UNIVERSIDADE FEDERAL DE VIÇOSA
CAMPUS DE RIO PARANAÍBA
SISTEMAS DE INFORMAÇÃO

THALLES FELIPE CALTABIANO

ALGORITMO GENÉTICO HÍBRIDO DE CHAVES ALEATÓRIAS VICIADAS PARA O PROBLEMA DE ESCALONAMENTO DE ENFERMAGEM

RIO PARANAÍBA
2019

THALLES FELIPE CALTABIANO

ALGORITMO GENÉTICO HÍBRIDO DE CHAVES ALEATÓRIAS VICIADAS PARA O PROBLEMA DE ESCALONAMENTO DE ENFERMAGEM

Monografia apresentada à Universidade Federal de Viçosa como parte das exigências para a aprovação na disciplina Trabalho de Conclusão de Curso II

Orientador: Prof. Dr. Matheus Nohra Haddad

RIO PARANAÍBA

2019

Resumo

Em um ambiente hospitalar, a equipe de enfermagem impacta diretamente na qualidade operacional dos hospitais. Além disso, este é considerado o setor com o maior número de trabalhadores, logo a geração de escalas de trabalho eficientes para os enfermeiros é o primeiro, e um dos passos mais importantes para que o setor opere com bom desempenho. O Problema de Escalonamento de Enfermagem (PEE), consiste na elaboração de escalas de trabalho visando obter uma solução viável que minimize a ocorrência de restrições como leis trabalhistas, preferências dos funcionários e demandas das instituições. Este trabalho busca encontrar escalas de enfermagem para o PEE proposto na *Second International Nurse Rostering Competition* (INRC-II), cujo objetivo é respeitar todas as restrições classificadas como fortes e minimizar as ocorrências das restrições denominadas fracas. Como o conjunto de restrições elaboradas pela competição qualificam o problema como NP-Difícil, uma abordagem que utilize heurísticas é necessária para que seja possível encontrar soluções para o problema em tempo hábil. Desta forma, este trabalho propõe como forma de gerar soluções às instâncias do INRC-II, a implementação de um algoritmo genético híbrido que faz uso da meta-heurística populacional Algoritmo Genético de Chaves Aleatórias Viciadas e da busca local *Random Variable Neighborhood Descent*. Os resultados encontrados classificariam o algoritmo proposto entre as 9 melhores equipes da competição, sendo este desempenho considerado satisfatório, visto que a proposta é inovadora e o número de melhorias possíveis sobre o método aplicado é grande.

Palavras-chaves: Problema de Escalonamento de Enfermagem, Algoritmo Genético de Chaves Aleatórias, INRC-II, Busca Local, Meta-heurísticas.

Abstract

In a hospital environment, a nursing team has a direct impact on the quality of hospital work, as well as being the largest of the hospital, so a generation of efficient scheduling for the nurses is the first, and one of the most important steps for the sector operate well. The Nursing Scheduling Problem (NSP), as it is called in the literature, consists in the elaboration of work scales aiming to obtain a viable solution that minimizes the occurrence of restrictions such as labor laws, employee preferences and institutional demands. This work seeks to find nursing scales for the NSP proposed in the Second International Nurse Rostering Competition (INRC-II), whose objective is to respect all the strong restrictions and to minimize the occurrences of the week restrictions. Since the set of constraints elaborated by the competition qualify the problem as NP-HARD, an approach that uses heuristics is necessary for reach solutions to the problem in acceptable computational time. In this way, this work offers as a way to generate solutions for INRC-II instances, an implementation of a genetic algorithm that uses the metaheuristic Biased Random-key Genetic Algorithm and local search Random Variable Neighborhood Descent. The results found would rank the proposed algorithm among the 9 best teams in the competition, and this performance was considered satisfactory, since the proposal is innovative and the number of possible improvements on the applied method is large.

Key-words: Nurse Scheduling Problem, Biased Random-key Genetic Algorithms, INRC-II, Local Search, metaheuristic.

