UNIVERSIDADE ESTADUAL DE MARINGÁ
CENTRO DE TECNOLOGIA
DEPARTAMENTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

GUSTAVO BORELLI BEDENDO

Investigação de um novo algoritmo genético para o problema de reescalonamento de enfermeiros

Maringá
2019

GUSTAVO BORELLI BEDENDO

Investigação de um novo algoritmo genético para o problema de reescalonamento de enfermeiros

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação do Departamento de Informática, Centro de Tecnologia da Universidade Estadual de Maringá, como requisito parcial para obtenção do titulo de Mestre em Ciência da Computação.

Orientador: Prof. Dr. Ademir Aparecido Constantino

Maringá
2019

Dados Internacionais de Catalogação-na-Publicação (CIP)
(Biblioteca Central - UEM, Maringá - PR, Brasil)

Bedendo, Gustavo Borelli

B411i Investigação de um novo algoritmo genético para o problema de reescalonamento de enfermeiros / Gustavo Borelli Bedendo. -- Maringá, PR, 2019.
92 f.: il. color., figs., tabs.

Orientador: Prof. Dr. Ademir Aparecido Constantino.
Dissertação (Mestrado) - Universidade Estadual de Maringá, Centro de Ciências da Tecnologia, Departamento de Engenharia de Informatica, Programa de Pós-Graduação em Ciência da Computação, 2019.

1. Reescalonamento de enfermeiros. 2. Meta-heurística. 3. Otimização combinatória. 4. Pesquisa operacional. 5. Algoritmos genéticos. I. Constantino, Ademir Aparecido, orient. II. Universidade Estadual de Maringá. Centro de Ciências da Tecnologia. Departamento de Engenharia de Informatica. Programa de Pós-Graduação em Ciência da Computação. III. Título.

CDD 23.ed. 005.453

Ademir Henrique dos Santos - CRB-9/1065

FOLHA DE APROVAÇÃO

GUSTAVO BORELLI BEDENDO

Investigação de um novo algoritmo genético para o problema de reescalonamento de enfermeiros

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação do Departamento de Informática, Centro de Tecnologia da Universidade Estadual de Maringá, como requisito parcial para obtenção do título de Mestre em Ciência da Computação pela Banca Examinadora composta pelos membros:

BANCA EXAMINADORA

Prof. Dr. Ademir Aparecido Constantino
Universidade Estadual de Maringá – DIN/UEM

Prof. Dr. Rodrigo Calvo
Universidade Estadual de Maringá – DIN/UEM

Prof. Dr. Rodrigo Clemente Thom de Souza
Universidade Federal do Paraná – UFPR-Jandaia do Sul

Aprovada em: 26 de junho de 2019.
Local da defesa: Sala 101, Bloco C56, *campus* da Universidade Estadual de Maringá.

AGRADECIMENTOS

Agradeço à Universidade Estadual de Maringá e ao Departamento de Informática pela estrutura e conhecimentos disponibilizados nesta caminhada e as pessoas que contribuíram na realização deste trabalho. Em especial:

À minha família pelo carinho, apoio e incentivo, desde o princípio de meus estudos.

Ao meu orientador professor Dr. Ademir Aparecido Constantino pelo apoio, comentários e sugestões no desenvolvimento deste trabalho.

Aos demais professores tanto de minha graduação quanto pós-graduação, que sem o conhecimento repassado por eles, não seria possível o desenvolvimento deste estudo.

À Inês, que sempre me auxiliou com prazos e problemas durante a pós-graduação.

À minha namorada Karina, que nunca deixou de me apoiar e incentivar, mesmo nos momentos de maior complicação.

À Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) e ao Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) pelo apoio financeiro concedido a este trabalho.

Meu muitíssimo obrigado.

Investigação de um novo algoritmo genético para o problema de reescalonamento de enfermeiros

RESUMO

Um alocador de enfermeiros têm a função de alocar cada enfermeiro em determinado turno de trabalho ao longo de um período de tempo. Este problema de alocação aplicado a enfermeiros é conhecido como problema de escalonamento de enfermeiros. Quando imprevistos acontecem, a validade do escalonamento original é comprometida, e portanto, deve ser restaurada através da correção do escalonamento, esta correção é conhecida como reescalonamento. Um algoritmo de reescalonamento tem a função de atribuir turnos para os enfermeiros, de modo a solucionar os imprevistos e obedecer as condições iniciais as quais o escalonamento original foi construído. Este trabalho investiga a aplicação de algoritmos genéticos com chaves aleatórias, utilizando o conceito de pareamento máximo, proveniente da teoria dos grafos, em sua decodificação. Para realização dos experimentos, foi criada uma nova base de testes (*benchmark*) para o problema em questão, utilizando diferentes quantidades de enfermeiros e ausências. Para comparação e análise dos resultados, um modelo matemático foi desenvolvido e implementado. Limitando o tempo de execução em vinte minutos, os resultados computacionais e análise demonstraram que o algoritmo proposto é capaz de obter resultados ótimos ou sub-ótimos de qualidade, similares ao método exato, superando-o em alguns casos e utilizando uma fração do tempo computacional demandado.

Palavras-chave: Algoritmo genético. RKGA. Meta-heurística. Reescalonamento de enfermeiros. Algoritmo híbrido.

Investigation of a new genetic algorithm for the nurse rerostering problem

ABSTRACT

A nurse allocator has the function of allocating each nurse in a given work shift over a period of time. This allocation problem applied to nurses is known as Nurse Scheduling Problem. When unforeseen events occur, the validity of the original scheduling is compromised, and therefore must be restored through scheduling correction, this correction is known as rescheduling or rerostering. A rerostering algorithm has the function of assigning shifts for the nurses, in order to solve the contingencies required and obey the initial conditions to which the original scheduling was constructed. This work investigates the application of genetic algorithms with random keys, using the concept of maximum pairing, derived from graph theory, in its decoding. To perform the experiments, a new benchmarking database was created for the problem in question, using different numbers of nurses and absences. In order to compare and analyze the results, a mathematical model was developed and implemented in the OPL language and then executed in the CPLEX solver. By limiting the execution time to twenty minutes, the computational results and analysis demonstrated that the proposed algorithm is able to obtain optimal or sub-optimal results similar to the exact method, using a fraction of the computational time.

Keywords: Genetic Algorithm. BRKGA. Meta-heuristics. Nurse rescheduling. Nurse reallocating. Job allocating. Job rescheduling. Hybrid algorithm.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS E ABREVIATURAS

AE: Algoritmo Evolutivo
AG: Algoritmo Genético
AGCA: Algoritmo Genético de Chaves Aleatórias
BFS: *Breath First Search*
DFS: *Depth First Search*
FIFO: *First In First Out*
FILO: *First In Last Out*
HK: Hopcroft-Karp
IA: Inteligência Artificial
MILP: *Mixed Integer Linear Programming*
NP: Não-determinístico Polinomial
NSPLIB: *Nurse Scheduling Problem Library*
NU: Não Utilizado
OPL: *Optimization Programming Language*
PEE: Problemas de Escalonamento de Enfermeiros
PEP: Problema de Escalonamento de Pessoas
PI: Programação Inteira
PRE: Problema de Reescalonamento de Enfermeiros
RF: Restrição Flexível
RKGA: *Random Key Genetic Algorithms*
RR: Restrição Rídida

SUMÁRIO

1 Introdução

1.1 Considerações Iniciais

A contextualização do conceito de reescalonamento necessita de um conhecimento sobre a origem deste problema, o escalonamento. Problemas de Escalonamento são problemas de otimização, onde se deseja designar tarefas para uma quantidade de trabalhadores dentro de um horizonte de tempo, tais escalas são avaliadas utilizando variáveis de qualidade compondo uma função objetivo. Tais designações são sujeitas a diferentes tipos de restrições que variam de acordo com cada formulação específica do problema.

Dentro desta classe de problemas, tem-se especificamente o Problema de Escalonamento de Pessoal (PEP), onde recursos humanos são designados para cumprir determinadas tarefas. O PEP é um problema clássico de otimização estudado a mais de quatro décadas (Burke et al., 2004) e, sendo um problema classificado como NP-Completo (Moz e Pato, 2007), caracteriza-se como um desafio acadêmico, dada a inexistência de algoritmos polinomiais para a sua resolução.

Um desafio rotineiro para chefes de enfermagem ou gerentes que trabalham em hospitais ou unidades de saúde é exatamente o problema de escalonamento de pessoal, processo com o intuito de prover qualidade e segurança no atendimento. Assim surge o Problema de Escalonamento de Enfermeiros (PEE), cujo objetivo é a elaboração de escalas de trabalho para os trabalhadores que, neste caso, são enfermeiros do local, onde devem ser respeitadas uma variedade de regulamentações legais, demandas operacionais e bem-estar do quadro de funcionários.

É importante ressaltar a relevância que o escalonamento de enfermeiros possui no âmbito prático. Um ponto importante diz respeito a rotatividade de funcionários, onde quão menor for a satisfação profissional maior será as chances de rotatividade alta (Unruh e Zhang, 2014). Um estudo recente (Collini et al., 2015) relaciona a alta rotatividade com

aumento de custos por parte da instituição, outrossim, outro estudo cita a deterioração da qualidade de atendimento relacionada com a insatisfação pessoal no trabalho (Aiken et al, 2015).

A qualidade no atendimento é de importância fundamental para a prevenção de acidentes. Estes podem acarretar em riscos de vida aos pacientes, processos legais, dentre outras implicações (Newman e Maylor, 2002) (Aiken et al., 2012). Problemas na equipe de enfermeiros, como estresse e fadiga são causas comuns de reclamações e podem potencialmente causar impactos negativos na qualidade de atendimento.

A qualidade de uma escala de trabalho está intimamente relacionada com a qualidade do serviço que será prestado. Quando realizado de forma eficaz o escalonamento proporciona ao funcionário um período de descanso adequado, evita a fadiga e estresse acumulado e, principalmente, é responsável por melhorar a qualidade de vida do mesmo. Dentro da realidade de escalonamento de enfermeiros deve-se levar em consideração diversos fatores na criação de uma escala, exemplifica-se as questões legais, requisitos da organização e qualidade de vida dos funcionários. Encontrar um bom balanceamento entres os diversos fatores é uma tarefa de grande complexidade computacional (Clark e Walker, 2011).

Apesar da importância de uma escala de qualidade, geralmente este processo é feito manualmente, sendo uma tarefa custosa e de difícil execução. O emprego de mecanismos automatizados pode tornar a execução desta tarefa mais eficiente (Burke et al., 2004).

Uma vez que a escala já tenha sido gerada, o Problema de Reescalonamento de Enfermeiros (PRE), de complexidade NP-Díficil (Moz e Pato, 2003), surge quando imprevistos ocorrem durante sua execução e, devido a estes, a escala criada se torna inviável. O conceito de inviabilidade está relacionado a violações de restrições que são imperativas, exemplifica-se os casos de alterações nas leis que regem sobre a classe dos enfermeiros. Em um cenário hipotético, onde uma lei altera a quantidade de horas diárias permitidas de trabalho de 8 para 6, esta certamente impactaria negativamente a escala previamente criada, exigindo então um processo de remanejamento de pessoal, pois, devido a alteração legal, muitos enfermeiros estariam trabalhando em situação de ilegalidade. Outro exemplo, este o mais utilizado pela literatura (Clark, 2015), são os problemas ocasionados pela ausência eventual de um ou mais enfermeiros, causando violações de demanda operacional, exigindo um remanejamento de enfermeiros para satisfação de todas as restrições.

O PRE pode ser considerado um tipo específico de PEE, porém, possui restrições adicionais e uma diferente métrica de qualidade de escala, onde um dos principais objetivos de otimização passa a ser a minimização de diferenças entre as escalas antiga e nova.

De forma geral, é preferível realizar alterações considerando o mesmo quadro de enfermeiros que foi utilizado no escalonamento original, visto que isso não acarretaria em custos adicionais com a contração de novos funcionários, ademais, não há dúvidas de que a adição de novos funcionários simplificaria a resolução do problema (Cheang, 2003).

Deve-se, também, levar em consideração que mudanças no escalonamento podem levar a frustração dos funcionários, pois os mesmos tendem a organizar suas vidas pessoais baseando-se no escalonamento previamente informado, causando inconveniências caso as alterações prejudiquem essa organização (Moz e Pato; 2007). Com isto, conforme diversos estudos na literatura (Mutingi e Mbohwa, 2017), pode-se dizer que a diferença entre escalas é uma das variáveis mais típicas dentro da função objetivo, onde objetiva-se a sua minimização;

Devido à alta complexidade e tempo computacional inviável demandado para sua resolução (Moz e Pato, 2003), meta-heurísticas, como algoritmos genéticos, ou algoritmos aproximativos são utilizados para resolver problemas NP-Completos, pois apesar de não garantirem o resultado ótimo como em métodos exatos, ainda são capazes de encontrar boas soluções dentro um prazo aceitável (Burke et al., 2004).

Conforme dito anteriormente, o problema de reescalonamento está diretamente relacionado ao problema de escalonamento de enfermeiros, porém, aquele, até o momento, tem recebido pouca atenção da comunidade científica (Clark e Walker, 2011). Dentre fatores que poderiam explicar a menor atenção ao assunto, salienta-se a ausência de uma base de testes com um grande número de instâncias e com diferentes graus de dificuldade para o problema, dificultando uma melhor avaliação de resultados.

1.2 Objetivos e Contribuições

O objetivo geral deste trabalho é a investigação de um algoritmo genético para o problema de reescalonameto de enfermeiros.

Os objetivos específicos são:

- Propor um algoritmo genético de chaves aleatórias combinado ao algoritmo de Hopcroft-Karp (AGCA-HK) para resolução do problema.
- Investigar os resultados encontrados pelo algoritmo proposto, comparando com resultados encontrados por um método exato.
- Propor uma nova base de *benchmark* para o problema em questão.

Dentre as contribuições deste trabalho, ademais de um novo método de resolução para o PRE, utilizando uma técnica computacional nova, este trabalho visa contribuir com outros estudos na área com a disponibilização pública da base criada, em conjunto com os resultados obtidos tanto pelo algoritmo proposto, quanto pelo método exato.

1.3 Motivações e Justificativas

Conforme citado anteriormente, a elaboração de escalas de trabalho dentro do contexto de PRE é uma tarefa custosa e difícil, do ponto de vista prático, impondo uma carga de trabalho considerável para o responsável, e acadêmica, visto que é um problema classificado como NP-Difícil, demonstrando a dificuldade de resolução do mesmo por métodos exatos dentro de prazos aceitáveis. Ainda mais, dada a frequência e dinamicidade do problema, é essencial o desenvolvimento, avaliação e utilização de métodos automatizados eficientes para a resolução do mesmo.

Os estudos na área de PRE presentes na literatura indicam o potencial de utilização de algoritmos heurísticos para a resolução de instâncias do problema em tempos computacionais aceitáveis.

Até o presente momento, não foi encontrado outro estudo na literatura correlata utilizando a técnica computacional proposta neste trabalho, ou realizou um estudo comparativo com um método exato para validação e avaliação de resultados utilizando um número grande de instâncias, demonstrando duas consideráveis lacunas na área de estudo do PRE.

Ainda mais, apesar de fruto necessário para o trabalho proposto, a criação de uma base pública e seus resultados teve efeito positivo e motivador para proposição deste estudo.

1.4 Organização do texto

A organização é descrita a seguir: No capítulo 2, o problema de escalonamento e reescalonamento de enfermeiros é matematicamente definido, além do detalhamento de conceitos básicos e nomenclaturas comumente utilizadas. No capítulo 3, são apresentados revisões literárias acerca de teoria dos grafos e algoritmos evolutivos, introduzindo a base do algoritmo proposto para este estudo. Ao final deste capítulo, uma revisão dos estudos sobre PRE é apresentado. A proposta é formalmente apresentada no capítulo 4, incluindo as características da criação da base de testes (*benchmark*) e do algoritmo genético proposto. Os resultados encontrados por este estudo, em conjunto com sua análise, se encontram no capítulo 5. O capítulo 6 apresenta as conclusões obtidas a partir da análise do algoritmo proposto, assim como algumas ideias para trabalhos futuros.

2

Descrição do problema

Neste capítulo, são introduzidos os conceitos, características e origem do problema de reescalonamento de enfermeiros.

No desenvolvimento deste capítulo, as diferentes características e especialidades de cada problema são apresentadas, resultando então, após a devida contextualização de conceitos, no problema alvo deste trabalho, o problema de reescalonamento de enfermeiros.

2.1 Problema de escalonamento de pessoas

O PEP, geralmente, é um problema de otimização, onde objetiva-se designar determinadas tarefas para funcionários de uma empresa ou instituição. Essas designações são realizadas por um determinado período de tempo, que pode variar desde poucos dias a vários meses. Como é natural de problemas de otimização, a construção dessas escalas de trabalho é sujeita a diversas restrições trabalhistas e organizacionais, de modo que o resultado deve ser uma escala que satisfaça todos os requisitos impostos.

Para definição e diferenciação entre os conceitos apresentados neste trabalho, define-se:

- **Provedor:** No PEP, uma pessoa é um provedor de mão de obra , onde este está apto a realizar tarefas, conforme definição abaixo.
- **Tarefa:** Uma tarefa é definida por um conjunto de ações que são exigidas no escopo do problema. Em um exemplo prático ilustrativo, uma tarefa pode consistir na ação de cuidar de pacientes.
- **Atribuição:** Consiste em incumbir a determinado provedor a realização de uma tarefa.
- **Período:** Define a quantidade de dias que irá compor a escala.

- **Jornada:** Sequência de atribuições realizadas durante o período para um enfermeiro.

- **Escala:** Conjunto de jornadas, que juntas satisfazem todas as restrições do problema.

Este problema está presente nos mais variados ambientes, de modo geral, onde tarefas devem ser cumpridas por funcionários durante um espaço de tempo, este estará presente. A literatura (Bergh *et al*, 2013) divide o problema em categorias, dependendo da característica das restrições. Salientamos três:

- **Escalonamento baseado em projetos:** Neste caso, vários projetos necessitam de uma quantidade de funcionários. O problema reside em designar os funcionários para executar tarefas dentro desses projetos.

- **Escalonamento baseado em demanda flutuante:** O problema aqui se baseia em uma demanda que se altera no tempo. A alocação então é dinâmica, realiza baseada na demanda exigida naquele instante. Ex.: restaurantes e *call centers*.

- **Escalonamento baseado em permanência:** Neste caso podemos especificar os ambientes hospitalares e policiais, onde a demanda e exigência são constantes e são definidos previamente.

Conforme dito anteriormente, esta tarefa ainda é comumente realizada de forma manual e isso impõe diversos custos para a organização. Por ser uma tarefa árdua, um funcionário é designado para essa função, porém, caso esse processo fosse automatizado, este funcionário poderia desempenhar outras funções. Ademais, por possuir diversas soluções, encontrar uma solução menos custosa é tarefa ainda mais difícil, podendo resultar na utilização de uma escala viável, porém de baixa qualidade.

A pesquisa operacional relacionada com PEP está intimamente relacionada com o termo custo, pois, a sua redução, é o objetivo de grande parte das organizações. A redução de custos pode ser obtida em diferentes frentes, todas dependentes da construção de uma escala de qualidade, exemplifica-se:

- **Redução de custo direto:** Utilizando métodos automatizados é possível reduzir a quantidade de horas extras praticadas por um trabalhador ou ainda reduzir o quadro de funcionários, impactando diretamente e positivamente na folha salarial da instituição.

- **Redução de absenteísmo e atrasos:** Insatisfação profissional e fadiga estão dentre as principais causas de absenteísmo, pois os funcionários são sujeitos a maior deterioração de suas saúdes, tornando-se um custo para a instituição.

- **Aumento na qualidade do serviço prestado:** Se o serviço é bem prestado, isso reflete em um ganho para a instituição. O aumento na qualidade de serviço está relacionado com prazer profissional dos funcionários, que, quando bem tratados e satisfeitos, apreciam mais seu ambiente profissional e trabalham de forma mais eficiente.

O PEP é um problema geral que possui suas especificidades, o problema alvo deste trabalho, o problema de reescalonamento de enfermeiros, é um tipo especial deste. Uma melhor compreensão deste alvo é apresentado nas seções a seguir.

2.2 Problema de escalonamento de enfermeiros

Nesta seção são introduzidos os conceitos básicos do problema de escalonamento de enfermeiros. Entender este problema é crucial visto que o PRE é uma modificação deste.

Em um ambiente hospitalar, o serviço deve ser ininterrupto e idealmente de qualidade. Ademais das definições apresentadas para o PEP, dentro deste ambiente existem diversas partes que o definem:

- **Enfermeiro:** No PEE, o trabalhador que realizará as tarefas é um enfermeiro, que pode ser fixo, flutuante ou temporário. Enfermeiros fixos podem trabalhar apenas em um local, flutuantes podem trabalhar em diversos setores ou unidades e os temporários são contratados quando existe algum imprevisto que exija uma maior mão de obra.

- **Turno:** Cada unidade de saúde pode ter diferentes tipos de turnos. Estes turnos tem determinada duração e não são parciais, ou seja considera-se que serão cumpridos integralmente. O horário de início e fim de cada turno também é variável, mas em geral os turnos matutinos se iniciam entre as 6:00 e 9:00, os turnos vespertinos podem ser iniciar logo nas primeiras horas da tarde ou logo antes do entardecer e os turnos noturnos se iniciam nas primeiras horas da noite ou madrugada. Suas durações em geral, encontradas na literatura, são de 6, 8 e 12 horas (Cheang, 2003).

- **Tarefa:** Cada tarefa, extraída da demanda exigida pela organização, deve ser designada para um enfermeiro. Considera-se a tarefa satisfeita quando um enfermeiro está presente naquele turno daquele dia.

O PEE é encontrado na literatura modelado de várias formas, utilizando diferentes restrições e objetivos (Cheang, 2003; Clark, 2015). Não é objeto deste estudo realizar uma análise destas diferentes modelagens, tal análise demandaria fugir demasiadamente do escopo deste trabalho. Porém, para a definição do modelo utilizado neste trabalho, foi realizado um comparativo dos modelos encontrados na literatura, de forma que o modelo

proposto por este estudo seja condizente com o comumente encontrado na literatura (Mutingi e Mbohwa, 2017; Cheang, 2003).

A seguir são apresentadas as restrições comumente utilizadas na literatura de escalonamento de enfermeiros, divididas em duas categorias, rígidas e flexíveis. As rígidas refletem a viabilidade de uma solução, onde o seu descumprimento não é permitido. Em contraste, as flexíveis, são violações indesejáveis, porém permitidas, que são refletidas a um custo na função objetivo do problema.

1. **Restrições Rígidas**

   (a) Cada enfermeiro pode ser designado para apenas um turno de cada dia (Moz e Pato, 2008; Bard e Purmono, 2006; Kitada e Morizawa, 2013, Maenhout e Vanhoucke, 2013; Constantino *et al*, 2013; Constantino *et al*, 2015).

   (b) As demandas operacionais de cada turno de cada setor devem ser satisfeitas (Aickelin e Dowsland, 2004; Moz e Pato, 2008; Kitada e Morizawa, 2013, Maenhout e Vanhoucke, 2013, Constantino *et al*, 2013; Constantino *et al*, 2015).

   (c) Restrições de padrão de designações. Ex: Trabalhar em um turno matutino após um turno vespertino. (Aickelin e Dowsland, 2004; Moz e Pato, 2008; Bard e Purmono, 2006; Kitada e Morizawa, 2013, Maenhout e Vanhoucke, 2013; Constantino *et al*, 2013).

   (d) Limite de atribuições durante o período (Aickelin e Dowsland, 2004; Moz e Pato, 2008; Bard e Purmono, 2006; Kitada e Morizawa, 2013, Maenhout e Vanhoucke, 2013; Constantino *et al*, 2015).

   (e) Qualidade quantitativa, medida através das capacidades de cada enfermeiro, deve ser superior ao exigido (Kitada e Morizawa, 2010).

   (f) Nenhum funcionário deve trabalhar durante sete dias consecutivos (Moz e Pato, 2008; Baumelt, 2016).

   (g) Número máximo de designações consecutivas para um tipo de turno (Aickelin e Dowsland, 2004; Maenhout e Vanhoucke, 2011; Constantino *et al*, 2015).

   (h) Todo funcionário deve ter descanso semanal de no mínimo dois dias (Clark e Walker, 2011).

   (i) Um funcionário não poderá trabalhar em determinados turnos ou dias em casos de ausência previamente notificada ou devido a restrições contratuais (Moz e Pato, 2008).

2. **Restrições Flexíveis**

   (a) As demandas operacionais de cada turno de cada setor devem ser satisfeitas (Bard e Purmono, 2006; Maenhout e Vanhoucke, 2013).

(b) Padrões de trabalho indesejáveis são penalizados, mas permitidos (Bard e Purmono, 2006; Maenhout e Vanhoucke, 2011).

(c) Alguns enfermeiros não poderão trabalhar no turno da noite consecutivamente (Moz e Pato, 2008).

(d) A solicitação de folga em determinados turnos pode ser descumprida com determinado custo associado (Dowsland e Thompsom, 2000).

3. **Objetivos típicos**

(a) Minimizar a diferença entre a quantidade de enfermeiros designada para o turno e sua exigência (Clark e Walker, 2011).

(b) Minimizar o uso de enfermeiros externos (Bard e Purmono, 2006).

(c) A relação entre funcionários é quantificada e consta como objetivo de otimização (Clark e Walker, 2011).

(d) A preferência dos enfermeiros por determinados padrões de designações é quantificada e objetiva-se sua otimização (Dowsland e Thompsom, 2000; Gutjarh e Rauner, 2007; Moz e Pato, 2008).

(e) Minimizar a discrepância entre as preferências dos enfermeiros (Constantino *et al*, 2015).

