UNIVERSIDADE FEDERAL DE GOIÁS
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA DE PRODUÇÃO

MODELOS DE PROGRAMAÇÃO LINEAR INTEIRA PARA VARIANTES DO PROBLEMA DE PROGRAMAÇÃO DE PROJETOS COM RESTRIÇÃO DE RECURSOS

LUCIANA VIEIRA DE MELO

Catalão - GO
2018

MODELOS DE PROGRAMAÇÃO LINEAR INTEIRA PARA VARIANTES DO PROBLEMA DE PROGRAMAÇÃO DE PROJETOS COM RESTRIÇÃO DE RECURSOS

LUCIANA VIEIRA DE MELO

Dissertação apresentada ao Programa de Pós-graduação em Engenharia de Produção da Universidade Federal de Goiás, como parte dos requisitos para obtenção do título de **MESTRE EM ENGENHARIA DE PRODUÇÃO.**

Área de Concentração: Engenharia de Operações e Processos Industriais.

Orientador: Prof. Dr. Thiago Alves de Queiroz

Catalão – GO

2018

Ficha de identificação da obra elaborada pelo autor, através do
Programa de Geração Automática do Sistema de Bibliotecas da UFG.

Melo, Luciana Vieira de
Modelos de Programação Linear Inteira para Variantes do Problema de Programação de Projetos com Restrição de Recursos [manuscrito] / Luciana Vieira de Melo. - 2018.
lxxiv, 80 f.: il.

Orientador: Prof. Dr. Thiago Alves de Queiroz.
Dissertação (Mestrado) - Universidade Federal de Goiás, Programa de Pós-Graduação em Engenharia de Produção, Catalão, 2018.
Bibliografia.
Inclui gráfico, tabelas, lista de figuras, lista de tabelas.

1. Problema de Programação de Projetos com Restrição de Recursos. 2. Makespan. 3. Múltiplas habilidades. 4. Múltiplos modos. 5. Tempos de atraso. 3. Programação Linear Inteira.
I. Queiroz, Dr. Thiago Alves de, orient. II. Título.

CDU 658.5

Aos meus pais, Waldir e Maria do Carmo,

ao meu irmão André e ao meu noivo Gustavo,

pelo incentivo e apoio.

AGRADECIMENTOS

Agradeço a todos que acreditaram em meu potencial e que contribuíram no desenvolvimento desta pesquisa. Em especial, agradeço:

A Deus, por estar sempre iluminando meus passos, me dando sabedoria e perseverança para não desistir dos meus sonhos;

Aos meus pais, Waldir Vieira da Silva e Maria do Carmo de Melo Silva, e ao meu irmão, André Vieira de Melo, por estarem sempre ao meu lado, me incentivando e apoiando em todos os momentos;

Ao meu noivo, Gustavo Henrique Correia Mariano, por todo amor, dedicação, companheirismo, paciência e, principalmente, por entender minha ausência em alguns momentos;

Ao meu orientador, professor Thiago Alves de Queiroz, pela atenção, confiança, dedicação, paciência, pelos ensinamentos e, principalmente, pela orientação decisiva para tornar realidade este trabalho;

A todos os professores que contribuíram com minha formação e a todos meus amigos pela amizade e companheirismo.

“O começo de todas as ciências é o espanto de as coisas serem o que são. ”

Aristóteles

MELO, L. V. **Modelos de Programação Linear Inteira para Variantes do Problema de Programação de Projetos com Restrição de Recursos.** 80p. Dissertação de Mestrado, Universidade Federal de Goiás, Catalão, GO. 2018.

RESUMO

O problema de programação de projetos com restrição de recursos tem sua importância tanto na parte teórica, como no domínio da pesquisa operacional, quanto na prática, com o gerenciamento de projetos nos ambientes corporativos e outras aplicações. Neste contexto, foram estudados modelos de programação linear inteira (mista), resolvidos com a ajuda de uma biblioteca de otimização, para o problema de programação de projetos com restrição de recursos. O problema busca pela minimização do *makespan*, isto é, o tempo de conclusão total do projeto, dado o escalonamento de atividades. Para alcançar o objetivo do presente trabalho, utiliza-se uma abordagem quantitativa e a pesquisa é classificada como descritiva quanto ao objetivo, e bibliográfica e experimental quanto aos procedimentos técnicos utilizados. O primeiro modelo considera dois tipos de variáveis de decisão, enquanto no segundo modelo há apenas um tipo de variável. Ao considerar a inserção de restrições reais, em particular, da restrição de múltiplas habilidades, múltiplos modos e de tempos de atraso, obtém-se, respectivamente, o terceiro, quarto e quinto modelos a partir do segundo, com a respectiva adição dessas restrições. Busca-se analisar os modelos com relação ao tempo de otimização e a quantidade de instâncias resolvidas de forma ótima. Os resultados dos experimentos computacionais indicam que o segundo modelo é um pouco mais competitivo do que o primeiro, pois conseguiu resolver um maior número de instâncias, apresentar soluções com um menor *gap* e requerer menos tempo computacional. Por isso é que os demais modelos partiram do segundo para a adição de restrições práticas. Os resultados dos experimentos computacionais indicam que os modelos desenvolvidos com restrições práticas tiveram um desempenho melhor (isto é, quanto ao número de instâncias resolvidas, valor do *gap* e do tempo computacional) para as instâncias menores. Portanto, desenvolver modelos que sejam capazes de resolver instâncias de médio e grande porte torna-se um desafio, porém, pode trazer grandes vantagens para o ambiente corporativo, auxiliando a tomada de decisão pelos gestores, reduzindo desperdícios, melhorando custos e, assim, trazendo bem-estar pessoal.

Palavras-chave: problema de programação de projetos com restrição de recursos, *makespan*, múltiplas habilidades, múltiplos modos, tempos de atraso, programação linear inteira.

MELO, L. V. **Integer Linear Programming Models for Variants of the Resource Constrained Project Scheduling Problem.** 80p. Masters Dissertation. Federal University of Goiás, Catalão, GO. 2018.

ABSTRACT

The resource-constrained project scheduling problem has its importance both in the theoretical part, as in the field of operational research, and in practice, with project management in corporate environments and other applications. In this context, some integer linear programming models, solved with the help of an optimization library, were studied for the resource-constrained project scheduling problem. This problem aims at minimizing the makespan, namely, the total completion time of the project, given the scheduling of activities. In order to achieve this objective, a quantitative approach is used, and the research is classified as descriptive with regard to its objective, and bibliographical and experimental with regard to the technical procedures used. The first model has two types of decision variables, while in the second model there is only one type of variable. When considering the insertion of real constraints, in particular, the multi-skill, the multi-mode and time lags, the third, fourth and fifth models are obtained, respectively, from the second model with the addition of such constraints. The models are analyzed with regard to the runtime and the amount of instances solved in optimality. The results of the computational experiments indicate that the second model is a bit more competitive in comparison with the first one, since it was able to solve a larger number of instances, present solutions with a smaller gap and require less computational time. Therefore, the other models started from the second with the addition of practical constraints. The results of the computational experiments indicate that the models with practical constraints can have better performance (that is, related to the number of instances solved, gap value and computational time) when smaller instances are considered. Therefore, developing models that are capable of solving medium and large size instances is a challenge, but, it can bring great advantages for the corporate environment, helping managers in making decisions, reducing waste, improving costs and thus bringing personal well-being.

Keywords: resource constrained project scheduling problem, makespan, multi-skill, multi-mode, time lag, integer linear programming.

LISTA DE FIGURAS

Figura 3.1 – Solução viável para a instância da Tabela 3.1 38

Figura 3.2 – Diferença entre uma malha inteira e outra com os *canonical dissections* 42

Figura 4.1 – Quantidade de instâncias resolvidas na otimalidade para o conjunto J30 56

Figura 4.2 – Tempo médio de otimização (em segundos) para o conjunto J30 56

Figura 4.3 – Valor do gap médio (em porcentagem) para o conjunto J30 56

Figura 4.4 – Quantidade de instâncias resolvidas na otimalidade para o conjunto J60 59

Figura 4.5 – Tempo médio de otimização (em segundos) para o conjunto J60 59

Figura 4.6 – Valor do gap médio (em porcentagem) para o conjunto J60 60

LISTA DE QUADROS

Quadro 2.1 – Restrições práticas para o RCPSP 27

Quadro 2.2 – Trabalhos que resolveram o RCPSP por algum método exato 28

Quadro 2.3 – Trabalhos que resolveram o RCPSP por algum método heurístico 32

LISTA DE TABELAS

Tabela 3.1 – Exemplo de uma instância para o RCPSP 37

Tabela 4.1 – Resultados para o conjunto J30 obtidos ao resolver os Modelos 1 e 2 54

Tabela 4.2 – Resultados para o conjunto J60 obtidos ao resolver os Modelos 1 e 2 57

Tabela 4.3 – Resultados das instâncias do RCPSP com múltiplas habilidades 61

Tabela 4.4 – Resultados para os conjuntos MMLIB50 e MMLIB100 obtidos ao resolver o Modelo 4 64

Tabela 4.5 – Resultados para o conjunto L30 e L60 69

SUMÁRIO

CAPÍTULO 1

INTRODUÇÃO A PROGRAMAÇÃO DE PROJETOS 15

1.1 Questão, hipótese e objetivos do trabalho 17

1.2 Justificativa 18

1.3 Metodologia 20

1.4 Estrutura do trabalho 21

CAPÍTULO 2

REVISÃO BIBLIOGRÁFICA 22

2.1 Otimização combinatória 22

2.1.1 Método branch-and-bound 23

2.1.2 Método branch-and-cut 23

2.2 Problemas de programação ou *scheduling* 24

2.3 Problemas de programação de projetos 25

2.4 Restrições práticas 26

2.5 Métodos exatos para o RCPSP 27

2.6 Métodos heurísticos para o RCPSP 31

CAPÍTULO 3

MODELOS MATEMÁTICOS PARA O PROBLEMA DE PROGRAMAÇÃO DE PROJETOS COM RESTRIÇÃO DE RECURSOS 36

3.1 Definição do RCPSP 36

3.2 Modelos para o RCPSP 38

3.3 Limitantes para o RCPSP 41

3.4 Variantes do RCPSP 45

3.4.1 Múltiplas habilidades 46

3.4.2 Múltiplos modos 48

3.4.3 Tempos de atraso: mínimo e máximo 49

CAPÍTULO 4

RESULTADOS COMPUTACIONAIS 52

4.1 Resultados para o RCPSP 52

4.2 Resultados para o RCPSP com múltiplas habilidades 60

4.3 Resultados para o RCPSP com múltiplos modos 63

4.4 Resultados para o RCPSP com tempos de atraso 68

CAPÍTULO 5

CONCLUSÕES E TRABALHOS FUTUROS 72

REFERÊNCIAS 75

CAPÍTULO 1

INTRODUÇÃO A PROGRAMAÇÃO DE PROJETOS

A gestão de projetos teve sua importância reconhecida a partir das mudanças macroeconômicas, devido à necessidade das empresas de se manterem competitivas e mais produtivas. Dessa forma, a execução de projetos representa progresso e uma economia mais eficiente. Geralmente, projetos a nível nacional são grandiosos e dispendiosos, por isso um gerenciamento eficiente e eficaz é fundamental para alcançar sucesso, expandir a receita, reduzir custos e possíveis lucros perdidos (HABIBI; BARZINPOUR; SADJADI, 2018).

Ainda de acordo com Habibi, Barzinpour e Sadjadi (2018), a gestão de projetos possui relevância tanto no aspecto teórico quanto no prático. Na visão teórica, a programação de projetos é muito discutida na área de otimização, por esse motivo tem provocado o interesse de muitos pesquisadores no âmbito da pesquisa operacional. Na visão prática, tem-se que a otimização da programação de um determinado projeto, que está inserida no processo de gerenciamento de projetos, é capaz de retornar de maneira satisfatória a conclusão do projeto e a minimização de custos.

De acordo com Almeida, Correia e Saldanha-da-Gama (2016), a programação de projetos refere-se a um assunto significativo na gestão de projetos e compreende em definir um cronograma para as atividades que compõem o projeto, a fim de otimizar uma medida de desempenho preestabelecida, respeitando as restrições impostas, tais como a precedência e o uso de recursos. Neste sentido, a programação de projetos tem sido evidenciada e muito praticada, pelo motivo de diversas empresas atuarem com amplos e complexos projetos.

A programação de atividades direcionadas a gestão de projetos torna-se fundamental para alcançar o sucesso em uma empresa ou qualquer tipo de negócio. Um

problema qualquer pode ser compreendido por meio de um estudo detalhado das relações e particularidades das atividades de um projeto. Além disso, a programação permite identificar, compreender e investigar alternativas de tornar o processo produtivo mais eficiente. Portanto, as empresas buscam concluir os projetos em um menor tempo possível e com a máxima utilização dos recursos disponíveis (SILVA; VIEIRA; SILVA, 2017).

Problemas dentro da gestão de projetos podem ser modelados como Problemas de Otimização Combinatória, que estão presentes em atividades do nosso cotidiano, como em serviços de transporte público, controle de tráfego aéreo, serviços de entrega de uma forma geral, dentre outros. Grande parte dos problemas de otimização combinatória podem ser modelados usando uma formulação matemática, com o objetivo de minimizar ou maximizar, variáveis e um conjunto de restrições provenientes do problema. A busca por soluções ótimas desses modelos torna-se um dos grandes desafios da área de otimização (MAURI, 2008).

Os problemas de otimização combinatória podem ser resolvidos via métodos exatos e heurísticas. Os métodos exatos obtêm uma solução ótima, por meio de algoritmos que testam explicitamente (ou implicitamente) todas as possíveis soluções, enquanto nos métodos heurísticos, uma solução é obtida sem geralmente saber da qualidade desta em comparação com a solução ótima. Problemas de otimização combinatória são geralmente classificados como NP-difíceis (GAREY; JOHNSON, 1979), fazendo com que aplicação de métodos exatos (ou a resoluções de modelos matemáticos por esses métodos) seja limitada a instâncias relativamente pequenas, por isso, as heurísticas têm sido muito utilizadas nos casos reais (ARENALES et al., 2007).

O problema de interesse deste estudo, que está dentro da gestão de projetos, é o Problema de Programação de Projetos com Restrição de Recursos (*Resource Constrained Project Scheduling Problem* – RCPSP), que considera um conjunto de atividades, um conjunto de tipos de recursos e um conjunto composto de pares ordenados para estabelecer a relação de precedência entre as atividades (GAFAROV; LAZAREV; WERNER, 2010). Cada atividade possui um tempo estabelecido de execução e consome certa quantidade de unidades de cada recurso, considerando que os recursos são limitados. A intenção do RCPSP é determinar os tempos de início de cada atividade respeitando as relações de precedência e a disponibilidade máxima de cada recurso, de forma que o *makespan*, isto é, a duração total do projeto, seja mínimo (MENDES; GONÇALVES, 2003).

Pinedo (2012) diz que o RCPSP se faz presente principalmente em indústrias de manufatura e de serviços, portanto, as decisões tomadas a respeito ocorrem à nível operacional. Dessa forma, as pistas de um aeroporto, os ônibus no transporte público, as limitações de máquinas em um fábrica, além de outros exemplos, podem representar os recursos. Nesse sentido, as decolagens e aterrissagens em um aeroporto, o deslocamento

de passageiros pelo transporte público e as operações em uma linha de produção representam as atividades.

Na literatura é possível encontrar muitos trabalhos que buscam resolver o RCPSP aplicado a situações reais, como é o caso de Singh (2014), em que um estudo de caso foi desenvolvido em uma empresa especializada em projetos. Em Oztemel e Selam (2017) há um algoritmo baseado no comportamento de abelhas proposto para o RCPSP com aplicação em indústrias de moldagem por injeção. Silva, Vieira e Silva (2017) apresentaram um estudo de caso em uma planta de análise química de minério de ferro.

Portanto, encontrar uma solução ótima em um tempo computacional satisfatório para o RCPSP, que tem aplicação real, é um desafio de pesquisadores que têm utilizado métodos exatos para resolvê-lo (PINEDO, 2012). Outra consideração também bastante importante dentro do RCPSP é a inclusão de restrições que modelam situações reais (isto é, restrições práticas), como o caso de múltiplos modos, diferentes habilidades, janelas de tempo e tempo de atraso entre atividades, projetos não renováveis, atividades que podem ser interrompidas e depois continuadas, tempo de preparo diferente entre a execução de uma atividade e outra, ambiente de trabalho com máquinas diferentes, atividades que dependem de certos recursos para serem processadas, entre outros (HARTMANN; BRISKORN, 2010).

1.1 Questão, hipótese e objetivos do trabalho

Nesta pesquisa, busca-se responder a seguinte questão: é possível obter um modelo de programação linear inteira que consiga resolver instâncias com número relativamente médio ou grande de atividades, para o problema de programação de projetos com restrição de recursos na presença de restrições reais? Sobre essa questão, levanta-se a hipótese de que é possível criar modelos de programação linear inteira para expressar o problema e as suas restrições, utilizando um conjunto de variáveis, uma função objetivo e inequações/equações. Com isso, os modelos podem ser competitivos (desempenho satisfatório ou melhor no caso de comparação) quanto as seguintes medidas de desempenho: resolver um maior número de instâncias, obter soluções com um *gap* menor e demandar menos tempo computacional. No caso de modelos de programação linear inteira (mista), considera-se a aplicação de um algoritmo exato (isto é, *branch-and-cut*), disponível em pacotes de otimização, para resolvê-los e, assim, obter a solução ótima do problema.

Essa questão foi levantada com base nos autores Hartmann e Briskorn (2010), que afirmaram que realizar a programação de projetos exige um trabalho importante e árduo diante da presença de relações de precedência entre atividades e de recursos limitados.

Com isso, o uso de um modelo matemático permite representar o problema real que busca pela programação de atividades sujeitas a restrições de precedência e de recursos, com o objetivo de minimizar o *makespan*. Além disso, o RCPSP é um problema comum na gestão de projetos, com diversas aplicações práticas e pode ter uma solução obtida por métodos exatos e heurísticos.

Diante do exposto, este trabalho tem como objetivo estudar duas propostas distintas de modelo de programação linear inteira (mista) para o RCPSP, com o propósito de minimizar o *makespan*. Os modelos são fundamentados em formulações de dois índices, indexadas no tempo, cuja diferença entre eles está na quantidade de variáveis utilizadas. O primeiro modelo assume dois tipos de variáveis, uma para definir o tempo de início de cada atividade e outra para definir o *makespan* do projeto. No segundo modelo, a variável para definir o *makespan* é substituída e um novo conjunto de restrições é adotado para modelar as restrições de precedência. Para melhorar a resolução dos dois modelos, apresentam-se métodos para obter um limitante inferior (*Lower Bound*) e um limitante superior (*Upper Bound*) do *makespan*, com isso, reduzindo o espaço de busca durante a resolução dos modelos. A partir dos dois modelos iniciais, identifica-se aquele que apresenta melhor desempenho em termos de tempo de otimização e número de soluções ótimas encontradas, para que novas restrições reais possam ser adicionadas, como a de recursos que possuem certas habilidades e, assim, não podem ser usados por algumas atividades (essa restrição é conhecida na literatura como *multi-skill)*, bem como a de múltiplos modos (*multi-mode*) e de tempos de atraso mínimo e máximo (*minimal and maximal time lags*).

Para alcançar o objetivo geral, apontam-se os seguintes objetivos específicos:

i) estudar modelos de programação linear inteira para o RCPSP e variantes com restrições reais;
ii) codificar os modelos usando alguma linguagem de programação e uma biblioteca de otimização;
iii) comparar os modelos quanto ao tempo de otimização, quantidade de instâncias resolvidas de forma ótima e *gap* da solução.

1.2 Justificativa

De acordo com Silva (2012), dois elementos são de grande relevância para o RCPSP, quais sejam: as atividades, que fazem parte das etapas de execução do projeto, e os recursos, que são fundamentais para a execução de uma atividade, pois cada uma delas consome certa quantidade dos recursos disponíveis. As atividades, entre si, possuem relações de precedência, que vão definir a sequência das mesmas. Essas relações são

geralmente denominadas *finish-to-start,* ou seja, é necessário que as atividades predecessoras sejam finalizadas para que depois uma sucessora comece a ser processada. Em geral, o problema requer que todas as atividades do projeto sejam executadas em um menor tempo possível, respeitando as relações de precedência e a disponibilidade de recursos. Dessa forma, um dos elementos que precisa ser observado está relacionado com os recursos utilizados para executar as atividades, podendo ser classificados em recursos renováveis, no qual podem ser reutilizados, e em não-renováveis.

Um dos motivos para a realização desta pesquisa está no grande número de aplicações do RCPSP, em especial, na execução de atividades por uma ou mais máquinas, para descobrir qual a melhor sequência das mesmas, de forma a minimizar tempo ou outro fator desejado. Aplicações práticas surgem no gerenciamento de projetos nas empresas, em obras da engenharia, no desenvolvimento de mecanismos para microprocessadores para máquinas, entre outras (ÖZDAMAR; ULUSOY, 1995). Quando este problema é bem resolvido, ou seja, há métodos capazes de fornecer boas soluções em tempo hábil, tem-se um melhor auxílio para a tomada de decisões rápidas e precisas dentro das empresas (HARTMANN; BRISKORN, 2010). Considera-se, também, que a inclusão de restrições reais no problema faz com que as soluções obtidas pelos modelos possam de fato representar a realidade e, assim, serem consideradas pelos tomadores de decisão.

No contexto de trabalhos sobre o RCPSP, a literatura é vasta, mostrando que esse problema tem sua importância a nível científico e tecnológico, principalmente por ser um problema NP-Difícil e que surge em diferentes contextos reais. Em Özdamar e Ulusoy (1995) uma pesquisa sobre os trabalhos que resolveram o RCPSP foi realizada, com diferentes tipos de restrições e representações de restrições presentes em problemas do mundo real. Em Ke e Liu (2010) foi desenvolvido um modelo para o RCPSP com parâmetros incertos, com a finalidade de definir o cronograma de alocação de recursos para balancear o tempo de conclusão das atividades e o custo total do projeto. Um algoritmo inteligente híbrido que integra a simulação *fuzzy* com o algoritmo genético foi elaborado. Em Singh (2014) foi desenvolvido um algoritmo híbrido com restrições de prioridade para o RCPSP.

Com relação aos métodos exatos, Naber e Kolisch (2014) e Baumann, Fündeling e Trautmann (2015) desenvolveram modelos de programação linear inteira, que são solucionados por meio de algoritmos exatos enumerativos. A partir disso, busca-se contribuir com a literatura estudando modelos de programação linear inteira para o RCPSP, com a resolução do modelo ocorrendo por meio de métodos exatos, como o *branch-and-cut* disponível em bibliotecas de otimização. Além de estudar os modelos para o RCPSP, considera-se a extensão deles modelos para a inclusão de restrições práticas definidas sobre o problema.

