HANNA PAMPLONA HORTENCIO

TÉCNICAS DE PESQUISA OPERACIONAL APLICADAS AO PROBLEMA DE PROGRAMAÇÃO DE CIRURGIAS ELETIVAS

São Paulo
2019

HANNA PAMPLONA HORTENCIO

TÉCNICAS DE PESQUISA OPERACIONAL APLICADAS AO PROBLEMA DE PROGRAMAÇÃO DE CIRURGIAS ELETIVAS

Dissertação apresentada à Escola Politécnica da Universidade de São Paulo para obtenção do Título de Mestre em Ciências.

São Paulo
2019

HANNA PAMPLONA HORTENCIO

TÉCNICAS DE PESQUISA OPERACIONAL APLICADAS AO PROBLEMA DE PROGRAMAÇÃO DE CIRURGIAS ELETIVAS

Dissertação apresentada à Escola Politécnica da Universidade de São Paulo para obtenção do Título de Mestre em Ciências.

Área de Concentração:
Engenharia de Produção

Orientador:
Profª Drª Débora Pretti Ronconi

São Paulo
2019

Autorizo a reprodução e divulgação total ou parcial deste trabalho, por qualquer meio convencional ou eletrônico, para fins de estudo e pesquisa, desde que citada a fonte.

Este exemplar foi revisado e corrigido em relação à versão original, sob responsabilidade única do autor e com a anuência de seu orientador.

São Paulo, _____ de _____ de _____

Assinatura do autor: _____

Assinatura do orientador: _____

Catalogação-na-publicação

Hortencio, Hanna
Técnicas de Pesquisa Operacional aplicadas ao problema de programação de cirurgias eletivas / H. Hortencio -- versão corr. -- São Paulo, 2019.
99 p.

Dissertação (Mestrado) - Escola Politécnica da Universidade de São Paulo. Departamento de Engenharia de Produção.

1.pesquisa operacional 2.programação de atividades 3.programação de cirurgias I.Universidade de São Paulo. Escola Politécnica. Departamento de Engenharia de Produção II.t.

AGRADECIMENTOS

Primeiramente, agradeço aos meus pais, Carla e Ricardo, pela dedicação, apoio e amor incondicional. Por terem me encorajado a seguir meus sonhos mesmo que isso significasse mudar para uma nova cidade e ficar mais longe de casa. Agradeço à minha avó, Zélia, e à minha irmã, Alexa, que foram e sempre serão fonte de inspiração e fortaleza. Cada palavra de encorajamento, estímulo e carinho foram fundamentais para que me tornasse a pessoa que sou e chegasse até o cumprimento desse sonho.

Agradeço o apoio e carinho da minha família e amigos que estiveram comigo nesse caminho cheio de altos e baixos e que sempre me encorajaram para seguir firme. Cito, em especial, Victor Fontenele que foi meu companheiro e melhor amigo durante toda essa jornada. Por ter me dado forças nos momentos de exaustão e por ser minha família em São Paulo.

Expresso minha enorme gratidão à minha orientadora, Débora Pretti Ronconi, que não mediu esforços em contribuir para o meu crescimento acadêmico. Por ter compartilhado seu conhecimento e experiência, e por ter me ensinado o amor pela educação e pela pesquisa. E, por fim, pela amizade que foi construída nesses três anos.

Agradeço aos meus amigos da pós-graduação Augusto Molke, Bruna Carvalho, Fernanda Rocha, Lorena Abadia, Luis Felipe, Sérgio Bassi e Victória Multran pelo companheirismo e apoio nessa jornada. Com vocês até os dias de desânimo e esgotamento estavam recheados com risadas e felicidade. Vocês foram verdadeiros presentes que o mestrado me deu.

Um agradecimento especial aos meus colegas da Neolog, Luis Eduardo Coraini, Tomás Chaib e Willian Constâncio, que me ajudaram na reta final desse projeto. Vocês me deram o ânimo final que eu precisava para continuar e conseguir alcançar esse sonho. Muito obrigada.

Agradeço a todos os funcionários e professores do Departamento de Engenharia de Produção da Escola Politécnica da USP pela sua dedicação à educação e ao programa de pós-graduação. Um agradecimento especial a Lídia Silva que estava sempre disposta a ajudar e a compartilhar seus conhecimentos.

Por fim, agradeço aos mestres Abraão Saraiva Júnior, Anselmo Pitombeira e Maxweel Rodrigues que foram inspiração e incentivo na escolha de ingressar no mundo da acadêmico.

A todos, obrigada por permitirem que essa dissertação fosse realidade.

RESUMO

Atualmente, os hospitais se veem obrigados a melhorar sua produtividade. Os centros cirúrgicos, além de ser um dos setores com maiores custos, também é o que mais gera receita dentro de um hospital, dessa forma torna-se extremamente importante o gerenciamento eficiente desse setor. O métodos de otimização para programação de cirurgias podem ser usados como ferramentas para reduzir filas e ociosidade nos centros cirúrgicos, aumentando sua produtividade. O Problema de Programação de Cirurgias Eletivas com Múltiplos Recursos e Múltiplas Etapas consiste em alocar os recursos às etapas do processo cirúrgico dos pacientes, considerando as diferentes necessidades e rotas de cada paciente e, então, programar essas etapas no tempo respeitando a disponibilidade dos recursos e a sequência das etapas do processo cirúrgico dos pacientes. Esse problema é classificado na literatura como *NP-hard* e pode ser descrito como um *Job Shop Flexível* com *blocking* e função objetivo de minimização do número de pacientes não atendidos e do instante de término da última etapa, o *makespan*. O Objetivo desse trabalho é propor um modelo matemático e uma heurística construtiva para a resolução desse problema. O modelo matemático *Multi-Mode Blocking Job Shop* (MMBJS) apresentado em Pham e Klikert (2008) é explorado e algumas melhorias são apontadas neste trabalho. Um modelo matemático de Programação Linear Inteira Mista alternativo é proposto, a fim de reduzir o esforço computacional, ajustar o cálculo do *makespan* e sugerir uma estratégia de priorização de pacientes. Testes computacionais foram realizados, a fim de comparar o modelo MMJBS e o modelo proposto. Para instâncias em que todos os pacientes são atendidos, as soluções encontradas pelo CPLEX para ambos modelos são iguais, porém o tempo computacional necessário para encontrar uma solução ótima é em média 45% menor no modelo proposto. Também foram realizados testes computacionais com objetivo de observar o comportamento do modelo com diferentes configurações de recursos. Para instâncias com 15 pacientes, os testes apontam que o tempo computacional para encontrar a solução ótima é superior a 2h de processamento. Dessa forma, uma heurística construtiva é proposta, com objetivo de gerar soluções factíveis com pouco esforço computacional. A heurística proposta aloca cada etapa do tratamento de cada paciente aos recursos necessários, respeitando as janelas de disponibilidade dos recursos e buscando reduzir a folga no sistema. Um exemplo de aplicação da heurística construtiva é apresentado. As propostas para trabalhos futuros são apresentadas no capítulo final desta dissertação.

Palavras-Chave programação de atividades, cirurgia eletivas, *job shop flexível*, *makespan*, múltiplos recursos, problema em dois níveis, heurística construtiva, programação linear inteira mista.

ABSTRACT

For the past few years, hospitals have been forced to improve their productivity, with surgical centers being one of the sectors with higher costs within such organizations, but also the ones that generate the most revenue. Thus, optimization methods for surgical programming are tools that can be used to reduce queues and idleness in these sectors and consequently achieve the aforementioned goals. The "Problem of Programming Multiple Surgical Resources with Multiple Steps"consists in allocating the existing resources to each surgery stage that a patient will need to go through, considering the different needs, sequence and specificities of each of them, and then scheduling these steps in time. This type of problem is classified in the current literature as an NP-hard problem, being described as a Flexible Job Shop with blocking and an objective function that seeks to minimize the number of patients not served and the total makespan. The general purpose of this research is to propose a mathematical model and a constructive heuristic for this type problem. The proposed model explores the mathematical model Multi-Mode Blocking Job Shop (MMBJS) presented in Pham and Klikert (2008) suggesting improvements through the use of an alternative Mixed Integer Linear Programming that aims to: reduce the computational effort, adjust the makespan calculation and suggest a strategy of patients prioritization. In order to prove the benefits of the proposed enhancements, computational tests were performed to compare the MMJBS model and the proposed model, identifying that for instances where in which all patients are attended, the solutions found by CPLEX for both models are the same, but with a lower computational time the proposed model (45% average reduction). Also, other computational tests were performed to observe the behavior of the model with different configurations of resources. For instances with 15 patients, the tests indicate that the computational time to find the optimal solution is greater than 2 hours of processing. Thus a constructive heuristic is proposed, it aims to generate feasible solutions with little computational effort. The proposed heuristic allocates each surgery stage of a patient to the necessary resources, respecting the available windows and seeking to reduce the total slack in the system. An example of the application of the constructive heuristic is also presented. At last, future works proposals are presented in the final chapter of this dissertation.

Keywords scheduling, surgery, job shop flexible, makespan, multiple surgical resources, bilevel problem, constructive heuristic, mixed integer linear programming.

LISTA DE FIGURAS

1 Gráfico de *Gantt* para um exemplo do problema de programação de cirurgias. Fonte: Elaborado pela autora 4

2 Etapas do processo de pacientes eletivos. Fonte: Elaborado pela autora . . 6

3 Exemplo de programação de etapas com *blocking*. Fonte: Elaborado pela autora 7

4 Causas de atrasos nas Cirurgias. Fonte: Weinbroum, Ekstei e Ezri(2003) . 10

5 Exemplo representado por grafo do fluxo de pacientes no problema MMBJS. Fonte: Elaborado pela autora 15

6 Solução do exemplo representado por grafo do fluxo de pacientes no problema MMBJS. Fonte: Elaborado pela autora 16

7 Formação de módulos - Modelo MMBJS. Fonte: Elaborado pela autora . . 18

8 Gráfico *Gantt* do exemplo em Pham e Klinkert (2008) - Modelo MMBJS. Fonte: Elaborado pela autora 24

9 Gráfico *Gantt* do exemplo em Pham e Klinkert (2008) - Novo *makespan*. Fonte: Elaborado pela autora 27

10 Gráfico *Gantt* - função objetivo: número de pacientes atendidos. Fonte: Elaborado pela autora 38

11 Gráfico *Gantt* - função objetivo: número de pacientes atendidos com retirada dos tempos ociosos Fonte: Elaborado pela autora 38

12 Gráfico *Gantt* - função objetivo completa. Fonte: Elaborado pela autora . 39

13 Exemplo da utilização de movimentos de ajuste. Fonte: Elaborado pela autora. 66

14 Representação das opções de ociosidade encontrada no sistema. Fonte: Elaborado pela autora. 67

15 Exemplo para ilustrar o funcionamento da heurística construtiva - parte 1. Fonte: Elaborado pela autora. 69

16 Exemplo para ilustrar o funcionamento da heurística construtiva - parte 2. Fonte: Elaborado pela autora 70

17 Exemplo para ilustrar o funcionamento da heurística construtiva - parte 3. Fonte: Elaborado pela autora. 71

18 Representação dos instantes de disponibilidade do recurso e do paciente. Fonte: Elaborado pela autora. 75

19 Exemplo - regra de ordenação de recursos por distância. Fonte: Elaborado pela autora. 76

20 Representação em árvore dos conjuntos de regras com comparação com CR base. Fonte: Elaborado pela autora. 81

21 Gráfico das regras que utilizam o critério *Combinação* entre prioridade e flexibilidade. Fonte: Elaborado pela autora. 82

LISTA DE TABELAS

1 Exemplo de sistema bloqueado de cirurgias 7

2 Resumo dos principais trabalhos do nível operacional *offline* 13

3 Notações utilizadas no modelo MMBJS 17

4 Dados do exemplo (Pham e Klinkert 2008): módulos e recursos 22

5 Dados do exemplo (Pham e Klinkert 2008): tempos 23

6 Teste de sensibilidade de α com dados das Tabelas 4 e 5 28

7 Análise das restrições (3.3), (3.4), (3.8) e (3.9) com relação ao tempo computacional em segundos 29

8 Modelo proposto: modificações no modelo MMBJS 31

9 Notações modelo proposto. 33

10 Dados do exemplo problema em dois níveis: módulos e recursos. 37

11 Dados do exemplo problema em dois níveis: tempos. 37

12 Solução do modelo proposto considerando prioridade dos pacientes definindo a variável y_{ij}. 40

13 Solução do modelo proposto considerando prioridade dos pacientes definindo a variável w_i. 41

14 Número de pacientes por especialidade cirúrgica e por tipo de pós-cirurgia para cada conjunto de instâncias 44

15 Comparação entre os modelos MMBJS, MMBJS* e Proposto: solução. . . 45

16 Comparação entre os modelos MMBJS, MMBJS* e Proposto: tempos e *GAP*. 45

17 Resumo das características dos grupos de instâncias 47

18 Análise sensibilidade - Grupo 0. 48

19 Análise sensibilidade - Grupo 1.1. 50

20 Análise sensibilidade - Grupo 1.2. 51

21 Análise sensibilidade - Grupo 2.1. 53

22 Análise sensibilidade - Grupo 2.2. 54

23 Análise sensibilidade - Grupo 3.1. 55

24 Análise sensibilidade - Grupo 3.2. 57

25 Hierarquia para alocação de recursos. 59

26 Notações utilizanda na Heurística Construtiva. 61

27 Exemplo - combinação regras prioridade e flexibilidade. 73

28 Exemplo - regra de ordenação de recursos por distância não-decrescente . . 76

29 Exemplo - regra de ordenação de recursos por distância considerando o tempo de processamento. 77

30 Conjunto de regras de ordenação. 78

31 Comparação dos conjuntos de regras com base. 80

32 Comparação com solução ótima - Grupo 0 86

33 Comparação com solução ótima - Grupo 1.1 87

34 Comparação com solução ótima - Grupo 1.2 88

35 Comparação com solução ótima - Grupo 2.1 89

36 Comparação com solução ótima - Grupo 2.2 90

37 Comparação com solução ótima- Grupo 3.1 91

38 Comparação com solução ótima - Grupo 3.2 92

SUMÁRIO

1 Introdução **1**

2 O problema de programação de cirurgias **3**

2.1 Definição do problema 3

2.2 Objetivos 8

2.3 Revisão de literatura 8

3 Modelo proposto em Pham e Klinkert (2008) **14**

3.1 Formulação do problema 16

3.2 Exemplo 21

3.3 Análise da função objetivo 23

3.3.1 Cálculo do *makespan* 25

3.3.2 Definição do α 26

3.4 Análise das restrições 28

3.5 Comentários adicionais 29

4 Proposta de modelo matemático **31**

4.1 Formulação proposta 31

4.1.1 Função objetivo para o problema em dois níveis 36

4.1.2 Prioridade de pacientes 39

4.2 Resultados computacionais 42

4.2.1 Geração das instâncias 42

4.2.2 Avaliação comparativa das formulações 44

4.2.3 Análise de sensibilidade para o modelo proposto 46

5 Heurística construtiva **58**

5.1 Heurística proposta 58

5.1.1 Movimentos de ajuste 65

5.1.2 Exemplo 68

5.2 Regras de ordenação 71

5.2.1 Regras de ordenação de pacientes 71

5.2.2 Regras de ordenação de tipos de recursos 74

5.2.3 Regras de ordenação de recursos 75

5.3 Experimentos numéricos 77

5.3.1 Comparações entre conjuntos de regras 79

5.3.2 Comparação com o resultado ótimo 83

6 Conclusões **93**

Referências **96**

1 INTRODUÇÃO

Atualmente os hospitais, elementos críticos no sistema de saúde, vêm sentindo a necessidade de aperfeiçoar sua produtividade. O envelhecimento da população, o aumento da demanda por tratamento de doenças crônicas e os orçamentos para a saúde cada vez mais limitados (sejam eles públicos ou privados) são algumas das causas dessa necessidade (GUPTA; DENTON, 2008; ROUGH; MCDANIEL; ROUGH, 2010; DEMEULEMEESTER *et al.*, 2013; SAMUT; CAFRI, 2016). De acordo com Cardoen (2010) uma grande preocupação dos hospitais é reduzir os custos e melhorar seus resultados financeiros, além de aumentar o nível de satisfação dos seus pacientes.

O centro cirúrgico é o setor que gera mais receita em um hospital podendo chegar a 70% da receita total e é também um dos setores hospitalares com maiores custos operacionais (cerca de 40% dos custos de um hospital) (MOLINA-PARIENTE; FRAMINAN, 2009). O alto nível de atividade clínica e o considerável uso de pessoal, equipamento e tecnologia geram grandes sacrifícios financeiros (KAYE *et al.*, 2015). Filas, ociosidade e atrasos em cirurgias acarretam problemas nos estágios subsequentes, como as Unidades de Tratamento Intensivo (UTI) (PANNI *et al.*, 2013). Assim, é extremamente importante para os hospitais gerenciarem seus centros cirúrgicos de forma eficiente.

Segundo a Portaria 1101 do Ministério da Saúde do governo brasileiro (2002) a média necessária de leitos de UTI é cerca de 4% a 10% dos leitos disponíveis. Conforme Almeida (2006) o percentual de leitos estabelecido pela Organização Mundial de Saúde (OMS) é de 7% a 10%, mas o número de leitos ofertados pelos hospitais no Brasil não chega nem no mínimo, com menos de 2% nas regiões Norte e Nordeste.

Além da escassez de leitos de UTI, o número de cirurgias ofertadas pelos hospitais também não atende a população brasileira satisfatoriamente. Uma pesquisa realizada pelo Instituto Data Folha (2014) sobre o SUS no estado de São Paulo revelou que 63% dos entrevistados não estão satisfeitos com a saúde no Brasil, concedendo nota igual ou inferior a 50% para o sistema. Para o Conselho Federal de Medicina (2014) os números só revelam a fragilidade das políticas públicas e os gargalos na gestão da saúde da saúde do Brasil, já

conhecidas pelos profissionais que atuam nesse sistema. Um ponto preocupante também apresentado pela pesquisa é a espera para marcação de consultas e exames preliminares no sistema público de saúde. Cinco a cada dez entrevistados tiveram que esperar até seis meses para conseguir um horário e para 30% dos entrevistados a espera durou mais de seis meses.

O aumento da utilização dos centros cirúrgicos dependem de fatores como sequenciamento e programação apropriados, *setups* rápidos e cirurgias iniciadas com pontualidade (KAYE *et al.*, 2015). Conforme Gupta e Denton (2008), os modelos de sequenciamento e programação de pesquisa operacional, já utilizados em outras tipos de serviços como restaurantes e hotéis, podem ser aplicados em hospitais para obtenção de bons resultados. Dentre as ferramentas mais utilizadas na literatura para sequenciar e programar cirurgias estão simulação, programação matemática, heurísticas e meta-heurísticas (DEMEULEMEESTER *et al.*, 2013). Enquanto as funções objetivo mais presentes estão relacionadas com a utilização dos recursos hospitalares (HORTENCIO; RONCONI; MIYAKE, 2017).

Esse trabalho propõe o desenvolvimento de ferramentas de otimização para auxiliar na programação de cirurgias eletivas, aquelas que podem ser agendadas previamente, a fim de maximizar o número de pacientes atendidos e melhorar a utilização dos recursos disponíveis. Esse problema possui um alto grau de complexidade e, portanto, além do desenvolvimento do modelo matemático, serão elaborados métodos alternativos que buscarão fornecer resultados de qualidade com tempo de execução aceitáveis. Como, por exemplo, heurísticas construtivas.

Tendo em vista o cenário apresentado, os objetivos desse trabalho são: *i)* analisar a bibliografia do problema de programação de cirurgias eletivas; *ii)* propor um modelo matemático para o problema de programação de cirurgias eletivas em um ambiente *Jobshop* com múltiplas etapas e múltiplos recursos, e ainda, considerar que uma fração da etapa de recuperação (pós-cirurgia) possa ocorrer na sala de cirurgia e *iii)* propor uma heurística construtiva que forneça resultados de qualidade em tempo computacional aceitável.

A estrutura do presente trabalho é dividida em cinco partes. A caracterização do problema, os objetivos do trabalho e revisão bibliográfica são apresentados no Capítulo 2. O Capítulo 3 contém a análise do modelo matemático de programação linear inteira mista proposto por Pham e Klinkert (2008). No Capítulo 4 é apresentada uma proposta de formulação matemática do problema de programação de cirurgias eletivas. No Capítulo 5 é proposta uma heurística construtiva para a resolução do problema. E por fim, no Capítulo 6 são apresentadas as conclusões desse trabalho e propostas para trabalhos futuros.

2 O PROBLEMA DE PROGRAMAÇÃO DE CIRURGIAS

Nesse capítulo é descrito o problema de programação de cirurgias, apresentando as suas principais particularidades. Em seguida, os objetivos da dissertação são apontados e uma revisão de literatura é realizada acerca do tema.

2.1 Definição do problema

O problema de programação de cirurgias consiste em disponibilizar os recursos (médicos, enfermeiros, acomodações, equipamentos e instrumentos) às necessidades dos pacientes, quando e onde são necessários (HALL, 2012). Quatro níveis hierárquicos classificam esse problema: estratégico, tático, operacional *offline* e operacional *online* (DEMEULEMEESTER *et al.*, 2013; HANS; VANBERKEL, 2012). Para cada nível, Hans e Vanberkel (2012) exemplificam os tipos de problemas encontrados:

1. Estratégico: dimensionamento de centros cirúrgicos, equipamentos e pessoal, mix de serviços que define os tipos de cirurgias e volumes ofertados e dimensionamento de departamentos auxiliares, como enfermarias e UTI;

2. Tático: sequenciamento semanal de centros cirúrgicos, dividindo por especialidade ou por cirurgião, discute os impactos do planejamento de cirurgias eletivas nos departamentos subsequentes e conflitos na utilização de equipamentos;

3. Operacional *offline*: programação individual dos pacientes, considerando todos os recursos necessários e a sequência das atividades;

4. Operacional *online*: monitoramento e controle das atividades diárias, se necessários, ajustes são feitos e cirurgias são canceladas ou remarcadas.

O problema de programação de cirurgias considerando neste trabalho é classificado como operacional *offline*, uma vez que tem como objetivo alocar os recursos aos pacientes

e definir a programação das atividades de cada paciente individualmente. Dessa forma esse trabalho busca responder duas perguntas básicas de programação de atividades:

- Como *alocar* os recursos necessários as atividades dos pacientes?
- Como *programar* no tempo as atividades depois de alocados os recursos necessários?

Como resposta da primeira pergunta, um conjunto de recursos disponíveis são designados para cada etapa do procedimento cirúrgico. Na segunda, são definidos os instantes de início e término de cada uma das etapas, bem como das atividades de *setup* e limpeza, definindo assim a sequência dos pacientes no sistema. Na Figura 1 é apresentado o Gráfico de *Gantt* do resultado da programação de um exemplo com três pacientes (A, B e C). A primeira coluna indica as etapas (pré-cirurgia, cirurgia e pós-cirurgia), enquanto que na segunda coluna estão os recursos necessários para realizar cada etapa. A etapa de cirurgia neste exemplo utiliza os recursos *sala de cirurgia, cirurgião, anestesista* e *enfermeiro cirúrgico*. Após determinado quais recursos são necessários para cada uma das etapas, o processo de programação de cirurgias determina quais os instantes de início e término de cada atividade, levando em consideração as limitações de recursos e tempos de processamento dos pacientes.

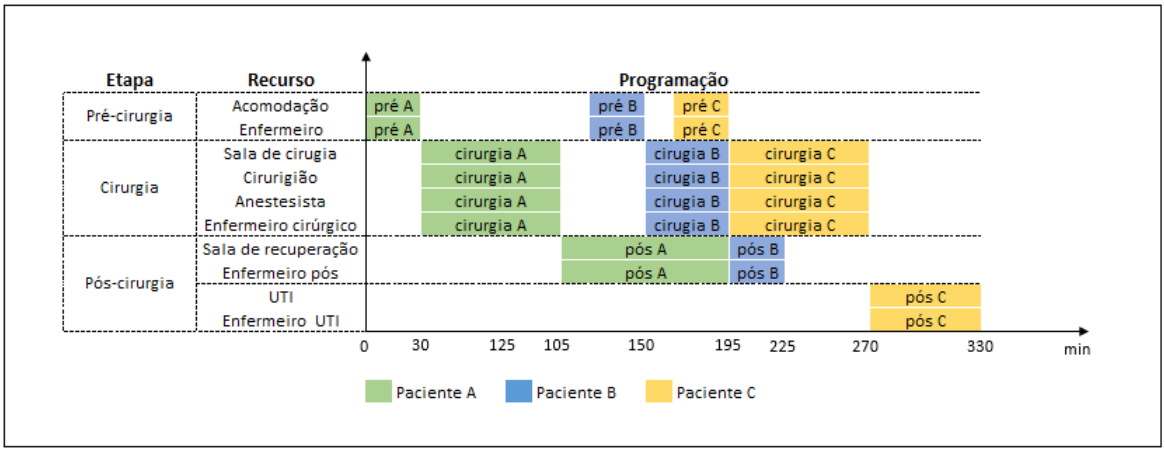

Figura 1: Gráfico de *Gantt* para um exemplo do problema de programação de cirurgias. Fonte: Elaborado pela autora

A literatura considera duas classes de pacientes: eletivos e não eletivos. A primeira classe diz respeito a pacientes cujas cirurgias podem ser agendadas previamente. Já a segunda classe, são pacientes cujas cirurgias são inesperadas e, portanto, não podem ser agendadas com antecedência (DEMEULEMEESTER *et al.*, 2013).

Os pacientes eletivos, por sua vez, são classificados conforme o tempo de permanência após a cirurgia: pacientes *ambulatoriais* ou pacientes *hospitalares*. Os pacientes *ambu-*

latoriais normalmente entram e saem do hospital no mesmo dia, pois são submetidos a cirurgias simples com procedimentos padronizados, como, procedimentos cirúrgicos minimamente invasivos. Os pacientes *hospitalares*, por outro lado, têm um pós-operatório mais prolongado (SAMUDRA *et al.*, 2016).

A classificação dos pacientes não eletivos é realizada de acordo com o tempo aceitável para iniciar o atendimento. Os pacientes cujo atendimento deve ser iniciado imediatamente são classificados como *emergência*, enquanto pacientes que devem ser admitidos em um curto período de tempo são chamados de *urgência*. Esse curto período de tempo normalmente é menor que o tempo necessário para desocupar um leito de pacientes eletivos (JITTAMAI; KANGWANSURA, 2011). Nesse sentido, a chegada e permanência de pacientes não eletivos é imprevisível e sua programação não será contemplada nesse trabalho.

Os pacientes eletivos, tanto ambulatorial como hospitalar, têm sua cirurgia agendada previamente e são instruídos de chegar algumas horas antes do horário da cirurgia para inciarem a etapa de *pré-cirurgia*. Nessa etapa, os pacientes, após darem entrada na recepção do hospital e se instalarem em uma acomodação, são atendidos por um enfermeiro que os orientam sobre a condução da cirurgia e garantem a realização das atividades recomendadas antes da cirurgia (por exemplo, fazer exames ou tomar medicamentos necessários para a cirurgia). Após esse processo o paciente está liberado para seguir para a etapa seguinte.

Posteriormente, o paciente é transportado até uma sala de cirurgia na qual devem estar todos os recursos materiais e humanos necessários para dar início à cirurgia. Então, o paciente é anestesiado pelo anestesista e um (ou vários) cirurgião(ões) iniciam o procedimento. Outros colaboradores também podem estar presentes, como enfermeiros-cirúrgicos e instrumentistas. Após o término da etapa de *cirurgia* o paciente é liberado para a *pós-cirurgia*.

A etapa de *pós-cirurgia* pode ser conduzida em dois possíveis ambientes: Sala de Recuperação (SR) ou na Unidade de Tratamento Intensivo (UTI). Pacientes críticos, como idosos, pessoas com problemas cardio-respiratórios ou que foram submetidas a cirurgias consideradas de risco têm, normalmente, a etapa de *pós-cirurgia* conduzida na UTI. Enquanto pacientes não críticos e pacientes eletivos ambulatoriais têm, em sua maioria, o pós-operatório realizado na SR.

Em alguns casos, entretanto, pacientes considerados não críticos podem apresentar complicações durante a *cirurgia* ou *pós-cirurgia* e pode ser solicitada a transferência da

SR para a UTI. Esse trabalho considera que no momento do agendamento é possível determinar os tempos de processamento de todas as três etapas, bem como onde ocorrerá a etapa de *pós-cirurgia*, desconsiderando assim esses casos.

Após a etapa de *pós-cirurgia* o paciente retorna para sua acomodação e é acompanhado por uma equipe até que receba alta e possa deixar o hospital. A Figura 2 resume o processo dos pacientes eletivos para o problema estudado. Em destaque estão as três etapas do processo cirúrgico que serão consideradas na programação de cirurgias nesse trabalho: *pré-cirurgia*, *cirurgia* e *pós-cirurgia*.

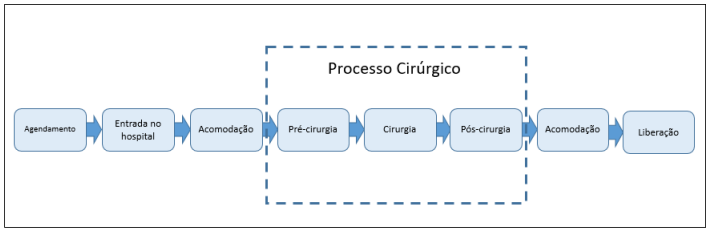

Figura 2: Etapas do processo de pacientes eletivos. Fonte: Elaborado pela autora

Para cada etapa do processo cirúrgico é necessário um conjunto de recursos humanos e/ou materiais que deve estar disponível durante todo tempo de processamento da etapa. As três etapas do processo cirúrgico são consideradas *etapas com bloqueio* (do inglês, *blocking*), visto que quando um recurso é utilizado por uma etapa, este não é liberado até que o paciente seja transferido para a próxima etapa (PHAM; KLINKERT, 2008).

As consequências de um sistema ter etapas com *blocking* são ilustradas na Figura 3. Considere a programação das etapas de *pré-cirurgia*, *cirurgia* e *pós-cirurgia* de três pacientes (A, B e C). Na primeira coluna temos as etapas do processo cirúrgico, já na segunda, são apresentados os recursos utilizados em cada uma. A etapa de pós-cirurgia dos pacientes A e B é realizada na Sala de Recuperação (SR), enquanto essa etapa do paciente C deve ser realizada na Unidade de Tratamento Intensivo (UTI). Observe que o Paciente A entra no sistema no instante zero, é atendido utilizando o recurso *Acomodação* e *Enfermeiro* e sem espera, entra na etapa seguinte (30 min). Após 75 minutos de *cirurgia*, o paciente inicia a etapa de *pós-cirurgia* também sem espera, e deixa o sistema no instante 195 (min). O Paciente B, por sua vez, entra no sistema no instante 75 (min) e após ser atendido na *pré-cirurgia* é encaminhado para a etapa de *cirurgia* no instante 105 (min). Após 45 minutos a cirurgia termina (150 min), mas o Paciente B não pode ser transferido para a próxima etapa, pois o Paciente A ainda está utilizando os recursos *SR* e *Enfermeiro SR*. Portanto todos os recursos utilizados na etapa de *cirurgia* do Paciente B permanecem com *blocking* até que o paciente possa ser liberado para a *pós-cirurgia* (195 min), quando

Tabela 1: Exemplo de sistema bloqueado de cirurgias

Salas cirúrgicas	Segunda-feira	Terça-feira	Quarta-feira	Quinta-feira	Sexta-feira
Sala 1	Ortopédica	Geral	Ortopédica	Livre	Oftalmológica
Sala 2	Plástica	Geral	Oftalmológica	Plástica	Livre
Sala 3	Livre	Ortopédica	Plástica	Ortopédica	Geral

o Paciente A sai do sistema.

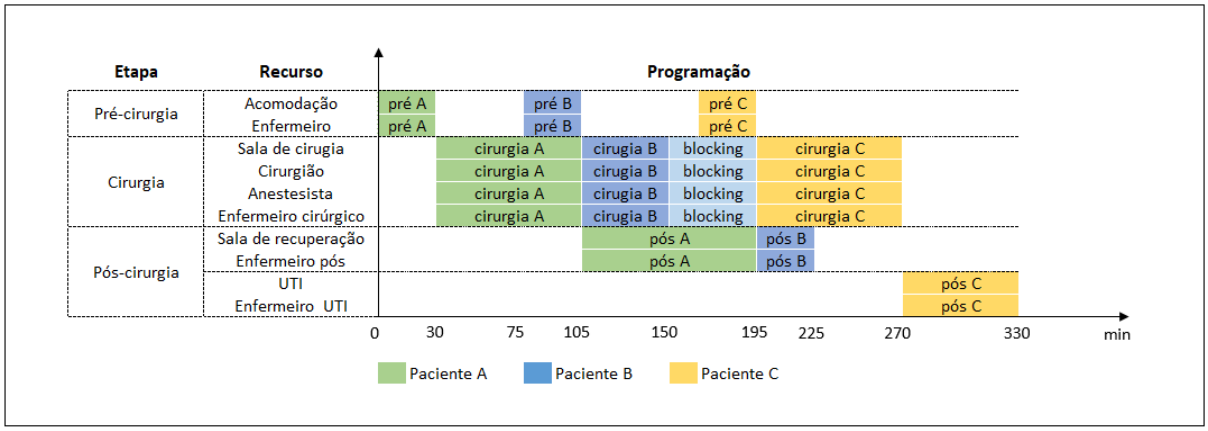

Figura 3: Exemplo de programação de etapas com *blocking*. Fonte: Elaborado pela autora

Dois tipos de sistemas são considerados na literatura de programação de cirurgias: *sistemas bloqueados* e *não bloqueados*. Nos *sistemas bloqueados* as salas de cirurgia são organizadas em blocos, ou seja, alguns períodos são reservados para uma especialidade, um cirurgião ou um conjunto de cirurgiões. Em *sistemas não bloqueados* não existe nenhum tipo de reserva, assim qualquer especialidade ou cirurgião pode agendar sua cirurgia em qualquer período que haja disponibilidade da sala de cirurgia. Esta dissertação utilizará o modelo de *sistema bloqueado* restringindo os horários das salas cirúrgicas somente às especialidades que reservaram no período. A definição dos horários de cada especialidade é realizada no nível tático, através do Planejamento Mestre de Produção e não faz parte do escopo desse trabalho, mas será utilizado como dado de entrada. A Tabela 1 apresenta um exemplo de um *sistema bloqueado* de cirurgias.