A lista acima, apesar de superficial e apresentar apenas as restrições mais comuns, deixa clara a alta variabilidade de modelos para o PEE. É comum autores utilizarem um mesmo conjunto de restrições, como no caso dos quatro trabalhos de Margarida Moz e Margarida Pato (2003, 2004, 2007, 2008). Nota-se também a inexistência de trabalhos de diferentes autores que compartilham um mesmo conjunto de restrições, dificultando a identificação das restrições mais adequadas para o problema, ressalvando que esta característica é comum devido a alta diversidade de regulamentações legais, contratuais e também é influenciada por aspectos institucionais (Clark *et al*, 2013).

Estas escalas, porém, por serem realizadas previamente, não possuem garantia de cumprimento, afinal seres humanos estão sujeitos aos mais variados problemas, como adoecimentos e complicações familiares. Quando essa escala não pode ser cumprida devido a ausência de um funcionário, exige-se então um reescalonamento, sendo este definido na seção a seguir.

2.3 Problema de reescalonamento de enfermeiros

2.3.1 Conceitos e definições

O problema de reescalonamento de enfermeiros, consiste em redesignar tarefas para uma quantidade definida de enfermeiros. As tarefas a serem designadas estão distribuídas

dentro um espaço de tempo também definido a *priori*. Essas designações, porém, devem ser feitas levando em consideração as novas restrições inerentes ao problema.

Em adição aos conceitos apresentados anteriormente, no contexto de reescalonamento, surgem ao menos duas novas definições:

- **Ausência:** É definido pela impossibilidade de um enfermeiro comparecer ao seu local de trabalho durante um ou mais dias.
- **Remanejo:** Pela não satisfação de requisitos obrigatórios, alterações devem ser feitas na escala original, um remanejo ou diferença acontece quando um enfermeiro trabalha, na nova escala, em um setor ou turno diferente do originalmente designado.

Estes dois novos conceitos dão origem a duas novas restrições, que utilizadas em conjunto com as restrições apresentadas na seção anterior, completam o conjunto de restrições do PRE:

1. Restrições Rígidas
   (a) O enfermeiro deverá ser obrigatoriamente designado para o turno livre no dia que estiver ausente.
2. Restrições Flexíveis
   (a) Os enfermeiros devem preferencialmente trabalhar no mesmo turno e setor, nos quais foram previamente alocados.

Em contraste com as restrições utilizadas no PEE, as restrições acima são utilizadas de forma geral nos estudos na área de reescalonamento, havendo pouca ou nenhuma diferença na sua utilização (Mutingi e Mbohwa, 2017).

Para exemplificar como um reescalonamento pode ser efetuado, uma análise pode ser feita utilizando a Tabela 2.1. Na porção esquerda da tabela a escala original é exibida, criada a partir da resolução do problema de escalonamento de enfermeiros, onde as três ultimas linhas apresentam a demanda operacional de cada dia e turno. Porém, pela ausência do enfermeiro 4 no sexto dia, destacada em vermelho, verifica-se que a demanda do turno da manhã não é satisfeita, portanto deixa de ser uma solução viável. A porção direita da mesma tabela, apresenta um possível reescalonamento para a escala original, satisfazendo todas as restrições do problema e realizando apenas três alterações, assinaladas em verde.

Tabela 2.1: Exemplo de escalonamento e reescalonamento, onde a designação problemática está destacada em vermelho na escala original, e um possível reescalonamento com três alterações em verde.

	Escala Original							Reescalonamento						
	Dias							Dias						
Enfermeiro	1	2	3	4	5	6	7	1	2	3	4	5	6	7
E-1	M	L	T	N	L	T	N	M	L	T	N	L	T	N
E-2	M	M	M	T	T	L	M	M	M	M	T	T	L	M
E-3	T	N	N	L	M	N	N	T	N	N	L	M	**M**	N
E-4	N	L	T	L	L	**M**	M	N	L	T	L	L	**L**	M
E-5	L	N	L	M	M	T	T	L	N	L	M	M	**N**	T
M (Manhã) - T (Tarde) - N (Noite) - L (Livre)														
	Demanda operacional													
	Dias							Dias						
Período	1	2	3	4	5	6	7	1	2	3	4	5	6	7
Manhã	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Tarde	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Noite	1	1	1	1	1	1	1	1	1	1	1	1	1	1

3

Revisão Literária

Neste capítulo, os diferentes conceitos, técnicas e teorias relacionadas a este estudo são apresentados, utilizando como base, estudos e conhecimento produzidos anteriormente. As informações apresentadas neste capítulo são essenciais para o entendimento das técnicas resolutivas propostas por este estudo. Objetiva-se, então, a introdução de conceitos e preparação para a apresentação das técnicas propostas no capítulo seguinte.

3.1 Problemas de otimização

Nas áreas da matemática, ciência da computação e pesquisa operacional, um problema de otimização ou otimização matemática pode ser entendido como a busca de uma resposta contida dentro de um conjunto, onde se objetiva encontrar a resposta mais adequada.

Em casos mais simples, esta otimização consiste na maximização ou minimização de uma única variável, em casos mais complexos a combinação de diferentes valores é buscada de forma a melhor responder a questão inicial. Além da quantidade de variáveis, deve-se também considerar as restrições impostas para os valores buscados, assim como o domínio imposto para essas variáveis.

O problema de otimização, apresentado pela equação 3.1, realiza a seguinte pergunta: "Quais os valores de x e y, distintos, tais que, somados, resultam no maior valor possível?"

$$\begin{aligned} &\text{maximize } f(x, y) \\ &\quad \text{subject to} \end{aligned} \tag{3.1}$$

$$f(x, y) = x + y; \tag{3.2}$$

$$x \neq y; \tag{3.3}$$

$$x,\ y \in \mathbb{N}; \tag{3.4}$$

$$x,\ y \leq 10; \tag{3.5}$$

A equação 3.1 apresenta a categoria do problema, de maximização, e apresenta qual função deverá ser maximizada, *f(x,y)*. A pergunta é feita pela definição da função, na equação 3.2, enquanto as restrições e domínio das variáveis x e y, são introduzidas pelas equações 3.3, 3.4 e 3.5. Para este problema, tem-se duas respostas ótimas: x=10, y=9 e x=9, y=10, onde o resultado para ambos os casos é 19.

Utilizando o exemplo acima, os principais conceitos de problemas de otimização são apresentados:

- Tipo de problema
- Variáveis
- Função objetivo
- Restrições
- Espaço de busca
- Soluções candidatas
- Soluções viáveis
- Soluções inviáveis

No exemplo apresentado anteriormente, o tipo de problema é de maximização, onde as variáveis são x e y e a função objetivo, também chamada de função de custo, é definida no corpo do modelo, em conjunto com suas restrições. Estas restrições podem ser categorizadas como rígidas ou flexíveis, a depender da sua obrigatoriedade. Caso uma restrição seja mandatória, a mesma será classificada como rígida, caso contrário, sendo apenas uma restrição indesejável, será classificada como flexível, sendo passível de penalização. O espaço de busca (Monmarché *et al*, 2000) do problema apresentado é uma porção do espaço euclidiano $\mathbb{N}^2$. Dentro do espaço de busca, temos um conjunto de soluções, esse conjunto completo compõe o conjunto de soluções candidatas. O conjunto de soluções candidatas que satisfazem todas as restrições definidas pelo modelo é chamado de conjunto de soluções viáveis, quando não, definem o conjunto de soluções inviáveis (Singh, 2008).

Para exemplificar:

- (x=*10*, y=8) é uma solução candidata e viável.
- (x=*10*, y=10) é uma solução candidata e inviável.
- (x=*11*, y=11) é uma solução não candidata, pois está fora do espaço de busca e inviável.

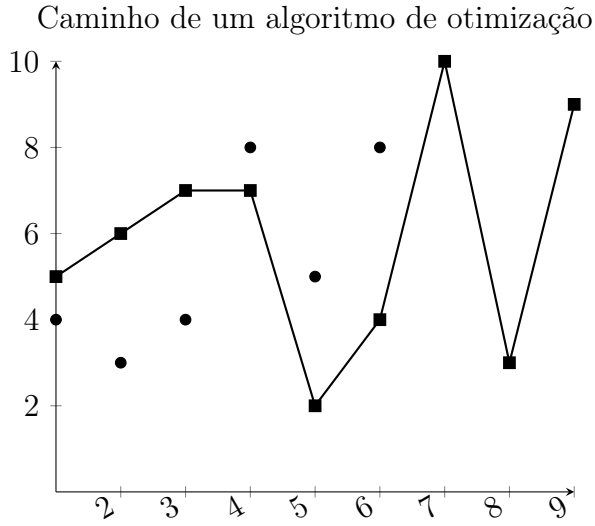


Figura 3.1: Imagem ilustrativa de um algoritmo de busca e a as soluções encontradas no processo

Dentro do espaço de busca alguns conceitos também são primordiais para compreender a busca de soluções. Na figura 3.1, todos os pontos fazem parte do espaço de busca, porém, os pontos conectados representam o "caminho"do algoritmo na busca de seu objetivo. Nesta mesma figura, algumas definições comumente utilizadas no escopo de otimização e espaço de busca são exemplificadas:

Mínimo local: Este ponto é exemplificado pelo par (8, 3), pois representa o menor valor, dadas as soluções em sua proximidade.

Máximo local: Este ponto é exemplificado pelo par (9, 9), pois representa o maior valor, dadas as soluções em sua proximidade.

Mínimo global: Este ponto é exemplificado pelo par (5, 2), pois representa o menor valor de todo espaço de busca

Máximo global: Este ponto é exemplificado pelo par (7, 10), pois representa o maior valor de todo espaço de busca.

Platô: Esta definição pode ser exemplificada pela linha entre os pares (3,7) e (4,7), identificando uma manutenção do valor Y, ou seja, sem melhora, seja para um problema de maximização ou de minimização.

Em algoritmos de busca, o objetivo de se encontrar o máximo ou mínimo global é dificultado pela presença de máximos e mínimos locais e platôs, pois estes, em geral introduzem obstáculos na busca por melhores soluções.

Problemas de otimização também podem ser classificados de acordo com sua função objetivo. Quando esta função objetivo é composta por múltiplas variáveis, diz-se que esta função é multi-objetivo, conforme exemplos a seguir.

- Função de um único objetivo: Ex.:$f(x) = x$
- Função bi-objetivo: Ex.: $f(x, y) = x \times y$

3.2 Teoria dos grafos

A teoria dos grafos é um ramo de estudo da matemática onde se estuda a relação de elementos de um conjunto. Neste contexto, os conceitos e técnicas utilizadas por este trabalho são apresentados nesta seção.

3.2.1 Conceitos e definições

Um grafo G é definido por uma tupla (V, E), onde V representa um conjunto de vértices e E um conjunto de arestas. Analogamente, os vértices de um grafo são representações de elementos, enquanto arestas representam a relação entre estes.

Os elementos básicos de um grafo são assim definidos:

- **Grafo:** Definido pela tupla (V, E), onde ambos V e E são conjuntos, sendo V um conjunto não vazio.
- **Vértice:** Representação de um elemento de determinado conjunto.
- **Aresta:** Representação de uma relação entre vértices.
  - Arestas Direcionadas: Quando estas arestas possuem direção.
  - Arestas Não-Direcionadas: Quando estas arestas possuem direção dupla, ou seja, são em ambos os sentidos.
- **Incidência:** Se uma aresta conecta dois vértices, então ambos os vértices são considerados incidentes à aresta.
- **Adjacência:** Se existe uma aresta conectando dois vértices, então ambos são considerados adjacentes um ao outro.
- **Grau:** A quantidade de arestas que são incidentes à um vértice. Considerando arestas direcionadas, pode-se diferenciar grau de entrada e grau de saída.
- **Fonte:** Um vértice é uma fonte se possui grau de entrada igual a 0.
- **Sumidouro:** Um vértice é um sumidouro se possui grau de saída igual a 0.
- **Ordem:** A quantidade de vértices de um grafo.
- **Caminho:** Sequência de arestas que conectam um vértice A a outro vértice B, caso seja um grafo direcionado, a direção das arestas deverá ser considerada.

A Figura 3.1, ilustra a representação de um grafo G(V, E), sendo estes:

- V = {0, 1, 2, 3, 4, 5}
- E = {(0, 2, 2), (0, 3, 1), (1, 2, 5), (2, 4, 5), (2, 5, 11), (5, 4, 7)}

O conjunto de arestas acima, representado por uma lista de adjacência, tem a seguinte explanação: o primeiro elemento da tupla, representa o início da aresta, o segundo, seu fim, e o terceiro valor representa o custo associado a essa associação. Esse conjunto pode ainda ser representado por matrizes de incidência e adjacência, apresentados na Tabela 2.3, e sua visualização na Figura 3.1.

Tabela 3.1: Representação de um grafo por matrizes de adjacência e incidência

	Matriz de adjacência						Matriz de incidência					
-	0	1	2	3	4	5	(0, 2, 2)	(0, 3, 1)	(1, 2, 5)	(2, 4, 5)	(2, 5, 11)	(5, 4, 7)
0	0	0	2	1	0	0	+1	+1	0	0	0	0
1	0	0	5	0	0	0	0	0	+1	0	0	0
2	0	0	0	0	5	11	-1	0	-1	+1	+1	0
3	0	0	0	0	0	0	0	-1	0	0	0	0
4	0	0	0	0	0	0	0	0	0	-1	0	-1
5	0	0	0	0	7	0	0	0	0	0	-1	+1

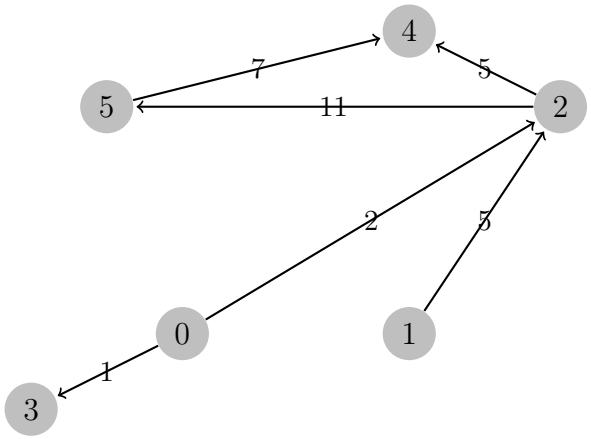


Figura 3.2: Exemplo de grafo com arestas direcionadas com pesos

A utilização de grafos é ampla na literatura de problemas de otimização, seja na forma de modelagem ou no auxílio na resolução de problemas computacionais (Deo, 2017). As possibilidades de utilização de grafos são simplesmente incontáveis, elementos e relações são convertidos em grafos, onde podem ser processados por algoritmos específicos. A Tabela 3.2 apresenta alguns exemplos de utilização de grafos para problemas reais.

Dentre as diversas categorias de grafos, uma em particular é de interesse deste trabalho:

Grafos bipartidos: Um grafo é dito bipartido se, e somente se, o seu conjunto de vértices pode ser particionado em dois conjuntos V1 e V2, tal que todas as arestas

Tabela 3.2: Exemplos de utilização de grafos

Problema	Vértice	Aresta
Redes de computadores	Computadores	Cabos
Transporte	Cidades	Rodovivias
Jogos de tabuleiro	Posições no tabuleiro	Possiveis movimentos
Relações pessoais	Pessoas	Amizade, Relacionamento, etc.
Escalonamento	Tarefas ou Funcionários	Restrições ou Permissões

deste grafo possuam início e fim em conjuntos distintos. A Figura 3.2 ilustra um grafo bipartido.

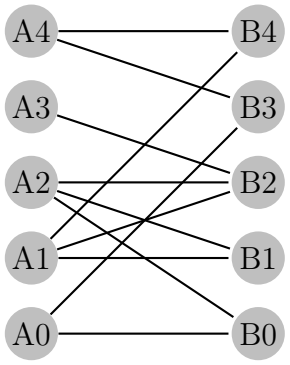


Figura 3.3: Exemplo de grafo bipartido

3.2.2 Algoritmos em grafos

Existem algoritmos específicos para serem utilizados em grafos, esses algoritmos auxiliam a resolver problemas simples e comuns, possuindo a grande vantagem de executarem em tempo polinomial. Dada a vasta quantidade de algoritmos existentes, este trabalho introduzirá apenas os algoritmos relevantes em seu propósito.

- **Busca em largura** - (*BFS - breath first search*)
- **Busca em profundidade** - (*DFS - depth first search*)

O algoritmo de busca em largura é um algoritmo de busca não informada, onde o objetivo é visitar todos os vértices de um grafo. O algoritmo sistematicamente visita os vértices, finalizando sua execução quando não existem mais vértices não visitados. Cada iteração do algoritmo, que utiliza uma estrutura de fila em sua busca, processa um vértice, onde os adjacentes a este são considerados para entrarem na fila caso ainda foram visitados. O algoritmo BFS executa em tempo $O(|V| + |E|)$, segundo a notação Bachmann-Landau, também chamada de notação assintótica.

O algoritmo DFS é similar ao BFS, diferenciando-se, em uma implementação não recursiva, na estrutura de dados utilizada. No BFS, a estrutura utilizada é uma fila, que

segue o padrão FIFO (*first in first out*), enquanto no DFS utiliza-se uma pilha que é do tipo FILO (*first in last out*).

O Algoritmos 1 apresenta o pseudo-código de ambos os algoritmos.

Algoritmo 1 Algoritmo *BFS e DFS*

```
BREATH-FIRST-SEARCH(G, r)
 1  F → Fila de vértices = ∅
 2  V → Conjunto de vértices de G
 3  Coloque r em F
 4  Assinale r como visitado
 5  repeat
 6        v ← Primeiro elemento de F
 7        repeat
 8              w ← vizinho de v
 9              if  w não foi visitado
                    BFS → enfileirar(w)
                    DFS → empilhar(w)
                    assinale w como visitado
10        until (v possuir vizinhos)
11  until (F = ∅)
```

3.2.3 Problema de emparelhamento máximo

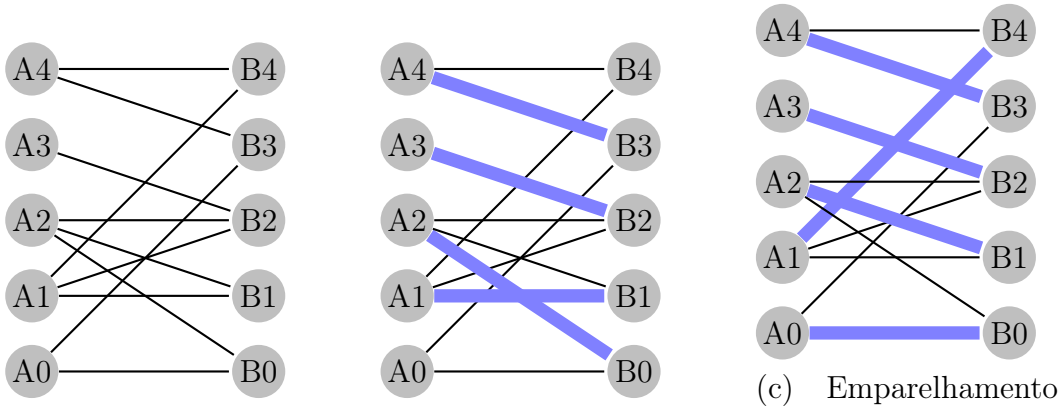
Um emparelhamento em um grafo G consiste em um conjunto de arestas A tal que, todo vértice de G seja incidente em no máximo um elemento deste conjunto. Similarmente, pode-se definir estas arestas como independentes tomadas duas a duas, formando um conjunto estável de vértices. Em grafos bipartidos esta relação é ainda mais óbvia, onde as arestas do conjunto A interligam vértices de um conjunto V1 e V2, independentes entre si.

Encontrar um emparelhamento máximo M consiste em buscar um emparelhamento, ou seja, um conjunto de arestas, tal que, não existe um emparelhamento M', com $\|M'\| > \|M\|$. Um emparelhamento perfeito ocorre quando todos os vértices do grafo são saturados, porém nem todos os grafos possuem esta característica.

Analogamente ao problema alvo deste trabalho, posicionando trabalhadores como vértices em um conjunto V1 e tarefas em um conjunto V2, um emparelhamento máximo idealmente satisfaria todas as tarefas, não sendo necessário um emparelhamento perfeito, pois existem mais enfermeiros do que tarefas a serem realizadas.

A Figura 3.4 apresenta exemplos de grafos bipartidos, emparelhamento não máximo e máximo.

Diretamente relacionados com emparelhamentos, estão os conceitos de caminhos alternantes e caminhos aumentantes, que são assim definidos:



(a) Grafo bipartido (b) Emparelhamento (c) Emparelhamento máximo

Figura 3.4: Exemplo de grafo bipartido (a), emparelhamento (b) e emparelhamento máximo (c)

Caminho alternantes: Um caminho alternante em relação a um emparelhamento A consiste em encontrar um caminho B, iniciando-se em um vértice não saturado, onde as arestas deste caminho estão, alternadamente, dentro e fora de A.

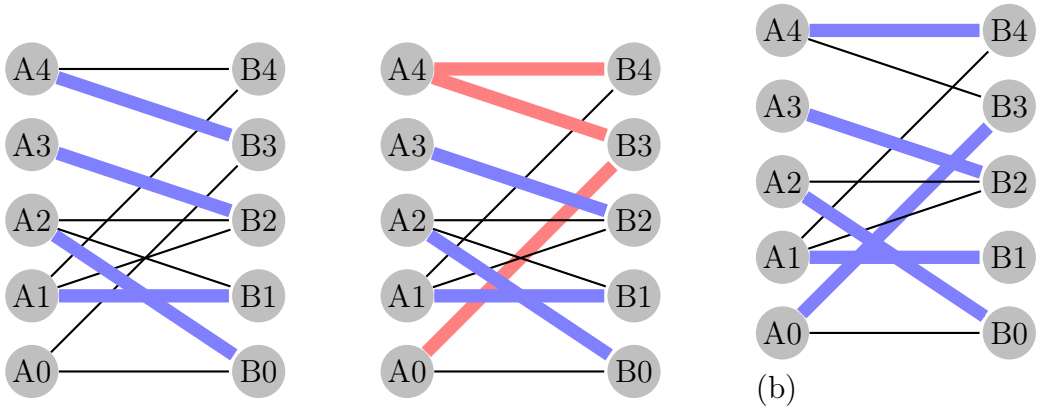
Caminho aumentante: Um caminho alternante e aumentante em relação a A termina em um vértice não saturado.

Vértice livre ou não saturado: É um vértice que não pertence ao emparelhamento.

Aresta acoplada e não acoplada: São arestas que pertencem e não pertencem ao emparelhamento, respectivamente.

Um caminho aumentante inicia em um vértice não saturado e alternadamente utiliza arestas dentro e fora do emparelhamento atual e finaliza seu trajeto em um vértice também não saturado. A utilização deste conceito em grafos bipartidos fica ainda mais clara, onde um caminho aumentante se inicia em um vértice não saturado do grupo V1 e termina seu trajeto em outro vértice não saturado do grupo V2, lembrando que em grafos bipartidos os vértices podem sem divididos em dois grupos distintos, onde esses grupos não possuem arestas entre si. Conforme exemplos na Figura 3.5.

O algoritmo de Hopcroft-Karp concebido por John Hopcroft e Richar Karp em 1973, é um algoritmo que recebe um grafo bipartido como entrada e retorna um emparelhamento máximo deste grafo. Dentre outros algoritmos para busca de emparelhamentos máximos, a semelhança se encontra no conceito de caminhos aumentantes, porém a vantagem deste algoritmo sobre os demais incorre na sua eficiência. Ao invés de encontrar um único caminho aumentante a cada iteração sobre os vértices, o algoritmo busca um conjunto máximo de caminhos aumentantes de comprimento menor, resultando, em termos de complexidade, em um tempo de execução $O\sqrt{|V|}$, superior assintoticamente ao tempo de execução baseado no algoritmo de Fork-Fulkerson, que é $O|V|$ (Hopcroft e Karp, 1973; Cordella *et al*, 2001).



(a) Emparelhamento (b) Caminho aumentante (b) Após caminho aumentante

Figura 3.5: Exemplo de emparelhamento (a), caminho aumentante (b) e emparelhamento máximo (c)

Utilizando o conceito de caminhos aumentantes o algoritmo HK segue o conceito a seguir:

- Um emparelhamento M **não** é maximo se existe um caminho aumentante em relação a este. O inverso também é verdade.

Utilizando o algoritmo BFS para encontrar caminhos aumentantes e DFS para calcular o comprimento do caminho, o pseudocódigo geral é apresentado pelo Algoritmo 2.

Algoritmo 2 *Hopcrcoft-Karp*

```
HOPCROFT-KARP()
1   M → Emparelhamento máximo
2   repeat
3       ea → Emparelhamento atual
4       ca ←aug
5       repeat
6           aresta ← primeiro elemento de ca
7           if aresta é uma aresta acoplada
                retire aresta do emparelhamento atual
8           if aresta é uma aresta não acoplada
                adicione aresta ao emparelhamento atual
9       until (Todas as arestas de ca serem processadas)
10  until (Existir caminho aumentante aug)
```

3.3 Algoritmos Evolutivos

Dentre os diversos métodos de resolução de problemas de otimização, um subconjunto deste é composto de algoritmos evolutivos, que são inspirados em conceitos de IA (Fogel,

1997). Ademais deste, outros métodos como redes neurais, métodos *fuzzy* são populares na literatura. A vantagem daquele sobre outros métodos reside na necessidade da inserção de apenas uma porção de conhecimento sobre o problema no algoritmo, tendo este o potencial de ser utilizado em uma ampla gama de problemas.

Conforme estudos (Fogel, 1997; Fogel, 2005), algumas vantagens e características de algoritmos evolutivos são destacadas:

- AEs são algoritmos quantitativos, sendo portanto, passíveis de otimizações em seus parâmetros.
- Também são capazes de implementar restrições que não são possíveis em outros métodos, demonstrando robustez e adquirindo um balanceamento entre eficácia e eficiência.
- Estes algoritmos também são capazes de se combinar com outras técnicas, resultando em algoritmos meméticos, sendo possível escalar de um algoritmo evolutivo puro para um algoritmo evolutivo híbrido.
- Possuem a capacidade de serem utilizados em problemas multi-objetivo.

