e *hyper-heuristic*. As instâncias usadas para experimentos variam de 10-200 tarefas e 3-12 sondas. Segundo os autores, a abordagem hiper-heurística foi capaz de encontrar soluções ótimas ou quase ótimas para qualquer instância em alguns segundos. Além disso, o método de aproximação encontrou soluções para instâncias maiores, onde o algoritmo exato falhou.

Pérez et al. (2016) propuseram uma reformulação decomposta do modelo linear binário de Costa e Ferreira Filho (2004) para o problema de programação de *workover* com uma frota homogênea de sondas terrestres. O modelo matemático apresentado encontrou novas soluções exatas para grandes instâncias com 125 poços e 10 sondas, superando outros métodos da literatura como algoritmo genético 2-opt de Douro e Lorenzoni (2009) e *Simulated Annealing*

de Ribeiro et al. (2011) com melhores soluções e menores tempos computacionais. Os autores também apontam a importância de testar tais modelos em poços marítimos.

Santos (2018) utiliza um modelo matemático e algoritmos de busca local visando determinar uma frota de sondas marítimas a fim de minimizar os custos de contratação, operação e ociosidade das sondas. Os resultados demonstraram que as buscas locais combinadas com uma solução inicial fornecida pelo modelo matemático entregam melhores soluções em relação ao modelo de programação linear inteira mista para cálculo do orçamento.

Carrilho et al. (2018) propõem a elaboração de uma formulação *bucket-indexed* multi-critério para o dimensionamento sondas marítimas heterogêneas para tarefas de construção de poços comparando os resultados com a clássica formulação *time-indexed*. As conclusões do trabalho evidenciam que, para as instâncias testadas, a formulação *bucket-indexed* proposta supera a formulação *time-indexed*, reduzindo os tempos computacionais.

O resultado das publicações encontradas nessa revisão de literatura estão sintetizados na Tabela 5.1. Os artigos são classificados de acordo com autores, ano de publicação, ambiente do campo de petróleo (terrestre/marítimo), tipo de tarefa (construção de poço/workover), definição do problema, técnica usada (heurística, programação matemática, simulação), direção de otimização (maximizar/minimizar) e função objetivo (valor presente líquido, produção de petróleo, perda de produção de óleo, etc.).

Tabela 5.1: Resumo com publicações e classificações do problema

Autores	Ano	Campo de Óleo	Tarefas	Frota de Sondas	Definição do Problema	Técnica	Direção da Otimização	Função Objetivo
ARONOFSKY; WILLIAMS	1962	Subterrâneo	Perfuração	Homogênea	Programação de Sondas	Programação Linear Paramétrica	Maximizar	Discounted Cumulative Cash Flow
BRENNAN et al.	1977	-	Workover	Homogênea	Programação de Sondas	Técnicas Aproximativas	Minimizar	Perda de Produção de Óleo
EAGLE	1996	Terrestre	Perfuração	Homogênea	Programação de Sondas	Simulated Annealing (SA)	Maximizar	Valor Presente Líquido (VPL)
IYER et al.	1998	Marítimo	Perfuração	Homogênea	Desenvolvimento do Campo de Óleo e Programação de Sondas	PLIM + Branch e Bound	Maximizar	Valor Presente Líquido (VPL)
PAIVA et al.	2000	Terrestre	Workover	Homogênea	Roteamento e Programação de Sondas	SA	Minimizar	Perda de Produção de Óleo + Custo de Transporte
ALOISE et al.	2002	Terrestre	Workover	Homogênea	Roteamento e Programação de Sondas	Colônia de Formigas + Path-Relinking	Minimizar	Perda de Produção de Óleo
COSTA; FERREIRA FILHO	2004	Terrestre	Workover	Homogênea	Roteamento e Programação de Sondas	HMPT	Minimizar	Perda de Produção de Óleo
COSTA; FILHO	2005	Terrestre	Workover	Homogênea	Roteamento e Programação de Sondas	HMD	Minimizar	Perda de Produção de Óleo
ALVES; FILHO	2006	Terrestre	Workover	Homogênea	Programação de Sondas	GA	Minimizar	Perda de Produção de Óleo
ALOISE et al.	2006	Terrestre	Workover	Heterogênea	Roteamento e Programação de Sondas	VNS	Minimizar	Perda de Produção de Óleo
LITVAK et al.	2007	Marítimo	Perfuração e Completação	Homogênea	Desenvolvimento do Campo de Óleo e Programação de Sondas	GA	Maximizar	Economic Indicators / Oil e Gas Recovery
ONWUNALU et al.	2008	Marítimo	Perfuração	Homogênea	Programação de Sondas	GA com estatística	Maximizar	Valor Presente Líquido (VPL)
IRGENS; LAVENUE	2007	Marítimo	Perfuração	Heterogênea	Programação de Sondas	Busca Local Estocástica	Maximizar / Minimizar	Produção de Óleo e Custo de Transporte
IRGENS et al.	2008	Marítimo	Perfuração	Heterogênea	Programação de Sondas	Busca Local Estocástica	Maximizar / Minimizar	Produção de Óleo e Custo de Transporte
GUPTA; GROSSMANN	2012	Marítimo	Perfuração	-	Desenvolvimento do Campo de Óleo e Programação de Sondas	Programação Não Linear Inteira Mista Não Convexa Multiperíodo	Maximizar	Net Present Value (NPV)
FALEX	2009	Marítimo	Perfuração	Heterogênea	Programação de Sondas	Simulação de Reservatório	Minimizar	Net Present Loss (NPL)
GONÇALVES	2009	Marítimo	Perfuração	Heterogênea	Roteamento e Programação de Sondas	GA	Maximizar	Expected Monetary Value (EMV)
DOURO; LORENZONI	2009	Terrestre	Workover	Homogênea	Programação de Sondas	GA-2opt	Minimizar	Perda de Produção de Óleo
PACHECO et al.	2009	Terrestre	Workover	Homogênea	Programação de Sondas	Bubble Swap	Minimizar	Perda de Produção de Óleo
PACHECO et al.	2012	Terrestre	Workover	Homogênea	Roteamento e Programação de Sondas	GRASP Híbrido com Path-Relinking	Minimizar	Perda de Produção de Óleo
AL GHARBI	2011	Terrestre	Perfuração	Homogênea	Roteamento e Programação de Sondas	Heurísticas construtivas com Dijkstra	Minimizar	Custo de Operação + Custo de Movimentação
RIBEIRO et al.	2011	Terrestre	Workover	Homogênea	Roteamento e Programação de Sondas	SA	Minimizar	Perda de Produção de Óleo
RIBEIRO et al.	2012a	Terrestre	Workover	Homogênea	Roteamento e Programação de Sondas	ILS, Clustering Search (CS) e ALNS	Minimizar	Perda de Produção de Óleo
RIBEIRO et al.	2012b	Terrestre	Workover	Heterogênea	Roteamento e Programação de Sondas	Branch-Price-e-Cut	Minimizar	Perda de Produção de Óleo
BASSI et al.	2012	Marítimo	Workover	Heterogênea	Programação de Sondas	Simulação-Optimização (GRASP)	Minimizar	Perda de Produção de Óleo
RIBEIRO et al.	2014	Terrestre	Workover	Heterogênea	Roteamento e Programação de Sondas	VNS, Branch-Price-e-Cut, ALNS e GA-Híbrido	Minimizar	Perda de Produção de Óleo
MARQUES et al.	2014	Marítimo	Workover	Homogênea	Programação de Sondas	PLIM	Minimizar	Número de Sondas
BISSOLI et al.	2016	Terrestre	Workover	Homogênea / Heterogênea	Roteamento e Programação de Sondas	PLIM e ALNS	Minimizar	Perda de Produção de Óleo e Custos de Sonda
MONEMI et al.	2015	Terrestre	Workover	Heterogênea	Programação de Sondas	PLIM, Branch-Price-e-Cut e Hiper-Heurística	Minimizar	Perda de Produção de Óleo
PÉREZ et al.	2016	Terrestre	Workover	Homogênea	Programação de Sondas	Programação Linear Binária	Minimizar	Perda de Produção de Óleo
SANTOS	2018	Marítimo	Perfuração e Completação	Heterogênea	Programação de Sondas	PLIM + Busca Local	Minimizar	Custo de Contratação + Custo de Operação + Custo Ociosidade
CARRILHO et al.	2018	Marítimo	Perfuração e Completação	Heterogênea	Programação de Sondas	PLIM	Minimizar	Número de Sondas
Essa dissertação	2019	Marítimo	Perfuração e Completação	Heterogênea	Programação de Sondas	PLIM	Minimizar	Tempo de Conclusão das Tarefas