Para problema de programação de cirurgias várias funções objetivos são encontradas na literatura. Algumas são comuns a problemas de *scheduling* em geral, como minimização do *makespan*, instante de término da última atividade, e minimização das horas extras, enquanto outras são características do ambiente hospitalar, como tempo de espera e preferência dos pacientes e funcionários do hospital, além de número de pacientes atendidos no horizonte de tempo utilizado. Conforme a revisão de literatura de Hortencio, Ronconi

e Miyake (2017) cerca de 78% dos trabalhos entre os anos de 2000 e 2017 que foram analisados utilizam funções objetivo que tratam da capacidade de gerenciar os recursos hospitalares eficientemente, enquanto 32% importam-se com a percepção de qualidade do paciente e somente 22% se preocupam com as preferências dos funcionários do hospital.

2.2 Objetivos

O objetivo principal dessa dissertação é discutir o tema programação de cirurgias eletivas considerando três etapas com *blocking*, no nível operacional *offline* em um sistema bloqueado. A função objetivo considerada busca prioritariamente minimizar o número de pacientes indeferidos, ou seja, maximizar o número de pacientes atendidos pelo sistema no horizonte de tempo e reduzir o *makespan* alcançando assim uma melhor utilização dos recursos disponíveis. Não é objetivo desse trabalho definir os tipos de cirurgias que sã oferecidos pelo hospital, as especialidades dos colaboradores, ou disponibilidade dos recursos humanos e materiais. Esses dados são informações de entrada para que a programação aloque os recursos aos pacientes e sequencie as atividades. Nesse contexto, são propostos e analisados um modelo Programação Linear Inteira Mista (PLIM) e um método de solução heurística.

2.3 Revisão de literatura

Nessa seção, uma breve revisão da literatura é realizada abordando os principais trabalhos de programação de cirurgias eletivas encontrados na literatura que utilizam como estratégia de solução modelagem matemática, métodos exatos, métodos heurísticos ou simulação.

O problema de programação de cirurgias faz parte de um problema mais amplo chamado *programação do processo cirúrgico* apresentado no trabalho clássico de Magerlein e Martin (1978). Em seu estudo, os autores dividem o problema em duas etapas: *programação avançada* e *procedimento de alocação* (ou programação de cirurgias). A primeira etapa preocupa-se em determinar qual dia o paciente é atendido, já a segunda, os recursos necessários são alocados e os instantes de início e término são determinados. O *procedimento de alocação* está dentro do escopo do problema estudado nesse trabalho e encontra-se no nível operacional *offline*.

Apesar da etapa de *programação avançada* não o objeto de estudo deste trabalho, ela

tem sido estudada amplamente na literatura como o problema de Planejamento Mestre de Cirurgias do nível tático. Banditori, Cappanera e Visintin (2014) desenvolvem um modelo de Programação Inteira Mista (PLIM) para balancear a utilização das salas de cirurgia e dos leitos pós-cirúrgicos para esse problema. A robustez do modelo é testada utilizando simulação por eventos discretos. Outro problema do nível tático pode ser encontrado no trabalho dos autores Ceschia e Schaerf (2016) que utilizam uma heurística de busca em vizinhança para resolver o Problema de Admissão de Pacientes buscando maximizar a utilização das salas de cirurgia.

Outros problemas não considerados no escopo desse trabalho estão no nível estratégico. Um exemplo é o Mulholland, Abrahamse e Bahl (2005) que busca determinar uma solução ótima para o problema de *Mix* de Cirurgias utilizando Programação Linear (PL), a fim de maximizar a receita do hospital e dos cirurgiões. Em Ma *et al.* (2011) um modelo PLIM é desenvolvido e o método *branch-and-price* é utilizado para resolver este mesmo problema. A função objetivo proposta é de minimização de custos respeitando a capacidade de recurso na etapa de cirurgia.

No nível operacional *offline*, a literatura em Programação de Cirurgias divide os trabalhos de acordo com o sistema de reserva das salas cirúrgicas: *sistemas bloqueados* e *não bloqueados*. Em Fei, Meskens e Chu (2007) um sistema bloqueado é modelado a fim de minimizar os custos de horas extras do centro cirúrgico. Os cirurgiões são considerados generalistas e os leitos de pós-cirurgia sempre estão disponíveis. Em outro trabalho, os mesmos autores Fei, Meskens e Chu (2010) trabalham com um sistema não bloqueado utilizando a mesma função objetivo. Ambos modelos possuem duas fases de resolução, na primeira, utiliza-se uma heurística baseada no método de Geração de Colunas e na segunda, Algoritmos Genéticos. Dexter *et al.* (2002) desenvolvem um modelo de PL para determinar como as mudanças na alocação dos cirurgiões em um período, sem alterar o número de horas disponíveis, pode interferir nos custos. Através dos resultados, observou-se que os custos variáveis podem aumentar em até 34%, indicando, assim, que sistemas bloqueados são mais adequados a hospitais que buscam reduzir os custos sem impactar negativamente na qualidade do atendimento.

Weinbroum, Ekstein e Ezri (2003) investigam quais os motivos que levam a baixa eficiência nas salas de cirurgia. Na Figura 4 apresentam-se as principais causas dos atrasos em cirurgias. Observa-se que cerca de 58,5% dos atrasos são causados pela indisponibilidade das salas de cirurgia e 7,4% pela indisponibilidade dos cirurgiões, sendo assim a etapa de cirurgia é responsável por quase 66% dos atrasos. As demais causas são relativas as atividades adjacentes ao centro cirúrgico, sendo 12,1% geradas por atrasos no atendi-

mento pré-cirúrgico e 9,8%, por UTI congestionadas. Nesse contexto, Pham e Klinkert (2008) introduzem um modelo de PLIM que considera, além da etapa de cirurgia, duas etapas adjacentes: pré-cirurgia e pós-cirurgia. Atacando, assim, cerca de 88% das causas de atrasos nas cirurgias. Procurando aproximar-se da realidade dos hospitais, o modelo também permite a inclusão de quantos recursos forem necessários em cada uma das etapas e ainda considera suas especialidades. Por exemplo, para uma cirurgia ortopédica podem ser incluídos os recursos: anestesista, sala de cirurgia, cirurgião ortopédico e aparelho de radioscopia. A fim de simplificar a fase de *alocação de recursos*, o modelo recebe como dado de entrada os conjuntos de recursos (chamados módulos), suas disponibilidades e suas conexões com pacientes e etapas.

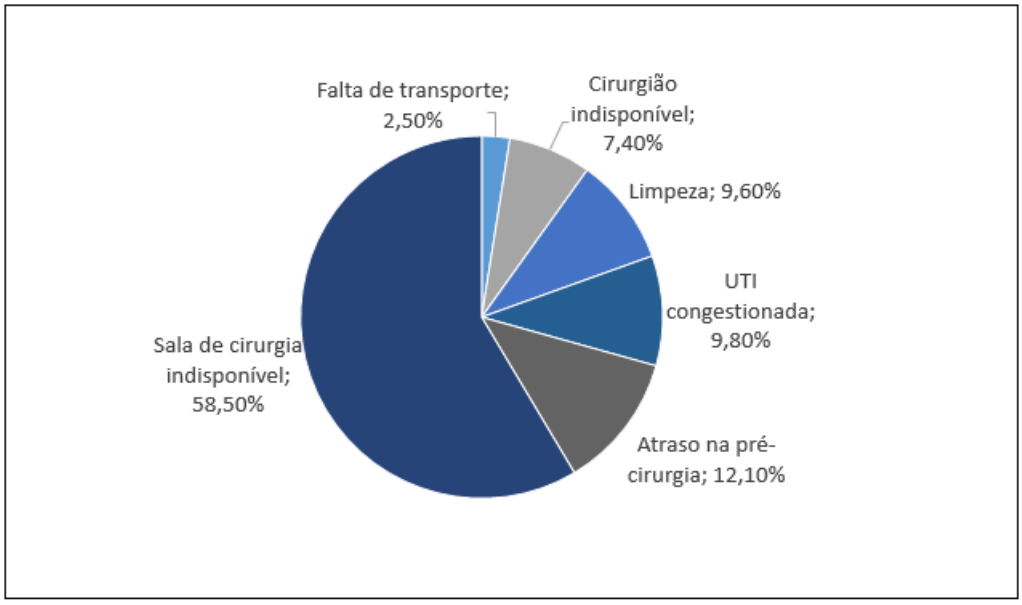

Figura 4: Causas de atrasos nas Cirurgias. Fonte: Weinbroum, Ekstei e Ezri(2003)

A seguir, são apresentados trabalhos de programação de cirurgias que consideram diversos níveis de complexidade, desde um único recurso e etapa até trabalhos com várias etapas e recursos. Vale ressaltar que a literatura de programação de cirurgias com múltiplos recursos e múltiplas etapas é relativamente escassa.

O trabalho dos autores Conforti, Guerriero e Guido (2010) apresenta um modelo multiobjetivo que além de programar cirurgias para pacientes eletivos ambulatoriais, define a distribuição ótima das especialidade em um centro cirúrgico. Abordando, assim, simultaneamente problemas do nível tático e operacional *offline*. A programação de cirurgias é baseada nas prioridades clínicas dos pacientes e considera uma única etapa e um único recurso, chamado time cirúrgico, na programação.

Fei *et al.* (2008) estudam o problema de programação de cirurgias considerando

uma única etapa que utiliza como recurso uma sala de operação multi-funcional com objetivo de minimizar o custo operacional total. Os mesmos autores em Fei *et al.* (2009) desenvolvem um modelo que otimiza o planejamento tático de um centro de cirurgias endoscópicas utilizando uma heurística baseada no método de Geração de Colunas e, em seguida, usando uma adaptação do algoritmo exato proposto em Gonzalez e Sahni (1979) define a programação dos pacientes. Nesse trabalho também considera-se a cirurgia como única etapa, porém dois tipos de salas de cirurgias especializadas são utilizadas conforme a necessidade do paciente.

Testi e Tànfani(2009) desenvolvem um modelo de PLIM que além de definir a programação das cirurgias (nível operacional), também determina a reserva de blocos de períodos para cada especialidade em um sistema bloqueado (nível tático). Duas etapas são consideradas, cirurgia e pós-cirurgia. Os pacientes são alocados às salas de cirurgias, que admite-se ter todos os recursos necessários como cirurgiões, enfermeiros e anestesistas, e o número de pacientes atendidos no período não pode ser superior ao número de leitos de UTI disponíveis. A função objetivo utilizada busca maximizar o bem-estar do paciente, que é medido a partir de várias variáveis relacionadas com o tempo de espera.

Em seu trabalho, Cardoen, Demeulemeester e Beliën (2009a) desenvolvem um modelo de programação de cirurgias em um centro de cirurgias ambulatoriais considerando duas etapas: cirurgia e pós-cirurgia com um recurso cada. Seis elementos compõem a função objetivo do problema, entre eles: priorizar crianças e pacientes graves e minimizar os picos de utilização das salas de recuperação. Os mesmos autores em Cardoen, Demeulemeester e Beliën (2009b) também desenvolvem um trabalho que busca programar procedimentos cirúrgicos menos invasivos utilizando uma abordagem *branch-and-price* e os mesmos seis elementos na função multiobjetivo. Além das etapas cirúrgica e pós-cirúrgica, o procedimento de limpeza do centro cirúrgico para pacientes com doenças infecciosas também é contemplado. Em ambos trabalhos, dados coletados em um hospital na Bélgica são utilizados para testar o modelo desenvolvido.

Perdomo, Augusto e Xie (2008) modelam um problema de programação de cirurgia que considera duas etapas (cirurgia e pós-cirurgia) e dois recursos (sala de cirurgia e leito de pós-cirurgia) buscando minimizar o instante de término do atendimento de todos os pacientes. Um modelo de programação dinâmica foi utilizado para resolver subproblemas encontrados após a aplicação do método de Relaxação Lagrangeana, como resultado, os métodos apresentaram bons limitantes inferiores para o modelo. No trabalho subsequente, os autores Augusto, Xie e Perdomo (2010) consideram as mesmas etapas e recursos, mas permitem que a etapa pós-cirúrgica seja realizada na sala de cirurgia ou no leito de pós-

cirurgia. Ou seja, admitem que se não houver leitos de pós-cirurgia disponíveis, o paciente pode realizar parte da etapa de pós-cirurgia utilizando a sala de cirurgia.

Em seu trabalho os autores Gartner e Kolisch (2014) desenvolvem um modelo de programação de cirurgias baseado no modelo de programação de tarefas no ambiente *Job Shop Flexível* no qual, os pacientes são avaliados e enquadrados em grupos. O objetivo do modelo é definir fluxos diferentes para cada grupo de pacientes de forma a maximizar a contribuição marginal de cada paciente. Várias etapas são consideradas nesse trabalho que, apesar de ser classificado como operacional *offline*, possui um nível de detalhes menor que os trabalhos Pham e Klinkert (2008) e Augusto, Xie e Perdomo (2010).

Riise, Mannino e Burke (2016) propõem em seu trabalho um modelo genérico para o problema de programação de cirurgias que utiliza uma extensão do modelo clássico de grafos disjuntivos para problemas de programação de projetos com recursos limitados (em inglês, *resource-constrained project scheduling problem* - RCPSP). O modelo considera multi-recursos e multi-etapas buscando atender as várias particularidades do problema de programação de cirurgias encontradas nos hospitais. A função multiobjetivo possui quatro termos que são normalizados e ponderados segundo as necessidades do hospital, sendo portanto um método de otimização multiobjetivo à *priori*. A utilização desse método pode ser levar a resultados diferentes, dependendo dos valores de ponderação (T'KINDT; BILLAUT, 2006).

A partir da revisão de literatura apresentada e resumida na Tabela 2, pode-se observar que uma grande parte dos trabalhos de programação de cirurgias no nível operacional *offline* não considera múltiplos estágios e múltiplos recursos. Além disso, existe uma quantidade limitada de trabalhos que contemplam a etapa de pré-cirurgia, causa de 12,1% dos atrasos nas cirurgias. Nesta pesquisa vamos aprimorar e ampliar o trabalho de Pham e Klinkert (2008) que considera múltiplos recursos e múltiplas etapas (pré-cirurgia, cirurgia e pós-cirurgia) propondo um modelo matemático de Programação Linear Inteira Mista (PLIM) alternativo para o problema em dois níveis. Adicionalmente uma heurística construtiva é proposta explorando as particularidades do ambiente *Job Shop Flexível*.

Tabela 2: Resumo dos principais trabalhos do nível operacional *offline*

Trabalhos	Número de etapas	Pré cirurgia	Múltiplos recursos	Método/ Metodologia
Conforti, Guerriero e Guido (2010)	1			Modelo matemático multiobjetivo
Fei *et al. (2008)*	1			Método com duas fases heurísticas
Fei *et al. (2009)*	1		x	Método com duas fases heurísticas
Fei, Meskens e Chu (2007)	2			Método com duas fases heurísticas
Fei, Meskens e Chu (2010)	2			Método com duas fases heurísticas
Testi e Tànfani (2009)	2			Modelo matemático PLIM
Cardoen, Demeulemeester e Beliën (2009a)	2			Modelo matemático multiobjetivo
Cardoen, Demeulemeester e Beliën (2009b)	2			Método com duas fases heurísticas
Perdomo, Augusto e Xie (2008)	2		x	Modelo de programação dinâmica
Augusto, Perdomo e Xie (2010)	2		x	Modelo de programação dinâmica
Pham e Klinkert (2008)	3	x	x	Modelo matemático PLIM
Gatner e Kolish (2014)	várias	x		Modelo matemático PLIM
Riise, Mannino e Burke (2016)	várias	x	x	Método heurístico

3 MODELO PROPOSTO EM PHAM E KLINKERT (2008)

Este capítulo tem como objetivo apresentar e analisar o modelo matemático desenvolvido pelos autores (PHAM; KLINKERT, 2008) para o problema de programação de cirurgias eletivas denominado *Multi-Mode Blocking Job Shop* (MMBJS). Os autores descrevem o problema como um ambiente de máquinas *Job Shop Flexível* com três etapas (pré-cirurgia, cirurgia e pós-cirurgia), no qual a última etapa ocorre em dois possíveis locais: unidade de tratamento intensivo (UTI) ou sala de recuperação (SR). Para Pinedo (2012), o *Job Shop* é um ambiente com m máquinas onde cada tarefa j possui seu próprio roteiro. Cada tarefa é composta por uma sequência de operações a serem executadas. Quando uma operação pode ser realizada por mais de uma máquina, este ambiente é denominado *Job Shop Flexível*. Fazendo uma analogia ao problema estudado, as tarefas e suas operações correspondem aos pacientes e as etapas do processo cirúrgico que serão submetidos, respectivamente. Enquanto, as máquinas representam os conjuntos de recursos disponíveis para processar uma etapa do processo cirúrgico de um paciente.

A Figura 5 é uma representação de um exemplo utilizando um grafo disjuntivo G=(N,A,E), onde N é o conjunto de nós que representam todas as operações, A é o conjunto de arcos conjuntivos e E é o conjunto de arcos disjuntivos (BALAS, 1969). Neste grafo, os arcos conjuntivos definem a precedência entre as operações das tarefas e os arcos disjuntivos, por sua vez, não têm orientação e indicam todas as possíveis opções de precedência em cada máquina. A escolha da direção dos arcos disjuntivos estabelece a ordem de execução das atividades nas máquinas.

No exemplo, o grafo é composto por sete máquinas e três tarefas com três operações cada. Os nós 0 e 10 são nós fictícios que indicam o início e fim da programação. Como mais de uma máquina pode ser utilizada para processar cada uma cadas operações, esse ambiente é dito *Flexível*. Os arcos conjuntivos em preto indicam a ordem das operações para cada tarefa. Do exemplo, a sequência de operações das tarefas são 1-2-3, 4-5-6 e 7-8-9. Adicionalmente, os arcos coloridos apresentam os arcos disjuntivos que indicam

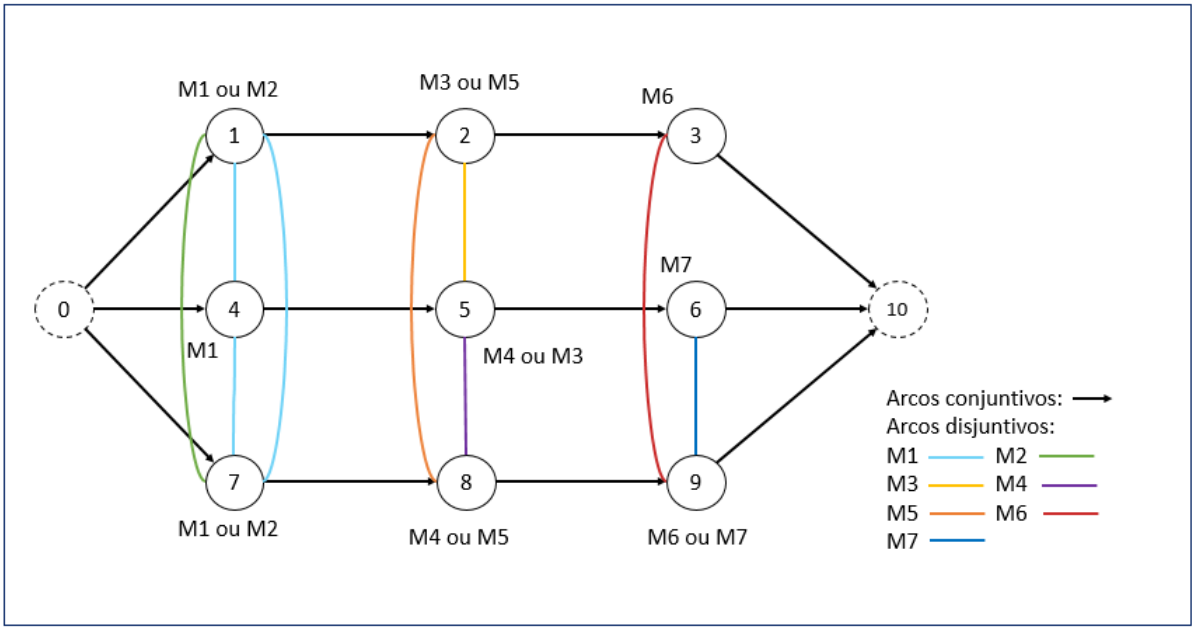

Figura 5: Exemplo representado por grafo do fluxo de pacientes no problema MMBJS. Fonte: Elaborado pela autora

todas as opções de ordens de operações possíveis em cada máquina. Uma solução para esse exemplo é apresentada na Figura 6, onde a máquina 1 processa as operações seguindo a ordem 1-7-4, as máquinas 3, 4, 5 e 6 processam, respectivamente, as operações 2, 5, 8 e 3, e a máquina 7 processa primeiro a operação 6 para então processar a operação 9.

Para o problema tratado nesse trabalho, os pacientes são equivalentes às tarefas do problema clássico e, consequentemente, as operações correspondem às etapas do processo cirúrgico. Para a execução de cada etapa é necessário um conjunto de recursos diferentes (por exemplo, médicos, enfermeiros, leitos) que são agrupados em módulos, análogos às máquinas na representação em grafo do ambiente *Job Shop Flexível*. Os módulos consistem em um conjunto de recursos que estão disponíveis em um mesmo período de tempo e são capazes de processar uma etapa do atendimento de um determinado paciente. Portanto, é possível que mais de um módulo esteja apto a processar uma etapa (múltiplos módulo ou *multi-mode*), contudo somente um módulo pode ser escolhido para processá-la.

Quando uma etapa é iniciada todos os recursos que compõem o módulo ficam impedidos de atender outro paciente até que a etapa seja concluída e o paciente atendido seja transferido para a próxima etapa ou saia do sistema, esse processo é chamado de *blocking*. O *blocking* ocorre para impedir que o paciente fique aguardando entre duas etapas do processo cirúrgico sem estar alocado a algum módulo.

A função objetivo adotada nesse modelo busca maximizar o número de pacientes aten-

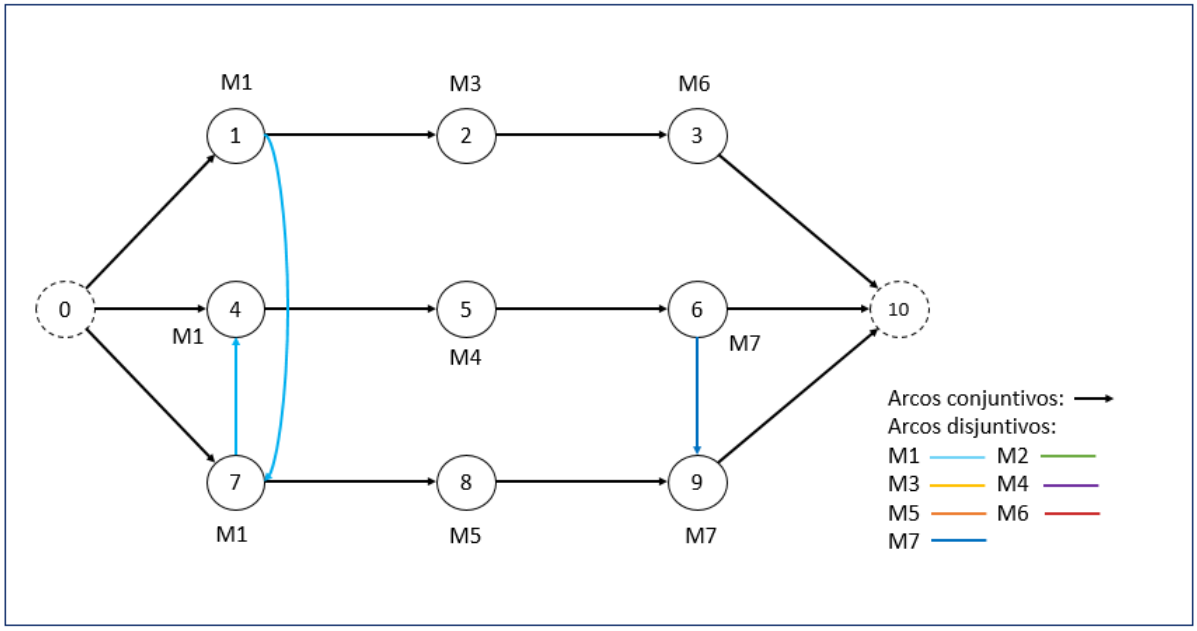

Figura 6: Solução do exemplo representado por grafo do fluxo de pacientes no problema MMBJS. Fonte: Elaborado pela autora

didos e minimizar o *makespan*, de forma a maximizar a utilização dos recursos disponíveis. Portanto trata-se de uma função objetivo de minimização com dois termos: *makespan* e somatório do instante de início das atividades de cada paciente. O modelo matemático de Pham e Kinkert (2008) é apresentado a seguir.

3.1 Formulação do problema

Nessa seção é apresentado o modelo de Programação Linear Inteira Mista (PLIM, do inglês *mixed integer linear programming - MILP*) para programação de cirurgias eletivas como um problema *MMBJS*. A Tabela 3 apresenta as notações utilizadas pelos autores.

O conjunto $\Im$ contém todos os pacientes disponíveis no instante zero para agendamento. Para cada paciente $J \in \Im$ existe um conjunto de etapas do atendimento (operações), o presente trabalho define três etapas para cada paciente: pré-cirurgia, cirurgia e pós-cirurgia. Assim, matematicamente um paciente representa um conjunto de etapas $J = (J_1, J_2, ..., J_k, ..., J_{|J|})$, no qual a operação J_k representa a k-ésima operação do paciente J. O conjunto de operações (i) de todos os pacientes é denominado I. O conjunto de duas operações consecutivas ($i = J_k$ e $j = J_{k+1}$) de um mesmo paciente J é definido como O_J. O paciente ao qual a operação i pertence é chamado de J^i.

O conjunto M representa todos os recursos necessários para realizar o atendimento

Tabela 3: Notações utilizadas no modelo MMBJS

Conjuntos	
$\Im$	Conjunto de pacientes
I	Conjunto de operações
O_j	Conjunto de pares consecutivos de operações do paciente $J \in \Im$
M	Conjunto de recursos
M_i^r	Conjunto de recursos contidos no módulo r que atendem a operação $i \in I$, $r \in R_i$, $M_i^r \subseteq M$
R	Conjunto de índices dos módulos, $R = \cup_{i \in I} R_i$
R_i	Conjunto de índices de módulos possíveis para a operação $i \in I$

Parâmetros	
p_i^r	Tempo de processamento da operação i no módulo r, $i \in I, r \in R_i$
p_i^{su}	Tempo de *setup* antes da operação $i \in I$
p_i^{cl}	Tempo de limpeza depois da operação $i \in I$
b_i	Tempo máximo de espera permitido para a operação $i \in I$ antes do paciente mover para a próxima operação
e^r	Início do intervalo de disponibilidade do módulo $r \in R$
f^r	Fim do intervalo de disponibilidade do módulo $r \in R$
H	Número grande $H = \sum_{i \in I} \max_{r \in R_i}\{p_i^r\} + \max_{r \in R_i, i \in I}\{p_i^r\} + \max_{i \in I}\{p^s u_i\} + \max_{i \in I}\{p^c l_i\}$
α	Fator de peso pequeno
J^i	Paciente ao qual a operação $i \in I$ pertence

Variáveis de decisão	
z_i^r	1, se o módulo r é designado à operação $i \in I$; 0, caso contrário
y_{ij}	1, se a operação j é processada depois da operação i; 0, caso contrário $i, j \in I$, $i < j$
x_i	Instante de início da operação $i \in I$
l_i	Instante de fim da operação $i \in I$
τ	*Makespan*

dos pacientes. Cada etapa de atendimento de um paciente necessita de um conjunto diferente de recursos para sua execução. Por exemplo, na pré-cirurgia o conjunto de recursos poderia ser composto por um enfermeiro e um leito na enfermaria, enquanto na cirurgia os recursos podem ser um enfermeiro, um cirurgião, um anestesista e uma sala de cirurgia.

Um conjunto de recursos que atendem a necessidade de uma operação e estão disponíveis em um mesmo período é denominado módulo. Na Figura 7 é apresentado um exemplo de formação de módulos constituídos por três recursos: um enfermeiro, um cirurgião e uma sala de cirurgia. Cada recurso pode estar disponível em períodos diferentes. A disponibilidade do módulo representa o período em que todos os recursos que o compõe estão disponíveis. Por exemplo, o módulo 2 é composto pelo enfermeiro 2, cirurgião e sala de cirurgia e está disponível das 12h às 19h ,que representa o período em que todos os recursos estão disponíveis. Módulos que compartilham o mesmo recurso não podem ser utilizados no mesmo período por dois pacientes. Do exemplo, os módulos 1 e 3 não podem ser utilizados por dois pacientes entre os instantes 7h e 12h, porque compartilham os recursos cirurgião e sala de cirurgia. Para a formação dos módulos, leva-se em consideração o Planejamento Mestre de Cirurgias que determina para quais especialidades e/ou cirurgiões a sala de cirurgia está reservada.

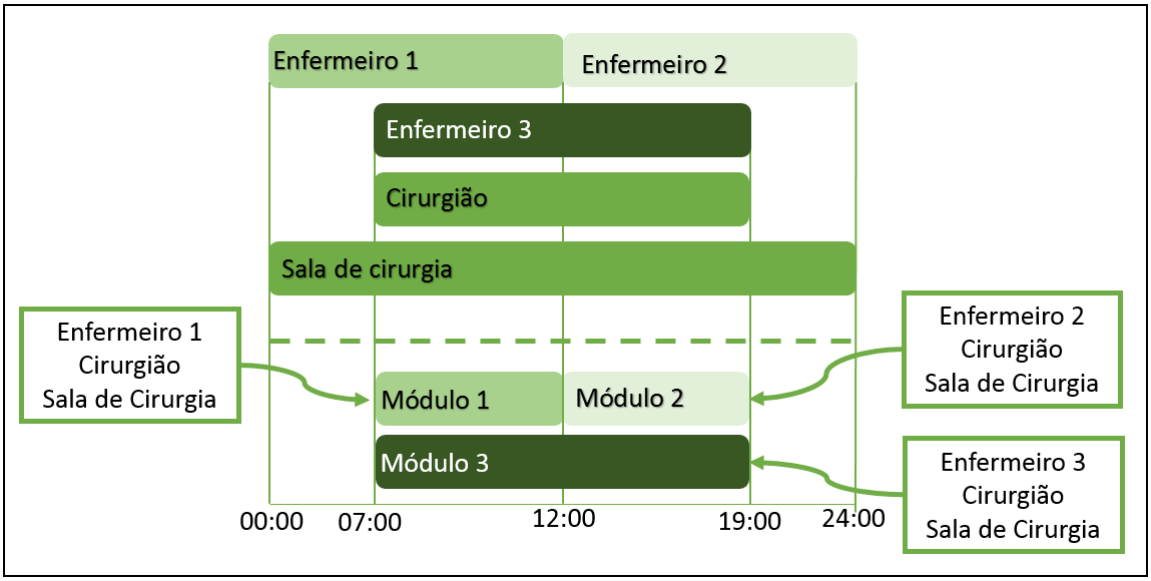

Figura 7: Formação de módulos - Modelo MMBJS. Fonte: Elaborado pela autora

O conjunto dos recursos que podem processar a operação i é definido como $M_i^r \in M$, onde r pertence ao conjunto dos índices dos módulos que podem processar a operação i ($r \in R_i$). Os pacientes não atendidos são processados pelo módulo fictício, ou seja, os pacientes que o modelo não consegue programar ficam em um módulo fictício que

não consome nenhum recurso do sistema. O conjunto de todos os índices dos módulos, incluindo o fictício, é definido por R. Por exemplo, considere quatro recursos $M = \{m_1, m_2, m_3, m_4\}$, duas operações $I = \{i_1, i_2\}$ e quatro módulos $R = \{1, 2, 3, 4\}$ descritos como $M_1^1 = \{m_1, m_2\}$, $M_1^2 = \{m_2, m_3\}$, $M_2^3 = \{m_4\}$ e $M_1^4 = M_2^4 = \{\}$. Assim, os conjuntos $R_1 = \{1, 2, 4\}$ e $R_2 = \{3, 4\}$ representam os índices de módulos possíveis para as operações 1 e 2, respectivamente. O módulo indexado como 4 representa o módulo fictício e não utiliza nenhum recurso. Todas as operações podem ser processadas pelo módulo fictício.

Cada operação $i \in I$ tem como parâmetros o tempo de processamento (p_i^r), que é dependente do módulo a ser utilizado, o tempo de *setup* (p_i^{su}), o tempo de limpeza (p_i^{cl}) no final da operação e o tempo máximo de espera após uma operação antes que o paciente seja transferido para a próxima operação (b_i). Os módulos ficam disponíveis em um intervalo indicado por $[e^r, f^r]$, na qual $0 \leq e^r \leq f^r$, que representa a janela de tempo no qual todos os recursos que compõe o módulo estão disponíveis. O módulo fictício fica disponível no final do horizonte de tempo da programação, ou seja, as operações não atendidas ficam alocadas de forma fictícia após o tempo disponível para a programação real. O tempo nesse módulo é determinado de forma aleatória dentre os tempos de processamento desse paciente. O parâmetro H é um número grande utilizado para auxiliar na construção das restrições, enquanto o parâmetro α é um número bem pequeno utilizado como fator de ponderação.

As variáveis de decisão z_i^r binárias indicam se o módulo r executa a operação i. As variáveis binárias y_{ij} determinam a precedência, não necessariamente imediata, entre duas operações $i, j \in I$. As variáveis reais x_i e l_i determinam o início e fim de cada operação, respectivamente. Finalmente, τ representa o *makespan* do sistema, ou seja, o instante em que o último paciente sai do sistema.

Modelo MMBJS:

$$\min \qquad \tau + \alpha \sum_{i \in I} x_i \tag{3.1}$$

Sujeito a:

$$\sum_{r \in R_i} z_i^r = 1 \quad i \in I \tag{3.2}$$

$$l_i - x_i - p_i^r z_i^r \geq 0 \quad i \in I, r \in R_i \tag{3.3}$$

$$l_i - x_i - \sum_{r \in R_i} p_i^r z_i^r \leq b_i \quad i \in I \tag{3.4}$$

$$l_i - x_j = 0 \quad (i,j) \in O_j, J \in \Im \tag{3.5}$$

$$x_j - l_i + H(2 - z_i^r - z_j^s) + H(1 - y_{ij}) \geq p_i^{cl} + p_j^{su} \quad i,j \in I, i < j, J^i \neq J^j, \tag{3.6}$$

$$r \in R_i, s \in R_j, M_i^r \cap M_j^s \neq 0$$

$$x_i - l_j + H(2 - z_i^r - z_j^s) + H y_{ij} \geq p_j^{cl} + p_i^{su} \quad i,j \in I, i < j, J^i \neq J^j, \tag{3.7}$$

$$r \in R_i, s \in R_j, M_i^r \cap M_j^s \neq 0$$

$$x_i - e^r z_i^r \geq p_i^{su} \quad i \in I, r \in R_i \tag{3.8}$$

$$\sum_{r \in R_i} f^r z_i^r - l_i \geq p_i^{cl} \quad i \in I \tag{3.9}$$

$$\tau - l_i \geq p_i^{cl} \quad i \in I \tag{3.10}$$

$$x_i, l_i, \tau \geq 0 \quad i \in I \tag{3.11}$$

$$y_{ij} \in \{0,1\} \quad i,j \in I, i < j \tag{3.12}$$

$$z_i^r \in \{0,1\} \quad i \in I, r \in R_i \tag{3.13}$$

A função objetivo (3.1) minimiza o valor do instante de término da última operação (*makespan*) e em segundo plano, ponderado pelo fator α, o somatório do instante de início de todas as operações. As restrições (3.2) atribuem um único módulo para cada operação, evitando assim que uma operação seja processada por mais de um módulo. As restrições (3.3) e (3.4) determinam os limites do instante de saída de uma atividade agendada. Uma operação deve terminar depois ou no mesmo instante que a soma do tempo de início e tempo de processamento da atividade, e antes da soma do instante de início, do tempo de processamento e do tempo máximo de espera permitido.