AEs são algoritmos de descida de gradiente estocástica derivados da clássica teoria evolutiva de Darwin (1859). Os princípio básicos de algoritmos evolutivos são:

Função *fitness*: Alvo do algoritmo evolutivo, análoga a função de custo definida anteriormente.

População: Conjunto de soluções.

Sobrevivência: Realizando a seleção de certa porção de indivíduos da população, espera-se a convergência, forçada por boas soluções, da população rumo a ainda melhores soluções, enquanto outros indivíduos da população são descartados, seguindo o conceito de sobrevivência do mais apto.

As soluções, também chamadas de indivíduos, dentro de uma população são uma representação de uma resposta para o problema. Todas as soluções são avaliadas e recebem um valor que determina a sua qualidade, chamada de *fitness*. Um *framework* genérico de um algoritmo evolutivo é compostos dos seguintes passos:

1. Crie a população:
    - Aleatória: Soluções são geradas sem conhecimento do problema.
    - Construtiva: Soluções são geradas baseadas em alguma heurística construtiva, utilizando algum conhecimento do problema.

2. Selecione indivíduos, chamados de pais, da população e produza novas soluções a partir da combinação destes.

3. Repita os passos 1 e 2 até que um critério de parada seja satisfeito.

Uma importante propriedade de algoritmos evolutivos está no seu processo de busca, tratando-se de um algoritmo populacional, este é capaz de superar, por exemplo, um algoritmo baseado em uma única solução, como o *Hill-Climbing*, sendo mais resistente inclusive a convergência prematura do que este. No entanto, algoritmos evolutivos não possuem a garantia de otimalidade como a de métodos exatos, como PI e MILP.

Algoritmos evolutivos se utilizam de forças distintas durante sua busca:

- Exploração: Nesta força busca-se encontrar regiões promissoras no espaço de busca.
- Aproveitamento: Aqui reside o potencial de otimização mais acentuada das soluções.

Na seção a seguir, os conceitos e definições de algoritmos genéticos, uma das classes de algoritmos evolutivos, é apresentada.

3.3.1 Algoritmos Genéticos

Algoritmos genéticos (Holland, 1975) são um dos mapeamentos mais simples do processo evolucionário para um sistema computacional. A utilização de termos relacionados a genética biológica é comum, por exemplo, o termo cromossomo, que nada mais é do que a denominação de uma solução e sua representação computacional.

Algoritmos genéticos clássicos se utilizam de três mecanismos genéricos:

- Seleção
- Cruzamento ou *crossover*
- Mutação

O processo de seleção consiste na escolha de indivíduos dentro da população. A forma com que este mecanismo é implementada impacta diretamente na pressão de seleção, onde uma menor pressão busca um maior potencial exploratório e uma maior resulta em uma convergência mais acentuada.

O mecanismo de seleção pode ser aplicado de diversas formas, inclusive de forma dinâmica, alterando o seu mecanismo baseado no *feedback* do algoritmo, destacam-se:

Aleatória: Soluções são escolhidas aleatoriamente. Possuem capacidade exploratória, porém carecem de pressão de seleção.

Elitista: Dadas as funções *fitness* das soluções, as melhores soluções serão escolhidas. Possui capacidade de convergência, porém, pode resultar em convergência prematura.

Roleta: As soluções possuem probabilidade *pr* de serem escolhidas, baseado na proporção do *fitness* calculado da população.

Torneio: Uma quantidade *k* de soluções disputa entre si, de forma que a melhor solução é selecionada. Com *k=1*, tem-se o método aleatório, enquanto com *k=tamanho da população*, a melhor solução é escolhida. É comum na literatura a utilização do torneio binário (*k=2*).

É importante ressaltar que não existe consenso no quesito de escolha de um método específico seleção, todas possuem vantagens e desvantagens, posicionando a responsabilidade de controle da busca para o pesquisador em sua busca por robustez. Essa afirmação também é valida para os demais operadores a seguir.

O operador de cruzamento consiste na geração de novos indivíduos, chamados de descendentes, a partir da combinação de dois pais selecionados pelo operador anterior. Diga-se que o cromossomo de uma solução é representado por um vetor de *bits*, 0 e 1, um descendente terá parte do vetor de um dos pais e parte do outro. Como é comum em algoritmos genéticos, existem diversas formas de se implementar o cruzamento, destacam-se algumas, considerando que dois descendentes serão criados a cada aplicação do operador:

Um ponto: O vetor é dividido em duas partes, não necessariamente iguais, onde uma porção irá para um descendente e a outra para outro filho.

Dois pontos: Similar ao caso anterior, porém o vetor é divido em três partes.

Uniforme: O vetor é divido em N partes, onde N é o tamanho do vetor.

Religação de caminho: É possível realizar o caminho de uma solução escolhida a outra, onde a cada modificação as soluções são mais similares. Todas essas modificações resultam em descendentes candidatos, originados a partir da combinação de seus pais. Com isto, é possível encontrar a melhor combinação dos pais selecionados, porém, impacta diretamente na eficiência do algoritmo por ser um processo custoso computacionalmente.

O último operador de um algoritmo genético clássico é o de mutação. A mutação consiste em dada probabilidade *pm*, a solução terá parte de seu cromossomo alterada aleatoriamente. Está é uma definição genérica, abrindo possibilidades para diferentes tipos de mutação, porém, dentre as diversas possibilidades, deve-se destacar que a probabilidade de mutação em casos gerais deve ser baixa, pois utilizando taxas altas de mutação o algoritmo terá o comportamento de uma busca aleatória. Porém, ao se utilizar taxas muito baixas, um dos mecanismos inerentes a AG de escape de ótimos locais será afetado, visto que no algoritmo genético clássico, a convergência baseada na sobrevivência do mais apto se contrasta diretamente com sua capacidade de exploração de novos locais de busca.

A Figura 3.6 mostra o fluxograma de um algoritmo genético clássico, com substituição populacional.

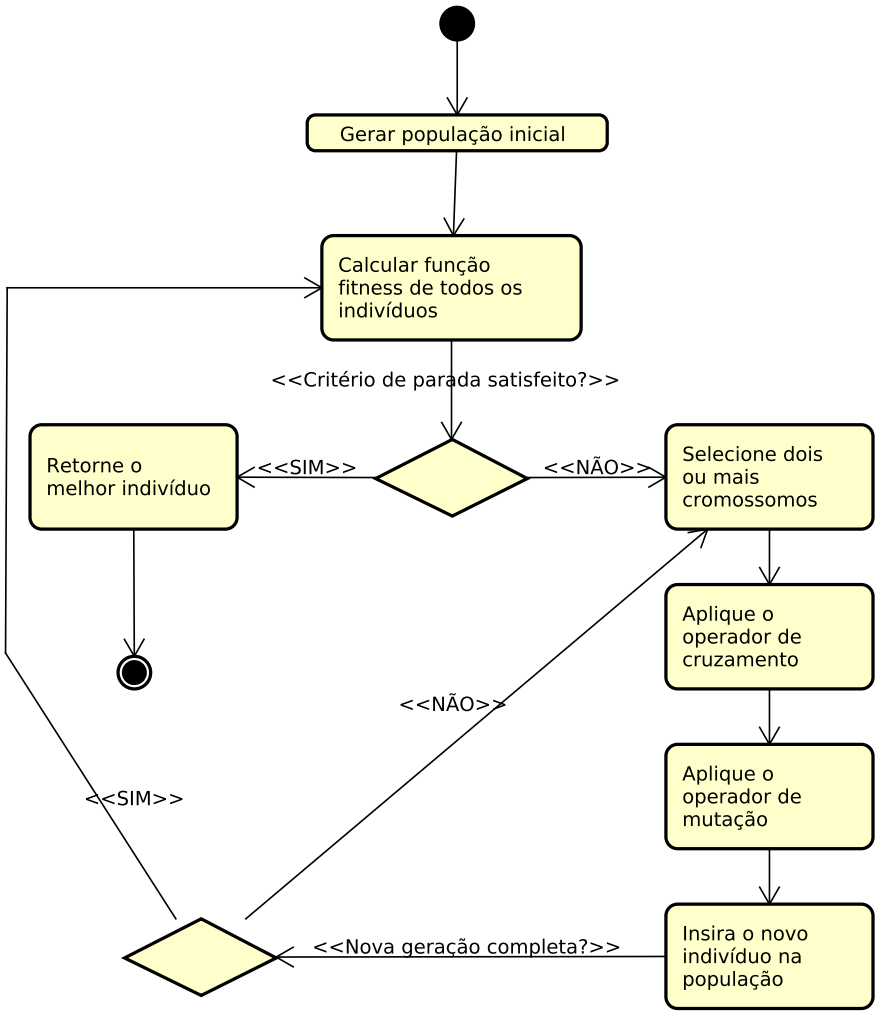


Figura 3.6: *Framework* de um algoritmo genético clássico

3.3.2 Algoritmos Genéticos de chaves aleatórias

Algoritmos genéticos de chaves aleatórias, doravante chamados de RKGAs (*Random Key Genetic Algorithms*), em muitos aspectos, são similares a algoritmos genéticos clássicos, mantendo seus operadores, afinal aqueles são uma classe de algoritmos genéticos.

Enquanto a codificação de cromossomos em algoritmos genéticos é realizada de forma direta, ou seja, em uma fidedigna representação da solução, RKGAs se utilizam de vetores, cujos valores estão no intervalo [0, 1[. Um processo, chamado decodificação, processa o vetor codificado criando então a real solução para o problema. Por utilizar um processo de decodificação, RKGAs são considerados buscadores indiretos, pois buscam soluções dentro do espaço de busca indiretamente, através dos vetores codificados.

O cruzamento entre indivíduos em RKGAs ocorrem utilizando os vetores codificados, pois caso o vetor decodificado seja utilizado, a perda de viabilidade poderia ocorrer. O cruzamento uniforme é o mais comumente utilizado em implementações de RKGAs, seu funcionamento é baseado em uma probabilidade p de transferência de informação de um pai para seu descendente. Esta probabilidade pode influenciar diretamente a convergência do algoritmo, exemplifica-se:

Elitismo: Em uma abordagem elitista, onde o primeiro pai é uma solução de boa qualidade, utilizar uma probabilidade $p \geq 0.6$, resultará no melhor aproveitamento das característica do pai elite.

Igualitarismo: Utilizando um probabilidade p=0.5, resultará em uma utilização igualitária de ambas as características dos indivíduos selecionados para cruzamento.

Note que este é mais um parâmetro passível de otimização, podendo ainda ser alterado dinamicamente, sendo possível um controle sobre o elitismo da transferência de informação. Utilizando esta probabilidade *pc*, um exemplo elitista pode ser assim exemplificado:

- *pc=0.7*
- Seja o Pai 1, o indivíduo elite.
- Seja o Pai 2, o indivíduo não elite.
- Transfira informação Pai 1 $\rightarrow$ Filho 1 e Pai 2 $\rightarrow$ Filho 2 com probabilide *pc*
- Transfira informação Pai 2 $\rightarrow$ Filho 1 e Pai 1 $\rightarrow$ Filho 2 com probabilide *1-pc*

Quando propostos, estes algoritmos (Bean, 1994), os pesquisadores evidenciaram o potencial de RKGAs em diversas áreas, com especial atenção nos casos onde o operador de cruzamento poderia resultar em soluções inviáveis. Utilizando um decodificador simples, o de ordenação, foi possível manter a viabilidade das soluções descendentes, eliminando o problema gerado por algoritmos genéticos com operadores clássicos. A Tabela 3.3 ilustra a utilização de vetores de chaves aleatórias em um problema simples. Com um possível cruzamento entre indivíduos sendo mostrado na Tabela 3.4.

Dadas 10 cidades, deve-se visita-las uma única vez de forma a minimizar a distância percorrida. As distâncias entre as cidades foram omitidas, bastando imaginar que cada trajeto entre dois pares de cidades têm suas distâncias definidas em uma tabela.

As características da modelagem são:

- As cidades são identificadas por índices de 0 a 9, associadas a uma chave aleatória, conforme definido acima.

- O processo de decodificação ocorre por ordenação crescente de valor da chave.

Nota-se que após a decodificação, uma ordem específica de visitação é definida, esta então passará por uma avaliação de qualidade. Mais importante ainda é a manutenção de viabilidade após o cruzamento, utilizando dois pontos, apresentado na Tabela 3.4. Percebe-se que não existem cidades repetidas, sendo então uma solução viável.

Tabela 3.3: Exemplo de indivíduos com aplicação de decodificação

	Indivíduo 1									
Cidade	0	1	2	3	4	5	6	7	8	9
Chave	0,1	0,4	0,2	0,3	0,3	0,6	0,5	0,8	0,7	0,3
	Indivíduo 2									
Cidade	0	1	2	3	4	5	6	7	8	9
Chave	0,9	0,2	0,1	0,3	0,4	0,6	0,7	0,2	0,3	0,4
	Após decodificação por ordem crescente									
	Indivíduo 1 - Decodificado									
Cidade	0	2	3	4	9	1	6	5	8	7
Chave	0,1	0,2	0,3	0,3	0,3	0,4	0,5	0,6	0,7	0,8
	Indivíduo 2 - Decodificado									
Cidade	2	1	7	3	8	4	9	5	6	0
Chave	0,1	0,2	0,2	0,3	0,3	0,4	0,4	0,6	0,7	0,9

Tabela 3.4: Exemplo de cruzamento entre dois indivíduos com geração de dois descendentes, utilizando cruzamento em dois pontos

	Indivíduo 1									
Cidade	0	1	2	3	4	5	6	7	8	9
Chave	0,1	0,4	0,2	0,3	0,3	0,6	0,5	0,8	0,7	0,3
	Indivíduo 2									
Cidade	0	1	2	3	4	5	6	7	8	9
Chave	0,9	0,2	0,1	0,3	0,4	0,6	0,7	0,2	0,3	0,4
	Após cruzamento									
	Descendente 1									
Cidade	0	1	2	3	4	5	6	7	8	9
Chave	0,1	0,4	0,2	0,3	0,4	0,6	0,7	0,8	0,7	0,3
	Indivíduo 2									
Cidade	0	1	2	3	4	5	6	7	8	9
Chave	0,9	0,2	0,1	0,3	0,3	0,6	0,5	0,2	0,3	0,4

Ademais das similaridades entre RKGAs e algoritmos genéticos clássicos, é importante salientar algumas peculiaridades. O operador de mutação, por exemplo, não ocorre alterando soluções existentes dentro da população, mas sim com a geração de soluções novas aleatórias. Este processo também pode ser chamado de imigração de soluções, reforçando o mecanismo de escape de ótimos locais e diversidade.

Um processo muito comum, também utilizado em RKGAs, é o processo de migração. Este processo se define pela manutenção de uma certa quantidade de soluções, que, sem

alterações, são transferidas para a próxima geração. Esta manutenção, geralmente de um conjunto de melhores soluções, chamadas de elites, colaboram com a convergência do algoritmo e aproveitamento das boas características dessas soluções, salientando que com esse processo o algoritmo se torna monotonicamente crescente.

Esta prática é muito comum em RKGAs com múltiplas populações, onde após determinada quantidade de gerações, as populações trocam soluções entre si. Isso gera um efeito benéfico em problemas multi-objetivo, onde é possível definir diferentes funções *fitness* para cada população, ou seja buscando diferentes espaços de busca, que quando combinadas possuem o potencial de encontrar soluções com bons valores no conjunto de objetivos propostos.

É preciso muito cuidado com a definição de quantidade de soluções a serem mantidas entre gerações, uma quantidade muito baixa não terá o efeito de convergência desejado, enquanto uma quantidade muito alta acarretará em uma convergência prematura e perda de diversidade, inibindo a capacidade de exploração do algoritmo.

O conceito de diversidade se aplica a todas as classes de algoritmos evolutivos e se define pela semelhança dos indivíduos de uma população. Caso uma porção considerável da população seja igual ou muito similar, o algoritmo se aproveitará demasiadamente desta similaridade, convergindo especificamente dentro daquele espaço de busca, incapaz de escapar de um eventual ótimo local.

O fluxograma de um possível RKGA é apresentado na Figura 3.7.

3.4 Trabalhos relacionados

Nesta seção foram analisados os trabalhos mais relevantes encontrados na literatura relacionados ao problema de reescalonamento. A análise é realizada em ordem cronológica, partindo de trabalhos mais antigos aos mais atuais.

Moz e Pato (2003) propuseram a resolução do problema utilizando o conceito de fluxo em redes. Em seu trabalho, duas abordagens foram implementadas e comparadas em um algoritmo heurístico e outro utilizando programação linear inteira baseado no modelo desenhado a partir dos fluxos. O algoritmo, utilizando programação linear, geralmente encontrava soluções ótimas, porém, utilizava uma quantidade maior de tempo, enquanto o algoritmo construtivo encontrava boas soluções em uma quantidade de tempo menor. É importante notar que apesar do algoritmo de programação linear inteira utilizar mais tempo, a diferença não é exagerada (Clark e Walker; 2011). No entanto o trabalho também conclui que em determinados casos complexos, o algoritmo levaria tempo considerável ou inviável para resolver.

Acreditando no potencial do algoritmo desenvolvido anteriormente, em 2004, Moz e Pato criaram dois novos modelos baseados em grafos. Ambos ainda utilizavam conceitos de fluxos em redes e programação linear inteira. Este trabalho, além das implementações

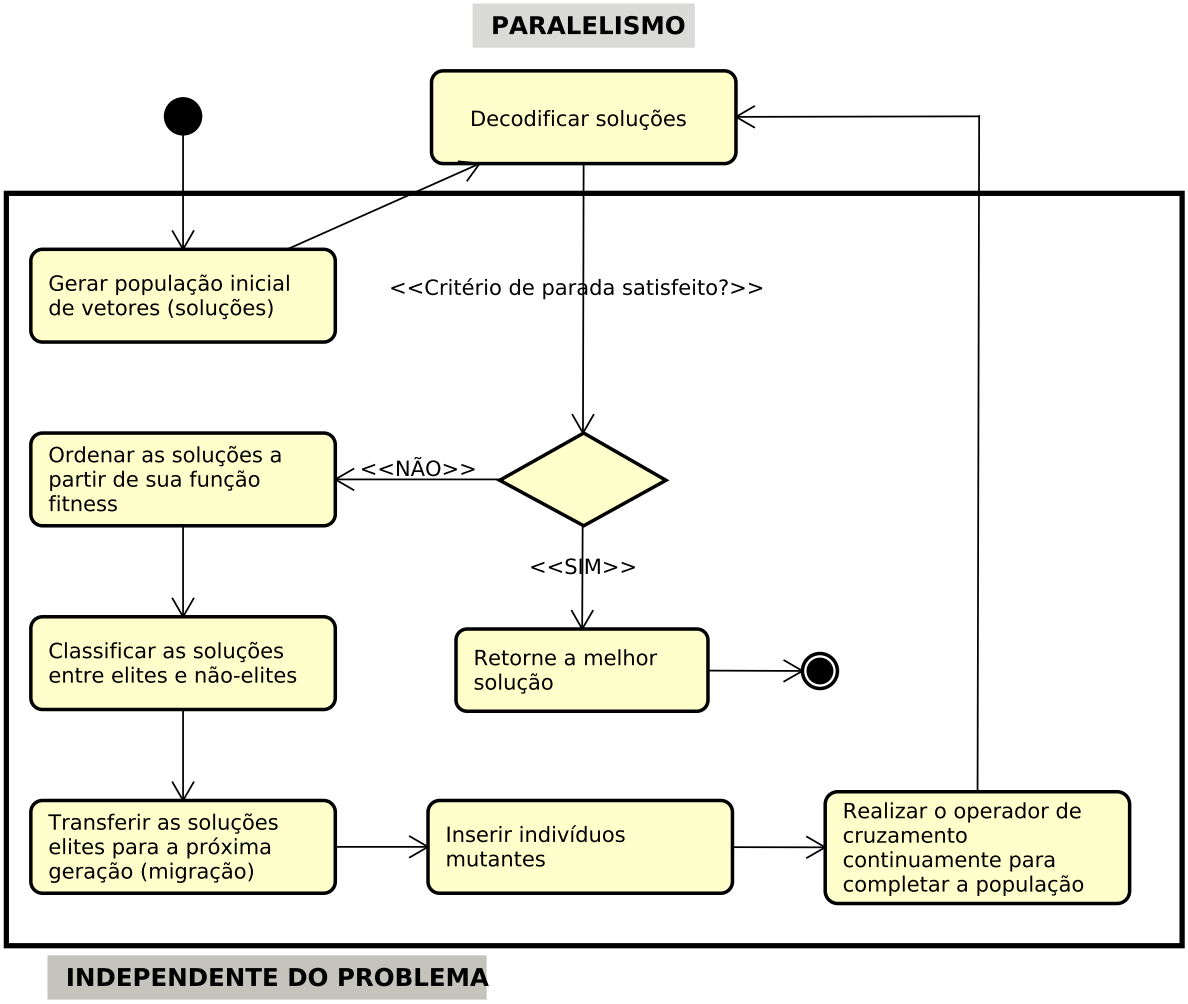


Figura 3.7: *Framework* de um algoritmo genético de chaves aleatórias com elitimismo

algorítmica, também utilizou dados reais de escalonamentos de um hospital em Lisboa, Portugal.

Bard e Purnomo (2005) contemplaram uma visão diferente do problema, onde os escalonamentos eram revisados dia-dia. Com essa visão, foi possível reduzir o impacto das alterações, reduzindo o esforço e complexidade do reescalonamento (Clark e Walker; 2011). Utilizando técnicas de programação linear inteira e *branch-and-price*, o algoritmo foi capaz de resolver eficientemente escalonamentos de até duzentos enfermeiros.

Em 2006, Bard e Purnomo continuaram utilizando programação inteira em seu novo estudo. Porém, este estudo se utilizou de uma abordagem diferente ao problema. No trabalho proposto, não existe uma quantidade de funcionários pré-determinada, mas apenas a demanda. Novos funcionários são adicionados ao quadro pessoal com o objetivo de minimizar o número de turnos com funcionários insuficientes. Extensivos testes foram realizados em dados de um hospital com quatrocentos leitos, onde, na maioria dos casos, o algoritmo solucionou o problema em questão de minutos.

Diferente de seus dois trabalhos iniciais, (Moz e Pato, 2007) utilizaram em seu estudo uma heurística evolutiva, especificamente, algoritmos genéticos. Inicialmente, duas técnicas distintas foram utilizadas na etapa de inicialização do algoritmo genético. Os autores implementaram diversos algoritmos genéticos distintos cujas diferenças se encontram na codificação das permutações de turnos e nos operadores genéticos utilizados. Notou-se que a qualidade dos resultados era dependente da ordenação inicial, portanto, era possível alcançar bons resultados alterando a ordenação inicial, por meio de uma hibridização. Testes computacionais, realizados novamente em dados de um hospital de Lisboa, mostraram que a utilização de algoritmos genéticos à resolução do problema melhorou a qualidade das soluções utilizando uma quantidade de tempo aceitável.

Com a perspectiva de melhoria, Moz e Pato (2008) continuaram os esforços utilizando algoritmos genéticos. Utilizando o princípio de Pareto, próprio para funções multi-objetivo, os autores desenvolveram uma função *fitness* com dois objetivos, onde apenas soluções não dominadas são aceitas na população. Utilizando o conceito de elitismo para convergência, ainda temos a introdução de uma solução utópica na população. Esta solução é gerada a partir de uma heurística construtiva, utilizando uma função *fitness* com apenas um objetivo. Como, de acordo com os autores, as funções descritas anteriormente são potencialmente conflitantes, a interação entre as soluções causa um balanceamento nos objetivos. Seus resultados superaram os trabalhos anteriores das mesmas em termos de competitividade, levando em consideração a qualidade da solução e esforço computacional. Com ênfase na minimização de alterações entre escalas, Kitada e Morizawa (2010) desenvolveram um estudo utilizando árvores recursivas, de modo a resolver problemas de reescalonamento onde os funcionários se ausentam por vários dias. Dividindo os problemas de ausência individualmente, os autores alcançaram bons resultados em dados obtidos de um hospital do Japão, totalizando 3840 instâncias. Os resultados ótimos em 95% dos casos. Infelizmente, esta base não está disponível publicamente.

Maenhout (2011) propôs um algoritmo evolutivo inspirado em técnicas de sucesso comprovadas em outros estudos de escalonamento de enfermeiros. Em seu estudo, são realizadas adaptações nestas técnicas para trabalharem de forma mais eficiente no problema de reescalonamento. Buscando instâncias já existentes na literatura, os autores se utilizaram da base NSPLib como ponto de partida. Selecionando instâncias da base citada, os autores inicialmente resolvem o problema de escalonamento. Posteriormente, impedimentos são adicionados, acarretando em soluções inviáveis prontas para serem utilizadas pelo algoritmo genético reescalonador. Na ocasião, os resultados encontrados foram animadores, porem deve-se salientar que, apesar dos bons resultados, foi encontrada uma inconsistência na relação de restrições do estudo.

Clark e Walker (2011) se voltaram para a programação linear inteira. Em seu estudo, apresentaram modelos distintos inspirados em diferentes abordagens na literatura. Com ênfase na satisfação pessoal de enfermeiros, e não monetários, o estudo alcança bons

resultados preliminares em uma quantidade de tempo razoável. Porém, o autor ressalta que o uso de técnicas exatas torna-se inadequado com a adição de restrições e aumento de complexidade do problema. Infelizmente as instâncias utilizadas pelos autores também não foram encontradas, porém, percebe-se que a cobertura exigida não é complexa, onde apenas de 2 a 3 enfermeiros são exigidos por turno. Esta característica, em conjunto com o horizonte de cobertura definido em 7 dias, não cria problemas de grande complexidade, logo é adequada para o proposto pelo estudo.