1.3 Metodologia

A metodologia de pesquisa foi definida a partir do objetivo geral e dos objetivos específicos que se pretendem atingir. A pesquisa refere-se a um método racional e sistemático a fim de possibilitar solução aos problemas em estudo. O desenvolvimento desta ocorre por meio das informações disponíveis e da aplicação minuciosa de técnicas, métodos e outros mecanismos científicos, o que vai desde a formulação do problema até a divulgação dos resultados (GIL, 2009).

As pesquisas podem ser classificadas em diferentes tipos, de acordo com alguns critérios. De acordo com Gil (2009), as pesquisas podem ser classificadas quanto aos seus objetivos, como sendo exploratória, descritiva e explicativa. Com relação aos procedimentos técnicos aplicados, uma pesquisa pode ser bibliográfica, documental, experimental, levantamento de dados, estudo de campo, estudo de caso e pesquisa-ação. Segundo Miguel (2012), quanto à abordagem da pesquisa, ela pode ser quantitativa e qualitativa.

Diante dessas possíveis classificações, esta pesquisa é classificada da seguinte forma:

- Com base nos objetivos: a pesquisa é considerada como descritiva, pois busca por estabelecer relações entre as variáveis do problema (GIL, 2009).
- Com relação aos procedimentos técnicos: a pesquisa é bibliográfica, por partir de outras pesquisas já realizadas, como artigos, teses, livros, dentre outros (SEVERINO, 2007). Além disso, é também experimental, uma vez que experimentos computacionais vão ser realizados para alcançar os objetivos do trabalho. A pesquisa experimental, de acordo com Bryman (1989), é muito utilizada em abordagens quantitativas, geralmente, envolvendo modelos matemáticos, simulação computacional e experimentos laboratoriais.
- Quanto à abordagem: a pesquisa é quantitativa, pois, segundo Bryman (1989), a principal preocupação da abordagem quantitativa é a ação de mensurar variáveis, sendo esta a forma de justificar o emprego dos modelos desenvolvidos. Além dessa, pode-se citar a causalidade que busca demonstrar como os eventos são, a possível generalização de resultados para que sejam aplicados além dos limites de uma investigação específica e a replicação, que representa a possibilidade de um pesquisador replicar os mesmos procedimentos de outro para certificar a validade preliminar.

Um modelo matemático pode representar um sistema qualquer, objetivando analisar as operações desse sistema e, então, checar o desempenho do sistema ou até mesmo otimizar a sua estrutura. O RCPSP foi resolvido por meio de modelos matemáticos cujas funções são todas lineares e as (algumas) variáveis assumem valores inteiros, tendo a sua

resolução por um método exato enumerativo do tipo *branch-and-cut* (ARENALES et al., 2007), disponível na biblioteca *Gurobi Optimizer*.

Assim, conforme os objetivos mencionados anteriormente e com o intuito de alcançá-los, executaram-se as seguintes atividades:

- Fazer uma revisão da literatura, para entender o problema e suas restrições, e estudar artigos que resolveram o RCPSP e suas variantes com restrições reais;
- Investigar, a partir do estudo da literatura, modelos de programação linear inteira para o RCPSP e algumas de suas variantes;
- Codificar os modelos usando uma linguagem de programação de alto nível e utilizar o algoritmo *branch-and-cut* da biblioteca *Gurobi Optimizer* para a resolução dos modelos considerados;
- Validar e testar os modelos em instâncias disponíveis na literatura;
- Analisar os modelos com relação ao tempo de otimização, ao número de instâncias resolvidas de forma ótima e o *gap* da solução final;

1.4 Estrutura do trabalho

Este trabalho está organizado em cinco capítulos. O primeiro capítulo apresentou brevemente sobre problemas de programação de projetos, com destaque para o problema sendo estudado, e otimização combinatória. Ainda neste capítulo foram abordados os objetivos, as justificativas e a metodologia para a realização da pesquisa. Os demais capítulos são:

- Capítulo 2: neste capítulo são apresentados conceitos e aplicações sobre os problemas de otimização combinatória e problemas de programação de projetos, bem como um levantamento de trabalhos que utilizaram métodos exatos e heurísticos para resolver o RCPSP e algumas de suas variantes.
- Capítulo 3: contempla primeiramente a definição formal do problema contendo os parâmetros que foram adotados, as restrições que devem ser obedecidas e o objetivo do problema, para em seguida, apresentar e discutir os modelos estudados.
- Capítulo 4: são apresentados os experimentos computacionais realizados, detalhando sobre como foi feita a implementação dos modelos, características das instâncias utilizadas para testá-los, bem como uma discussão dos resultados obtidos com cada modelo e uma comparação entre eles.
- Capítulo 5: apresenta as conclusões deste trabalho, a partir da análise feita sobre os modelos e com a inclusão das restrições práticas, e sugestões de pesquisas futuras.

CAPÍTULO 2

REVISÃO BIBLIOGRÁFICA

Este capítulo contém uma breve descrição sobre Otimização Combinatória e os tipos de problemas envolvidos. Em seguida, discute-se sobre Problemas de Programação, bem como Problemas de Programação de Projetos com as restrições mais comumente adotadas. Além disso, apresenta-se uma revisão de trabalhos que propuseram métodos exatos e heurísticos para o Problema de Programação de Projetos com Restrição de Recursos, que é o problema de interesse deste trabalho.

2.1 Otimização combinatória

Arroyo (2002) afirmou que a Otimização Combinatória se refere a um princípio de tomada de decisões para resolver problemas com um número finito de soluções, porém muito grande, e pode ser identificada em problemas de corte e empacotamento, problemas de planejamento e programação (*scheduling*) da produção, redes de telecomunicação, problemas de localização, roteamento de veículos, dentre outros.

Na área de engenharia de produção é muito comum aplicações modeladas como problemas de otimização combinatória, tais como: planejamento e programação da produção, projeto de leiaute do processo produtivo, local das instalações e do centro de distribuição de produtos, entre outros. Portanto, dimensionar a fabricação de um produto em uma determinada época, determinar a sequência mais adequada para o processamento de produtos em uma máquina, estabelecer um local mais apropriado e estratégico para instalar uma fábrica ou um centro de distribuição, bem como rotas planejadas para entregar

produtos, compreendem aplicações que podem ser resolvidas como problemas de otimização combinatória (ARENALES et al., 2007).

Nos problemas de otimização combinatória as soluções costumam ser representadas por números inteiros. O intuito é descobrir valores em um conjunto contavelmente finito (ou infinito), que correspondam a um ótimo global (MARTINS, 2010). O autor ainda disse que o Problema de Programação Linear Inteira (PPLI) é definido como um problema de otimização combinatória. Os métodos exatos (por exemplo, *branch-and-bound* e *branch-and-cut*) e heurísticos (construtivas, busca local, meta-heurísticas) costumam ser utilizados na resolução de PPLI, sendo que os primeiros têm como objetivo alcançar uma solução ótima para o problema, a partir de uma enumeração das possíveis soluções, e os segundos têm como objetivo encontrar boas soluções ou soluções aproximadas, porém não sendo necessariamente a ótima.

2.1.1 Método branch-and-bound

O método *branch-and-bound* (*B&B*) é representado por meio de uma árvore, no qual cada nó consiste em um problema de programação linear, sendo empregado para obter a solução ótima de PPLIs. O *B&B* é um método enumerativo que busca as soluções sobre a árvore por meio de ramificação e poda de nós (ARENALES et al. 2007).

De acordo com Goldbarg e Luna (2005), a palavra *branch* está relacionada com a divisão dentro do espaço de soluções que o método faz e a palavra *bound* está associada ao uso de limitantes estimados no decorrer da enumeração para podar nós cujas soluções não são atrativas (isto é, não são melhores do que a melhor solução conhecida). Portanto, o método *B&B* consiste em desenvolver uma enumeração dos possíveis nós que podem fornecer a solução ótima inteira do problema.

2.1.2 Método branch-and-cut

Wolsey (1998) afirmou que o método *branch-and-cut* considera o *B&B* com a introdução de planos de corte para agilizar a poda de nós da árvore de enumeração, sendo bastante aplicado para a resolução de PPLIs e disponíveis em diferentes bibliotecas de otimização linear inteira. Planos de corte são restrições válidas, também chamadas de desigualdades válidas, inseridas no PPLI para, geralmente, eliminar soluções fracionárias.

Neste sentido, Arenales et al. (2007) declararam que o *branch-and-cut* combina o método *B&B* e o de planos de corte, a fim de diminuir a quantidade de nós na árvore *B&B* e

agilizar a busca por uma solução ótima. Os modelos deste trabalho são de programação linear inteira (mista) e, então, têm a sua resolução feita por um método *branch-and-cut*, disponível na biblioteca de otimização *Gurobi Optimizer*.

2.2 Problemas de programação ou *scheduling*

Fuchigami e Rangel (2014) afirmaram que a programação da produção ou *scheduling* está associada com a otimização de medidas de desempenho, como por exemplo, aproveitamento dos recursos existentes, minimização dos custos de produção e entrega dos produtos no tempo determinado. Dessa forma, busca-se por um sistema com bom desempenho e que traga satisfação para os clientes.

A programação da produção está relacionada com a determinação de horários para começar e finalizar atividades ou ordens de produção. O problema de sequenciamento deriva do problema de programação, no qual determina a sequência de processamento das ordens no sistema produtivo (COLIN, 2011).

> [...] existem problemas de programação de atividades em uma máquina, em duas máquinas, em várias máquinas em paralelo, em várias máquinas em série, em máquinas com configurações genéricas, em ordens com tempos de processamento probabilísticos, em funções-objetivo que minimizam o número de atraso das entregas, o atraso total, o atraso ponderado, o atraso e o adiantamento, o horário de término de todas as atividades e assim por diante [...] (COLIN, 2011, p. 186).

Os problemas na área de produção são divididos entre os níveis estratégico, tático e operacional. No nível estratégico, as decisões são tomadas para fins de longo prazo e têm como atribuições a definição e o projeto do processo, pois se trata de altos investimentos. O nível tático fica responsável por planejar as atividades, sendo assim, o planejamento agregado da produção consiste em atividades relacionadas com níveis de mão-de-obra, subcontratação e hora extra, decisões essas que são tomadas ao longo de meses e até um ano. Além disso, o nível tático opera também no planejamento da quantidade de produção, ou seja, quando e quanto produzir determinado produto, ao longo de semanas e até meses. Por fim, no nível operacional, as decisões tomadas estão associadas com a atribuição de atividades (*jobs*) às máquinas e a programação (*scheduling*) das atividades nas máquinas, o que correspondem, respectivamente, na sequência de processamento das atividades e na definição do instante de início e término desse processamento. Devido à variedade de ambientes de produção, grande *mix* de produtos e do desempenho da programação da produção, pode-se encontrar uma vasta e rica literatura no que se refere a problemas de *scheduling* (PINEDO, 2012).

De acordo com Arenales et al. (2007), as principais medidas de desempenho para a verificação da qualidade de um programa de produção são: *makespan*, tempo de total de fluxo, atraso máximo, atraso total, *lateness* e o número de atividades atrasadas. O *makespan* pode ser definido como sendo o momento de conclusão do processamento de todas as atividades e é uma medida de finalidade do sistema produtivo. O tempo de fluxo total consiste na somatória dos momentos de término das atividades. As demais medidas de desempenho mencionadas estão associadas com as datas de entrega das atividades.

A solução de problemas de programação, como mencionado, pode ser por meio de métodos exatos e heurísticos. Os métodos exatos utilizam algoritmos que buscam a solução ótima para o problema, porém, ainda são limitados computacionalmente, logo torna-se uma grande possibilidade de pesquisa visando o aperfeiçoando desses métodos. Os métodos heurísticos não garantem uma solução ótima, porém são muito utilizados para resolver instâncias reais (de médio e grande portes) e buscam balancear a qualidade da solução e o esforço computacional (CRAVO, 2009).

2.3 Problema de programação de projetos

De acordo com Yamashita e Morabito (2007), o problema de programação de projetos está relacionado com a programação de atividades ao longo do tempo, que precisam de recursos para serem finalizadas. A programação pode considerar relações ou restrições de precedência entre as atividades, ou seja, algumas atividades podem começar somente quando outras terminarem.

Ao longo do planejamento de um projeto pode-se ter diversos tipos de objetivos, como a minimização do consumo de algum tipo de recurso, redução do custo total ou do tempo de duração do projeto. Além dos objetivos a serem alcançados, torna-se necessário que se respeite algumas restrições de um projeto, como por exemplo, a relação de precedência entre as atividades, disponibilidade de recursos e os prazos de entrega estabelecidos (AZEVEDO; PESSOA; TORRES, 2012).

Leal (2007) declarou que o problema de sequenciamento ou programação de projetos com restrição de recursos vem ganhando destaque e um crescente interesse no meio acadêmico e também no mundo corporativo. No meio empresarial é extremamente importante, pois é possível a adaptação desses problemas à realidade dos projetos. Brucker et al. (1999) disseram que esse tipo de problema é relevante para os pesquisadores, pois considera modelos mais ricos, embora a resolução torne-se mais complexa.

Para Rocha (2011), o problema de programação de projetos com restrição de recursos possui como objetivo definir a melhor sequência de processamento das atividades

com a finalidade de elaborar cronogramas. Com os recursos alocados nas atividades e a sequência de processamento estabelecida, torna-se possível definir o momento de iniciar cada atividade para a construção do cronograma de execução.

Os recursos podem ser renováveis ou não-renováveis. Os recursos renováveis estão acessíveis em cada período, por exemplo, mão-de-obra, equipamentos, ferramentas e máquinas. Os recursos não-renováveis são restritos ao longo do horizonte de planejamento, por exemplo, a quantidade de dinheiro disponível. Além dos recursos, tem-se também objetivos de minimização da duração total do projeto (*makespan*), do tempo de atraso ou de adiantamento e do custo do projeto, assim como a maximização de valores (ARENALES et al., 2007).

Portanto, como definiu Hartmann e Briskorn (2010), o RCPSP tem um conjunto de atividades, que constituem um projeto (podendo inclusive ter vários projetos em paralelo), sujeitas às restrições de precedência e de recursos, com o objetivo de minimizar o *makespan*. Da mesma forma, Özdamar e Ulusoy (1995) abordaram diferentes objetivos e restrições presentes no RCPSP, em especial, objetivos relacionados com o tempo e custo, e restrições de recursos (não-)renováveis, visto que são representações de problemas reais.

2.4 Restrições práticas

Restrições práticas no contexto do problema de programação de projetos consideram aspectos das situações reais. Dessa forma, a solução dos problemas com essas restrições pode contribuir com a tomada de decisão dos gestores, principalmente de indústrias e empresas que lidam com tais problemas.

Algumas variantes e extensões do RCPSP que têm sido investigadas, motivadas pela necessidade das indústrias incluem: custo de disponibilidade de recursos, tempo de atraso, dados incertos, tempo de preparo (conhecido na literatura como *setup time*), prioridade de projetos, múltiplas habilidades, múltiplos modos de execução, entre outros. Sendo assim, o Quadro 2.1 apresenta algumas restrições práticas para o RCPSP, indicando o autor e suas principais características. Percebe-se que essas restrições estão associadas as limitações presentes na rotina das empresas e que podem influenciar no tempo de conclusão do projeto.

Restrição	Autor	Características
Custo de disponibilidade de recursos	Yamashita e Morabito (2007)	Custo relacionado com a disponibilidade de cada recurso
Tempos de atraso mínimo e máximo	Hartmann e Briskorn (2010)	Atrasos de tempo mínimo e máximo permitido entre a conclusão de uma atividade e o início de outra.
Dados incertos	Ke e Liu (2010)	Quantifica dados imprecisos e está relacionada com incertezas dentro do problema
Tempo de preparo	Nadjafi e Majlesi (2014)	Tempo de preparação necessário para o processamento de uma atividade
Prioridade de projetos	Singh (2014)	Impõe que os projetos sejam escalonados conforme a prioridade imposta
Múltiplas habilidades	Myszkowski et al. (2018)	Cada atividade demanda um subconjunto de habilidades, enquanto cada recurso dispõe de um subconjunto de habilidades que consegue atender
Múltiplos modos	Muritiba, Rodrigues e Costa (2018)	Conjunto de modos de execução para cada atividade. Cada modo determina uma forma diferente de processar a atividade

Quadro 2.1 – Restrições práticas para o RCPSP.

Fonte: A autora.

2.5 Métodos exatos para o RCPSP

Considerando os métodos exatos, capazes de geralmente resolver instâncias de tamanho pequeno e média, com o objetivo de encontrar a solução ótima, o Quadro 2.2 sumariza alguns dos resultados já publicados na literatura, indicando o autor, o método que eles empregaram para resolver o RCPSP e se o problema continha alguma restrição prática adicional. Observa-se que as restrições práticas mais adotadas nesses trabalhos foram a de múltiplas habilidades e de múltiplos modos.

Autor	Método	Restrição Adicional
Heilmann (2003)	*Branch-and-bound*	Múltiplos modos e tempos de atraso
Yamashita e Morabito (2007)	*Branch-and-bound*	Custo de disponibilidade de recursos e múltiplos modos de execução
Bianco e Caramia (2012)	*Branch-and-bound*	-
Correia e Saldanha-da-Gama (2014)	Modelo de programação linear inteira mista	Múltiplas habilidades
Naber e Kolisch (2014)	Modelo de programação linear inteira	-
Baumann, Fündeling e Trautmann (2015)	Modelo de programação linear inteira	Trabalho
Javanmard, Nadjafi e Niaki (2017)	Modelo de programação linear inteira mista	Múltiplas habilidades
Silva, Vieira e Silva (2017)	Modelo de programação linear inteira mista	-

Quadro 2.2 – Trabalhos que resolveram o RCPSP por algum método exato.

Fonte: A autora.

Para resolver o RCPSP na presença da restrição de múltiplos modos e de tempos de atraso, ou intervalos de tempo mínimo e máximo entre a a execução de atividades, foi proposto por Heilmann (2003) um método exato, o *branch-and-bound* com busca em profundidade, cujo objetivo é definir um modo de execução e um instante de tempo para iniciar cada atividade, respeitando as restrições impostas, além de minimizar o *makespan*. O algoritmo do autor foi testado em instâncias de 10, 30 e 50 atividades e comparado com as soluções existentes na literatura. Os experimentos computacionais retornaram resultados interessantes, porém para instâncias maiores, o método não é muito indicado.

Yamashita e Morabito (2007) apresentaram um algoritmo *branch-and-bound* para resolver o RCPSP com custo de disponibilidade de recursos e atividades com múltiplos modos de execução, conhecido como *multi-mode resource availability cost problem*. Neste problema, dada uma atividade que precise de 10 unidades de tempo para ser processada

por 1 operador, a mesma poderá ser executada em 4 unidades de tempo por 2 operadores. O objetivo do estudo foi determinar um planejamento do processamento das atividades, a fim de minimizar o custo total de destinação de recursos, sem violar a data de conclusão do projeto, as relações de precedência entre as atividades e a disponibilidade de recursos. Os resultados comprovaram que o algoritmo proposto é viável computacionalmente para resolver instâncias com até 10 atividades.

Um algoritmo exato para o RCPSP foi desenvolvido por Bianco e Caramia (2012) no qual faz uso de relações de precedência generalizadas (*Generalized Precedence Relationships*). Os autores utilizaram um *branch-and-bound*, com o limitante inferior sendo determinado por uma relaxação Lagrangeana, com o intuito de acelerar a busca por uma solução de *makespan* mínimo. A abordagem dos autores foi testada em projetos com 30, 50, 80 e 100 atividades. Uma comparação com quatro algoritmos da literatura e com limitantes inferiores já conhecidos também foi realizada, sendo que a proposta dos autores retornou os melhores resultados.

Foi proposto um modelo de programação linear inteira mista em Correia e Saldanha-da-Gama (2014) para o RCPSP com múltiplas habilidades. O problema considera diversas atividades e cada atividade demanda mais de uma habilidade e, consequentemente, mais de um recurso poderá ser necessário para processá-las. Os custos estão relacionados com o consumo de recursos e envolvem custos fixos e variáveis. Um dos objetivos do artigo foi investigar a viabilidade de utilizar um *solver* para resolver o problema. Além disso, os autores buscaram comparar as soluções alcançadas com o objetivo direcionado a custos com o objetivo mais comum para esse problema, que é a minimização do *makespan*. Experimentos computacionais foram realizados com instâncias de 20 atividades e revelaram que um *solver* é capaz de gerar boas soluções em um tempo computacional aceitável para instâncias de tamanho realístico. Os resultados demonstraram que tanto o custo ótimo, quanto o *makespan* final, possuem relações com insumos, componentes de custos, quantidade de recurso e tempo de execução do projeto.

O trabalho de Naber e Kolisch (2014) apresenta modelos de programação linear inteira para variantes do RCPSP contendo recursos flexíveis. Esse tipo de problema é muito comum em situações reais e busca pela definição do melhor momento de início das atividades, com seus tempos de duração e recursos a serem utilizados, para minimizar o *makespan*, submetido a restrições de precedência e de limitação de múltiplos recursos. Os autores propuseram quatro modelos com tempo discreto baseados em outros modelos já desenvolvidos e, em seguida, fez-se uma comparação entre os modelos considerando o tamanho das instâncias, tempo de execução e a qualidade das soluções. As instâncias utilizadas foram obtidas de Fündeling e Trautmann (2010), divididas em dois conjuntos, um deles da base PSPLIB (*Project Scheduling Problem Library*) e os outros com 10, 20 e 40

atividades. Os resultados mostraram que o terceiro modelo foi mais eficiente que os demais em todos os casos de análise.

Baumann, Fündeling e Trautmann (2015) resolveram o RCPSP contendo recursos flexíveis e restrições de trabalho. Em Javanmard, Nadjafi e Niaki (2017), para resolver o RCPSP com múltiplas habilidades, dois modelos de programação linear inteira mista foram desenvolvidos a fim de minimizar o custo total de recrutamento para níveis de habilidades distintos. Os níveis mais altos de habilidades possuem um processamento mais rápido e, consequentemente, o custo é maior, e cada atividade pode demandar uma ou mais habilidades para serem processadas. Os modelos foram validados a partir de 30 instâncias de pequeno porte, sendo que os resultados revelaram que ambos os modelos são satisfatórios.

Em Silva, Vieira e Silva (2017), um modelo de programação linear inteira mista foi proposto para o RCPSP com o intuito de minimizar o *makespan*, sendo que as atividades possuem tempos definidos para serem executadas e consomem certas quantidades de recursos renováveis. Além disso, foi desenvolvido um *software* para atuar na interface entre o usuário e o modelo matemático, de forma que é possível incluir os dados de entrada, solucionar o problema e gerar os resultados graficamente, contendo a sequência das atividades, tempo de início e término e os recursos gastos. Dessa forma, um arquivo é gerado pelo *software* contendo informações e análises do processo de otimização, auxiliando na interpretação dos resultados por parte dos tomadores de decisão. O modelo foi adotado em uma planta de análise química de minério de ferro, na qual um estudo de caso foi realizado. Com o uso do *software* foi possível perceber que não havia planejamento algum sobre as atividades, visto que os funcionários responsáveis tinham apenas uma visão aproximada da melhor sequência, tomada com base na experiência. Com o sistema foi possível ter uma visão geral do processo, além de possibilitar uma análise mais detalhada, rápida e precisa do problema.