As restrições (3.5) obrigam a operação j subsequente de i no mesmo paciente ocorrer imediatamente após o término de i, ou seja, não existe intervalo de tempo entre duas operações subsequentes de um paciente. As restrições (3.6) e (3.7) impedem que dois módulos que possuem o mesmo recurso em sua composição sejam agendados simultaneamente. As restrições(3.8) e (3.9) garantem que o *setup* e a limpeza sejam realizados

dentro do período de disponibilidade do módulo $[e^r, f^r]$.

O conjunto de restrições (3.10) define o *makespan* como o maior valor do instante de término de todas as operações somado ao tempo de limpeza daquela operação, ou seja, é o instante em que todos os recursos do sistema são liberados. As restrições (3.11), (3.12) e (3.13) indicam o domínio das variáveis de decisão.

3.2 Exemplo

Apresentamos seguir os resultados obtidos a partir da aplicação do modelo MMBJS em um exemplo apresentado em Pham e Kinkert (2008). Os dados do exemplo são mostrados nas Tabelas 4 e 5. A Tabela 4 apresenta a distribuição dos recursos nos módulos e a janela de disponibilidade de cada um dos módulos. Note que o módulo fictício não utiliza nenhum recurso e que sua janela de tempo inicia após grande parte dos módulos não estarem disponíveis, excetuando o leito de UTI, fazendo com que os pacientes não atendidos sejam alocados para o fim do horizonte de tempo.

A Tabela 5 apresenta os tempos de processamento das etapas de cada paciente, os tempos de *setup*, limpeza e espera máxima. Observe que o tempo de processamento das operações nesse exemplo é independe do módulo a ser utilizado, porém isso não é obrigatório para todos os casos. Os tempos de *setup* e limpeza são nulos para as operações de pós e pré-cirurgia. Algumas divergências entre os dados encontrados na tabela e no exemplo em Pham e Kinkert (2008) são encontradas. Portanto os dados apresentados nessa dissertação foram alterados para corresponder aos dados encontrados no exemplo no trabalho dos autores. O valor do parâmetro α utilizado nesse exemplo foi 0,05.

A Figura 8 apresenta o Gráfico de *Gantt* correspondente a solução obtida através da resolução do exemplo proposto utilizando o modelo MMBJS, o *software* utilizado foi o CPLEX solver (versão 12.6). As etapas estão organizadas de acordo com os módulos utilizados. Conforme a Tabela 4, os módulos 1 e 2 podem processar pré-cirurgias, os módulos 5 até 8 são responsáveis por processar as etapas de cirurgia e, finalmente, os módulos 9, 10 e 11 processam as pós-cirurgias, enquanto o módulo fictício (12) pode processar qualquer etapa.

Tomando como exemplo a programação do paciente 5, temos que este entra no sistema no instante 615 min (x_i) no módulo 1 e tem sua etapa de pré-cirurgia finalizada após 30 minutos (aos 645 min). É realizado no módulo 3 o processo de *setup* o mais rápido possível (entre 640 e 660 min), fazendo com que o paciente aguarde e o módulo 1 permaneça em

Tabela 4: Dados do exemplo (Pham e Klinkert 2008): módulos e recursos

Módulo	Recursos	Disponibilidade
Pré-cirúrgico		
1	Enfermeiro pré-cirúrgico 1	480min - 960min
2	Enfermeiro pré-cirúrgico 1	1920min - 2400min
Cirúrgico		
3	Sala de cirurgia 1 Cirurgião 1 Enfermeiro cirúrgico 1 Anestesista 1	480min - 960min
4	Sala de cirurgia 2 Cirurgião 1 Enfermeiro cirúrgico 2 Anestesista 2	1920min - 2400min
5	Sala de cirurgia 1 Cirurgião 2 Enfermeiro cirúrgico 1 Anestesista 1	1920min - 2400min
6	Sala de cirurgia 2 Cirurgião 2 Enfermeiro cirúrgico 2 Anestesista 2	1920min - 2400min
7	Sala de cirurgia 2 Cirurgião 3 Enfermeiro cirúrgico 2 Anestesista 2	480min - 960min
8	Sala de cirurgia 2 Cirurgião 3 Enfermeiro cirúrgico 2 Anestesista 2	1920min - 2400min
Pós-cirúrgico		
9	Sala de recuperação 1	480min - 1020min
10	Sala de recuperação 2	1920min - 2460min
11	UTI	480min - 7200min
Fictício		
12	Ø	2460min - 7200min

Tabela 5: Dados do exemplo (Pham e Klinkert 2008): tempos

Paciente	Pré-cirurgia		Cirurgia		Pós-cirurgia	
	Duração	Módulos	Duração	Módulos	Duração	Módulos
1	30	1, 12	180	7, 12	60	9, 12
2	30	2, 12	135	8, 12	75	10, 12
3	30	1, 12	180	7, 12	60	9, 12
4	30	1, 2, 12	120	7, 8, 12	30	9, 10, 12
5	30	1, 2, 12	150*	3, 4, 12	1500	11, 12
6	30	1, 12	90	3*, 12	30	9, 12
7	30	2, 12	150*	4, 12	60	10, 12
8	30	2, 12	120	5, 6, 12	30	9, 10, 12
9	30	2, 12	105	5, 6, 12	90	9, 10, 12
10	30	2, 12	90	5, 6, 12	90	9, 10, 12
Setup	0	-	20	-	0	-
Limpeza	0	-	10	-	0	-
Espera máxima	15	-	15	-	0	-

*dados corrigidos

blocking por 15 minutos. Após o *setup*, o módulo 3 é liberado (aos 660 min) e o processo de cirurgia inicia. A cirurgia ocorre por 150 minutos e o paciente é liberado sem espera para a próxima etapa (aos 810 min). O módulo 3 inicia o processo de limpeza que dura 10 minutos após a transferência do paciente para a próxima operação. O módulo 11 processa a pós-cirurgia por 1500 minutos e o paciente sai do sistema (l_i) aos 2310 minutos. O paciente 4 foi o único paciente não atendido

3.3 Análise da função objetivo

A função objetivo proposta pelo modelo MMBJS possui dois termos: o *makespan* e o somatório dos instantes de início das operações ponderado por um fator α. O *makespan*, por definição, é o instante em que a última operação deixa o sistema, ou seja, é o maior valor do instante de término (l_i) dentre todas as operações acrescentado o valor do tempo de limpeza daquela operação. Nessa seção são analisados os dois termos que compõem a função objetivo e, a partir de observações retiradas dessa análise, uma nova função objetivo é apresentada na próxima seção. O exemplo apresentado na seção anterior será utilizado para ilustrar a influência de cada um dos termos da função objetivo, bem como as observações que embasaram o desenvolvimento da nova função objetivo.

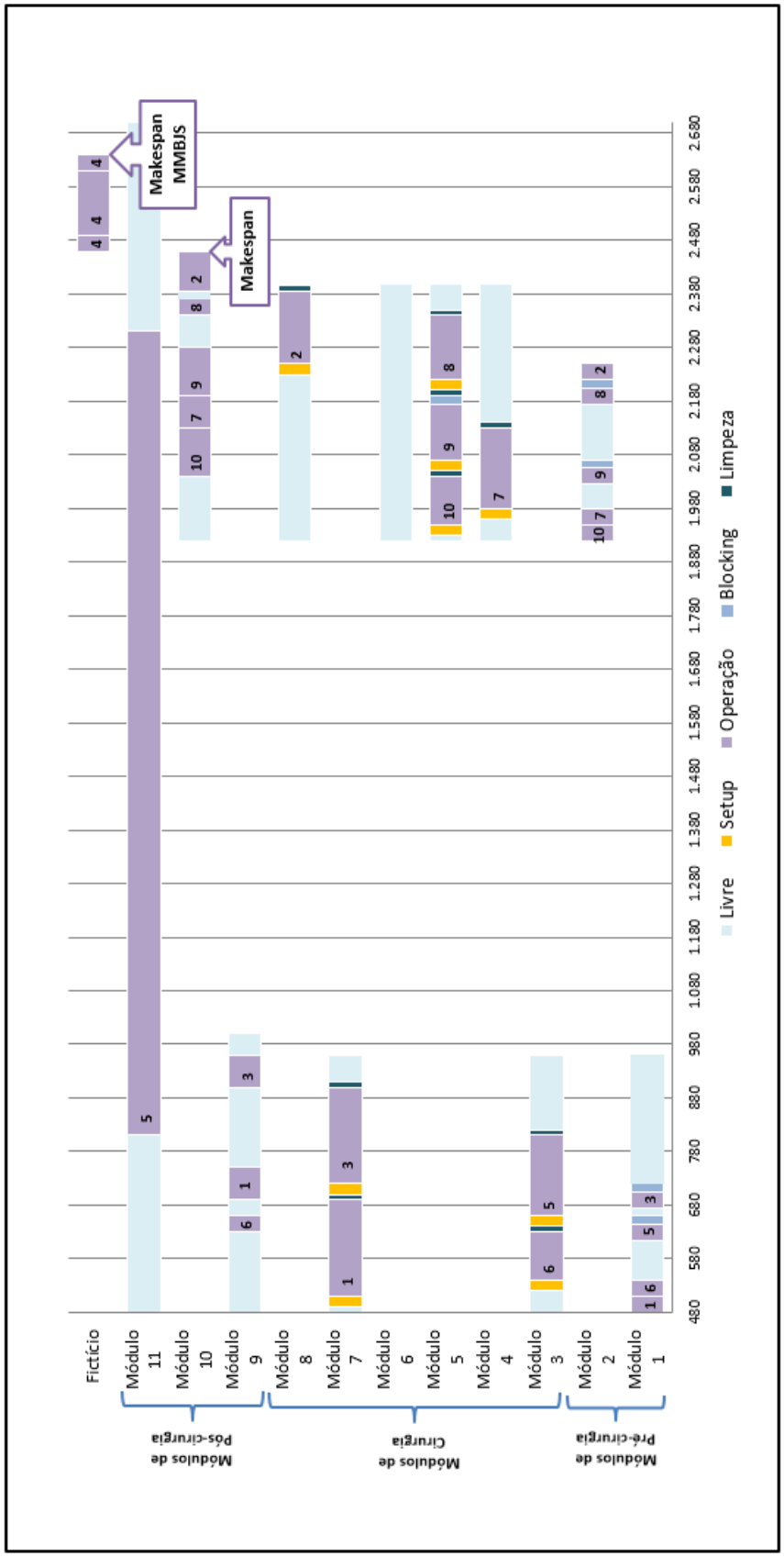

Figura 8: Gráfico *Gantt* do exemplo em Pham e Klinkert (2008) - Modelo MMBJS. Fonte: Elaborado pela autora

3.3.1 Cálculo do *makespan*

Makespan é definido como $\max(C_1, C_2, ..., C_k, ..., C_n)$, onde C_k indica o instante de término da operação k, ou seja, o *makespan* é o momento em que a última operação deixa o sistema. Minimizar o *makespan* normalmente melhora a utilização dos recursos. (Pinedo, 2010). Na formulação analisada o *makespan* é calculado a partir do conjunto de restrições (3.10) e (3.11) associadas com a minimização da função objetivo.

Das restrições (3.10), observa-se que todas as operações $i \in I$ são consideradas para o cálculo do *makespan*, incluindo aquelas que estão no módulo fictício. Como o módulo fictício não representa um conjunto real de recursos capazes de processar uma operação, os pacientes alocados a esse módulo são excluídos da programação e não deveriam ser considerados no cálculo do *makespan*. Além disso, o módulo fictício está no fim do horizonte de tempo, ou seja, sempre que uma operação for não atendida, o *makespan* corresponderá ao instante de término da última atividade alocada no módulo fictício. Dessa forma, o cálculo do *makespan* está incorreto.

Do exemplo, só um paciente (paciente 4) não é atendido e aparece alocado no módulo fictício. O valor do *makespan* calculado através do conjunto de restrições (3.10) é de 2640 (indicado na Figura 8 como Makespan MMBJS). Pela definição, *makespan* é o momento em que a última atividade deixa o sistema, ou seja, é o momento em que a última atividade deixa um módulo real, portanto o valor correto para o *makespan*, nesse caso, seria 2460 (indicado na Figura 8 como *Makespan*).

Considerar as atividades que estão no módulo fictício no cálculo do *makespan* permite que programações com piores valores para o *makespan* real sejam aceitas, uma vez que o valor real não está na função objetivo. Ou seja, para o exemplo anterior, se houver duas ou mais programações com o valor 2640 para o *makespan*, não existirá um critério de escolha entre essas programações, podendo ser escolhida a programação com pior "*makespan* real".

Para dar continuidade à analise dos termos da função objetivo é necessário corrigir o cálculo do *makespan*. As restrições (3.10) serão substituídas pelo conjunto (3.14) que garantem que só atividades alocadas a módulos reais (diferentes do módulo fictício) sejam consideradas.

$$\tau \geq l_i + p_i^{cl} - G(1 - \sum_{r \in R_i^d} z_i^r); \qquad i \in I \tag{3.14}$$

Onde R_i^d é o conjunto de todos os módulos reais e G é um número suficientemente grande.

Como por exemplo, $G = \max_{r \in R}\{f^r\}$.

3.3.2 Definição do α

Na função objetivo do modelo MMBJS (3.1), o termo do somatório dos instantes de início das operações ($\sum_{i \in I} x_i$) tem como objetivo reduzir o número de pacientes alocados no módulo fictício, ou seja, aqueles não atendidos pelo sistema. O parâmetro α que multiplica o termo reduz o seu valor, a fim de ponderar os dois componentes da função objetivo e garantir a programação ótima com menor *makespan* e menos pacientes não atendidos. No entanto, os autores não apresentam no trabalho o valor desse parâmetro α.

Com a substituição das restrições (3.10) pelas restrições (3.14), uma nova programação foi encontrada e está representada pelo gráfico de *Gantt* da Figura 9. O valor de α utilizado se manteve em 0,05. O novo *makespan* encontrado foi 2370 e o mesmo número de pacientes foram atendidos. Observe que o valor do *makespan* é menor que o apresentado na Figura 8 (2640) somente com a correção do cálculo do *makespan* e uma nova solução é encontrada não atendendo o paciente 2.

A definição de α é uma tarefa fundamental, pois tem função de ponderar os valores dos dois termos da função objetivo. Quando o fator α é muito grande, o termo associado ao somatório dos instantes de início podem assumir uma ordem de grandeza muito superior ao valor do *makespan*. Consequentemente, a solução encontrada antecipa o início de todas as etapas sem se preocupar com finalizá-las o mais rápido possível. Assim, torna-se aceitável, por exemplo, o agendamento de atividades com tempos de processamento grandes no final do horizonte de tempo, aumentando o valor do *makespan*, sem alterar significativamente o valor da função objetivo.

Por outro lado, quando o fator α é muito pequeno, o segundo termo torna-se pouco representativo tornando *makespan* o termo principal da função objetivo. Como as operações alocadas ao módulo fictício não são consideradas no cálculo do *makespan* (após a substituição das restrições (3.10) pelas (3.14)), muitos pacientes não são atendidos.

Utilizando os dados do exemplo apresentado anteriormente (Tabelas 4 e 5), a Tabela 6 apresenta o valor da função objetivo, assim como os valores dos dois termos que a constitui e o número de pacientes não atendidos em função da variação de α. Observe que o comportamento dos termos da função objetivo no exemplo são coerentes com o que foi discutido anteriormente. Portanto, um α grande [0,25;0,5] piora o valor do *makespan*,

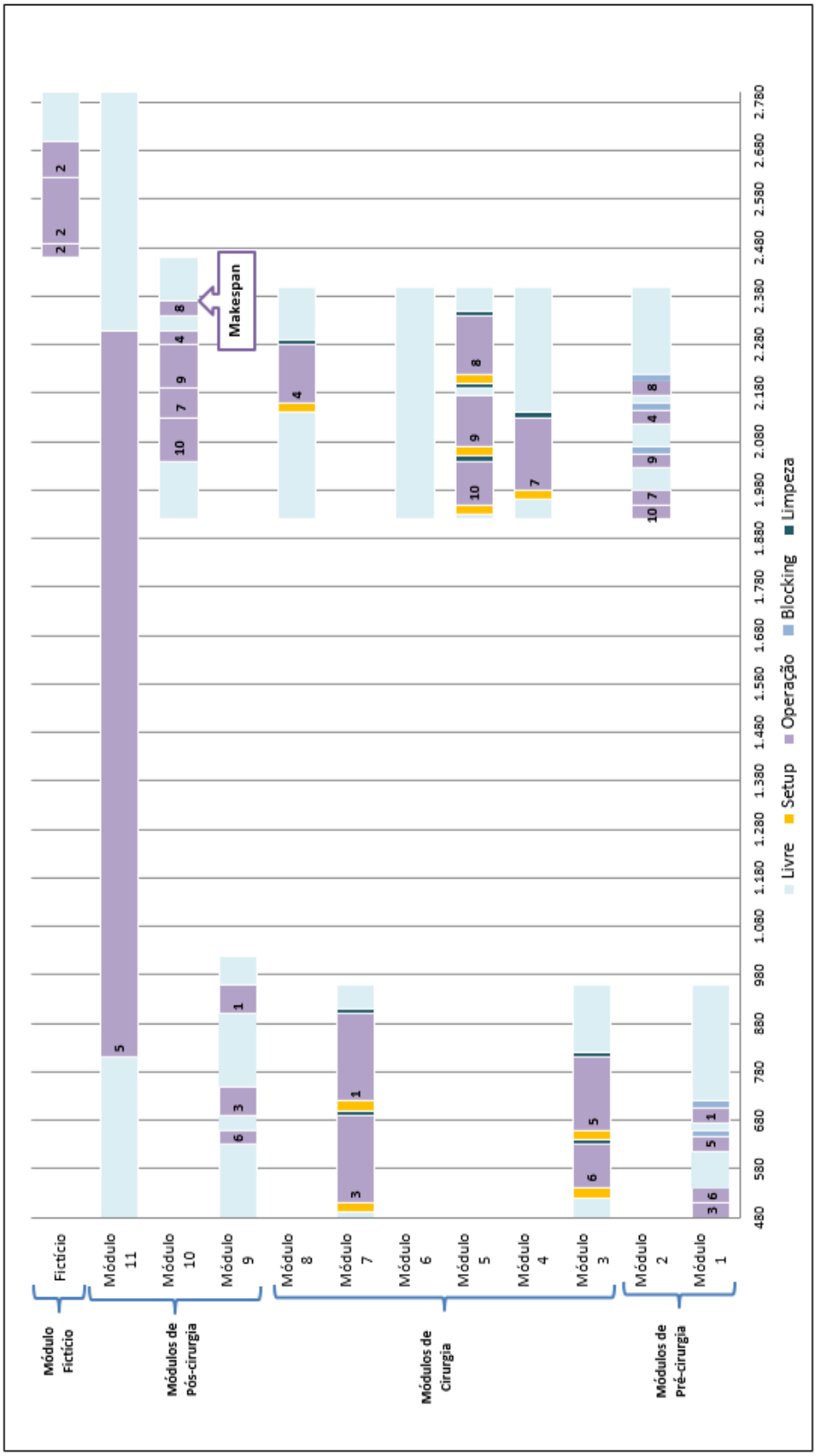

Figura 9: Gráfico *Gantt* do exemplo em Pham e Klinkert (2008) - Novo *makespan*. Fonte: Elaborado pela autora

Tabela 6: Teste de sensibilidade de α com dados das Tabelas 4 e 5

α	Função objetivo	*Makespan*	$\sum_{i \in I} x_i$	$\alpha \sum_{i \in I} x_i$	Pacientes não atendidos
0,500	32.880	2.400	60.960	30.480	1
0,250	17.640	2.400	60.960	15.240	1
0,240	17.029	2.370	61.080	14.659	1
0,050	5.424	2.370	61.080	3.054	1
0,020	3.592	2.370	61.080	1.222	1
0,015	3.242	870	158.145	2.372	7
0,000	0	0	217.920	0	10

enquanto um α pequeno [0;0,015] aumenta o número de pacientes deferidos.

Ainda na Tabela 6 observe que o *makespan* para $\alpha = 0,500$ é 2.400 porém, sabe-se que existe uma programação que também atende 9 pacientes e que possui menor *makespan*, por exemplo, a programação encontrada para $\alpha = 0,050$. Essa programação não foi escolhida ótima pelo modelo MMBJS com $\alpha = 0,500$, pois pelo cálculo da função objetivo para essa programação com este α temos que o somatório dos instantes de início (61.080) multiplicado por α adicionando o valor do *makespan* (2.370) é 32.910. Esse valor é pior que o resultado encontrado pelo modelo de otimização (32.880) apesar de atender 9 pacientes e encontrar o *makespan* menor. Dessa forma, observa-se que a definição do parâmetro α não é uma tarefa trivial e pode alterar consideravelmente o resultado do modelo.

3.4 Análise das restrições

O conjunto de restrições (3.3) e (3.4) determinam que o instante de término de uma atividade deve ser, no mínimo, igual ao seu instante de início adicionado ao tempo de processamento e, no máximo, igual à soma do instante de início, tempo de processamento e o tempo máximo de espera (b_i). As duas restrições possuem funções similares, definir os instantes de início e término das atividades, porém possuem estruturas diferentes. Na primeira expressão, existe uma restrição para cada atividade ($i \in I$) e para cada módulo disponível para aquela atividade ($r \in R_i$) enquanto na segunda, define-se uma restrição somente para cada atividade e os módulos são considerados no somatório nos módulos dos tempos de processamento ($\sum_{r \in R_i} p_i^r$). O artifício utilizado na segunda expressão reduz o número de restrições e pode representar ou não um ganho no tempo computacional empregado para a solução desse problema.

O mesmo ocorre para o conjunto de restrições (3.8) e (3.9) que definem que os instantes de início e término de cada atividade devem estar dentro da janela de tempo do módulo ao

Tabela 7: Análise das restrições (3.3), (3.4), (3.8) e (3.9) com relação ao tempo computacional em segundos

Modelo	Instâncias					
	1	2	3	4	5	média
original	58,89	43,29	107,87	190,41	136,84	107,46
restrições (3.3)	68,57	**31,35**	180,46	126,54	216,16	124,616
restrições (3.8)	86,48	52,75	82,95	99,98	95,57	83,546
restrições (3.3) e (3.8)	**48,62**	32,22	**68,34**	**98,34**	**68,22**	**63,148**

qual a atividade foi alocada. Dessa forma as restrições (3.3) e (3.8) podem ser substituídas pelas restrições (3.15) e (3.16), respectivamente.

$$l_i - x_i - \sum_{r \in R_i} p_i^r z_i^r \geq 0 \qquad i \in I \tag{3.15}$$

$$x_i - \sum_{r \in R_i} e^r z_i^r \geq p_i^{su} \qquad i \in I \tag{3.16}$$

Para ilustrar o impacto das alterações nas restrições, a Tabela 7 apresenta os tempos computacionais de cinco instâncias geradas conforme descrição apresentada na Seção 4.2.1 para 5 pacientes para cada alteração nas restrições. O modelo foi implementado no CPLEX ILOG versão 12.6 com os parâmetros padrões de configuração. Os experimentos foram realizados em um *cluster*, com um processador Intel(R) Xeon(R), com 8 núcleos e 4.20GHz. A memória RAM disponível foi de 512 GB e sistema operacional Linux.

Na primeira linha da Tabela 7 estão os tempos correspondentes à formulação original (MMBJS). Na segunda linha, somente as restrições (3.3) são substituídas pelas restrições (3.15), enquanto a terceira linha apresenta o equivalente para as restrições (3.8) que são substituídas pelas (3.16). Por fim, na última linha, apresentam-se os tempos computacionais de execução do modelo quando ambos conjuntos de restrições são substituídos. Observe que na maioria dos casos os tempos computacionais são consideravelmente menores quando as duas restrições são substituídas. A única exceção é a instância 2, onde a substituição das restrições (3.3) e (3.8) possui tempo de computacional 2,8% pior que em relação a substituição somente da restrição (3.3).

3.5 Comentários adicionais

Além das análises realizadas nas seções anteriores, alguns comentários adicionais devem ser realizados a fim de aprimorar a análise do modelo MMBJS. Do gráfico de *Gantt*

apresentado na Figura 9 observa-se que o paciente 2 não é atendido da programação, ou seja, está alocado ao módulo fictício. Ele está dentro do conjunto de pacientes que podem ser processados no segundo dia de programação, juntamente com os pacientes 4, 5, 7, 8, 9 e 10. O paciente 5 possui o maior tempo de processamento total dentre todos os pacientes do conjunto (1780 min), mas como utiliza um recurso com grande disponibilidade e pouca demanda é encaixado na programação, sem prejuízo ao valor do *makespan*. O restante dos pacientes do conjunto possuem tempo de processamento total inferior ou igual ao paciente 2 (240 min), dessa forma, este é excluído da programação a fim de minimizar o *makespan* da solução. Uma vez que a função objetivo proposta só considera o número de pacientes atendidos independentemente do seu tempo de processamento, existe uma tendência do modelo matemático a processar preferencialmente os pacientes com menores tempos de processamento e, consequentemente, não atender os pacientes com altos tempos de processamento. Em um ambiente hospitalar esse tipo de fenômeno pode ser indesejável, uma vez que os pacientes que necessitam de cirurgias mais complicadas e, portanto, mais demoradas poderão ser não atendidos na programação.

Uma solução para esse problema é apresentada pelos autores Pham e Klinkert (2008) que sugerem determinar as prioridades de atendimento dos pacientes através da fixação da variável y_{ij} com valor igual a um para a primeira atividade dos pacientes prioritários. Esse artifício busca garantir que os pacientes com etapas mais demoradas, ou aqueles pacientes que não foram atendidos na programação da semana anterior possam ser priorizados iniciando antes de outros pacientes.

Em outras palavras, essa solução define a ordem de programação dos pacientes prioritários quando estes concorrem com outros pacientes pelo mesmo recurso em módulos diferentes, garantindo que estes sejam atendidos. Porém, essa estratégia não garante que o melhor *makespan* seja escolhido, uma vez que a ordem dos pacientes é fixada. Dessa forma, é possível afirmar que podem existir outras programações que incluam os pacientes prioritários sem, necessariamente, determinar que estes sejam processados primeiro quando concorrerem por recursos com outros pacientes.

Além disso, determinar a sequência dos pacientes reduz o espaço de solução viável e pode excluir soluções que contenham os pacientes prioritários e tenham melhor *makespan*. Uma alternativa mais interessante é favorecer que esses pacientes sejam atendidos no horizonte de tempo liberando o modelo para determinar qual a melhor posição para cada paciente.

4 PROPOSTA DE MODELO MATEMÁTICO

Após considerar todas as análises apresentadas no capítulo anterior, algumas alterações são sugeridas ao modelo MMBJS, a fim de melhorar o seu desempenho. Na próxima seção é apresentado o modelo proposto e em seguida as são realizados testes computacionais.

4.1 Formulação proposta

O modelo apresentado nessa seção é proposto pela autora desse trabalho e consiste na revisão e aperfeiçoamento do modelo anterior. A Tabela 8 apresenta as modificações propostas, bem como os resultados esperados. Cada uma delas é discutida na sequência.

Conforme apresentado na Seção 3.3.1, o cálculo do *makespan* não está adequado utilizando as restrições (3.10) porque o módulo fictício está no final do horizonte de tempo. Portanto, se houver qualquer paciente indeferido, o *makespan* será definido como o instante de término da última atividade no módulo fictício. A fim, de resolver esse problema, sem modificar as restrições (3.10), optou-se por retirar o módulo fictício.

A definição do parâmetro α não é uma tarefa trivial e tem grande influência na

Tabela 8: Modelo proposto: modificações no modelo MMBJS

Modificações	Resultados esperados
Retirar o módulo fictício	O cálculo do *makespan* fica adequado sem a necessidade de modificar as restrições (3.10).
Retirar o parâmetro α da função objetivo	Evitar a tarefa não trivial de definir o α para cada instância
Reduzir o número de restrições	Reduzir o tempo computacional
Determinar a prioridade dos pacientes	Evitar que os pacientes com alto tempo de processamento sejam indeferidos

qualidade da solução, como pôde ser verificado na Seção 3.3.2. Esse parâmetro multiplica o somatório dos instantes de início de todas as atividades e busca trazer as atividades para mais próximo do início do horizonte de tempo, minimizando assim o número de pacientes alocados ao módulo fictício. Dessa forma, foi proposta uma função objetivo que também minimize o número de pacientes não atendidos, porém que não utilize o parâmetro α na sua constituição.

O conjunto de restrições (3.3) e (3.8) pode ser reduzido se substituído pelas restrições (3.13) e (3.14), respectivamente. Essa redução no número de restrições, conforme testes preliminares, pode apresentar uma diminuição no tempo de execução do modelo para encontrar uma solução ótima. Portanto, no modelo proposto faremos essa substituição.

Por fim, a solução ótima do modelo MMBJS tem uma tendência de agendar os pacientes com menores tempos de processamento e isso pode não ser interessante em um ambiente hospitalar. Dessa forma, o modelo proposto apresenta um função objetivo que leva em conta a prioridade de cada paciente.

A Tabela 9 apresenta as notações utilizadas no modelo proposto. As notações alheias à Tabela 3 estão indicadas e serão tratadas a seguir.

O parâmetro G é um número grande que será utilizado na nova função objetivo. O final do horizonte de tempo é um valor suficientemente grande para atender o propósito desse parâmetro. O fator de prioridade de cada paciente é definida por w_i. Pacientes com maiores fatores terão mais chances de ser atendidos na semana da programação. O *makespan* é indicado por *mkp* no modelo proposto para diferenciar do *makespan* do modelo MMBJS.

Tabela 9: Notações modelo proposto.

Conjuntos	
$\Im$	Conjunto de pacientes
I	Conjunto de operações
O_j	Conjunto de pares consecutivos de operações do paciente $J \in \Im$
M	Conjunto de recursos
M_i^r	Conjunto de recursos contidos no módulo r que atendem a operação $i \in I$, $r \in R_i$, $M_i^r \subseteq M$
R	Conjunto de índices dos módulos, $R = \cup_{i \in I} R_i$
R_i	Conjunto de índices de módulos possíveis para a operação $i \in I$
Parâmetros	
p_i^r	Tempo de processamento da operação i no módulo r, $i \in I, r \in R_i$
p_i^{su}	Tempo de *setup* para a operação $i \in I$ antes de operação
p_i^{cl}	Tempo de limpeza para a operação $i \in I$ depois da operação
b_i	Tempo máximo de espera permitido para a operação $i \in I$ antes do paciente mover para a próxima operação
e^r	Início do intervalo de disponibilidade do módulo $r \in R$
f^r	Fim do intervalo de disponibilidade do módulo $r \in R$
H	Número grande utilizado nas restrições. $H = \sum_{i \in I} \max_{r \in R_i}\{p_i^r\} + \max_{r \in R_i, i \in I}\{p_i^r\} + \max_{i \in I}\{p_i^{su}\} + \max_{i \in I}\{p_i^{cl}\}$
J^i	Paciente no qual a operação $i \in I$ pertence
G*	Número grande utilizado na função objetivo. $G = \max_{r \in R}\{f^r\}$
w_i*	Fator de prioridade do paciente $i \in I$
Variáveis de decisão	
z_i^r	1, se o módulo r é designado à operação $i \in I$; 0, caso contrário
y_{ij}	1, se a operação j é processada depois da operação i; 0, caso contrário $i, j \in I$, $i < j$
x_i	Instante de início da operação $i \in I$
l_i	Instante de fim da operação $i \in I$
mkp*	*Makespan*

*novas notações

Modelo Proposto:

$$min \quad mkp - G \sum_{i \in I} \sum_{r \in R_i} w_i z_i^r \tag{4.1}$$

Sujeito a:

$$\sum_{r \in R_i} z_i^r \leq 1 \quad i \in I \tag{4.2}$$

$$l_i - x_i - \sum_{r \in R_i} p_i^r z_i^r \geq 0 \quad i \in I \tag{4.3}$$

$$l_i - x_i - \sum_{r \in R_i} p_i^r z_i^r \leq b_i \quad i \in I \tag{4.4}$$

$$l_i - x_j = 0 \quad (i,j) \in O_j, J \in \Im \tag{4.5}$$

$$x_j - l_i + H(2 - z_i^r - z_j^s) + H(1 - y_{ij}) \geq p_i^{cl} + p_j^{su} \quad i,j \in I, i < j, J^i \neq J^j, \tag{4.6}$$

$$r \in R_i, s \in R_j, M_i^r \cap M_j^s \neq 0$$

$$x_i - l_j + H(2 - z_i^r - z_j^s) + H y_{ij} \geq p_j^{cl} + p_i^{su} \quad i,j \in I, i < j, J^i \neq J^j, \tag{4.7}$$

$$r \in R_i, s \in R_j, M_i^r \cap M_j^s \neq 0$$

$$x_i - \sum_{r \in R_i} e^r z_i^r \geq p_i^{su} \sum_{r \in R_i} z_i^r \quad i \in I \tag{4.8}$$

$$\sum_{r \in R_i} f^r z_i^r - l_i \geq p_i^{cl} \sum_{r \in R_i} z_i^r \quad i \in I \tag{4.9}$$

$$mkp - l_i \geq p_i^{cl} \quad i \in I \tag{4.10}$$

$$x_i, l_i, mkp \geq 0 \quad i \in I \tag{4.11}$$

$$y_{ij} \in \{0,1\} \quad i,j \in I, i < j \tag{4.12}$$

$$z_i^r \in \{0,1\} \quad i \in I, r \in R_i \tag{4.13}$$

O problema tratado nesse modelo é classificado como um problema de programação em dois níveis. Segundo Dempe (2002) os problemas em dois níveis são problemas de otimização matemática em que deseja-se minimizar ou maximizar uma função objetivo que possui variáveis que podem ser particionadas em dois conjuntos (x e y) e que após calculada a solução ótima do conjunto x, calcula-se o ótimo do conjunto y considerando o resultado de x. Em outras palavras, primeiramente encontra-se a solução do problema do nível superior (x) que será *input* para otimizar o problema do nível inferior (y).