Kitada e Morizawa (2013) utilizaram novamente a técnica de árvore recursiva de modo a resolver o problema de reescalonamento de enfermeiros, com ênfase, mais uma vez, em problemas onde os funcionários se ausentam por vários dias, porém, agora, subsequentes. Utilizando-se desta abordagem específica de resolução dos problemas individualmente, os autores alcançaram resultados onde novas escalas viáveis foram encontradas com poucas alterações em relação a escala original. Em ambos os trabalhos destes autores, as informações foram obtidas através de seu *abstract*, pois, os trabalhos em si, estão disponíveis apenas na língua japonesa.

4

Proposta

Este capítulo apresenta na seção 4.1 a modelagem matemática do PRE proposto por este trabalho. Na seção 4.2, a metodologia de criação da base de testes é apresentada, em conjunto com seus dados de entrada. Na seção 4.3, o algoritmo proposto por este estudo para solucionar o problema de reescalonamento de enfermeiros, é apresentado.

4.1 Modelagem matemática do PRE

No modelo proposto, a definição de algumas variáveis e restrições são primordiais para seu correto entendimento.

A primeira variável a ser considerada é a quantidade de enfermeiros, essa quantidade é expressa pela variável N, definindo então um conjunto Enf=$\{1, 2, \ldots, N\}$. O conjunto de dias ou horizonte de designações, é definido por Dias=$\{1, 2, 3, \ldots, D\}$, onde D representa o último dia da escala. Dentro das designações possíveis para cada enfermeiro a cada dia, têm-se o conjunto Turnos=$\{$Manhã, Tarde, Noite, Livre$\}$, representado também pelo intervalo [1, 4]. Dentro de um hospital, há a possibilidade de diferentes unidades independentes. Essa característica foi incorporada ao modelo através do conjunto Set=$\{1, 2, \ldots, S\}$, onde S representa o número de setores. Cada turno de cada setor possui sua própria demanda de enfermeiros, essas são representadas através de números naturais. Todos os turnos, a exceção dos turnos livres de cada setor, possuem uma demanda positiva, exigindo então ao menos um enfermeiro. A carga horária dos turnos utilizada por este trabalho é de 6 horas para os turnos matutinos e vespertinos e 12 horas para o turno noturno.

A lista a seguir apresenta as restrições utilizadas no modelo proposto por este trabalho.

1. Restrições Rígidas

(a) Cada enfermeiro pode ser designado para apenas um turno de cada dia.

(b) As demandas operacionais de cada turno de cada setor devem ser satisfeitas.

(c) Após um turno Tarde, um enfermeiro não poderá trabalhar no turno Manhã consecutivamente.

(d) Após um turno Noite, um enfermeiro obrigatoriamente deverá ser designado para o turno Livre no dia subsequente.

(e) Os enfermeiros apenas poderão trabalhar em setores nos quais possuem competência.

(f) Os enfermeiros possuem um limite de horas trabalhadas de 144 horas por mês.

(g) Todo enfermeiro deve ser designado para o turno Livre (24 horas) ao menos uma vez a cada 7 dias consecutivos.

(h) O enfermeiro deverá ser obrigatoriamente designado para o turno Livre no dia que estiver ausente.

2. Restrições Flexíveis

(a) A quantidade de alterações na escala original é objeto de minimização.

As restrições acima apresentadas são encontradas na literatura correlata (Mutingi e Mbohwa, 2017; Cheang, 2003). Algumas destas restrições (1a, 1b, 1c, 1d e 1g) são comuns em quase todos os estudos na área. A adição de setores independentes (1e), apesar de menos comum, também já foi estudada anteriormente (Isken e Hancock, 1991; Wright, 2013) e reflete algo comum em escalonamentos reais (Legrain *et al*, 2015). A utilização de limite superior de horas ou designações também é frequente nos estudos correlatos. O limite de 144 horas mensais (1f) segue uma adaptação, e encontra um meio termo entre uma carga horária semanal de 30 e 40 horas, fixando este limite em 36 horas, que quando multiplicado por quatro semanas, resulta em uma carga horária total de 144 horas.

A restrição 1(b) introduz uma variável de entrada, *Dop*, definida por uma matriz de três dimensões, sendo indexada por setor do conjunto *Set*, dia do conjunto *Dias* e turno do conjunto *Turnos*, respectivamente, onde cada valor pertence ao conjunto $\mathbb{N}^+$, com exceção dos turnos Livres, que são fixados com valor zero. A equação 4.1 demonstra sua definição.

$$Dop_{sdt} = \begin{cases} v & \text{se } t \neq \text{Livre e } v \geq 1 \\ 0 & \text{se } t = \text{Livre} \end{cases} \qquad \forall s \in Set, d \in Dias, t \in Turnos \qquad (4.1)$$

A restrição 1(e) utiliza a variável *Cap*, sendo sua definição uma matriz binária de duas dimensões, indexada respectivamente por, enfermeiro do conjunto *Enf* e setor do conjunto

Set é definida na equação 4.2.

$$Cap_{ns} = \begin{cases} 1 & \text{se o enfermeiro } n \text{ pode trabalhar no setor } s \\ 0 & \text{caso contrário} \end{cases} \quad \forall n \in Enf, s \in Set \tag{4.2}$$

A quantidade de possíveis designações é definida por um conjunto $Dp = \{1, 2, 3, ..., p\}$, onde $p{=}S \times \|Turnos\|$, é um conjunto binário em que o turno e setor designado podem ser encontrados pelas formulas 4.5 e 4.6, respectivamente.

$$1 = \sum_1^p Dp_p \tag{4.3}$$

$$X = \sum_1^p (Dp_p \times p) \tag{4.4}$$

$$SD = \lceil (X \div 4) \rceil \tag{4.5}$$

$$TD = X \ mod \ 4 \tag{4.6}$$

A razão pela qual um reescalonamento é necessário, é devido a ausência de um ou mais enfermeiros, causando então um comprometimento da demanda operacional. Essas ausências são definidas na forma de uma matriz binária, *Aus*, de duas dimensões indexada, respectivamente, por enfermeiros do conjunto *Enf* e dias do conjunto *Dias*, onde o valor 0 representa a possibilidade de trabalho por parte do enfermeiro naquele dia e 1 caso contrário.

De forma resumida, os dados de entrada e as variáveis de decisão para o problema de reescalonamento de enfermeiros, de acordo com modelo o proposto, são apresentados na Tabela 4.1.

Tabela 4.1: Dados de entrada e variáveis de decisão da modelagem proposta

Índices	
s	Setores
n	Enfermeiros
d	Dias
p	$s \times 4$

Entrada - PRE	
Enf	{1, 2, 3, 4, ..., n}
Dias	{1, 2, 3, 4, ..., d}
Set	{1, 2, 3, 4, ..., s}
Dp	{1, 2, 3, 4, ..., p} :
Cap	Matriz 2D $\in$ {0, 1} : Indexadores ({*Enf*},{*Set*})
Turnos	{1 (Manhã), 2 (Tarde), 3 (Noite) , 4(Livre)}
Hpt	{0 (Livre), 6 (Manhã), 6 (Tarde), 12 (Noite)}
Dop	Matriz 3D : Indexadores ({*Set*}, {*Dias*}, {*Turnos*}) $\in$ {0, 1}
Mht	144 horas
EscalaOriginal	Matriz 3D : Indexadores ({*Enf*}, {*Dias*}, {*Dp*}) $\in$ {0, 1}
Aus	Matriz 2D : Indexadores ({*Enf*}, {*Dias*}) $\in$ {0, 1}
Dpa	Dia da Primeira Ausência : $\in$ [5,10] $\vee$ [19,24]

Variáveis de Decisão - PRE	
Variável	Descrição
D_{ndp}	Essencial - Designação : $\forall n \in Enf,\ \forall d \in Dias,\ \forall c \in Dp$
Htd_{nd}	Auxiliar - Horas de trabalho por dia : $\forall n \in Enf,\ \forall d \in Dias$
Htm_{n}	Auxiliar - Horas de trabalho por mês : $\forall n \in Enf$
Dif_{nd}	Auxiliar - Diferenças entre escalas : $\forall n \in Enf,\ \forall d \in Dias$

O modelo matemático proposto para o PRE, sujeito as restrições citadas é apresentado a seguir:

$$minimize \sum_{1}^{n}\sum_{1}^{d} Dif_{nd} \tag{4.7}$$

subject to (4.8)

$$\sum_{1}^{p} D_{ndp} = 1 \qquad \forall n \in Enf,\ \forall d \in Dias,\ \forall p \in Dp \tag{4.9}$$

$$\sum_{1}^{p} (A_{ndp} \times EscalaOriginal_{ndp}) = 1 \qquad \forall n \in Enf,\ \forall d \in [1, (Dpa - 1)] \tag{4.10}$$

$$| \sum_{1}^{p}(A_{ndp} \times p) - \sum_{1}^{p}(EscalaOriginal_{ndp} \times p) | = M' \qquad \forall n \in Enf, \forall d \in [Dpa, Dias] \tag{4.11}$$

$$M' - (4 \times Q) = M'' \qquad Q \in \mathbb{N} \tag{4.12}$$

$$\frac{M''}{p} = Dif_{nd} \qquad p \in Dp \quad \frac{M''}{p} \leq Dif_{nd} < \frac{M''}{p} + 1 \tag{4.13}$$

$$\sum_{1}^{p}(A_{ndp} \times Hpt_{p\ mod\ 4}) = Htd_{nd} \qquad \forall n \in Enf,\ \forall d \in Dias \tag{4.14}$$

$$\sum_{1}^{d} Htd_{\mathrm{nd}} = Htm_{n} \qquad \forall n \in Enf \tag{4.15}$$

$$Htm_{n} \leq Mht \qquad \forall n \in Enf \tag{4.16}$$

$$\sum_{1}^{p} D_{ndp} + \sum_{1}^{p'} D_{n(d+1)p'} \leq 1 \qquad \begin{array}{ll} \forall n \in Enf & \forall d \in Dias - 1 \\ p\ mod\ 4 = 2 & p'\ mod\ 4 = 1 \end{array} \tag{4.17}$$

$$3\sum_{1}^{p} D_{ndp} + \sum_{1}^{p'} D_{n(d+1)p'} \leq 1 \qquad \begin{array}{ll} \forall n \in Enf & \forall d \in Dias - 1 \\ p\ mod\ 4 = 3 & p'\ mod\ 4 \neq 0 \end{array} \tag{4.18}$$

$$\sum_{1}^{p} \left(D_{ndp} \times Cap_{n\lceil (p/4) \rceil}\right) = 1 \qquad \forall n \in Enf,\ \forall d \in Dias \tag{4.19}$$

$$\sum_{dx}^{dx+6} A_{ndp} \geq 6 \qquad \begin{array}{ll} \forall n \in Enf & \forall dx \in Dias - 6 \\ \forall p \in Dp & . \\ p\ mod\ 4 \neq 0 & . \end{array} \tag{4.20}$$

$$\sum_{1}^{n} D_{ndp} \geq Dop_{sdt} \qquad \begin{array}{ll} \forall s \in Set & \forall d \in Dias \\ \forall t \in \{1,2,3\} & \\ p = (((s-1) \times 4) + t) & \end{array} \tag{4.21}$$

A equação 4.7 apresenta a função objetivo do modelo matemático, de reduzir a quantidade de alterações realizadas na escala (restrição 2a). Pelo modelo proposto,

na equação 4.13, nota-se que apenas mudanças de turno são contabilizadas, enquanto mudanças de setor não interferem nesta variável. A restrição de apenas uma designação por dia (1a) é introduzida pela equação 4.9. Conforme preconizado na literatura, as alterações que ocorrem em um reescalonamento devem ocorrer a partir do primeiro dia de ausências, portanto as designações realizadas antes deste evento são mantidas, o que pode ser verificada pela equação 4.10. A restrição de horas trabalhadas por mês (1f) é verificada pelas restrições 2.16, 2.17 e 2.18. O descanso obrigatório após os turnos Tarde (1c) e Noite (1d), são aplicados pelas equações 4.17 e 4.18 respectivamente. A restrição de um enfermeiro ser habilitado para trabalhar apenas em setores seletos (1e) é compelido pela equação 4.19. O descanso mínimo obrigatório a cada sete dias (1g) é aplicado pela equação 4.20 e por fim a a satisfação da demanda operacional (1b) é aplicada pela equação 4.21.

4.2 Base de testes

A ausência de uma base de dados para testes, também conhecida como base de *benchmark* e resultados para comparação dificulta a análise de diferentes abordagens para o problema. Apesar da existência da base proposta por Moz e Pato (2007), a dificuldade de utilização, quantidade de instâncias e generalidade da base, motivou a criação de uma base para este estudo.

Não há estudo conclusivo, utilizando métodos exatos, que apontam diretamente para uma determinada característica de uma instância, que afeta diretamente a sua dificuldade de resolução. Ou seja, não basta apenas aumentar a quantidade de enfermeiros a serem reescalonados, pois isso não necessariamente refletirá em uma maior dificuldade de resolução, apesar de ser algo considerado.

Problemas com demasiadas restrições tendem a ser mais difíceis de resolver (Smith-Miles e Lopes, 2012), principalmente por métodos exatos. Ainda, a adição de demasiadas restrições, geraria uma base especializada e isto é evitado na base criada. O objetivo da criação desta base, além da necessidade criação para investigação do algoritmo, reside na disponibilização de uma nova base, com restrições comumente utilizadas na literatura, possuindo uma quantidade considerável de instâncias e diferentes níveis de dificuldade.

A metodologia de criação de uma base de reescalonamento se inicia, inevitavelmente, na criação de uma escala inicial, remetendo ao problema de escalonamento de enfermeiros. Antes de maior aprofundamento nas restrições, funções objetivo e dimensionalidade das instâncias, uma visão geral é apresentada nas etapas a seguir.

- Preparação (Criação de instâncias iniciais de escalonamento): Nesta pré-etapa, instâncias foram criadas de forma controlada. Dentre essas características, encontram-se as demandas operacionais, capacidade de trabalho, dias de folga e a quantidade de

enfermeiros inicial, lembrando que durante a criação da demanda de uma instância, a quantidade de enfermeiros que é necessária para sua viabilidade é incerta, portanto esta quantidade é minimizada na etapa a seguir.

- Etapa 1 (Escalonamento 1): Nesta etapa, as instâncias são resolvidas com o objetivo de minimizar a quantidade de enfermeiros necessária para resolução do problema.

- Etapa 2 (Escalonamento 2): Após a exclusão de enfermeiros extras pela etapa 1, nesta nova etapa, o objetivo consiste na minimização das horas trabalhadas da escala. A utilização deste objetivo se baseia em dois pontos, o primeiro remete à ideia comum em pesquisa operacional de redução de custo, onde menos horas trabalhadas resulta em um custo menor. Já o segundo ponto visa dar uma margem de liberdade maior na etapa seguinte, onde uma maior quantidade de enfermeiros com turnos livres e com carga horária mensal não completa, facilita e viabiliza o reescalonamento, pois caso uma instância trabalhe com exatamente o mínimo de funcionários, sem nenhuma margem para alterações, o problema reescalonamento provavelmente se tornaria impossível de ser solucionado.

- Etapa 3 (Reescalonamento): Nesta última etapa, o objetivo agora é de minimização de alterações, que é o objetivo mais comum do problema, salientando que os resultados obtidos nesta etapa, são utilizados para comparação e análise posterior.

As restrições utilizadas em cada etapa são apresentadas na Tabela 4.1.

Tabela 4.2: Restrições utilizadas nas diferentes etapas. RR: restrição rígida, RF: restrição flexível, NU: não utilizada.

Descrição da restrição	Restrições								
	Etapa 1			Etapa 2			Etapa 3		
	RR	RF	NU	RR	RF	NU	RR	RF	NU
Apenas uma atribuição por dia	X			X			X		
Demanda operacional	X			X			X		
Padrões proibidos de trabalho	X			X			X		
Capacidade do enfermeiro	X			X			X		
Carga horária mensal mínima	X			X					X
Carga horária mensal máxima	X			X			X		
Descanso a cada sete dias de trabalho	X			X			X		
Conceder solicitações de dias de folga	X			X					X
Alterações na escala original			X			X		X	

A exclusão da restrição de carga horária mínima na etapa 3 se deve ao fato de afetar a viabilidade de resolução do problema. Caso um enfermeiro se ausente por diversos dias, este pode não ser capaz de satisfazer esta restrição, porém do ponto de vista operacional, a escala ainda poderia ser corrigida. Esta defasagem de carga horária poderia ser incluída

em uma nova restrição no processo de escalonamento de um período subsequente, onde este enfermeiro deverá trabalhar uma quantidade de horas maior, de forma a compensar as ausências. Isto foi proposto, apesar de não ser utilizado, pela base de Moz e Pato, onde é definido um débito ou crédito de horas para cada enfermeiro.

As solicitações de dias de folga foram utilizadas pois abrem espaço para outras abordagens ao problema, não alvos de estudo, onde estas concessões poderiam fazer parte do conjunto de restrições flexíveis, apresentando outro mecanismo de avaliação da escala e não apenas a minimização de alterações.

Para cada etapa descrita anteriormente, um modelo matemático foi desenvolvido. Estes modelos foram implementados na linguagem OPL e executados no resolvedor CPLEX. Os modelos das etapas 1 e 2 são apresentados a seguir, o modelo da etapa 3 pode ser encontrado na subseção 4.1.1, de definição matemática do PRE. As tabelas 4.3 e 4.4 apresentam os respectivos dados de entrada e variáveis de decisão das etapas 1 e 2.

Tabela 4.3: Dados de entrada e variáveis - Etapa 1

Entrada - Etapa 1	
Nome	Descrição
Enf	{1, 2, 3, 4, ..., N} : N = número de enfermeiros
Dias	{1, 2, 3, 4, ..., D} : D = número de dias
Set	{1, 2, 3, 4, ..., S} : S = número de setores
Dp	{1, 2, 3, 4, ..., P} : P = número de setores $\times 4$
Cap	Capacidade de trabalho - Matriz 2D : Indexadores ({*Enf*}, {*Set*}) $\in$ {0, 1}
Turnos	{1 (Manhã), 2 (Tarde), 3 (Noite) , 4(Livre)}
Hpt	{0 (Livre), 6 (Manhã), 6 (Tarde), 12 (Noite)} - Horas por turno
Dop	Matriz 3D : Indexadores ({*Set*}, {*Dias*}, {*Turnos*}) - Demanda operacional
Rdf	Matriz 2D : Indexadores ({*Enf*}, {*Dia*}) - Dias de folga $\in$ {0, 1}
Maxht	144 horas - Máximo de horas trabalhadas
Minht	132 horas - Mínimo de horas trabalhadas

Variáveis de Decisão - Etapa 1	
Variável	Descrição
D_{ndp}	Essencial - Designação : $\forall n \in Enf$, $\forall d \in Dias$, $\forall c \in Dp \in \{0, 1\}$
Htd_{nd}	Auxiliar - Horas de trabalho por dia : $\forall n \in Enf$, $\forall d \in Dias$
Htm_{n}	Auxiliar - Horas de trabalho por mês : $\forall n \in Enf$
ENU_n	Auxiliar - Enfermeiros não utilizados : $\forall n \in Enf$

$$minimize \sum_{1}^{n} \sum_{1}^{d} ENU_n \tag{4.22}$$

subject to (4.23)

$$\sum_{1}^{p} D_{ndp} = 1 \qquad \forall n \in Enf,\ \forall d \in Dias,\ \forall p \in Dp \tag{4.24}$$

$$\sum_{1}^{p} (A_{ndp} \times Hpt_{p\ mod\ 4}) = Htd_{nd} \qquad \forall n \in Enf,\ \forall d \in Dias \tag{4.25}$$

$$\sum_{1}^{d} Htd_{\mathrm{nd}} = Htm_{n} \qquad \forall n \in Enf \tag{4.26}$$

$$Minht \times ENU_{n} \geq Htm_{n} \qquad \forall n \in Enf \tag{4.27}$$

$$Maxht \times ENU_{n} \leq Htm_{n} \qquad \forall n \in Enf \tag{4.28}$$

$$\sum_{1}^{p} D_{ndp} + \sum_{1}^{p'} D_{n(d+1)p'} \leq 1 \qquad \begin{array}{r} \forall n \in Enf \quad \forall d \in Dias - 1 \\ p\ mod\ 4 = 2 \quad p'\ mod\ 4 = 1 \end{array} \tag{4.29}$$

$$\sum_{1}^{p} D_{ndp} + \sum_{1}^{p'} D_{n(d+1)p'} \leq 1 \qquad \begin{array}{r} \forall n \in Enf \quad \forall d \in Dias - 1 \\ p\ mod\ 4 = 3 \quad p'\ mod\ 4 \neq 0 \end{array} \tag{4.30}$$

$$\sum_{1}^{p} \left(D_{ndp} \times Cap_{n\lceil (p/4) \rceil}\right) = 1 \qquad \forall n \in Enf,\ \forall d \in Dias \tag{4.31}$$

$$D_{ndp} \times Rdf_{nd} \neq 1 \qquad \begin{array}{r} \forall n \in Enf \quad \forall d \in Dias \\ \forall p \in Dp \quad p\ mod\ 4 \neq 0 \end{array} \tag{4.32}$$

$$\sum_{dx}^{dx+6} A_{ndp} \geq 6 \qquad \begin{array}{r} \forall n \in Enf \\ \forall dx \in Dias - 6 \\ \forall p \in Dp \\ p\ mod\ 4 \neq 0 \end{array} \tag{4.33}$$

$$\sum_{1}^{n} D_{ndp} \geq Dop_{sdt} \qquad \begin{array}{r} \forall s \in Set \\ \forall d \in Dias \\ \forall t \in \{1, 2, 3\} \\ p = (((s-1) \times 4) + t) \end{array} \tag{4.34}$$

O objetivo de minimização de enfermeiros é apresentado na função objetivo (eq. 4.22). A restrição de designação única por dia é aplicada pela equação 4.24. As restrições em relação a carga horária mínima e máxima são encontradas com a combinação das equações 4.25, 4.26, 4.27 e 4.28. A proibição de determinados padrões de trabalho são reforçadas pelas equações 4.29 e 4.30. Cada enfermeiro pode trabalhar apenas em setores pré-definidos, visto na equação 4.31. As solicitações de dias de folga devem ser obrigatoriamente satisfeitas nesta etapa (eq. 4.32). A restrição de ao menos um descanso a cada sete dias é aplicada pela equação 4.33. E finalmente, a demanda operacional deve ser satisfeita compulsoriamente (eq. 4.34).

Tabela 4.4: Dados de entrada e variáveis - Etapa 2

Entrada - Etapa 2	
Nome	Descrição
Enf	{1, 2, 3, 4, ..., N} : N = número de enfermeiros
Dias	{1, 2, 3, 4, ..., D} : D = número de dias
Set	{1, 2, 3, 4, ..., S} : S = número de setores
Dp	{1, 2, 3, 4, ..., P} : P = número de setores ×4
Cap	Matriz 2D : Indexadores ({Enf}, {Set}) - Capacidade de trabalho ∈ {0, 1}
Turnos	{1 (Manhã), 2 (Tarde), 3 (Noite) , 4(Livre)}
Hpt	{0 (Livre), 6 (Manhã), 6 (Tarde), 12 (Noite)} - Horas por turno
Dop	Matriz 3D : Indexadores ({Set}, {Dias}, {Turnos}) - Demanda operacional
Rdf	Matriz 2D : Indexadores ({Enf}, {Dia}) - Dias de folga ∈ {0, 1}
Maxht	144 horas - Máximo de horas trabalhadas
Minht	132 horas - Mínimo de horas trabalhadas

Variáveis de Decisão - Etapa 2	
Variável	Descrição
D_{ndp}	Essencial - Designação : $\forall n \in Enf,\ \forall d \in Dias,\ \forall c \in Dp$
Htd_{nd}	Auxiliar - Horas de trabalho por dia : $\forall n \in Enf,\ \forall d \in Dias$
Htm_{n}	Auxiliar - Horas de trabalho por mês : $\forall n \in Enf$

$$minimize \sum_{1}^{n}\sum_{1}^{d} Htm_n \tag{4.35}$$

subject to (4.36)

$$\sum_{1}^{p} D_{ndp} = 1 \qquad \forall n \in Enf,\ \forall d \in Dias,\ \forall p \in Dp \tag{4.37}$$

$$\sum_{1}^{p} (A_{ndp} \times Hpt_{p\ mod\ 4}) = Htd_{nd} \qquad \forall n \in Enf,\ \forall d \in Dias \tag{4.38}$$

$$\sum_{1}^{d} Htd_{\mathrm{nd}} = Htm_n \qquad \forall n \in Enf \tag{4.39}$$

$$Minht \leq Htm_n \qquad \forall n \in Enf \tag{4.40}$$

$$Maxht \geq Htm_n \qquad \forall n \in Enf \tag{4.41}$$

$$\sum_{1}^{p} D_{ndp} + \sum_{1}^{p'} D_{n(d+1)p'} \leq 1 \qquad \begin{array}{r} \forall n \in Enf \quad \forall d \in Dias - 1 \\ p\ mod\ 4 = 2 \quad p'\ mod\ 4 = 1 \end{array} \tag{4.42}$$

$$\sum_{1}^{p} D_{ndp} + \sum_{1}^{p'} D_{n(d+1)p'} \leq 1 \qquad \begin{array}{r} \forall n \in Enf \quad \forall d \in Dias - 1 \\ p\ mod\ 4 = 3 \quad p'\ mod\ 4 \neq 0 \end{array} \tag{4.43}$$

$$\sum_{1}^{p} \left(D_{ndp} \times Cap_{n\lceil(p/4)\rceil}\right) = 1 \qquad \forall n \in Enf,\ \forall d \in Dias \tag{4.44}$$

$$D_{ndp} \times Rdf_{nd} \neq 1 \qquad \begin{array}{r} \forall n \in Enf \quad \forall d \in Dias \\ \forall p \in Dp \quad p\ mod\ 4 \neq 0 \end{array} \tag{4.45}$$

$$\sum_{dx}^{dx+6} A_{ndp} \geq 6 \qquad \begin{array}{r} \forall n \in Enf \\ \forall dx \in Dias - 6 \\ \forall p \in PA \\ p\ mod\ 4 \neq 0 \end{array} \tag{4.46}$$

$$\sum_{1}^{n} D_{ndp} \geq Dop_{sdt} \qquad \begin{array}{r} \forall s \in Set \\ \forall d \in Dias \\ \forall t \in \{1, 2, 3\} \\ p = (((s-1) \times 4) + t) \end{array} \tag{4.47}$$

Neste segundo modelo, as diferenças em relação ao modelo anterior residem: na função objetivo (eq. 4.35), onde agora, conforme apresentado anteriormente, objetiva-se a minimização de horas trabalhadas por mês da escala. E na remoção da variável *ENU* (eq. 4.40, eq. 4.41), pois a identificação de enfermeiros extras não é mais objeto de minimização.