O RCPSP tornou-se um problema padrão no contexto da programação de projetos, consequentemente, muitas extensões do RCPSP têm sido investigadas. Além dos trabalhos acima citados, que resolveram o RCPSP por algum método exato, há também alguns *surveys* que discutiram aspectos gerais do problema e de restrições reais. Hartmann e Briskorn (2010), por exemplo, comentaram sobre a restrição de tempos de atraso mínimo e máximo, que compreendem atrasos ou intervalos de tempo mínimo e máximo entre o término de uma atividade *i* e o início de outra atividade *j*. Essa restrição pode também ser estudada como o tempo mínimo entre o instante de início de *i* e o início de *j*, entre o instante de início de *i* e o de término de *j*, ou tempo mínimo entre o término de *i* e o término de *j*. Portanto, tratam-se de formas diferentes de analisar os tempos de atraso na perspectiva do RCPSP. Analogamente, tem-se distintas formas para analisar o atraso máximo.

Em Mika, Rózycki e Waligóra (2012) foi considerado um modelo matemático para tratar o tempo de preparo entre atividades no RCPSP com restrições de múltiplos modos. Os autores descreveram diferentes tipos de operações e custos de preparo que podem ocorrer no problema, bem como apresentaram uma formulação geral de programação linear inteira para o problema.

2.6 Métodos heurísticos para o RCPSP

Quanto aos métodos heurísticos, desenvolvidos para resolver de modo rápido e satisfatório instâncias grandes para o RCPSP, o Quadro 2.3 traz um resumo de algumas contribuições feitas pela literatura. Nota-se nesses trabalhos que há um interesse maior pelas restrições de múltiplas habilidades e de múltiplos modos.

Mendes e Gonçalves (2003) desenvolveram um algoritmo genético, fundamentado na técnica de seleção natural e genética, para resolver o RCPSP, em que a representação cromossômica consiste em chaves aleatórias. Por meio de uma heurística fundamentada em prioridades determinadas pelo algoritmo genético, a programação das atividades é realizada, retornando um sequenciamento ativo e parametrizado das atividades. O algoritmo foi submetido a testes e um estudo comparativo foi realizado com três conjuntos de problemas, com 30, 60 e 120 atividades, respectivamente. Os resultados computacionais do algoritmo desenvolvido indicaram desempenho superior quanto a qualidade da solução em comparação com outras heurísticas.

Baradaran et al. (2010) desenvolveram uma heurística para o RCPSP em redes PERT. Nesse caso, as atividades demandam vários tipos de recursos com duração aleatória, com o objetivo de minimizar o *makespan*. Uma busca dispersa híbrida foi sugerida. Para gerar novas soluções, a heurística utiliza três operadores, a saber: cruzamento de dois pontos (*two-point crossover*), *path-relinking* e permutação. O desempenho da heurística foi verificado através da comparação de seus resultados com a solução ótima, considerando redes pequenas. Os resultados demonstraram que a heurística é eficaz para redes de pequeno porte e também para problemas reais.

Hoehene, Kuster e Lorenzoni (2010) abordaram o RCPSP com múltiplos modos de processamento. A heurística busca dispersa foi aplicada, uma vez que ela atua sobre uma população de soluções e adota procedimentos para retornar novas soluções mediante a combinação de soluções anteriores. A partir dos experimentos, notou-se que o algoritmo é fácil de ser alterado, demonstrando a sua flexibilidade, para, inclusive, acrescentar outros procedimentos, como o aninhamento de conjuntos de referências. Sendo assim, os resultados gerados pela heurística para instâncias com 20 atividades foram comparados

com outros algoritmos e revelaram a sua competitividade, uma vez que foi possível encontrar os mesmos resultados da literatura.

Autor	Método	Restrição Adicional
Mendes e Gonçalves (2003)	Algoritmo genético	-
Baradaran et al. (2010)	Busca dispersa híbrida	-
Hoehene, Kuster e Lorenzoni (2010)	Busca dispersa com aninhamento de conjuntos e referências	Múltiplos modos de processamento
Ke e Liu (2010)	Algoritmo híbrido com simulação *fuzzy* e algoritmo genético	Dados incertos
Azevedo, Pessoa e Torres (2012)	Algoritmo genético	Tempo de atraso mínimo
Rivera et al. (2013)	Algoritmo híbrido combinando GRASP, busca dispersa e *justification*.	-
Singh (2014)	Algoritmo híbrido e *Analytic Hierarchy Process*	Prioridade de projetos
Bukata, Sucha e Hanzálek (2015)	Busca tabu	-
Almeida, Correia e Saldanha-da-Gama (2016)	Programação paralela	Múltiplas habilidades
Liu et al. (2017)	Decomposição em janelas de tempo	-
Oztemel e Selam (2017)	Algoritmo de abelhas	Múltiplos modos
Muritiba, Rodrigues e Costa (2018)	Algoritmo com *Path-Relinking*	Múltiplos modos
Wang e Zheng (2018)	Algoritmo de otimização *multi-objective fruit fly*	Múltiplas habilidades

Quadro 2.3 – Trabalhos que resolveram o RCPSP por algum método heurístico.

Fonte: A autora.

Uma variante do RCPSP com restrições *fuzzy* para modelar incertezas no problema foi resolvida por Ke e Liu (2010) cujo objetivo é definir o cronograma de atribuição de recursos a fim de balancear o tempo de conclusão do projeto, considerando o custo total. Três tipos de modelos *fuzzy* foram construídos para o RCPSP, conforme os objetivos esperados, como a minimização de custos e da maximização de credibilidade. Um algoritmo híbrido que abrange simulação *fuzzy* e algoritmo genético foram desenvolvidos pelos autores para resolver várias instâncias e mostrar a eficácia de cada um.

Azevedo, Pessoa e Torres (2012) adotaram também um algoritmo genético para resolver uma variante do RCPSP, no qual um estudo de caso foi realizado em uma empresa especializada em petróleo e gás. O algoritmo foi adaptado para lidar com algumas peculiaridades da empresa, a fim de criar uma ferramenta para contribuir e facilitar com a tomada de decisões voltadas para a destinação de recursos e planejamento dos projetos. Todas as soluções obtidas estavam de acordo com restrições de precedência na presença de um intervalo de tempo mínimo entre os inícios das atividades (conhecido na literatura, como *time lag*) e de recursos. Além disso, um modelo de programação linear inteira mista foi utilizado para comparar com os resultados da heurística, para testes computacionais sobre instâncias com até 20 atividades. Os resultados indicaram que o cronograma do projeto é mais sensível para os recursos com maior especialização. Isso permitiu identificar possibilidades de melhorias futuras, além de evidenciar vantagens de se utilizar métodos de otimização no ambiente empresarial.

Um método heurístico híbrido foi proposto por Rivera et al. (2013) para resolver o RCPSP, em que se combinou o *Greedy Randomized Adaptive Search Procedure* (GRASP) com busca dispersa (*Scatter Search*) e um método de *justification*. O método busca encontrar soluções aproximadas, em que a cada iteração há a construção de uma solução, seguida da aplicação de procedimentos de melhoria. Os testes foram realizados a partir de três conjuntos, com instâncias de 30, 60 e 120 atividades, sendo comparado com resultados da literatura. Com a heurística, os autores conseguiram uma melhora em torno de 68%, justificando a necessidade de procedimentos para melhorar o desempenho de qualquer método heurístico.

Singh (2014) apresentou um algoritmo híbrido para resolver o RCPSP com a restrição de prioridade de projetos, com o objetivo de minimizar o *makespan* e o custo de penalidade relacionado ao atraso de projetos de maior prioridade. Por meio de um algoritmo híbrido com regras de prioridade e usando o *Analytic Hierarchy Process*, que concede pesos aos projetos com comparações na forma de pares, o cronograma do projeto foi elaborado. O estudo de caso foi feito em uma empresa de projetos, para desenvolver o cronograma de

cinco projetos, cada um com oito atividades. Os resultados computacionais revelaram a eficácia do algoritmo sugerido se comparado com outros existentes.

Bukata, Sucha e Hanzálek (2015) desenvolveram uma heurística de busca Tabu paralela para a busca de boas soluções, em um tempo baixo de processamento, para o RCPSP. A heurística retorna soluções com base em movimentos de troca, respeitando as relações de precedência, ao mesmo tempo em que as soluções iguais são armazenadas em uma lista tabu para evitar possíveis repetições e a busca ficar parada em ótimos locais. Para os testes computacionais, os autores utilizaram quatro conjuntos, com instâncias contendo 30, 60, 90 e 120 atividades, respectivamente. O algoritmo proposto foi capaz de superar vários outros métodos existentes quanto à qualidade da solução e a quantidade de instâncias com a melhor solução conhecida.

Almeida, Correia e Saldanha-da-Gama (2016) propuseram uma heurística paralela para o RCPSP com múltiplas habilidades. Para tanto, dois conceitos foram abordados, o peso do recurso e o agrupamento de atividades. O objetivo do RCPSP com múltiplas habilidades é definir, para cada atividade, o início de seu processamento, os recursos que devem ser alocados para cada atividade e as habilidades requeridas, visando minimizar o *makespan.* A heurística foi testada utilizando dois conjuntos de instâncias, cada um contendo 216 instâncias. As instâncias do primeiro conjunto possuem 20 atividades e a do segundo 40 atividades. Os experimentos computacionais indicaram que a heurística é eficaz para a busca de soluções de qualidade em tempo reduzido.

Para resolver o RCPSP, uma heurística baseada em decomposição em janelas de tempo foi proposta por Liu et al. (2017). A heurística considera a escolha de atividades e, assim, divide o espaço de busca de forma que as soluções são geradas por meio de um planejamento serial. Dessa forma, um método de decomposição foi utilizado para dividir o espaço viável do problema original em subespaços. A heurística foi comparada com outros métodos existentes, mostrando que a estrutura de decomposição é eficaz e traz resultados competitivos, além de ser possível combiná-la com outras heurísticas.

Em Oztemel e Selam (2017) há um algoritmo baseado no comportamento de abelhas, conhecido por otimização de colônia de abelhas, sendo proposto para o RCPSP com recursos únicos e limitados, e múltiplos modos de execução, com o objetivo de encontrar o menor tempo de conclusão de projetos de moldagem por injeção. Os autores usaram um gerador de projeto, o PROGEN, para definir as limitações do projeto de fabricação de moldes. Os testes foram realizados a partir de instâncias com o número de atividades variando entre 10 e 80. Com a alteração nos modos de execução, o espaço de busca da solução aumentou consideravelmente. Após a implementação do algoritmo, o mesmo mostrou-se eficiente, gerando cronogramas apropriados para os projetos dessa natureza, mesmo para um número elevado de atividades e recursos limitados. Os autores

notaram que melhorias ainda são precisas para estender o algoritmo e considerar orçamento e algumas especificidades do problema real.

Um algoritmo *Path-Relinking* foi proposto por Muritiba, Rodrigues e Costa (2018) para resolver o RCPSP com a restrição de múltiplos modos, no qual as atividades que fazem parte de um projeto dispõem de um conjunto de modos de execução, e para cada modo, o tempo de duração das atividades, o consumo de recursos renováveis e não-renováveis são diferentes, cujo objetivo foi minimizar o *makespan*. Experimentos computacionais foram realizados sobre três conjuntos distintos de instâncias, com o número de atividades variando entre 10 e 100, a fim de comparar os resultados obtidos com métodos da literatura. O algoritmo mostrou-se eficiente, gerando resultados satisfatórios para os mesmos conjuntos de instâncias adotados na literatura, apresentando tempos computacionais similares.

Wang e Zheng (2018) utilizaram um algoritmo multiobjetivo que simula o comportamento da mosca-de-fruta para resolver o RCPSP com múltiplas habilidades, com o objetivo de minimizar o *makespan*. Os autores utilizaram 15 instâncias de 100 atividades e 15 instâncias de 200 atividades para validá-lo. Os resultados computacionais mostraram a eficácia do algoritmo quando comparado com outros métodos da literatura.

CAPÍTULO 3

MODELOS MATEMÁTICOS PARA O PROBLEMA DE PROGRAMAÇÃO DE PROJETOS COM RESTRIÇÃO DE RECURSOS

Este capítulo traz uma descrição formal do problema em estudo, apresentando os parâmetros utilizados, as restrições que devem ser respeitadas e a função objetivo que mede a qualidade das soluções. Na sequência, as restrições utilizadas são comentadas, para depois expor e discutir os modelos investigados neste trabalho.

3.1 Definição do RCPSP

Para resolver o RCPSP é necessário que as relações de precedência, em *pred*, entre as atividades sejam satisfeitas; a quantidade de cada recurso consumida durante cada instante de tempo não deve ultrapassar a disponibilidade total do respectivo recurso; e, que o instante de início de cada atividade seja determinado; tal que o objetivo é minimizar o *makespan*, que resulta no tempo de conclusão total do projeto, ou no tempo de conclusão da última atividade (BRUCKER et al., 1998). A partir disso, tem-se os seguintes conjuntos definidos para o problema:

- ***V* = {0, 1, 2, ..., *n*, *n+1*}:** possui as *n* atividades que fazem parte do projeto, sendo que as atividades *0* e *n+1* são fictícias;
- ***R* = {1, 2, ..., *r*}:** conjunto dos *r* recursos renováveis;
- ***pred*:** conjunto de pares ordenados *(i, j)* de atividades com relação de precedência, indicando que a atividade *j* só poderá iniciar após o término da atividade *i*;

- C_{ib}: quantidade do recurso *b* que a atividade *i* consome durante o seu processamento, sendo que as atividades fictícias não consomem recursos;
- R_b: disponibilidade total do recurso *b*, que é renovável;
- d_i: tempo de processamento (isto é, duração) da atividade *i*, assumido como um valor inteiro, sendo que as atividades fictícias possuem duração igual a zero.

Além disso, o problema requer que cada atividade, uma vez iniciada, não seja interrompida até o término do seu tempo de processamento, ou seja, não é permitido preempção de atividades; as atividades podem ocorrer em paralelo, desde que o consumo dos recursos seja respeitado em cada instante de tempo; a partir do tempo zero, as atividades podem iniciar a qualquer momento, desde que as relações de precedência sejam respeitadas; as atividades não podem ser fracionadas; e, o tempo de preparo entre as atividades é desprezível, sendo considerado igual à zero.

Apresenta-se uma instância de exemplo para o RCPSP, que contêm 6 atividades e 2 recursos renováveis, ambos com capacidade igual a 5, isto é, a quantidade máxima disponível do recurso em cada instante de tempo para ser usado ao processar as atividades. A Tabela 3.1 apresenta os dados da instância, contemplando as 6 atividades que fazem parte de um projeto, considerando que as atividades 0 e 7 são fictícias; a duração de cada uma; a quantidade de recurso consumida por cada atividade, e por fim, as atividades que as precedem, ou seja, as atividades que precisam ser processadas primeiro para que a atividade corrente seja processada (por exemplo, a atividade 3 depende das atividades 1 e 2 finalizarem para iniciar o seu processamento). Uma solução viável para essa instância é apresentada na Figura 3.1.

Tabela 3.1 – Exemplo de uma instância para o RCPSP.

Atividade j	d_j	C_{j1}	C_{j2}	Predecessor
0	0	0	0	-
1	2	2	2	-
2	3	2	1	-
3	3	4	3	1, 2
4	2	1	2	3
5	4	3	1	4
6	3	5	3	5
7	0	0	0	-

Fonte: A autora.

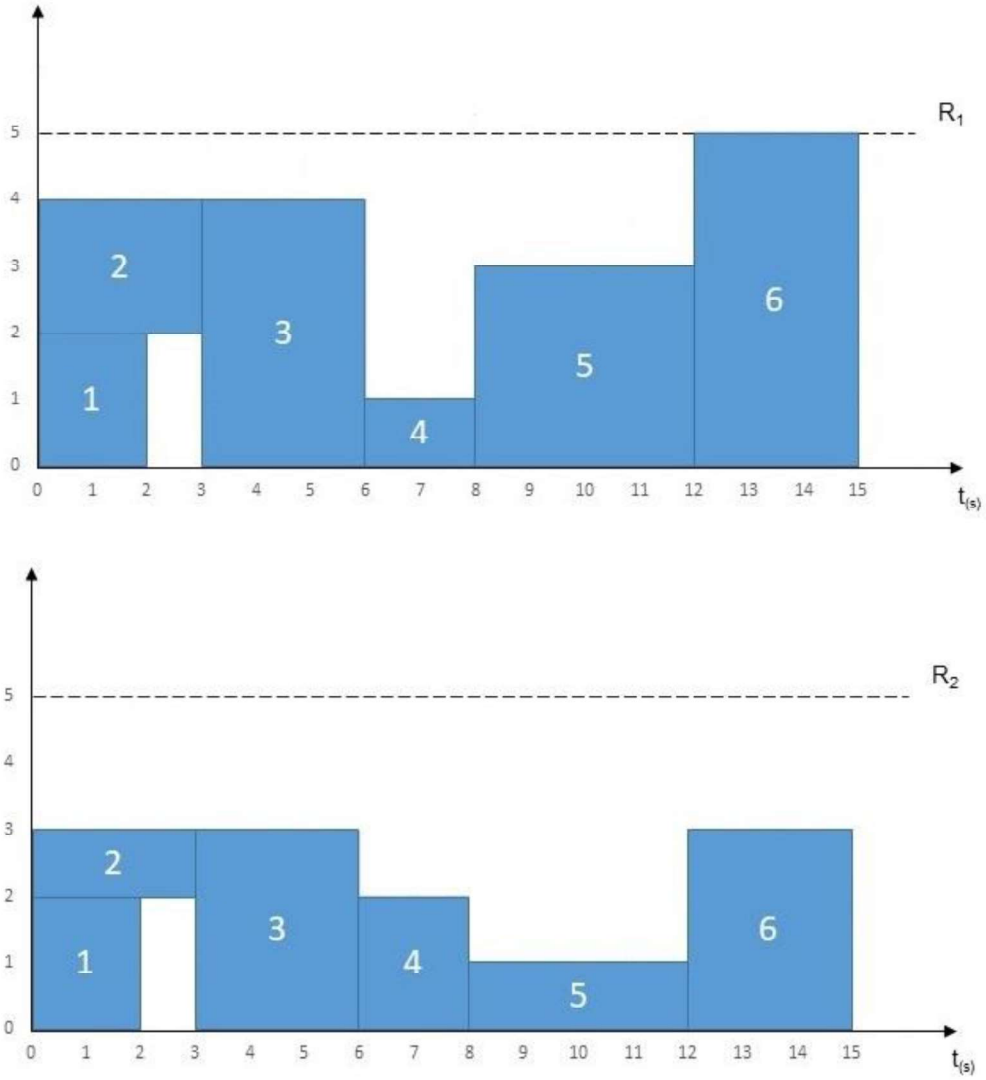

Figura 3.1 – Solução viável para a instância da Tabela 3.1.

Fonte: A autora.

A solução viável para a instância da Tabela 3.1, representada pela Figura 3.1, mostra as 6 atividades sendo processadas ao longo do horizonte de tempo, de maneira que as restrições de precedência e de disponibilidade de recursos são satisfeitas.

3.2 Modelos para o RCPSP

Os modelos de programação linear inteira apresentados para o RCPSP utilizam variáveis inteiras que indicam o momento de início de cada atividade. Uma referência para

esses modelos, cujas variáveis inteiras são indexadas nas atividades e no tempo, e outros, baseados em fluxos e com variáveis contínuas, é o trabalho de Koné et al. (2011).

Discute-se inicialmente dois modelos para o RCPSP, sendo que as atividades são programadas em um horizonte de tempo, variando de [*0, T*], em que *T* é um limitante superior para o *makespan*. Um valor para *T* pode ser obtido somando o tempo de processamento de todas as atividades, isto é, $T = \sum_{i \in V} d_i$.

O primeiro modelo discutido é de programação linear inteira mista e envolve dois tipos de variáveis: *h*, que é não-negativa, para representar o *makespan*; e, x_{ip}, que é binária, para indicar com o valor 1 se a atividade *i* inicia seu processamento no tempo *p,* e o valor 0 para o caso contrário, para *i* = 0, ..., *n+1,* e todo $p \in P$. O instante de início *p* de processamento de uma atividade, pertencente ao conjunto *P,* refere-se a todos os possíveis instantes de início de uma atividade, que pode variar de uma em uma unidade no intervalo [*0, T – d*$_i$], pois os tempos de processamento são inteiros. O primeiro modelo (isto é, Modelo 1) é, então, descrito como:

Minimizar *h* (3.1)

Sujeito a:

$$\sum_{p \in P} x_{ip} = 1, \quad i = 0,1,...,n+1 \tag{3.2}$$

$$\sum_{i=0}^{n+1} \sum_{p \in \{t - d_i + 1 \le p \le t\}} C_{ib} x_{ip} \le R_b, \quad \forall t \in P,\ b = 1,2...,r \tag{3.3}$$

$$x_{it} \le \sum_{q \in P, q \ge t + d_i} x_{jq}, \quad \forall (i,j) \in pred,\ \forall t \in P \tag{3.4}$$

$$h \ge \sum_{p \in P} x_{ip}(p + d_i), \quad i = 0,1,...,n+1 \tag{3.5}$$

$$LI \le h \le LS \tag{3.6}$$

$$x_{ip} \in \{0,1\}, \quad i = 0,1,...,n+1,\ \forall p \in P \tag{3.7}$$

A função objetivo (3.1) representa a minimização do *makespan*. As restrições em (3.2) garantem que toda atividade deve ser iniciada em exatamente um instante de tempo *p*, podendo ter atividades que iniciam no mesmo instante de tempo. As restrições em (3.3) impõem que o consumo de recursos pelas atividades iniciadas no tempo *p* respeite a disponibilidade de cada recurso por instante de tempo. As restrições (3.4) asseguram que a

relação de precedência entre as atividades seja satisfeita, impondo que uma atividade só pode iniciar após o término de todas as suas predecessoras. Nas restrições em (3.5), tem-se o cálculo do *makespan*, que deve ser maior ou igual ao instante de término da última atividade processada, enquanto a restrição (3.6) impõe que o *makespan* deve ser maior ou igual a um limitante inferior (LI) e menor ou igual a um limitante superior (LS) para a solução ótima, os quais são discutidos na Seção 3.3. As restrições (3.7) indicam o domínio das variáveis x_{ip}.

É importante mencionar que a função objetivo e as restrições (3.4) do primeiro modelo são diferentes do modelo que serviu como base, descrito em Koné et al. (2011), ao passo que as restrições (3.2) e (3.7) são as mesmas descritas em Koné et al. (2011). As restrições (3.3) e (3.5) são propostas diferentes em comparação ao trabalho de tais autores.