Lista de ilustrações

Lista de tabelas

Lista de Algoritmos

Sumário

1 Introdução

O setor de enfermagem além de ter grande importância para o contexto operacional de um hospital contém o maior número de trabalhadores. Segundo Gaidzinski (1998), o setor representa no Brasil, cerca de 60% de todo pessoal envolvido no ambiente hospitalar. Sendo assim, uma gestão eficiente das equipes de enfermagem está diretamente relacionada com a qualidade dos serviços prestados pelos hospitais. Um bom gerenciamento de uma equipe de enfermagem começa pela elaboração de escalas, onde os enfermeiros devem ser alocados de forma que seja garantido um serviço ininterrupto, ou seja, 24 horas por dia durante os 7 dias da semana (SOUZA et al., 2011).

Não é difícil perceber que a geração de escalas de enfermagem é uma tarefa desafiadora, é necessário levar em conta variáveis como os dias de folga e férias, número de enfermeiros mínimo trabalhando em cada dia e turno, características específicas de cada profissional, capacitação, leis trabalhistas, preferências, entre muitas outras. Uma escala que não seja capaz de cumprir os requisitos citados pode trazer consequências ruins para o ambiente de trabalho, as quais podem afetar diretamente a qualidade do serviço de enfermagem, entre elas pode-se citar: sobrecarga de trabalho, grande número de faltas sem justificativas, conflitos internos e adoecimento. Como a montagem de uma escala de enfermagem deve levar em conta uma infinidade de pontos, essa tarefa acaba tendo uma complexidade muito grande, tornando assim praticamente inviável a geração de uma escala eficiente de forma manual. Dado esse problema, muitos hospitais começaram a utilizar ferramentas computacionais, a fim de auxiliarem a criação e o gerenciamento de escalas de enfermagem (ROSSETTI; CARQUI, 2009) (SOUZA et al., 2011).

Uma das áreas da computação que pode auxiliar na criação de escalas de enfermagem eficiente é a Pesquisa Operacional, a qual segundo Poltosi (2007) visa encontrar uma boa solução, e se for possível a ótima, para diferentes tipos de problemas oriundos as atividades de uma organização. Com o objetivo de fomentar os estudos na área de Pesquisa Operacional, e de forma mais específica, o Problema de Escalonamento de Enfermagem (PEE), foi proposta uma competição intitulada *Second International Nurse Rostering Competition* (INRC-II). Para a competição foi elaborado um conjunto de restrições que quando não cumpridas penalizam a qualidade de uma escala de enfermagem, sendo que todas as restrições foram realizadas tomando como base diversas situações que realmente ocorrem durante a elaboração de uma escala de enfermagem para um hospital. Sendo assim, é provável que os métodos implementados na competição possam ser aplicados com sucesso em ambientes hospitalares reais.

O conjunto de restrições impostas pelo INRC-II qualificam o problema como NP-Difícil, ou seja, é necessário uma grande quantidade de tempo e processamento para que

seja possível encontrar uma solução exata para o problema, onde muitas vezes o tempo se torna demasiado grande, impossibilitando assim a geração de uma solução. Uma das alternativas utilizadas para contornar esse problema é a implementação de heurísticas e meta-heurísticas, as quais buscam criar métodos não exatos afim de gerar boas soluções em tempo aceitável. Este trabalho então propôs a utilização de uma meta-heurística composta por um Algoritmo Genético de Chaves Aleatórias Viciadas (BRKGA) e um *Random Variable Neighborhood Descent* (RVND) para gerar soluções as instâncias elaboradas no INRC-II.

1.1 Objetivo Geral

Este estudo tem como objetivo a elaboração de um algoritmo competitivo para o PEE proposto na competição elaborada pelo INRC-II. A eficiência das soluções encontradas foram medidas por meio de comparações com os resultados das equipes que participaram da competição.

1.2 Objetivos Específicos

- Gerar soluções capazes de classificar o algoritmo para a etapa final do INRC-II.
- Estudar como o algoritmo se comporta diante das restrições impostas no problema, identificando quais o mesmo apresenta maior facilidade e dificuldade em tratar.
- Incentivar pesquisadores a utilizarem um Algoritmo Genético de Chaves Aleatórias Viciadas para gerar soluções em problemas de escalonamento, visto que na literatura há uma carência de estudos que utilizam dessa técnica para o contexto mencionado.