Apresentados os modelos matemáticos das etapas anteriores à etapa de reescalonamento, os dados de entrada para a primeira etapa de escalonamento são apresentados. Buscando diferentes níveis de dificuldade, os pontos a seguir foram considerados:

Quantidade de enfermeiros: A quantidade de enfermeiros pode afetar negativamente ou positivamente os problemas de escalonamento e reescalonamento. Uma quantidade grande de enfermeiros gera um espaço de busca maior, dificultando o encontro do resultado ótimo. Em geral a demanda não cresce de forma proporcional ao número de enfermeiros, com uma relação (número de enfermeiros × demanda) mais "folgada" espera-se uma maior taxa de viabilidade.

Setores: A inclusão de setores independentes qualifica-se como um obstáculo na resolução de problemas, possuindo diferentes demandas e quantidade de enfermeiros capazes de serem designados. Lembrando que a designação por capacidade é rígida, portanto sua inclusão torna o problema mais restritivo.

Capacidade do enfermeiro: A capacidade de um enfermeiro de trabalhar em diferentes setores impacta em sua dificuldade. Em um cenário onde todos os enfermeiros possuem total capacidade, o problema seria simplificado, eliminando a restrição. Caso possuíssem capacidade única, o problema ficaria extremamente restritivo. Ademais, no último cenário seria possível quebrar o problema geral em subproblemas, onde cada setor seria escalonado e reescalonado independentemente.

Utilizando as ideias acima, sessenta instâncias foram criadas de forma controlada, possuindo alguns parâmetros aleatórios:

- **Setores:** As instâncias possuem 6, 8 ou 10 setores.
- **Quantidade inicial de enfermeiros:** 100, 120 e 150, respectivamente pela quantidade de setores.
- **Demanda operacional:** Para cada turno de cada setor de cada dia, a demanda foi definida no intervalo [1, ((quantidade de enfermeiros ÷ quantidade de setores) ÷ numero de turnos)], adicionando o valor de 1 ou 2 aleatoriamente.
- **Capacidade dos enfermeiros:** Definido de forma que raramente os enfermeiros possuam capacidade total ou única, sendo que a quantidade de setores que um enfermeiro é capaz de trabalhar é aproximadamente 40% da quantidade de setores. Salientando que todos os setores possuem uma quantidade mínima de enfermeiros habilitados, que é encontrado pela média da quantidade de enfermeiros pela quantidade de setores.

As instâncias criadas nesta pré-etapa, são então executadas pelo resolvedor CPLEX, compondo a Etapa 1 descrita anteriormente, reduzindo a quantidade de enfermeiros necessária.

Posterior a este processo e anterior a Etapa 2, de forma a alterar a relação da quantidade de enfermeiros com a demanda, as instâncias resultantes da Etapa 1 passaram por um processo de adição de enfermeiros, dividida em três categorias:

- **Mínimo:** A quantidade de enfermeiros resultante da Etapa 1 é mantida.
- **Adição de 10%:** Dos resultados da Etapa 1, um adicional de 10% de enfermeiros é adicionado, sem alteração na demanda.
- **Adição de 20%:** Dos resultados da Etapa 1, um adicional de 20% de enfermeiros é adicionado, sem alteração na demanda.

Com as modificações apresentadas, tem-se agora 120 instâncias de entrada para a segunda etapa do processo de criação da base, onde foi minimizado a quantidade de horas total das escalas.

Passada a segunda fase deste processo, problemas foram criados nas escalas resultantes. Estes problemas tomam forma na ausência de uma quantidade de enfermeiros durante certos dias. A dificuldade e tamanho de espaço de busca também é influenciada pela quantidade de enfermeiros ausentes e pelo momento em que ocorrem as ausências. Uma quantidade grande de ausências causa um impacto negativo para o reescalonamento, visto a redução da quantidade de enfermeiros durante o período.

Considerando o momento das ausências, quando nos dias iniciais da escala, permitem uma maior liberdade de alterações, visto que há mais dias para se alterar, porém ressalta-se a maior dificuldade de encontrar resultados ótimos em face do maior espaço de busca. Ausências que se iniciam na segunda metade da escala, possuem espaço de busca menor, porém possuem dificuldade aumentada tendo em vista as restrições impostas, principalmente de carga horária e de trabalho continuado.

Sendo assim, as 120 instâncias foram assim modificadas, gerando um total de 480 instâncias:

- 10% de ausências na primeira metade da escala, se iniciando entre os dias 5 e 10.
- 20% de ausências na primeira metade da escala, se iniciando entre os dias 5 e 10.
- 10% de ausências na segunda metade da escala, se iniciando entre os dias 19 e 24.
- 20% de ausências na segunda metade da escala, se iniciando entre os dias 19 e 24.

A figura 4.1 apresenta o processo completo de criação da base de testes, com todos os passos, incluindo os devidos escalonamentos, adição de enfermeiros, adição de ausências e reescalonamento.

A base final de testes contém 120 instâncias, divididas igualmente em 3 classes, extraídas após o processo de reescalonamento do resolvedor CPLEX. As instâncias são divididas de acordo com a dificuldade encontrada pelo método exato, ressaltando que, apesar da capacidade de paralelismo de ambas as técnicas propostas, para uma melhor comparação, ambas utilizaram uma única *thread*:

- **Classe Fácil:** Instâncias cujo ótimo foi encontrado em tempo de execução inferior a 10 minutos.
- **Classe Média:** Instâncias cujo ótimo foi encontrado em tempo de execução inferior a 20 minutos e superior a 10 minutos.
- **Classe Difícil:** Instâncias cujo ótimo não foi encontrado após 20 minutos de execução.

O número de 40 instâncias por classe foi decidido, pelo fator viabilidade. Durante a execução, muitas instâncias se mostraram inviáveis, mesmo após 20 minutos de execução. A classe difícil, que ao final, possuía o menor número de instâncias, foi utilizada como limitadora, para deixar a quantidade de instâncias por classe uniforme.

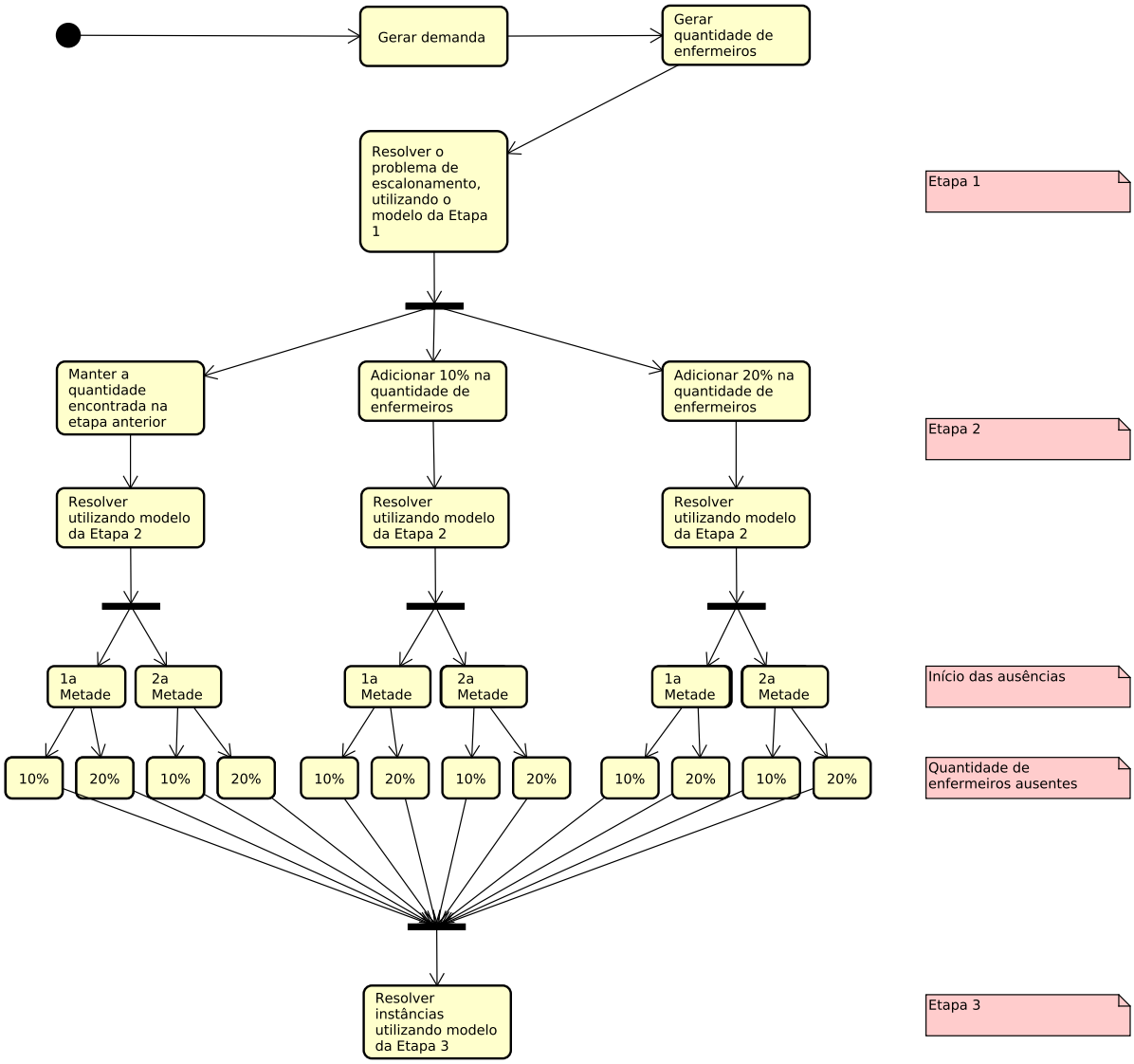


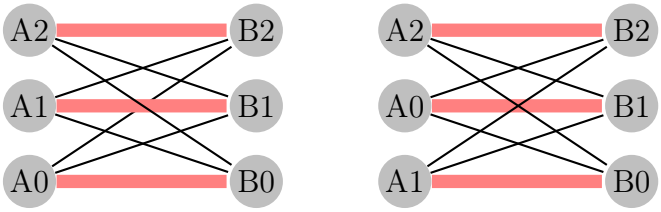
Figura 4.1: Processo completo de criação da base de testes

4.3 Algoritmo proposto - RKGA-HK

4.3.1 Cromossomo

O algoritmo genético de chaves aleatórias proposto por este estudo é baseado no conceito de grafos bipartidos. A cada dia de reescalonamento um grafo bipartido é construído, onde o primeiro conjunto de vértices (V1) é composto por enfermeiros e o segundo (V2) por tarefas a serem designadas. Utilizando o algoritmo de Hopcroft-Karp, uma solução viável é encontrada quando todos os dias do reescalonamento, com seus respectivos grafos bipartidos, tem um emparelhamento máximo encontrado, resultando na realização de todas as tarefas do problema. O termo emparelhamento máximo utilizado neste parágrafo, consiste no emparelhamento onde o número de arestas é igual ao número de vértices do conjunto V2.

No algoritmo HK, a ordem na qual os vértices são dispostos pode gerar resultados diferentes, conforme exemplo na figura 4.2, onde a ordenação alterou a designação dos vértices A1 e A0



(a) Resultado 1 (b) Resultado 2

Figura 4.2: Exemplo de resultados utilizando uma ordenação diferente no primeiro conjunto de vértices.

De forma resumida, cada cromossomo é composto por um conjuntos de chaves, com duas características distintas:

- Característica de ordem: São responsáveis por determinar a ordem dos vértices (enfermeiros) no algoritmo HK.
- Característica de decisão: São responsáveis por determinar quais designações são passíveis a mudança de turno.

Utilizando as definições de chaves acima, para cada dia, um vetor de tamanho N é utilizado. Em conjunto com um processo de decodificação por ordem crescente de valor, para determinar a ordem dos enfermeiros no grafo bipartido, resultando, em um vetor de tamanho $D \times N$.

Este mesmo conjunto de chaves aleatórias é utilizado como mecanismo de preservação de designações, utilizando a característica de decisão. Se tratando de um problema cujo objetivo é a minimização de alterações, a preservação de designações da escala original auxilia a resolução do problemas em duas frentes: a primeira consiste no impacto direto no objetivo do problema, a minimização de alterações, o segundo consiste na simplificação do processo de busca. Ao fixar certas designações, ocorre uma redução do tamanho do problema, resultando possivelmente em uma maior eficiência do algoritmo.

Esta decisão é baseada no valor da chave, caso o valor chave de um dia D e enfermeiro N seja inferior ao valor de corte, implicará na possibilidade apenas de mudança de setor, ocorrendo a manutenção da designação de turno. Caso o valor seja superior ao valor de corte, esta nova designação poderá ser tanto um novo setor, quanto um novo turno, obedecendo as regras de construção do grafo, descritas a seguir.

4.3.2 Procedimento de decodificação

Definidas chaves utilizadas pelo cromossomo, a visão geral do processo de decodificação pode ser dado pelos passos a seguir:

1. Utilizar a característica de decisão para determinar quais designações serão mantidas.
2. Para cada dia de reescalonamento:
   (a) Utilizar a ordenação obtida pelas chaves de ordem para adicionar vértices ao primeiro conjunto de vértices do grafo bipartido (Enfermeiros).
   (b) Adicionar vértices ao segundo conjunto de vértices do grafo bipartido, onde estes vértices representam as tarefas de tal dia de reescalonamento.
   (c) Para cada enfermeiro (vértice do primeiro conjunto):
      - Adicione arestas ligando-o a tarefas, onde tais ligações não violem nenhuma restrição rígida e obedeça ao critério de manutenção obtido pela sua chave de decisão.
      - Execute o algoritmo HK
3. Para cada dia com tarefas não satisfeitas:
   - Descarte as fixações realizadas pelas característica de decisão (Procedimento de correção)
   - Execute o algoritmo HK

Os grafos de cada dia são construídos de forma sequencial, ou seja, levam em consideração os emparelhamentos encontrados previamente. Isto é feito para que as soluções geradas ao final do último emparelhamento sejam viáveis considerando todas as restrições rígidas do problema, com exceção da restrição de demanda operacional. Em um exemplo prático, caso um enfermeiro esteja designado para trabalhar em um turno da tarde, na construção do grafo do dia subsequente não existirá aresta ligando-o a uma tarefa no turno da manhã, pois isto violaria a restrição de descanso entre turnos.

A construção dos grafos foi assim idealizada com o intuito de aliviar o fardo do algoritmo genético em seu processo de busca, lembrando que problemas com restrições demasiadas, em geral, possuem dificuldade maior de resolução.

Durante o processo de elaboração do algoritmo, outras abordagens foram testadas, onde todas as arestas eram consideradas, incluindo arestas que implicavam em violações rígidas. Porém tais abordagens foram descartadas por não apresentarem resultados satisfatórios durantes testes experimentais, apesar de tentativas de melhoria, que incluem, seleção de diferentes restrições permitidas na construção dos grafos e abordagens utilizando o princípio de Pareto.

A Tabela 4.5 e Figura 4.3 apresentam um exemplo para dois dias de escalonamento utilizando 3 enfermeiros.

Tabela 4.5: Exemplo de aplicação de cromossomo a um reescalonamento. As designações a serem mantidas, utilizando as chaves de decisão maiores que 0,5, estão destacadas em vermelho

	Enfermeiro 1		Enfermeiro 2		Enfermeiro 3		Enfermeiro 4	
	Dia 1	Dia 2	Dia 1	Dia 2	Dia 1	Dia 2	Dia 1	Dia 2
Chave decisão	0,3	0,7	0,7	0.3	0,8	0,8	0,7	0,7
Chave ordem	0,1	0,8	0,3	0,2	0,2	0,5	0,9	0,9

	Escalonamento anterior							
	Dia 1	Dia 2	Dia 1	Dia 2	Dia 1	Dia 2	Dia 1	Dia 2
Turno	M	T	N	L	M	M	M	T

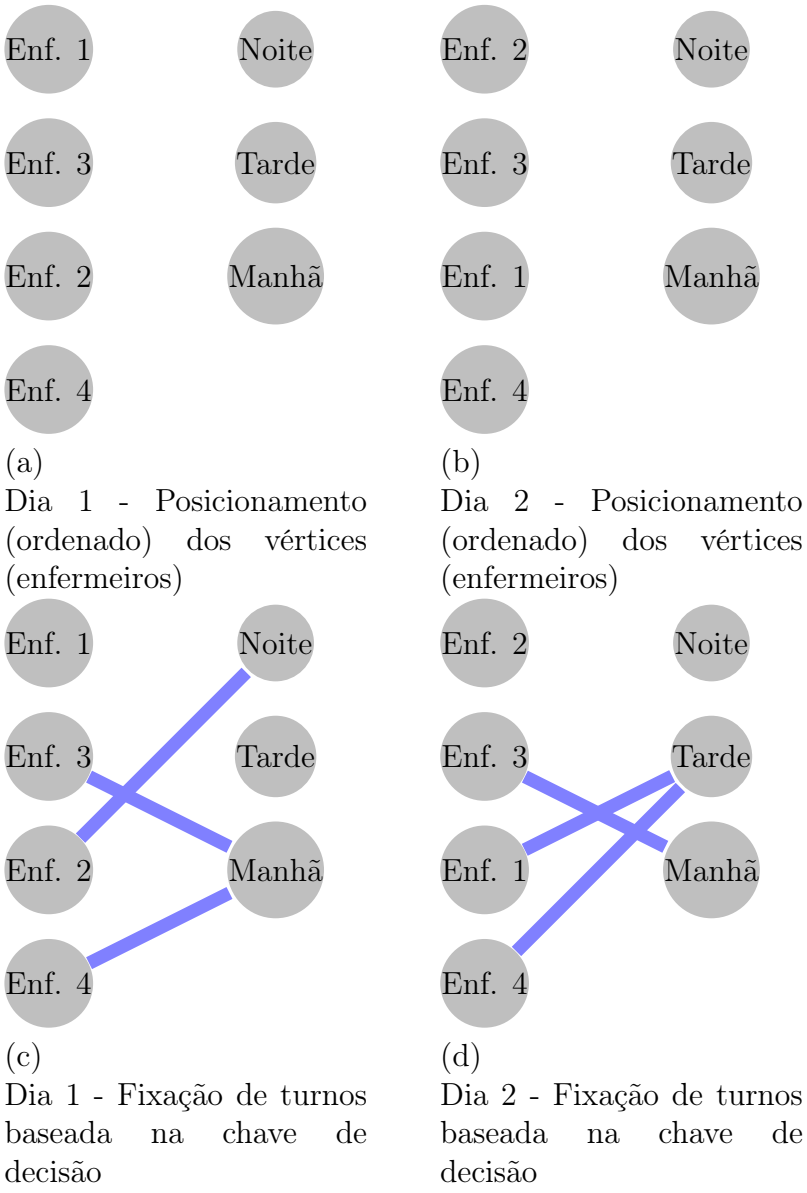
Conforme definido pelo conjunto de chaves de decisão, os valores destacados em vermelho identificam quais designações não são passíveis de mudança de turno. Pela definição das chaves de ordem, têm-se as seguintes ordens de vértices: no primeiro dia, Enfermeiro 1 $\rightarrow$ Enfermeiro 3 $\rightarrow$ Enfermeiro 2 $\rightarrow$ Enfermeiro 4, e para o segundo dia Enfermeiro 2 $\rightarrow$ Enfermeiro 3 $\rightarrow$ Enfermeiro 1 $\rightarrow$ Enfermeiro 4.

Suponha-se que para os dois dias de reescalonamento em questão, tenhamos a demanda operacional de um enfermeiro para cada turno. O grafo apresentado na Figura 4.3 representa esta situação.

Após a fixação de turnos, o restante do processo de decodificação consiste na inserção, para cada dia de reescalonamento, de arestas possíveis para cada vértice. A figura 4.4(a), mostra um exemplo onde o enfermeiro 1, no primeiro dia, possui as possibilidades de trabalhar nos turnos da tarde e manhã. Não existe aresta ligando-o ao turno da noite, pois esta ligação violaria a restrição de descanso entre turnos, pois no dia subsequente, este mesmo enfermeiro está designado para trabalhar no turno vespertino. O algoritmo HK prontamente encontra um emparelhamento máximo para este dia, adicionando a aresta destacada em verde, satisfazendo todas as tarefas do primeiro dia.

Realizado o reescalonamento do primeiro dia, é possível construir o grafo do segundo dia, mostrado pela Figura 4.4(b). Porém nota-se que não é possível encontrar um enfermeiro para realização da tarefa Noite. Os enfermeiros 1 e 3 estão fixados devido a chave de decisão, e o enfermeiro 2 não pode trabalhar em tal tarefa, pois ocorreria uma violação de restrição rígida, violando a regra de construção do grafo.

O segundo dia, por ter tarefas não satisfeitas, passará pelo procedimento de correção, onde as fixações de turno, com base nas chaves de decisão, serão descartadas. Este procedimento é, de certa forma, contraintuitivo, pois enfraquece a importância das chaves de decisão. Porém, as chaves de decisão ainda possuem papel fundamental, pois o seu descarte ocorre apenas em dias onde o seu obedecimento não é eficaz. Ademais, durante



(a) Dia 1 - Posicionamento (ordenado) dos vértices (enfermeiros)

(b) Dia 2 - Posicionamento (ordenado) dos vértices (enfermeiros)

(c) Dia 1 - Fixação de turnos baseada na chave de decisão

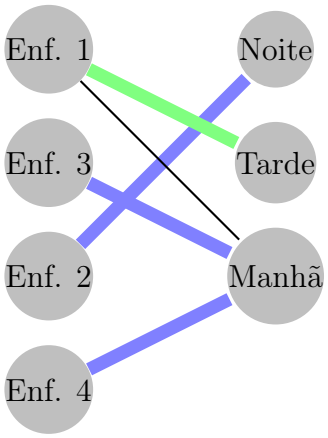
(d) Dia 2 - Fixação de turnos baseada na chave de decisão

Figura 4.3: Grafos bipartidos utilizando decodificação parcial do cromossomo

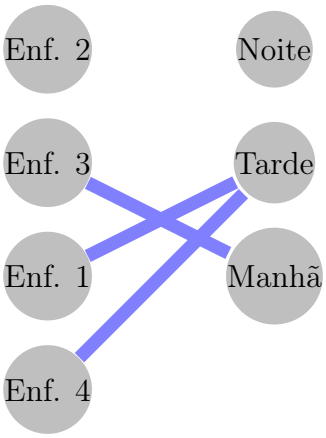
os reescalonamentos iniciais, a sua decisão é mantida, e com isso, os grafos bipartidos são mais restritos e com uma quantidade menor de alterações, característica desejada pelo problema.

A ideia que fundamenta este processo de correção consiste na característica de que quanto maior a convergência e menor quantidade de violações de demanda operacional, uma menor quantidade de procedimentos de correção será necessária, exaltando a importância das chaves de decisão. Também, é um procedimento que busca uma maior satisfação da demanda operacional, aumentando a quantidade de possibilidades de designações de turnos, em troca de uma maior quantidade de alterações.

As afirmações acima são amparadas por testes experimentais executados em uma amostra de 50 instâncias, observando-se a quantidade alterações e violações de demanda operacional. Tais testes também demonstraram que tal procedimento de correção en-

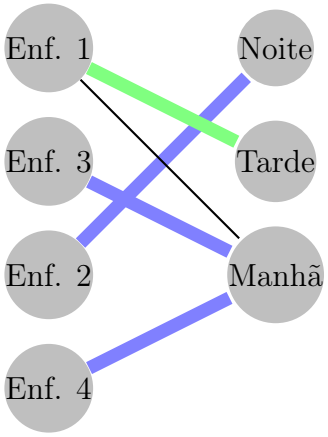


(a)
Dia 1 - Emparelhamento máximo encontrado para este dia

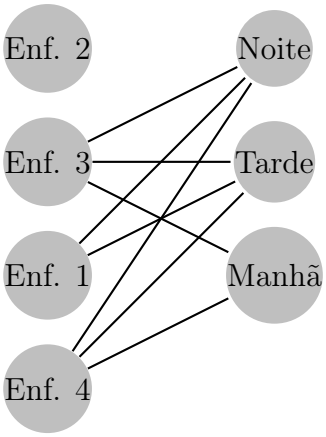


(b)
Dia 2 - Inclusão de arestas para o dia 2; não é possível realizar a tarefa Noite

Figura 4.4: Grafo bipartido onde não é possivel satisfazer todas as tarefas para o dia 2



(a)
Dia 1 - Emparelhamento máximo encontrado para este dia

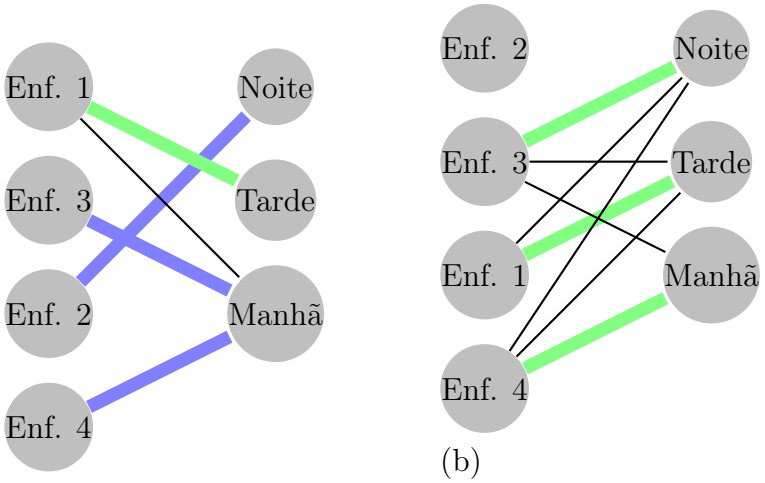


(b)
Dia 2 - Inclusão de arestas para o dia 2, descartando-se as chaves de decisão

Figura 4.5: Grafo bipartido com emparelhamento máximo encontrado para um dia e descarte das chaves de decisão no dia 2

controu resultados superiores, quando comparados com testes que não possuíam tal procedimento. Sendo assim, esta configuração foi escolhida para a execução e análise extensiva proposta por este trabalho.