A função objetivo proposta, assim como a encontrada no modelo MMBJS, é composta

por dois termos e minimiza o número de pacientes indeferidos e o *makespan*. O problema é em dois níveis uma vez que dentre as opções de programação que atendem o maior número de pacientes (problema do nível superior), escolhe-se a opção que minimiza o valor do *makespan* (problema do nível inferior).

O *makespan* é calculado utilizando as restrições (4.10) e (4.11) juntamente com a minimização da função objetivo. No modelo proposto, não existe mais o módulo fictício portanto, o valor do *makespan* é igual ao instante de fim da última atividade alocada a um módulo real acrescido ao tempo de limpeza. A exclusão do módulo fictício apresentou-se uma solução viável e intuitiva, uma vez que, com mínimos ajustes, o módulo fictício é dispensável para o modelo.

O número de atividades dos pacientes atendidos é calculado pelo termo $\sum_{i \in I} \sum_{r \in R_i} z_i^r$ e, consequentemente, obtém-se o número de pacientes atendidos dividindo esse valor pelo número de atividades por paciente. O parâmetro G garante que o número de pacientes atendidos seja o principal componente da função objetivo, enquanto o sinal de negativo ajusta o termo para o problema de minimização. Além de retirar o parâmetro α da função objetivo, esse termo proposto é ponderado pelo fator de prioridade de cada paciente agendado w_i, o que favorece que os pacientes com maiores prioridades sejam alocados na programação do horizonte de tempo.

As restrições (4.2) garantem que cada atividade só deve ser atendida por no máximo um módulo. No modelo MMBJS, as restrições equivalentes (3.2) estabeleciam que um módulo fosse alocado a cada atividade, dessa forma, as atividades não atendidas eram alocadas ao módulo fictício. Com a exclusão do módulo fictício, esse conjunto de equações deve ser convertido em inequações.

As restrições (4.3) e (4.4) garantem que o instante de início e término de uma atividade sejam no mínimo o tempo de processamento desta atividade e no máximo o tempo máximo de espera permitido. Conforme discutido na Seção 3.4, o conjunto das restrições (4.4) é reduzido em relação ao modelo MMBJS adicionando o somatório de $r \in R_i$. Com essa alteração espera-se que ocorra uma redução no tempo de execução do modelo.

O conjunto de restrições (4.5) garante que não haverá espera para o paciente entre duas atividade consecutivas, ou seja, após o fim de uma atividade paciente deve ser transferido imediatamente para a atividade subsequente, o final da atividade inclui o tempo máximo de espera, caso seja necessário. As restrições (4.6) e (4.7) garantem que os módulos que possuam o mesmo recurso não sejam agendados simultaneamente.

Os conjuntos de restrições (4.8) e (4.9) asseguram que as atividades serão alocadas

aos módulos dentro do seu intervalo de disponibilidade. Duas alterações foram realizadas nesses conjuntos com relação ao modelo MMBJS. No conjunto de restrições (4.9) foi adicionado o somatório em $r \in R_i$, reduzindo, assim, o número de restrições. Enquanto o termo $\sum_{r \in R_i} z_i^r$ foi adicionado ao fim das restrições dos dois conjuntos (4.8) e (4.9). Juntamente com a transformação das restrições (4.2) em inequações, essa segunda modificação permite eliminar o módulo fictício do modelo.

As restrições (4.10) determinam que o *makespan* é o maior instante de término de todas as atividades do sistema. As restrições (4.11), (4.12) e (4.13) determinam o domínio das variáveis de decisão.

4.1.1 Função objetivo para o problema em dois níveis

A função objetivo escolhida para o modelo proposto possui dois termos, onde cada termo representa um nível do problema. O nível superior $(-G \sum_{i \in I} \sum_{r \in R_i} z_i^r)$ garante que o maior número de pacientes seja atendido dentro do horizonte de tempo disponível. Enquanto, o nível inferior (mkp) garante que, dentre todos as opções de programações que atendem o maior número de pacientes, aquela com menor *makespan* seja escolhida.

Um exemplo foi realizado a fim de mostrar as diferenças entre duas situações possíveis: (1) utilizar o modelo minimizando somente o número de pacientes indeferidos e, posteriormente, realizar um ajuste manual para reduzir o *makespan* e (2) analisar o problema em dois níveis utilizando a função objetivo (4.1). Ressalta-se que o exemplo foi construído para evidenciar as diferenças entre as duas situações possíveis.

Os dados utilizados estão apresentados nas Tabelas 10 e 11. Observe que diferentemente do exemplo anterior, o tempo de processamento das atividades são dependentes dos módulos que as processam (Tabela 11). Por exemplo, a etapa de cirurgia do paciente 2, tem tempo de processamento 150 minutos quando processada pelo módulo 2 e 170 minutos quando processada pelo módulo 3. O fator de prioridade admitido foi 1 para todos os pacientes.

A Figura 10 apresenta o gráfico de *Gantt* que representa a solução obtida através da resolução do modelo matemático utilizando somente o elemento da função objetivo que visa minimizar o número de pacientes indeferidos. Nesse exemplo, somente o paciente 3 não é atendido e o *makespan* encontrado é 1050. Observe que não é possível processar todos os pacientes, pois os módulos de cirurgia (2 e 3) não tem disponibilidade suficiente portanto, estes são os módulos restritivos do sistema.

Tabela 10: Dados do exemplo problema em dois níveis: módulos e recursos.

Módulo	Recursos	Disponibilidade
Pré-cirúrgico		
1	Enfermeiro pré-cirúrgico 1	480min - 960min
Cirúrgico		
2	Sala de cirurgia 1 Cirurgiã 1 Enfermeiro cirúrgico 1 Anestesista 1	480min - 960min
3	Sala de cirurgia 2 Cirurgiã 2 Enfermeiro cirúrgico 2 Anestesista 2	480min - 960min
Pós-cirúrgico		
4	Sala de recuperação	480min - 1440min
5	UTI	480min - 2880min

Tabela 11: Dados do exemplo problema em dois níveis: tempos.

Paciente	*Pré-cirurgia* Módulos(Duração)	*Cirurgia* Módulos(Duração)	*Pós-cirurgia* Módulos(Duração)
1	1(30min)	2(190min), 3(190min)	5(100min)
2	1(30min)	2(150min), 3(170min)	4(60min)
3	1(30min)	2(190min), 3(180min)	4(100min)
4	1(30min)	2(150min), 3(170min)	5(150min)
5	1(30min)	2 (90min), 3(100min)	4(80min)
Setup	0	20	0
Limpeza	0	10	0
Espera Máxima	15	15	0

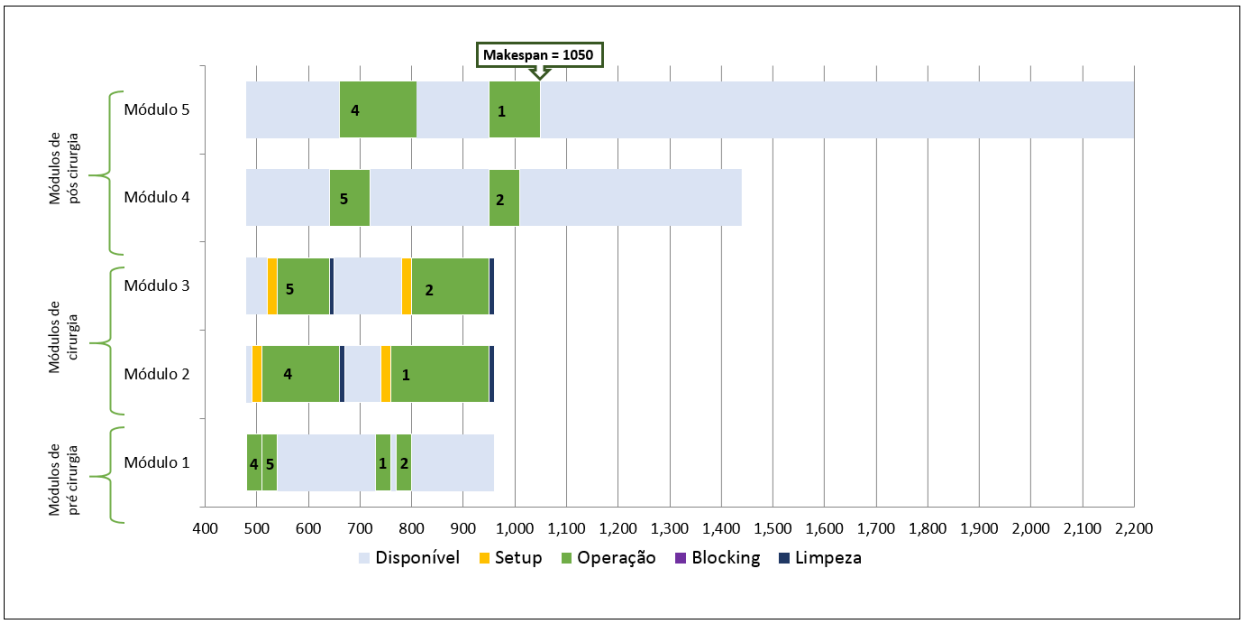

Figura 10: Gráfico *Gantt* - função objetivo: número de pacientes atendidos. Fonte: Elaborado pela autora

A Figura 11 apresenta a mesma sequência ajustada manualmente de forma a reduzir os tempos ociosos e, consequentemente, o *makespan* da solução apresentada anteriormente. Observe que o ajuste foi realizado antecipando a entrada dos pacientes 1 e 2, respeitando as restrições do modelo, e que o valor do *makespan* é reduzido para 980, apresentando uma melhora em relação à solução anterior. Assim como na solução anterior, o paciente 3 não é atendido.

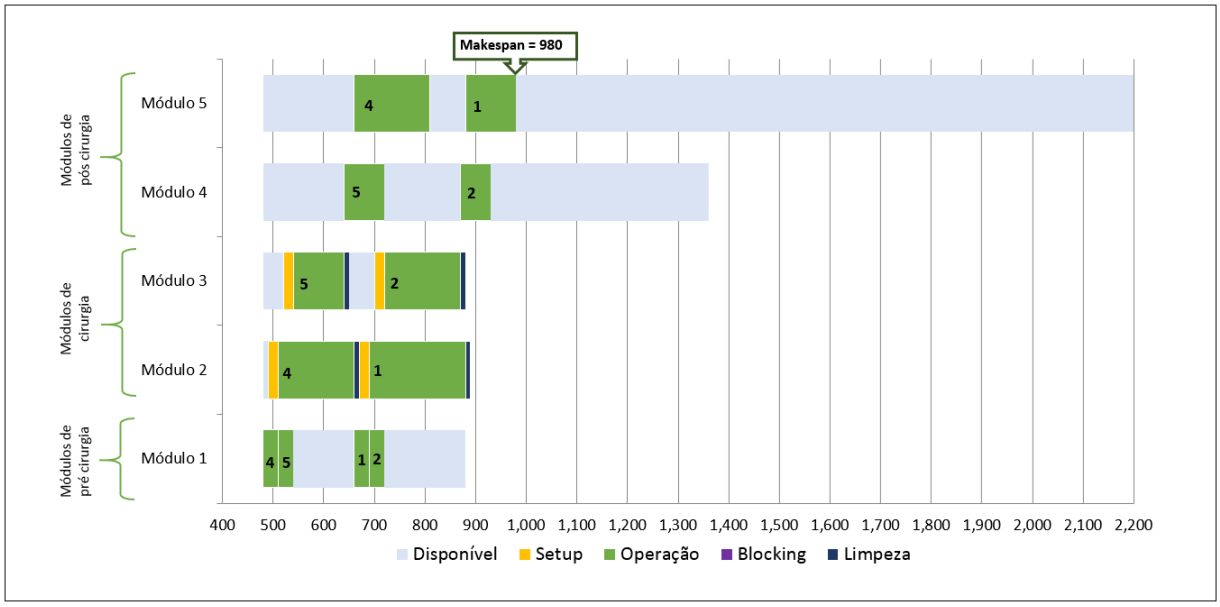

Figura 11: Gráfico *Gantt* - função objetivo: número de pacientes atendidos com retirada dos tempos ociosos Fonte: Elaborado pela autora

A Figura 12 mostra o gráfico de *Gantt* resultado do modelo matemático utilizando a função objetivo proposta. O número de pacientes atendidos se mantém em 4,como esperado, e o *makespan* é reduzido para 940. Isso ocorre pois, a função objetivo considera, além do número de pacientes não atendidos, o valor do *makespan* na sua composição. Assim, dentre todas as opções de programações que atendem 4 pacientes aquela que possuir menor *makespan* é escolhida. Dessa forma, a função objetivo proposta apresenta resultados diferentes e mais interessantes do que a opção de minimizar somente o número de pacientes indeferidos e, em seguida ajustar, manualmente para reduzir o tempo ocioso.

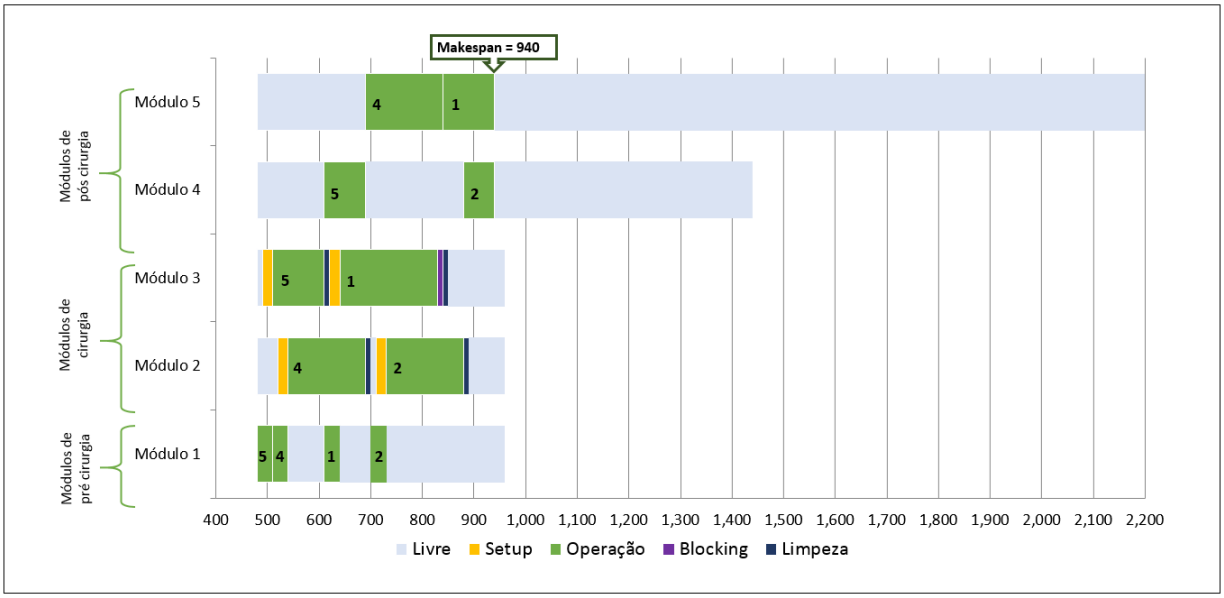

Figura 12: Gráfico *Gantt* - função objetivo completa. Fonte: Elaborado pela autora

4.1.2 Prioridade de pacientes

Como apresentado na Seção 3.5 muitas vezes é necessário definir a prioridade dos pacientes de forma a favorecer que determinados pacientes sejam programados dentro do horizonte de tempo. Para isso, os autores Pham e Klinkert (2008) sugerem determinar a ordem dos pacientes prioritários utilizando a variável y_{ij}. Essa alternativa determina que os pacientes prioritários sejam atendidos antes dos demais pacientes, ou seja, se o paciente 1 é prioritário então $y_{ij} = 1$ para $i = 1$, onde i corresponde à primeira atividade do paciente 1, e $j \in I$.

Utilizar essa abordagem pode resultar em programações insatisfatórias, por exemplo, considerando uma situação extrema em que os módulos possíveis para atender um paciente prioritário só estejam disponíveis no segundo dia de programação, o resultado obtido não alocará nenhum paciente no primeiro dia do horizonte de tempo, podendo reduzir o

Tabela 12: Solução do modelo proposto considerando prioridade dos pacientes definindo a variável y_{ij}.

Paciente prioritário	Pacientes atendidos	*Makespan*	1º paciente atendido
1	1,2,4 e 5	960	1
2	1,2,4 e 5	980	2
3	1,2,4 e 5	940	5
4	1,2,4 e 5	950	4
5	1,2,4 e 5	940	5

número de pacientes atendidos e aumentar o *makespan*.

Utilizando a estratégia de priorização de pacientes apresentada por Pham e Klinkert, resolveu-se o exemplo da seção anterior definindo, em cada instância, um paciente prioritário. Observa-se na Tabela 12 os pacientes atendidos na programação, o valor do *makespan* e o primeiro paciente a ser atendido em cada resultado encontrado. Da seção anterior, temos que a solução ótima encontrada sem definição de prioridade é aquela apresentada na Figura 12 (*makespan*= 940; 4 pacientes atendidos e paciente 3 não é atendido) que será utilizada para comparação com os resultados aqui encontrados.

Note que na instância em que o paciente 3 é prioritário, o resultado encontrado não contém esse paciente. Isso ocorre porque as restrições (4.6) e (4.7), que determinam as precedências das atividades, não impedem que o valor do início e fim das atividades desse paciente seja zero. Portanto, a solução escolhida atende a determinação $y_{7j} = 1$, define $x_7 = 0$ e $l_7 = 0$ (onde $i = 7$ corresponde a primeira atividade do paciente 3) e não atende o paciente 3, fazendo com que a solução encontrada seja a solução ótima do problema sem prioridade. Essa situação só ocorre para o modelo proposto, uma vez que este não obriga que todas as operações sejam alocadas em um módulo, permitindo que o instante de início e término do paciente prioritário seja zero. No modelo MMBJS, por outro lado, o comportamento da solução com aplicação dessa estratégia seria semelhante ao resultado encontrado para os pacientes 1, 2, 4 e 5, discutido a seguir.

Quando os pacientes prioritários são os pacientes 1, 2, 4 e 5, o resultado mostra que a programação sempre inicia com o paciente prioritário e o valor do *makespan* é sempre pior ou igual ao resultado ótimo encontrado sem prioridade. Essa situação ocorre, pois definir a posição de um paciente reduz o espaço de solução viável e tende piorar a solução encontrada. Observa-se também que nesses casos, o paciente prioritário não é excluído da programação, como no caso do paciente 3, pois é mais ou igualmente interessante para o método de solução do modelo escolher a solução apresentada na Tabela 12 do que uma solução sem o paciente prioritário.

Tabela 13: Solução do modelo proposto considerando prioridade dos pacientes definindo a variável w_i.

Paciente prioritário	Pacientes atendidos	*Makespan*	1º paciente atendido
1	1,2,4 e 5	940	5
2	1,2,4 e 5	940	5
3	2,3,4 e 5	960	3
4	1,2,4 e 5	940	5
5	1,2,4 e 5	940	5

No modelo matemático proposto sugere-se uma maneira alternativa de modelar a prioridade dos pacientes, utilizando um fator de prioridade para cada paciente (w_i). Observe que, nesse estudo, priorizar um paciente é favorecer que ele seja agendado no horizonte de tempo e não, necessariamente, determinar que ele seja o primeiro a ser atendido.

Utilizando o mesmo exemplo, a Tabela 13 apresenta os resultados encontrados quando cada um dos pacientes é prioritário. No exemplo, o paciente prioritário recebe $w_{i(prio)} = 5$ e os demais $w_{i(k)} = 1$, $k \in J$ e $k \neq prio$, onde i representa a primeira atividade de cada paciente k e prioritário. Como a sequência das atividades não foi determinada por nenhuma variável, o modelo apresenta como solução para os pacientes 1, 2, 4 e 5 a mesma solução ótima encontrada para o exemplo sem prioridade, como esperado.

No caso do paciente 3, como ele não aparece na solução ótima sem prioridade, o modelo busca a melhor solução que o contenha. Essa solução exclui o paciente 1, adiciona o paciente 3 na programação e possui *makespan* de 960. O fator de prioridade tem função de favorecer os pacientes prioritários, porém não garante que esses sejam atendidos na programação. Uma alternativa para determinar que um paciente seja impreterivelmente atendido é definir a expressão $\sum_{r \in R_i} z_i^r = 1$ para a primeira atividade desse paciente.

As soluções ótimas encontradas utilizando a definição da variável de posição (y_{ij}) apresentam resultados piores ou iguais para o *makespan* do que o valor obtido na solução encontrada utilizando os fatores de prioridade, e, em alguns casos, não exerce sua função adequadamente. A partir dos exemplos, conclui-se que o paciente prioritário não precisa obrigatoriamente ser o primeiro da sequência para que preferencialmente seja atendido no período. Ao determinar o início do paciente prioritário, estamos excluindo possíveis soluções que atendem esse paciente e que possuam melhores *makespan*. Dessa forma, atribuir fatores de prioridade aos pacientes é uma estratégia adequada para solucionar esse problema, sem contudo, prejudicar o *makespan* ou o número de pacientes atendidos.

4.2 Resultados computacionais

Nessa seção, os resultados obtidos pelos o modelo matemático MMBJS e pelo modelo proposto são apresentados e analisados. Além disso, a geração das instâncias utilizadas nos experimentos é descrita. Experimentos são realizados para comparar o modelo proposto com o MMBJS. Adicionalmente, experimentos são realizados variando parâmetros do sistema, a fim de realizar análises de sensibilidade no modelo proposto.

Os modelos matemáticos foram implementados no CPLEX ILOG versão12.6. Os parâmetros padrões de configurações não foram alterados, exceto pelo tempo de execução limitado em 2 horas. Os experimentos foram realizados em um *cluster*, do qual utilizou-se um processador Intel(R) Xeon(R), com 6 núcleos e 4.20GHz. A memória RAM disponível foi de 512 GB e sistema operacional Linux.

4.2.1 Geração das instâncias

A literatura não possui muitos trabalhos que tratam o problema de programação de cirurgias com múltiplos recursos e etapas, dessa forma também existe escassez de instâncias disponíveis. Ozkarahan (2000) propõe um conjunto de instâncias que contém algumas informações dos pacientes como tempo de processamento da etapa de cirurgia classificados por especialidade. Para seus experimentos computacionais Pham e Klinkert (2008) completam os dados apresentados por Ozkarahan (2000), inserindo os tempos de processamento das etapas de pré e pós-cirurgia, tempo máximo de espera, *setup* e limpeza, além da organização dos módulos. Uma vez que as instâncias completas não foram disponibilizadas, este trabalho também inclui a geração aleatória de instâncias semelhantes às encontradas em Pham e Klinkert (2008) para a execução dos experimentos.

O conjunto de instâncias considera três especialidades médicas (cirurgia ginecológica, ortopédica e torácica) nas quais os pacientes são distribuídos. Os tempos de processamento para cada especialidade foram gerados através de uma distribuição uniforme discreta, assim como os tempos de *setup*, limpeza, tempo máximo de espera e fator de prioridade de cada pacientes. A distribuição para cada parâmetro é apresentados a seguir.

- Tempo de processamento da etapa de pré-cirurgia: 40, 41,..., 60 minutos.
- Tempo de processamento da etapa de cirurgia ginecológica: 95, 96,..., 140 minutos.
- Tempo de processamento da etapa de cirurgia ortopédica: 95, 96,..., 200 minutos.

- Tempo de processamento da etapa de cirurgia torácica: 75, 76,..., 215 minutos.
- Tempo de processamento da etapa de pós-cirurgia na sala de recuperação: 90, 91,..., 150 minutos.
- Tempo de processamento da etapa de pós-cirurgia na UTI: 1500,1501,...,1800 minutos.
- Tempo de *setup*: 25, 26,..., 30 minutos.
- Tempo de limpeza: 15, 16,..., 25 minutos.
- Tempo de espera máxima: 15, 16,..., 20 minutos.
- Fator de prioridade dos pacientes: 1, 2,..., 10.

O gerador de números aleatórios e as sementes propostas por Taillard (1993) foram utilizados para gerar os parâmetros apresentados. Para as instâncias a serem utilizadas no modelo MMBJS, o tempo de processamento no módulo fictício admitido foi o limite superior definido para a distribuição uniforme discreta, por exemplo, para atividades de pré-cirurgia no módulo fictício o tempo de processamento é igual a 60 minutos.

A composição dos módulos foram determinados conforme sugerido em Pham e Klinkert (2008) e têm disponibilidade de 6 horas por dia, exceto os módulos de pós-cirurgia que têm disponibilidade de 24 (UTI) ou 7 horas por dia (SR). Os conjuntos de recursos necessários para cada tipo de módulo são apresentados a seguir:

- Módulo de pré-cirurgia: enfermeiro pré-cirúrgico;
- Módulo de cirurgia: sala de operação, cirurgião, enfermeiro cirúrgico e anestesista;
- Módulo de pós-cirurgia: UTI ou SR

São 4 módulos de cirurgia divididos por especialidade: 1 cirurgia ginecológica, 2 cirurgias ortopédicas e 1 cirurgia torácica. Para cada instância, cerca de 60% dos pacientes devem ser processados por módulos de UTI na etapa de pós-cirurgia, os demais são processados pelos módulos de SR. Para todas instâncias geradas o horizonte de tempo determinado é de 5 dias. A Tabela 14 apresenta a distribuição dos pacientes nas especialidades médicas e nos módulos de pós-cirurgia.

Dois tipos de testes são realizados nesse trabalho: avaliação comparativa dos modelos e analise de sensibilidade do modelo proposto. Para o primeiro teste, 5 instâncias com 10

Tabela 14: Número de pacientes por especialidade cirúrgica e por tipo de pós-cirurgia para cada conjunto de instâncias

Número	Especialidade cirúrgica			Pós-cirurgia	
Pacientes	Ginecologia	Ortopedia	Torácica	UTI	SR
5	1	3	1	3	2
7	1	4	2	4	3
10	2	5	3	6	4
12	3	6	4	7	5
15	3	7	5	9	6

pacientes são geradas a fim de avaliar o tempo computacional de cada formulação. No segundo teste, são gerados 7 grupos de instâncias com disponibilidades ou números de módulos diferentes com objetivo de analisar as alterações nas soluções. Para esses grupos são geradas instâncias com 5, 7, 10, 12 e 15 pacientes. Utilizamos 7 sementes Taillard (1993) para gerar 245 instâncias no total.

4.2.2 Avaliação comparativa das formulações

Nessa seção apresentaremos os experimentos realizados para comparar o modelo proposto com o modelo MMBJS analisando o tempo computacional necessário para a resolução de cada formulação. Como visto anteriormente, no modelo MMBJS o valor do *makespan* sofre interferência sempre que houver algum paciente alocado ao módulo fictício, uma vez que as atividades desse paciente são consideradas no cálculo do *makespan*. Dessa forma, os experimentos são realizados para instâncias em que o valor do *makespan* é calculado corretamente, ou seja, quando todos os pacientes são atendidos no horizonte de tempo definido.

Para essa análise são consideradas 7 instâncias com 10 pacientes, 16 recursos, 47 módulos reais distribuídos em 5 dias e 1 módulo fictício para o MMBJS. Os pacientes estão divididos em especialidades, sendo 2 da cirurgia ginecológica, 5 da cirurgia ortopédica e 3 da cirurgia torácica. Existem 2 módulos de pré-cirurgia e 4 módulos de cirurgia disponíveis 7 horas em cada um dos 5 dias. Os módulos de pós-cirurgia são divididos em Sala de Recuperação (SR) e Unidade de Tratamento Intensivo (UTI). Estão disponíveis 3 SR com 8,5 horas diárias, enquanto 2 UTI ficam disponíveis do início ao fim do horizonte de tempo. O módulo fictício exclusivo do modelo MMBJS tem início no fim do horizonte de tempo e tem duração suficiente para atender todos os pacientes. Para definir o valor de α, que não é determinado no modelo MMBJS, realizou-se os experimentos atribuindo vários valores à esse parâmetro.

Tabela 15: Comparação entre os modelos MMBJS, MMBJS* e Proposto: solução.

Instância	*Makespan*	Pacientes Atendidos	α
1	6761	10	0,0005
2	6560	10	0,0010
3	6783	10	0,5000
4	6736	10	0,5000
5	6603	10	0,0100
6	6804	10	0,5000
7	6633	10	0,5000

Tabela 16: Comparação entre os modelos MMBJS, MMBJS* e Proposto: tempos e *GAP*.

Instância	*MMBJS*		*MMBJS**		*Proposto*	
	Tempo(s)	*GAP*(%)	Tempo(s)	*GAP*(%)	Tempo(s)	*GAP*(%)
1	247,97	00,00	196,59	00,00	169,19	00,00
2	338,99	00,00	90,38	00,00	256,82	00,00
3	7214,19	29,90	7200,37	21,25	144,08	00,00
4	7206,98	13,25	1200,00	00,00	120,19	00,00
5	146,58	00,00	117,15	00,00	84,03	00,00
6	922,54	00,00	1188,33	00,00	407,17	00,00
7	7238,41	23,88	749,04	00,00	154,46	00,00

Na Tabela 15, a primeira coluna é a identificação das instâncias testadas e em seguida são apresentados os valores do *makespan*, α e o número de pacientes atendidos. Verifica-se que, com o valor do α adequado, é possível encontrar um resultado equivalente ao modelo proposto. Infelizmente encontrar o valor adequado do α não é uma tarefa trivial e, portanto, cada instância teve que ser testada várias vezes alterando esse parâmetro até encontrar um resultado equivalente. No total, oito valores para o parâmetro α foram testados (0,5; 0,1; 0,05; 0,01; 0,005; 0,001; 0,0005 e 0,0001).

Adicionalmente, na Tabela 16 é apresentado, para cada uma das sete instâncias, o tempo computacional em segundos necessário para chegar ao resultado ótimo ou o tempo máximo configurado para rodadr o problema utilizando os modelos MMBJS e o modelo proposto e o GAP encontrado. Na coluna MMBJS* estão os tempos computacionais utilizando o modelo MMBJS com alteração das restrições (3.3) e (3.4) a fim, de reduzir o número de restrições, e, possivelmente o tempo computacional. Em algumas instâncias o *software* CPLEX não encontrou uma solução ótima no tempo máximo estipulado de 2 horas (7200 segundos) para os modelos MMBJS e MMBJS*, portanto existem *GAPs* associados aos resultados encontrados que também são apresentados na Tabela 16.

Em regra geral, observa-se que os tempos computacionais do modelo MMBJS, sem a

substituição das restrições, é consideravelmente maior se comparado com os outros dois modelos. A única exceção foi a instância 6, na qual a substituição das restrições piorou o tempo computacional em 22%. Das 7 instâncias testadas, o modelo MMBJS não foi capaz de encontrar a solução ótima em 2 horas para 3 delas, com *GAP* médio de 22,34%.

Comparando o modelo MMBJS*, com a substituição das restrições, com o modelo proposto, verifica-se uma considerável redução no tempo computacional em algumas instâncias, chegando a uma redução máxima de 98%. Uma exceção foi a instância 2 que apresentou melhor tempo no modelo MMBJS*, com uma redução de 65%. O *solver* utilizando o modelo proposto encontrou a solução ótima em todas as 7 instâncias testadas, enquanto que utilizando o modelo MMBJS* não pode encontrar uma solução ótima em uma das instâncias, encontrando uma solução com *GAP* de 21,25%.

Através desse experimento foi possível verificar que o modelo proposto consegue encontrar soluções ótimas equivalentes ao modelo MMBJS em problemas que todos os pacientes são atendidos. No caso de problemas em que pelo menos um paciente não é atendido, o modelo MMBJS não retorna o valor correto do *makespan*, portanto não é possível fazer uma avaliação comparativa com o modelo proposto. Adicionalmente, algumas vantagens são verificadas em relação ao modelo MMBJS: A eliminação do parâmetro α permite redução no tempo computacional, uma vez que não será necessário testar vários valores para esse parâmetro; A substituição das restrições (3.3) e (3.4) também pode reduzir o tempo computacional. Dessa forma, o modelo proposto apresenta-se como uma boa alternativa ao modelo MMBJS com considerável redução do tempo computacional para problemas em que todos os pacientes são atendidos.

4.2.3 Análise de sensibilidade para o modelo proposto

Nessa seção são apresentados experimentos com diferentes arranjos de módulos a fim de analisar o comportamento do modelo proposto, tomando como base uma configuração inicial (*Grupo 0*). Os dois termos da função objetivo, número de pacientes atendidos e *makespan* são estudados em cada caso com finalidade de compreender as alterações no sistema. Três características do sistema foram modificadas: disponibilidade dos módulos, exceto dos módulos de UTI; número de módulos de pós-cirurgia de UTI e número de módulos de pré-cirurgia. Para cada característica, dois grupos de instâncias foram criados. No primeiro grupo, adicionou-se módulos ou incrementou-se a disponibilidade dos módulos em relação à configuração inicial, deixando o sistema com maior folga. No segundo grupo, a disponibilidade dos módulos é reduzida ou módulos são excluídos da programação,

Tabela 17: Resumo das características dos grupos de instâncias

Grupo	Módulos							
	Pré		Cirurgia		Pós SR		Pós UTI	
	quant.	disp.	quant.	disp.	quant.	disp.	quant.	disp.
0	2	6h	4	6h	3	7h	2	24h
1.1	2	**7h**	4	**7h**	3	**8h30**	2	24h
1.2	2	**5h**	4	**5h**	3	**5h30**	2	24h
2.1	2	6h	4	6h	3	7h	**3**	24h
2.2	2	6h	4	6h	3	7h	**1**	24h
3.1	**3**	6h	4	6h	3	7h	2	24h
3.2	**1**	6h	4	6h	3	7h	2	24h

consequentemente, o sistema fica mais restrito.

As características de cada um dos grupos são resumidas na Tabela 17. O *Grupo 0* apresenta a configuração inicial que é utilizada como base de comparação com os demais grupos. A quantidade de módulos e suas disponibilidades estão apresentadas na tabela e as modificações de cada grupo, em relação à configuração inicial, estão destacadas em negrito. Uma vez que o sistema possui mais disponibilidade nos *Grupos 1.1, 2.1* e *3.1*, espera-se que o número de pacientes atendidos aumente e/ou o valor do *makespan* diminua em relação ao *Grupo 0*. Por outro lado, nos *Grupos 1.2, 2.2* e *3.2* a disponibilidade dos módulos é mais restrita, então, a expectativa é que o número de pacientes atendidos reduza e/ou o valor o *makespan* aumente.

A Tabela 18 apresenta os resultados obtidos em todas as instâncias testadas no *Grupo 0*. Na coluna *ID* é apresentado a identificação da instância com o número do grupo, quantidade de pacientes disponíveis no sistema e o número de identificação da semente utilizada para gerar os dados. O valor do *makespan*, o número de pacientes atendidos e a lista dos pacientes não atendidos indicam parte da solução encontrada a partir da resolução das instâncias utilizando o modelo matemático proposto. O tempo utilizado para a realização dos testes foi delimitado em 2 horas (7200s), o tempo computacional indica o tempo necessário para chegar à solução ótima, porém, se ao final do tempo máximo a solução ótima não for encontrada, o *GAP* indica a diferença percentual entre a solução incumbente e o limitante inferior determinado pelo *solver*. Utilizou-se a tolerância para o *GAP* de 0,005%, ou seja, uma vez que uma solução encontra esse *GAP* a otimização é finalizada. As otimizações que atingem essa tolerância são indicadas nas tabelas com a indicação (*).