O segundo modelo é uma adaptação do primeiro modelo, sendo apenas de programação linear inteira, uma vez que a variável *h* é eliminada. Esse modelo pode ser encontrado em Pritsker, Watters e Wolfe (1969). As variáveis do segundo modelo são as mesmas x_{ip} do primeiro modelo, assim resultando no Modelo 2:

Minimizar $\sum_{p \in P} p x_{(n+1)p}$ (3.8)

Sujeito a:

(3.2), (3.3), (3.6) e (3.7)

$$\sum_{p \in P} p x_{jp} - \sum_{p \in P} p x_{ip} \geq d_i \,, \quad \forall (i,j) \in pred \tag{3.9}$$

A função objetivo (3.8) está relacionada com a minimização do *makespan*, que pode ser obtido diretamente a partir do tempo de conclusão da última atividade, que é a *n+1*. As restrições (3.2), (3.3), (3.6) e (3.7) são as mesmas do primeiro modelo. Por outro lado, apresentam-se as restrições (3.9) para as relações de precedência entre as atividades, impondo que a atividade *j* só poderá ser iniciada após a conclusão da *i*, para todas as atividades *i* que devem ser processadas antes da *j* iniciar. Observa-se que o segundo modelo possui uma restrição a menos, sendo objeto de comparação com o primeiro modelo nos experimentos computacionais.

3.3 Limitantes para o RCPSP

A partir de limitantes inferiores e superiores é possível reduzir o espaço de busca no momento de resolução de modelos de programação linear inteira. Para problemas de minimização, um limitante inferior válido pode ser obtido da solução da relaxação do modelo ou de um modelo relaxado (isto é, que não inclui todas as restrições do problema original), enquanto que um limitante superior válido advém de qualquer solução viável do problema, podendo ser utilizada uma heurística para tal fim (WOLSEY, 1998).

No caso dos modelos para o RCPSP, busca-se reduzir o espaço de soluções a partir da redução do domínio da variável que mensura o *makespan*. Portanto, um limitante inferior para o *makespan* representa o mínimo instante de tempo (ou o instante mais cedo) em que o projeto pode ser concluído. Por outro lado, um limitante superior para o *makespan* compreende um instante de tempo válido em que o projeto pode finalizar. Considerando um exemplo, a instância J301.1 da PSPLIB (*Project Scheduling Problem Library*), biblioteca com instâncias do RCPSP proposta por Kolisch e Sprecher (1997), tem-se que o valor do *makespan* é de 43, com um limitante inferior válido de 38 e um limitante superior válido de 57. Assim, para essa instância, o instante mais cedo em que o projeto poderia ser finalizado corresponde a 38 e um instante de tempo válido em que o projeto poderia terminar é de 57.

Em primeiro lugar, ao analisar o horizonte de tempo [*0, T*] e, sabendo que o valor do tempo de processamento d_i é um inteiro não-negativo, nota-se que não é preciso considerar todos os valores inteiros no intervalo [*0, T*] como possibilidade de instantes de tempo para iniciar uma atividade. Conforme a estratégia definida por Herz (1972), com relação aos *canonical dissections,* para a resolução de um problema de corte de itens bidimensionais, torna-se também suficiente para o RCPSP discretizar o horizonte de tempo em valores inteiros relacionados ao tempo de processamento em que as atividades podem iniciar. Com isso, adota-se os *canonical dissections* como as posições viáveis para iniciar as atividades.

Herz (1972) mostrou que cada item pode ser arranjado mais à esquerda e abaixo possível dentro do recipiente, até encostar nos lados de outros itens ou do recipiente. Portanto, faz-se uma discretização observando as dimensões dos itens e do recipiente, de forma a eliminar pontos que não vão interferir na solução ótima. A Figura 3.2 exemplifica as possíveis posições que itens bidimensionais podem ser arranjados dentro de um recipiente bidimensional. Dessa forma, tem-se a malha à esquerda com todos os inteiros e a malha à direita com os *canonical dissections*, sendo eliminado os pontos mais ao final do recipiente que não permitem que a menor dimensão coubesse dentro do recipiente. Para este exemplo foram considerados três itens, sendo o item 1 de largura 2 e comprimento 1; o item 2 de largura 2 e comprimento 2; o item 3 de largura 3 e comprimento 1; e o recipiente de largura e comprimento iguais a 5.

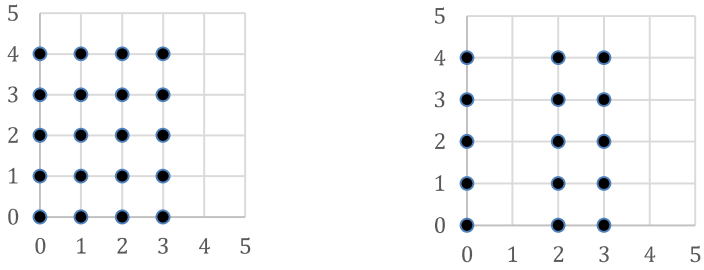

Figura 3.2 – Diferença entre uma malha inteira e outra com os *canonical dissections*.

Fonte: A autora.

Nota-se pela Figura 3.2 que alguns pontos foram eliminados. Dessa forma, ao fazer uma analogia entre o problema de corte com o RCPSP no que tange à discretização do horizonte de tempo, as atividades representam os itens e o tempo de processamento das atividades corresponde às dimensões dos itens. O objetivo com a discretização é agilizar a obtenção de uma solução ótima, uma vez que poderá acarretar em diminuição significativa do número de variáveis dos modelos, sendo essa uma contribuição para os modelos.

O conjunto *P* usado nas restrições para os modelos do RCPSP passa a considerar, a partir de agora, os instantes de tempo determinados sobre os *canonical dissections*. Para tanto, obtém-se tais instantes fazendo todas as combinações cônicas (isto é, combinações binárias não negativas) entre os tempos de processamento das atividades. Em outras palavras, um elemento *p* está em *P* se existe um subconjunto de atividades cuja soma de seus tempos de processamento seja exatamente *p*. Um algoritmo para calcular o conjunto *P* consiste no algoritmo DP descrito em Cintra et al. (2008).

Com relação a um limitante inferior (LI) para o RCPSP, resolve-se uma relaxação dos modelos apresentados, seguindo o apresentado em Drexl e Kimms (2001) que considerou uma relaxação Lagrangeana. Com isso, a relaxação considerada consiste em remover algumas restrições do modelo original, de forma que o modelo possa eventualmente ser resolvido mais rapidamente. No caso de uma relaxação Lagrangena, as restrições removidas do modelo são acrescentadas na função objetivo com variáveis multiplicando-as. Essas variáveis são os multiplicadores de Lagrange e funcionam como pesos que penalizam a função objetivo caso essas restrições sejam violadas (ARENALES et al., 2007).

Com isso, um limitante inferior LI para o *makespan*, dos dois modelos apresentados anteriormente para o RCPSP, é obtido por meio da resolução do respectivo modelo sem as restrições de consumo de recursos (3.3) e a de limitantes válidos (3.6). Conforme Drexl e

Kimms (2001) as restrições de consumo de recursos acabam dificultando a resolução do modelo para a obtenção de uma solução inteira, embora para algumas instâncias, resolver esse modelo relaxado possa ser bastante caro computacionalmente. Como a solução desse modelo relaxado pode não respeitar as restrições de consumo de recursos, embora respeite todas as demais restrições, tal solução passa a ser um limitante inferior para a solução ótima do RCPSP.

Com isso, no caso do primeiro modelo, com função objetivo (3.1) e restrições (3.2)-(3.7), o modelo relaxado para obter o limitante inferior consiste em resolver (3.1), (3.2), (3.4), (3.5) e (3.7). No caso do segundo modelo, com função objetivo (3.8) e restrições (3.2), (3.3), (3.6), (3.7) e (3.9), o modelo relaxado para obter o limitante inferior consiste em resolver (3.8), (3.2), (3.7) e (3.9).

Por outro lado, o cálculo do limitante superior (LS) do *makespan* foi baseado em um método heurístico, mais precisamente, no *List Scheduling*, definido em Gafarov, Lazarev e Werner (2010) e descrito nos 10 passos seguintes. Esta heurística escalona as atividades observando a lista de precedência e a disponibilidade de recursos em cada instante de tempo, de forma que uma atividade *i* é escalonada no instante *t* somente quando as suas predecessoras já foram processadas e a quantidade de cada recurso disponível naquele instante de tempo permite escalonar *i*. Os parâmetros/variáveis usados pela heurística são:

- ***V:*** corresponde ao conjunto de atividades que não possuem precedência, obtido do conjunto com todas as atividades da instância;
- $R_b(\tau)$**:** disponibilidade do recurso *b* no instante de tempo τ;
- S_i**:** tempo de início da atividade *i*;
- C_i**:** tempo de término da atividade *i*;
- d_i**:** duração da atividade *i*;
- C_{ib}**:** quantidade de recurso *b* que a atividade *i* precisa para ser executada;
- **t:** um instante de tempo, que pode variar de *0, 1, ..., T*;
- t_b**:** um instante de tempo considerando a utilização do recurso *b*.

Os passos de 1 a 10 a serem seguidos pela heurística *List Scheduling* são descritos a seguir, observando que ela finaliza quando todas as atividades tiverem sido escalonadas:

1. Dado um conjunto de atividades sem predecessores V e faça $R_b(\tau) = R_b$, para $\tau = 0,1,...,T, \;\; b = 1,2...,r$;
2. Se $V = \phi$, então vá para o passo 10;
3. Escolha a atividade $i \in V$ com o menor tempo de processamento d_i;

4. Se $i = 0$ (isto é, i é a atividade fictícia inicial), então faça $t = 0$. Caso contrário, faça $t = \max_{\{h|(h,i)\in pred\}}\{S_h + d_h\}$ (isto é, *t* é o maior instante de término dentre todas as atividades *h* que são predecessoras de i);
5. Se existir algum recurso b em que $C_{ib} > R_b(\tau)$ para qualquer $\tau \in [t, t + d_i)$, então calcule o menor instante de tempo $t_b > t$, de forma que o processamento de i deve ocorrer no intervalo $[t_b, t_b + d_i)$, dado o recurso b. Caso contrário, passe para a etapa 7;
6. Faça $t = t_b$, então volte para o passo 5 a fim de verificar novamente se o consumo de todos os recursos é respeitado para a atividade i no tempo *t*;
7. Faça o instante de início e término da atividade i ser $[S_i, C_i) = [t, t + d_i)$;
8. Atualize o uso dos recursos: $R_b(\tau) = R_b(\tau) - C_{ib}$, para $b = 1,2,...,r$, $\forall \tau \in [S_i, C_i)$;
9. Faça $V = V \setminus \{i\}$ (isto é, remova i do conjunto V, pois ela acabou de ser escalonada). Além disso, adicione ao conjunto V os sucessores *j* da atividade i, desde que $j \notin V$ e todos os predecessores de *j* já tenham sido escalonados. Volte para o passo 2;
10. PARE e retorne o escalonamento encontrado.

A heurística *List Scheduling* retorna com uma solução viável para o RCPSP, sendo um limitante superior do *makespan*. Inicialmente, a heurística encontra todas as atividades *V* que não possuem predecessor e inicializa as variáveis $R_b(\tau)$ que guardam a quantidade de cada recurso *b* ainda disponível em cada instante de tempo τ. O segundo passo diz que se o conjunto de atividades *V* sem precedência for vazio, então todas as atividades já foram escalonadas e a heurística pode finalizar. Caso contrário, no terceiro passo, escolhe-se a atividade de *V* que tem o menor tempo de processamento para ser escalonada. Para tanto, o quarto passo verifica se essa atividade é a fictícia 0, que neste caso é escalonada no instante de tempo zero, caso contrário, o instante de tempo *t* para a atividade *i* é definido observando, entre todas as atividades *h* predecessoras a *i*, qual delas foi a última a terminar, uma vez que *i* só pode começar depois dessa que terminou por último. Além disso, o instante de tempo *t* para a atividade *i* começar também depende da disponibilidade de recurso. Então, no quinto e sexto passos, verifica-se, para todos os recursos *b*, para a atividade *i*, qual o tempo *t* mais cedo no horizonte de planejamento (depois que terminou o processamento de todas as suas predecessoras) em que a atividade *i* pode iniciar, respeitando a disponibilidade de cada recurso.

Uma vez determinado o instante de tempo que a atividade *i* pode iniciar, o sétimo passo da heurística *List Scheduling* consiste em fazer essa atribuição para as variáveis que marcam o início e fim do processamento de *i*. Após escalonar *i*, os recursos são atualizados no oitavo passo, no qual se deve descontar das variáveis $R_b(\tau)$, na janela de processamento da atividade *i*, a quantidade C_{ib} que *i* consome do recurso *b*, para todos os recursos. O nono passo faz a inclusão no conjunto *V* de todas as atividades sucessoras da atividade *i* e que não possuem qualquer outra atividade predecessora (isto é, as suas predecessoras já foram escalonadas), voltando ao segundo passo para escalonar uma outra atividade de *V* e, assim, sucessivamente. Por fim, o décimo passo encerra a heurística e retorna o escalonamento encontrado. Vale observar que no caso da inclusão de outras restrições reais no RCPSP, a heurística pode deixar de ser válida, sendo preciso fazer modificações nela de acordo com as especificidades das novas restrições.

3.4 Variantes do RCPSP

Além das restrições de precedência e recursos já presentes no RCPSP, tem-se observado a inserção de outras restrições, como é o caso de múltiplas habilidades, múltiplos modos e a de tempos de atraso mínimo e máximo, de forma a tornar o problema mais próximo das situações reais encontradas nas aplicações do dia a dia.

Para Almeida, Correia e Saldanha-da-Gama (2016), no RCPSP cada recurso possui uma só habilidade, todavia no RCPSP com restrições de múltiplas habilidades, um recurso pode ter uma ou várias habilidades. Assim sendo, uma determinada atividade pode requisitar mais de um recurso para atender às habilidades exigidas para o seu processamento. Esse tipo de restrição é comum quando recursos humanos e máquinas multifuncionais estão inseridas em um projeto, uma vez que nem todos os recursos possuem as mesmas habilidades e/ou estão no mesmo nível. Em Wang e Zheng (2018), tal restrição é entendida a partir da ideia de que cada recurso dispõe de várias habilidades com determinados níveis. Além disso, cada atividade demanda certas habilidades em um dado nível para ser processada, sendo necessário determinar quais recursos estarão aptos para atender as atividades durante o seu processamento.

No RCPSP com a restrição de múltiplos modos, as atividades podem possuir múltiplos modos de execução. Em cada modo de execução, a atividade consome uma dada quantidade de cada recurso e, consequentemente, o tempo de duração do projeto, conforme o modo adotado pode ser diferente (SILVA; MORABITO; YAMASHITA, 2014). Essa restrição surge em ambientes que envolvem pessoas e/ou máquinas diferentes para executar atividades.

Quando o RCPSP considera um tempo de atraso mínimo e máximo entre atividades *i* e *j* (isto é, o *time lag*), tem-se que entre o término da atividade *i* e o início da *j* é preciso considerar, respectivamente, um tempo adicional d_{ij}^{min} e d_{ij}^{max}. No caso de atraso mínimo, tem-se que o início da atividade *j* só pode ocorrer após o término da atividade *i* e cumprir o tempo mínimo d_{ij}^{min}. No caso de atraso máximo, o início da atividade *j* deve ocorrer no máximo após o término de processamento da atividade *i* mais o tempo d_{ij}^{max} (SCHUTT et al., 2013).

3.4.1 Múltiplas habilidades

O RCPSP com múltiplas habilidades, que é objeto de estudo no presente trabalho, passa a considerar os seguintes conjuntos:

- S_b: quantidade total do recurso *b* que está disponível para atender as atividades;
- I_{ik}: quantidade de habilidade *k* que a atividade *i* requer;
- Q_{bk}: nível da habilidade *k* que o recurso *b* consegue atender.

O modelo para resolver o RCPSP com múltiplas habilidades parte do segundo modelo apresentado para o RCPSP, sendo também de programação linear inteira. Esse novo modelo considera um único tipo de variável binária, que é a x_{ipb}, que recebe valor 1 quando a atividade *i* inicia no tempo *p* e faz uso do recurso *b*, caso contrário, ela recebe valor zero. Além disso, quando comparado com o modelo no qual foi baseado, percebe-se que a função objetivo e as restrições (3.11), (3.13) e (3.17) são as mesmas, porém considerando o acréscimo de recursos com múltiplas habilidades, ao passo que as restrições (3.12), (3.14) e (3.15) são contribuições desta pesquisa. O terceiro modelo (isto é, Modelo 3), definido para o RCPSP com múltiplas habilidades, fica definido como:

$$\text{Minimizar } \sum_{b=1}^{r}\sum_{p\in P} px_{(n+1)pb} \tag{3.10}$$

$$\sum_{b=1}^{r}\sum_{p\in P} x_{ipb} = 1, \qquad i = 0,1,...,n+1 \tag{3.11}$$

$$\sum_{i=0}^{n+1}\sum_{p\in\{t-d_i+1\le p\le t\}} x_{ipb} \le 1, \qquad \forall t \in P, \quad b = 1,2,...,r \tag{3.12}$$

$$\sum_{b=1}^{r}\sum_{p\in P} px_{jpb} - \sum_{b=1}^{r}\sum_{p\in P} px_{ipb} \ge d_{i,} \quad \forall (i,j) \in pred \tag{3.13}$$

$$\sum_{i=0}^{n+1} \sum_{p \in P} \sum_{k=1}^{K} l_{ik} x_{ipb} \leq S_b , \qquad b = 1,2,...,r \tag{3.14}$$

$$\sum_{b=1}^{r} \sum_{p \in P} Q_{bk} x_{ipb} \geq l_{ik,} \qquad i = 0,1,...,n+1 \qquad k = 1,2,...,K \tag{3.15}$$

$$\sum_{b=1}^{r} \sum_{p \in P} p x_{(n+1)pb} \geq LI \tag{3.16}$$

$$x_{ipb} \in \{0,1\}, \qquad i = 0,1,...,n+1, \quad \forall p \in P, \quad b = 1,2,...,r \tag{3.17}$$

A função objetivo (3.10) está relacionada com uma solução de *makespan* mínimo. As restrições (3.11) indicam que cada atividade deve ser iniciada em exatamente um instante de tempo *p* e ser atendida por exatamente um recurso *b* (note que dentre todos os recursos com habilidades suficientes para suprir cada atividade, apenas um deles deve ser escolhido). As restrições em (3.12) definem que em um instante de tempo *p* e para um dado recurso *b*, no máximo uma atividade poderá requerer o recurso *b* para ser processada, sendo que este recurso deve ser capaz de atender todos os níveis de habilidade que a atividade exige. Assim, caso existam atividades sendo processadas em paralelo, elas não podem compartilhar o mesmo recurso *b*.

Nas restrições em (3.13), a relação de precedência entre as atividades é garantida, tal que a atividade *j* inicia somente após o término das atividades *i* que são suas predecessoras. As restrições (3.14) asseguram que o consumo do recurso *b* seja respeitado, ou seja, dentre todas as atividades que requerem o recurso *b*, a soma da quantidade das habilidades deve ser menor ou igual ao total disponível do recurso. Isso significa que um recurso tem capacidade limitada e é não-renovável, com isso, limitando a quantidade de atividades que o recurso consegue atender.

As restrições (3.15) garantem que o nível de habilidade *k* que a atividade *i* exige deve ser atendido por algum recurso *b*. Na restrição (3.16), impõe-se que o *makespan* seja pelo menos igual ao limitante inferior, que é calculado resolvendo um modelo relaxado sem as restrições (3.14). Nota-se que o limitante superior definido na Seção 3.3, usado para o RCPSP, deixa de ser válido na presença da restrição de múltiplas habilidades, pois agora os recursos possuem mais de uma habilidade e com níveis diferentes. Por fim, tem-se as restrições (3.17) que representam o domínio das variáveis.

O limitante inferior LI usado na restrição (3.16) para o RCPSP com múltiplas habilidades é obtido por meio da resolução do modelo relaxado com função objetivo (3.10) e restrições (3.11), (3.12), (3.13), (3.15) e (3.17).

3.4.2 Múltiplos modos

Para o RCPSP com múltiplos modos foram adicionadas restrições de acordo, considerando que as atividades podem ter vários modos de execução e, para cada modo, elas consomem uma certa quantidade dos recursos. Além disso, o modelo considera dois tipos de recursos, os renováveis e os não-renováveis. Este modelo, além de ser uma adaptação do segundo modelo para o RCPSP, segue o modelo já proposto em Mika, Rózycki e Waligóra (2012).

O quarto modelo (3.18)-(3.24) (isto é, Modelo 4), definido para o RCPSP com múltiplos modos, considera as variáveis de decisões binárias x_{imp}, que recebem valor 1 quando a atividade *i* tem modo de execução *m* e inicia no instante de tempo *p*, caso contrário, elas recebem valor zero. Além disso, tem-se os seguintes conjuntos, além dos já definidos anteriormente:

- M_i**:** modos de execução que a atividade *i* possui. As atividades fictícias *0* e *n+1* possuem apenas um modo de execução, têm duração igual a zero e não consomem recursos;
- **NR = {1, 2, ..., *nr*}:** conjunto dos *nr* recursos não-renováveis;
- NR_b**:** disponibilidade total do recurso *b*, que é não-renovável;
- C_{imb}**:** quantidade do recurso *b* que a atividade *i* consome durante seu processamento no modo de execução *m*;
- d_{im}**:** duração da atividade *i* no modo de execução *m*.

Minimizar $\sum_{m\in M_i}\sum_{p\in P} px_{(n+1)mp}$ (3.18)

$$\sum_{m\in M_i}\sum_{p\in P} x_{imp} = 1, \qquad i = 0,1,...,n+1 \tag{3.19}$$

$$\sum_{i=0}^{n+1}\sum_{m\in M_i}\sum_{p\in\{t-d_i+1\le p\le t\}} C_{imb}x_{imp} \le R_b, \qquad \forall t \in P, \qquad b = 1,2,...,r \tag{3.20}$$

$$\sum_{i=0}^{n+1}\sum_{m\in M_i}\sum_{p\in P} C_{imb}x_{imp} \le NR_b, \qquad b = 1,2,...,rn \tag{3.21}$$

$$\sum_{m\in M_j}\sum_{p\in P} px_{jmp} - \sum_{m\in M_i}\sum_{p\in P}(p+d_{im})x_{imp} \ge 0, \qquad \forall(i,j)\in pred \tag{3.22}$$

$$\sum_{m\in M_i}\sum_{p\in P} px_{(n+1)mp} \ge LI \tag{3.23}$$

$$x_{imp} \in \{0,1\}, \qquad i = 0,1,...,n+1, \quad \forall m \in M_i, \quad \forall p \in P \tag{3.24}$$

A função objetivo (3.18) calcula o *makespan*, que se deseja seja o mínimo possível. As restrições em (3.19) impõem que cada atividade deve ser iniciada em exatamente um instante de tempo *p* e processada em um único modo *m*. As restrições em (3.20) garantem que o consumo de recurso pelas atividades executadas no modo *m* em cada instante de tempo respeite a disponibilidade de cada recurso. Nas restrições (3.21), impõem-se que a quantidade total consumida, considerando todo o horizonte de tempo, de cada recurso não-renovável seja respeitada.