A Figura 4.5 apresenta o procedimento de correção aplicado ao segundo dia, com o descarte das chaves de decisão, onde um emparelhamento máximo com satisfação de todas as tarefas é encontrado na Figura 4.6.



(a)
Dia 1 - Emparelhamento máximo encontrado para este dia

(b)
Dia 2 - Emparelhamento máximo encontrado pelo algoritmo HK, arestas destacadas em verde

Figura 4.6: Grafo bipartido com emparelhamento máximo encontrado para ambos os dias, após o procedimento de correção

4.3.3 Diversificação e intensificação

Algoritmos genéticos dependem de um bom balanceamento entre diversificação e intensificação de busca. A diversificação visa permitir boa cobertura de busca e encontro de boas regiões, enquanto a intensificação visa o aprimorar a qualidade das soluções. Com isto, os mecanismos descritos abaixo foram adotados.

De forma a aumentar o poder de diversificação do algoritmo genético proposto, uma lista tabu com memória de curta duração foi utilizada para armazenar as soluções visitadas. Durante o procedimento de cruzamento, esta lista é consultada para determinar a aceitação da solução. Caso a solução gerada esteja presente, esta é descartada, caso contrário, é inserida no grupo de soluções candidatas da próxima geração de soluções. Por tal característica, este critério de aceitação inibe a possibilidade de que existam soluções iguais durante uma ou mais gerações, a depender da duração da memória da lista tabu (*tabu tenure*), e também auxilia o algoritmo proposto na exploração do espaço de busca.

A lista tabu deve ser implementada de modo que a sua memória propicie a devida exploração do espaço de soluções, porém não deve ser um fator inibidor. Por tal, a sua memória não pode ser infinita, pois em casos de convergência aguda, o algoritmo seria incapaz de aceitar novas soluções, visto que as soluções vizinhas, geradas a partir de cruzamentos, já foram todas inseridas na lista de proibição. Ademais, é computacionalmente inviável armazenar todas as soluções visitadas durante um processo de busca, pois isto demandaria um espaço em memória exorbitante.

Dentre as soluções consideradas para o problema descrito anteriormente, duas se destacaram. A primeira consiste em, baseado no *feedback* do algoritmo, identificar

quando a lista tabu se torna um fator demasiadamente inibidor. Neste caso a lista teria sua memória reiniciada. A desvantagem desta abordagem, identificada durante testes experimentais, reside no fato de que, quando identificada a característica inibidora da lista, o simples reinício da memória não é suficiente, pois a busca, geralmente, já se encontra demasiadamente convergida em um local, impossibilitando o seu escape.

Com isto, a decisão tomada foi de fixar uma quantidade de gerações que terão sua soluções armazenadas na memória, com isto espera-se que, quando tal patamar for atingido, o algoritmo já tenha se distanciado da região de busca, propiciando uma exploração mais ampla.

A decisão de se utilizar uma lista de proibições não consiste apenas em vantagens, devido ao fato de que pode prejudicar a qualidade da solução final encontrada pelo algoritmo. Pois caso o algoritmo não tenha um potencial de intensificação adequado, o algoritmo passará a buscar novas regiões, quando poderia encontrar boas soluções caso não fosse inibido pela lista tabu.

De forma a contra balancear as considerações anteriormente mencionadas, dois mecanismos de intensificação foram utilizados. O primeiro consiste na transferência, sem alterações, de uma quantidade de soluções de boa qualidade para a próxima geração. Tal mecanismo foi proposto por Goncalves e Resende (2010), denominado elitismo. A segunda consiste em um truncamento das soluções geradas. As soluções geradas são inicialmente adicionadas a um conjunto de tamanho T vezes maior que o tamanho da população. Deste conjunto, após ordenação com base na função *fitness*, a próxima geração é completada seguindo o *ranking* obtido do conjunto. Em um exemplo prático de sua aplicação, caso falte 50 soluções para completar a próxima geração de soluções, um conjunto de 500 soluções pode ser criado, onde as 50 melhores soluções são selecionadas para sobreviverem e fazerem parte da próxima geração.

Ademais dos mecanismos e conceitos citados, o algoritmo proposto se reinicia aleatoriamente quando, após determinada quantidade de gerações, não consegue melhorar a qualidade de sua melhor solução. Este mecanismo está de acordo com a ideia proposta por Goncalves e Resende (2010), onde o algoritmo se beneficia de convergências forçadas e rápidas, para encontrar um ótimo local e reiniciar a busca em outro local.

Desta forma, o uso de reinícios aleatórios deve se utilizar tanto de características de exploração, quanto de intensificação, para seu bom funcionamento. Afinal, o reinício prematuro pode prevenir o encontro de melhores soluções dentro do espaço de busca atual.

Outros mecanismos de exploração e intensificação, critérios de aceitação baseados na qualidade da solução e mutação acentuada, foram testados, porém descartados em fases experimentais, devido ao fato de ou não apresentarem melhora significativa ou por acarretarem em um tempo de execução muito maior, sem ganho de qualidade observável.

De forma a melhor esclarecer o posicionamento e funcionamentos destes mecanismos citados, o Algoritmo 3 apresenta o pseudo-código para uma iteração do algoritmo, ou

seja, o período entre a criação da população inicial e o seu término, seja por encontro de solução ótima ou por atingir o limite de gerações sem melhoria.

A chamada principal do algoritmo invoca este procedimento apresentado enquanto o tempo limite não for atingido ou enquanto a solução retornada não for a solução ótima buscada, caso conhecida. Dentro da chamada principal, as soluções retornadas são armazenadas de forma ordenada para que, no caso de a solução ótima não ter sido encontrada, ainda seja possível ter acesso a melhor solução global (dentre todas as iterações - reinícios). Esta afirmação é ainda mais válida para os casos da classe difícil, onde o valor ótimo não é conhecido.

Algoritmo 3 Algoritmo de uma iteração (reinício) do algoritmo proposto - RKGA-HK

ITERACAO-RKGA-HK()

```
1  TamanhoPopulacao ← Tamanho da População
2  ListaTabu ← NULL
3  TabuTenure ← Tamanho máximo da ListaTabu
4  FatorTruncamento ← TamanhoPopulacao × 5
5  GeracaoAtual ← Criar soluções aleatórias
6  MelhorAtual ← Melhor Solução desta iteração (reinício)
7  SemMelhoraMaximo ← Gerações sem melhoria da solução atual (critério de reinício)
8  SemMelhora ← 0
9  repeat
10     ProximaGeracao ← NULL
11     ProximaGeracao ← Adicionar soluções elite da geração atual sem alteração (migração)
12     ProximaGeracao ← Adicionar soluções mutantes (mutação)
13     CROSSOVER()
14     SemMelhora++
15     if Melhor solução da GeracaoAtual for superior à MelhorAtual
           SemMelhora ← 0
           MelhorAtual ← MelhordaGeracaoAtual

16     if Tamanho da ListaTabu ≥ TabuTenure
           Reiniciar ListaTabu

17 until (SemMelhora ≤ SemMelhoraMaximo)(Descarte populacional (reinício))
18 retornar MelhorAtual
```

CROSSOVER()

```
1  NovasSolucoes ← NULL
2  repeat
3      pai1 ← Seleção de solução por torneio binário
4      pai2 ← Seleção de solução aleatória
5      novaSolucao ← Gerar nova solução por cruzamento
6      if novaSolucao não está na ListaTabu
           Adicione novaSolucao na NovasSolucoes
           Adicione novaSolucao na ListaTabu
7  until (Tamanho da ProximaGeracao for inferior ao FatorTruncamento)
8  Complete a ProximaGeracao com as melhores soluções presentes em
   NovasSolucoes (truncamento)
9  Substitua a GeracaoAtual por ProximaGeracao (substituição populacional)
```

5 Resultados e observações

Este capítulo apresenta os parâmetros do algoritmo genético e de execução, assim como os resultados observados e análise comparativa.

Parte da análise necessita de conhecimento e interpretação de *boxplots*, uma breve explanação das características deste tipo de gráfico é apresentada em cada análise. Informações mais completas podem ser encontrados no estudo de Williamson (1989).

5.1 Considerações na parametrização

Algoritmos genéticos, conforme citado anteriormente, são altamente parametrizáveis. As diferentes combinações de parâmetros afetam diretamente os resultados finais, portanto, encontrar uma boa combinação de parâmetros é essencial para obter bons resultados (Jong, 2007).

Para a definição dos parâmetros, foi realizada uma análise observacional dos resultados em uma parcela das instâncias. Estas execuções analisadas são doravante chamadas de pré-execução. Vale ressaltar que diversas combinações de parâmetros foram testadas, as principais observações e seu impacto nos resultados são listadas a seguir.

- **Tamanho da população:** Apesar de estudo de referência na área de RKGAs (Goncalves e Resende, 2010) indicar que o tamanho da população deve ter uma relação com o tamanho do cromossomo, esta não foi a abordagem utilizada por este estudo. Esta decisão foi tomada pelo fato de que, as instâncias possuem uma grande diferença de tamanho, portanto, os cromossomos também refletem essa diferença. Foi observado que utilizando uma relação de tamanho, o algoritmo poderia ter alguma melhora, porém, esta vantagem positiva não era algo definitivo, pois produzia bons resultados em apenas algumas instâncias. Ademais, a manutenção de um tamanho fixo de população, tinha o efeito positivo de ter melhores tempos de execução, proporcionando ainda resultados satisfatórios.

- **Duração da memória Tabu (*Tabu tenure*):** De forma a controlar a diversidade da população e incentivar a exploração de novos espaços de busca, de início, foi considerado utilizar uma memória que mantêm todas as soluções visitadas. Prontamente esta abordagem gerou dois efeitos negativos, o primeiro consiste na vasta quantidade de memória computacional demandada para instâncias cuja execução era prolongada e segundo foi a incapacidade do algoritmo genético conseguir produzir novas soluções a partir do operador de cruzamento. Este segundo ponto negativo é efeito direto da lista tabu e da convergência do algoritmo, que apesar da existência de mutantes, não era capaz de criar soluções fora da região do espaço de busca atual. Em contraste, quando a duração de memória foi fixada em uma geração, o efeito explorador do algoritmo não foi manifestado, pois, apesar de proporcionar soluções distintas em uma única geração, não era o suficiente para o algoritmo buscar novos espaços de busca, causando revisitação precoce de soluções.

- **Operador de seleção**: Novamente, apesar da sugestão de utilizar uma abordagem elitista na seleção (Gonçalves e Resende, 2010), esta abordagem não produzia resultados consistentes e em muitos casos, encontrava dificuldade em encontrar soluções viáveis. Portanto, buscando ainda certo grau de favoritismo, o operador de seleção consiste em uma seleção por torneio binária para uma solução e escolha aleatória para outra. Esta abordagem se mostrou mais consistente e apresentou os melhores resultados durante a pré-execução.

- **Fator de truncamento**: O fator de truncamento neste algoritmo foi utilizado com o propósito de intensificação, porém, em testes onde o fator de truncamento era muito alto (ex.: 5, 10 $\times$ Tamanho da população), o efeito de convergência era muito acentuado e não produzia bons resultados. Além disso, por descartar uma quantidade considerável de soluções, há também um tempo computacional não aproveitado.

- **Valor de corte**: Foram testados diferentes valores para o valor de corte. Inicialmente, foi utilizado um valor comum, de 0,5, e foi observado que este valor apresentava bons resultados em instâncias cujo o reescalonamento se iniciava na segunda metade da escala original. Na busca de melhores resultados, o valor de corte foi continuamente diminuído até chegar no valor de 0,2, lembrando que quanto menor o valor de corte, maior será a probabilidade de manutenção das designações anteriores. Porém, ao analisar os resultados utilizando o valor de 0,2 foi observado que os bons resultados, encontrados nas instâncias com reescalonamento na segunda metade, foram perdidos, inclusive foi observado uma grande dificuldade de encontro de soluções viáveis. Outrossim, este valor de corte se mostrou interessante e apresentou bons resultados para instâncias cujo reescalonamento se inicia na primeira metade da

escala. Uma possibilidade desta característica pode estar relacionada a quantidade de designações que serão alteradas dependendo do início do reescalonamento.

- **Soluções elites e mutantes**: A porcentagem utilizada por este estudo seguiu a sugestão proposta por Gonçalves e Resende (2010).

- **Reinícios aleatórios**: Com o intuito de proporcionar uma diversificação e exploração em novos espaços de busca, foram considerados diferentes métodos de reinício. Buscando preservar parte da convergência, foi considerado manter parte das soluções elites, mesmo após o reinício do algoritmo. Observando os resultados, as soluções mantidas possuíam qualidade muito superior às novas soluções e causavam uma convergência em seu entorno, não causando o efeito desejado de busca de novos locais. Portanto, foi decidido pela completa reinicialização (descarte) das soluções. A cada reinício a melhor solução é armazenada, sendo comparada com a solução anteriormente encontrada, portanto, mesmo com o procedimento de reinício de população, o algoritmo ainda retorna a melhora solução encontrada, não importando em qual iteração ela foi encontrada. Ademais da característica de descarte, a quantidade de gerações para o reinício também foi analisada. Uma quantidade muito grande de gerações (ex.: 50) não produzia ganhos observáveis, apenas demandando um tempo maior de execução, enquanto uma quantidade pequena (ex.:10), tinha o efeito de reinício, quando ainda havia potencial de melhoria.

5.2 Parâmetros do algoritmo proposto e de execução

Parâmetros do algoritmo:

1. **Restrição rígida:** Número de violações (NV) da demanda operacional (única possível, dada a modelagem proposta)
2. **Restrição flexível**: Número de alterações (NA) em relação a escala original
3. **Função *fitness*:** $NA + NV \times$ Número de enfermeiros
4. **Valor de corte**
    - 0,25 caso o reescalonamento se inicie na primeira metade da escala.
    - 0,50 caso o reescalonamento se inicie na segunda metade da escala.
5. **Tamanho da população:** 100.
6. **Duração da memória da lista tabu (*Tabu tenure*):** 5 gerações.
7. **Fator de truncamento:** 3 $\times$ Tamanho da população.

8. **Soluções elites:** 25% do tamanho da população.
9. **Soluções mutantes:** 10% do tamanho da população.
10. **Reinício aleatório:** 20 gerações sem melhoria da melhor solução atual (não se confunde com a melhor solução global, que é a melhor solução encontrada até o momento).

Por utilizarem fatores aleatórios, RKGAs podem ter alta variabilidade na qualidade das soluções obtidas. Deste modo, trinta execuções, utilizando diferentes sementes aleatórias, foram feitas, e a média das soluções obtidas foi utilizada para análise.

Ambos, RKGAs e o resolvedor CPLEX, possibilitam a utilização de múltiplas *threads*, porém, para melhor comparação, os resultados e análise apresentados na seção a seguir, foram obtidos utilizando uma única *thread* para ambos os casos.

5.3 Resultados experimentais e análise

Nesta seção, são apresentados os resultados computacionais obtidos. A análise destes resultados é realizada em relação a média da qualidade das soluções obtidas e o tempo computacional despendido para sua execução. Dadas as características da base de testes utilizada, os resultados serão analisados por agrupamentos de instâncias, considerando a dificuldade relativa encontrada pelo método exato.

Os resultados e análises serão divididos em três partes, a primeira irá conter os resultados numéricos das classes fácil e média, ou seja, a quantidade de alterações encontrada por ambas as abordagens em conjunto com uma análise comparativa. Os resultados numéricos das instâncias da classe difícil estão apresentadas na segunda subseção, assim como um comparativo utilizando diferentes tamanhos de população. Uma visualização gráfica auxilia a análise desta comparação. A terceira subseção apresenta uma análise gráfica das características de convergência e da capacidade resolutiva do algoritmo proposto, utilizando como métrica, a quantidade de gerações em sua execução.

5.3.1 Resultados numéricos - Classes Fácil e Média

Na Tabela 5.1, os resultados encontrados tanto pelo método exato (CPLEX) quanto pelo algoritmo genético (RKGA-HK) são apresentados. A coluna "Alt." representa a quantidade de alterações e a coluna Tempo indica o tempo de execução em segundos. Subdivididos em casos, melhor, pior e média, os resultados foram obtidos pela média dos resultados, portanto, nos casos pior e melhor, os piores e melhores resultados, respectivamente, foram somados e divididos pela quantidade de instâncias. Na média dos casos, todos os resultados foram somados e divididos pelo total de execuções.

Tabela 5.1: Resultados computacionais das classes fácil e média, de acordo com a dificuldade

	CPLEX		RKGA-HK					
			Melhor		Pior		Média	
Classe	Alt.	Tempo	Alt.	Tempo	Alt.	Tempo	Alt.	Tempo
Fácil	**56,68**	509,60	56,85	117,42	57,30	252,92	57,06	185,21
Média	**57,73**	1030,76	57,88	210,27	58,18	365,42	58,06	278,28

Na classe de instâncias fáceis, o algoritmo genético se mostrou competitivo tanto em termos de qualidade de solução, quanto em velocidade de execução. A diferença no pior e melhor caso são praticamente insignificantes, onde o algoritmo genético na média de execuções em ambos os casos, encontrou soluções com menos de uma alteração em relação ao método exato. Em contraste com essa diferença, o algoritmo genético possui um tempo de execução muito inferior, levando aproximadamente um quarto do tempo no melhor caso e metade do tempo no pior caso.

Considerando as instâncias de média dificuldade, curiosamente, o algoritmo genético se comportou melhor do que nas instâncias de dificuldade fácil, demonstrado pela diferença ainda menor entre seus resultados e os resultados ótimos encontrados, pois, apesar de, na média, ambos os casos obterem reescalonamentos com diferença inferior a uma alteração do resultado ótimo, no melhor caso, nas instâncias fáceis, o algoritmo genético obteve uma diferença de 0,17 alterações, enquanto nas instâncias médias, a diferença foi de 0,15 alterações. Apesar de ambas serem praticamente iguais, no pior caso estas diferenças aumentam, pois nos casos fáceis e médios, a diferença foi de 0,62 e 0,45 alterações, respectivamente.

Ressalta-se que, mesmo nos piores resultados, o algoritmo RKGA-HK obteve resultados com diferenças inferiores a uma alteração, demonstrando que, em ambas as classes, o algoritmo foi capaz de encontrar soluções ótimas ou sub-ótimas de grande qualidade em uma fração do tempo utilizado pelo método exato.

Tabela 5.2: Tabela de resultados utilizando os melhores casos - Classe Fácil

Classe Fácil	
Nº de resultados ótimos ou iguais/Total de instâncias	Porcentagem
37/40	92,5%

Instâncias cujo ótimo não foi encontrado pelo algoritmo RKGA-HK

Instância	Resultado CPLEX	Melhor Resultado RKGA-HK
p10-1st-abs20-t8-18.dat	**85**	88
min-2nd-abs10-t6-6.dat	**52**	54
min-2nd-abs10-t10-7.dat	**66**	68

Nas execuções das instâncias da classe fácil (Tabela 5.2), o algoritmo genético encontrou 92,5% de resultados ótimos. O algoritmo proposto não encontrou o resultado ótimo

para três instâncias, porém, nota-se que apesar de sub-ótimas, as soluções encontradas possuem boa qualidade, se distanciando no máximo em três alterações.

Tabela 5.3: Tabela de resultados utilizando os melhores casos - Classe Média

Classe Média	
Nº de resultados ótimos ou iguais/Total de instâncias	Porcentagem
38/40	95%

Instâncias cujo ótimo não foi encontrado pelo algoritmo RKGA-HK

Instância	Resultado CPLEX	Melhor Resultado RKGA-HK
min-1st-abs20-t6-9.dat	**84**	89
min-1st-abs20-t6-13.dat	**60**	61

Curiosamente, conforme a tabela 5.3., novamente, o algoritmo genético proposto se comportou ligeiramente melhor nas instâncias de dificuldade média, encontrando o resultado ótimo em 95% das instâncias. Porém, comparando os resultados não ótimos encontrados, nota-se que para as instâncias da classe fácil, os resultados são iguais ou inferiores a 3, enquanto para as instâncias da classe média, as diferenças são iguais ou inferiores a 5, ocorrendo um ligeiro aumento na quantidade de alterações nos reescalonamentos.

5.3.2 Resultados numéricos - Classe Difícil

Nas instâncias da classe difícil (Tabela 5.4), a eficiência não se manteve, encontrando apenas 37,5% de resultados ótimos. Porém na maior parte dos casos, a distância do resultado ótimo foi igual ou inferior cinco alterações, demonstrando boa capacidade de encontrar soluções de boa qualidade. Porém, nota-se que o resultado encontrado para a instância "min-1st-abs10-t10-5.dat", o melhor resultado encontrado está 34 alterações pior que o resultado encontrado pelo método exato. Esta diferença foi considerada alta, por isso, motivou uma bateria de execuções específica para as instâncias desta classe, com intuito de avaliar a diferença de desempenho.

Imaginando que tal comportamento poderia ser devido ao tamanho da população do algoritmo genético, que conforme Gonçalves e Resende (2010) e Gonçalves *et al* (2014), indica a existência de uma relação entre o tamanho do problema e o tamanho do cromossomo utilizado, uma nova bateria de execuções foi realizada utilizando o dobro de tamanho da população original, ou seja, nesta bateria de execuções, o tamanho da população foi fixado em duzentas soluções. O resultado comparativo é apresentado na Tabela 5.5.

Utilizando um tamanho maior de população, o algoritmo proposto encontra uma maior quantidade de soluções iguais ao do método exato, dentro do tempo limite. E em alguns casos foi capaz de superar tais resultados, conforme apresentado na Tabela 5.6.

Tabela 5.4: Tabela de resultados utilizando os melhores casos - Classe Difícil

Classe Difícil	
Nº de resultados ótimos ou iguais/Total de instâncias	Porcentagem
15/40	37,5%

Instâncias cujo ótimo não foi encontrado pelo algoritmo RKGA-HK

Instância	Resultado CPLEX	Melhor Resultado RKGA-HK
min-1st-abs10-t10-8.dat	**73**	78
min-1st-abs10-t8-11.dat	**59**	61
min-1st-abs10-t8-12.dat	**50**	54
min-1st-abs20-t10-13.dat	**110**	122
min-1st-abs20-t6-12.dat	**74**	75
min-1st-abs10-t10-6.dat	**64**	68
min-1st-abs10-t8-4.dat	**70**	77
min-1st-abs10-t10-3.dat	**55**	58
min-1st-abs20-t10-16.dat	**120**	127
min-1st-abs10-t8-13.dat	**55**	57
min-1st-abs20-t6-14.dat	**97**	110
min-1st-abs10-t10-4.dat	**64**	68
min-1st-abs10-t8-17.dat	**52**	55
min-1st-abs20-t8-12.dat	**112**	128
min-1st-abs10-t6-17.dat	**59**	61
min-1st-abs10-t10-12.dat	**82**	84
min-1st-abs20-t10-20.dat	**120**	126
min-1st-abs10-t6-14.dat	**47**	48
min-1st-abs20-t8-17.dat	**105**	118
min-1st-abs20-t8-14.dat	**108**	116
min-1st-abs10-t6-5.dat	**65**	76
min-1st-abs10-t10-10.dat	**71**	72
min-1st-abs10-t8-9.dat	**95**	112
min-1st-abs10-t10-5.dat	**88**	122
min-1st-abs10-t8-10.dat	**54**	57

Tabela 5.5: Resultados computacionais da classes difícil, com variação no tamanho da população de 100 para 200.

CPLEX		RKGA-HK						
			Melhor		Pior		Média	
Alt.	Tempo	População	Alt.	Tempo	Alt.	Tempo	Alt.	Tempo
71,98	1524,51	100	76,48	1201,47	89,65	1203,20	80,89	1202,14
71,98	1524,51	200	72,93	1255,31	80,08	1308,10	76,94	1245,21

A Figura 5.1 apresenta o gráfico de convergência para a instância onde o algoritmo proposto superou o método exato de forma mais contundente, no caso, a instância "min-1st-abs10-t8-1.dat". A figura apresenta a convergência de ambas as execuções, com tamanhos de população diferentes.