Observe que todos os pacientes do sistema são atendidos nas instâncias com 5 e 7 pacientes e que, conforme aumenta o número de pacientes, o número de pacientes que não

Tabela 18: Análise sensibilidade - Grupo 0.

	ID	*Makespan*	# pacientes atendidos	pacientes não atendidos	*GAP*(%)	Tempo computacional(s)
5 pacientes	0_5_1	2d 13h 54min	5	-	0,00*	3,87
	0_5_2	1d 17h 02min	5	-	0,00*	1,55
	0_5_3	2d 12h 42min	5	-	0,00	1,93
	0_5_4	2d 13h 10min	5	-	0,00	9,53
	0_5_5	2d 13h 08min	5	-	0,00*	4,51
	0_5_6	2d 13h 22min	5	-	0,00	3,91
	0_5_7	2d 13h 16min	5	-	0,00	1,79
7 pacientes	0_7_1	3d 13h 24min	7	-	0,00*	16,02
	0_7_2	2d 18h 09min	7	-	0,00	13,08
	0_7_3	3d 13h 53min	7	-	0,00*	11,57
	0_7_4	3d 13h 46min	7	-	0,00*	12,87
	0_7_5	2d 16h 51min	7	-	0,00*	13,01
	0_7_6	3d 13h 14min	7	-	0,00*	41,10
	0_7_7	3d 14h 51min	7	-	0,00*	18,00
10 pacientes	0_10_1	3d 13h 43min	8	6,9	0,00*	203,40
	0_10_2	4d 14h 35min	10	-	0,00*	155,33
	0_10_3	3d 15h 34min	8	9, 10	0,00*	305,11
	0_10_4	4d 15h 00min	9	8	0,00*	256,55
	0_10_5	4d 14h 07min	10	-	0,00*	169,66
	0_10_6	4d 18h 52min	10	-	0,00*	137,39
	0_10_7	4d 12h 53min	9	2	0,00*	92,12
12 pacientes	0_12_1	3d 14h 41min	9	1,6,9	0,00	5234,92
	0_12_2	4d 17h 16min	11	6	0,00*	280,50
	0_12_3	4d 18h 16min	10	9,10	0,00*	314,52
	0_12_4	4d 15h 12min	10	8,10	0,00*	463,40
	0_12_5	4d 16h 30min	11	11	0,00*	176,55
	0_12_6	4d 15h 51min	10	8,10	0,00*	466,42
	0_12_7	4d 12h 53min	10	1,2	0,00*	1526,61
15 pacientes	0_15_1	4d 14h 35min	11	1,6,9,10	28,73	7266,51
	0_15_2	4d 17h 16min	12	1,6,14	4,67	7200,36
	0_15_3	4d 19h 41min	11	7,9,10,13	1,54	7200,30
	0_15_4	4d 15h 12min	11	6,8,10,14	20,27	7200,35
	0_15_5	4d 18h 15min	12	2,11,12	2,52	7200,39
	0_15_6	4d 17h 20min	12	4,8,10	0,00*	6090,29
	0_15_7	4d 15h 12min	12	1,2,13	0,02	7200,31

* indica que o *GAP* de tolerância foi atingido

são atendidos também aumenta, isso ocorre, pois o sistema passa a ficar sobrecarregado e não tem mais disponibilidade para entender os pacientes adicionais. O valor do *makespan* também se altera com o número de pacientes da instância, essa variável vai aumentando a medida que o sistema vai atendendo mais pacientes.

Analisando o número de pacientes atendidos nas instâncias 0_10_4 e 0_12_4 temos que um aumento de 9 para 10 pacientes atendidos, ou seja, a solução para 12 pacientes consegue atender 10 pacientes e a instância com 10 pacientes atende somente 9 com os mesmos recursos. Esse resultado pode ser justificado pelo fato da solução de 0_12_4 incluir dois pacientes que não estavam disponíveis na instância 0_10_4, os pacientes 11 e 12. Dessa forma, a instância 0_10_4 apesar de ter 10 pacientes disponíveis para atendimento, não poderia ter o mesmo resultado encontrado em 0_12_4. Por outro lado, o número mínimo de pacientes que a instância 0_12_4 deveria atender é 9, já que que todos os pacientes disponíveis em 0_10_4 estão disponíveis em 0_12_4 e, portanto, é possível encontrar a mesma programação. Pode-ser concluir que é possível que instâncias maiores atendam mais pacientes que instâncias menores, porém o contrário nunca pode acontecer.

Para instâncias pequenas (número de pacientes ≤ 7), o *solver* utilizando o modelo proposto encontra a solução ótima em um tempo computacional reduzido (< 45 segundos), conforme o número de pacientes aumenta, o tempo computacional necessário para encontrar uma solução ótima aumenta, média de 2min e 34s para as instâncias com 10 pacientes. Nas instâncias com 15 pacientes, o *solver* já não consegue, na maioria dos casos, encontrar uma solução ótima em 2 horas de processamento, apresentando um *GAP* de até 28,73%.

No *Grupo 1.1* a disponibilidade dos módulos de pré-cirurgia, cirurgia e pós-cirurgia em SR é aumentada, resultando em um sistema mais ocioso capaz de atender mais pacientes e/ou atender o mesmo número de pacientes em um tempo menor. Os resultados apresentados na Tabela 19 confirmam essa suposição. Observe que nas instâncias menores o *makespan* encontrado é menor ou igual aquele encontrado no *Grupo 0*, enquanto que na maioria das instâncias maiores, o número de pacientes atendidos aumenta, conforme esperado. Adicionalmente, o número de instâncias que o *software* não encontra uma solução ótima em duas horas é menor (8,6%) comparativamente com o *Grupo 0* (17,1%) e o maior GAP apresentado é 10,59%.

No *Grupo 1.2*, as disponibilidade dos módulos é reduzida fazendo com que o sistema fique mais restrito. Espera-se que o número de pacientes atendidos reduza e que o *makespan* aumente para o mesmo número de pacientes atendidos no *Grupo 0*. A Tabela 20

Tabela 19: Análise sensibilidade - Grupo 1.1.

	ID	*Makespan*	# pacientes atendidos	pacientes não atendidos	*GAP*(%)	Tempo computacional(s)
5 pacientes	1.1_5_1	1d 16h 26min	5	-	0,00	4,65
	1.1_5_2	1d 15h 23min	5	-	0,00	1,79
	1.1_5_3	1d 17h 13min	5	-	0,00	1,43
	1.1_5_4	1d 15h 31min	5	-	0,00*	1,58
	1.1_5_5	1d 15h 12min	5	-	0,00*	3,92
	1.1_5_6	1d 16h 27min	5	-	0,00	4,55
	1.1_5_7	1d 17h 19min	5	-	0,00	1,19
7 pacientes	1.1_7_1	2d 16h 58min	7	-	0,00*	12,32
	1.1_7_2	2d 17h 16min	7	-	0,00	14,52
	1.1_7_3	3d 13h 46min	7	-	0,00*	11,29
	1.1_7_4	2d 17h 46min	7	-	0,00*	12,12
	1.1_7_5	2d 16h 51min	7	-	0,00	30,99
	1.1_7_6	2d 17h 34min	7	-	0,00	14,51
	1.1_7_7	3d 14h 51min	7	-	0,00*	42,99
10 pacientes	1.1_10_1	4d 16h 41min	10	-	0,00*	169,19
	1.1_10_2	4d 13h 20min	10	-	0,00*	256,82
	1.1_10_3	4d 17h 03min	10	-	0,00*	144,08
	1.1_10_4	4d 16h 16min	10	-	0,00*	120,19
	1.1_10_5	4d 14h 03min	10	-	0,00*	84,03
	1.1_10_6	4d 17h 24min	10	-	0,00*	407,17
	1.1_10_7	4d 14h 33min	10	-	0,00*	154,46
12 pacientes	1.1_12_1	4d 16h 41min	11	9	0,00*	406,94
	1.1_12_2	4d 13h 42min	11	6	0,00*	828,83
	1.1_12_3	4d 17h 03min	11	10	0,00*	308,29
	1.1_12_4	4d 16h 16min	11	8	0,00*	635,18
	1.1_12_5	4d 14h 03min	11	11	0,00*	548,68
	1.1_12_6	4d 16h 18min	11	8	0,00*	466,82
	1.1_12_7	4d 14h 33min	11	2	0,00*	264,75
15 pacientes	1.1_15_1	4d 15h 23min	13	6,9	10,59	7244,93
	1.1_15_2	4d 14h 34min	13	1,6	7,54	7200,30
	1.1_15_3	4d 15h 03min	13	9,10	0,00*	1645,91
	1.1_15_4	4d 16h 16min	13	8,10	0,00*	1831,44
	1.1_15_5	4d 14h 07min	13	10,11	7,01	7200,34
	1.1_15_6	4d 15h 06min	13	8,10	0,00*	930,88
	1.1_15_7	4d 14h 33min	13	2,13	0,00*	1803,00

* indica que o *GAP* de tolerância foi atingido

Tabela 20: Análise sensibilidade - Grupo 1.2.

	ID	*Makespan*	# pacientes atendidos	pacientes não atendidos	*GAP* (%)	Tempo computacional (s)
5 pacientes	1.2_5_1	3d 12h 45min	5	-	0,00*	13,75
	1.2_5_2	2d 13h 27min	4	5	0,00*	8,08
	1.2_5_3	2d 14h 48min	5	-	0,00	7,69
	1.2_5_4	2d 14h 42min	5	-	0,00	4,09
	1.2_5_5	2d 14h 53min	5	-	0,00	5,31
	1.2_5_6	2d 13h 22min	5	-	0,00	3,66
	1.2_5_7	2d 16h 30min	5	-	0,00	3,80
7 pacientes	1.2_7_1	3d 15h 11min	7	-	0,00*	6,63
	1.2_7_2	4d 12h 54min	6	5	0,00	7,76
	1.2_7_3	4d 14h 33min	7	-	0,00	15,12
	1.2_7_4	4d 13h 14min	7	-	0,00	15,50
	1.2_7_5	4d 12h 42min	6	5	0,00*	17,73
	1.2_7_6	4d 13h 10min	7	-	0,00*	15,56
	1.2_7_7	4d 13h 12min	7	-	0,00*	5,96
10 pacientes	1.2_10_1	4d 13h 25min	8	6,9	0,00*	144,30
	1.2_10_2	4d 12h 54min	7	1,5,6	0,00	93,63
	1.2_10_3	4d 15h 09min	8	8,10	0,00*	161,63
	1.2_10_4	4d 14h 29min	8	8,9	0,00*	139,80
	1.2_10_5	4d 12h 54min	7	2,5,10	0,00*	90,14
	1.2_10_6	4d 14h 03min	8	8,10	0,00*	81,72
	1.2_10_7	4d 13h 12min	8	1,2	0,00*	71,76
12 pacientes	1.2_12_1	4d 13h 25min	9	6,9,10	0,00*	629,76
	1.2_12_2	4d 13h 26min	8	1,5,6,10	0,00*	256,36
	1.2_12_3	4d 15h 03min	9	8,9,10	0,00*	320,41
	1.2_12_4	4d 14h 57min	9	8,9,10	0,00*	443,55
	1.2_12_5	4d 12h 54min	8	2,5,10,11	0,00*	281,39
	1.2_12_6	4d 13h 11min	9	6,8,10	0,00*	160,02
	1.2_12_7	4d 13h 12min	9	1,2,8	0,00*	102,35
15 pacientes	1.2_15_1	4d 14h 35min	10	1,6,9,10,12	0,00*	1722,51
	1.2_15_2	4d 13h 26min	9	1,5,6,10,13,14	0,00*	2456,22
	1.2_15_3	4d 15h 03min	9	5,7,8,9,10,11	0,00*	2312,14
	1.2_15_4	4d 14h 29min	9	6,7,8,9,10,14	21,07	7214,71
	1.2_15_5	4d 13h 14min	8	2,4,5,10,11,12,13	0,00*	4152,80
	1.2_15_6	4d 14h 19min	10	1,6,8,10,14	0,00*	4751,38
	1.2_15_7	4d 13h 27min	10	1,2,7,11,13	0,00*	702,43

* indica que o *GAP* de tolerância foi atingido

apresenta os resultados encontrados para as instâncias deste grupo. Conforme esperado, o número de pacientes atendidos é reduzido consideravelmente de forma que até para instância com 5 pacientes (1.2_5_2) o *solver* utilizando o modelo proposto não seja capaz de encontrar uma solução que atenda todos os pacientes. Observando as instâncias 0_10_1 e 1.2_10_1 temos que, em ambas, 8 pacientes são atendidos pelo sistema, mas como no *Grupo 1.2* o sistema é mais restrito o valor do *makespan* aumenta em 1.422 minutos em relação ao resultado da instância do *Grupo 0*. Nesse grupo somente para uma instância o *solver* não encontrou o valor ótimo em 2 horas de processamento.

Os resultados encontrados com a adição de um módulo de pós-cirurgia em UTI na configuração inicial são apresentados na Tabela 21. Devem aparecer mudanças entre os resultados desse grupo e do *Grupo 0*, se o módulo de pós-cirurgia em UTI for um módulo restritivo, ou seja, se esse módulo restringir o atendimento de um paciente adicional. Da Tabela 21 observa-se que o número de pacientes atendidos com a adição de um módulo de pós-cirurgia em UTI aumenta para instâncias com 10, 12 e 15 pacientes, chegando a atender até 2 pacientes a mais. Para instâncias com 7 pacientes, o número de pacientes atendidos se mantém em 7, mas o *makespan* encontrado é inferior ao encontrado no *Grupo 0* (redução média de 16h 36min). Já as instâncias com 5 pacientes não sofrem qualquer alteração, isso significa que esse módulo não é restritivo quando o sistema tem que atender só 5 pacientes. Nesse grupo existem 7 instâncias em que o *solver* não é capaz de encontrar uma solução ótima em 2 horas, uma a mais que o *Grupo 0*.

Dos resultados apresentados na Tabela 22, concluiu-se que o módulo de pós-cirurgia em UTI restringe o sistema para instâncias com número de pacientes maior ou igual a 7. Como exemplo, tomemos as instâncias 0_10_1 e 2.2_10_1, observe que o número de pacientes atendidos quando existem dois módulos de pós-cirurgia em UTI é 8 e quando retira-se um desses módulos esse valor é reduzido para 6 pacientes atendidos. O número de pacientes atendidos nesse grupo é reduzido em 71% das instâncias, enquanto o valor do *makespan* aumenta para todas as instâncias que mantém o número de pacientes atendidos. Os resultados encontrados nesse grupo mostram a relevância de um bom dimensionamento do número de UTI em um hospital. Nesse grupo, todas as instâncias encontraram valores ótimos em menos de 2 horas.

O *Grupo 3.1* busca identificar se os módulos de pré-cirurgia restringem o sistema. Os resultados encontrados a partir da solução das instâncias adicionando um módulo de pré-cirurgia a configuração inicial são apresentados na Tabela 23. Analisando comparativamente com o *Grupo 0*, a adição de um módulo de pré-cirurgia permite uma redução do *makespan* em 31% das instâncias em que o *solver* encontrou uma solução ótima nos

Tabela 21: Análise sensibilidade - Grupo 2.1.

	ID	*Makespan*	# pacientes atendidos	pacientes não atendidos	*GAP*(%)	Tempo computacional(s)
5 pacientes	2.1_5_1	2d 13h 54min	5	-	0,00*	13,41
	2.1_5_2	1d 16h 32min	5	-	0,00*	3,25
	2.1_5_3	2d 12h 42min	5	-	0,00	1,44
	2.1_5_4	2d 13h 10min	5	-	0,00	3,39
	2.1_5_5	2d 13h 08min	5	-	0,00*	7,00
	2.1_5_6	2d 13h 22min	5	-	0,00	7,74
	2.1_5_7	2d 13h 16min	5	-	0,00	1,58
7 pacientes	2.1_7_1	2d 15h 46min	7	-	0,00*	44,29
	2.1_7_2	2d 14h 09min	7	-	0,00*	6,60
	2.1_7_3	2d 15h 00min	7	-	0,00*	29,44
	2.1_7_4	2d 14h 45min	7	-	0,00*	18,26
	2.1_7_5	2d 14h 31min	7	-	0,00*	11,95
	2.1_7_6	2d 16h 39min	7	-	0,00	55,97
	2.1_7_7	2d 17h 01min	7	-	0,00	38,24
10 pacientes	2.1_10_1	4d 13h 25min	10	-	0,00*	2369,01
	2.1_10_2	4d 12h 54min	10	-	0,00*	1496,47
	2.1_10_3	4d 13h 03min	10	-	0,00	2556,41
	2.1_10_4	3d 16h 00min	10	-	0,00*	5287,31
	2.1_10_5	3d 15h 18min	10	-	0,00*	392,11
	2.1_10_6	4d 12h 37min	10	-	0,00	4938,11
	2.1_10_7	4d 12h 53min	10	-	0,00*	3147,42
12 pacientes	2.1_12_1	4d 13h 25min	11	9	5,87	7200,30
	2.1_12_2	4d 14h 40min	12	-	0,02	7200,35
	2.1_12_3	4d 15h 42min	12	-	0,03	7200,32
	2.1_12_4	4d 15h 00min	12	-	0,01	7210,26
	2.1_12_5	4d 14h 26min	12	-	0,02	7200,06
	2.1_12_6	4d 14h 56min	12	-	0,02	7200,09
	2.1_12_7	4d 13h 36min	12	-	0,00*	387,01
15 pacientes	2.1_15_1	4d 16h 31min	13	9,12	8,05	7200,12
	2.1_15_2	4d 17h 11min	15	-	0,03	7210,87
	2.1_15_3	4d 15h 16min	13	7,10	4,23	7200,10
	2.1_15_4	4d 15h 12min	13	8,14	3,80	7200,47
	2.1_15_5	4d 15h 00min	13	11,12	3,53	7204,47
	2.1_15_6	4d 15h 51min	14	8	1,13	7200,09
	2.1_15_7	4d 19h 29min	14	2	1,18	7200,08

* indica que o *GAP* de tolerância foi atingido

Tabela 22: Análise sensibilidade - Grupo 2.2.

	ID	*Makespan*	# pacientes atendidos	pacientes não atendidos	*GAP* (%)	Tempo computacional(s)
5 pacientes	2.2_5_1	3d 13h 34min	5	-	0,00*	8,38
	2.2_5_2	3d 14h 49min	5	-	0,00*	0,86
	2.2_5_3	3d 15h 34min	5	-	0,00	1,04
	2.2_5_4	3d 14h 40min	5	-	0,00	1,36
	2.2_5_5	3d 14h 42min	5	-	0,00	1,10
	2.2_5_6	3d 14h 43min	5	-	0,00	0,93
	2.2_5_7	3d 16h 37min	5	-	0,00*	1,06
7 pacientes	2.2_7_1	3d 13h 34min	6	6	0,00	9,76
	2.2_7_2	4d 15h 34min	7	-	0,00	1,95
	2.2_7_3	3d 15h 34min	6	6	0,00	3,27
	2.2_7_4	3d 14h 40min	6	6	0,00*	10,54
	2.2_7_5	4d 14h 59min	7	-	0,00*	9,86
	2.2_7_6	4d 15h 36min	7	-	0,00	3,53
	2.2_7_7	3d 14h 51min	6	2	0,00*	5,28
10 pacientes	2.2_10_1	3d 14h 52min	6	1,6,9,10	0,00	85,61
	2.2_10_2	4d 17h 16min	7	1,6,9	0,00*	49,43
	2.2_10_3	3d 15h 34min	6	6,8,9,10	0,00	22,38
	2.2_10_4	4d 15h 12min	7	1,8,10	0,00	88,11
	2.2_10_5	4d 12h 54min	7	1,2,10	0,00	76,11
	2.2_10_6	4d 17h 16min	7	1,8,10	0,00*	29,83
	2.2_10_7	3d 13h 35min	6	1,2,8,10	0,00*	83,17
12 pacientes	2.2_12_1	3d 14h 52min	7	1,6,8,9,10	0,00	184,39
	2.2_12_2	4d 16h 57min	8	1,2,9,10	0,00*	144,10
	2.2_12_3	4d 18h 16min	8	6,8,9,10	0,00	120,35
	2.2_12_4	4d 15h 12min	8	1,6,8,10	0,00	128,72
	2.2_12_5	4d 12h 54min	8	1,2,10,11	0,00	125,91
	2.2_12_6	4d 16h 51min	8	1,8,9,10	0,00*	157,46
	2.2_12_7	3d 13h 40min	7	1,2,8,10,11	0,00*	206,63
15 pacientes	2.2_15_1	4d 12h 23min	9	1,6,8,9,10,13	0,00	6500,56
	2.2_15_2	4d 16h 57min	10	1,2,9,10,13	0,00*	423,92
	2.2_15_3	4d 18h 16min	9	6,7,8,9,10,13	0,00	2164,57
	2.2_15_4	4d 16h 49min	10	1,2,6,8,10	0,00	1033,06
	2.2_15_5	4d 14h 07min	10	1,2,10,11,13	0,00*	4549,86
	2.2_15_6	4d 14h 24min	10	1,6,8,9,10	0,00	443,77
	2.2_15_7	3d 14h 51min	9	1,2,8,10,11,13	0,00*	5448,09

* indica que o *GAP* de tolerância foi atingido

Tabela 23: Análise sensibilidade - Grupo 3.1.

	ID	*Makespan*	# pacientes atendidos	pacientes não atendidos	*GAP*(%)	Tempo computacional(s)
5 pacientes	3.1_5_1	2d 13h 54min	5	-	0,00*	4,2
	3.1_5_2	1d 17h 02min	5	-	0,00*	2,74
	3.1_5_3	2d 12h 42min	5	-	0,00*	0,58
	3.1_5_4	2d 13h 03min	5	-	0,00	0,64
	3.1_5_5	2d 13h 08min	5	-	0,00	0,65
	3.1_5_6	2d 13h 08min	5	-	0,00*	6,21
	3.1_5_7	2d 13h 14min	5	-	0,00	0,88
7 pacientes	3.1_7_1	3d 13h 24min	7	-	0,00*	18,39
	3.1_7_2	2d 18h 09min	7	-	0,00	8,1
	3.1_7_3	3d 13h 50min	7	-	0,00*	3,56
	3.1_7_4	3d 13h 46min	7	-	0,00*	10,4
	3.1_7_5	2d 16h 51min	7	-	0,00	14,23
	3.1_7_6	3d 13h 14min	7	-	0,00*	7,49
	3.1_7_7	3d 14h 49min	7	-	0,00*	12,79
10 pacientes	3.1_10_1	3d 13h 43min	8	6,9	0,00*	176,45
	3.1_10_2	4d 14h 35min	10	-	0,00*	59,85
	3.1_10_3	3d 15h 31min	8	8,10	0,00*	223,19
	3.1_10_4	4d 14h 49min	9	8	0,00*	239,81
	3.1_10_5	4d 14h 07min	10	-	0,00*	69,85
	3.1_10_6	4d 18h 52min	10	-	0,00*	200,82
	3.1_10_7	4d 12h 52min	9	2	0,00*	294,4
12 pacientes	3.1_12_1	3d 14h 41min	9	1,6,9	0,00*	1104,07
	3.1_12_2	4d 17h 16min	11	6	0,00*	160,6
	3.1_12_3	4d 18h 16min	10	8,10	0,00*	636,82
	3.1_12_4	4d 15h 03min	10	8,10	0,00*	479,67
	3.1_12_5	4d 16h 28min	11	11	0,00*	495,17
	3.1_12_6	4d 15h 51min	10	8,10	0,00*	510,73
	3.1_12_7	4d 12h 52min	10	1,2	0,00*	524,52
15 pacientes	3.1_15_1	4d 14h 35min	11	1,6,9,10	22,18	7200,23
	3.1_15_2	4d 17h 16min	12	1,6,14	0,00*	1969,64
	3.1_15_3	4d 16h 20min	10	5,7,8,9,10	15,61	7254,29
	3.1_15_4	4d 15h 3min	11	6,8,10,14	20,28	7200,22
	3.1_15_5	4d 15h 48min	12	2,11,12	2,5	7200,26
	3.1_15_6	4d 17h 10min	12	4,8,10	6,02	7200,17
	3.1_15_7	4d 15h 12min	12	1,2,13	0,02	7200,19

* indica que o *GAP* de tolerância foi atingido

dois grupos. O número de pacientes atendidos não sofreu nenhuma melhora em relação ao *Grupo 0* para as instâncias com $GAP = 0,00\%$.

Finalmente, o *Grupo 3.2* reduz em uma unidade o número de módulos de pré-cirurgia disponíveis por dia para atender os pacientes. A Tabela 24 apresenta os resultados dessas instâncias. Pode-se observar que o número de pacientes não atendidos aumenta em 1 ou 2 unidades em cerca de 66,7% das instâncias com 10, 12 e 15 pacientes. Para as instâncias que atendem o mesmo número de pacientes, o *makespan* encontrado pelo *solver* é pior em 81% das instâncias. Portanto, os módulos de pré-cirurgia são módulos restritivos para esse conjunto de instâncias.

Os experimentos realizados analisam o comportamento do modelo proposto com alterações nas disponibilidade e número de módulos presentes no sistema. Conforme esperado, as soluções encontradas pelo *solver* para instâncias mais restritas obtiveram resultados iguais ou piores que os resultados do *Grupo 0*. De maneira oposta, nas instâncias que adicionou-se ociosidade, o *solver* utilizando o modelo proposto encontrou resultados mais favoráveis. Assim, o modelo obteve o comportamento esperado para as instâncias testadas.

O tempo computacional máximo estipulado foi de 2 horas. O número de instâncias em que o *solver* não conseguiu encontrar uma solução ótima nesse período varia com a configuração dos módulos e o número de pacientes, mas para as instâncias de 15 pacientes esse valor é de cerca de 60% considerando os 7 grupos. Hospitais de grande porte realizam cirurgias de dezenas de pacientes em um único dia, como o problema tratado nesse trabalho está classificado como um *Job Shop Flexível* que é considerado um problema *NP-hard*, o tempo computacional necessário para encontrar o valor ótimo de instâncias de maiores dimensões cresce exponencialmente. Dessa forma, utilizar abordagens heurísticas é uma boa solução para encontrar um resultado satisfatório em um tempo computacional razoável.

Tabela 24: Análise sensibilidade - Grupo 3.2.

	ID	*Makespan*	# pacientes atendidos	pacientes não atendidos	*GAP*(%)	Tempo computacional (s)
5 pacientes	3.2_5_1	2d 13h 58min	5	-	0,00*	6,43
	3.2_5_2	2d 15h 08min	5	-	0,00*	2,95
	3.2_5_3	2d 14h 42min	5	-	0,00*	2,42
	3.2_5_4	2d 14h 32min	5	-	0,00*	6,22
	3.2_5_5	2d 14h 34min	5	-	0,00	3,22
	3.2_5_6	2d 13h 22min	5	-	0,00	2,18
	3.2_5_7	2d 14h 55min	5	-	0,00	4,53
7 pacientes	3.2_7_1	3d 14h 55min	7	-	0,00*	7,13
	3.2_7_2	3d 13h 11min	7	-	0,00*	7,44
	3.2_7_3	3d 14h 51min	7	-	0,00*	4,43
	3.2_7_4	3d 13h 48min	7	-	0,00*	4,04
	3.2_7_5	3d 13h 57min	7	-	0,00*	5,26
	3.2_7_6	3d 14h 38min	7	-	0,00*	7,74
	3.2_7_7	3d 16h 20min	7	-	0,00*	10,34
10 pacientes	3.2_10_1	4d 13h 16min	8	6,9	0,00*	238,49
	3.2_10_2	4d 14h 35min	9	6	0,00*	247,16
	3.2_10_3	4d 13h 16min	8	9,10	0,00	113,26
	3.2_10_4	4d 15h 04min	9	8	0,00*	135,65
	3.2_10_5	4d 14h 25min	9	10	0,00*	330,83
	3.2_10_6	4d 17h 13min	9	8	0,00*	148,46
	3.2_10_7	4d 14h 35min	9	2	0,00*	131,36
12 pacientes	3.2_12_1	4d 13h 27min	9	6,9,10	0,00*	1113,01
	3.2_12_2	4d 14h 32min	10	6,10	0,00*	1080,81
	3.2_12_3	4d 14h 23min	9	8,9,10	0,00*	354,91
	3.2_12_4	4d 15h 12min	10	8,10	0,00*	631,98
	3.2_12_5	4d 14h 25min	10	10,11	0,00*	1107,96
	3.2_12_6	4d 16h 48min	10	8,10	0,00*	268,65
	3.2_12_7	4d 14h 37min	10	1,2	0,00*	556,48
15 pacientes	3.2_15_1	4d 15h 0min	10	1,5,6,9,10	34,24	7209,04
	3.2_15_2	4d 14h 32min	10	1,6,10,14,15	18,27	7200,26
	3.2_15_3	4d 14h 53min	10	5,8,9,10,13	14,47	7200,19
	3.2_15_4	4d 14h 40min	10	6,8,9,10,14	16,63	7200,16
	3.2_15_5	4d 14h 45min	10	4,10,11,12,13	23,92	7211,41
	3.2_15_6	4d 18h 53min	10	4,5,8,10,14	0,00*	5652,64
	3.2_15_7	4d 14h 35min	10	1,2,5,8,13	13,91	7200,14

* indica que o *GAP* de tolerância foi atingido

5 HEURÍSTICA CONSTRUTIVA

As heurísticas construtivas são algoritmos que constroem soluções para problemas de otimização de forma gradativa. Ou seja, uma solução é criada passo a passo, onde em cada passo são adicionados componentes individuais até a formação completa da solução. Tipicamente são utilizadas regras para definir a ordem da adição desses componentes individuais, chamadas de regras de ordenação (MOCCELLIN *et al.*,2018).

O pequeno esforço computacional desse método é uma das razões que motiva o seu estudo em aplicações práticas. A literatura mostra que apesar de não garantir a otimalidade das soluções geradas, esse método apresenta bom desempenho quando as características do problema são bem exploradas (DAMM, 2016).

Diversas heurísticas construtivas estão disponíveis na literatura tanto para problemas de *scheduling* (*e.g.* DOH *et al.*, 2013; GHOLAMI; SOTSKOV, 2014; MELO; RONCONI, 2015; ROSSI; NAGANO; SAGAWA, 2017), como para vários outros problemas clássicos de otimização (*e.g.* XU; CHIU, 2001; SOBREIRO; NAGANO, 2013). As heurísticas construtivas também podem ser utilizadas para gerar uma solução inicial de boa qualidade para que, posteriormente, o espaço de soluções viáveis seja explorado por uma metaheurística.

Neste capítulo é apresentada uma proposta de heurística construtiva para o problema de programação de cirurgias eletivas com múltiplos estágios e recursos. As características do problema são exploradas, a fim de encontrar regras de ordenação adequadas para solucionar o problema utilizando um pequeno esforço computacional.

5.1 Heurística proposta

No problema de programação de cirurgias estudado, os recursos disponíveis para processar os pacientes são finitos. Essa característica sugere que existe uma decisão de quantos e quais pacientes devem ser processados por esse conjunto de recursos. Dessa forma, é

Tabela 25: Hierarquia para alocação de recursos.

Pacientes			
	Etapa		
		Tipo de recurso	
			Recurso

permitido que alguns pacientes não sejam atendidos por falta de capacidade de processamento dos recursos.

Outra característica importante é que os pacientes possuem múltiplas etapas que devem ser processadas por vários recursos diferentes. Ou seja, se faz necessário programar todas as etapas com todos os recursos necessários deste paciente para considerar que seu processamento foi finalizado. Dessa forma, não é vantajoso para solução do problema programar somente parte das etapas de um pacientes, ou programar etapas sem que todos os recursos necessários sejam alocados.

Considerando essas duas características do problema, o método hurístico proposto impõe que o resultado encontrado seja sempre uma solução em que todos os pacientes tenham todas suas etapas e recursos alocados. Ou seja, não é permitido que somente uma fração das etapas de um paciente seja programada, ou que uma etapa seja programada sem que todos os recursos necessários sejam alocados.

Para garantir esse resultado, o modelo matemático proposto utiliza o conceito de *módulo de recurso* que representa um conjunto de recursos disponíveis para processar a etapa de um paciente. Porém, essa abordagem não leva em conta o tempo de pré-processamento dos dados de entrada. Para gerar um *módulo* é necessário analisar as janelas de tempo dos recursos a fim de garantir que todos os recursos estejam disponíveis no tempo que o módulo está disponível. A heurística construtiva proporta busca tratar esse problema de uma forma diferente, excluindo o conceito de *módulo*, alocando cada recurso separadamente e utilizando uma hierarquia de alocação.

Na Tabela 25 é apresentada a hierarquia para alocação de recursos considerada para o desenvolvimento da heurística proposta. Considere um exemplo com três pacientes P1,P2,P3, no qual cada paciente deverá passar por duas etapas S1, S2. Na primeira etapa é necessária a alocação de um cirurgião e uma sala de cirurgia enquanto na segunda, somente uma sala de recuperação é necessária. Estão disponíveis dois cirurgiões Cirurgião1, Cirurgião2, três salas de cirurgia Sala1, Sala2, Sala3 e duas salas de recuperação SR1, SR2.

No primeiro nível, um paciente é escolhido para ser processado. Um conjunto de regras de ordenação são utilizados para definir a sequência de programação de pacientes. Considerando o exemplo, a regra de ordenação define como ordem P3, P1, P2, dessa forma, o paciente P3 é o primeiro a ser alocado.

Após a escolha do paciente, suas etapas são consideradas uma a uma em ordem pré-definida pela necessidade do paciente. Não é permitido programar uma etapa sem que todas as etapas anteriores tenham sido programadas. Ou seja, é programada primeiro a etapa S1 do paciente P3.

Uma vez que cada etapa pode ser processada por um ou mais tipos de recursos, é necessário escolher qual o tipo de recurso será escolhido para ser alocado. Para a etapa S1 do paciente P3 dois tipos de recursos são necessários: *cirurgião* e *sala de cirurgia*. Regras de ordenação são utilizadas para definir a ordem mais adequada para alocar cada tipo de recurso. No exemplo, a regra de ordenação determina que seria mais adequado alocar primeiro o tipo de recursos *cirurgião*.

A última escolha é realizada no nível do recurso. No exemplo, dentre os recursos disponíveis do tipo de recurso *cirurgião*, definiu-se utilizado uma regra de ordenação que a ordem para testar os recursos deveria ser Cirurgião2, Cirurgião1.