As restrições em (3.22) garantem a relação de precedência entre as atividades, de tal forma que a atividade *j*, executada em um de seus modos M_j, deve iniciar somente após o término da atividade *i,* executada em um de seus modos M_i. A restrição (3.23) assegura que o *makespan* seja pelo menos igual ao limitante inferior LI, que é obtido pela resolução de um modelo relaxado. Por último, têm-se as restrições (3.24) que apresentam o domínio das variáveis.

O limitante inferior para o RCPSP com múltiplos modos é também obtido pela resolução de um modelo relaxado, sem as restrições de consumo de recursos renováveis (3.20) e não-renováveis (3.21). Neste caso, o modelo relaxado tem função objetivo (3.18) e restrições (3.19), (3.22) e (3.24).

3.4.3 Tempos de atraso: mínimo e máximo

Para o RCPSP com tempos de atraso mínimo e máximo com relação a execução das atividades, tem-se restrições que consideram uma janela de tempo que define o instante de tempo mínimo e máximo para o início de uma atividade *j* dado o término de uma atividade *i*. Para a definição do modelo de programação linear inteira para essa variante do RCPSP, tem-se os seguintes parâmetros/conjuntos:

- ***lag*:** conjunto de pares ordenados *(i, j)* de atividades que possuem relações de atraso mínimo e máximo;
- $\boldsymbol{d_{ij}^{min}}$**:** duração mínima para a preparação (isto é, atraso) após o término da atividade *i* para que a atividade *j* possa ser iniciada, para os pares de atividades *(i, j)* em *lag*;
- $\boldsymbol{d_{ij}^{max}}$**:** máximo instante após o término da atividade *i* para que a atividade *j* seja iniciada, para os pares de atividades *(i, j)* em *lag*;

O quinto modelo (3.25)-(3.32) (isto é, Modelo 5), definido para o RCPSP com tempos de atraso, parte do segundo modelo para o RCPSP. Sendo assim, ele considera as

variáveis de decisões binárias x_{ip}, que recebem valor 1 quando a atividade *i* inicia no instante de tempo *p*, caso contrário, tem-se o valor zero. Vale mencionar que a ideia para formular as restrições que representam os tempos de atraso foram obtidas de Hartmann e Briskorn (2010), que apresentam uma discussão mais detalhada dessa variante do RCPSP.

Minimizar $\sum_{p\in P} p x_{(n+1)p}$ (3.25)

$$\sum_{p\in P} x_{ip} = 1, \qquad i = 0,1,...,n+1 \tag{3.26}$$

$$\sum_{i=0}^{n+1} \sum_{p\in\{t-d_i+1\le p\le t\}} C_{ib} x_{ip} \le R_b, \qquad \forall t \in P, \qquad b = 1,2...,r \tag{3.27}$$

$$\sum_{p\in P} p x_{jp} - \sum_{p\in P} p x_{ip} \ge d_i, \quad \forall (i,j) \in pred \tag{3.28}$$

$$\sum_{p\in P} p x_{jp} \ge \sum_{p\in P} x_{ip} + d_i + d_{ij}^{\min}, \qquad \forall (i,j) \in lag \tag{3.29}$$

$$\sum_{p\in P} p x_{jp} \le \sum_{p\in P} x_{ip} + d_i + d_{ij}^{\max}, \qquad \forall (i,j) \in lag \tag{3.30}$$

$$\sum_{b\in r} \sum_{p\in P} p x_{(n+1)p} \ge LI \tag{3.31}$$

$$x_{ip} \in \{0,1\}, \qquad i = 0,1,...,n+1, \qquad \forall p \in P \tag{3.32}$$

A função objetivo (3.25) está associada com a minimização do *makespan*, que é medido pelo término da última atividade do projeto. As restrições (3.26) garantem que toda atividade deve ser iniciada em exatamente um instante de tempo *p*, com a possibilidade de atividades em paralelo. As restrições (3.27) garantem que a quantidade de cada recurso consumida e em cada instante de tempo respeite a quantidade disponível do referido recurso. As relações de precedência entre as atividades são satisfeitas por meio das restrições (3.28), de forma que uma atividade *j* só pode iniciar após o processamento de todas as suas predecessoras *i*.

As restrições (3.29) e (3.30) impõem que os tempos de atraso sejam atendidos entre as atividades, isto é, nas restrições (3.29), tem-se o tempo mínimo que a atividade *j* deve esperar para iniciar após o término do processamento da atividade *i*. Por outro lado, em (3.30), tem-se máximo instante de tempo permitido, após a conclusão da atividade *i*, para que haja o início da atividade *j*. A restrição em (3.31) define que o *makespan* deve ser pelo

menos igual a um limitante inferior LI. Finalmente, as restrições (3.32) indicam o domínio das variáveis.

O limitante inferior para o RCPSP com tempos de atraso mínimo e máximo é alcançado pela resolução do modelo relaxado, que não inclui as restrições de consumo de recursos, isto é, resolve-se o modelo relaxado com função objetivo (3.25) e restrições (3.26), (3.28), (3.29), (3.30) e (3.32).

CAPÍTULO 4

RESULTADOS COMPUTACIONAIS

Os modelos de programação matemática para o RCPSP apresentados no capítulo anterior foram codificados na linguagem C++. Além disso, utilizou-se o pacote de otimização *Gurobi Optimizer* na versão 8.0 para a resolução dos modelos, que se deu por um método exato, mais especificamente, o *branch-and-cut*, para tentar obter uma solução ótima. O computador adotado para os experimentos possui sistema operacional Ubuntu 16.04 LTS, com processador Intel Core i5 de 3,4 GHz e 8 GB de memória RAM.

A linguagem de programação C++ foi adotada, pois já se tinha um domínio básico da mesma e utilizou-se o *Gurobi Optimizer* por ele já ter uma implementação eficiente do método *branch-and-cut*. O computador utilizado possui configuração adequada e correspondente ao que a literatura tem usado, além de serem os disponíveis em laboratório para realizar os experimentos. Além disso, todas as instâncias usadas nessa pesquisa estão disponíveis em *https://taq.catalao.ufg.br/p/17980-cursos-ministrados-e-instancias*.

4.1 Resultados para o RCPSP

Os testes considerando os dois primeiros modelos desenvolvidos para o RCPSP sem qualquer restrição prática ocorreram sobre instâncias da literatura da base PSPLIB[1], descrita por Kolisch e Sprecher (1997). Utilizaram-se dois conjuntos, J30 e J60, para um total de 960 instâncias. As instâncias do conjunto J30 possuem 32 atividades cada, sendo a

[1] http://www.om-db.wi.tum.de/psplib/

primeira e a última atividade fictícias; quatro recursos; tempo de processamento das atividades; quantidade de cada recurso que cada uma das atividades consome; e, a relação de precedência entre as atividades. No conjunto J60, a única diferença para o J30 diz respeito a quantidade de atividades, que agora é de 62 por instância, incluindo as duas fictícias. Cada instância teve um tempo de limite de 3.600 segundos para ser resolvida por cada modelo, sendo que a melhor solução encontrada dentro deste tempo é reportada nas Tabelas 4.1 e 4.2. Considera-se esse tempo limite por estar de acordo com os experimentos reportados na literatura (MINGOZZI et al., 1998).

Na Tabela 4.1 estão os resultados das 480 instâncias do conjunto J30 para o primeiro e o segundo modelo, isto é, Modelo 1, definido em (3.1)-(3.7) e Modelo 2, definido em (3.2), (3.3) e (3.6)-(3.9). Os resultados estão organizados por grupos de 10 instâncias cada, ou seja, a instância J301 refere-se a um grupo com 10 instâncias, tal que os valores apresentados são a média do referido grupo.

Os dados em cada linha da Tabela 4.1 são: nome do grupo; média do limitante inferior (LI), obtida da resolução do respectivo modelo relaxado pelo *Gurobi*; média do limitante superior (LS), calculado pela heurística *List Scheduling*; número de instâncias do grupo resolvidas de forma ótima; média do valor de solução, isto é, do *makespan*; média do tempo de otimização para resolver somente o modelo do RCPSP (ou seja, não está incluso o tempo para obter o limitante inferior) em segundos; e, a média do *gap* (que é calculado pelo *Gurobi* e corresponde a diferença percentual entre o limitante superior e o inferior, ambos calculados pelo próprio *Gurobi*), em porcentagem. Quando o *gap* é zero, então a solução é ótima, além disso, as médias reportadas consideram as 10 instâncias do grupo. No final da tabela consta a média, o desvio padrão, o mínimo e o máximo, para todos os grupos.

Conforme os resultados na Tabela 4.1, nota-se que das 480 instâncias, 460 foram resolvidas na otimalidade para o Modelo 1, isto é, 95,8%, e 462 para o Modelo 2, que equivale a 96,3%, ou seja, para essas instâncias o tempo de otimização foi inferior aos 3.600 segundos e o *gap* foi de 0,0%, sendo destacadas em negrito na coluna “GAP” da tabela. Ao considerar a quantidade de grupos que tiveram todas as instâncias resolvidas na otimalidade, com o Modelo 1 foi possível resolver 42 grupos totalmente, ao passo que com o Modelo 2 este número passou para 44. Para os demais conjuntos de instâncias, o tempo de otimização atingiu os 3.600 segundos para pelo menos uma instância, com o *gap* diferente de 0,0%, resultando em uma solução viável, mas não ótima.

Com relação ao tempo computacional dos modelos na Tabela 4.1, nota-se que o tempo limite não foi atingido para nenhum grupo de instância. Com relação ao Modelo 1, o tempo médio foi de 201,84 segundos, ao passo que no Modelo 2, o tempo médio foi de 214,41 segundos. Nota-se que o aumento no tempo médio do Modelo 2 em comparação

com o Modelo 1 foi influenciado pelos grupos de instâncias que exigiram mais tempo para serem resolvidos (por exemplo, grupos J309, J3013, J3025, J3029, J3041 e J3045). Isso pode ter ocorrido por alguma dificuldade que o Modelo 2 teve (por exemplo, gerar uma relaxação linear mais fraca) para resolver essas instâncias em comparação com o Modelo 1.

Tabela 4.1 – Resultados para o conjunto J30 obtidos ao resolver os Modelos 1 e 2.

Grupo	Modelo 1						Modelo 2					
	LI	LS	QTD.	Sol.	Tempo (s)	GAP (%)	LI	LS	QTD.	Sol.	Tempo (s)	GAP (%)
J301	43,9	65,1	10	49,3	0,7	**0,0**	43,6	65,1	10	49,3	0,5	**0,0**
J302	45,2	60,7	10	47,1	0,4	**0,0**	45,2	60,7	10	47,1	0,2	**0,0**
J303	58,4	75,1	10	60,0	0,5	**0,0**	58,4	75,1	10	60,0	0,1	**0,0**
J304	51,4	69,0	10	51,4	0,3	**0,0**	51,4	69,0	10	51,4	0,1	**0,0**
J305	48,9	81,0	10	67,6	21,5	**0,0**	48,9	81,0	10	67,6	41,1	**0,0**
J306	46,5	68,1	10	50,1	1,1	**0,0**	46,5	68,1	10	50,1	1,5	**0,0**
J307	44,8	63,3	10	46,5	0,5	**0,0**	44,8	63,3	10	46,5	0,3	**0,0**
J308	49,9	62,7	10	49,9	0,3	**0,0**	49,9	62,7	10	49,9	0,1	**0,0**
J309	49,5	94,3	9	74,8	650,5	0,2	49,5	94,3	10	74,8	800,8	**0,0**
J3010	47,1	63,9	9	49,6	1,8	**0,0**	47,1	63,9	10	49,6	2,8	**0,0**
J3011	54,3	70,1	10	55,0	0,7	**0,0**	54,3	70,1	10	55,0	0,6	**0,0**
J3012	49,2	66,7	10	49,2	0,4	**0,0**	49,2	66,7	10	49,2	0,2	**0,0**
J3013	44,9	88,9	3	73,2	2.675,7	6,5	44,9	88,9	3	73,7	2.960,9	8,7
J3014	47,7	66,0	10	50,9	17,3	**0,0**	47,7	66,0	10	50,9	47,2	**0,0**
J3015	52,4	67,5	10	53,0	1,0	**0,0**	52,4	67,5	10	53,0	1,1	**0,0**
J3016	45,7	61,4	10	45,7	0,4	**0,0**	45,7	61,4	10	45,7	0,3	**0,0**
J3017	51,7	73,8	10	58,3	1,0	**0,0**	51,7	73,8	10	58,3	0,9	**0,0**
J3018	50,2	71,2	10	54,4	0,6	**0,0**	50,2	71,2	10	51,1	0,2	**0,0**
J3019	50,1	66,6	10	51,4	0,4	**0,0**	50,1	66,6	10	51,4	0,1	**0,0**
J3020	50,2	63,7	10	50,2	0,3	**0,0**	50,2	63,7	10	50,2	0,1	**0,0**
J3021	50,8	84,7	10	68,5	14,4	**0,0**	50,8	84,7	10	68,5	16,9	**0,0**
J3022	50,3	71,9	10	54,2	1,4	**0,0**	50,3	71,9	10	54,2	1,4	**0,0**
J3023	54,5	72,0	10	55,9	0,6	**0,0**	54,5	72,0	10	55,9	0,3	**0,0**
J3024	51,8	68,0	10	51,8	0,4	**0,0**	51,8	68,0	10	51,8	0,1	**0,0**
J3025	47,8	93,4	10	76,1	689,3	**0,0**	47,8	93,4	10	76,1	891,8	**0,0**
J3026	53,9	69,9	10	56,0	1,0	**0,0**	53,9	69,9	10	56,0	1,4	**0,0**
J3027	56,2	75,7	10	56,5	0,8	**0,0**	56,2	75,7	10	56,5	0,4	**0,0**
J3028	56,5	69,5	10	56,5	0,5	**0,0**	56,5	69,5	10	56,5	0,2	**0,0**
J3029	50,6	105,4	3	89,9	2.784,9	8,6	50,6	105,4	3	88,5	2.799,5	7,1

(continua)

(conclusão)

Grupo	Modelo 1						Modelo 2					
	LI	LS	QTD.	Sol.	Tempo (s)	GAP (%)	LI	LS	QTD.	Sol.	Tempo (s)	GAP (%)
J3030	50,4	70,6	10	55,2	5,7	**0,0**	50,4	70,6	10	51,9	13,1	**0,0**
J3031	52,9	69,7	10	54,3	1,9	**0,0**	52,9	69,6	10	54,3	2,1	**0,0**
J3032	54,9	71,4	10	54,9	0,6	**0,0**	54,9	71,4	10	54,9	0,4	**0,0**
J3033	53,6	74,0	10	60,6	1,1	**0,0**	53,6	74,0	10	59,3	0,5	**0,0**
J3034	55,7	71,0	10	58,6	0,5	**0,0**	55,7	71,0	10	58,6	0,1	**0,0**
J3035	56,9	72,4	10	58,0	0,6	**0,0**	56,9	72,4	10	58,0	0,1	**0,0**
J3036	57,7	71,0	10	57,7	0,4	**0,0**	57,7	71,0	10	57,7	0,1	**0,0**
J3037	53,5	93,3	10	76,7	96,5	**0,0**	53,5	93,3	10	76,7	109,0	**0,0**
J3038	55,9	76,3	10	60,3	1,0	**0,0**	55,9	76,3	10	59,6	0,7	**0,0**
J3039	56,2	70,3	10	58,7	0,6	**0,0**	56,2	70,3	10	58,7	0,2	**0,0**
J3040	56,4	69,7	10	56,4	0,5	**0,0**	56,4	69,7	10	56,4	0,1	**0,0**
J3041	57,2	109,2	9	91,2	989,2	0,4	57,2	109,2	9	91,1	1.101,6	0,1
J3042	57,7	74,6	10	61,8	2,3	**0,0**	57,7	74,6	10	61,8	2,8	**0,0**
J3043	56,5	75,5	10	57,5	1,1	**0,0**	56,5	75,5	10	57,5	0,9	**0,0**
J3044	54,1	69,4	10	54,1	0,5	**0,0**	54,1	69,4	10	54,1	0,2	**0,0**
J3045	60,5	115,1	7	97,3	1.695,5	3,0	60,5	115,1	7	96,6	1.473,2	1,4
J3046	55,1	71,5	10	59,2	19,7	**0,0**	55,1	71,5	10	59,2	14,3	**0,0**
J3047	54,4	70,5	10	56,0	1,1	**0,0**	54,4	70,5	10	55,9	0,7	**0,0**
J3048	55,2	68,2	10	55,2	0,6	**0,0**	55,2	68,2	10	55,2	0,2	**0,0**
Média	52,3	74,4	9,6	59,1	201,8	0,4	52,3	74,4	9,6	58,9	214,4	0,4
Desvio Padrão	4,1	12,1	1,5	11,4	614,1	1,6	4,1	12,1	1,5	11,4	639,7	1,6
Mínimo	43,9	60,7	3,0	45,7	0,3	0,0	43,6	60,7	3,0	45,7	0,1	0,0
Máximo	60,5	115,1	10,0	97,3	2784,9	8,6	60,5	115,1	10,0	96,6	2960,9	8,7

As Figuras 4.1, 4.2 e 4.3 sumarizam os resultados apresentados na Tabela 4.1 com relação aos Modelos 1 e 2 para o conjunto J30.

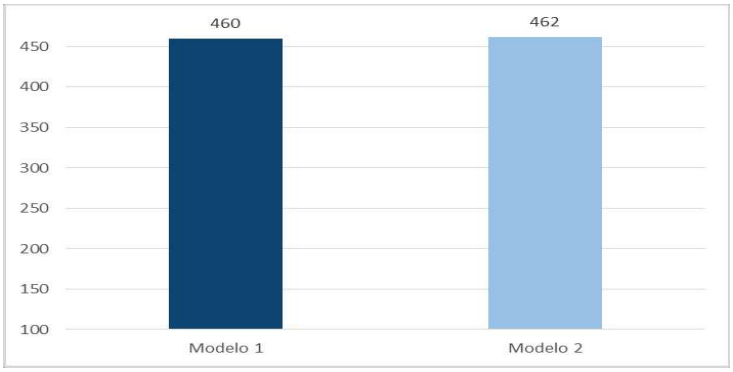

Figura 4.1 – Quantidade de instâncias resolvidas na otimalidade para o conjunto J30.

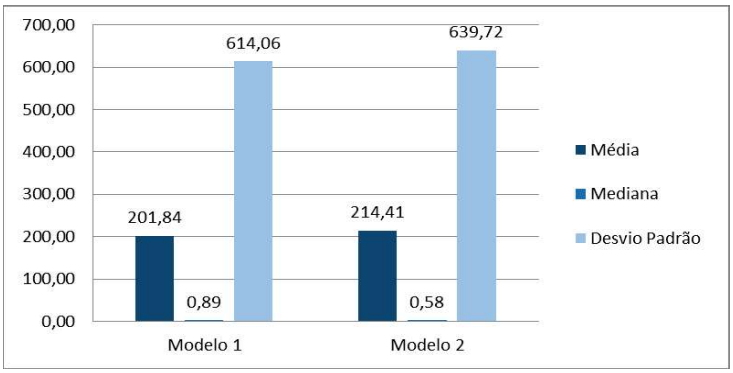

Figura 4.2 – Tempo médio de otimização (em segundos) para o conjunto J30.

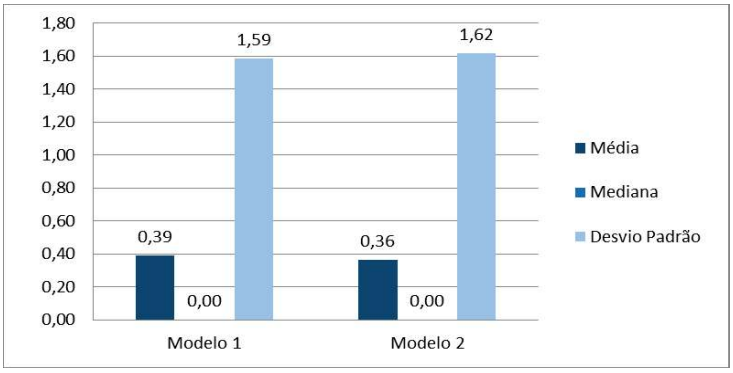

Figura 4.3 – Valor do *gap* médio (em porcentagem) para o conjunto J30.

A Figura 4.1 mostra que de 480 instâncias do conjunto J30, o Modelo 1 resolveu 460 de forma ótima, ao passo que o Modelo 2 resolveu 462 delas. A Figura 4.2 apresenta o tempo médio geral de otimização, isto é, calculado sobre as 480 instâncias, para ambos os

modelos, de forma que o Modelo 1 precisou de 201,84 segundos e o do Modelo 2 de 214,41 segundos. A Figura 4.3 exibe o GAP médio geral, sendo de 0,39% para o Modelo 1 e de 0,36% para o Modelo 2. Para todos os dados analisados nessas figuras foram considerados, além da média, também a mediana e o desvio padrão. Portanto, conclui-se que o Modelo 2 resolveu mais instâncias, com um menor *gap* médio geral, embora o desvio padrão tenha sido ligeiramente maior e o tempo computacional médio também foi ligeiramente maior.

A Tabela 4.2 contém os resultados dos Modelos 1 e 2 sobre as 480 instâncias do conjunto J60. Os valores apresentados nessa tabela seguem a mesma explicação da Tabela 4.1. Portanto, das 480 instâncias, 386 foram resolvidas na otimalidade pelo Modelo 1, ou seja, 80,4%, e 389 pelo Modelo 2, que corresponde a 81,0%. Além disso, 42 dos 48 grupos tiveram pelo menos uma instância resolvida na otimalidade pelos Modelos 1 e 2. Com relação ao *gap* médio, o do Modelo 1 foi de 3,79%, enquanto o do Modelo 2 foi de 2,35%. No Modelo 1, o pior *gap* ocorreu para o grupo J6045, sendo de 36,3%. No caso do Modelo 2, o pior *gap* foi para o grupo J6025, sendo de 20,2%. Nota-se que, por envolver mais atividades no conjunto J60, os modelos encontraram maior dificuldade em resolver as instâncias de forma ótima dentro do tempo limite ao comparar com o conjunto J30.

Tabela 4.2 – Resultados para o conjunto J60 obtidos ao resolver os Modelos 1 e 2.