2 Referencial Teórico

2.1 A Competição

O *International Nurse Rostering Competition* (INRC) teve sua primeira edição no ano de 2010 e foi publicado por Haspeslagh et al. (2010). Segundo os autores, a competição detinha três principais objetivos. Primeiramente atrair competidores de todas as áreas de pesquisa, possibilitando a criação de abordagens multidisciplinares na tentativa de resolver o problema de escalas de enfermagem. O segundo objetivo foi tentar diminuir a lacuna existente entre a pesquisa e prática dentro da área de Pesquisa Operacional. O terceiro, por sua vez, foi estimular a comunidade a debater cada vez mais sobre a pesquisa de escalas e cronogramas.

A primeira edição trouxe excelentes resultados, onde foram encontradas e relatadas novas e melhores soluções, sendo que para algumas instâncias até mesmo soluções ótimas foram alcançadas. Após os bons resultados do INRC , uma nova edição da competição foi proposta, intitulada como INRC-II e elaborada por Ceschia et al. (2015b).

2.2 Descrição do Problema

O INRC–II diferentemente de seu predecessor propôs uma abordagem com menos restrições, porém com uma formulação de múltiplos estágios para o problema. Os participantes deveriam criar um método de busca (*solver* - solucionador) capaz de processar informações e resolver um único estágio do problema, equivalente a uma semana. Em seguida a solução encontrada foi enviada para o simulador da competição, onde é simulada e avaliada. Todo o processo é feito de forma iterativa, semana por semana, até que chegue ao fim do horizonte estabelecido pela instância, que varia entre 4 ou 8 semanas.

Na última etapa existe um validador capaz de concatenar todas as soluções e transformá-la em uma só, a solução final tem então a sua viabilidade testada, e caso não quebre nenhuma restrição do tipo forte, uma função objetivo quantifica a qualidade da solução final. A Figura 1 mostra um exemplo de soluções geradas pelo *solver* de um competidor fictício para um horizonte de planejamento de 4 semanas. Neste exemplo, cada linha das matrizes representam um enfermeiro, e cada coluna um dos sete dias da semana. Para cada posição da matriz existe um turno, e uma competência a ser exercida, sendo os turnos representados da seguinte forma: (*early* – E), vespertino (*Day* – D), noturno (*night*– N), madrugada (*late* – L) e (*day-off*– **-**)

Para que toda a natureza de multi-estágios do problema seja possível de ser im-

Escala Gerada para a 1ª semana

	D1	D2	D3	D4	D5	D6	D7
N0	D nurse	L headnurse	-	N nurse	-	L nurse	E headnurse
N1	D caretaker	-	N nurse	L nurse	E nurse	D caretaker	D nurse
N2	N nurse	-	D nurse	-	N nurse	D nurse	-

Escala Gerada para a 2ª semana

	D1	D2	D3	D4	D5	D6	D7
N0	E nurse	L headnurse	-	N nurse	-	-	L headnurse
N1	D caretaker	-	N nurse	L nurse	L nurse	D caretaker	D nurse
N2	E nurse	-	N nurse	-	N nurse	D nurse	-

Escala Gerada para a 3ª semana

	D1	D2	D3	D4	D5	D6	D7
N0	L nurse	L headnurse	-	D nurse	-	E nurse	E headnurse
N1	N caretaker	-	D nurse	N nurse	E nurse	L caretaker	L nurse
N2	N nurse	-	-	L nurse	L nurse	D nurse	D nurs

Escala Gerada para a 4ª semana

	D1	D2	D3	D4	D5	D6	D7
N0	N nurse	D headnurse	-	N nurse	-	E nurse	N headnurse
N1	E caretaker	-	D nurse	L nurse	E nurse	D caretaker	D nurse
N2	N nurse	-	D nurse	-	E nurse	D nurse	-

Figura 1 – Exemplo de 4 soluções semanais geradas por um *solver*, baseado em Portella (2017).

plementada, é necessário que alguns arquivos sejam gerados e manipulados até que se alcance a solução final. São denominados pelo INRC-II de: *Scenario* (Cenário), *Week data* (Dados da Semana), *History* (Histórico) e *Solution* (Solução).

2.2.1 Cenário

Esse arquivo contém todas as informações comuns a todos os estágios do problema.