Após a alocação do cirurgião, a algoritmo avança para o próximo tipo de recurso (sala de cirurgia) e aloca um recurso disponível. Com todos os recursos da etapa S1 alocados, o algoritmo avança para a próxima etapa. Concluídas as alocações de todas as etapas do paciente P3, o algoritmo passa para o próximo paciente da lista de prioridades (P1).

Caso não seja possível alocar um tipo de recurso, todas as etapas já finalizadas são canceladas e o paciente é excluído da programação. Dessa forma, hierarquicamente, os recursos são alocados garantindo que todas as etapas de todos os pacientes foram alocadas com todos os recursos necessários.

Na sequência, o pseudocódigo da heurística construtiva proposta é apresentado com uma breve discussão do seu funcionamento. A Tabela 26 indica todas as notações utilizadas no pseudocódigo.

Na Seção 5.2 são apresentadas as regras de ordenação utilizadas no algoritmo. Uma regra de ordenação será utilizada para cada um dos níveis hierárquicos apresentados na Tabela 25, exceto para o nível *Etapa* que é alocada de forma cronológica.

Tabela 26: Notações utilizanda na Heurística Construtiva.

Conjunto	
P	Conjunto de pacientes p
S_p	Conjunto de etapas s necessárias para processar o paciente $p \in P$
T	Conjunto de tipos de recursos t
R^t	Conjunto de recursos r do tipo $t \in T_{sp}$ disponíveis
Parâmetros	
$P[k]$	Indica o k-ésimo paciente ordenado segundo algum critério
$T_{sp}[u]$	Indica o u-ésimo tipo de recurso ordenado segundo algum critério
$R^t[z]$	Indica o z-ésimo recurso ordenado segundo algum critério

Algoritmo 1 Heurística Construtiva - Parte 1.

1: **entrada:** parâmetros referentes aos pacientes e recursos
2: **saída:** alocações de recursos nas etapas dos pacientes
3: Ordene no vetor $P[\ \cdot\ ]$ os pacientes $p \in P$ conforme *regra de ordenação de pacientes*
4: **para** $k \leftarrow 1$ **até** $|P|$ **faça** ▷ Escolhe o paciente
5: $\quad p \leftarrow P[k]$
6: $\quad$ **para** $s \leftarrow 1$ **até** $|S_p|$ **faça** ▷ Escolhe a etapa
7: $\quad\quad$ Ordene no vetor $T_{sp}[\ \cdot\ ]$ os tipos de recursos $t \in T_{sp}$ conforme *regra de ordenação de tipos de recursos*
8: $\quad\quad$ **para** $u \leftarrow 1$ **até** $|T_{sp}|$ **faça** ▷ Escolhe o tipo de recurso
9: $\quad\quad\quad t \leftarrow T_{sp}[u]$
10: $\quad\quad\quad$ Ordene no vetor $R^t[\ \cdot\ ]$ os recursos $r \in R^t$ conforme *regra de ordenação de recursos*
11: $\quad\quad\quad$ **para** $z \leftarrow 1$ **até** $|R^t|$ **faça** ▷ Escolhe o recurso
12: $\quad\quad\quad\quad r \leftarrow R^t[z]$
13: $\quad\quad\quad\quad$ **se** r é recurso chave **então**
14: $\quad\quad\quad\quad\quad$ **se** é possível alocar o recurso **então**
15: $\quad\quad\quad\quad\quad\quad$ Alocar recurso
16: $\quad\quad\quad\quad\quad$ **senão**
17: $\quad\quad\quad\quad\quad\quad z \leftarrow z + 1$
18: $\quad\quad\quad\quad\quad\quad$ vá para a linha 13
19: $\quad\quad\quad\quad\quad$ **fim se**
20: $\quad\quad\quad\quad$ **senão**
21: $\quad\quad\quad\quad\quad$ **se** é possível alocar recurso no mesmo instante que o recurso chave **então**
22: $\quad\quad\quad\quad\quad\quad$ Alocar recurso
23: $\quad\quad\quad\quad\quad$ **senão**
24: $\quad\quad\quad\quad\quad\quad z \leftarrow z + 1$
25: $\quad\quad\quad\quad\quad\quad$ vá para a linha 13
26: $\quad\quad\quad\quad\quad$ **fim se**
27: $\quad\quad\quad\quad$ **fim se**

Algoritmo 2 Heurística Construtiva - Parte 2.

```
28:                 se nenhum recurso foi alocado então
29:                     z ← 1                                                    ▷ início da lista de recursos
30:                     se é possível modificar todos os recursos já alocados para atender a disponibilidade do recurso em teste então
31:                         Alterar todos os recursos já alocados ao paciente
32:                         Alocar recurso
33:                     senão
34:                         z ← z + 1
35:                         vá para a linha 13
36:                     fim se
37:                 fim se
38:                 se nenhum recurso foi alocado então
39:                     Cancelar a alocação de todos os recursos já alocados ao paciente
40:                     k ← k + 1
41:                     vá para a linha 6
42:                 fim se
43:                 z ← z + 1
44:             fim para
45:             u ← u + 1
46:         fim para
47:         s ← s + 1
48:     fim para
49:     k ← k + 1
50: fim para
```

No pseudocódigo o processo de desenvolvimento da solução inicia sequenciando todos os pacientes, usando uma *regra de ordenação de pacientes*, e em seguida o paciente mais prioritário é escolhido para iniciar o procedimento de alocação (*loop*: 4-50). As etapas do paciente (S_p) são sequenciadas em ordem cronológica e escolhidas uma a uma para serem processadas (*loop*: 6-48).

O conjunto de recursos (R) são agrupados em conjuntos por *tipos de recursos* (R^t), por exemplo, existe um conjunto de recursos do tipo Enfermeiro que correspondem a todos os enfermeiros disponíveis no sistema. Um dos dados de entrada do procedimento heurístico é o conjunto de tipos de recursos necessários para cada etapa de cada paciente (T_{sp}).

Os tipos de recursos necessários para processar cada etapa são ordenados considerando uma *regra de ordenação de tipos de recursos* e um tipo de recurso é eleito para ser processado (*loop*: 8-46). Os recursos disponíveis daquele tipo de serviço são priorizados utilizando a *regra de ordenação de recursos* e um recurso é escolhido para ser testado (*loop*: 11-44).

O tempo de processamento ($proces_{sp}$) da etapa de um paciente pode variar de acordo com o conjunto de recursos escolhidos para processá-la. Para esse estudo, o recurso que determina o tempo de processamento é chamado de recurso chave. Por exemplo, se é necessário uma *sala de cirurgia* e um *cirurgião* para processar uma etapa de um paciente e o recurso *cirurgião* é chave. Então, o *cirurgião* é o recurso que armazena a informação do tempo de processamento da etapa. Caso exista mais de um *cirurgião* no sistema é possível que tempo de processamento da etapa seja diferente dependendo da escolha do recurso chave.

No método proposto, uma vez escolhido todos os elementos (paciente, etapa, tipo de recurso e recurso), o processo de alocação é iniciado. Como o *recurso chave* determina o tempo de processamento este sempre deve ser alocado primeiro. Portanto, o teste inicial (decisão: 13-20) verifica se o recurso em teste é chave. Em caso afirmativo, é testado (decisão: 14-19) se o recurso pode ser alocado considerando as restrições temporais do sistema, tais como: intervalo de disponibilidade do recurso e alocação de etapas anteriores do paciente. Se o recurso respeitar todas as restrições temporais ele é alocado, caso contrário, o próximo recurso do vetor R^t é selecionado e os testes são realizados novamente.

Caso o recurso não seja chave (decisão: 20-27), é obrigatório que já exista um recurso chave associado a essa etapa. Assim um teste (decisão: 21-26) verifica se o recurso escolhido pode ser alocado no mesmo instante do recurso chave escolhido, considerando

as restrições temporais do sistema. Caso afirmativo, o recurso é alocado no mesmo instante que o recurso chave. Caso contrário, o próximo recurso do vetor R^t é selecionado e os testes são realizados novamente. Em alguns casos, mesmo após passar por todo vetor R^t nenhum recurso é alocado. Um teste (decisão: 28-37) verifica se essa situação ocorre. Caso positivo, testa-se se é possível modificar a alocação dos recursos que já foram alocados para aquele paciente respeitando as restrições temporais (decisão: 30-36). Essas alterações na alocação dos recursos já alocados é denominada *Movimentos de ajuste* e são apresentadas com mais detalhes na Seção 5.1.1.

Exemplificando, se o tipo de recurso em teste é chave para a segunda etapa do sistema então, obrigatoriamente os recursos da primeira etapa já foram alocados. Considerando que nenhum recurso chave da segunda etapa é compatível temporariamente com estes recursos já alocados, altera-se os tempos da primeira etapa para que seja possível alocar algum recurso chave na segunda etapa. Caso seja possível realizar as alterações, elas são feitas e o recurso em teste é alocado. Caso contrário, o próximo recurso do vetor R^t é selecionado e os testes são realizados novamente.

Mesmo após considerar as alterações e percorrer todo o vetor R^t é possível que não exista um recurso que seja compatível com os recursos já alocados. Nessa situação é necessário cancelar todos os recursos já alocados àquele paciente e atualizar o vetor P para inciar os testes com o próximo paciente (decisão: 38-42).

5.1.1 Movimentos de ajuste

Nessa seção são apresentados alguns movimentos de ajuste na alocação dos recursos que são realizados durante a construção da solução heurística (decisão: 30-36). Esses movimentos são utilizados quando não é possível alocar algum tipo de recurso, sem modificar os recursos já alocados ao paciente anteriormente, podendo ser em etapas diferentes ou não. Observe que sem esses ajustes a alocação do paciente seria descartada.

Na Figura 13, apresenta-se um recorte do Gráfico de *Gantt* da programação da etapa 3 de um paciente, na qual é necessário realizar um movimento de ajuste para garantir sua alocação completa. Os recursos A, B e C possuem períodos de indisponibilidade que podem ser relativas à janela de tempo do recurso ou à utilização do recurso por outro paciente. As etapas dos pacientes são sempre acompanhadas das atividades de *setup* e limpeza.

Na primeira parte da figura, as etapas 1 e 2 foram alocadas no instante mais cedo

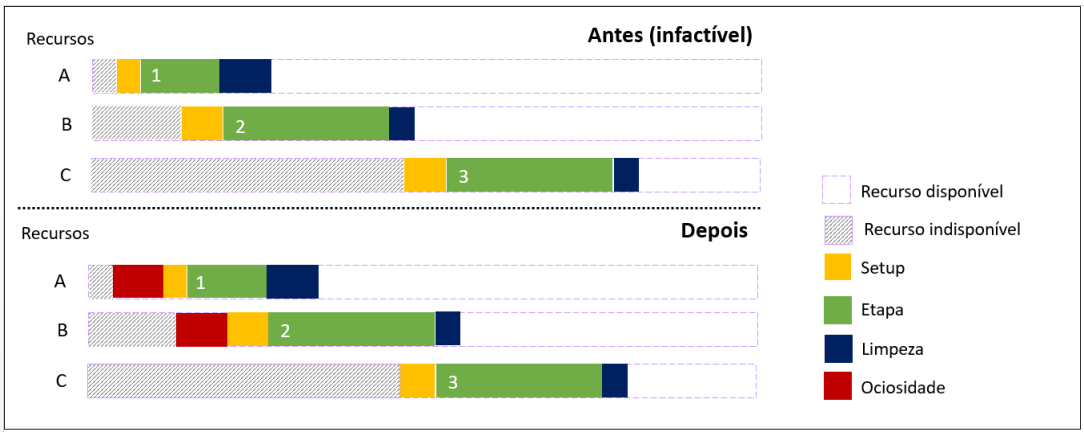

Figura 13: Exemplo da utilização de movimentos de ajuste. Fonte: Elaborado pela autora.

possível, porém ao tentar alocar a terceira etapa do paciente ocorre uma incompatibilidade temporal e portanto a solução torna-se infactível. Um movimento de ajuste é realizado para evitar que o paciente seja descartado da programação. Após o movimento, um período de ociosidade é adicionado aos recursos A e B para garantir que o fim da etapa 2 coincida com o início da etapa 3, tornando a solução factível.

Quando um recurso é escolhido para ser testado, algumas restrições temporais são avaliadas para garantir que a alocação seja compatível com os demais recursos já alocados. Três situações podem surgir dessa avaliação. Na Figura 14 são apresentadas essas situações: sem ociosidade, com ociosidade no paciente ou com ociosidade no recurso. A seguir são discutidas cada umas delas.

Na Figura 14 representa-se um recorte do Gráfico de *Gantt* da alocação da etapa 3 do mesmo paciente apresentado na Figura 5.1.1. Na primeira situação (a), a alocação mais cedo possível no Recurso C da terceira etapa do paciente coincide exatamente com o término do processamento da etapa anterior no Recurso B. Portanto, não ocorre ociosidade na alocação do Recurso C na etapa 3 do paciente.

Quando a ociosidade ocorre no recurso (b), denominado *folga*, o recurso em teste ainda está ocupado quando o paciente é liberado da etapa anterior. Do exemplo, a terceira etapa do paciente só pode iniciar depois que a segunda etapa esteja concluída. Para garantir as restrições temporais do paciente, é adicionado períodos de ociosidade nos Recursos A e B após o início da sua disponibilidade.

Na terceira situação (c), o Recurso C também está ocupado quando o paciente é liberado da etapa 2. Porém, nesse caso, não é possível atrasar a alocação das etapas 1 e 2, como ocorre na situação (b), pois o Recurso A não tem disponibilidade. Dessa forma, é necessário inserir um período de ociosidade na etapa intermediária do paciente para

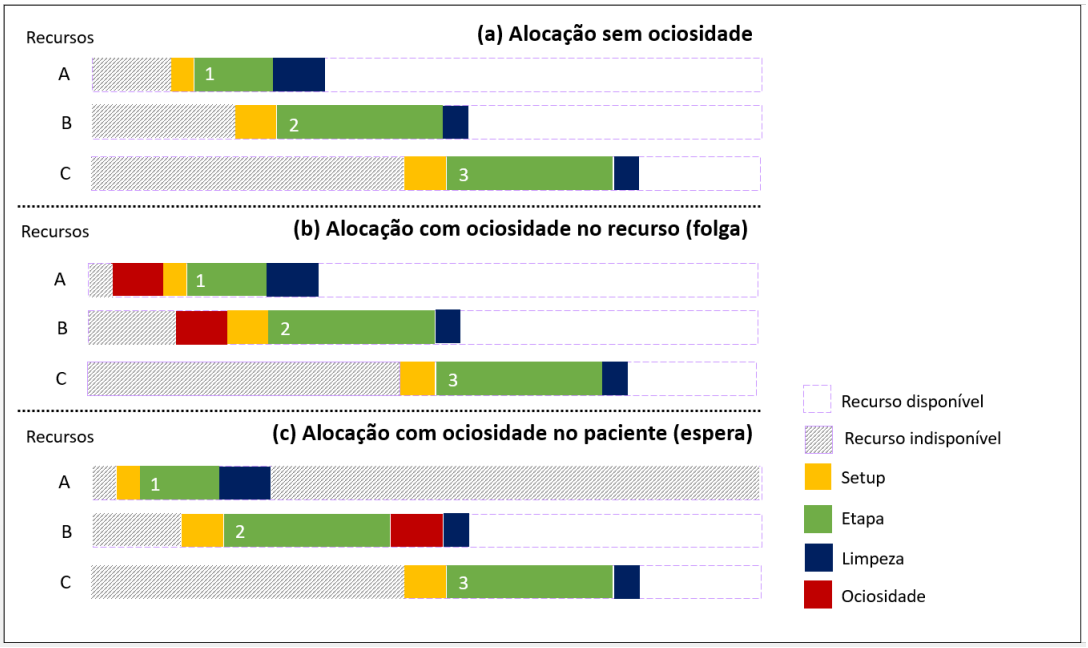

Figura 14: Representação das opções de ociosidade encontrada no sistema. Fonte: Elaborado pela autora.

garantir que as restrições temporais sejam respeitadas. Durante a *espera* do paciente, o Recurso B fica bloqueado (*blocking*) e não pode processar outros pacientes. Observe que essa situação só ocorre em cenários de alocação da última etapa e que o *blocking* só pode ser adicionado à etapa intermediária.

Durante a alocação dos recursos, a heurística construtiva deve identificar qual situação apresentada anteriormente deve ser utilizada para garantir que haja compatibilidade entre os recursos alocados e o recurso em teste. A seguir apresenta-se uma descrição detalhada da linha 31 da Heurística Construtiva para explicar como os movimentos de ajuste estão organizados no código.

O primeiro teste realizado (decisão: 3-14) corresponde a linha 31 da heurística construtiva que identifica que é necessário realizar um movimento de ajuste para garantir a programação do paciente, uma vez que nenhum recurso do conjunto pode ser alocado sem esses movimentos.

No segundo teste (decisão: 4-13) é identificado que tipo de movimento de ajuste que deve ser realizado. Caso seja possível antecipar todos os recursos alocados, será utilizada a situação (b) da Figura 14 para realizar o movimento de ajuste dos recursos já alocados, adicionando folga. Para o recurso em teste, é utilizada a alocação sem ociosidade, situação (a) da Figura 14.

Caso não seja possível antecipar todos os recursos alocados, um novo teste é realizado

Algoritmo 3 Movimentos de ajuste

1: **entrada:** recurso em teste que não é compatível com os recursos já alocados em uma etapa de um paciente
2: **saída:** escolha do tipo de alocação
3: **se** necessário realizar movimento de ajuste **então**
4: **se** é possível antecipar os recursos já alocados **então**
5: Adicionar folga aos recursos já alocados ▷ Alocação com folga
6: Alocar recurso em teste sem ociosidade ▷ Alocação sem ociosidade
7: **senão**
8: **se** etapa em teste é a terceira e período de *blocking* é menor que tempo máximo permitido para *blocking* **então**
9: Adicionar *blocking* no recurso em teste ▷ Alocação com espera
10: **senão**
11: Descartar o recurso em teste
12: **fim se**
13: **fim se**
14: **fim se**

(decisão: 8-12) para garantir que esse cenário só seja aplicado para alocação de recursos na terceira etapa e quando o tempo de *blocking* é menor ou igual que o tempo máximo permitido de espera. Caso não atenda esses dois requisitos, o recurso é descartado. Por outro lado, se o teste for satisfeito, os recursos utilizados na etapa 2 são movimentados conforme situação (c) da Figura 14 adicionando um período de *blocking*. Para o recurso em teste, a alocação sem ociosidade é utilizada (situação (a) da Figura 14). Os recursos da etapa 1 seguem sem modificação.

5.1.2 Exemplo

Na sequência, é apresentado um exemplo para ilustrar o funcionamento da heurística construtiva. Nas Figuras 15, 16 e 17 são apresentadas representações em forma de gráfico de *Gantt* da programação de dois pacientes utilizando a heurística construtiva proposta. Nesse exemplo, cada paciente tem três etapas (pré-cirurgia, cirurgia e pós-cirurgia), onde a primeira etapa só precisa de um recurso (enfermeiro), a segunda dois recursos(cirurgião vascular ou cardíaco e anestesista) e a terceira um recurso (sala de recuperação).

Os dois pacientes são ordenados conforme linha 3 do algorítmo da Heurística Construtiva {Paciente 1, Paciente 2}. O Paciente 1 é escolhido para dar início a programação (linha: 5). As três etapas do Paciente 1 são ordenadas e a etapa 1 é escolhida (linha: 6). Como só existe um tipo de recurso necessário para a etapa 1 do Paciente 1, o tipo de recurso *Enfermeiro* que é tipo de recurso chave e é selecionado (linha: 9). No conjunto de recursos do tipo de recurso Enfermeiro só existe o Recurso 1 que é escolhido para

processar a etapa 1 do Paciente 1. Na Figura 15, o Recurso 1 é alocado à etapa 1 do Paciente 1 como recurso chave (linha: 15).

Após a alocação de todos os tipos de recursos necessários para essa etapa, a próxima etapa é selecionada (linha: 47). Dois tipos de recurso são necessários para a etapa 2 e é necessário alocar primeiro o recurso chave (cirurgião vascular), portanto a sequência de alocação fica definida como {Cirugião vascular, Anestesista} (linha: 7). O conjunto de recursos do tipo Cirurgião vascular possui o Recurso 2 que é selecionado (linha:12) e alocado à etapa 2 do Paciente 1 (linha: 15). Atualizando a lista de tipos de recurso (linha: 45), o tipo de recurso Anestesista é selecionado. Três recursos estão disponíveis e considerando a *regra de ordenação de recursos* são ordenados como {Recurso 3, Recurso 4, Recurso 5}. Testando os três recursos (decisão: 21-26), verifica-se que não é possível alocá-los no mesmo instante do recurso chave. A Figura 15 apresenta a infactibilidade de tentar alocar o Recurso 3.

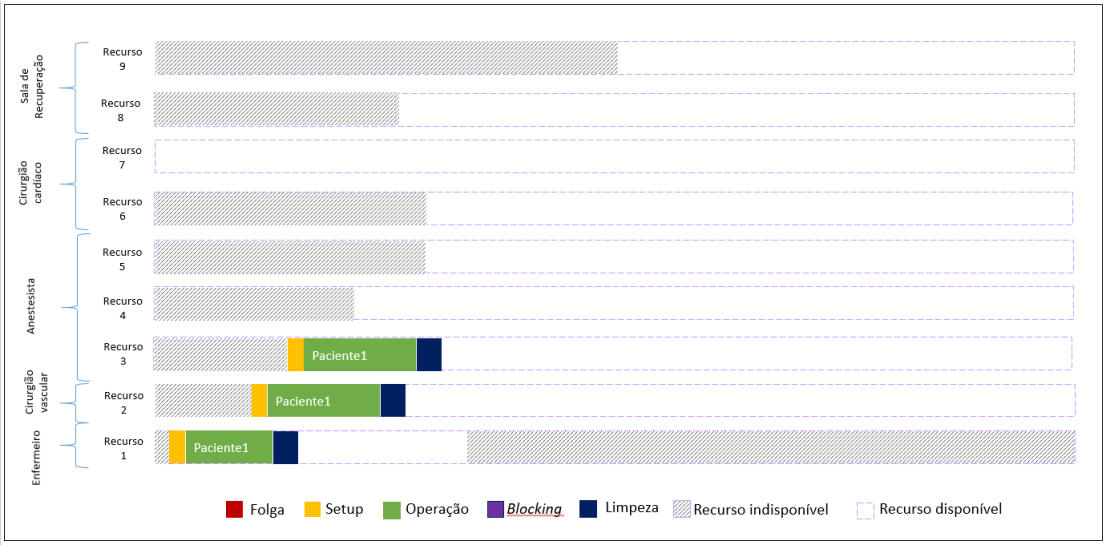

Figura 15: Exemplo para ilustrar o funcionamento da heurística construtiva - parte 1. Fonte: Elaborado pela autora.

Utilizando o movimento de ajuste *Alocação com ociosidade no recurso*, aos Recursos 1 e 2 foram adicionados períodos de ociosidades de forma que o Recurso 3 pudesse ser alocado. A Figura 16 apresenta esse movimento. Finalizando o processo de alocação de todos os tipos de recursos necessários na etapa 2, passa-se à etapa 3. Da mesma forma que a etapa 1, só um tipo de recurso é necessário e só um recurso está disponível para atender essa etapa. Dessa forma, o Recurso 9 é alocado como recurso chave (linha: 15). Finalizadas todas as etapas, a lista de pacientes é atualizada (linha: 49).

O Paciente 2 inicia a alocação da etapa 1 com o Recurso 1, único recurso disponível do

tipo Enfermeiro (linha: 15). Na alocação da etapa 2, os tipos de recursos são priorizados conforme regra e a sequência de alocação fica {Cirurgião cardíaco, Anestesista}. Dois recursos do tipo Cirurgião cardíaco estão disponíveis, uma regra de ordenação de recursos é utilizada e a sequência de teste que fica {Recurso 6, Recurso 7}. O Recurso 6 é alocado à etapa 2 do Paciente 2 como recurso chave (linha: 15). Os recursos do tipo Anestesista são organizados e a lista de ordenação definida é {Recurso 5 , Recurso 3 e Recurso 4}. O Recurso 5 é alocado no mesmo instante do Recurso 6 chave (linha: 22).

A última etapa do Paciente 2, precisa só do tipo de recurso Sala de Recuperação (SR) que possui dois recursos disponíveis. Os dois recursos são testados e é verificado que nenhum dos dois podem ser alocados respeitando as restrições dos recursos já alocados, portanto é necessário verificar se é possível realizar um movimento de ajuste. A Figura 16 apresenta a infactibilidade da alocação do Recurso 9.

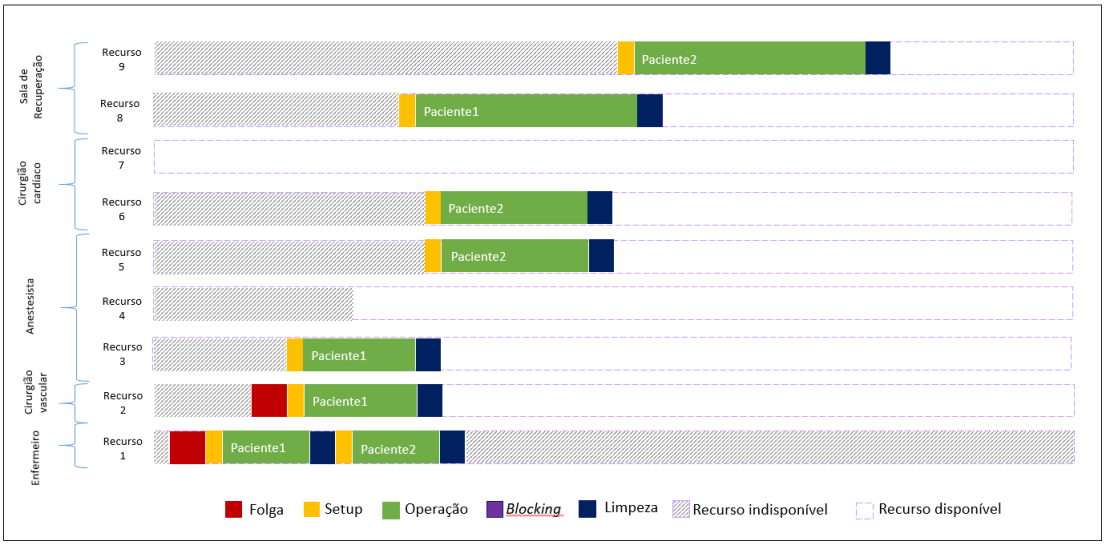

Figura 16: Exemplo para ilustrar o funcionamento da heurística construtiva - parte 2. Fonte: Elaborado pela autora

Com o movimento de ajuste *Alocação com ociosidade no paciente*, o Recurso 9 é alocado ao Paciente 2. Na Figura 17 é apresentado a situação final após o movimento. Como não é possível movimentar a alocação no Recurso 1, um período de *blocking* é adicionado aos recursos da etapa 2 (Recursos 6 e 7) garantindo que o início e fim da etapa 2 possam coincidir com o fim da etapas 1 e início da etapa 3, respectivamente.

A heurística construtiva proposta encontrou uma das soluções ótimas para esse exemplo com menor *makespan* e atendendo todos os pacientes disponíveis. Do exemplo, é possível compreender a importância dos movimentos de ajuste no algoritmo para garantir um maior número de alocações. Também pode-se observar como a hierarquia de alocação garante que todas as etapas e todos os recursos necessários sejam programados. A seção

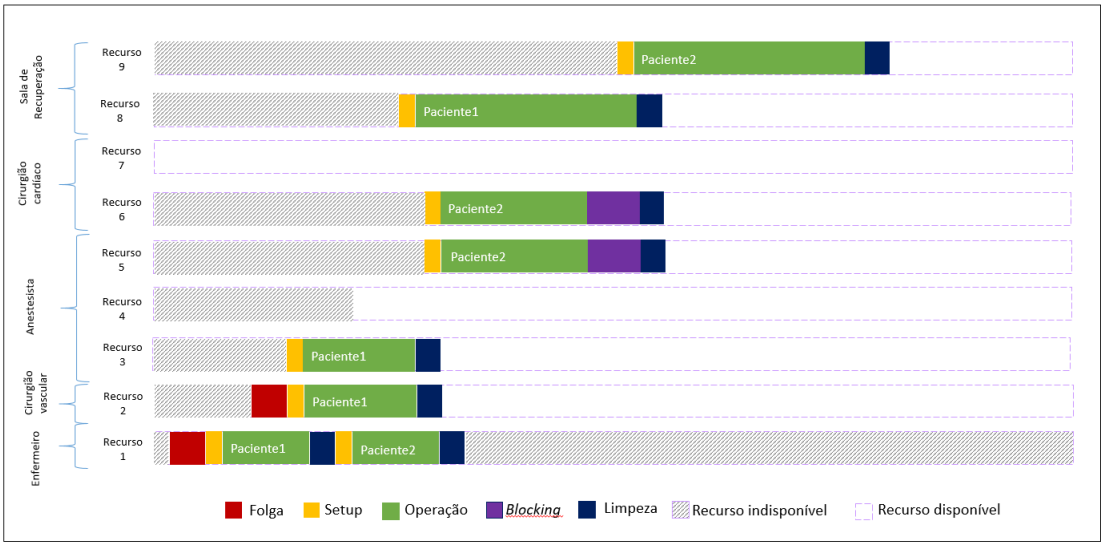

Figura 17: Exemplo para ilustrar o funcionamento da heurística construtiva - parte 3. Fonte: Elaborado pela autora.

a seguir apresenta as regras de ordenação que podem ser utilizadas para o algoritmo proposto.

5.2 Regras de ordenação

Nessa seção são apresentadas as regras de ordenação utilizadas na Heurística construtiva. As regras de ordenação são utilizadas em três níveis: *pacientes* (linha: 3), *tipos de recurso* (linha:7) e *recursos* (linha: 10). A seguir são apresentadas e discutidas as regras utilizadas em cada um dos níveis.

5.2.1 Regras de ordenação de pacientes

A escolha da ordem de alocação dos pacientes está no nível mais superior da hierarquia, portanto, é uma escolha que tem grande influência nos resultados obtidos pela heurística construtiva. Alguns critérios de ordenação foram utilizados para priorizar os pacientes: *prioridade*, *flexibilidade* e *tempo médio de processamento*, além de combinações dessas critérios.

No problema estudado, os pacientes possuem *prioridades* ($Prioridade_p$) de atendimento conforme a gravidade da doença. A organização da lista de pacientes considerando esse critério é adequada uma vez que é interessante favorecer que determinados pacientes graves sejam programados dentro do horizonte de tempo do problema. Dessa forma, essa regra prioriza os pacientes com maior *prioridade* em ordem não-crescente.

Na heurística proposta utiliza-se o conceito de *Grau de flexibilidade entre operações e máquinas* apresentado em Melo e Ronconi (2015) que explora as características do ambiente *Job Shop Flexível*. O *grau de flexibilidade* tem origem na ideia intuitiva de alocar primeiro as operações j com menos opções de máquinas alternativas ($|M_j|$). Matematicamente, o *grau de flexibilidade* apresentado em Melo e Ronconi (2015) de cada operação j pode ser definido por:

$$Flexibilidade_j = \frac{|M_j|}{m} \tag{5.1}$$

onde m é o número de máquinas no sistema, $|M_j|$ representa a quantidade de máquinas alternativas que pode processar a operação j e $0 \leq Flexibilidade_j \leq 1$.

Na equação (5.2) o conceito é adequado para o problema estudado, onde $|R|$ indica a quantidade de todos os recursos disponíveis no sistema e $|R_p|$ indica a quantidade de recursos disponíveis para processar o paciente p. Portanto, ao ser utilizada essa regra, a lista de pacientes é ordenada por ordem não-decrescente.

$$Flexibilidade.paciente_p = \frac{|R_p|}{|R|} \tag{5.2}$$

O outro critério utilizado é a soma dos *tempo de processamento médio* em todas as etapas de um paciente. Uma vez que mais de um recurso pode processar um paciente em uma etapa, o cálculo dessa regra considera a média por etapa e é apresentado na Equação (5.3).

$$\text{Tempo.processamento.médio}_p = \sum_{s \in S_p} \left(\frac{\sum_{r \in R_{ps}} Tempo.processamento_{prs}}{|R_{ps}|} \right) \tag{5.3}$$

onde S_p indica o conjunto de etapas do paciente p, R_{ps} define o conjunto de recursos que podem processar a etapa s do paciente p.

Duas ordenações são possíveis considerando o critério de *tempo de processamento médio*. A primeira prioriza pacientes com menor *tempo de processamento médio*, do inglês *Shortest Processing Time* (SPT), em outras palavras segue ordem não-decrescente considerando esse critério. Essa regra é ótima em um sistema de máquinas em paralelo cuja função objetivo é minimizar o somatório dos instantes de finalização das tarefas (PINEDO, 2012). Como o problema estudado está inserido em um ambiente de máquinas flexível, ou seja, com máquinas em paralelo em cada estágio, e busca minimizar o *makespan*, que é o

Tabela 27: Exemplo - combinação regras prioridade e flexibilidade.

Paciente	$Flexibilidade.paciente_p$	$Prioridade_p$	Combinação
A	30	5	7,5
B	30	4	6,6
C	40	2	5,8
D	35	2	5,3

instante de fim da última tarefa a deixar o sistema, incluiu-se o *SPT* na lista de regras a serem testadas.

Também foi implementada a regra inversa ao *SPT*, chamada *MRW* (do inglês, *Most Remaining Work*) que apresenta bons resultados na minimização do *makespan* em ambiente de máquina *Flow Shop Flexível* (RUIZ; SERIFOGLU; URLINGS, 2014). Essa regra prefere ordenar primeiro os pacientes com maior *tempo de processamento médio* em ordem não-crescente.

Foi elaborada uma regra que combina a *prioridade*, que é uma particularidade desse problema, e o *grau de flexibilidade* que considera as características do ambiente de máquina *Job Shop Flexível*. A equação (5.4) apresenta a combinação das regras, onde $0 \leq \alpha \leq 1$:

$$Regra.\text{combinação} = Flexibilidade.paciente_p(1-\alpha) - Prioridade_p(\alpha) \tag{5.4}$$

A parcela que contém *prioridade* está negativa, uma vez que as listas de *grau de flexibilidade* e *prioridade* têm ordenação oposta. Em outras palavras, a lista *prioridade* tem ordenação não-crescente enquanto a lista utilizada no *grau de flexibilidade* possui ordenação não-decrescente. Para elucidar essa especificidade será apresentado um exemplo a seguir, em que $\alpha = 0,9$.

Na Tabela 27 os dois primeiros pacientes (A e B) possuem o mesmo *grau de flexibilidade* portanto, a ordem entre eles deve ocorrer de forma não-crescente por *prioridade*. Já os pacientes C e D, ambos com mesma *prioridade*, devem ser ordenados entre eles de forma não-decrescente pelo *grau de flexibilidade*. A regra de combinação de *grau de flexibilidade* e *prioridade* com uma das parcelas negativas garante que a lista seja ordenada adequadamente quando utilizada a ordenação não-decrescente, situação que não ocorreria se ambas parcelas fossem positivas. No exemplo, como o valor do α é alto, a regra pondera dando mais peso à *prioridade*, portanto a ordem de alocação de pacientes será {D,C,B,A}.