Com base na literatura e como apresentado na Tabela 5.1, nota-se uma pesquisa extensa abordando o problema de roteamento e programação de sondas de *workover* e uma escassez de publicações que abordam o problema de programação de sondas heterogêneas para o construção de poços. Essa dissertação visa preencher uma lacuna da literatura, abordando o problema de programação de sondas heterogêneas para tarefas de construção de poços marítimos usando uma formulação de programação inteira mista chamada *bucket-indexed* e aplicando o modelo em instâncias reais em uma indústria brasileira de óleo e gás que serão descritas a seguir.

5.3 Descrição das Instâncias

As instâncias usadas para testar a formulação proposta consistem em dados reais de uma indústria de óleo e gás brasileira para o problema de programação de sondas marítimas para construção de poços. Esse problema pode ser tratado como um problema de máquinas paralelas idênticas, tal como o descrito no Capítulo 2.

Os dados consistem em um conjunto de tarefas de construção de poços como perfuração, avaliação e completação, as quais são executadas sempre nessa sequência, mantendo uma relação de precedência entre si. Geralmente, os tempos de processamento dessas tarefas são grandes, motivando o uso da formulação *bucket-indexed*. Dado que o tamanho do *bucket* assuma o valor do menor tempo de processamento, se os tempos de processamento são grandes, a cardinalidade do conjunto de *buckets* será menor, diminuindo o porte do modelo.

Nas instâncias tratadas o tempo de processamento das tarefas não depende da sonda que está executando-as, análogo ao problema com máquinas idênticas descrito. No entanto, características técnicas e contratuais indicam que as sondas são elegíveis e por isso executam apenas um determinado subconjunto de tarefas.

As sondas estão disponíveis durante o período contratual, porém alguns eventos como adequação operacional e paradas programadas podem bloquear parte desse período, fazendo com que haja períodos de disponibilidade não contínuos.

Os dados compreendem um horizonte de planejamento de cerca de 5 anos para programação das tarefas, considerado o período de planejamento estratégico da companhia.

5.4 Resultados

Nesta seção serão apresentadas as instâncias reais usadas para os experimentos computacionais e os resultados obtidos pelos modelos matemáticos *time-indexed* (TI) e *bucket-indexed* (BI) desenvolvidos. Ao final da seção uma análise comparativa será feita a fim de mostrar as vantagens e desvantagens da nova formulação.

Seis instâncias reais do problema de programação de sondas marítimas para construção de poços foram usadas para testar os modelos desenvolvidos nessa dissertação. As instâncias estão apresentadas na Tabela 5.2, onde estão descritos a quantidade de tarefas, sondas e indisponibilidades, bem como o número de períodos e *buckets*, dado o tamanho do *bucket* de $\Delta = \min_{j \in J} p_j$.