Grupo	Modelo 1						Modelo 2					
	LI	LS	QTD.	Sol.	Tempo (s)	GAP (%)	LI	LS	QTD.	Sol.	Tempo (s)	GAP (%)
J601	69,0	102,2	10	75,5	6,8	**0,0**	69,0	102,2	10	75,5	2,8	**0,0**
J602	66,0	91,1	10	67,4	1,9	**0,0**	66,0	91,1	10	67,4	0,4	**0,0**
J603	69,2	95,2	10	70,5	2,0	**0,0**	69,2	95,2	10	70,5	0,3	**0,0**
J604	69,7	92,7	10	69,7	1,5	**0,0**	69,7	92,7	10	69,7	0,1	**0,0**
J605	65,6	111,0	3	85,0	2.603,9	5,9	65,6	111,0	3	85,4	2.585,1	6,5
J606	66,6	91,4	10	67,0	3,5	**0,0**	66,6	91,4	10	68,8	1,5	**0,0**
J607	70,4	101,8	10	70,4	2,9	**0,0**	70,4	101,8	10	70,4	0,5	**0,0**
J608	69,0	96,3	10	69,0	2,1	**0,0**	69,0	96,3	10	69,0	0,4	**0,0**
J609	68,6	122,6	0	108,6	3.600,0	19,5	68,6	122,6	0	108,1	3.600,0	19,4
J6010	72,4	99,6	10	72,5	4,4	**0,0**	72,4	99,6	10	72,5	1,7	**0,0**
J6011	67,1	91,1	10	67,1	2,3	**0,0**	67,1	91,1	10	67,1	0,7	**0,0**
J6012	64,7	90,9	10	64,7	2,1	**0,0**	64,7	90,9	10	64,7	0,5	**0,0**
J6013	61,7	130,1	0	124,2	3.600,0	20,9	60,3	128,7	0	116,0	3.600,0	12,4
J6014	65,9	96,6	7	67,0	1.091,1	1,0	65,9	96,6	7	67,0	1.085,3	1,0
J6015	74,2	98,5	10	74,2	2,9	**0,0**	74,2	98,5	10	74,2	0,9	**0,0**
J6016	64,5	94,5	10	64,5	2,6	**0,0**	64,5	94,5	10	61,5	0,8	**0,0**

(continua)

(conclusão)

Grupo	Modelo 1						Modelo 2					
	LI	LS	QTD.	Sol.	Tempo (s)	GAP (%)	LI	LS	QTD.	Sol.	Tempo (s)	GAP (%)
J6017	69,2	103,4	10	75,9	42,3	**0,0**	69,2	103,4	10	75,9	8,3	**0,0**
J6018	76,9	101,0	10	77,6	2,9	**0,0**	76,9	101,0	10	77,6	0,3	**0,0**
J6019	72,1	103,6	10	73,1	3,1	**0,0**	72,1	103,6	10	73,1	0,4	**0,0**
J6020	74,4	105,6	10	74,4	2,7	**0,0**	74,4	105,6	10	74,4	0,2	**0,0**
J6021	72,0	124,4	5	97,4	2.845,1	4,6	72,0	124,4	7	93,0	2.609,0	3,8
J6022	69,6	102,5	10	70,8	9,4	**0,0**	69,6	102,5	10	70,8	5,8	**0,0**
J6023	71,9	102,5	10	72,2	3,4	**0,0**	71,9	102,5	10	72,2	0,5	**0,0**
J6024	71,1	101,8	10	71,1	2,8	**0,0**	71,1	101,8	10	71,1	0,4	**0,0**
J6025	71,3	135,7	0	126,0	3.600,0	29,5	71,9	138,6	0	120,2	3.600,0	20,2
J6026	72,1	100,9	10	73,5	48,0	**0,0**	72,1	100,9	10	73,5	44,0	**0,0**
J6027	75,2	101,3	10	75,2	3,3	**0,0**	75,2	101,3	10	75,2	0,8	**0,0**
J6028	75,7	103,2	10	75,7	3,1	**0,0**	75,7	103,2	10	75,7	0,7	**0,0**
J6029	72,7	146,7	0	144,4	3.600,0	24,0	74,7	151,5	0	141,3	3.600,0	14,6
J6030	75,0	103,5	6	78,4	1.450,6	2,5	75,0	103,5	6	78,9	1.446,1	3,1
J6031	71,0	98,9	10	71,0	3,9	**0,0**	71,0	98,9	10	71,0	1,3	**0,0**
J6032	79,7	108,7	10	79,7	3,9	**0,0**	79,7	108,7	10	79,7	1,0	**0,0**
J6033	81,5	116,6	10	89,7	16,9	**0,0**	81,5	116,6	10	89,7	3,6	**0,0**
J6034	74,6	103,2	10	76,2	4,0	**0,0**	74,6	103,2	10	76,2	0,5	**0,0**
J6035	75,9	111,4	10	76,9	4,8	**0,0**	75,9	111,4	10	76,9	0,4	**0,0**
J6036	73,8	99,8	10	73,8	2,5	**0,0**	73,8	99,8	10	73,8	0,2	**0,0**
J6037	74,7	122,2	3	104,3	2.810,9	7,7	77,1	129,7	4	103,4	2.369,3	4,4
J6038	73,8	106,5	10	75,0	58,1	**0,0**	73,8	106,5	10	75,0	91,1	**0,0**
J6039	78,6	105,4	10	78,7	4,0	**0,0**	78,6	105,4	10	78,7	0,6	**0,0**
J6040	78,7	108,2	10	78,7	3,4	**0,0**	78,7	108,2	10	78,7	0,4	**0,0**
J6041	79,6	158,6	0	154,1	3.600,0	26,4	80,3	155,9	0	135,9	3.600,0	15,0
J6042	76,3	104,5	9	78,7	409,1	0,4	76,3	104,5	9	78,8	398,2	0,6
J6043	79,5	105,6	10	79,6	4,6	**0,0**	79,5	105,6	10	79,6	1,0	**0,0**
J6044	76,0	101,5	10	76,0	3,2	**0,0**	76,0	101,5	10	76,0	0,6	**0,0**
J6045	69,9	150,7	0	148,4	3.600,0	36,3	66,0	141,0	0	136,7	3.600,0	8,4
J6046	76,1	110,4	3	81,6	2.549,6	3,3	76,1	110,4	3	81,6	2.550,1	3,5
J6047	73,0	106,4	10	73,0	7,3	**0,0**	73,0	106,4	10	73,0	1,6	**0,0**
J6048	78,8	103,1	10	78,8	3,5	**0,0**	78,8	103,1	10	78,8	0,9	**0,0**
Média	72,4	107,6	8,0	82,9	742,4	3,8	72,4	107,6	8,1	81,8	725,4	2,4
Desvio Padrão	4,6	15,1	3,6	21,7	1.337,3	8,9	4,8	15,0	3,6	18,9	1.317,2	5,2
Mínimo	61,7	90,9	0,0	64,5	1,5	**0,0**	60,3	90,9	0,0	61,5	0,1	**0,0**
Máximo	81,5	158,6	10,0	154,1	3.600,0	36,3	81,5	155,9	10,0	141,3	3.600,0	20,2

Com relação ao tempo computacional dos modelos na Tabela 4.2, percebe-se que o tempo limite foi atingido para todas as instâncias dos grupos J609, J6013, J6025, J6029, J6041 e J6045, tanto para o Modelo 1 quanto para o Modelo 2. O tempo médio para o Modelo 1 foi de 742,42 segundos, enquanto que para o Modelo 2, o tempo médio foi de 725,39 segundos. As Figuras 4.4, 4.5 e 4.6 trazem um resumo dos resultados na Tabela 4.2, em particular, a quantidade de instâncias resolvidas de forma ótima, o tempo médio geral e o *gap* médio geral, para o conjunto J60.

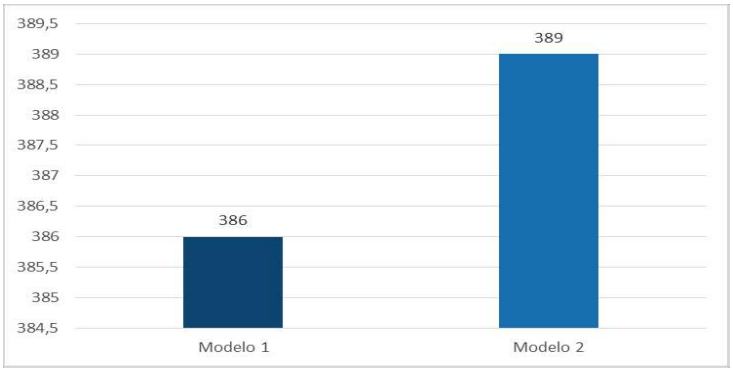

Figura 4.4 – Quantidade de instâncias resolvidas na otimalidade para o conjunto J60.

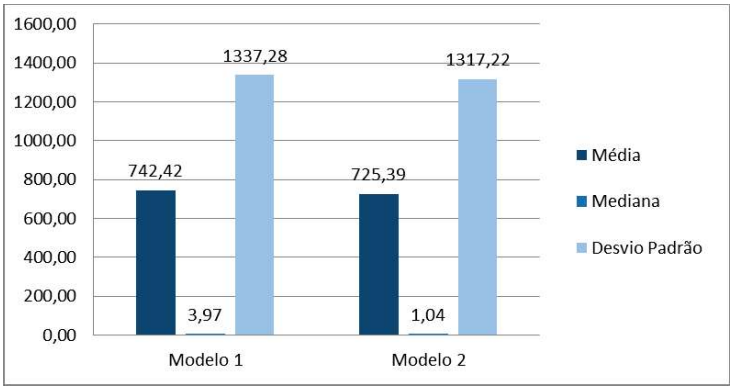

Figura 4.5 – Tempo médio de otimização (em segundos) para o conjunto J60.

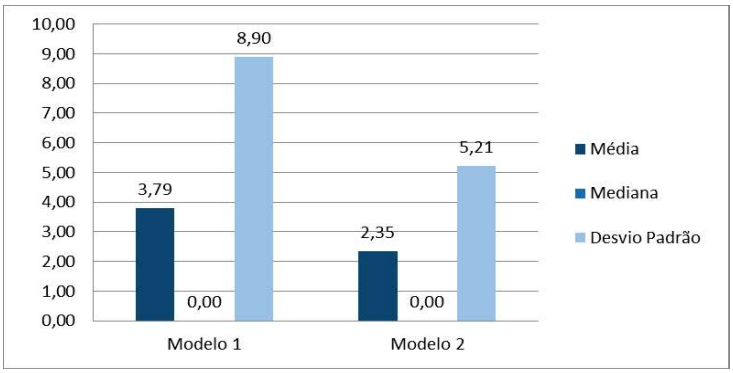

Figura 4.6 – Valor do *gap* médio (em porcentagem) para o conjunto J60.

A Figura 4.4 mostra que das 480 instâncias do conjunto J60, o Modelo 1 foi capaz de resolver 386 de forma ótima dentro do tempo limite de 3.600 segundos, enquanto o Modelo 2 conseguiu resolver 389 instâncias, três a mais. A Figura 4.5 apresenta o tempo médio geral de otimização para ambos os modelos, sendo o do Modelo 1 de 742,42 segundos e do Modelo 2 de 725,39 segundos. A Figura 4.6 exibe o *gap* médio geral, sendo de 3,79% para o Modelo 1 e de 2,35% para o Modelo 2.

Mais uma vez, conclui-se que o Modelo 2 pode ser melhor do que o Modelo 1, conseguindo resolver mais instâncias de forma ótima e agora em um menor tempo computacional médio. Por isso, o Modelo 2 foi usado como base para adição da restrição de múltiplas habilidades, múltiplos modos e de tempos de atraso mínimo e máximo no RCPSP. No entanto, é importante destacar que esta conclusão poderia ser corroborada por meio de testes estatísticos para verificar se há significância entre os modelos, sendo uma proposta de trabalho futuro.

4.2 Resultados para o RCPSP com múltiplas habilidades

Os testes realizados sobre o modelo que resolve o RCPSP com múltiplas habilidades, ocorreram sobre instâncias da base iMOPSE[2] (*Intelligent Multi Objective Project Scheduling Environment*) proposta por Myszkowski et al. (2018).

Utilizaram-se um total de 36 instâncias para os testes. A quantidade de atividades presente nas instâncias é de 12, 17, 102 ou 202, sendo a primeira e a última atividade fictícias; a quantidade de recursos é de 3, 5, 6, 7, 9, 10 ou 20; a quantidade de tipos de habilidades é de 3, 5, 6, 7, 9 ou 15. Além dessas informações, tem-se também o tempo de

[2] http://imopse.ii.pwr.wroc.pl/download.html

processamento das atividades; a relação de precedência entre as atividades; o nível de habilidade que o recurso consegue atender; a quantidade da habilidade que a atividade requer; e, por fim, a quantidade total de cada recurso para atender as atividades.

Cada instância teve um tempo de limite de 3.600 segundos para ser resolvida pelo Modelo 3, definido em (3.10)-(3.17), sendo que a melhor solução encontrada dentro deste tempo é reportada na Tabela 4.3. Nota-se que algumas instâncias agora envolvem maior número de atividades e recursos, o que pode inviabilizar a obtenção de alguma solução pelo *Gurobi* ao resolver o modelo, dado o tempo limite.

A Tabela 4.3 contém os resultados dessas 36 instâncias, que precisaram ser adaptadas para o Modelo 3 para considerar mais de uma habilidade por recurso e níveis de habilidade diferentes para as atividades. Os dados em cada linha da tabela são: nome da instância, limitante inferior (LI) obtido ao resolver o modelo relaxado pelo *Gurobi*; valor da solução retornado pelo *Gurobi*, que é o *makespan*; tempo de otimização em segundos (não está incluso o tempo para obter o limitante inferior); e, o valor do *gap* em porcentagem, retornado pelo *Gurobi*. No final da tabela há a média, o desvio padrão, o mínimo e o máximo dos dados como um todo.

Tabela 4.3 – Resultados das instâncias do RCPSP com múltiplas habilidades.

Instância	Modelo 3			
	LI	Sol.	Tempo (s)	GAP (%)
10_3_5_3	93,0	93,0	0,5	**0,0**
10_5_8_5	80,0	80,0	0,5	**0,0**
10_7_10_7	104,0	104,0	0,8	**0,0**
15_3_5_3	230,0	230,0	1,5	**0,0**
15_6_10_6	102,0	102,0	2,0	**0,0**
15_9_12_9	90,0	90,0	4,3	**0,0**
100_5_20_9_D3	389,0	389,0	857,5	**0,0**
100_5_22_15	493,0	493,0	1.466,2	**0,0**
100_5_46_15	2566,0	2.566,0	188,7	**0,0**
100_5_48_9	498,0	498,0	1.690,3	**0,0**
100_5_64_9	492,0	492,0	1.210,9	**0,0**
100_10_26_15	2.253,0	2.253,0	1.927,0	**0,0**
100_5_64_15	-	-	3.600,0	**-**
100_10_27_9_D2	-	-	3.600,0	**-**
100_10_47_9	304,0	304,0	3.323,6	**0,0**

(continua)

(conclusão)

Instância	Modelo 3			
	LI	Sol.	Tempo (s)	GAP (%)
100_10_48_15	244,0	2.395,0	3.600,0	89,0
100_10_64_9	255,0	-	3.600,0	-
100_10_65_15	-	-	3.600,0	-
100_20_22_15	156,0	156,0	2.303,4	**0,0**
100_20_23_9_D1	172,0	172,0	1.226,4	**0,0**
100_20_46_15	2.612,0	2.612,0	501,5	**0,0**
100_20_47_9	-	-	3.600,0	-
100_20_65_9	-	-	3.600,0	-
100_20_65_15	2.401,0	2.401,0	543,3	**0,0**
200_10_50_9	4.809,0	4.809,0	838,3	**0,0**
200_10_50_15	-	-	3.600,0	-
200_10_84_9	-	-	3.600,0	-
200_10_85_15	-	-	3.600,0	-
200_10_128_15	-	-	3.600,0	-
200_10_135_9_D6	-	-	3.600,0	-
200_20_54_15	-	-	3.600,0	-
200_20_55_9	-	-	3.600,0	-
200_20_97_9	-	-	3.600,0	-
200_20_97_15	-	-	3.600,0	-
200_20_145_15	-	-	3.600,0	-
200_20_150_9_D5	-	-	3.600,0	-
Média	917,2	1.065,2	2.246,9	4,7
Desvio Padrão	1.293,8	1.347,4	1.526,9	20,4
Mínimo	80,0	80,0	0,5	**0,0**
Máximo	4.809,0	4.809,0	3.600,0	89,0

(-) A instância não foi resolvida, ou seja, o Gurobi não conseguiu encontrar qualquer solução viável dentro do tempo limite de 3.600 segundos.

Observando a Tabela 4.3, das 36 instâncias, 18 foram resolvidas de forma ótima, o que corresponde a 50,0%. Essas instâncias são as com até 102 atividades, sendo apenas uma de 202 atividades (incluindo as duas fictícias). Apenas para uma das instâncias com 102 atividades e 5 tipos de recursos (isto é, instância 100_5_64_15), que não foi possível retornar qualquer solução viável, notando que esta instância possui 64 relações de precedência e 15 tipos de habilidade. Para as demais instâncias com 202 atividades, não foi possível obter qualquer solução viável dado o tempo limite considerado.

Com relação ao limitante inferior na Tabela 4.3, obtido ao resolver o modelo relaxado associado ao Modelo 3, nota-se que ele coincidiu com a solução ótima retornada pelo *Gurobi* ao resolver o Modelo 3, indicando que os limitantes encontrados ao resolver o modelo relaxado foram justos. O tempo computacional médio considerando as 18 instâncias com solução apresentada foi de 893,7 segundos, mas houve uma instância com tempo superior aos 3.000 segundos (que é a 100_10_47_9). Portanto, observa-se que o Modelo 3 apresentou dificuldades para resolver as instâncias maiores, sendo recomendado para quando há relativamente poucas atividades, tipos de recursos, relações de precedência e tipos de habilidades.

4.3 Resultados para o RCPSP com múltiplos modos

As instâncias para o RCPSP com múltiplos modos foram obtidas da base MMLIB[3] (*Multi-mode Library*) desenvolvida por Peteghem e Vanhoucke (2014). Utilizaram-se dois conjuntos, MMLIB50 e MMLIB100, para um total de 1080 instâncias. As instâncias do conjunto MMLIB50 possuem 52 atividades, sendo a primeira e a última fictícias; dois recursos renováveis e dois não-renováveis; tempo de processamento das atividades e quantidade de consumo de recurso, por cada atividade, para cada um dos três modos de execução; relação de precedência entre as atividades; e, a disponibilidade de recurso para atender as atividades. O conjunto MMLIB100 possui as mesmas características do conjunto MMLIB50, exceto a quantidade de atividades que neste caso é de 102, incluindo as fictícias.

Para a resolução de cada instância foi imposto um tempo de limite de 3.600 segundos para o *Gurobi*, considerando o Modelo 4, definido em (3.18)-(3.24), sendo que a melhor solução encontrada dentro deste tempo foi reportada. A Tabela 4.4 contempla os resultados das 540 instâncias do conjunto MMLIB50 e 540 instâncias do conjunto MMLIB100.

Os resultados na Tabela 4.4 estão organizados por grupos de 5 instâncias, ou seja, MM501 refere-se ao primeiro grupo contendo 5 instâncias, de forma que os valores apresentados são a média para o referido grupo. Assim, os dados em cada linha da tabela são: nome do grupo; média do limitante inferior (LI), obtido ao resolver o modelo relaxado pelo *Gurobi*; número de instâncias do grupo resolvidas de forma ótima; média do valor de solução, isto é, do *makespan*; média do tempo de otimização em segundos (não está incluso o tempo gasto para obter o limitante inferior); e, o média do valor do *gap*, em

[3] http://mmlib.eu/index.php

porcentagem, retornado pelo *Gurobi*. No final da tabela, tem-se também a média geral, o desvio padrão, o mínimo e o máximo, considerando todos os grupos.

Conforme os resultados na Tabela 4.4, 431 instâncias foram resolvidas de forma ótima, correspondendo a 79,8% das instâncias para o conjunto MMLIB50 e sendo este resultado de 372 (isto é, 68,9%) instâncias para o conjunto MMLIB100. Ou seja, para todas essas instâncias o GAP foi de 0,0% dentro do tempo limite. Com relação a quantidade de grupos que obtiveram todas as instâncias resolvidas de forma ótima, para o conjunto MMLIB50 foi possível resolver 75 dos 108 grupos, enquanto para o conjunto MMLIB100 foi possível resolver 63 dos 108 grupos. Para os demais grupos de instâncias, o tempo de otimização alcançou os 3.600 segundos para pelo menos uma instância, com o *gap* diferente de 0,0%, gerando apenas uma solução viável e não provada ótima.

Tabela 4.4 – Resultados para os conjuntos MMLIB50 e MMLIB100 obtidos ao resolver o Modelo 4.