Uma combinação entre as regras *prioridade* e soma dos *tempos de processamento médio* tem inspiração no problema da mochila. Neste problema deve-se guardar em uma mochila objetos $i \in I$ que possuem diferentes pesos (p_i) e valores (v_i). A função objetivo é maximizar o valor respeitando a única restrição de peso máximo permitido da mochila. A ordenação não-decrescente dos itens utilizando o critério valor por unidade de peso (v_i/p_i) retorna a solução ótima da relaxação linear do problema (ARENALES *et al.*, 2011).

Fazendo uma analogia ao problema em estudo, o horizonte de tempo disponível para a alocação dos pacientes seria a restrição de peso máximo da mochila, enquanto a *prioridade* e a soma dos *tempos de processamento médios* seriam o valor e o peso de um paciente, respectivamente. Dessa forma, foi desenvolvida uma regra com ordenação não-crescente considerando a Equação (5.5).

$$Regra.mochila.prioridade_p = \frac{Prioridade_p}{\text{Tempo.processamento.médio}_p} \tag{5.5}$$

Seguindo a mesma analogia e introduzindo as características do ambiente de máquina *Job Shop Flexível*, considera-se que os pacientes com menor capacidade de ser alocados devido a menor quantidade de recursos disponíveis têm maior importância na alocação. Em outras palavras, o *grau de flexibilidade* é analogamente equivalente ao valor (v_i) de um objeto no problema da mochila. Assim, na Equação (5.6) é apresentada essa regra que deve ter ordem não-crescente.

$$Regra.mochila.flexibilidade_p = \frac{Flexibilidade.paciente_p}{\text{Tempo.processamento.médio}_p} \tag{5.6}$$

Todas as regras mencionadas até agora também podem ser utilizadas como regra de desempate quando existe empate entre pacientes na regra principal. Por exemplo, se existem dois pacientes com *prioridades* iguais, pode-se utilizar a *flexibilidade* como regra desempate. Nesse trabalho também foi incluído como regra de desempate o número de identificação do paciente (id) em ordenação não-decrescente.

5.2.2 Regras de ordenação de tipos de recursos

A flexibilidade também é incorporada na escolha de qual *tipo de recurso* deve ser alocado primeiro. Por exemplo, se uma etapa precisa de dois tipos de recursos (enfermeiro e cirurgião) e existem três recursos do tipo enfermeiro enquanto somente um recurso do tipo cirurgião, primeiro escolhe-se o cirurgião para, posteriormente, buscar um enfermeiro

que possa ser alocado no mesmo período. Na Equação (5.7) é apresentada essa regra.

$$Flexibilidade.tipo.de.recurso_{pt} = \frac{|R^t|}{|R|} \tag{5.7}$$

onde R^t é o conjunto de recursos disponíveis do tipo t e R indica o conjunto de recursos disponíveis no sistema.

5.2.3 Regras de ordenação de recursos

A regra de ordenação utilizada para os *recursos* busca alocar o recurso que estiver mais próximo no tempo dos recursos já alocados. Ou seja, busca-se reduzir a distância ou eliminar os movimentos de ajuste. Considere o exemplo apresentado na Seção 5.1.2, quando é necessário fazer a escolha do tipo de recurso Anestesista para o Paciente 2 (Figura 16). Como já existe um recurso chave alocado a etapa (Recurso 6), compara-se os recursos disponíveis com recurso chave, dessa forma, seguindo essa regra de ordenação por menor distância de movimento de ajuste, tem-se a ordem {Recurso 5, Recurso 3, Recurso 4}. Na Equação (5.8) apresenta-se o cálculo realizado para definir a menor distância.

$$Distância_{psr} = |paciente.disponível_p - recurso.disponível_r - setup_{ps}| \tag{5.8}$$

onde $paciente.disponivel_p$ e $recurso.disponivel_r$ indica, respectivamente, o instante que o paciente p e o recurso r estão disponíveis no sistema e o $setup_{ps}$ define o valor do *setup* da etapa e do paciente que estão sendo alocados. Observe que o valor da distância é definido em módulo.

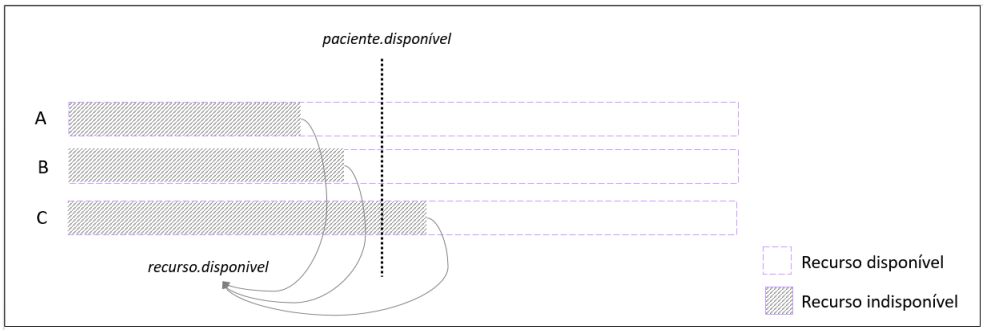

Figura 18: Representação dos instantes de disponibilidade do recurso e do paciente. Fonte: Elaborado pela autora.

Na Figura 18 é apresentado um esquema com três recursos (A,B e C) com instantes de disponibilidade diferentes, também é indicado o instante em que o paciente está disponível

Tabela 28: Exemplo - regra de ordenação de recursos por distância não-decrescente

Recurso	recurso.disponível	paciente.disponivel	*setup*	Distância
A	650	700	20	30
B	680	700	20	0
C	730	700	20	50

após ser processado na etapa anterior.

Na Tabela 28 são apresentados os dados necessários para calcular o valor da distância e a Figura 19 apresenta o gráfico de *Gantt* de como ficaria se os recursos fossem escolhidos. No exemplo, como a ordenação é feita de forma não-decrescente a ordem dos recursos a serem testados é {B,A,C}.

Para o recurso A a distância é de 30, ou seja, o recurso deve ter uma ociosidade de 30 minutos até que se inicie as etapas do paciente. Esse caso está associado à situação (b) da Figura 14. O segundo recurso (B) tem distância 0, ou seja, não existe ociosidade no recurso ou no paciente e é equivalente a situação (a) da Figura 14.

O recurso C tem em módulo distância de 50 minutos, nesse caso, o paciente fica ocioso até o recurso esteja disponível. Se esse recurso for escolhido, é necessário fazer um movimento de ajuste nas etapas já alocadas do paciente para atender as restrições do problema. A situação (c) da Figura 14 apresenta esse caso.

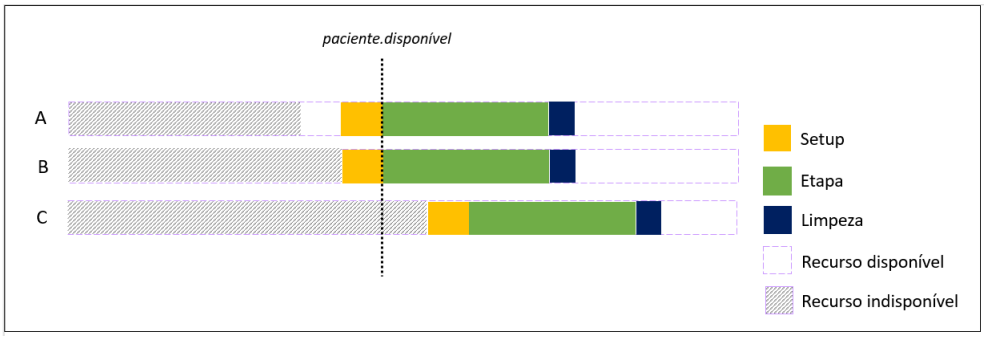

Figura 19: Exemplo - regra de ordenação de recursos por distância. Fonte: Elaborado pela autora.

Como visto anteriormente, o *tempo de processamento*(TP) de uma etapa pode modificar dependendo do recurso escolhido. Portanto, introduzir esse conceito na ordenação dos recursos pode ser uma alternativa interessante. Para isso, uma nova regra foi desenvolvida para ordenar recursos.

Nessa regra, a ordenação é realizada em pares. Quando o valor da distância é menor que 10% do TP médio dos dois recursos, o TP é considerado como regra. No Algoritmo 4 é apresentado como ocorre a decisão de qual regra será utilizada.

Algoritmo 4 Regra de ordenação de recursos por distância considerando o tempo de processamento.

1: **entrada:** $recurso_1, recurso_2$
2: **saída:** regra de ordenação dos recursos
3: $distancia_{ps1} = paciente.disponível_p - recurso.disponível_1 - setup_{ps}$
4: $distancia_{ps2} = paciente.disponível_p - recurso.disponível_2 - setup_{ps}$
5: $tempo.processamento.médio = (tempo.processamento_1 + tempo.processamento_2)/2$
6: **se** $|distância_1 - distância_2| \geq tempo.processamento.médio(10\%)$ **então**
7: Ordenar por TP
8: **senão**
9: Ordenar por distância
10: **fim se**

Tabela 29: Exemplo - regra de ordenação de recursos por distância considerando o tempo de processamento.

Recurso	Tempo.processamento	Distância
A	250	30
B	310	0
C	400	50

Para ilustrar, tome o exemplo anterior da Tabela 28 onde o TP de cada um dos recursos é apresentado na Tabela 29. Comparando os recursos A e B temos que o 10% do tempo médio de processamento é menor que a distância desses dois recursos. Nesse caso, o TP é considerado e o recurso A é ordenado frente à B. Analisando o recurso A e C temos que 10% do tempo de processamento médio é maior que a diferença entre as distâncias, portanto o TP não é utilizado para a ordenação. Ficando o ordenado o recurso A antes do C. Por último, é realizada uma análise entre os recursos B e C, mais uma vez o tempo médio de processamento é menor que a distância entre eles, prevalecendo a ordenação pela distância. Dessa forma a ordem final fica {A,B,C}, diferente da ordenação considerando somente a distância.

5.3 Experimentos numéricos

Nessa seção são apresentados os experimentos numéricos realizados a fim de analisar o desempenho de cada uma das regras mencionadas na seção anterior. Considerando que existem três níveis de ordenação, o grupo de regras escolhido em cada teste é denominado Conjunto de Regras (CR) e contempla as regras de ordenação para pacientes, com critério desempate, tipo de recursos e recursos.

Foram testados os 7 grupos de instâncias apresentados na Seção 4.2.1, no qual cada um dos grupos são geradas instâncias com 5, 7, 10, 12 e 15 pacientes para 7 sementes de

Tabela 30: Conjunto de regras de ordenação.

Conjunto de regras	Regra ordenação pacientes	Regra ordenação tipo recurso	Regra ordenação recurso
0	Flexibilidade (ID)	Flexibilidade	Distância
1	Flexibilidade (ID)	Flexibilidade	Distância com TP
2	Flexibilidade (SPT)	Flexibilidade	Distância
3	Flexibilidade (SPT)	Flexibilidade	Distância com TP
4	Flexibilidade (MRW)	Flexibilidade	Distância
5	Flexibilidade (MRW)	Flexibilidade	Distância com TP
6	Flexibilidade (Prioridade)	Flexibilidade	Distância
7	Flexibilidade (Prioridade)	Flexibilidade	Distância com TP
8	Prioridade (ID)	Flexibilidade	Distância
9	Prioridade (ID)	Flexibilidade	Distância com TP
10	Prioridade (Flexibilidade)	Flexibilidade	Distância
11	Prioridade (Flexibilidade)	Flexibilidade	Distância com TP
12	Combinação (ID)	Flexibilidade	Distância com TP
13	Combinação (SPT)	Flexibilidade	Distância com TP
14	Combinação (MRW)	Flexibilidade	Distância com TP
15	SPT (ID)	Flexibilidade	Distância com TP
16	SPT (Prioridade)	Flexibilidade	Distância com TP
17	SPT (Flexibilidade)	Flexibilidade	Distância com TP
18	MRW (ID)	Flexibilidade	Distância com TP
19	MRW (Prioridade)	Flexibilidade	Distância com TP
20	MRW (Flexibilidade)	Flexibilidade	Distância com TP
21	Mochila prioridade (ID)	Flexibilidade	Distância com TP
22	Mochila prioridade (SPT)	Flexibilidade	Distância com TP
23	Mochila prioridade (MRW)	Flexibilidade	Distância com TP
24	Mochila prioridade (Flexibilidade)	Flexibilidade	Distância com TP
25	Mochila flexibilidade (ID)	Flexibilidade	Distância com TP
26	Mochila flexibilidade (SPT)	Flexibilidade	Distância com TP
27	Mochila flexibilidade (MRW)	Flexibilidade	Distância com TP
28	Mochila flexibilidade (Prioridade)	Flexibilidade	Distância com TP

Taillard (1993) diferentes. No total foram testadas 245 instâncias para cada conjunto de regras.

Na Tabela 30 é apresentado os CR testados com as regras para cada nível. No nível do paciente, além da regra principal, também considera-se uma regra desempate que está apresentada entre parêntesis na segunda coluna. Os CR12, 13 e 14 utilizam a regra *Combinação entre prioridade e flexibilidade* com o valor de α variando de 0,1 até 0,9 Equação (5.4).

5.3.1 Comparações entre conjuntos de regras

Para comparar a efetividade de cada conjunto de regras utilizou-se o CR0 como base. Em outras palavras, compara-se o desempenho do conjunto de regras em teste para cada instância com o CR base a fim de analisar sua performance. O desempenho do conjunto de regras é considerado melhor se os resultados obtidos atendem mais pacientes ou se atendem o mesmo número de pacientes com o valor do *makespan* menor em comparação com o CR0. Para ser considerado empate, os números de pacientes atendidos e o valor do *makespan* devem ser idênticos.

Na Tabela 31 é apresentada os resultados obtidos em cada um dos conjuntos de regras e na Figura 20 são apresentados esses resultados em forma de árvore. Na Tabela 31 a segunda e terceira colunas representam o número de instâncias em que o conjunto de regras obteve resultados melhores e empataram com que o CR0, respectivamente. A última coluna apresenta os resultados da análise que correspondem ao percentual de melhoria ou empate em relação ao total de 245 instâncias testadas.

Na Figura 20 é apresentada uma representação gráfica em forma de árvore dos conjuntos de regras testados. Cada coluna corresponde a um dos níveis de ordenação (paciente, tipo de recurso e recurso). Além disso, foi adicionado o nível de desempate do paciente, indicado na segunda coluna. Na última coluna está a identificação do CR e o valor em percentual da melhoria e empate desse CR em relação ao conjunto base.

Os Conjuntos de Regras que utilizam a regra *Combinação* entre prioridade e flexibilidade na ordenação de pacientes (CR12, 13 e 14) têm resultados que variam em relação ao valor de α e a regra desempate utilizada. Para analisar esse cenário, a Figura 21 apresenta um gráfico variando o valor de α para cada um dos critérios desempate.

Observe que nos extremos, onde $\alpha = 0$ e $\alpha = 1$, conforme Equação (5.4), o valor de alguma das parcelas (flexibilidade ou prioridade) é zero. Dessa forma, quando $\alpha = 0$ somente a parcela da *flexibilidade* se mantém e seus resultados são correspondentes aos CR1 (ID), CR3 (SPT) e CR5 (MRW). Caso o valor de $\alpha = 1$, somente a parcela da prioridade se mantém e os testes são equivalentes aos testes realizados no CR9 para o critério desempate ID.

A utilização do *Grau de flexibilidade* como regra de ordenação de pacientes apresenta resultados superiores em relação ao CR0 variando de 68% a 82% de melhoria. Esse resultado era esperado, uma vez que essa regra favorece pacientes com menores chances de alocação, devido ao baixo número de recursos disponíveis para ele, utilizando, assim,

Tabela 31: Comparação dos conjuntos de regras com base.

Conjunto de regras	# melhor que CR0	# empata com CR0	Percentual melhor ou empate
1	168	34	82%
2	81	97	73%
3	161	18	73%
4	51	116	68%
5	155	12	68%
6	91	89	73%
7	161	12	71%
8	64	18	33%
9	68	5	30%
10	69	14	34%
11	69	6	31%
12	167	14	74-44%
13	156	14	69-41%
14	168	13	74-39%
15	60	0	24%
16	60	0	24%
17	60	0	24%
18	102	7	44%
19	102	7	44%
20	103	7	45%
21	132	3	55%
22	130	4	55%
23	128	3	53%
24	128	4	54%
25	160	21	74%
26	161	18	73%
27	157	16	71%
28	163	17	73%

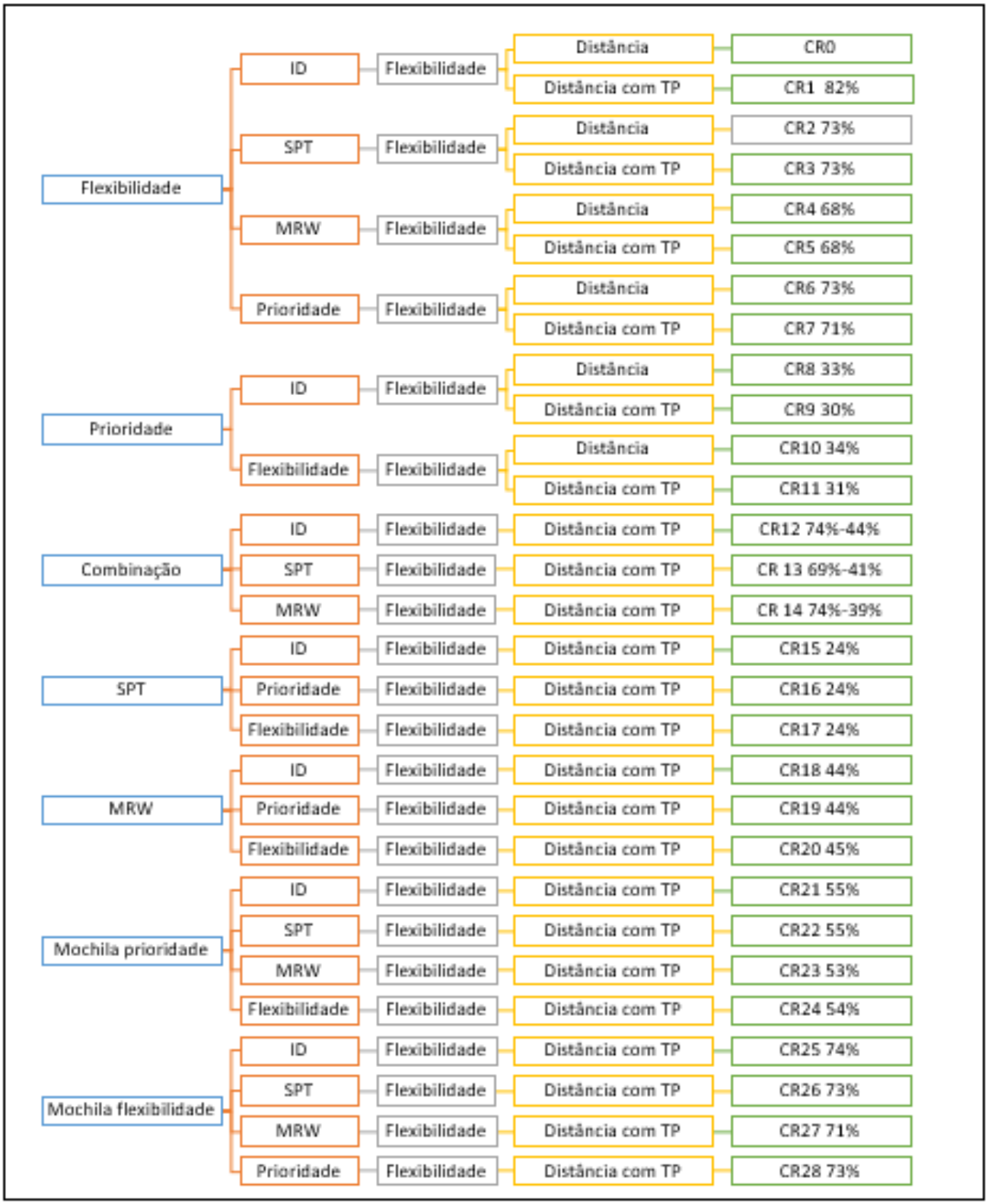

Figura 20: Representação em árvore dos conjuntos de regras com comparação com CR base. Fonte: Elaborado pela autora.

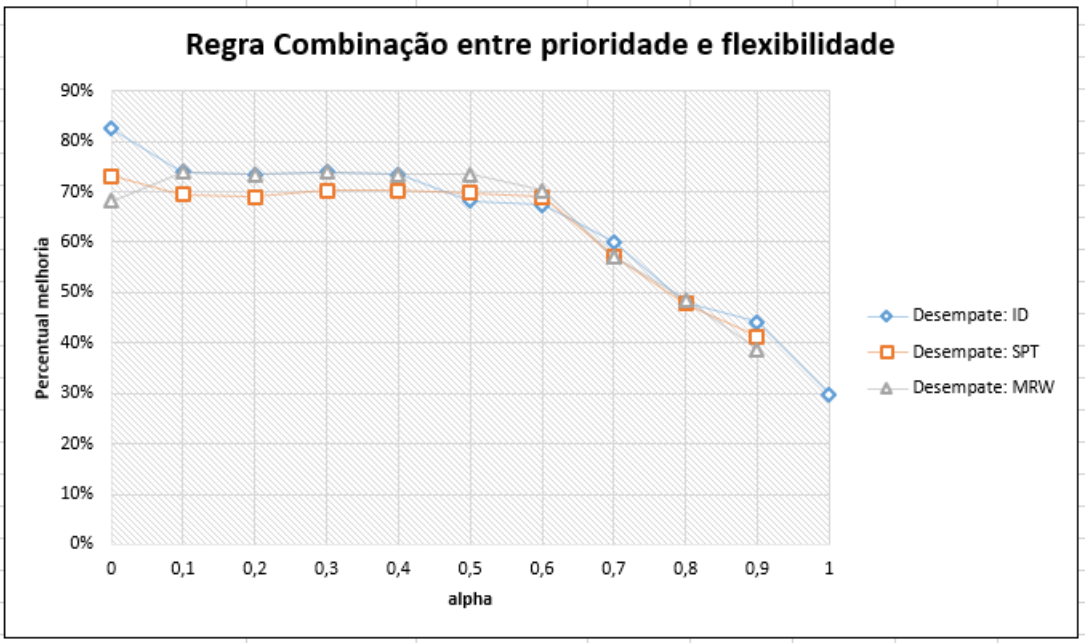

Figura 21: Gráfico das regras que utilizam o critério *Combinação* entre prioridade e flexibilidade. Fonte: Elaborado pela autora.

melhor o sistema como todo.

Por outro lado, analisando os percentuais de melhoria em relação ao CR0, é possível perceber que utilizar a *Prioridade* como regra de ordenação de pacientes não apresenta bons resultados (CR8-11) com valores variando entre 30% e 34%. Esse resultado era esperado, uma vez que a *prioridade* não tem qualquer relação com a disponibilidade de recursos no sistema. Por outro lado, levar em conta esse aspecto do problema é interessante para garantir que os pacientes, cuja urgência no atendimento é maior sejam colocados no conjunto de pacientes atendidos.

Dessa forma, foi desenvolvida uma regra que tenta unir os bons resultados encontrados com o critério *Grau de flexibilidade* com a *Prioridade*. No Gráfico 21 é apresentado os resultados em relação ao valor do α. Quanto mais a esquerda maior a influência do critério *Grau de flexibilidade* e quanto mais a direito, maior a influência da *Prioridade*. Conforme esperado, quanto maior o valor do α pior é o valor do percentual de melhoria em relação ao CR0.

As regras *SPT* e *MRW* não apresentam bons resultados quando utilizadas como regra principal de ordenação de paciente, obtendo melhoria só de 24% e 45%, respectivamente. Três critérios desempate foram utilizados (*ID*, *Prioridade* e *Grau de flexrbilidade*), porém, para ambas regras, o desempate não obteve grande influência nos resultados.

Analisando as regras que foram inspiradas no problema da mochila observa-se que os

testes utilizando a *Prioridade* não foram tão vantajosos ficando com 55% de melhoria em relação ao CR0, no máximo. Por outro lado, quando o *Grau de flexibilidade* é utilizado analogamente como o valor dos objetos a serem colocados na mochila os percentuais de melhoria chegam a 74% em relação ao CR0.

As regras de desempate dos pacientes utilizadas que obtiveram os melhores resultados foram *ID* e *SPT*, com o *ID* superando ligeiramente a regra *SPT*. O único cenário que a regra *MRW* supera a regra *SPT* é quando a regra de pacientes é *Combinação*. Utilizar a *Prioridade* como critério desempate para o *Grau de flexibilidade* (CR6-7) também apresentou bons resultados, 71% e 73%.

Já a regra de ordenação de recurso que obteve melhores resultados foi a *Distância com Tempo de Processamento*, resultado já esperado uma vez que utilizar somente a *Distância* poderia gerar cenários com escolha de recursos com tempo de processamento grandes e piorar o resultado da heurística. Esse cenário só se inverte quando utilizada a *Prioridade* como regra principal de escolha de pacientes, nesse caso, a regra sem considerar o Tempo de processamento do recurso foi melhor. As regras CR6 e CR7 também apresentam resultados contrários ao esperado, com 71% e 73% de melhora respectivamente. Porém, quando se analisa esses CR na Tabela 31, é possível perceber que no CR7 o número de instâncias que foram melhores que o CR0 é bem superior do que esse mesmo número para o CR6. Em outras palavras, apesar do CR6 ganhar no somatório da quantidade de instâncias melhores e empates, o CR7 apresenta um número superior de instâncias melhores. O mesmo ocorre com os pares CR2 e CR3, e CR4 e CR5.

5.3.2 Comparação com o resultado ótimo

Dentre todas as regras apresentadas o conjunto de regras 1 foi o que obteve melhores resultados. Nesse conjunto de regras, 82% das instâncias foram melhores ou empataram com os resultados do conjunto de regras base. Nessa seção é realizada uma comparação desse conjunto de regras com os resultados obtidos pelo modelo matemático apresentado no Capítulo 5.

As Tabelas 32, 33 34, 35, 36, 37 e 38 apresentam os resultados obtidos em cada uma das 245 instâncias testadas. O tempo médio para processar todas as instâncias em um computador com processador Intel(R) Core(TM), com 4 núcleos, 2.80GHz, com memória RAM disponível de 16 GB, sistema operacional *Windows* foi de 3 segundos e utilizando linguagem C++ no IDE *CodeBlocks*.

Conforme esperado, o tempo de processamento das instâncias é bem menor que o tempo utilizado para encontrar a solução ótima do problema. Para chegar ao ótimo o tempo de processamento de instâncias pequenas (número de pacientes ≤ 7) chega a 45 segundos por instância, para instâncias com 10 pacientes esse valor fica em média 2min 34s por instância e para problemas maiores (número de pacientes ≥ 12) extrapolando as duas horas de limite, e, dessa forma, não garantindo o ótimo.

Analisando o número de pacientes atendidos, principal componente da função objetivo do modelo matemático, podemos verificar que a heurística alocou a mesma quantidade de pacientes que o modelo matemático proposto em 68 instâncias (27,7%). Para instâncias pequenas (número de pacientes ≤ 7) o número máximo de pacientes não alocados em relação ao resultado obtido pelo modelo foi 3.

Os piores resultados em relação número de pacientes não alocados em relação a solução do modelo matemático foram obtidos nos grupos 2.1 e 2.2, chegando a 9 pacientes em uma instância com 15 pacientes disponíveis. Os grupos 0, 1.1, 1.2, 2.1 e 2.2 deixaram de alocar no máximo 4 pacientes em relação à solução encontrada pelo modelo matemático para todos os tamanhos de instâncias.

Três instâncias empataram no número de pacientes e valor do *makespan* com o resultado obtido pelo modelo matemático (1.1_5_6, 1.1_7_5 e 1.2_5_6), ou seja, conseguiram encontrar o valor ótimo do problema. Ainda que esse cenário só tenha ocorrido em poucas instâncias é importante relembrar que a heurística construtiva tem objetivo de encontrar boas soluções sem garantia de otimalidade e, ainda sim, conseguiu encontrar o valor ótimo para algumas instâncias.

Além dessas instâncias ótimas, a heurística construtiva encontrou um resultado melhor para as instâncias 0_10_3, 1.2_5_2 e 3.1_10_3 que o modelo matemático. Observe que essas três instâncias atingem o GAP limite de 0,005% definido nos testes do modelo matemático proposto. Dessa forma, mostraremos que o valor encontrado pela heurística está dentro desse limite de tolerância. Analisando a função objetivo do modelo proposto temos:

$$min \quad mkp - G \sum_{i \in I} \sum_{r \in R_i} z_i^r w_i \tag{5.9}$$

Onde G possui ordem de grandeza entre 10^3 até 10^4 e o *makespan* está em minutos.

A diferença entre o *makespan* do modelo proposto e a heurística é de 37 min para a instância 0_10_3, 4 min para a instância 1.2_5_2 e 31 min para 3.1_10_3. Considerando a diferença de ordens de grandeza da parcela que considera o número de pacientes atendi-

dos e o *makespan*, pode-se afirmar o *GAP* limite (0,005%) é atingido antes que o modelo matemático possa encontrar a solução apresentada na heurística. Portanto pode-se afirmar a solução encontrada na heurística construtiva está dentro do limite de tolerância da solução encontrada no modelo matemático.

Conforme esperado, os grupos de instâncias com menos recursos disponíveis ou disponibilidade de recursos mais restrita (1.2, 2.2 e 3.2) apresentam piores resultados em relação à quantidade de pacientes atendidos e *makespan* que os grupos menos restritos ou com mais recursos (1.1, 2.1 e 3.1).