Tabela 5.2: Instâncias

Instância	# Tarefas	# Sondas	# Indisponibilidades	# Períodos	# *Buckets*	Δ
1	15	10	84	2464	36	69
2	68	31	74	2401	62	39
3	110	32	88	2261	162	14
4	100	35	68	2296	329	7
5	324	41	20	2345	336	7
6	638	73	155	2464	353	7

Os modelos foram implementados na linguagem de programação Julia utilizando JuMP. Os experimentos computacionais foram executados no computador com Intel® Core™ i5-4440 CPU 3.10 GHz com 16.00 GB RAM de memória usando o *solver* Gurobi 8.1.0. Para fins de comparação entre as formulações TI e BI limitamos o tempo de execução dos modelos em uma hora (3600 s).

Foram testadas duas abordagens para tratar os períodos indisponíveis das máquinas. A primeira delas consiste em criar uma tarefa para cada período indisponível da máquina. Essas tarefas, chamadas de indisponibilidade na Tabela 5.2, tem início e fim definidos de acordo com o período indisponível representado. Neste modelo, o conjunto de tarefas J seria formado pelas tarefas e as indisponibilidades, sendo sua cardinalidade $|J| = \#Tarefas + \#Indisponibilidades$. Os resultados estão apresentados na Tabela 5.3.

Na Tabela 5.3 são apresentados para cada um dos modelos o número de variáveis e restrições geradas, o *gap* percentual alcançado e o tempo de solução em segundos. Vale observar que o número de variáveis informadas para o modelo TI são sempre inteiras, diferente do modelo BI em que são informados o número de total de variáveis indicando as variáveis inteiras e contínuas do modelo e entre parênteses apenas o número de variáveis inteiras.

Tabela 5.3: Resultados - Indisponibilidade como Tarefa

	TI				BI			
Instância	# Variáveis	# Restrições	Gap (%)	Tempo (s)	# Variáveis	# Restrições	Gap (%)	Tempo (s)
1	27170	51849	0	5,55	2760(920)	8455	0	**0,04**
2	381073	458045	0	111,25	62232(20744)	174402	0	**1,42**
3	591415	688851	0	115,25	248913(82971)	684608	0	**8,94**
4	452755	535590	0	60,30	377667(125889)	1036124	0	**18,69**
5	2072715	2169289	0	345,83	1714842(571614)	4623942	0	**308,82**
6	-	-	-	-	5881308(1960436)	15815621	100	3600

Nessa abordagem, cinco das seis instâncias foram resolvidas na otimalidade por ambos os modelos. No entanto, o que chama atenção é o tempo de solução para atingir o ótimo. Como é possível notar na Tabela 5.3, os tempos destacados em negrito são os menores tempos de solução para cada instância, e todos eles foram alcançados pela formulação BI proposta. Isso se deve ao fato de que o número de variáveis inteiras geradas pelo modelo BI é bem menor que em relação as variáveis inteiras geradas pelo modelo TI. Em contrapartida, o número de restrições do modelo BI é bem maior se comparado ao modelo TI, mas isso não influencia no desempenho do modelo, pois como visto no Capítulo 4, a maioria dessas restrições são geradas para definir os limites das variáveis contínuas.

A sexta instância, de maior porte dentre as instâncias testadas, foi executada pelo modelo BI, mas com o tempo de 3600 s não conseguiu melhorar a solução, mantendo um *gap* de 100%. Para essa instância o modelo TI não conseguiu ser construído devido a falta de memória da máquina usada para os testes. Essa situação aponta que o modelo BI não resolve bem todas as instâncias de grande porte, isso porque, assim como a modelo TI, o modelo também cresce de forma pseudopolinomial.

A segunda abordagem para tratar o problema de períodos indisponíveis nas máquinas consiste em limitar a geração de variáveis pela interseção entre as datas de liberação e datas limite da tarefa com a janela de disponibilidade da máquina. Para o modelo TI, isso é feito na condição de criação da variável $x_{m,j,t}$ apresentada no Algoritmo 1. No modelo BI, além de controlar a criação das variáveis em *buckets* que exista disponibilidade de máquina descrita nos Algoritmos 6 e 7, é necessário também controlar o limite das variáveis contínuas através das restrições (4-21)-(4-24).

O resultados da segunda abordagem estão apresentados na Tabela 5.4. Tal como a Tabela 5.3, a Tabela 5.4 apresenta o número de variáveis e restrições, o *gap* percentual e tempo de solução em segundos.

Os resultados desta segunda abordagem também indicam que cinco das seis instâncias foram resolvidas na otimalidade por ambos os modelos. Assim como nos resultados anteriores, o modelo BI teve desempenho superior

Tabela 5.4: Resultados - Disponibilidade como Limite de Variável

	TI				BI			
Instância	# Variáveis	# Restrições	Gap (%)	Tempo (s)	# Variáveis	# Restrições	Gap (%)	Tempo (s)
1	2350	27009	0	0,32	306(102)	1631	0	**0,01**
2	123723	210231	0	40,27	20652(6884)	61372	0	**0,92**
3	290030	391900	0	68,79	122058(40686)	343908	0	**7,87**
4	246882	334235	0	31,75	206616(68872)	579128	0	**10,55**
5	1000894	1102074	0	955,81	834186(278062)	2266414	0	**179,25**
6	2771246	2973990	100	3600	2275323(758441)	6162926	**74,39**	3600

ao modelo TI em relação aos tempos de solução destacados na Tabela 5.4. Além disso, é possível observar que entre as duas abordagens o modelo BI desempenhou melhor nesta segunda.

Nessa abordagem, a sexta instância foi executada pelo modelo BI alcançando em 3600 s um *gap* de 74,39%. Dessa vez, a instância também conseguiu ser executada pelo modelo TI mas manteve o *gap* de 100% ao final dos 3600 s.

Após realizar os testes com as duas abordagens, pode-se concluir que para ambos os modelos a segunda abordagem superou a primeira sugerindo que para esse problema a melhor forma de tratar os períodos indisponíveis de máquinas é gerando cortes nas variáveis e não acrescentando tarefas de indisponibilidade a cada instância.