- Horizonte de planejamento: Responsável por estabelecer quantas semanas devem ser processadas no total, 4 ou 8.
- Competências: Um enfermeiro pode exercer as funções de *head nurse* (chefe de enfermagem), *nurse* (enfermeiro), *caretaker* (cuidador), e *trainee*. É possível que um enfermeiro possua mais de uma competência, mas só pode exercer uma delas para o dia e turno em que for escalado para trabalhar.
- Contrato: Cada enfermeiro possui um tipo específico de contrato *full time* (integral), *part time* (meio expediente), entre outros. Os contratos definem limites para a distribuição e atribuições dentro do horizonte de planejamento, e podem ser descritos como:
  - Número mínimo e máximo de atribuições no horizonte de planejamento.
  - Número mínimo e máximo de dias úteis consecutivos que certo enfermeiro pode trabalha.
  - Número mínimo e máximo de folgas consecutivas.
  - Número mínimo e máximo de finais de semana trabalhados no horizonte de planejamento
  - Uma variável booleana, responsável por dizer se dado enfermeiro já completou os dias em que deve trabalhar nos finais de semana. Dessa forma, é possível saber se o mesmo deve ou não ser escalado para os dois dias do final de semana.
- Enfermeiro: Para cada enfermeiro é fornecido o seu nome, o tipo de contrato e suas competências.
- Tipos de turno: Para cada tipo de turno matutino (*early* – E), vespertino (*Day* – D), noturno (*night*– N) e madrugada (*late*– L) é dado o número mínimo e máximo de atribuições consecutivas para determinado turno, assim como uma matriz de sucessões de turnos proibidos. Por exemplo, um funcionário que trabalhou na segunda-feira no turno noturno não pode trabalhar no turno diurno da terça-feira seguinte.

2.2.2 Dados da Semana

Esse arquivo contém unicamente os dados referentes a semana que está sendo computada.

- Requisitos: São passados para cada turno, competência, e dia da semana, um número mínimo e ótimo de enfermeiros necessários para cumprir o trabalho.
- Solicitações de enfermeiros: Um enfermeiro pode solicitar não trabalhar em determinado turno em um determinado dia. Esse arquivo atribui um nome de enfermeiro a essas condições.

  Essas informações variam de semana para semana, de acordo com o número de pacientes do hospital e das preferências específicas dos enfermeiros.

2.2.3 Histórico

Esse arquivo contém as informações que devem ser levadas de uma semana para outra. Caso contrário, não seria possível a semana seguinte cumprir todas as restrições do problema corretamente. Possui basicamente dois tipos de informação:

- Dados de Borda: Sua função é evitar atribuições consecutivas em:
  - Turnos que foram trabalhados no último dia da semana anterior, ou um valor vazio caso o enfermeiro tenha recebido esse dia de folga.
  - Número de turnos do mesmo tipo que foram repetidos durante a semana anterior, e o número de turnos consecutivos trabalhados no geral.
  - Número de dias consecutivos de folga.
- Contadores: Coletam valores específicos, que são acumulados semana a semana e são verificados somente no fim do horizonte de planejamento (4 ou 8 semanas). São eles:
  - Número total de turnos trabalhados.
  - Número total de finais de semana trabalhados.

2.2.4 Arquivo de Solução

Esse arquivo é a saída produzida pelo *solver* (solucionador) do competidor, contém para cada enfermeiro da instância o seu nome, dia da semana em que trabalhou, turno e competência. Uma saída para cada estágio foi representada na Figura 1, contudo, esse arquivo de solução se difere da figura citada pelo fato de todas as informações serem

representadas apenas em forma de texto. Vale citar que existe também um arquivo complementar ao de solução, sua função é permitir que o competidor, caso tenha interesse, passe informações extras (que não estão contidas no arquivo de solução gerado) para as próximas chamadas do *solver*. A Figura 2 demonstra como todos os arquivos citados se relacionam com o simulador para um horizonte de planejamento de 4 semanas. Os arquivos representados pela cor azul são referentes as entradas do solucionador, e os representados em amarelo são as saídas do solucionador. O simulador deve receber o arquivo de histórico e de solução após as chamadas do solucionador, desta forma é possível verificar se a solução encontrada é valida e também gerar um novo arquivo de histórico para as próximas semanas a serem processadas.