Tabela 32: Comparação com solução ótima - Grupo 0

	ID	Modelo matemático		GAP	Heurística construtiva	
		Makespan	# pacientes atendidos		*Makespan*	# pacientes atendidos
5 pacientes	0_5_1	2d 13h 54min	5	0,00*	2d 13h 55min	5
	0_5_2	1d 17h 02min	5	0,00*	2d 13h 23min	5
	0_5_3	2d 12h 42min	5	0,00	2d 13h 9min	5
	0_5_4	2d 13h 10min	5	0,00	2d 13h 57min	5
	0_5_5	2d 13h 08min	5	0,00*	2d 13h 16min	5
	0_5_6	2d 13h 22min	5	0,00	2d 14h 21min	5
	0_5_7	2d 13h 16min	5	0,00	2d 13h 26min	5
7 pacientes	0_7_1	3d 13h 24min	7	0,00*	4d 14h 53min	7
	0_7_2	2d 18h 09min	7	0,00	13d 14h 54min	7
	0_7_3	3d 13h 53min	7	0,00*	3d 15h 34min	7
	0_7_4	3d 13h 46min	7	0,00*	4d 14h 26min	7
	0_7_5	2d 16h 51min	7	0,00*	4d 12h 46min	7
	0_7_6	3d 13h 14min	7	0,00*	4d 13h 18min	7
	0_7_7	3d 14h 51min	7	0,00*	3d 16h 8min	7
10 pacientes	0_10_1	3d 13h 43min	8	0,00*	4d 14h 53min	8
	0_10_2	4d 14h 35min	10	0,00*	3d 14h 54min	8
	0_10_3	3d 15h 34min	8	0,00*	3d 14h 57min	8
	0_10_4	4d 15h 00min	9	0,00*	4d 18h 23min	9
	0_10_5	4d 14h 07min	10	0,00*	3d 16h 29min	8
	0_10_6	4d 18h 52min	10	0,00*	4d 13h 18min	8
	0_10_7	4d 12h 53min	9	0,00*	4d 13h 2min	8
12 pacientes	0_12_1	3d 14h 41min	9	0,00	4d 14h 53min	9
	0_12_2	4d 17h 16min	11	0,00*	4d 14h 27min	9
	0_12_3	4d 18h 16min	10	0,00*	4d 14h 26min	9
	0_12_4	4d 15h 12min	10	0,00*	3d 17h 38min	8
	0_12_5	4d 16h 30min	11	0,00*	4d 14h 7min	9
	0_12_6	4d 15h 51min	10	0,00*	4d 14h 22min	9
	0_12_7	4d 12h 53min	10	0,00*	4d 13h 3min	9
15 pacientes	0_15_1	4d 14h 35min	11	28,73	4d 12h 56min	8
	0_15_2	4d 17h 16min	12	4,67	4d 14h 50min	9
	0_15_3	4d 19h 41min	11	1,54	4d 18h 16min	10
	0_15_4	4d 15h 12min	11	20,27	4d 14h 43min	10
	0_15_5	4d 18h 15min	12	2,52	4d 14h 12min	10
	0_15_6	4d 17h 20min	12	0,00*	4d 14h 25min	9
	0_15_7	4d 15h 12min	12	0,02	4d 16h 2min	9

* indica que o *GAP* de tolerância foi atingido

Tabela 33: Comparação com solução ótima - Grupo 1.1

	ID	Modelo matemático		GAP	Heurística construtiva	
		Makespan	# pacientes atendidos		*Makespan*	# pacientes atendidos
5 pacientes	1.1_5_1	1d 16h 26min	5	0,00*	2d 13h 55min	5
	1.1_5_2	1d 15h 23min	5	0,00	1d 15h 38min	5
	1.1_5_3	1d 17h 13min	5	0,00*	2d 13h 9min	5
	1.1_5_4	1d 15h 31min	5	0,00*	2d 13h 57min	5
	1.1_5_5	1d 15h 12min	5	0,00	2d 13h 16min	5
	1.1_5_6	1d 16h 27min	5	0,00	1d 16h 27min	5
	1.1_5_7	1d 17h 19min	5	0,00*	2d 13h 26min	5
7 pacientes	1.1_7_1	2d 16h 58min	7	0,00*	3d 13h 25min	7
	1.1_7_2	2d 17h 16min	7	0,00	3d 14h 54min	7
	1.1_7_3	3d 13h 46min	7	0,00*	3d 15h 34min	7
	1.1_7_4	2d 17h 46min	7	0,00*	3d 14h 40min	7
	1.1_7_5	2d 16h 51min	7	0,00	2d 16h 51min	7
	1.1_7_6	2d 17h 34min	7	0,00	2d 18h 9min	7
	1.1_7_7	3d 14h 51min	7	0,00*	3d 16h 8min	7
10 pacientes	1.1_10_1	4d 16h 41min	10	0,00*	3d 17h 32min	8
	1.1_10_2	4d 13h 20min	10	0,00*	3d 14h 54min	8
	1.1_10_3	4d 17h 03min	10	0,00*	4d 17h 3min	9
	1.1_10_4	4d 16h 16min	10	0,00*	4d 18h 23min	9
	1.1_10_5	4d 14h 03min	10	0,00*	3d 16h 29min	8
	1.1_10_6	4d 17h 24min	10	0,00*	3d 16h 39min	8
	1.1_10_7	4d 14h 33min	10	0,00*	3d 13h 31min	8
12 pacientes	1.1_12_1	4d 16h 41min	11	0,00*	4d 13h 46min	9
	1.1_12_2	4d 13h 42min	11	0,00*	3d 14h 54min	9
	1.1_12_3	4d 17h 03min	11	0,00*	4d 15h 11min	10
	1.1_12_4	4d 16h 16min	11	0,00*	3d 17h 38min	9
	1.1_12_5	4d 14h 03min	11	0,00*	4d 14h 7min	9
	1.1_12_6	4d 16h 18min	11	0,00*	3d 16h 10min	9
	1.1_12_7	4d 14h 33min	11	0,00*	4d 12h 50min	9
15 pacientes	1.1_15_1	4d 15h 23min	13	10,59	4d 12h 56min	11
	1.1_15_2	4d 14h 34min	13	7,54	4d 15h 38min	12
	1.1_15_3	4d 15h 03min	13	0,00*	4d 13h 3min	12
	1.1_15_4	4d 16h 16min	13	0,00*	3d 18h 4min	11
	1.1_15_5	4d 14h 07min	13	7,01	4d 17h 42min	11
	1.1_15_6	4d 15h 06min	13	0,00*	4d 15h 56min	12
	1.1_15_7	4d 14h 33min	13	0,00*	4d 16h 2min	12

* indica que o *GAP* de tolerância foi atingido

Tabela 34: Comparação com solução ótima - Grupo 1.2

	ID	Modelo matemático		GAP	Heurística construtiva	
		Makespan	# pacientes atendidos		*Makespan*	# pacientes atendidos
5 pacientes	1.2_5_1	3d 12h 45min	5	0,00*	3d 13h 25min	5
	1.2_5_2	2d 13h 27min	4	0,00*	2d 13h 23min	4
	1.2_5_3	2d 14h 48min	5	0,00	4d 13h 3min	5
	1.2_5_4	2d 14h 42min	5	0,00	3d 12h 54min	4
	1.2_5_5	2d 14h 53min	5	0,00	4d 12h 42min	4
	1.2_5_6	2d 13h 22min	5	0,00	2d 13h 22min	5
	1.2_5_7	2d 16h 30min	5	0,00	4d 13h 3min	5
7 pacientes	1.2_7_1	3d 15h 11min	7	0,00*	4d 12h 56min	5
	1.2_7_2	4d 12h 54min	6	0,00	4d 12h 54min	5
	1.2_7_3	4d 14h 33min	7	0,00	4d 13h 3min	5
	1.2_7_4	4d 13h 14min	7	0,00	4d 13h 14min	6
	1.2_7_5	4d 12h 42min	6	0,00*	4d 12h 46min	4
	1.2_7_6	4d 13h 10min	7	0,00*	4d 13h 18min	6
	1.2_7_7	4d 13h 12min	7	0,00*	4d 13h 12min	6
10 pacientes	1.2_10_1	4d 13h 25min	8	0,00*	4d 12h 56min	6
	1.2_10_2	4d 12h 54min	7	0,00	4d 12h 54min	6
	1.2_10_3	4d 15h 09min	8	0,00*	4d 13h 3min	6
	1.2_10_4	4d 14h 29min	8	0,00*	4d 13h 14min	7
	1.2_10_5	4d 12h 54min	7	0,00*	4d 12h 46min	5
	1.2_10_6	4d 14h 03min	8	0,00*	4d 13h 18min	7
	1.2_10_7	4d 13h 12min	8	0,00*	4d 13h 12min	7
12 pacientes	1.2_12_1	4d 13h 25min	9	0,00*	4d 12h 56min	8
	1.2_12_2	4d 13h 26min	8	0,00*	4d 13h 26min	7
	1.2_12_3	4d 15h 03min	9	0,00*	4d 13h 3min	6
	1.2_12_4	4d 14h 57min	9	0,00*	4d 13h 14min	7
	1.2_12_5	4d 12h 54min	8	0,00*	4d 12h 46min	5
	1.2_12_6	4d 13h 11min	9	0,00*	4d 13h 18min	8
	1.2_12_7	4d 13h 12min	9	0,00*	4d 13h 12min	7
15 pacientes	1.2_15_1	4d 14h 35min	10	0,00*	4d 13h 17min	8
	1.2_15_2	4d 13h 26min	9	0,00*	4d 13h 7min	8
	1.2_15_3	4d 15h 03min	9	0,00*	4d 13h 17min	7
	1.2_15_4	4d 14h 29min	9	21,07	4d 13h 14min	7
	1.2_15_5	4d 13h 14min	8	0,00*	4d 13h 14min	5
	1.2_15_6	4d 14h 19min	10	0,00*	4d 13h 18min	9
	1.2_15_7	4d 13h 27min	10	0,00*	4d 13h 27min	7

* indica que o *GAP* de tolerância foi atingido

Tabela 35: Comparação com solução ótima - Grupo 2.1

	ID	Modelo matemático			Heurística construtiva	
		Makespan	# pacientes atendidos	GAP	*Makespan*	# pacientes atendidos
5 pacientes	2.1_5_1	2d 13h 54min	5	0,00*	4d 14h 53min	5
	2.1_5_2	1d 16h 32min	5	0,00*	4d 18h 54min	4
	2.1_5_3	2d 12h 42min	5	0,00	4d 13h 3min	5
	2.1_5_4	2d 13h 10min	5	0,00	4d 13h 56min	5
	2.1_5_5	2d 13h 08min	5	0,00*	4d 16h 41min	5
	2.1_5_6	2d 13h 22min	5	0,00	4d 13h 18min	5
	2.1_5_7	2d 13h 16min	5	0,00	4d 13h 2min	5
7 pacientes	2.1_7_1	2d 15h 46min	7	0,00*	4d 16h 13min	5
	2.1_7_2	2d 14h 09min	7	0,00*	4d 18h 54min	5
	2.1_7_3	2d 15h 00min	7	0,00*	4d 18h 12min	5
	2.1_7_4	2d 14h 45min	7	0,00*	4d 15h 12min	6
	2.1_7_5	2d 14h 31min	7	0,00*	4d 16h 41min	5
	2.1_7_6	2d 16h 39min	7	0,00	4d 15h 36min	5
	2.1_7_7	2d 17h 01min	7	0,00	4d 14h 29min	5
10 pacientes	2.1_10_1	4d 13h 25min	10	0,00*	4d 14h 53min	6
	2.1_10_2	4d 12h 54min	10	0,00*	4d 16h 50min	6
	2.1_10_3	4d 13h 03min	10	0,00	4d 15h 3min	6
	2.1_10_4	3d 16h 00min	10	0,00*	4d 14h 29min	6
	2.1_10_5	3d 15h 18min	10	0,00*	4d 14h 7min	6
	2.1_10_6	4d 12h 37min	10	0,00	4d 16h 13min	6
	2.1_10_7	4d 12h 53min	10	0,00*	4d 17h 48min	6
12 pacientes	2.1_12_1	4d 13h 25min	11	5,87	4d 13h 57min	6
	2.1_12_2	4d 14h 40min	12	0,02	4d 16h 50min	7
	2.1_12_3	4d 15h 42min	12	0,03	4d 15h 3min	6
	2.1_12_4	4d 15h 00min	12	0,01	4d 14h 29min	6
	2.1_12_5	4d 14h 26min	12	0,02	4d 14h 12min	6
	2.1_12_6	4d 14h 56min	12	0,02	4d 16h 13min	6
	2.1_12_7	4d 13h 36min	12	0,00*	4d 17h 48min	6
15 pacientes	2.1_15_1	4d 16h 31min	13	8,05	4d 17h 19min	6
	2.1_15_2	4d 17h 11min	15	0,03	4d 14h 58min	6
	2.1_15_3	4d 15h 16min	13	4,23	4d 14h 27min	6
	2.1_15_4	4d 15h 12min	13	3,80	4d 16h 24min	7
	2.1_15_5	4d 15h 00min	13	3,53	4d 17h 16min	6
	2.1_15_6	4d 15h 51min	14	1,13	4d 18h 34min	7
	2.1_15_7	4d 19h 29min	14	1,18	4d 17h 54min	6

* indica que o *GAP* de tolerância foi atingido

Tabela 36: Comparação com solução ótima - Grupo 2.2

	ID	Modelo matemático			Heurística construtiva	
		Makespan	# pacientes atendidos	GAP	*Makespan*	# pacientes atendidos
5 pacientes	2.2_5_1	3d 13h 34min	5	0,00*	4d 13h 46min	4
	2.2_5_2	3d 14h 49min	5	0,00*	4d 14h 35min	4
	2.2_5_3	3d 15h 34min	5	0,00	4d 13h 3min	4
	2.2_5_4	3d 14h 40min	5	0,00	3d 15h 6min	4
	2.2_5_5	3d 14h 42min	5	0,00	3d 15h 2min	4
	2.2_5_6	3d 14h 43min	5	0,00	4d 14h 22min	4
	2.2_5_7	3d 16h 37min	5	0,00*	4d 13h 2min	4
7 pacientes	2.2_7_1	3d 13h 34min	6	0,00	4d 14h 12min	5
	2.2_7_2	4d 15h 34min	7	0,00	4d 14h 40min	5
	2.2_7_3	3d 15h 34min	6	0,00	4d 14h 26min	5
	2.2_7_4	3d 14h 40min	6	0,00*	4d 13h 56min	5
	2.2_7_5	4d 14h 59min	7	0,00*	4d 14h 12min	4
	2.2_7_6	4d 15h 36min	7	0,00	4d 12h 37min	4
	2.2_7_7	3d 14h 51min	6	0,00*	4d 13h 3min	5
10 pacientes	2.2_10_1	3d 14h 52min	6	0,00	4d 14h 12min	5
	2.2_10_2	4d 17h 16min	7	0,00*	4d 14h 40min	5
	2.2_10_3	3d 15h 34min	6	0,00	4d 14h 26min	5
	2.2_10_4	4d 15h 12min	7	0,00	4d 14h 43min	5
	2.2_10_5	4d 12h 54min	7	0,00	4d 14h 7min	5
	2.2_10_6	4d 17h 16min	7	0,00*	4d 14h 22min	5
	2.2_10_7	3d 13h 35min	6	0,00*	4d 13h 3min	5
12 pacientes	2.2_12_1	3d 14h 52min	7	0,00	4d 14h 12min	5
	2.2_12_2	4d 16h 57min	8	0,00*	4d 14h 40min	5
	2.2_12_3	4d 18h 16min	8	0,00	4d 13h 35min	5
	2.2_12_4	4d 15h 12min	8	0,00	4d 14h 43min	5
	2.2_12_5	4d 12h 54min	8	0,00	4d 14h 12min	5
	2.2_12_6	4d 16h 51min	8	0,00*	4d 13h 29min	5
	2.2_12_7	3d 13h 40min	7	0,00*	4d 12h 50min	5
15 pacientes	2.2_15_1	4d 12h 23min	9	0,00	4d 13h 57min	5
	2.2_15_2	4d 16h 57min	10	0,00*	4d 13h 44min	5
	2.2_15_3	4d 18h 16min	9	0,00	4d 14h 1min	6
	2.2_15_4	4d 16h 49min	10	0,00	4d 14h 30min	5
	2.2_15_5	4d 14h 07min	10	0,00*	4d 14h 12min	6
	2.2_15_6	4d 14h 24min	10	0,00	4d 18h 12min	7
	2.2_15_7	3d 14h 51min	9	0,00*	4d 13h 12min	6

* indica que o *GAP* de tolerância foi atingido

Tabela 37: Comparação com solução ótima- Grupo 3.1

	ID	Modelo matemático		GAP	Heurística construtiva	
		Makespan	# pacientes atendidos		*Makespan*	# pacientes atendidos
5 pacientes	3.1_5_1	2d 13h 54min	5	0,00*	2d 14h 0min	5
	3.1_5_2	1d 17h 02min	5	0,00*	2d 13h 19min	5
	3.1_5_3	2d 12h 42min	5	0,00*	2d 13h 15min	5
	3.1_5_4	2d 13h 03min	5	0,00	2d 13h 57min	5
	3.1_5_5	2d 13h 08min	5	0,00	2d 13h 15min	5
	3.1_5_6	2d 13h 08min	5	0,00*	2d 14h 17min	5
	3.1_5_7	2d 13h 14min	5	0,00	2d 13h 20min	5
7 pacientes	3.1_7_1	3d 13h 24min	7	0,00*	4d 14h 46min	7
	3.1_7_2	2d 18h 09min	7	0,00	3d 14h 46min	7
	3.1_7_3	3d 13h 50min	7	0,00*	3d 15h 32min	7
	3.1_7_4	3d 13h 46min	7	0,00*	4d 14h 16min	7
	3.1_7_5	2d 16h 51min	7	0,00	4d 12h 55min	7
	3.1_7_6	3d 13h 14min	7	0,00*	3d 13h 40min	7
	3.1_7_7	3d 14h 49min	7	0,00*	3d 16h 2min	7
10 pacientes	3.1_10_1	3d 13h 43min	8	0,00*	4d 14h 46min	8
	3.1_10_2	4d 14h 35min	10	0,00*	3d 14h 46min	8
	3.1_10_3	3d 15h 31min	8	0,00*	3d 14h 50min	8
	3.1_10_4	4d 14h 49min	9	0,00*	4d 18h 29min	9
	3.1_10_5	4d 14h 07min	10	0,00*	4d 12h 38min	7
	3.1_10_6	4d 18h 52min	10	0,00*	3d 16h 39min	8
	3.1_10_7	4d 12h 52min	9	0,00*	4d 13h 13min	8
12 pacientes	3.1_12_1	3d 14h 41min	9	0,00*	4d 14h 46min	9
	3.1_12_2	4d 17h 16min	11	0,00*	4d 14h 34min	9
	3.1_12_3	4d 18h 16min	10	0,00*	4d 18h 12min	9
	3.1_12_4	4d 15h 03min	10	0,00*	3d 17h 27min	8
	3.1_12_5	4d 16h 28min	11	0,00*	4d 12h 38min	7
	3.1_12_6	4d 15h 51min	10	0,00*	4d 14h 21min	9
	3.1_12_7	4d 12h 52min	10	0,00*	4d 13h 13min	8
15 pacientes	3.1_15_1	4d 14h 35min	11	22,18	4d 13h 5min	8
	3.1_15_2	4d 17h 16min	12	0,00*	4d 14h 34min	8
	3.1_15_3	4d 16h 20min	10	15,61	4d 18h 14min	9
	3.1_15_4	4d 15h 3min	11	20,28	4d 14h 27min	10
	3.1_15_5	4d 15h 48min	12	2,5	4d 14h 16min	10
	3.1_15_6	4d 17h 10min	12	6,02	4d 14h 29min	9
	3.1_15_7	4d 15h 12min	12	0,02	4d 16h 0min	9

* indica que o *GAP* de tolerância foi atingido

Tabela 38: Comparação com solução ótima - Grupo 3.2

	ID	Modelo matemático		GAP	Heurística construtiva	
		Makespan	# pacientes atendidos		*Makespan*	# pacientes atendidos
5 pacientes	3.2_5_1	2d 13h 58min	5	0,00*	2d 14h 6min	5
	3.2_5_2	2d 15h 08min	5	0,00*	4d 14h 39min	5
	3.2_5_3	2d 14h 42min	5	0,00*	3d 13h 9min	5
	3.2_5_4	2d 14h 32min	5	0,00*	3d 13h 37min	5
	3.2_5_5	2d 14h 34min	5	0,00	3d 13h 59min	5
	3.2_5_6	2d 13h 22min	5	0,00	4d 13h 14min	5
	3.2_5_7	2d 14h 55min	5	0,00	3d 14h 6min	5
7 pacientes	3.2_7_1	3d 14h 55min	7	0,00*	4d 13h 32min	6
	3.2_7_2	3d 13h 11min	7	0,00*	4d 13h 41min	6
	3.2_7_3	3d 14h 51min	7	0,00*	4d 13h 57min	6
	3.2_7_4	3d 13h 48min	7	0,00*	4d 13h 18min	6
	3.2_7_5	3d 13h 57min	7	0,00*	4d 12h 52min	6
	3.2_7_6	3d 14h 38min	7	0,00*	4d 14h 34min	6
	3.2_7_7	3d 16h 20min	7	0,00*	4d 13h 2min	6
10 pacientes	3.2_10_1	4d 13h 16min	8	0,00*	4d 13h 32min	7
	3.2_10_2	4d 14h 35min	9	0,00*	4d 14h 39min	7
	3.2_10_3	4d 13h 16min	8	0,00	4d 13h 57min	7
	3.2_10_4	4d 15h 04min	9	0,00*	4d 13h 18min	7
	3.2_10_5	4d 14h 25min	9	0,00*	4d 12h 52min	7
	3.2_10_6	4d 17h 13min	9	0,00*	4d 14h 34min	7
	3.2_10_7	4d 14h 35min	9	0,00*	4d 13h 2min	6
12 pacientes	3.2_12_1	4d 13h 27min	9	0,00*	4d 14h 46min	8
	3.2_12_2	4d 14h 32min	10	0,00*	4d 14h 30min	8
	3.2_12_3	4d 14h 23min	9	0,00*	4d 14h 39min	7
	3.2_12_4	4d 15h 12min	10	0,00*	4d 13h 18min	7
	3.2_12_5	4d 14h 25min	10	0,00*	4d 14h 39min	8
	3.2_12_6	4d 16h 48min	10	0,00*	4d 13h 14min	7
	3.2_12_7	4d 14h 37min	10	0,00*	4d 13h 2min	7
15 pacientes	3.2_15_1	4d 15h 0min	10	34,24	4d 13h 32min	7
	3.2_15_2	4d 14h 32min	10	18,27	4d 13h 52min	7
	3.2_15_3	4d 14h 53min	10	14,47	4d 14h 39min	7
	3.2_15_4	4d 14h 40min	10	16,63	4d 14h 26min	7
	3.2_15_5	4d 14h 45min	10	23,92	4d 13h 12min	7
	3.2_15_6	4d 18h 53min	10	0,00*	4d 14h 54min	8
	3.2_15_7	4d 14h 35min	10	13,91	4d 16h 0min	8

* indica que o *GAP* de tolerância foi atingido

6 CONCLUSÕES

Este trabalho apresentou um estudo para o Problema de Programação de Cirurgias Eletivas com Múltiplos Recursos e Múltiplas Etapas.

Com objetivo de estudar e analisar os principais trabalhos de programação de cirurgias eletivas, uma revisão de literatura foi desenvolvida. O Problema de Programação de Cirurgias Eletivas está inserido no nível hierárquico operacional e tem como objetivo *alocar* recursos às etapas dos pacientes e *programar* essa etapas no tempo. Da revisão apresentada, foi possível identificar que grande parte dos trabalhos não consideram múltiplos estágios e múltiplos recursos em seus modelos e, ainda, não tratam da etapa de pré-cirurgia, responsável por 12% dos atrasos em cirurgias (WEINBROUM; EKSTEI; EZRI, 2003).

Em seu trabalho os autores Pham e Klinkert (2008) apresentam um modelo de programação linear inteira mista em um ambiente *Job Shop Flexível* para o problema com múltiplos recursos e etapas. O presente trabalho buscou estudar e aprimorar o modelo dos autores, com objetivo de apresentar um modelo que possa ser utilizado para encontrar a solução ótima para pequenas instâncias. Algumas melhorias foram apontadas como: retirada do módulo fictício e do parâmetro α da função objetivo, redução do número de restrições e uma proposta para considerar a priorização dos pacientes.

Nesse sentido, um modelo foi proposto com uma nova função objetivo para problema em dois níveis, excluindo o fator α. No primeiro nível, o *solver* utilizando o modelo proposto encontra soluções que atendem o maior número de pacientes atendidos e, em seguida, busca dentre essas soluções aqueles que possuem menor *makespan*. Testes foram realizados apontando a importância de tratar esse problema em dois níveis.

No sentido de reduzir o número de restrições do modelo, dois conjuntos de restrições foram alteradas. Testes preliminares apontam que a redução das restrições também reduz o esforço computacional necessário para que o *solver* encontre a solução ótima. A fim de reduzir a complexidade dos dados de entrada, o módulo fictício foi retirado do modelo

através de pequenas alterações em alguns conjuntos de restrições. Com essas alterações, as etapas dos pacientes não atendidos não ficam alocadas ao módulo fictício deixando o resultado da solução mais intuitivo.

Adicionalmente, uma proposta para priorizar os pacientes foi apresentada. Uma vez que a função objetivo só leva em consideração o número de pacientes atendidos, a solução encontrada pode apresentar um cenário que priorize o atendimento de pacientes com tempo de processamento reduzido. Esse cenário não é desejável em hospitais, pois pacientes mais graves, normalmente possuem etapas mais demoradas. Dessa forma, testes foram realizados comparando a estratégia de priorização apresentada em Pham e Klinkert (2008) com a estratégia de priorização de pacientes proposta. Os testes realizados apontam que a estratégia proposta favorece o atendimento dos pacientes prioritários sem contudo piorar o valor do *makespan* da solução. Sendo assim, uma estratégia mais adequada quando comparada com a estratégia dos autores.

Por fim, foram realizados testes a fim de analisar a sensibilidade do modelo. Conforme esperado, instâncias com disponibilidade de recursos mais apertadas obtiveram resultados piores, enquanto instâncias com mais ociosidade no sistema tiveram resultados melhores. Para as instâncias com 15 pacientes em cerca de 60% dos testes o *solver* não pode encontrar uma solução ótima.

O problema tratado nesse trabalho pode ser visto como *Job Shop Flexível* com minimização do *makespan* que é classificado como *NP-hard* (GAREY; JOHNSON; SETHI, 1976). Dessa forma, o esforço computacional exigido para solucionar instâncias de grande porte é muito grande. Nesse contexto, a fim de apresentar uma alternativa para programar instâncias de grande porte, esse trabalho também apresenta uma proposta de heurística construtiva.

Para utilizar o modelo matemático é necessário realizar um pré-processamento dos dados de entrada para criar os módulos analisando as janelas de tempo de cada um dos recursos. Esse pré-processamento reduz consideravelmente o tempo para encontrar o ótimo, pois não é necessário alocar recurso a recurso e sim o módulo que contém vários recursos. Porém, esse pré-processamento também é um problema de combinatório que pode ter um custo alto para a operação de um hospital. A proposta de heurística construtiva dispensa esse etapa, uma vez que considera cada recurso disponível separadamente.

O Problema de Programação de Cirurgias determina que todas as etapas de um paciente devem ser processadas, se pelo uma delas for processada. Da mesma forma, para processar uma etapa é necessário que todos os recursos necessários estejam disponíveis.

Dessa forma, a heurística construtiva proposta busca alocar cada paciente e cada etapa separadamente, evitando que alguma etapa ou algum recurso não seja agendado por falta de disponibilidade.

Testes foram realizados com as mesmas instâncias utilizadas no teste do matemático a fim de comparar os resultados da heurística e seu tempo computacional. Apesar de em somente 27,7% das instâncias a heurística alocou a mesma quantidade de pacientes, o esforço computacional é consideravelmente reduzido, rodando as 245 instâncias em apenas 3 segundos.

A seguir apresentam-se propostas de trabalho futuros:

- Incrementar a heurística construtiva com uma estrutura que permita a alocação de pacientes considerando regras dinâmicas. Como, por exemplo, a flexibilidade dinâmica que analisa instante a instante quantos recursos estão disponíveis para processar aquele paciente;
- Propor e testar novas regras de ordenação para o tipo de recursos;
- Testar regra de alocação de recurso que considere o nivelamento dos recursos disponíveis. Em um hospital, a programação dos recursos deve ser equilibrada para não sobrecarregar somente alguns recursos. Uma possibilidade seria o estudo da inserção desse objetivo como um terceiro nível;
- Testar meta-heurísticas diferentes considerando os resultados obtidos pela heurística construtiva como semente.

REFERÊNCIAS

ALMEIDA, A. N. P. M. D. *Análise da Oferta de Leitos de UTI no Brasil*. 22 p. Tese (Doutorado) — 1 Universidade Federal do Rio Grande do Sul, 2006.

ARENALES, M. et al. *Pesquisa operacional [recurso eletrônico]*. recurso digital. Rio de Janeiro: Elsevier: ABEPRO, 2011. 524 p.

AUGUSTO, V.; XIE, X.; PERDOMO, V. Operating theatre scheduling with patient recovery in both operating rooms and recovery beds. *Computers and Industrial Engineering*, v. 58, p. 231–238, 2010.

BALAS, E. Machine Sequencing Via Disjunctive Graphs: An Implicit Enumeration Algorithm. *Operations Research*, v. 17, n. 6, p. 941–957, 1969.

BANDITORI, C.; CAPPANERA, P.; VISINTIN, F. Investigating the Relationship Between Resources Balancing and Robustness in Master Surgical Scheduling. In: Matta et al. (Ed.). *Springer Proceedings in Mathematics and Statistics*. Milan, Italy: Springer, Cham, 2014. p. 149–162.

CARDOEN, B. Operating room planning and scheduling: Solving a surgical case sequencing problem. *4OR*, v. 8, p. 101–104, 2010.

CARDOEN, B.; DEMEULEMEESTER, E.; BELIËN, J. Optimizing a multiple objective surgical case sequencing problem. *International Journal of Production Economics*, v. 119, p. 354–366, 2009.

CARDOEN, B.; DEMEULEMEESTER, E.; BELIËN, J. Sequencing surgical cases in a day-care environment: An exact branch-and-price approach. *Computers and Operations Research*, v. 36, n. 9, p. 2660–2669, 2009.

CESCHIA, S.; SCHAERF, A. Dynamic patient admission scheduling with operating room constraints, flexible horizons, and patient delays. *Journal of Scheduling*, v. 19, p. 377–389, 2016.

CONFORTI, D.; GUERRIERO, F.; GUIDO, R. A multi-objective block scheduling model for the management of surgical operating rooms: New solution approaches via genetic algorithms. In: *2010 IEEE Workshop on Health Care Management, WHCM 2010*. Venice, Italy: IEEE, 2010. p. 1–5.

CONSELHO FEDERAL DE MEDICINA. *Datafolha mostra como a população paulista enxerga a oferta da assistência no estado*. São Paulo: [s.n.], 2014. Disponível em: ⟨https://portal.cfm.org.br/⟩.

DAMM, R. B. *Métodos mono e multiobjetivo para o problema de escalonamento de técnicos de campo*. 199 p. Tese (Doutorado) — Universidade de São Paulo, 2016.

DEMEULEMEESTER, E. et al. Operating Room Planning and Scheduling. In: B.T. Denton (Ed.). *Handbook of Healthcare Operations Management: Methods and Applications*. New York: Springer Science+ Business Media, 2013. cap. 5, p. 121–152.

DEMPE, S. *Foundations of Bilevel Programming*. 1. ed. Boston: Springer US, 2002. 309 p.

DEXTER, F. et al. Use of Linear Programming to Estimate Impact of Changes in a Hospital's Operating Room Time Allocation on Perioperative Variable Costs. *Anesthesiology*, v. 96, p. 718–24, 2002.

DOH, H.-H. et al. A priority scheduling approach for flexible job shops with multiple process plans. *International Journal of Production Research*, v. 51, n. 12, p. 3748–3764, 2013.

FEI, H. et al. Solving surgical cases assignment problem by a branch-and-price approach. *International Journal of Production Economics*, v. 112, p. 96–108, 2008.

FEI, H.; MESKENS, N.; CHU, C. An operating theatre planning and scheduling problem in the case of a "block scheduling" strategy. In: *Service Systems and Service Management, 2006 International Conference on*. Troyes, France: IEEE, 2007.

FEI, H.; MESKENS, N.; CHU, C. A planning and scheduling problem for an operating theatre using an open scheduling strategy. *Computers and Industrial Engineering*, 2010.

FEI, H. et al. The endoscopy scheduling problem: A case study with two specialised operating rooms. *International Journal of Production Economics*, v. 120, p. 452–462, 2009.

GAREY, M.; JOHNSON, D.; SETHI, R. The Complexity of Flowshop and Jobshop Scheduling. *Mathematics of Operations Research*, v. 1, n. 2, p. 117–129, 1976.

GARTNER, D.; KOLISCH, R. Scheduling the hospital-wide flow of elective patients. *European Journal of Operational Research*, v. 233, p. 689–699, 2014.

GHOLAMI, O.; SOTSKOV, Y. N. A fast heuristic algorithm for solving parallel-machine job-shop scheduling problems. *The International Journal of Advanced Manufacturing Technology*, v. 70, p. 531–546, 2014.

GONZALEZ, T.; SAHNI, S. Open shop scheduling do minimize finish time. *Association for Computing Machinery*, v. 23, n. 4, p. 665–679, 1979.

GUPTA, D.; DENTON, B. Appointment scheduling in health care : Challenges and opportunities. *IIE Transactions*, v. 40, p. 800–8019, 2008.

HANS, E. W.; VANBERKEL, P. T. Operating Theatre Planning and Scheduling. In: HALL, R. (Ed.). *Handbook of Healthcare System Scheduling*. 1. ed. New York: Springer Science+Business Media, 2012. cap. 5, p. 105–130.

HORTENCIO, H. P.; RONCONI, D. P.; MIYAKE, D. I. Produtividade na programação de centros cirúrgicos: revisão de literatura. In: *XX Simpósio de Administração da Produção, Logística e Operações Internacionais*. São Paulo: FGV-EAESP, 2017.

INSTITUTO DATA FOLHA. *Opinião dos Paulistas sobre o Atendimento Público na Área da Saúde*. São Paulo: [s.n.], 2014. 50 p.

JITTAMAI, P.; KANGWANSURA, T. A hospital admission planning model for emergency and elective patients under stochastic resource requirements and no-shows. In: *IEEE International Conference on Industrial Engineering and Engineering Management*. Singapore, Singapore: IEEE, 2011.

KAYE, A. D. et al. Effective Strategies in Improving Operating Room Case Delays and Cancellations at an Academic Medical Center. *Journal of Medical Practice Management*, v. 30, n. 6, p. 24–29, 2015.

MA, G. et al. Solving the strategic case mix problem optimally by using branch-and-price algorithms. In: *International Conference on Operational Research Applied to Health Services (ORAHS)*. Leuven, Belgium: ORAHS, 2011. p. 1–10.

MAGERLEIN, J. M.; MARTIN, J. B. Surgical Demand Scheduling: A, Review. *Health Services Research*, v. 13, p. 418–433, 1978.

MELO, E. L.; RONCONI, D. P. Regras de prioridade eficientes que exploram características do Job Shop Flexível para a minimização do atraso total Palavras-chave. *Produção*, v. 25, n. 1, p. 79–91, 2015.

MOCCELLIN, J. V. et al. Heuristic algorithms for scheduling hybrid flow shops with machine blocking and setup times. *Journal of the Brazilian Society of Mechanical Sciences and Engineering*, v. 40, p. 40, 2018.

MOLINA-PARIENTE, J.; FRAMINAN, J. Testing planning policies for solving the Elective Case Scheduling phase: A real application. In: *35th international conference on operational research applied to health services*. Leuven, Belgium: ORAHS, 2009. p. 1–10.

MULHOLLAND, M. W.; ABRAHAMSE, P.; BAHL, V. Linear programming to optimize performance in a Department of Surgery. *Journal of the American College of Surgeons*, v. 200, p. 861–868, 2005.

NELSON, A.; ALMEIDA, P. M. D. Universidade Federal do Rio Grande do Sul Escola de Administração Curso de Especialização em Políticas Públicas. 2006.

OZKARAHAN, I. Allocation of surgeries to operating rooms by goal programing. *Journal of Medical Systems*, v. 24, p. 339–378, 2000.

PANNI, M. K. et al. Improving operating room first start efficiency - Value of both checklist and a pre-operative facilitator. *Acta Anaesthesiologica Scandinavica*, v. 57, p. 1118–1123, 2013.

PERDOMO, V.; AUGUSTO, V.; XIE, X. Operating Theatre Scheduling Using Lagrangian Relaxation. *European Journal of Industrial Engineering*, v. 2, n. 2, p. 172–189, 2008.

PHAM, D. N.; KLINKERT, A. Surgical case scheduling as a generalized job shop scheduling problem. *European Journal of Operational Research*, v. 185, p. 1011–1025, 2008.

PINEDO, M. L. *Scheduling - Theory, Algorithms, and Systems*. 4. ed. New York: Springer-Verlag New York, 2012. XX, 676 p.

RIISE, A.; MANNINO, C.; BURKE, E. K. Modelling and solving generalised operational surgery scheduling problems. *Computers and Operations Research*, v. 66, p. 1–11, 2016.

ROSSI, F. L.; NAGANO, M. S.; SAGAWA, J. K. An effective constructive heuristic for permutation flow shop scheduling problem with total flow time criterion. *International Journal of Advanced Manufacturing Technology*, v. 90, p. 93–107, 2017.

ROUGH, S. S.; MCDANIEL, M.; ROUGH, J. R. Effective use of workload and productivity monitoring tools in health-system pharmacy, part 1. *Am J Health-Syst Pharm*, v. 67, p. 300–311, 2010.

RUIZ, R.; SERIFOGLU, F. S.; URLING, T. Modeling realistic hybrid flexible flowshop scheduling problems. *Computers & Operations Research*, v. 35, n. 4, p. 1151–1175, 2014.

SAMUDRA, M. et al. Scheduling operating rooms: achievements, challenges and pitfalls. *Journal of Scheduling*, v. 19, p. 493–525, 2016.

SAMUT, P. K.; CAFRI, R. Analysis of the Efficiency Determinants of Health Systems in OECD Countries by DEA and Panel Tobit. *Social Indicators Research*, v. 129, p. 113–132, 2016.

SOBREIRO, V. A.; NAGANO, M. S. Proposta de uma heurística construtiva baseada na TOC para definição de mix de produção. *Produção*, v. 23, n. 3, p. 468–477, 2013.

TAILLARD, E. Benchmarks for basic scheduling problems. *European Journal of Operational Research*, v. 64, p. 278–285, 1993.

TESTI, A.; TÀNFANI, E. Tactical and operational decisions for operating room planning: Efficiency and welfare implications. *Health Care Management Science*, v. 12, p. 363– 373, 2009.

T'KINDT, V.; BILLAUT, J.-C. *Multicriteria Scheduling Theory, Models and Algorithms*. 2. ed. Berlin: Springer-Verlag Berlin Heidelberg, 2006. 360 p.

WEINBROUM, A. A.; EKSTEIN, P.; EZRI, T. Efficiency of the operating room suite. *American Journal of Surgery*, v. 185, p. 244–250, 2003.

XU, J.; CHIU, S. Y. Effective heuristic procedures for a field technician scheduling problem. *Journal of Heuristics*, v. 7, n. 5, p. 495–509, 2001.