Além disso, vale ressaltar que o bom desempenho do modelo BI em todas as instâncias, se deve ao fato de que o tamanho do *bucket* (Δ) é sempre maior do que 2. Se o *bucket* tem tamanho $\Delta = 1$, o número de variáveis $z_{m,j,b,k}$ é o dobro do número de variáveis $x_{m,j,t}$ devido ao conjunto binário k, se $\Delta = 2$ o número de variáveis $z_{m,j,b,k}$ é igual ao número de variáveis $x_{m,j,t}$ porém com uma relaxação linear pior que gerada pela formulação TI. Para qualquer valor de $\Delta > 2$, a formulação BI começa a gerar menos variáveis inteiras melhorando o desempenho.

Por fim, também conclui-se que a formulação BI apesar de algumas limitações, como o tamanho do *bucket* $\Delta \leq 2$, é uma boa alternativa de programação linear inteira mista para tratar problemas de programação de máquinas paralelas idênticas, obtendo resultados ótimos em menores tempos computacionais que a formulação TI.

6
Considerações finais

Esta dissertação teve como objetivos propor uma formulação *bucket-indexed* para a programação de máquinas paralelas idênticas e resolver instâncias reais de uma indústria de óleo e gás brasileira para o problema de programação de sondas marítimas para tarefas de construção de poços.

Os objetivos foram alcançados, explicitando as principais contribuições do trabalho. A nova formulação *bucket-indexed*, baseada no modelo de Boland et al. (2016), contribui para o conhecimento teórico, estendendo a literatura de métodos exatos para resolver o problema de programação de máquinas paralelas. Além disso, a abordagem de características de problemas práticos da indústria na formulação contribui para aplicações do modelo em instâncias reais.

O problema de programação de máquinas idênticas tratado consiste em programar um conjunto de tarefas em máquinas durante um horizonte de planejamento, considerando restrições de data de liberação e data limite para o início das tarefas, relação de precedência entre pares de tarefas, elegibilidade e disponibilidade de máquinas. Para resolver este problema, duas formulações foram apresentadas: a clássica formulação *time-indexed* (TI) e a formulação *bucket-indexed* (BI) proposta.

O modelo de programação linear inteira TI é um método exato bastante usado na maioria dos problemas complexos de planejamento da produção devido a simplicidade da implementação. Esse modelo tem uma relaxação linear forte, sendo usada para desenvolver planos de cortes, geração de colunas, algoritmos aproximativos. A desvantagem dessa formulação está relacionada ao seu porte, que cresce pseudopolinomialmente com o tamanho da instância, podendo em alguns casos ser intratáveis.

A formulação BI consiste na discretização do horizonte de planejamento em períodos de tempo mais longos chamados de *buckets*, os quais tem tamanho definido por um valor entre 1 e o menor tempo de processamento da instância. Quanto maior o tamanho do *bucket*, menor é o porte do modelo gerado. O modelo de programação linear inteira mista proposto nesta dissertação não corrige a desvantagem do crescimento pseudopolinomial, mas a mitiga a partir da escolha do tamanho do *bucket*: quanto maior o tamanho, menor é o crescimento do porte do modelo.

A formulação proposta é uma extensão do trabalho de Boland et al. (2016) e aborda características de problemas reais da indústria. As contribuições da dissertação incluem novas condições de geração de variáveis, o tratamento de máquinas paralelas idênticas considerando elegibilidade e disponibilidade das mesmas e a relação de precedência entre pares de tarefas.

A fim de avaliar o modelo proposto, instâncias de uma indústria de óleo e gás brasileira para o problema de programação de sondas marítimas para construção de poços durante um horizonte de 5 anos foram testadas. Na literatura o problema de programação de sondas heterogêneas para a construção de poços é pouco abordado, se comparado com problemas de roteamento de sondas de *workover*, sendo esta dissertação importante por abordar um tema que é de grande relevância prática devido aos altos investimentos associados aos custos de sondas.

Uma característica comum a essas instâncias beneficia a aplicação dessa formulação: os longos tempos de processamento das tarefas, que permitem discretizar o horizonte de planejamento em menos *buckets* e, consequentemente, gerando modelos de menor porte.

Duas abordagens foram testadas utilizando as formulações TI e BI. A primeira delas consiste em tratar os períodos indisponíveis das máquinas como tarefas de indisponibilidade. Para cinco das seis instâncias utilizadas, os modelos chegaram a otimalidade, porém a formulação BI em menor tempo computacional. A instância 6, de maior porte, não foi resolvida por nenhum dos modelos, destacando que o modelo BI também não lida bem com instâncias de grande porte.

A segunda abordagem inclui os períodos indisponíveis das máquinas como condição de criação das variáveis e restrições de limite das variáveis contínuas, para o caso da formulação BI. Assim como a primeira abordagem, os resultados mostraram que para as mesmas cinco instâncias o modelo BI supera o modelo TI alcançando a otimalidade em menor tempo computacional. A sexta instância por sua vez foi executada por ambos os modelos, mas devido ao limite de tempo de 3600s conclui sua execução com *gaps* muito altos: 74,39% para o modelo BI e 100% para o modelo TI.

Dos resultados obtidos pela nova formulação é possível notar que apesar de alcançar melhor desempenho na maioria das instâncias testadas, reduzindo os tempos computacionais das soluções, a formulação BI ainda tem limitações quanto ao tamanho do modelo em relação a instâncias de grande porte, sugerindo mais estudos sobre a técnica.

6.1 Sugestões para trabalhos futuros

Alguns aspectos importantes podem ser estudados na nova formulação proposta a fim de aprimorar o modelo, considerando novas restrições práticas do problema de programação de máquinas paralelas. Portanto, algumas recomendações para trabalhos futuros incluem:

• Considerar *buckets* com tamanhos variando de acordo com a elegibilidade da máquina, i.e., o tamanho do *bucket* igual ao menor tempo de processamento das tarefas que podem ser executados na máquina.

• Avaliar as instâncias da indústria utilizando outras funções objetivo que envolvam métricas de orçamento.