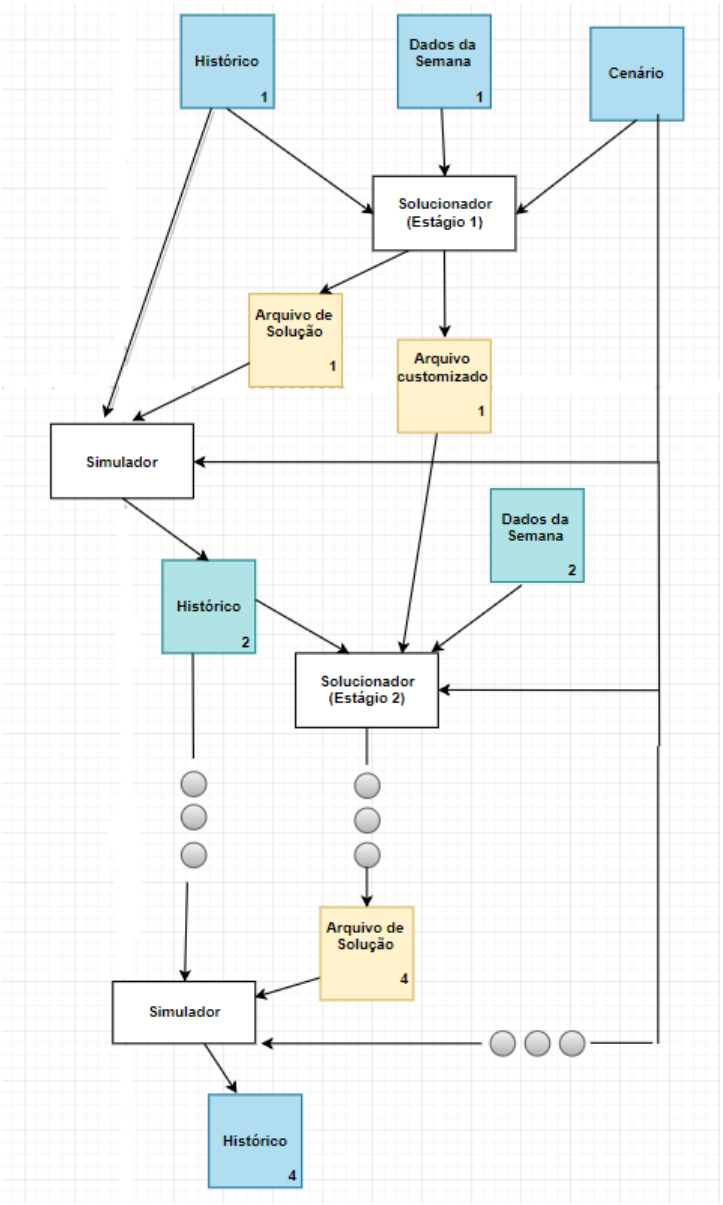

Figura 2 – Relação entre os arquivos, simulador e solucionador, baseado em Ceschia et al. (2015b).

A Figura 3 relaciona os arquivos com o validador, sendo ambas referentes a um horizonte de planejamento de 4 semanas. O validador recebe os arquivos de histórico inicial, cenário, todos os dados da semana e todos os arquivos de solução gerados pelo solucionador. Como saída o validador gera um relatório de validação contendo a concatenação de todas as escalas geradas, assim como o custo da solução final encontrada.