Tabela 5.6: Tabela de resultados com tamanhos de população diferentes - Classe Difícil

Classe Difícil		
População	Nº de resultados iguais/Total de instâncias	Porcentagem
100	15/40	37,5%
200	**25/40**	**62,5%**

Instâncias com RKGA-HK superior ao CPLEX no tempo limite

Instância	Resultado CPLEX	RKGA-HK (população=200)
min-1st-abs10-t10-12.dat	82	**81**
min-1st-abs20-t10-20.dat	120	**119**
min-1st-abs10-t6-14.dat	47	**46**
min-1st-abs10-t8-9.dat	95	**94**
min-1st-abs10-t8-1.dat	84	**77**

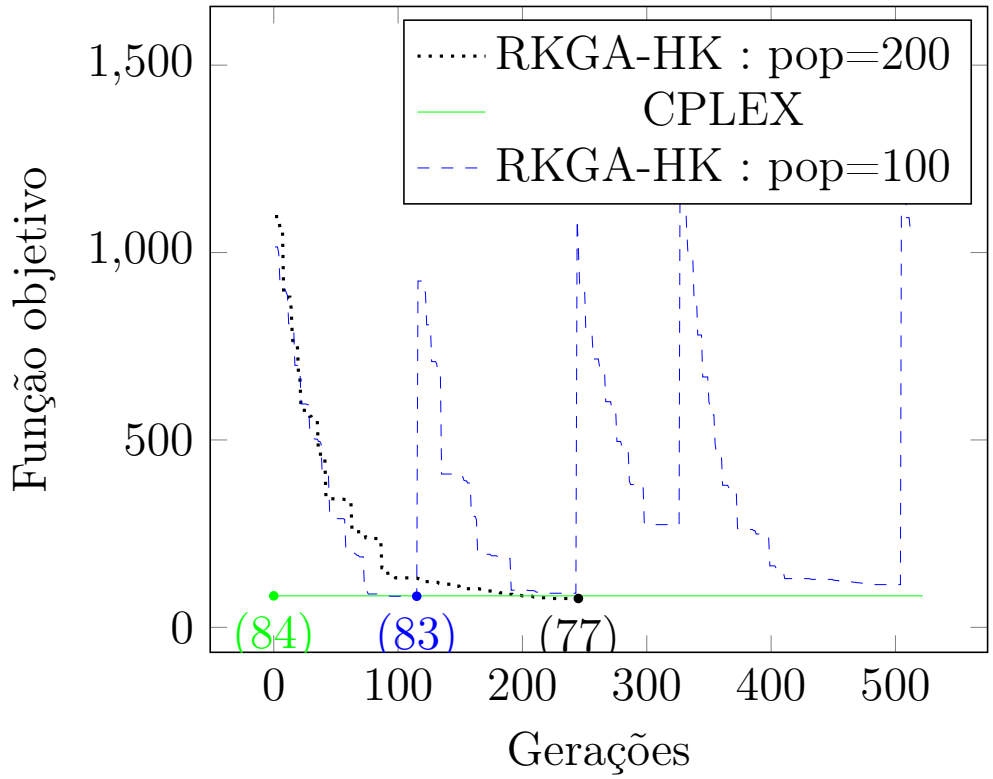


Figura 5.1: Gráfico de convergência da instância "min-1st-abs10-t8-1.dat", em azul o algoritmo proposto com tamanho de população igual 100, em preto tamanho igual a 200 e em verde o resultado encontrado pelo método exato

Apesar da execução com população de tamanho 100 também superar o resultado exato, essa configuração não se mostra robusta, conforme notado na Figura 5.1, pois nos reinícios subsequentes, o algoritmo apenas encontra resultados razoavelmente distantes do resultado encontrado pelo método exato. Em contra-partida, apesar de não conter reinícios aleatórios em suas execuções, devido ao tempo limite, o algoritmo proposto com população de tamanho duzentos supera o resultado encontrado pelo método exato.

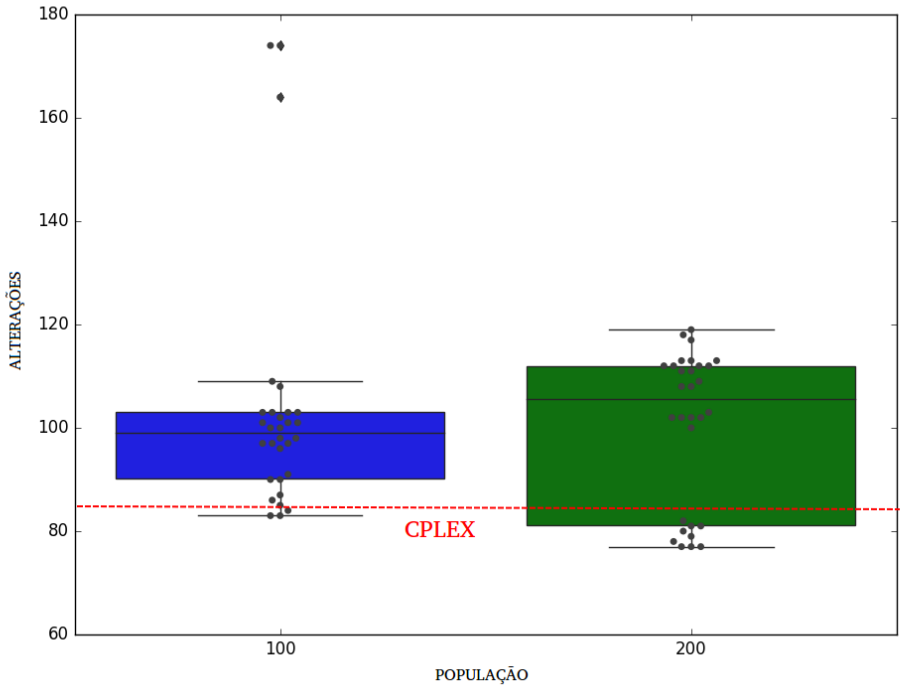


Figura 5.2: Gráfico de distribuição (*boxplot*) da instância min-1st-abs10-t8-1, com as diferentes populações

Utilizando esta mesma instância, com o gráfico de distribuição (*boxplot)* apresentado na Figura 5.2, algumas informações podem ser extraídas:

1. Mediana: Representada pela linha horizontal dentro das "caixas"(*boxes*), nota-se que a mediana dos resultados encontrados pelo algoritmo utilizando a população original foi superior, tendo valor inferior à mediana encontrada pela execução com população dobrada. Ademais, nota-se que ocorreu uma maior concentração de resultados ao redor da mediana no primeiro caso, acarretando inclusive em uma diminuição no tamanho dos quartis.

2. *Outliers:* A presença de *outliers*, ou seja, de resultados que fogem do comportamento padrão, são representados por pontos inferiores ou superiores às linhas horizontais paralelas às "caixas". Neste exemplo, a presença destes tipos de resultados é encontrada apenas na execução utilizando a população original, indicando certa inconsistência quando comparada à execução com população dobrada.

3. Dispersão: A dispersão dos dados pode ser representada pelo intervalo entre quartis, encontrado pela diferença entre o terceiro quartil e o primeiro quartil, em outras palavras, o tamanho da "caixa". Conforme dito anteriormente, a concentração de resultados ao redor da mediana, contribuiu para a redução do tamanho no primeiro caso, com a população original.

4. Simetria: Em ambas as execuções, a mediana ficou acima da região central da "caixa", demonstrando uma tendência negativa dos resultados, ou seja, a tendência de outros resultados estarem mais próximos do terceiro quartil.

5. Mínimos: Os melhores resultados são representados pelos pontos entre o primeiro quartil e a linha horizontal inferior. Nota-se que na totalidade destes resultados, a execução com população dobrada foi superior, demonstrando ganho de qualidade, apesar das informações citadas anteriormente.

6. Análise geral: Apesar de informações discrepantes em relação a consistência do algoritmo, resultado da análise da simetria e da posição da mediana, a execução do algoritmo RKGA-HK com população dobrada encontrou resultados superiores quando comparados com a execução utilizando o tamanho original da população.

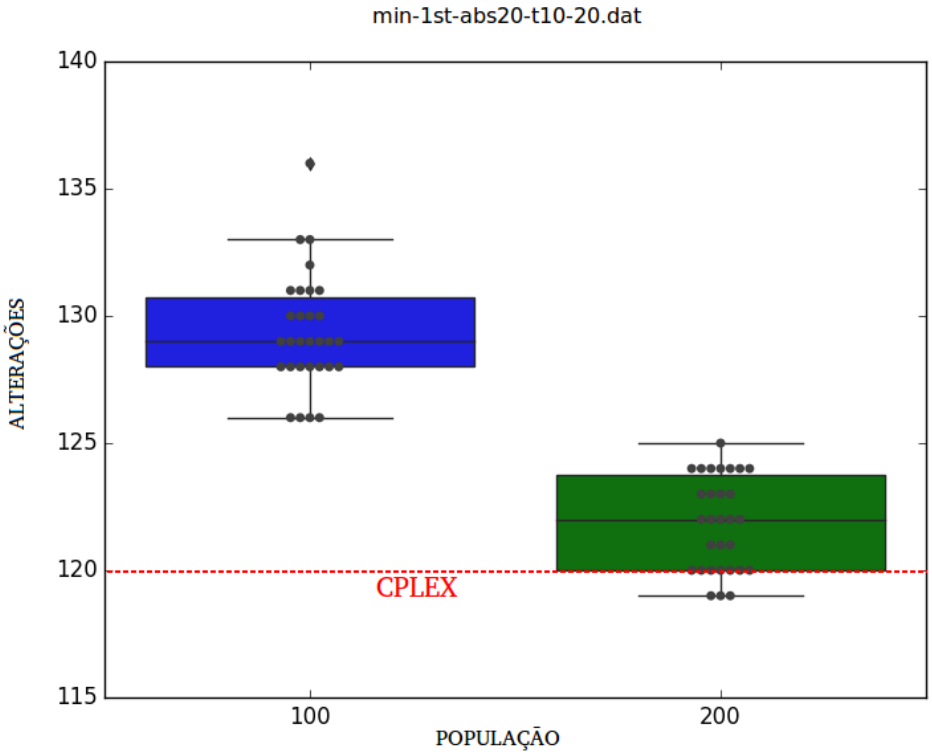


Figura 5.3: Gráfico de distribuição (*boxplot*) da instância min-1st-abs20-t10-20, com as diferentes populações

Um outro comparativo utilizando *boxplot*, apresentado na Figura 5.3, têm as suas informações assim analisadas:

1. Mediana: Nota-se, primeiramente, que a posição da mediana dos resultados com a população original ficou superior à mediana dos resultados utilizando população dobrada, indicando resultados piores de forma geral.

2. *Outliers:* Neste exemplo, a presença destes tipos de resultados é encontrada apenas na execução utilizando a população original, indicando ligeira inconsistência quando comparada à execução com população dobrada.

3. Dispersão: Diferentemente do exemplo anterior, neste exemplo, o tamanho das "caixas", foi mais próximo, com ligeira vantagem para a execução com tamanho original, indicando consistência em ambas as execuções.

4. Simetria: Nota-se que em ambos os casos, a mediana se posicionou ou em uma região central da caixa ou mais próxima do primeiro quartil, indicando uma tendência positiva de outros resultados.

5. Resultados: Na totalidade dos pontos, a execução com tamanho dobrado de população superou a execução com tamanho original, onde todos os pontos (resultados) da execução com população 200, estão abaixo da linha horizontal inferior do primeiro *boxplot*.

Analisando ambos os gráficos anteriores, há certa dúvida sobre a real qualidade de se utilizar a população dobrada, pois apesar dos ganhos em termos de qualidade de resultado, equiparando ou superando o método exato no tempo limite, o fator tempo inibe uma característica essencial do algoritmo, a da diversificação, fato que pode ter influenciado a ligeira inconsistência encontrada na Figura 5.2.

5.4 Análise de convergência e diversificação

Nesta seção, é analisada a capacidade de convergência do algoritmo, assim como sua capacidade de diversificação, *escape* de ótimos locais e capacidade de encontro de soluções viáveis. Todas essas características são essenciais na busca por um algoritmo genético robusto.

Tabela 5.7: Gerações para encontro de solução viável

Classe	0 Gerações	<50 gerações	>50 gerações
Fácil	1021	179	0
Médio	1131	69	0
Difícil	452	511	237

A Tabela 5.7 apresenta uma característica interessante do algoritmo, a facilidade de encontrar soluções viáveis. Nas instâncias das classes fácil e média, considerando todas as 1200 execuções, o algoritmo nunca necessitou mais de 50 gerações para encontrar uma solução viável. A maior parte , 85% (1021) e 94% (1131), para as classes fácil e média, respectivamente, foram encontradas logo antes da primeira geração, ou seja durante a criação das soluções iniciais do algoritmo proposto, demonstrando a capacidade resolutiva

do algoritmo. Apesar da dificuldade aumentada, na classe difícil, 80% (452+511) das execuções encontraram uma solução viável com menos de 50 gerações.

Esta característica é de suma importância, pois o algoritmo pode utilizar suas características na busca por otimização, e não viabilidade. Durante testes experimentais, conforme dito anteriormente, outras configurações e processos de decodificação foram experimentados, porém, notou-se que o algoritmo gastava demasiado tempo para encontrar soluções viáveis, restando pouco tempo para de fato realizar a otimização das soluções.

Tabela 5.8: Gerações até o melhor resultado

Classe	Qtde. Gerações		
	Melhor caso	Pior caso	Média
Fácil	36.42	64.425	120.95
Médio	43.77	70.54	111.7
Difícil	50.45	136.65	269.52

Considerando a capacidade de convergência, que, conforme citado na Seção 4, é algo desejado para contra-balancear o processo de exploração do algoritmo, pelas informações da Tabela 5.8, nota-se que este efeito desejado foi satisfeito, visto que o algoritmo rapidamente encontra o seu melhor resultado, utilizando uma quantidade pequena de gerações. Lembrando que isto foi desejado, pois utilizado em conjunto com reinícios aleatórios, o algoritmo deve ser capaz de encontrar diferentes regiões promissoras. Isso é evidenciado pela coluna pior caso da tabela analisada, onde, incapaz de encontrar melhores soluções, o algoritmo se reinicia para escapar do ótimo local onde se encontra. Porém, tal procedimento demanda uma quantidade maior de gerações em seu processo de busca.

A Figura 5.4 apresenta um exemplo de execução de uma instância de dificuldade média, onde o algoritmo, apesar de alcançar bons resultados em sua busca, necessitou de dois reinícios para encontrar o resultado ótimo, representado pela linha em verde.

Apesar da eventual necessidade de reinícios aleatórios, em muitas execuções o algoritmo encontra o resultado ótimo na primeira iteração. Esse comportamento foi comumente observado nas instâncias das classes fácil e média.

Um exemplo disto é apresentado nas Figuras 5.5 e 5.6, onde o algoritmo encontra o resultado ótimo rapidamente, aproximadamente 40 e 80 gerações, respectivamente, sem a necessidade de reinício da população.

A Tabela 5.9 apresenta as frequências com que o algoritmo encontra o resultado ótimo, baseado na quantidade de reinícios utilizados.

Novamente, nota-se que o algoritmo se comportou de forma melhor nas instâncias médias, encontrando o resultado ótimo na primeira iteração em 85% das execuções, contrastando com 75% das execuções nas instâncias da classe fácil. Nota-se também que uma porcentagem das execuções necessitaram mais de dois reinícios, 20% e 10%,

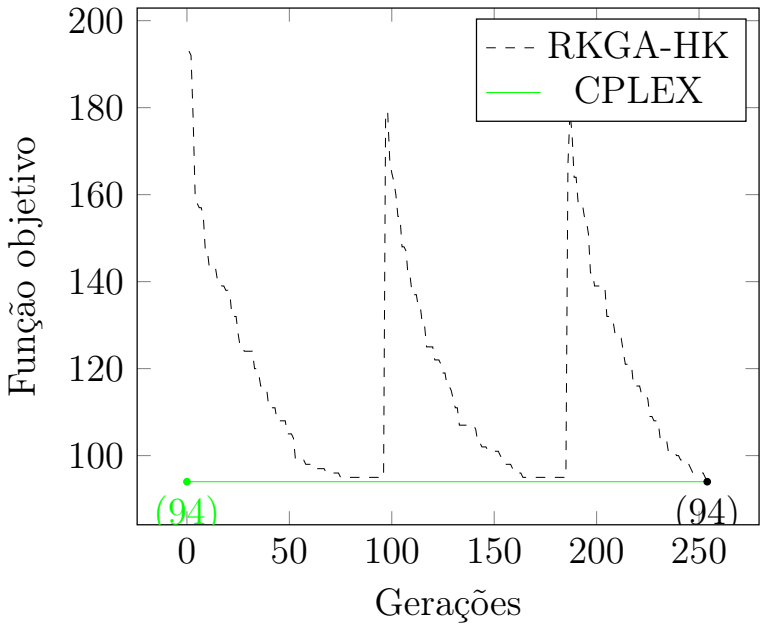


Figura 5.4: Exemplo de instância do caso médio, que necessitou reinício para encontro de resultado ótimo

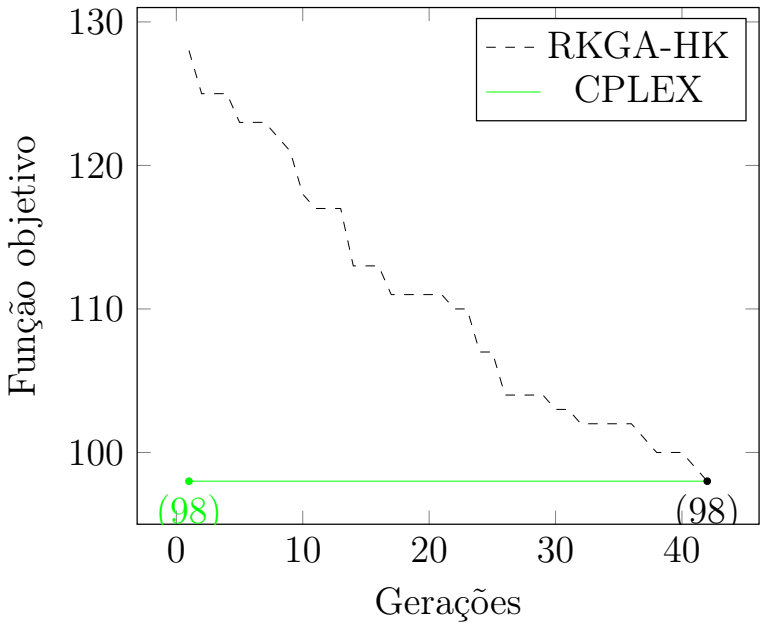


Figura 5.5: Exemplo de instância de dificuldade fácil, que não necessitou reinício para encontro de resultado ótimo

Tabela 5.9: Número de reinícios para encontro da melhor solução, ótima ou não

Classe	0 reinícios	1 reinício	2 reinícios	>2 reinícios
Fácil	71%	7%	3%	20%
Médio	85%	5%	0%	10%

nas classes média e fácil, respectivamente. Nestas porcentagens, incluem os casos onde o resultado ótimo não foi encontrado, por isso a quantidade de reinícios foi maior. A

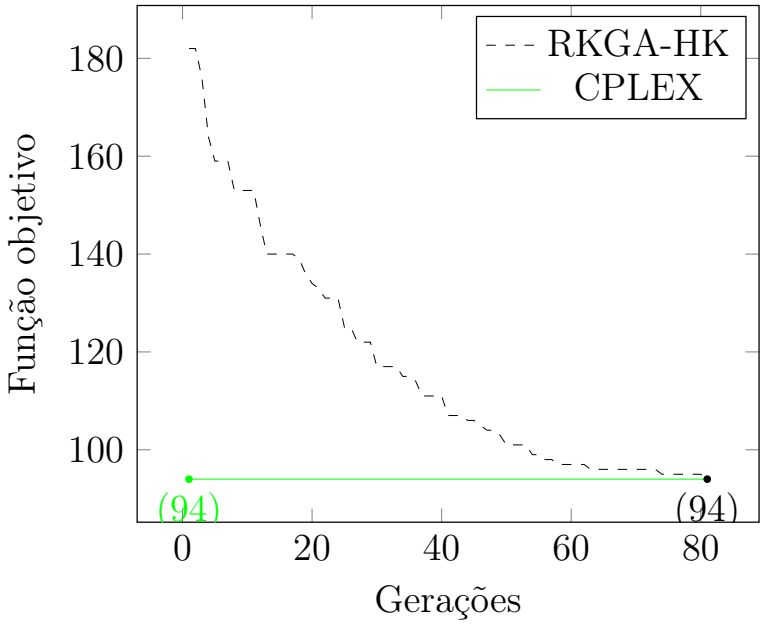


Figura 5.6: Exemplo de instância de dificuldade média, que não necessitou reinício para encontro de resultado ótimo

classe difícil não foi tabelada, pois a quantidade de reinícios é sempre a maior possível considerando o tempo limite de execução.

Um exemplo da classe fácil, onde o resultado ótimo não foi encontrado, mesmo após seguidos reinícios é apresentado na Figura 5.7. Nota-se que o algoritmo se aproxima do resultado ótimo, porém não encontra tal resultado.

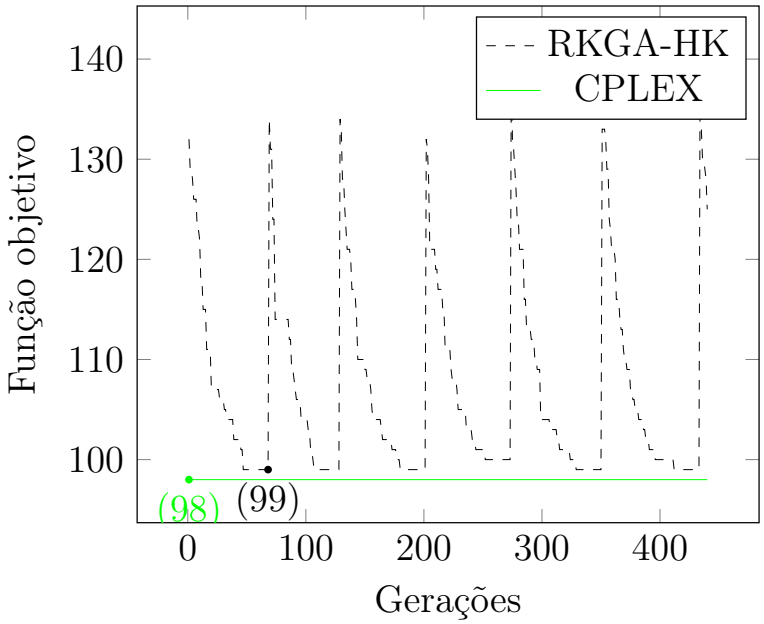


Figura 5.7: Exemplo de instância de dificuldade fácil, sem encontro de resultado ótimo, mesmo com reinícios aleatórios

Nas instâncias da classe difícil, o critério de parada foi unicamente o tempo de execução. Portanto, em alguns casos o algoritmo encontra o melhor resultado, porém pelo desconhecimento de otimalidade, reinicia continuamente a sua população, encontrando muitas vezes o mesmo resultado. Tal comportamento é visualizado na Figura 5.8, onde o algoritmo encontra o resultado ótimo em múltiplas iterações (reinícios).

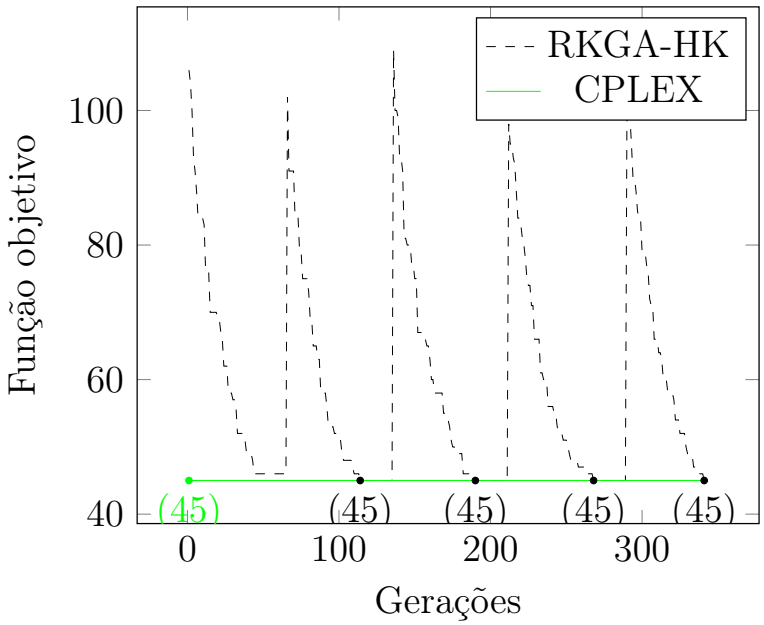


Figura 5.8: Exemplo de instância da classe difícil, onde, em múltiplas iterações, o algoritmo encontra o melhor resultado

5.4.1 Análise Estatística

De forma a melhorar corroborar os resultados analisados anteriormente, foi realizado um teste estatístico. O teste escolhido para este estudo foi o teste de Wilcoxon. Este teste escolhido é um teste não-parametrizado e se apresenta como uma alternativa ao conhecido teste de *t-student*, para dados pareados. Diferentemente deste ultimo, o teste de Wilcoxon é utilizado quando os dados a serem analisados não apresentam distribuição normal.

Este teste se baseia no *rankeamento* dos resultados, onde, comparados os dois resultados (CPLEX e RKGA-HK), é feito a atribuição dos valores 0, 1 e -1. Utilizando os resultados obtidos pelo método exato como referência, o valor 1 é atribuido quando há um resultado superior do resolvedor CPLEX e -1 quando o resultados for inferior, o valor 0 é atribuido quando há empate nos resultados.

O Teste de Wilcoxon é baseado na distribuição dos resultados em torno de sua mediana. Com isto, duas hipóteses são definidas:

- **Hipotese Nula (H0):** Caso está hipótese seja satisfeita, não há diferenças significativas entre as abordagens.

- **Hipotese Alternativa (H1):** Ao se refutar a Hipótese Nula (H0), aceita-se a hipótese alternativa, indicando diferenças significativas entre as abordagens.

Utilizando o valor de confiança de 95%, a hipótese nula é rejeitada quando o valor de p for inferior a 0,05 e aceita caso contrário. Maiores informações sobre o Teste de Wilcoxon podem ser encontradas no estudo de revisão de Whitley e Ball (2002).

Conforme analisado anteriormente, as métricas mais importantes consideradas por este estudo foram a qualidade da solução final e o tempo computacional demandado. Considerando a característica do problema e as restrições temporais impostas neste estudo, a qualidade da solução foi considerada um fator de maior relevância, devido ao fato de 20 minutos, no contexto do problema, não ser considerado um tempo demasiadamente grande.