• Estudar o efeito do tamanho do *bucket* em relação ao tempo computacional, qualidade da solução e relaxação linear.

• Incorporar incertezas ao modelo, desenvolvendo uma formulação *bucket-indexed* estocástica e analisar como se comporta a solução com o tamanho do *bucket* variando a cada cenário.

• Estudar aplicações de planos de cortes e geração de colunas no modelo *bucket-indexed* para melhorar tempos computacionais de instâncias que ainda apresentam altos tempos computacionais.

7
Referências bibliográficas

AL GHARBI, S. H. **Drilling rig schedule optimization**. Dissertação (Mestrado em Ciência da Computação) — King Fahd University of Petroleum and Minerals, Saudi Arabia, 2011. 5.2, 5.1

ALOISE, D.; NORONHA, T.; MAIA, R.; BITTENCOURT, V. G.; ALOISE, D. J. Heurísticas de colônia de formigas com path-relinking para o problema de otimização da alocação de sondas de produção terrestre–spt. In: **Anais do XXXIV SBPO - Simpósio Brasileiro de Pesquisa Operacional**. Rio de Janeiro, RJ, Brasil: SOBRAPO, 2002. 5.2, 5.1

ALOISE, D. J.; ALOISE, D.; ROCHA, C. T.; RIBEIRO, C. C.; FILHO, J. C. R.; MOURA, L. S. Scheduling workover rigs for onshore oil production. **Discrete Applied Mathematics**, v. 154, n. 5, p. 695–702, 2006. 5.2, 5.1

ALVES, V.; FILHO, V. J. M. F. Proposta de algoritmo genético para a solução do problema de roteamento e sequenciamento de sondas de manutenção. In: **Anais do XXXVIII Simpósio Brasileiro de Pesquisa Operacional**. Goiânia, GO, Brasil: SOBRAPO, 2006. 5.2, 5.1

ARONOFSKY, J.; WILLIAMS, A. The use of linear programming and mathematical models in under-ground oil production. **Management Science**, v. 8, n. 4, p. 394–407, 1962. 5.2, 5.1

BAPTISTE, P.; SADYKOV, R. On scheduling a single machine to minimize a piecewise linear objective function: A compact MIP formulation. **Naval Research Logistics (NRL)**, v. 56, n. 6, p. 487–502, 2009. 3.1

BASSI, H. V.; FILHO, V. J. M. F.; BAHIENSE, L. Planning and scheduling a fleet of rigs using simulation–optimization. **Computers & Industrial Engineering**, v. 63, n. 4, p. 1074–1088, 2012. 5.2, 5.1

BEHERA, D. K. Complexity on parallel machine scheduling: A review. In: **Proceedings of the INCOSET 2012 – International Conference on Emerging Trends in Science**. Tiruchirappalli, Tamilnadu, India: Engineering and Technology, 2012. p. 373–381. 1

BISSOLI, D. C.; CHAVES, G. L. D.; RIBEIRO, G. M. Drivers to the workover rig problem. **Journal of Petroleum Science and Engineering**, v. 139, p. 13–22, 2016. 5.2, 5.1

BOLAND, N.; CLEMENT, R.; WATERER, H. A bucket indexed formulation for nonpreemptive single machine scheduling problems. **INFORMS Journal on Computing**, v. 28, n. 1, p. 14–30, 2016. 1, 3.1, 4.1, 4.2, 4.3, 4.3, 8, 12, 6, A

BRENNAN, J. J.; BARNES, J. W.; KNAPP, R. M. Scheduling a backlog of oilwell workovers. **Journal of Petroleum Technology**, v. 29, n. 12, p. 1651–1653, 1977. 5.2, 5.1

CARRILHO, L. M.; ANDRADE, T.; RIBAS, G.; HAMACHER, S. Dimensionamento e sequenciamento de sondas usando a formulação bucket-indexed. In: **Anais do 50º SBPO - Simpósio Brasileiro de Pesquisa Operacional**. Rio de Janeiro, RJ, Brasil: SOBRAPO, 2018. 5.2, 5.1

CHENG, T.; SIN, C. A state-of-the-art review of parallel-machine scheduling research. **European Journal of Operational Research**, v. 47, n. 3, p. 271–292, 1990. 2

CORDONE, R.; HOSTEINS, P.; RIGHINI, G. A branch-and-bound algorithm for the prize-collecting single-machine scheduling problem with deadlines and total tardiness minimization. **INFORMS Journal on Computing**, v. 30, n. 1, p. 168–180, 2017. 3.1

COSTA, L. R.; FERREIRA FILHO, V. J. M. Uma heurística para o problema do planejamento de itinerários de sondas em intervenções de poços de petróleo. In: **Anais do XXXVI Simpósio Brasileiro de Pesquisa Operacional**. São João del Rei, MG, Brasil: SOBRAPO, 2004. 5.2, 5.1

COSTA, L. R.; FILHO, V. J. M. F. Uma heurística de montagem dinâmica para o problema de otimização de itinerários de sondas. In: **Anais do XXXVII Simpósio Brasileiro de Pesquisa Operacional**. Gramado, RS, Brasil: SOBRAPO, 2005. 5.2, 5.1

DOURO, R. F.; LORENZONI, L. L. Um algoritmo genético-2opt aplicado ao problema de otimização de itinerário de sondas de produção terrestre. In: **Anais do XLI Simpósio Brasileiro de Pesquisa Operacional**. Porto Seguro, BA, Brasil: SOBRAPO, 2009. 5.2, 5.1

DYER, M. E.; WOLSEY, L. A. Formulating the single machine sequencing problem with release dates as a mixed integer program. **Discrete Applied Mathematics**, v. 26, n. 2-3, p. 255–270, 1990. 3.1

EAGLE, K. Using simulated annealing to schedule oil field drilling rigs. **Interfaces**, v. 26, n. 6, p. 35–43, 1996. 5.2, 5.1