Grupo	Modelo 4 - MMLIB50					Grupo	Modelo 4 - MMLIB100				
	LI	QTD.	Sol.	Tempo (s)	GAP (%)		LI	QTD.	Sol.	Tempo (s)	GAP (%)
MM501	17,2	4	27,4	748,4	0,8	MM1001	23,0	0	35,4	3.600,0	7,4
MM502	19,2	4	23,6	974,1	0,8	MM1002	23,2	1	27,6	3.229,9	5,1
MM503	19,2	5	19,6	25,2	**0,0**	MM1003	22,4	4	22,6	734,4	0,9
MM504	17,8	4	19,8	731,1	0,9	MM1004	24,0	4	24,6	737,1	1,7
MM505	18,6	5	18,6	3,0	**0,0**	MM1005	20,8	4	21,6	1.382,7	1,0
MM506	22,0	5	22,0	3,0	**0,0**	MM1006	22,4	4	22,6	733,5	1,0
MM507	19,6	0	46,2	3.600,0	11,2	MM1007	23,8	0	53,0	3.600,0	12,0
MM508	18,8	0	39,0	3.600,0	12,8	MM1008	23,4	0	54,4	3.600,0	19,2
MM509	17,6	0	28,6	3.600,0	8,1	MM1009	23,6	0	34,6	3.600,0	5,5
MM5010	17,2	0	32,4	3.600,0	12,4	MM10010	20,8	0	33,0	3.600,0	8,9
MM5011	18,6	0	27,2	3.600,0	3,7	MM10011	23,6	1	35,8	3.004,4	4,2
MM5012	17,8	0	26,8	3.600,0	4,4	MM10012	22,2	0	82,0	3.600,0	23,8
MM5013	19,4	5	26,0	8,7	**0,0**	MM10013	22,4	5	30,8	45,7	**0,0**
MM5014	19,4	5	21,0	5,5	**0,0**	MM10014	23,0	5	24,2	31,7	**0,0**
MM5015	20,4	5	20,4	1,6	**0,0**	MM10015	24,2	5	24,2	8,7	**0,0**
MM5016	19,2	5	19,2	1,8	**0,0**	MM10016	23,6	5	23,6	12,1	**0,0**
MM5017	17,2	5	17,2	1,6	**0,0**	MM10017	21,4	5	21,4	8,4	**0,0**
MM5018	19,2	5	19,2	1,5	**0,0**	MM10018	20,6	5	20,6	9,5	**0,0**
MM5019	18,0	5	28,0	36,6	**0,0**	MM10019	22,4	5	29,2	119,9	**0,0**
MM5020	22,8	4	25,8	1.156,9	0,8	MM10020	21,6	4	25,4	1.081,6	0,8

(continua)

(continuação)

Grupo	Modelo 4 - MMLIB50					Grupo	Modelo 4 - MMLIB100				
	LI	QTD.	Sol.	Tempo (s)	GAP (%)		LI	QTD.	Sol.	Tempo (s)	GAP (%)
MM5021	18,6	5	19,0	407,5	**0,0**	MM10021	22,0	5	22,4	501,3	**0,0**
MM5022	20,0	4	20,4	723,6	0,9	MM10022	21,2	4	22,4	1.193,3	0,8
MM5023	18,0	5	18,8	4,1	**0,0**	MM10023	23,0	5	23,2	17,3	**0,0**
MM5024	18,6	5	18,6	15,4	**0,0**	MM10024	23,8	5	23,8	21,6	**0,0**
MM5025	18,4	5	26,6	9,0	**0,0**	MM10025	26,4	5	31,2	46,9	**0,0**
MM5026	18,2	5	20,6	5,1	**0,0**	MM10026	20,2	5	22,8	26,6	**0,0**
MM5027	18,0	5	18,0	1,5	**0,0**	MM10027	24,2	5	24,2	8,9	**0,0**
MM5028	21,2	5	21,2	1,7	**0,0**	MM10028	21,6	5	21,6	10,9	**0,0**
MM5029	18,0	5	18,0	1,5	**0,0**	MM10029	23,4	5	23,4	8,5	**0,0**
MM5030	18,2	5	18,2	1,6	**0,0**	MM10030	21,6	5	21,6	9,2	**0,0**
MM5031	17,8	5	26,4	14,8	**0,0**	MM10031	24,2	5	29,6	79,1	**0,0**
MM5032	18,2	5	22,2	9,7	**0,0**	MM10032	20,0	5	23,0	70,7	**0,0**
MM5033	20,8	5	20,8	1,8	**0,0**	MM10033	22,8	5	22,8	12,0	**0,0**
MM5034	21,2	5	21,4	2,5	**0,0**	MM10034	20,6	5	20,6	17,6	**0,0**
MM5035	20,2	5	20,2	1,9	**0,0**	MM10035	21,2	5	21,2	9,9	**0,0**
MM5036	20,2	5	20,2	2,0	**0,0**	MM10036	24,2	5	24,2	12,3	**0,0**
MM5037	26,6	5	41,4	300,9	**0,0**	MM10037*	36,0	1	50,5	1.931,6	4,9
MM5038	27,6	5	34,8	637,1	**0,0**	MM10038	32,2	1	41,6	2.984,9	5,1
MM5039	27,8	5	28,0	7,7	**0,0**	MM10039	33,2	4	33,6	1.282,2	0,6
MM5040	26,4	5	28,2	51,9	**0,0**	MM10040	30,8	1	33,8	2.891,6	3,6
MM5041	28,2	5	28,6	3,5	**0,0**	MM10041	30,4	5	30,6	78,5	**0,0**
MM5042	27,2	5	27,6	10,9	**0,0**	MM10042	34,2	5	34,2	41,4	**0,0**
MM5043*	26,8	0	70,3	3.600,0	16,3	MM10043*	36,0	0	321,0	3.600,0	79,4
MM5044	26,6	0	51,2	3.600,0	14,5	MM10044	32,4	0	144,4	3.600,0	51,2
MM5045	27,4	0	43,4	3.600,0	11,0	MM10045*	32,0	0	108,5	3.600,0	32,5
MM5046	25,8	0	43,4	3.600,0	12,2	MM10046	30,0	0	73,4	3.600,0	26,8
MM5047	25,6	0	42,0	3.600,0	10,3	MM10047	36,6	0	103,8	3.600,0	24,5
MM5048	28,0	0	38,0	3.600,0	9,3	MM10048	31,4	0	108,0	3.600,0	30,5
MM5049	24,2	5	38,8	84,5	**0,0**	MM10049	30,6	5	43,4	215,3	**0,0**
MM5050	29,0	5	34,8	14,9	**0,0**	MM10050	29,8	5	36,0	405,1	**0,0**
MM5051	25,6	5	25,6	2,5	**0,0**	MM10051	33,2	5	33,2	37,2	**0,0**
MM5052	26,2	5	27,6	7,1	**0,0**	MM10052	33,0	5	33,6	50,3	**0,0**
MM5053	27,0	5	27,0	1,8	**0,0**	MM10053	35,2	5	35,2	14,6	**0,0**
MM5054	24,2	5	24,4	1,9	**0,0**	MM10054	31,4	5	31,4	25,7	**0,0**
MM5055	25,0	4	40,0	936,2	0,5	MM10055*	31,0	0	48,3	3.600,0	4,6

(continua)

(continuação)

Grupo	Modelo 4 - MMLIB50					Grupo	Modelo 4 - MMLIB100				
	LI	QTD.	Sol.	Tempo (s)	GAP (%)		LI	QTD.	Sol.	Tempo (s)	GAP (%)
MM5056	24,8	2	34,8	2.763,5	3,8	MM10056	31,0	1	40,6	3.053,6	2,4
MM5057	27,4	4	27,8	744,2	0,6	MM10057	34,2	5	34,2	73,3	**0,0**
MM5058	29,2	4	31,4	743,8	0,7	MM10058	31,4	3	34,2	1.506,5	1,8
MM5059	27,2	5	27,8	11,2	**0,0**	MM10059	32,6	4	33,2	1.058,6	0,6
MM5060	28,0	5	28,0	8,8	**0,0**	MM10060	30,8	5	31,0	96,7	**0,0**
MM5061	28,8	5	40,0	27,2	**0,0**	MM10061	30,8	5	44,4	216,5	**0,0**
MM5062	28,2	5	31,4	21,3	**0,0**	MM10062	32,0	5	35,6	323,5	**0,0**
MM5063	28,0	5	28,0	2,1	**0,0**	MM10063	33,2	5	33,2	29,7	**0,0**
MM5064	27,2	5	27,4	2,9	**0,0**	MM10064	31,2	5	31,2	30,2	**0,0**
MM5065	27,0	5	27,0	1,6	**0,0**	MM10065	33,0	5	33,0	17,0	**0,0**
MM5066	27,6	5	27,6	2,1	**0,0**	MM10066	31,8	5	31,8	17,9	**0,0**
MM5067	26,2	5	38,6	64,4	**0,0**	MM10067	34,6	5	44,2	861,2	**0,0**
MM5068	25,0	5	30,4	30,8	**0,0**	MM10068	31,6	5	36,4	511,1	**0,0**
MM5069	24,4	5	24,4	2,5	**0,0**	MM10069	31,8	5	31,8	41,8	**0,0**
MM5070	25,0	5	26,0	8,3	**0,0**	MM10070	31,0	5	31,0	47,5	**0,0**
MM5071	27,6	5	27,6	2,4	**0,0**	MM10071	32,2	5	32,2	30,3	**0,0**
MM5072	25,6	5	25,6	2,5	**0,0**	MM10072	32,6	5	32,6	33,7	**0,0**
MM5073	44,4	4	72,2	1.304,9	0,9	MM10073*	54,5	0	105,0	3.600,0	18,4
MM5074	43,2	2	57,0	2.207,4	1,4	MM10074*	57,0	0	92,0	3.600,0	24,2
MM5075	45,4	5	46,2	15,5	**0,0**	MM10075	59,8	5	60,0	239,9	**0,0**
MM5076	38,2	5	43,0	207,4	**0,0**	MM10076	54,8	1	62,6	2.894,5	7,5
MM5077	44,0	5	45,0	10,8	**0,0**	MM10077	61,6	5	62,2	66,2	**0,0**
MM5078	45,2	5	46,0	13,7	**0,0**	MM10078	51,8	5	52,4	321,2	**0,0**
MM5079	-	-	-	-	-	MM10079	-	-	-	-	-
MM5080	43,0	0	94,0	3.600,0	27,0	MM10080	51,6	0	228,6	3.600,0	62,3
MM5081	50,0	1	69,4	2.915,7	17,8	MM10081*	53,0	0	115,0	3.600,0	39,1
MM5082	44,0	0	66,2	3.600,0	16,6	MM10082	54,6	0	163,4	3.600,0	51,1
MM5083	45,2	2	57,0	2.685,0	6,8	MM10083	52,0	0	107,3	3.600,0	34,5
MM5084	41,6	0	61,4	3.600,0	13,7	MM10084	53,0	0	151,0	3.600,0	45,8
MM5085	42,8	5	73,2	231,0	**0,0**	MM10085*	59,3	3	75,7	1.814,2	**0,0**
MM5086	46,0	5	57,2	101,6	**0,0**	MM10086	59,6	5	67,8	847,9	**0,0**
MM5087	43,0	5	43,0	5,9	**0,0**	MM10087	58,4	5	58,4	43,0	**0,0**
MM5088	43,4	5	45,4	64,2	**0,0**	MM10088	51,0	5	52,6	129,7	**0,0**
MM5089	47,4	5	47,4	6,0	**0,0**	MM10089	55,2	5	55,2	31,9	**0,0**
MM5090	46,4	5	46,4	8,3	**0,0**	MM10090	57,6	5	57,6	35,5	**0,0**

(continua)

(conclusão)

Grupo	Modelo 4 - MMLIB50					Grupo	Modelo 4 - MMLIB100				
	LI	QTD.	Sol.	Tempo (s)	GAP (%)		LI	QTD.	Sol.	Tempo (s)	GAP (%)
MM5091	40,6	4	67,0	1.133,5	0,3	MM10091*	56,0	0	115,0	3.600,0	27,0
MM5092*	41,8	2	57,5	2.762,2	1,3	MM10092*	56,5	0	78,8	3.600,0	10,5
MM5093	44,4	5	46,0	104,1	**0,0**	MM10093	58,4	5	58,4	78,6	**0,0**
MM5094	39,6	3	46,6	2.092,8	1,6	MM10094	53,6	2	61,0	2.401,4	7,0
MM5095	44,4	5	46,2	34,2	**0,0**	MM10095	56,0	5	56,0	188,2	**0,0**
MM5096	41,6	4	43,6	807,6	0,8	MM10096	52,0	2	55,0	2.185,3	2,1
MM5097	42,8	5	65,4	157,7	**0,0**	MM10097	52,6	4	77,2	1.701,2	0,3
MM5098	44,8	5	56,4	101,9	**0,0**	MM10098	52,2	5	63,0	1.130,0	**0,0**
MM5099	38,6	5	38,6	8,8	**0,0**	MM10099	54,8	5	54,8	35,4	**0,0**
MM50100	44,8	5	46,2	17,5	**0,0**	MM100100	54,4	5	54,6	59,1	**0,0**
MM50101	39,4	5	39,4	4,1	**0,0**	MM100101	57,4	5	57,4	30,2	**0,0**
MM50102	38,4	5	38,4	5,7	**0,0**	MM100102	53,0	5	53,0	34,3	**0,0**
MM50103	43,4	5	68,2	146,6	**0,0**	MM100103*	52,5	2	81,5	1.414,6	**0,0**
MM50104	45,4	5	54,2	113,6	**0,0**	MM100104	54,4	4	67,8	1.596,2	0,3
MM50105	40,6	5	40,6	7,2	**0,0**	MM100105	58,0	5	58,0	55,2	**0,0**
MM50106	44,0	5	47,4	52,2	**0,0**	MM100106	57,0	4	57,2	776,8	0,3
MM50107	43,8	5	43,8	6,1	**0,0**	MM100107	57,0	4	57,2	776,8	0,3
MM50108	43,8	5	43,8	7,4	**0,0**	MM100108	53,6	5	53,6	63,4	**0,0**
Média	29,6	4,0	36,4	773,3	2,1	**Média**	36,6	3,5	53,1	1.207,9	6,5
Desvio Padrão	10,3	1,8	15,7	1.306,9	4,9	**Desvio Padrão**	13,9	2,1	43,1	1.456,1	14,5
Mínimo	17,2	0,0	17,2	1,5	**0,0**	**Mínimo**	20,0	0,0	20,6	8,4	**0,0**
Máximo	50,0	5,0	94,0	3.600,0	27,0	**Máximo**	61,6	5,0	321,0	3.600,0	79,4

(-) Nenhuma instância foi resolvida, ou seja, o *Gurobi* não conseguiu encontrar qualquer solução viável dentro do tempo limite de 3.600 segundos.

(*) Pelo menos uma instância do grupo não foi resolvida, ou seja, o *Gurobi* não conseguiu encontrar qualquer solução viável dentro do tempo limite de 3.600 segundos.

Com relação ao *gap* médio reportado na Tabela 4.4, para o conjunto MMLIB50, este valor foi 2,1% (com desvio de padrão de 4,9%), enquanto que para o conjunto MMLIB100 foi de 6,5% (com desvio padrão de 14,5%). Ao observar o maior GAP no conjunto MMLIB50, este foi de 27,0% para o grupo MM5080, enquanto que para o conjunto MMLIB100, tal GAP foi de 79,4% para o grupo MM10043. Nota-se, assim, que o Modelo 4 não se mostrou tão satisfatório enquanto aplicado sobre as instâncias do conjunto MMLIB100, que envolve praticamente o dobro de atividades em comparação com o MMLIB50.

Ao considerar os limitantes inferiores reportados na Tabela 4.4 a partir da resolução do modelo relaxado associado ao Modelo 4, tem-se que para 32 grupos do conjunto MMLIB50, o valor médio desse limitante coincidiu com o valor médio da solução para o

grupo. Por outro lado, no caso do conjunto MMLIB100, o número de grupos com essa característica foi de 34. Com relação ao tempo computacional, o tempo médio foi de 773,3 segundos (com desvio padrão de 1.306,9 segundos) para o conjunto MMLIB50, ao passo que foi de 1.207,9 segundos (com desvio padrão de 1.456,1 segundos) para o MMLIB100. Percebe-se, ainda, que o tempo limite foi atingido para todas as instâncias de 15 grupos do conjunto MMLIB50, sendo eles: J507, J508, J509, J5010, J5011, J5012, J5043, J5044, J5045, J5046, J5047, J5048, J5080, J5082 e J5084. Para o conjunto MMLIB100, isso ocorreu para 22 grupos, a saber: J1001, J1007, J1008, J1009, J10010, J10012, J10043, J10044, J10045, J10046, J10047, J10048, J10055, J10073, J10074, J10080, J10081, J10082, J10083, J10084, J10091 e J10092.

4.4 Resultados para o RCPSP com tempos de atraso

Para os testes que consideram o RCPSP com as restrições de tempos atraso mínimo e máximo, adaptaram-se as instâncias da PSPLIB, em particular, os conjuntos J30 e J60, resultando nos respectivos conjuntos L30 e L60. A adaptação consistiu em definir, para todas as atividades que têm relação de precedência (isto é, fazer o conjunto *lag* ser igual ao *pred*), um valor de d_{ij}^{min} *e o* d_{ij}^{max}. Neste caso, considerou-se d_{ij}^{min} como sendo o mínimo valor entre a duração d_i da atividade i e a duração d_j da atividade j, depois dividido por 3. Para o valor de d_{ij}^{max}, considerou-se o máximo entre d_i e d_j, depois multiplicado por 10. Além disso, para a resolução de cada instância, impôs-se um tempo limite de 3.600 segundos, conforme estipulado nos testes anteriores, sendo que a melhor solução encontrada dentro deste tempo limite foi reportada na Tabela 4.5.

Na Tabela 4.5 estão os resultados para as respectivas 480 instâncias do conjunto L30 e L60, dada a resolução do Modelo 5, definido em (3.25)-(3.32). Os valores apresentados nessa tabela seguem a mesma interpretação da Tabela 4.4, porém, os resultados estão organizados em grupos de 10 instâncias cada, de forma que os valores apresentados correspondem a média do referido grupo.

De acordo com os resultados da Tabela 4.5, das 480 instâncias para o conjunto L30, 438 foram resolvidas de forma ótima (ou seja, 91,3%) e 345 para o conjunto L60, o que equivale a 71,9%. Além disso, todos os 48 grupos do conjunto L30 tiveram pelo menos uma instância resolvida de forma ótima, sendo que 25 grupos tiveram todas as 10 instâncias do grupo resolvidas na otimalidade. Para o conjunto L60, para 44 grupos, pelo menos uma instância teve a solução ótima obtida, enquanto que para 8 grupos, todas as 10 instâncias do grupo tiveram a solução ótima encontrada dentro do tempo limite.

Tabela 4.5 – Resultados para o conjunto L30 e L60.

Grupo	Modelo 5 - L30					Grupo	Modelo 5 - L60				
	LI	QTD.	Sol.	Tempo (s)	GAP (%)		LI	QTD.	Sol.	Tempo (s)	GAP (%)
L301*	51,3	9	55,9	0,5	**0,0**	L601*	81,7	9	85,9	14,4	**0,0**
L302	53,0	10	54,3	0,4	**0,0**	L602	78,2	10	78,5	2,8	**0,0**
L303	70,0	10	71,1	0,3	**0,0**	L603*	81,4	9	82,4	3,6	**0,0**
L304*	62,6	9	62,6	0,2	**0,0**	L604*	81,3	8	81,3	1,7	**0,0**
L305*	57,2	9	72,8	22,6	**0,0**	L605*	75,5	5	88,8	1.941,9	1,6
L306	54,3	10	56,5	1,7	**0,0**	L606	79,3	10	79,5	7,5	**0,0**
L307	51,4	10	53,2	0,5	**0,0**	L607*	82,8	9	82,8	3,0	**0,0**
L308	59,4	10	59,4	0,4	**0,0**	L608*	82,3	9	82,3	3,8	**0,0**
L309	59,1	10	78,8	99,4	**0,0**	L609*	82,1	0	112,7	3.600,0	12,0
L3010	55,4	10	56,3	2,4	**0,0**	L6010*	85,0	9	85,0	9,7	**0,0**
L3011	64,9	10	65,2	1,9	**0,0**	L6011*	80,3	8	80,3	7,6	**0,0**
L3012	58,1	10	58,1	0,7	**0,0**	L6012*	77,0	9	77,0	4,9	**0,0**
L3013*	50,8	3	76,4	2.529,5	3,6	L6013*	74,4	0	146,7	3.600,0	27,7
L3014*	53,9	9	55,3	3,7	**0,0**	L6014*	74,9	9	74,9	13,8	**0,0**
L3015	61,2	10	61,4	1,0	**0,0**	L6015*	85,9	8	85,9	11,0	**0,0**
L3016	53,0	10	53,0	0,6	**0,0**	L6016	75,7	10	75,7	8,1	**0,0**
L3017*	63,6	7	69,1	1,2	**0,0**	L6017	83,0	10	87,5	24,1	**0,0**
L3018	58,7	10	61,1	0,6	**0,0**	L6018*	87,9	8	89,0	7,1	**0,0**
L3019	59,0	10	59,9	0,3	**0,0**	L6019	86,4	10	88,7	4,8	**0,0**
L3020	58,5	10	58,5	0,3	**0,0**	L6020*	89,0	7	89,0	2,1	**0,0**
L3021*	59,8	9	76,0	12,4	**0,0**	L6021*	81,6	4	100,0	2.131,4	**0,0**
L3022	59,6	10	61,6	1,4	**0,0**	L6022*	83,0	9	84,3	8,4	**0,0**
L3023	64,2	10	65,1	0,5	**0,0**	L6023	86,8	10	87,1	5,6	**0,0**
L3024*	60,4	9	60,4	0,4	**0,0**	L6024	84,1	10	84,1	4,0	**0,0**
L3025	56,7	10	82,5	209,8	**0,0**	L6025	81,0	0	129,4	3.600,0	12,5
L3026*	62,0	9	63,6	1,3	**0,0**	L6026*	83,7	7	84,0	12,5	**0,0**
L3027	65,7	10	65,7	0,7	**0,0**	L6027*	87,9	9	87,9	7,2	**0,0**
L3028	67,6	10	67,6	0,6	**0,0**	L6028*	88,0	6	88,0	6,6	**0,0**
L3029	59,8	6	92,8	1.804,1	4,5	L6029	86,5	0	156,0	3.600,0	5,8
L3030*	60,6	9	62,0	3,9	**0,0**	L6030	89,9	10	90,4	167,5	**0,0**
L3031*	64,4	9	64,7	1,2	**0,0**	L6031*	84,7	9	84,7	11,8	**0,0**
L3032*	65,1	9	65,1	0,9	**0,0**	L6032*	96,8	8	96,8	10,8	**0,0**
L3033	63,5	10	69,3	1,2	**0,0**	L6033*	99,4	9	105,1	32,8	**0,0**

(continua)

(conclusão)

Grupo	Modelo - 5 L30					Grupo	Modelo 5 - L60				
	LI	QTD.	Sol.	Tempo (s)	GAP (%)		LI	QTD.	Sol.	Tempo (s)	GAP (%)
L3034*	66,3	9	68,9	0,6	**0,0**	L6034*	85,4	8	85,9	4,4	**0,0**
L3035	68,1	10	68,7	0,3	**0,0**	L6035*	89,3	9	91,6	6,1	**0,0**
L3036*	68,2	9	68,2	0,3	**0,0**	L6036*	88,7	9	88,7	3,0	**0,0**
L3037*	63,7	9	84,4	40,4	**0,0**	L6037*	91,3	2	114,0	1.739,5	1,6
L3038*	67,4	7	70,3	1,3	**0,0**	L6038*	89,1	9	90,2	13,7	**0,0**
L3039	67,4	10	69,5	0,7	**0,0**	L6039*	94,9	7	95,0	7,3	**0,0**
L3040*	66,1	7	66,1	0,4	**0,0**	L6040*	97,8	9	97,8	5,9	**0,0**
L3041*	66,6	9	97,6	208,9	**0,0**	L6041*	97,1	1	135,8	3.414,4	14,9
L3042	70,2	10	72,0	2,9	**0,0**	L6042*	91,9	8	92,9	59,0	**0,0**
L3043*	67,1	9	67,7	0,9	**0,0**	L6043*	89,5	6	89,7	10,7	**0,0**
L3044*	63,0	9	63,0	0,6	**0,0**	L6044*	91,6	7	91,6	6,1	**0,0**
L3045*	69,9	7	103,1	1.222,0	1,4	L6045	87,3	-	-	3.600,0	-
L3046	65,7	10	67,1	5,0	**0,0**	L6046*	88,3	9	90,1	552,2	**0,0**
L3047	65,9	10	66,5	1,5	**0,0**	L6047*	87,7	6	87,7	13,0	**0,0**
L3048*	64,4	7	64,4	0,9	**0,0**	L6048*	91,0	7	91,0	8,4	**0,0**
Média	61,8	9,1	67,4	129,0	0,2	**Média**	85,8	7,3	93,1	589,9	1,6
Desvio Padrão	5,3	1,4	10,7	471,5	0,8	**Desvio Padrão**	6,0	3,0	17,3	1232,9	5,1
Mínimo	50,8	3,0	53,0	0,2	**0,0**	**Mínimo**	74,4	0,0	74,9	1,7	**0,0**
Máximo	70,2	10,0	103,1	2.529,5	4,5	**Máximo**	99,4	10,0	156,0	3600,0	27,7

(-) Nenhuma instância foi resolvida, ou seja, o *Gurobi* não conseguiu encontrar qualquer solução viável dentro do tempo limite de 3.600 segundos.