Com isto, de forma a analisar as métricas de forma conjunta, ambas foram multiplicadas por um fator de relevância, conforme descrito a seguir, salientando-se que os dados foram submetidos a um processo de normalização, para ficarem no intervalo de 0 a 1.

- **Classe Fácil e Classe Média:**

  1. Relevância de 100% na qualidade da solução
  2. Relevância de 95% na qualidade da solução e 5% no tempo computacional demandado
  3. Relevância de 90% na qualidade da solução e 10% no tempo computacional demandado

- **Classe Difícil:** Neste classe não há comparação temporal, portanto foi realizada a comparação utilizando 100% de relevância na qualidade da solução, utilizando os tamanhos de população 100 e 200.

Na Tabela 5.10 são apresentados os testes estatísticos aplicados aos resultados da classe fácil. Nos três testes, utilizando diferentes relevâncias, a hipótese nula foi rejeitada, aceitando-se a hipótese alternativa, indicando diferenças significativas entre as abordagens. Utilizando relevância total na qualidade da solução, o método exato foi indicado como a melhor abordagem, porém, utilizando a relevância temporal, o algoritmo RKGA-HK foi indicado como a melhor abordagem.

Na Tabela 5.11 são apresentados os testes estatísticos aplicados aos resultados da classe média. Nota-se que a hipótese nula foi rejeitada quando considerada a relevância temporal, mais uma vez, com a indicação do algoritmo RKGA-HK como a melhor abordagem. Porém, nesta classe de instâncias, a hipótese nula foi aceita quando considerada apenas o fator de qualidade, indicando que não houve diferenças significativas entre as abordagens.

Tabela 5.10: Análise estatística dos resultados da classe fácil utilizando o Teste de Wilcoxon

Classe Fácil - Teste de Wilcoxon - Confiança 95%			
Relevância	Qualidade 100% Tempo 0%	Qualidade 95% Tempo 5%	Qualidade 90% Tempo 10%
Valor de p	0,005	$< 0,001$	$< 0,001$
Melhor abordagem	**CPLEX**	**RKGA-HK**	**RKGA-HK**

Tabela 5.11: Análise estatística dos resultados da classe média utilizando o Teste de Wilcoxon

Classe Média - Teste de Wilcoxon - Confiança 95%			
Relevância	Qualidade 100% Tempo 0%	Qualidade 95% Tempo 5%	Qualidade 90% Tempo 10%
Valor de p	0,68	$< 0,001$	$< 0,001$
Melhor abordagem	**EMPATE**	**RKGA-HK**	**RKGA-HK**

Na Tabela 5.12 são apresentados os testes estatísticos aplicados aos resultados obtidos na classe difícil. Corroborando as análises anteriores, o resolvedor CPLEX foi indicado como a melhor abordagem para esta classe de instâncias.

Tabela 5.12: Análise estatística dos resultados da classe difícil utilizando o Teste de Wilcoxon

Classe Difícil - Teste de Wilcoxon - Confiança 95%		
Relevância e População	Qualidade 100% População 100	Qualidade 100% População 200
Valor de p	$< 0,001$	$< 0,001$
Melhor abordagem	**CPLEX**	**CPLEX**

6

Conclusão

Nesta dissertação foram abordados dois métodos de resolução para o problema de reescalonamento de enfermeiros, um método exato, que serviu como guia de análise, e um algoritmo genético utilizando chaves aleatórias, que foi analisado conforme sua qualidade de resolução, convergência e tempo de execução.

No trabalho desenvolvido, foi primeiramente construído uma base de testes para uma comparação entre os diferentes métodos aplicados. A base criada apresenta diferentes graus de dificuldade, medida pelo esforço do método exato aplicado, podendo se tornar uma base de referência para futuros trabalhos na área. Segundo, utilizando conceitos de Teoria dos Grafos e algoritmos genéticos de chaves aleatórias, um algoritmo evolutivo foi desenvolvido e analisado comparativamente com o método exato, obedecendo os mesmos limites e critérios de parada.

Das diversas implementações, a melhor, obtida a partir de diversos testes experimentais e configurações de parâmetros, foi apresentada no decorrer deste trabalho, apresentando suas características e conceitos que ampararam a sua criação.

Como conclusão geral, os resultados obtidos neste trabalho apontam a competitividade do algoritmo genético de chaves aleatórias proposto. Os resultados alcançados pelo algoritmo proposto se demonstraram ótimos ou quase ótimos, utilizando uma fração do tempo de execução do método exato.

Desta conclusão também se extrai que o método exato, a despeito de comumente não ser considerado para grandes instâncias, devido ao aumento de tempo computacional exagerado, encontrou resultados ótimos para duas classes de dificuldade, as classes média e fácil, utilizando tempo computacional inferior a vinte minutos.

Na classe difícil, a conclusão é ambígua, pois, apesar do método exato superar o algoritmo proposto em relação a qualidade da solução em muitos casos, o algoritmo proposto também foi capaz de superar o método exato em certos casos, quando teve o tamanho de sua população aumentado.

Disto se conclui a dificuldade de parametrização de algoritmos genéticos para o encontro de boa eficiência em termos de qualidade e tempo computacional.

Com tudo isto, conclui-se que o algoritmo proposto é, de fato, competitivo e deve ser considerado para a resolução, porém deve-se considerar os diversos parâmetros a serem ajustados para a sua boa eficiência.

6.1 Trabalhos futuros

Em trabalhos futuros, pretende-se avaliar o algoritmo proposto, utilizando a base de testes proposta por Moz e Pato(2007). Diversos ajustes deverão ser feitos no algoritmo proposto, de forma a se adequar a modelagem proposta pelas autoras.

Outro trabalho futuro está relacionado com a "memetização"do algoritmo proposto. Neste trabalho, um algoritmo de busca local será implementado e analisado para avaliar a sua eficiência em termos de qualidade de solução e tempo computacional.

Com as fundações de criação de base criadas, outras bases poderão ser criadas utilizando diversos conjuntos de restrições, destas, um estudo avaliará o grau de dificuldade de resolução dos problemas, utilizando um método exato.

REFERÊNCIAS

AICKELIN, Uwe; DOWSLAND, Kathryn A.. An indirect Genetic Algorithm for a nurse-scheduling problem. Computers Operations Research, [s.l.], v. 31, n. 5, p.761-778, abr. 2004. Elsevier BV.

AIKEN, L. H. et al. Patient safety, satisfaction, and quality of hospital care: cross sectional surveys of nurses and patients in 12 countries in Europe and the United States. Bmj, [s.l.], v. 344, n. 202, p.1717-1717, 20 mar. 2012. BMJ.

BARD, Jonathan F.; PURNOMO, Hadi W.. Hospital-wide reactive scheduling of nurses with preference considerations. Iie Transactions, [s.l.], v. 37, n. 7, p.589-608, jul. 2005. Informa UK Limited.

BARD, Jonathan F.; PURNOMO, Hadi W.. Incremental changes in the workforce to accommodate changes in demand. Health Care Management Science, [s.l.], v. 9, n. 1, p.71-85, fev. 2006. Springer Nature.

BEAN, James C.. Genetic Algorithms and Random Keys for Sequencing and Optimization. Orsa Journal On Computing, [s.l.], v. 6, n. 2, p.154-160, maio 1994. Institute for Operations Research and the Management Sciences (INFORMS).

BERGH, Jorne van Den et al. Personnel scheduling: A literature review. European Journal Of Operational Research, [s.l.], v. 226, n. 3, p.367-385, maio 2013. Elsevier BV. http://dx.doi.org/10.1016/j.ejor.2012.11.029.

BURKE, Edmund K. et al. The State of the Art of Nurse Rostering. Journal Of Scheduling, [s.l.], v. 7, n. 6, p.441-499, nov. 2004. Springer Nature.

CHEANG, B et al. Nurse rostering problems—a bibliographic survey. European Journal Of Operational Research, [s.l.], v. 151, n. 3, p.447-460, dez. 2003. Elsevier BV.

CLARK, Alistair.; WALKER, H. (2011) Nurse rescheduling with shift preferences and minimal disruption. Journal of Applied Operational Research, 3 (3). pp. 148-162. ISSN 1735-8523 Disponível em: http://eprints.uwe.ac.uk/17608 *

CLARK, Alistair et al. Rescheduling nursing shifts: scoping the challenge and examining the potential of mathematical model based tools. Journal Of Nursing Management, [s.l.], v. 23, n. 4, p.411-420, 8 ago. 2013. Wiley.

COLLINI, Stevie A.; GUIDROZ, Ashley M.; PEREZ, Lisa M.. Turnover in health care: the mediating effects of employee engagement. Journal Of Nursing Management, [s.l.], v. 23, n. 2, p.169-178, 4 out. 2013. Wiley.

CONSTANTINO, Ademir Aparecido et al. A heuristic algorithm based on multi-assignment procedures for nurse scheduling. Annals Of Operations Research, [s.l.], p.165-183, 3 abr. 2013. Springer Nature. http://dx.doi.org/10.1007/s10479-013-1357-9.

CONSTANTINO, Ademir Aparecido et al. A Variable Neighbourhood Search for Nurse Scheduling with Balanced Preference Satisfaction. Proceedings Of The 17th International Conference On Enterprise Information Systems, [s.l.], p.462-470, 2015. SCITEPRESS - Science and and Technology Publications. http://dx.doi.org/10.5220/0005364404620470.

CORDELLA, Luigi. P.; FOGGIA, Pasquale; SANSONE, Carlo; VENTO, Mario. 2001. An improved algorithm for matching large graphs. In Proceedings of the 3rd IAPR-TC15 Workshop on Graph-based Representation in Pattern Recognition, 149–159

DARWIN, Charles. The origin of species by means of natural selection. London: J. Murray; 1859.

DOWSLAND, K; THOMPSON, J M. Solving a nurse scheduling problem with knapsacks, networks and tabu search. Journal Of The Operational Research Society, [s.l.], v. 51, n. 7, p.825-833, jul. 2000. Informa UK Limited.

FOGEL, David B.. Evolutionary algorithms in theory and practice. Complexity, [s.l.], v. 2, n. 4, p.26-27, mar. 1997. Wiley.

FOGEL, David B.. Evolutionary Computation. Ieee Press Series On Computational Intelligence, [s.l.], p.1-999, 28 out. 2005. John Wiley Sons, Inc..

GONÇALVES, José Fernando; RESENDE, Mauricio G. C.. Biased random-key genetic algorithms for combinatorial optimization. Journal Of Heuristics, [s.l.], v. 17, n. 5, p.487-525, 27 ago. 2010. Springer Nature. http://dx.doi.org/10.1007/s10732-010-9143-1.

GONÇALVES, José Fernando; RESENDE, Mauricio G.c.; TOSO, Rodrigo F.. AN EXPERIMENTAL COMPARISON OF BIASED AND UNBIASED RANDOM-KEY GENETIC ALGORITHMS. Pesquisa Operacional, [s.l.], v. 34, n. 2, p.143-164, ago. 2014. FapUNIFESP (SciELO). http://dx.doi.org/10.1590/0101-7438.2014.034.02.0143.

GUTJAHR, Walter J.; RAUNER, Marion S.. An ACO algorithm for a dynamic regional nurse-scheduling problem in Austria. Computers Operations Research, [s.l.], v. 34, n. 3, p.642-666, mar. 2007. Elsevier BV.

HOLLAND, John. Adaptation in Natural and Artificial Systems. Cambridge, MA:MIT Press. Second edition (1992). (First edition, University of Michigan Press, 1975).

HOPCROFT, John E.; KARP, Richard M.. An $n^{5/2}$ Algorithm for Maximum Matchings in Bipartite Graphs. Siam Journal On Computing, [s.l.], v. 2, n. 4, p.225-231, dez. 1973. Society for Industrial Applied Mathematics (SIAM).

ISKEN, M. W. and W. HANCOCK, “A heuristic approach to Nurse scheduling in hospital units with non-stationary, urgent demand, and a fixed staff size,”Journal of the Society for Health Systems, 2 (2), 24–41 (1991)

JONG, Kenneth de. Parameter Setting in EAs: a 30 Year Perspective. Parameter Setting In Evolutionary Algorithms, [s.l.], p.1-18, 2007. Springer Berlin Heidelberg. http://dx.doi.org/10.1007/978-3-540-69432-8_1.

KITADA, Manabu; MORIZAWA, Kazuko. A heuristic method for nurse rerostering problem with a sudden absence for several consecutive days. International Journal of Emerging Technology and Advanced Engineering , 3(11):353–361, 2013.

KITADA Manabu; MORIZAWA, Kazuko. A heuristic method in nurse rerostering following a sudden absence of nurses. Proceedings of the 11th Asia Pacific Industrial Engineering and Management Systems Conference, Manila, 2011

LEGRAIN, Antoine; BOUARAB, Hocine; LAHRICHI, Nadia. The Nurse Scheduling Problem in Real-Life. Journal Of Medical Systems, [s.l.], v. 39, n. 1, p.1-11, 3 dez. 2014. Springer Nature.

MAENHOUT, Broos; VANHOUCKE, Mario. An evolutionary approach for the nurse rerostering problem. Computers Operations Research, [s.l.], v. 38, n. 10, p.1400-1411, out. 2011. Elsevier BV.

MAENHOUT, Broos; VANHOUCKE, Mario. An Artificial Immune System Based Approach for Solving the Nurse Re-rostering Problem. Evolutionary Computation In Combinatorial Optimization, [s.l.], p.97-108, 2013. Springer Berlin Heidelberg.

MONMARCHÉ, N.; VENTURINI, G.; SLIMANE, M.. On how Pachycondyla apicalis ants suggest a new search algorithm. Future Generation Computer Systems, [s.l.], v. 16, n. 8, p.937-946, jun. 2000. Elsevier BV.

MOZ, Margarida; PATO, Margarida Vaz. An Integer Multicommodity Flow Model Applied to the Rerostering of Nurse Schedules. Annals Of Operations Research, [s.l.], v. 119, n. 1/4, p.285-301, 2003. Springer Nature.

MOZ, Margarida; PATO, Margarida Vaz. Solving the Problem of Rerostering Nurse Schedules with Hard Constraints: New Multicommodity Flow Models. Annals Of Operations Research, [s.l.], v. 128, n. 1-4, p.179-197, abr. 2004. Springer Nature.

MOZ, Margarida; PATO, Margarida Vaz. A genetic algorithm approach to a nurse rerostering problem. Computers Operations Research, [s.l.], v. 34, n. 3, p.667-691, mar. 2007. Elsevier BV.

MUTINGI, M. and MBOHWA, C. (2017). The nurse rerostering problem: An explorative study.Proceedings of the International Conference on Industrial Engineering and Operations Management, Rabat, Morocco.*

NEWMAN, Karin; MAYLOR, Uvanney; CHANSARKAR, Bal. “The nurse satisfaction, service quality and nurse retention chain”. Journal Of Management In Medicine, [s.l.], v. 16, n. 4, p.271-291, ago. 2002. Emerald.

PATO, Margarida Vaz; MOZ, Margarida. Solving a bi-objective nurse rerostering problem by using a utopic Pareto genetic heuristic. Journal Of Heuristics, [s.l.], v. 14, n. 4, p.359-374, 17 ago. 2007. Springer Nature.

SINGH, Hemant Kumar; ALAM, Khairul; RAY, Tapabrata. Use of Infeasible Solutions During Constrained Evolutionary Search: A Short Survey. Lecture Notes In Computer Science, [s.l.], p.193-205, 2016. Springer International Publishing.

SMITH-MILES, Kate; LOPES, Leo. Measuring instance difficulty for combinatorial optimization problems. Computers Operations Research, [s.l.], v. 39, n. 5, p.875-889, maio 2012. Elsevier BV.

UNRUH, Lynn Y.; ZHANG, Ning Jackie. Newly Licensed Registered Nurse Job Turnover and Turnover Intent. Journal For Nurses In Professional Development, [s.l.], v. 30, n. 5, p.220-230, 2014. Ovid Technologies (Wolters Kluwer Health).

WHITLEY, Elise; BALL, Jonathan. Statistics review 6: Nonparametric methods. Critical Care, [s.l.], v. 6, n. 6, p.509-513, 2002. Springer Nature. http://dx.doi.org/10.1186/cc1820.

WILLIAMSON, David F.. The Box Plot: A Simple Visual Method to Interpret Data. Annals Of Internal Medicine, [s.l.], v. 110, n. 11, p.916-921, 1 jun. 1989. American College of Physicians. http://dx.doi.org/10.7326/0003-4819-110-11-916.

WRIGHT, P. Daniel; MAHAR, Stephen. Centralized nurse scheduling to simultaneously improve schedule cost and nurse satisfaction. Omega, [s.l.], v. 41, n. 6, p.1042-1052, dez. 2013. Elsevier BV.

Apêndices

.1 Resultados computacionais - Classe Fácil

Instância	Alterações	
	CPLEX	RKGA-HK
p10-2nd-abs10-t10-14.dat	59	59
p20-1st-abs20-t8-8.dat	74	74
p20-2nd-abs10-t8-18.dat	38	38
p20-1st-abs10-t8-2.dat	28	28
p10-2nd-abs10-t10-3.dat	64	64
p10-2nd-abs20-t10-18.dat	80	80
p20-1st-abs20-t8-5.dat	72	72
p10-2nd-abs10-t10-13.dat	51	51
p10-2nd-abs10-t10-11.dat	54	54
min-2nd-abs10-t10-3.dat	39	39
p10-2nd-abs10-t10-10.dat	45	45
p10-2nd-abs20-t10-17.dat	96	96
p10-2nd-abs10-t10-1.dat	50	50
p10-2nd-abs10-t10-5.dat	55	55
p10-2nd-abs10-t10-20.dat	56	56
p10-2nd-abs20-t10-1.dat	98	98
min-2nd-abs10-t10-2.dat	42	42
p10-2nd-abs20-t10-19.dat	94	94
p10-2nd-abs10-t10-4.dat	59	59
p10-1st-abs20-t8-18.dat	**85**	88
p20-2nd-abs10-t8-3.dat	32	32
p10-2nd-abs10-t10-17.dat	38	38
p10-2nd-abs10-t10-9.dat	41	41
p10-2nd-abs10-t10-12.dat	39	39
min-2nd-abs10-t6-6.dat	**52**	54
p10-2nd-abs10-t10-15.dat	49	49
p10-1st-abs10-t8-2.dat	19	19
p20-1st-abs20-t6-5.dat	80	80
p10-2nd-abs20-t10-15.dat	97	97
p20-1st-abs10-t8-6.dat	37	37
p10-2nd-abs10-t10-18.dat	57	57
p10-1st-abs20-t8-8.dat	75	75
p10-2nd-abs10-t10-6.dat	51	51
p10-2nd-abs10-t10-7.dat	49	49
p10-1st-abs10-t8-3.dat	24	24
min-2nd-abs10-t10-7.dat	**66**	68
p10-2nd-abs10-t10-2.dat	51	51
p10-1st-abs20-t8-4.dat	65	65
p10-2nd-abs10-t10-8.dat	33	33
p10-1st-abs20-t8-6.dat	73	73

.2 Resultados computacionais - Classe Média

Instância	Alterações	
	CPLEX	RKGA-HK
min-1st-abs20-t6-9.dat	**84**	89
p20-1st-abs20-t10-10.dat	94	94
p20-1st-abs20-t10-17.dat	80	80
p20-1st-abs10-t10-6.dat	30	30
p20-1st-abs20-t8-3.dat	74	74
p20-1st-abs10-t10-1.dat	52	52
p20-1st-abs10-t10-10.dat	29	29
p20-1st-abs10-t10-18.dat	48	48
p20-1st-abs20-t10-5.dat	96	96
p10-1st-abs20-t10-16.dat	91	91
p10-1st-abs10-t10-7.dat	31	31
p10-1st-abs20-t10-2.dat	67	67
p20-1st-abs20-t10-1.dat	94	94
p20-1st-abs20-t10-11.dat	76	76
p20-1st-abs10-t10-20.dat	41	41
p20-1st-abs10-t10-8.dat	31	31
p20-1st-abs20-t10-6.dat	72	72
p10-1st-abs10-t10-16.dat	37	37
min-1st-abs20-t6-13.dat	**60**	61
p10-1st-abs10-t8-4.dat	39	39
p20-1st-abs10-t10-16.dat	36	36
p20-1st-abs10-t8-11.dat	32	32
p20-2nd-abs10-t10-8.dat	45	45
p20-2nd-abs10-t10-13.dat	36	36
p20-1st-abs10-t10-4.dat	39	39
min-1st-abs10-t10-16.dat	49	49
p20-1st-abs10-t8-8.dat	31	31
p20-1st-abs10-t10-13.dat	27	27
p20-1st-abs20-t10-7.dat	81	81
p20-1st-abs10-t10-5.dat	37	37
p20-2nd-abs10-t10-15.dat	34	34
p20-1st-abs10-t10-17.dat	36	36
p20-1st-abs10-t10-19.dat	39	39
p20-1st-abs20-t10-18.dat	84	84
p20-1st-abs20-t10-2.dat	82	82
p20-1st-abs20-t10-13.dat	71	71
p20-1st-abs20-t10-4.dat	68	68
p20-1st-abs20-t10-19.dat	85	85
min-1st-abs20-t10-7.dat	85	85
p20-1st-abs20-t10-8.dat	86	86

.3 Resultados computacionais - Classe Difícil - População 100

Instância	Alterações	
	CPLEX	RKGA-HK
min-1st-abs10-t10-8.dat	**73**	78
min-1st-abs10-t8-11.dat	**59**	61
min-1st-abs10-t8-12.dat	**50**	54
min-1st-abs20-t10-13.dat	**110**	122
p20-1st-abs10-t10-12.dat	34	34
min-1st-abs20-t6-12.dat	**74**	75
p20-1st-abs10-t8-18.dat	34	34
p10-1st-abs20-t10-7.dat	71	71
min-1st-abs10-t10-6.dat	**64**	68
p20-1st-abs20-t10-3.dat	85	85
min-1st-abs10-t8-4.dat	**70**	77
min-1st-abs10-t10-3.dat	**55**	58
p20-1st-abs20-t10-14.dat	109	109
p20-1st-abs10-t10-14.dat	42	42
p10-1st-abs10-t10-3.dat	40	40
min-1st-abs20-t10-16.dat	**120**	127
p20-1st-abs20-t10-15.dat	82	82
min-1st-abs10-t8-13.dat	**55**	57
min-1st-abs20-t6-14.dat	**97**	110
p20-1st-abs10-t10-15.dat	46	46
min-1st-abs10-t10-4.dat	**64**	68
min-1st-abs10-t8-17.dat	**52**	55
min-1st-abs20-t8-12.dat	**112**	128
min-1st-abs10-t6-17.dat	**59**	61
min-1st-abs10-t10-12.dat	**82**	84
min-1st-abs20-t10-20.dat	**120**	126
min-1st-abs10-t6-14.dat	**47**	48
p20-1st-abs20-t8-2.dat	65	65
p10-1st-abs10-t6-3.dat	30	30
min-1st-abs20-t8-17.dat	**105**	118
min-1st-abs20-t8-14.dat	**108**	116
min-1st-abs10-t6-5.dat	**65**	76
min-1st-abs10-t10-10.dat	**71**	72
p20-1st-abs20-t10-16.dat	89	89
min-1st-abs10-t8-9.dat	**95**	112
min-1st-abs10-t8-1.dat	84	**83**
p20-1st-abs10-t10-7.dat	45	45
min-1st-abs10-t10-5.dat	**88**	122
p20-1st-abs20-t8-15.dat	74	74
min-1st-abs10-t8-10.dat	**54**	57

.4 Resultados computacionais - Classe Difícil - População 200

Instância	Alterações	
	CPLEX	RKGA-HK
min-1st-abs10-t10-8.dat	**73**	75
min-1st-abs10-t8-11.dat	59	59
min-1st-abs10-t8-12.dat	**50**	53
min-1st-abs20-t10-13.dat	**110**	115
p20-1st-abs10-t10-12.dat	34	34
min-1st-abs20-t6-12.dat	74	74
p20-1st-abs10-t8-18.dat	34	34
p10-1st-abs20-t10-7.dat	71	71
min-1st-abs10-t10-6.dat	64	64
p20-1st-abs20-t10-3.dat	85	85
min-1st-abs10-t8-4.dat	**70**	75
min-1st-abs10-t10-3.dat	55	55
p20-1st-abs20-t10-14.dat	109	109
p20-1st-abs10-t10-14.dat	42	42
p10-1st-abs10-t10-3.dat	40	40
min-1st-abs20-t10-16.dat	**120**	121
p20-1st-abs20-t10-15.dat	82	82
min-1st-abs10-t8-13.dat	**55**	56
min-1st-abs20-t6-14.dat	**97**	99
p20-1st-abs10-t10-15.dat	46	46
min-1st-abs10-t10-4.dat	**64**	66
min-1st-abs10-t8-17.dat	**52**	53
min-1st-abs20-t8-12.dat	**112**	115
min-1st-abs10-t6-17.dat	59	59
min-1st-abs10-t10-12.dat	82	**81**
min-1st-abs20-t10-20.dat	120	**119**
min-1st-abs10-t6-14.dat	47	**46**
p20-1st-abs20-t8-2.dat	65	65
p10-1st-abs10-t6-3.dat	30	30
min-1st-abs20-t8-17.dat	**105**	108
min-1st-abs20-t8-14.dat	**108**	111
min-1st-abs10-t6-5.dat	65	**64**
min-1st-abs10-t10-10.dat	71	71
p20-1st-abs20-t10-16.dat	89	89
min-1st-abs10-t8-9.dat	95	**94**
min-1st-abs10-t8-1.dat	84	**77**
p20-1st-abs10-t10-7.dat	45	45
min-1st-abs10-t10-5.dat	**88**	96
p20-1st-abs20-t8-15.dat	74	74
min-1st-abs10-t8-10.dat	**54**	57