EDIS, E. B.; OZKARAHAN, I. A combined integer/constraint programming approach to a resource-constrained parallel machine scheduling problem with machine eligibility restrictions. **Engineering Optimization**, v. 43, n. 2, p. 135–157, 2011. 3.1

FALEX, A. **Planejamento da frota de sondas para atendimento de uma campanha de perfuração de um campo**. Tese (Doutorado em Engenharia Petróleo) — Universidade Federal do Rio de Janeiro, Rio de Janeiro, RJ, Brasil, 2009. 5.2, 5.1

GONÇALVES, R. K. **Otimização de alocação de sondas para exploração off-shore de hidrocarbonetos por algoritmos genéticos**. Dissertação

(Especialização em Business Intelligence) — Departmento de Engenharia Elétrica, Pontíficia Universidade Católica do Rio de Janeiro, Rio de Janeiro, RJ, Brasil, Rio de Janeiro, RJ, Brasil, 2009. 5.2, 5.1

GRAHAM, R. L.; LAWLER, E. L.; LENSTRA, J. K.; KAN, A. R. Optimization and approximation in deterministic sequencing and scheduling: a survey. **Annals of Discrete Mathematics**, v. 5, p. 287–326, 1979. 2

GUPTA, V.; GROSSMANN, I. E. An efficient multiperiod MINLP model for optimal planning of offshore oil and gas field infrastructure. **Industrial & Engineering Chemistry Research**, v. 51, n. 19, p. 6823–6840, 2012. 5.2, 5.1

IRGENS, M.; GUZMAN, R. P.; STAMATOPOULOS, J. G.; JACKSON, K. Optimization for operational decision support: The rig management case. In: **SPE Annual Technical Conference and Exhibition**. Denver, Colorado, USA: Society of Petroleum Engineers, 2008. 5.2, 5.1

IRGENS, M.; LAVENUE, W. Use of advanced optimization techniques to manage a complex drilling schedule. In: **SPE Annual Technical Conference and Exhibition**. Anaheim, CA, USA: Society of Petroleum Engineers, 2007. 5.2, 5.1

IYER, R.; GROSSMANN, I.; VASANTHARAJAN, S.; CULLICK, A. Optimal planning and scheduling of offshore oil field infrastructure investment and operations. **Industrial & Engineering Chemistry Research**, v. 37, n. 4, p. 1380–1397, 1998. 5.2, 5.1

JÄGER, S. Approximating total weighted completion time on identical parallel machines with precedence constraints and release dates. **Operations Research Letters**, v. 46, n. 5, p. 505–509, 2018. 3.1

KAABI, J.; HARRATH, Y. A survey of parallel machine scheduling under availability constraints. **International Journal of Computer and Information Technology**, v. 3, n. 2, p. 238–245, 2014. 1

LI, S. Scheduling to minimize total weighted completion time via time-indexed linear programming relaxations. In: **2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS)**. Berkeley, CA, USA: IEEE, 2017. p. 283–294. 3.1

LITVAK, M. L.; GANE, B. R.; WILLIAMS, G.; MANSFIELD, M.; ANGERT, P. F.; MACDONALD, C. J.; MCMURRAY, L. S.; SKINNER, R. C.; WALKER, G. J. et al. Field development optimization technology. In: **SPE reservoir simulation symposium**. Houston, TX, USA: Society of Petroleum Engineers, 2007. 5.2, 5.1

LUSTOSA, L. J.; MESQUITA, M. A. de; QUELHAS, O.; OLIVEIRA, R. J. **Planejamento e controle da produção**. Rio de Janeiro, RJ, Brasil: Elsevier, 2008. 2

MARQUES, L. C.; MACHADO, F. A. P. de P.; OLIVEIRA, F. C. de; HAMACHER, S. Sizing and scheduling resources: A decision support system applied to oil rigs scheduling. In: **XLVI SBPO - Simpósio Brasileiro de Pesquisa Operacional**. Salvador, BA, Brasil: SOBRAPO, 2014. 5.2, 5.1

MOKOTOFF, E. Parallel machine scheduling problems: A survey. **Asia-Pacific Journal of Operational Research**, v. 18, n. 2, p. 193, 2001. 1

MONEMI, R. N.; DANACH, K.; KHALIL, W.; GELAREH, S.; JR, F. C. L.; ALOISE, D. J. Solution methods for scheduling of heterogeneous parallel machines applied to the workover rig problem. **Expert Systems with Applications**, v. 42, n. 9, p. 4493–4505, 2015. 5.2, 5.1

NOCEDAL, J.; WRIGHT, S. J. Sequential quadratic programming. **Numerical optimization**, Springer, p. 529–562, 2006. 3.1

ONWUNALU, J. E.; LITVAK, M. L.; DURLOFSKY, L. J.; AZIZ, K. et al. Application of statistical proxies to speed up field development optimization procedures. In: **Proceedings of Abu Dhabi International Petroleum Exhibition and Conference**. Abu Dhabi, United Arab Emirates: Society of Petroleum Engineers, 2008. 5.2, 5.1

ÖZPEYNIRCI, S. A heuristic approach based on time-indexed modelling for scheduling and tool loading in flexible manufacturing systems. **The International Journal of Advanced Manufacturing Technology**, v. 77, n. 5-8, p. 1269–1274, 2015. 3.1

PACHECO, A. V. F.; DIAS FILHO, A. C. T.; RIBEIRO, G. M. Uma heurística para o problema da alocação de sondas de produção em poços de petróleo. Salvador, BA, Brasil, 2009. 5.2, 5.1

PACHECO, A. V. F.; RIBEIRO, G. M.; MAURI, G. R. A grasp with path-relinking for the workover rig scheduling problem. **Nature-Inspired Computing Design, Development, and Applications**, p. 90–103, 2012. 5.2, 5.1