(*) Pelo menos uma instância do grupo não foi resolvida, ou seja, o *Gurobi* não conseguiu encontrar qualquer solução viável dentro do tempo limite de 3.600 segundos.

Com relação ao valor do *gap* na Tabela 4.5, o valor médio para o conjunto L30 foi de 0,2% (com desvio padrão de 0,8%), enquanto que o do conjunto L60 foi de 1,6% (com desvio padrão de 5,1%). O pior *gap* no L30 foi para o grupo L3029 e corresponde a 4,5%, enquanto que no L60, o pior *gap* foi de 27,7%, para o grupo L6013. É importante mencionar que para os grupos L6041 e L6045 não foi possível obter solução (isto é, modelo inviável) para qualquer instância do grupo, devido ao valor do multiplicador utilizado, que foi de 10 para o tempo máximo. Portanto, utilizou-se um multiplicador igual a 20 para o tempo máximo, de forma que apenas uma instância do grupo L6041 foi resolvida dentro do tempo limite, assim indicando a necessidade de usar um multiplicador maior.

Quanto ao tempo computacional, nota-se que o tempo limite foi atingido para todas as instâncias de 5 grupos do conjunto L60, sendo eles: L609, L6013, L6025, L6029 e L6045. O tempo médio foi de 129,0 segundos (com desvio padrão de 471,5 segundos) para o

conjunto L30 e de 589,9 segundos (com desvio padrão de 1.232,9 segundos) para o conjunto L60.

CAPÍTULO 5

CONCLUSÕES E TRABALHOS FUTUROS

Os problemas de programação da produção estão presentes em atividades cotidianas, especialmente o problema de programação de projetos com restrição de recursos investigado neste trabalho. Sendo assim, apresentaram-se modelos de programação linear inteira (mista) para o RCPSP com a inclusão de algumas restrições reais (isto é, múltiplas habilidades, múltiplos modos de execução e tempos de atraso entre atividades), os quais foram resolvidos por meio de um método exato do tipo *branch-and-cut*. O primeiro e o segundo modelo foram considerados para o RCPSP sem considerar a adição de qualquer restrição real, com o intuito de compará-los em termos de número de instâncias resolvidas de forma ótima, *gap* da solução e tempo de otimização.

Como forma de reduzir o espaço de busca da solução ótima, relacionada ao *makespan*, para os modelos para o RCPSP, considerou-se o cálculo de um limitante inferior, dado pela resolução da versão relaxada do respectivo modelo, isto é, do modelo sem a restrição que respeita o consumo dos recursos, uma vez que essa restrição pode ser um dos gargalos na hora da resolução. Como limitante superior, considerou-se uma solução viável obtida pela heurística *List Scheduling* definida na literatura, que vai escalonando as atividades sem precedência e com menor tempo de processamento primeiro.

Os resultados dos experimentos computacionais apontaram que o segundo modelo foi ligeiramente melhor, uma vez que obteve um melhor desempenho se comparado com o primeiro modelo nas 960 instâncias adotadas para os testes, envolvendo 32 e 62 atividades. Com o segundo modelo foi possível resolver um maior número de instâncias e ter mais soluções com um *gap* menor, apesar do tempo computacional médio para o conjunto de instâncias com 32 atividades ter sido levemente maior. Observou-se também que ambos os modelos começaram a ter dificuldades para resolver, dentro do tempo limite de uma hora, as instâncias com 62 atividades.

A partir do segundo modelo, fez-se a adição da restrição de múltiplas habilidades no RCPSP, que impõe níveis de habilidade requeridos pelas atividades e que somente certos recursos conseguem atender, sendo então preciso escolher qual recurso atende cada atividade conforme a disponibilidade total dos recursos. Com isso, desenvolveu-se um terceiro modelo, baseado no segundo modelo, adicionando a restrição de múltiplas habilidades. Os resultados computacionais indicaram que o terceiro modelo aplicado sobre as 36 instâncias adaptadas para incluir a restrição de múltiplas habilidades conseguiu resolver 50,0% (que corresponde a 18 de 36) das instâncias de forma ótima, com um tempo computacional médio de 2.246,9 segundos e desvio padrão de 1.526,9 segundos. Para as demais instâncias, em que não foi possível obter qualquer solução dentro do tempo limite de uma hora, o número de atividades é superior a 202, representando instâncias de tamanho relativamente grande.

A restrição de múltiplos modos também foi acrescentada ao RCPSP, dando origem ao quarto modelo. Nessa restrição, uma atividade pode ter mais de um modo de execução, o que implica em definir durações e consumos de recursos diferentes para as atividades. Com isso, dependendo do modo escolhido, a duração total do projeto pode ser maior/menor. Os resultados computacionais apontaram que o quarto modelo aplicado sobre as 1080 instâncias, divididas em dois conjuntos, um com 52 e o outro com 102 atividades, permitiu resolver 79,8% das instâncias na otimalidade para o conjunto com 52 atividades e 68,9% para o conjunto com 102 atividades, com um tempo computacional médio de 773,3 e 1.207,9 segundos, respectivamente. Além disso, o *gap* médio geral foi de, respectivamente, 2,1% e 6,5%. Com isso, esse modelo foi satisfatório para as instâncias com menos atividades, começando a ter dificuldade para resolver as instâncias com 102 atividades.

A restrição de tempos de atraso, que modela uma janela de tempo mínimo e máximo para que uma atividade possa começar após o término de outra, foi incluída no segundo modelo para dar origem ao quinto modelo. Os resultados computacionais sobre esse quinto modelo, considerando 980 instâncias adaptadas da literatura, separadas em dois conjuntos, o primeiro com 32 atividades e o segundo com 62 atividades, permitiram obter a solução de ótima de 91,3% das instâncias com 32 atividades e de 71,9% das instâncias com 62 atividades, com um tempo computacional médio respectivo de 129,0 e de 589,9 segundos. Nota-se ainda que o *gap* médio geral foi de 0,2% para o primeiro conjunto de instâncias e de 1,6% para o segundo conjunto, indicando que esse modelo precisaria de mais tempo para resolver instâncias que envolvem mais atividades.

A partir da análise dos cinco modelos considerados para o RCPSP sem/com restrições reais, conclui-se que desenvolver um modelo de programação matemática para resolver instâncias de médio e grande porte, dado um tempo limite de uma hora de execução, é um desafio para aqueles que buscam por soluções ótimas como forma de

tomar suas decisões. Desta forma, quanto a questão de pesquisa apresentada inicialmente, pode-se dizer, a partir dos resultados obtidos, que foi possível obter um modelo de programação linear inteira para o RCPSP na presença de restrições reais capaz de resolver instâncias com número relativamente satisfatório de atividades (isto é, com até 100 atividades).

A dificuldade em resolver o RCPSP, que é um problema NP-Difícil, ainda mais quando aplicado à contextos reais, impulsiona a comunidade científica a desenvolver modelos matemáticos mais elaborados em busca da solução ótima e outros métodos de solução. Acredita-se que tais modelos podem trazer grandes vantagens para o ambiente corporativo, assim como colabora no processo de tomada de decisão, redução de desperdícios e no tempo de duração total de projetos. Portanto, como sugestão de trabalhos futuros que podem ser realizados a partir deste estudo, destaca-se:

- Usar a solução do modelo relaxado como solução inicial para o *Gurobi* durante a resolução do respectivo modelo para o problema, pois, para diversas instâncias, o limitante inferior dado pelo modelo relaxado coincidiu com a solução retornada pelo *Gurobi*. Como foi utilizado apenas o valor do limitante inferior como entrada para o modelo do problema resolvido pelo Gurobi, em vários casos o Gurobi demorou para encontrar uma solução com o valor de, pelo menos, o limitante inferior;
- Considerar outras regras de escolha de atividades no passo 3 da heurística *List Scheduling*, pois conforme Gafarov, Lazarev e Werner (2010), a regra adotada pode influenciar na solução final e, com isso, na redução do limitante superior;
- Estudar e desenvolver modelos matemáticos para o RCPSP que inclua outras restrições reais, como tempos de preparo, atividades com tempo de processamento incertos, recursos estocásticos, janelas de tempo, entre outras. Outra linha de trabalho poderia considerar o impacto de outros tipos de função objetivo, além do *makespan*, como minimizar o atraso total, o fluxo total, o custo do projeto, entre outras (PINEDO, 2012); e,
- Aprimorar os modelos com a inserção e proposta de desigualdades válidas, bem como adotando outros limitantes inferiores, pela relaxação Lagrangeana, e superiores, pelo desenvolvimento de outros métodos heurísticos, em especial, de meta-heurísticas como Algoritmo Genético, Busca Tabu e Busca Local Iterada.

REFERÊNCIAS

ALMEIDA, B. F.; CORREIA, I.; SALDANHA-DA-GAMA, F. Priority-based heuristics for the multi-skill resource constrained project scheduling problem. **Expert Systems with Applications**, v. 57, p. 91-103, 2016.

ARENALES, M.; ARMENTANO, V.; MORABITO, R.; YANASSE, H. **Pesquisa Operacional.** Rio de Janeiro: Campus, 2007.

ARROYO, J. E. C. **Heurísticas e Metaheurísticas para Otimização Combinatória Multiobjetivo.** 2002, 231p. Tese (Doutorado em Engenharia Elétrica) – Universidade Estadual de Campinas, Faculdade de Engenharia Elétrica e de Computação, Campinas.

AZEVEDO, G. H. I.; PESSOA, A. A.; TORRES, C. R. R. Problemas de escalonamento de projetos com restrição de recursos: um estudo de caso no setor de petróleo e gás. **Pesquisa Operacional para o Desenvolvimento,** Rio de Janeiro, v. 4, n. 3, p. 288-303, 2012.

BARADARAN, S.; GHOMI, S. M. T. F.; MOBINI, M.; HASHEMIN. S. S. A hybrid scatter search approach for resource-constrained project scheduling problem in PERT-type networks. **Advances in Engineering Software**, v. 41, p. 966-975, 2010.

BAUMANN, P.; FÜNDELING, C.-U.; TRAUTMANN, N. The resource constrained project scheduling problem with work-content constraints. In: Schwindt, C.; Zimmermann, J. (eds.), **Handbook on Project Management and Scheduling,** Springer International Publishing, v. 1, p. 533-544, 2015.

BIANCO, L.; CARAMIA, M. An exact algorithm to minimize the makespan in project scheduling with scarce resources and generalized precedence relations. **European Journal of Operational Research**, v. 219, n.1, p. 73-85, 2012.

BRUCKER, P.; DREXL, A.; MOHRING, R.; NEUMANN, K.; PESCH, E. Resource-constrained project scheduling: notation, classification, models, and methods. **European Journal of Operational Research**, v. 112, n. 1, p.3-41, 1999.

BRUCKER, P.; KNUST, S.; SCHOO, A.; THIELE, O. A Branch and bound algorithm for the resource-constrained project scheduling problem. **European Journal of Operational Research**, v. 107, p. 272-288, 1998.

BRYMAN, A. **Research methods and organization studies.** London: Uniwin Hyman, 1989. 224 p.

BUKATA, L.; SUCHA, P.; HANZÁLEK, Z. Solving the resource constrained project scheduling problem using the parallel tabu search designed for the CUDA platform. **Journal of Parallel and Distributed Computing**, v. 77, p. 58-68, 2015.

CINTRA, G. F.; MIYAZAWA, F. K.; WAKABAYASHI, Y.; XAVIER, E.C. Algorithms for two-dimensional cutting stock and strip packing problems using dynamic programming and column generation. **European Journal of Operational Research**, v. 191, n. 1, p. 61-85, 2008.

COLIN, E. C. **Pesquisa Operacional:** 170 aplicações em estratégia, finanças, logística, produção, marketing e vendas. Rio de Janeiro: LTC, 2011.

CORREIA, I.; SALDANHA-DA-GAMA, F. The impact of fixed and variable costs in a multi-skill project scheduling problem: An empirical study. **Computers & Industrial Engineering**, v. 72, p. 230-238, 2014.

CRAVO, G. L. **Escalonamento de projetos com restrições de recursos e múltiplos modos de processamento:** soluções heurísticas e uma aplicação à programação de manutenção industrial. 2009. 87 f. Dissertação (Mestrado em Informática) – Universidade Federal do Espírito Santo, Vitória, 2009.

DREXL, A.; KIMMS, A. Optimization guided lower and upper bounds for the resource investment problem. **Journal of the Operational Research Society**, v. 52, n. 3, p. 340-351, 2001.

FUCHIGAMI, H. Y.; RANGEL, S. Uma análise de estudos de casos em sequenciamento da produção. **Anais do XLVI SBPO Simpósio Brasileiro de Pesquisa Operacional**, Salvador, p. 159-170, 2014.

FÜNDELING, C.-U.; TRAUTMANN, N. A priority-rule method for project scheduling with work-content constraints. **European Journal of Operational Research**, v. 203, n. 3 p. 568-574, 2010.

GAFAROV, E. R.; LAZAREV, A. A.; WERNER, F. **On lower and upper bounds for the resource-constrained project scheduling problem.** Technical Report, Series: 2010-08. Fakultät für Mathematik, Otto-von-Guericke-Universität, 2010. 27p.

GAREY, M. R.; JOHNSON, D. S. **Computers and intractability: a guide to the theory of NP- completeness**. São Francisco: Freeman,1979.

GIL, A.C. **Como elaborar projetos de pesquisa.** 4. ed. São Paulo: Editora Atlas, 2009.

GOLDBARG, M. C.; LUNA, H. P. **Otimização combinatória e programação linear:** modelos e algoritmos. 2ª ed. Rio de Janeiro: Campus, 2005.

HABIBI, F.; BARZINPOUR, F.; SADJADI, S. J. Resource constrained project scheduling problem: review of past and recent developments. **Journal of Project Management**, v. 3, n. 2, p. 55-88, 2018.

HARTMANN, S.; BRISKORN, D. A survey of variants and extensions of the resource-constrained project scheduling problem. **European Journal of Operational Research**, v. 207, n.1, p. 1-14, 2010.

HEILMANN, R. A branch-and-bound procedure for the multi-mode resource-constrained project scheduling problem with minimum and maximum time lags. **European Journal of Operational Research**, v. 144, p. 348-365, 2003.

HERZ, J. C. Recursive computational procedure for two-dimensional stock cutting. **IBM Journal of Research and Development**, IBM, v. 16, n. 5, p. 462-469, 1972.

HOEHENE, L. G.; KUSTER, L. C. C.; LORENZONI, L. L. Scatter search com aninhamento de conjuntos referências aplicado ao problema de escalonamento de projetos com restrição de recursos e múltiplos modos de processamento. **Anais do XLII SBPO Simpósio Brasileiro de Pesquisa Operacional,** Rio Grande do Sul, p. 1561-1571, 2010.

JAVANMARD, S.; NADJAFI, B. A.; NIAKI, S. T. A. Preemptive multi-skilled resource investment project scheduling problem: Mathematical modelling and solution approaches. **Computers and Chemical Engineering**, v. 96, p. 55-68, 2017.

KE, H.; LIU, B. Fuzzy project scheduling problem and its hybrid intelligent algorithm. **Applied Mathematical Modelling,** v. 34, p. 301-308, 2010.

KOLISCH, R.; SPRECHER, A. PSPLIB - A project scheduling problem library: or software - ORSEP operations research software exchange program. **European Journal of Operational Research**, v. 96, n. 1, p. 205-216, 1997.

KONÉ, O.; ARTIGUES, C.; LOPEZ, P.; MONGEAU, M. Event-based MILP models for resource-constrained project scheduling problems. **Computers & Operations Research**, v. 38, n.1, p. 3-13, 2011.

LEAL, A. J. S. **Algoritmos de investigação operacional para um problema de sequenciamento de projetos.** 2007. 123 f. Dissertação (Mestrado em Engenharia Industrial) - Universidade do Minho, Escola de Engenharia, Portugal, 2007.

LIU, Z.; YANG, L.; DENG, R.; TIAN, J. An effective approach with feasible space decomposition to solve resource-constrained project scheduling problems. **Automation in Construction**, v.75, p. 1-9, 2017.

MARTINS, J. P. **O problema do agendamento semanal de aulas.** 2010. 83 p. Dissertação (Mestrado em Ciência da Computação) - Universidade Federal de Goiás, Instituto de Informática, Goiânia.

MAURI, G. R. **Novas abordagens para representação e obtenção de limitantes e soluções para alguns problemas de otimização combinatória.** 2008. 239 f. Tese (Doutorado em Computação Aplicada) – Instituto Nacional de Pesquisas Espaciais, São José dos Campos, 2008.

MENDES, J. J. M.; GONÇALVES, J. F. Um algoritmo genético para o problema do sequenciamento de projetos com recursos limitados. **Investigação Operacional**, v. 23, p. 179-195, 2003.

MIGUEL, P. A. C. **Metodologia de pesquisa em engenharia de produção e gestão de operações.** 2ª ed. Rio de Janeiro: Campus, 2012.

MIKA, M.; RÓZYCKI, R.; WALIGÓRA, G. Some new concepts of setup costs in multimode resource-constrained project scheduling problem. **Anais do XVII MMRAR International Conference on Methods & Models in Automation & Robotics**, Miedzyzdrojie, p. 470-474, 2012.

MINGOZZI, A.; MANIEZZO, V.; RICCIARDELLI, S.; BIANCO, L. An exact algorithm for the resource-constrained project scheduling problem based on a new mathematical formulation. **Management Science**, v. 44, n. 5, p. 714-729, 1998.

MURITIBA, A. E. F.; RODRIGUES, C. D.; COSTA, F. A. A path-relinking algorithm for the multi-mode resource constrained project scheduling problem. **Computers and Operations Research**, v. 92, p. 145-154, 2018.

MYSZKOWSKI, P. B.; LASZCZYK, M.; NIKULIN, I.; SKOWROŃSKI, M. IMOPSE: a library for bicriteria optimization in multi-skill resource-constrained project scheduling problem. **Soft Computing**, v. 22, n. 235, p. 1-14, 2018.

MYSZKOWSKI, P. B.; OLECH, L. P.; LASZCZYK, M.; SKOWRRONSKI, M. E. Hybrid differential evolution and greedy algorithm (DEGR) for solving multi-skill resource-constrained project scheduling problem. **Applied Soft Computing**, v. 62, p. 1-14, 2018.

NABER, A.; KOLISCH, R. MIP models for resource-constrained project scheduling with flexible resource profiles. **European Journal of Operational Research**, v. 239, p. 335-348, 2014.

NADJAFI, B. A.; MAJLESI, M. Resource constrained project scheduling problem with setup times after preemptive processes. **Computers and Chemical Engineering**, v. 69, p. 16-25, 2014.

ÖZDAMAR, L.; ULUSOY, G. A survey on the resource-constrained project scheduling problem. **IIE Transactions**, v. 27, n. 5, p. 574-586, 1995.

OZTEMEL, E.; SELAM, A. A. Bees algorithm for multi-mode, resource-constrained project scheduling in molding industry. **Computers & Industrial Engineering**, v. 112, p. 187-196, 2017.

PETEGHEM, V. V.; VANHOUCKE, M. An experimental investigation of metaheuristics for the multi-mode resource-constrained project scheduling problem on new dataset instances. **European Journal of Operational Research**, v. 235, n. 1, p. 62-72, 2014.

PINEDO, M. L. **Scheduling: theory, algorithms, and systems**. 4ª ed. New York: Springer, 2012.

PRITSKER, A. A. B.; WATTERS, L. J.; WOLFE, P. M. Multi-project scheduling with limited resources: a zero-one programming approach. **Management Science**, v. 16, n. 1, p. 93-108, 1969.

RIVERA, J. C.; MORENO-V, L. F.; DÍAZ-S, F. J.; PEÑA-Z, G. L. A hybrid heuristic algorithm for solving the resource constrained project scheduling problem (RCPSP). **Revista EIA**, v. 10, n. 20, p. 87-100, 2013.

ROCHA, I. M. **Uma abordagem otimizada para o problema de alocação de equipes e escalonamento de tarefas para a obtenção de cronogramas eficientes.** 2011. 120 f. Dissertação (Mestrado em Ciência da Computação) – Universidade Estadual do Ceará, Fortaleza, 2011.

SEVERINO, A. J. **Metodologia do trabalho científico.** 23ª ed. São Paulo: Cortez, 2007.

SCHUTT, A.; FEYDY, T.; STUCKEY, P. J.; WALLACE, M. G. Solving RCPSP/max by lazy clause generation. **Journal of Scheduling**, v. 16, n. 3, p. 273-289, 2013.

SILVA, A. R. V. Um método híbrido para um problema de escalonamento de projetos. **Anais do XLIV SBPO Simpósio Brasileiro de Pesquisa Operacional**, Rio de Janeiro, p. 3400-3411, 2012.

SILVA, B. J. V.; MORABITO, R.; YAMASHITA, D. S. Otimização na programação de montagens na indústria aeronáutica. **Gestão & Produção**, v. 21, p. 33-44, 2014.

SILVA, P. J.; VIEIRA, C. S.; SILVA, A. L. Modelagem e solução de problemas de sequenciamento em projetos com restrição de recursos. **Produto & Produção**, v. 18, n.1, p. 12-24, 2017.

SINGH, A. Resource constrained multi-project scheduling with priority rules & analytic hierarchy process. **Procedia Engineering**, v. 69, p. 725-734, 2014.

SILVA, P. J.; VIEIRA, C. S.; SILVA, A. L. Modelagem e solução de problemas de sequenciamento em projetos com restrição de recursos. **Produto & Produção**, v. 18, n.1, p. 12-24, 2017.

WANG, L.; ZHENG, X. A knowledge-guided multi-objective fruit fly optimization algorithm for the multi-skill resource constrained project scheduling problem. **Swarm and Evolutionary Computation**, v. 38, p. 54-63, 2018.

WOLSEY, L. A. **Integer programming**. New York: Wiley, 1998.

YAMASHITA, D. S.; MORABITO, R. Um algoritmo *branch-and-bound* para o problema de programação de projetos com custo de disponibilidade de recursos e múltiplos modos. **Gestão & Produção**, v. 14, n. 3, p. 545-555, 2007.