PAIVA, R.; SCHIOZER, D.; BORDALO, S. Optimizing the itinerary of workover rigs. In: **16th World Petroleum Congress**. Calgary, Canada: World Petroleum Congress, 2000. 5.2, 5.1

PÉREZ, M.; OLIVEIRA, F.; HAMACHER, S. A new mathematical model for the workover rig scheduling problem. **Pesquisa Operacional**, v. 36, n. 2, p. 241–257, 2016. 5.2, 5.1

PINEDO, M. L. **Scheduling: theory, algorithms, and systems**. 5th. ed. New York: Springer, 2016. 1, 2

RIBEIRO, G. M.; DESAULNIERS, G.; DESROSIERS, J. A branch-price-and-cut algorithm for the workover rig routing problem. **Computers & Operations Research**, v. 39, n. 12, p. 3305–3315, 2012a. 5.2, 5.1

RIBEIRO, G. M.; DESAULNIERS, G.; DESROSIERS, J.; VIDAL, T.; VIEIRA, B. S. Efficient heuristics for the workover rig routing problem with a heterogeneous fleet and a finite horizon. **Journal of Heuristics**, v. 20, n. 6, p. 677–708, 2014. 5.2, 5.1

RIBEIRO, G. M.; LAPORTE, G.; MAURI, G. R. A comparison of three metaheuristics for the workover rig routing problem. **European Journal of Operational Research**, v. 220, n. 1, p. 28–36, 2012b. 5.2, 5.1

RIBEIRO, G. M.; MAURI, G. R.; LORENA, L. A. N. A simple and robust simulated annealing algorithm for scheduling workover rigs on onshore oil fields. **Computers & Industrial Engineering**, v. 60, n. 4, p. 519–526, 2011. 5.2, 5.1

RIEDLER, D.-I. M. **Advances in Decomposition Approaches for Mixed Integer Linear Programming**. Tese (Doutorado em Ciências Técnicas) — Technische Universität Wien, 2018. 4.1

RIEDLER, M.; JATSCHKA, T.; MASCHLER, J.; RAIDL, G. R. An iterative time-bucket refinement algorithm for a high-resolution resource-constrained project scheduling problem. **International Transactions in Operational Research**, 2017. 4.1

SANTOS, I. M. **Mathematical Programming Models and Local Search Algorithms for the Offshore Rig Scheduling Problem**. Dissertação (Mestrado em Engenharia de Produção) — Departamento de Engenharia Industrial, Pontíficia Universidade Católica, Rio de Janeiro, RJ, Brasil, 2018. 5.2, 5.1

SOUSA, J. P.; WOLSEY, L. A. A time indexed formulation of non-preemptive single machine scheduling problems. **Mathematical Programming**, v. 54, n. 1-3, p. 353–367, 1992. 3.1

VAN DEN AKKER, J.; HURKENS, C. A.; SAVELSBERGH, M. W. Time-indexed formulations for machine scheduling problems: Column generation. **INFORMS Journal on Computing**, v. 12, n. 2, p. 111–124, 2000. 3.1

VAN DEN AKKER, J.; VAN HOESEL, C.; SAVELSBERGH, M. W. A polyhedral approach to single-machine scheduling problems. **Mathematical Programming**, v. 85, n. 3, p. 541–572, 1999. 3.1

A
Formulação *Bucket-Indexed* – Modelo proposto completo

A formulação *bucket-indexed* proposta nesta dissertação, extensão do modelo de Boland et al. (2016), é um modelo de programação linear inteira mista para tratar o problema de programação de máquinas paralelas.

A função objetivo e as restrições a seguir definem esse modelo, considerando as hipóteses apresentadas no Capítulo 4, a fim de garantir a viabilidade da programação e cumprimento das restrições do problema descritas no Capítulo 2.

$$\min \sum_{m\in M_j}\sum_{j\in J}\sum_{b\in[1,B]}\sum_{k\in K}((b+P_j+k-2)z_{m,j,b,k}+v_{m,j,b+P_j+k-1,k})\Delta \quad (4\text{-}6)$$

$$u_{m,j,b,k}+v_{m,j,b+P_j+k-1,k} = (2-k-\pi_j)z_{m,j,b,k} \;\; \forall m \in M_j,\; \forall j \in J,\; \forall b \in [1,B],\; \forall k \in K \quad (4\text{-}1)$$

$$\left((1-k)(1-\pi_j)+\frac{1}{\Delta}\right) z_{m,j,b,k} \leq u_{m,j,b,k} \;\; \forall m \in M_j,\; \forall j \in J,\; \forall b \in [1,B],\; \forall k \in K \quad (4\text{-}2)$$

$$u_{m,j,b,k} \leq (1-k\pi_j)z_{m,j,b,k} \forall m \in M_j,\; \forall j \in J,\; \forall b \in [1,B],\; \forall k \in K \quad (4\text{-}3)$$

$$\sum_{j\in J}\sum_{k\in K}\sum_{\hat{b}\in[b-P_j+2,b]} z_{m,j,\hat{b},k} \leq 1 \;\; \forall m \in M_j,\; \forall b \in [1,B] \quad (4\text{-}4)$$

$$\sum_{j\in J}\sum_{k\in K}\left(u_{m,j,b,k}+v_{m,j,b,k}+\sum_{\hat{b}\in[b-P_j+2,b-1]} z_{m,j,\hat{b},k}\right) \leq 1 \;\; \forall m \in M_j,\; \forall b \in [1,B] \quad (4\text{-}5)$$

$$\sum_{m \in M_j} \sum_{b \in [1,B]} \sum_{k \in K} z_{m,j,b,k} = 1 \quad \forall j \in J \tag{4-7}$$

$$inicio_j = \sum_{min M_{\hat{j}}} \sum_{b \in [1,B]} \sum_{k \in K} ((b-1) z_{m,\hat{j},b,k} - u_{m,\hat{j},b,k}) \Delta + \Delta + 1 \tag{4-11}$$

$$inicio_j - inicio_{\hat{j}} \geq dif_{j,\hat{j}} \forall (j,\hat{j}) \in J \mid prec_{j,\hat{j}} = 1 \tag{4-12}$$

$$u_{m,j,R_j,0} \leq \rho_j z_{m,j,R_j,0} \quad \forall m, \ \forall j \mid 1 - \pi_j < \rho_j \leq 1 \tag{4-14}$$

$$u_{m,j,R_j,1} \leq \rho_j z_{m,j,R_j,1} \quad \forall m, \ \forall j \mid \rho_j \leq 1 - \pi_j \tag{4-15}$$

$$\bar{\delta_j} z_{m,j,\bar{D_j},0} \leq u_{m,j,\bar{D_j},0} \quad \forall m, \ \forall j \mid 1 - \pi_j < \bar{\delta_j} \leq 1 \tag{4-16}$$

$$\bar{\delta_j} z_{m,j,\bar{D_j},1} \leq u_{m,j,\bar{D_j},1} \quad \forall m, \ \forall j \mid \bar{\delta_j} \leq 1 - \pi_j \tag{4-17}$$

$$\sum_{m \in M_j} \sum_{a \in A_m} w_{a,j} = 1 \ \forall j \in J \tag{4-18}$$

$$\sum_{b \in [1,B]} \sum_{k \in K} z_{m,j,b,k} = \sum_{a \in A_m} w_{a,j} \ \forall m \in M_j \ \forall j \in J \tag{4-19}$$

$$u_{m,j,S_a,0} \leq \theta_a z_{m,j,S_a,0} \quad \forall m, \forall j, \forall a | a \in A_m \wedge 1 - \pi_j < \theta_a \leq 1 \tag{4-21}$$

$$u_{m,j,S_a,1} \leq \theta_a z_{m,j,S_a,1} \quad \forall m, \forall j, \forall a | a \in A_m \wedge \theta_a \leq 1 - \pi_j \tag{4-22}$$

$$\phi_{a,j} z_{m,j,F_{a,j},0} \leq u_{m,j,F_{a,j},0} \quad \forall m, \forall j, \forall a | a \in A_m \wedge 1 - \pi_j < \phi_{a,j} \leq 1 \quad (4\text{-}23)$$

$$\phi_{a,j} z_{m,j,F_{a,j},1} \leq u_{m,j,F_{a,j},1} \quad \forall m, \forall j, \forall a | a \in A_m \wedge \phi_{a,j} \leq 1 - \pi_j \quad (4\text{-}24)$$

$$inicio_j \in \mathbb{Z}^+ \; \forall j \in J \quad (4\text{-}13)$$

$$w_{a,j} \in \{0,1\} \; \forall m \in M_j, \; \forall a \in A_m, \; \forall j \in J \quad (4\text{-}20)$$

$$z_{m,j,b,k} \in \{0,1\} \; \forall m \in M_j, \; \forall j \in J, \; \forall b \in [1,B], \; \forall k \in K \mid existeZeU_{m,j,b,k,a} = 1 \quad (4\text{-}25)$$

$$u_{m,j,b,k} \in [0,1] \; \forall m \in M_j, \; \forall j \in J, \; \forall b \in [1,B], \; \forall k \in K \mid existeZeU_{m,j,b,k,a} = 1 \quad (4\text{-}26)$$

$$v_{m,j,b,k} \in [0,1] \; \forall m \in M_j, \; \forall j \in J, \; \forall b \in [1,B], \; \forall k \in K \mid existeV_{m,j,b,k,a} = 1 \quad (4\text{-}27)$$

A função objetivo (4-6) visa minimizar a soma dos tempos de conclusão das tarefas. As equações (4-1) são responsáveis por fazer a relação entre as variáveis do modelo e permitir a omissão da variável $v_{m,j,b,k}$. As restrições (4-2) e (4-3) estabelecem, respectivamente, esses limites inferiores e superiores das variáveis considerando os domínios válidos quando $k = 0$ e $k = 1$. As restrições (4-4) indicam que no máximo 1 tarefa pode iniciar no *bucket* b e na máquina m sem que haja sobreposição. As restrições (4-5) garantem que as tarefas executadas na máquina m no *bucket* b não sejam sobrepostas. As restrições (4-7) indicam que cada tarefa j inicia exatamente uma vez e, portanto, é executada sem preempção em uma máquina. As restrições (4-11) calculam o

início das tarefas apenas para simplificar a leitura das restrições (4-12). As restrições (4-12) indicam que a precedência entre as tarefas $j, \hat{j}$ é respeitada, considerando pelo menos a distância mínima $dif_{j,\hat{j}}$ entre seus inícios. As restrições (4-14) e (4-15) indicam limite superior as variáveis $u_{m,j,R_j,k}$ para $k = 0$ e $k = 1$ garantindo o cumprimento das datas de liberação. As restrições (4-16) e (4-17) indicam o limite inferior as variáveis $u_{m,j,\bar{D}_j,k}$ para $k = 0$ e $k = 1$ garantindo o cumprimento das datas limite. As restrições (4-18) indicam que a tarefa j inicia na disponibilidade a. As restrições (4-19) relacionam as variáveis $z_{m,j,b,k}$ e $w_{a,j}$ garantindo que apenas se a tarefa j iniciar na m, ela irá ocupar a disponibilidade $a \in A_m$. Analogamente, as restrições (4-21)-(4-24) garantem que as tarefas serão executadas apenas dentro das disponibilidades. As restrições (4-13) e (4-20) impõem a integralidade das variáveis $inicio_j$ e $w_{a,j}$, respectivamente. As restrições (4-25) - (4-27) impõem o domínio das variáveis binárias e contínuas e asseguram que só serão geradas caso os parâmetros $existeZeU_{m,j,b,k,a} = 1$ e $existeV_{m,j,b,k,a} = 1$, isto é, respeitando as datas de liberação, as datas limites, a elegibilidade e a disponibilidade das máquinas.