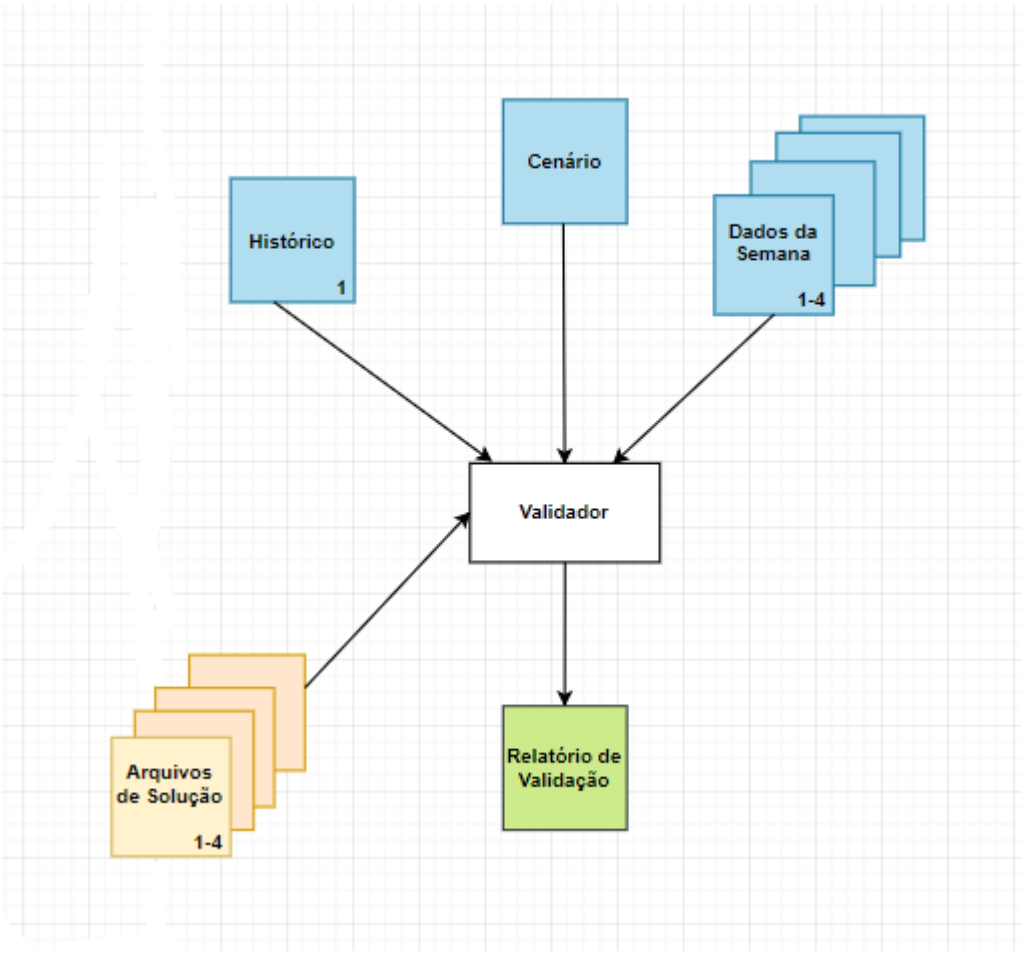

Figura 3 – Relação entre os arquivos e o validador, baseado em Ceschia et al. (2015b).

2.3 Restrições

A maioria das restrições impostas pelo problema são definidas para cada semana separadamente, mas também existem algumas restrições globais, que serão consideradas somente no final do horizonte de planejamento. Além da divisão citada acima, existem restrições chamadas fortes (H), as quais o *solver* deve garantir que sempre sejam satisfeitas para que uma solução seja válida, e também existem as fracas (S), onde cada uma carrega um determinado peso, sendo aplicado toda vez que esse tipo de restrição não é cumprida, diminuindo assim a sua qualidade. Um modelo matemático da competição foi proposto

em Ceschia et al. (2015a) sendo de grande importância para o correto entendimento das restrições, as quais muitas vezes em sua forma descritiva acabam não ficando tão claras.

2.3.1 Restrições para uma única semana

As restrições fortes (H) são:

- H1: O enfermeiro pode trabalhar apenas um turno por dia.
- H2: O número de enfermeiros para cada turno e para cada competência dever ser pelo menos igual ao requerimento mínimo.
- H3: A sucessão de turnos para dois dias consecutivos deve ser respeitada dentro do cenário, por exemplo: caso um enfermeiro trabalhe no turno noturno de uma terça-feira o mesmo não poderá trabalhar no turno diurno da quarta-feira.
- H4: Não se pode escalar um enfermeiro para trabalhar em uma competência a qual não possui.

Os pesos referentes as restrições fracas (S) estão dispostos logo após seu nome, por exemplo: S1(30). Toda vez que essa restrição não for cumprida deve ser aplicada na solução uma penalização com peso 30. As restrições fracas (S) são:

- S1(30): Existe um número ótimo de quantos enfermeiros com determinada competência devem estar presentes em cada turno. Para cada enfermeiro ausente, o custo da restrição será incidido, já para excedentes não há penalização.
- S2(15/30): Há um número mínimo e máximo de dias consecutivos trabalhados e de alocações consecutivas para um mesmo turno. As penalizações são respectivamente 30 para dias consecutivos e 15 para turnos seguidos.
- S3(30): Número mínimo e máximo de dias consecutivos de folga. Cada dia extra ou ausente é multiplicado pelo peso definido.
- S4(10): O peso dessa restrição é aplicado para cada enfermeiro designado a trabalhar em um turno que não é de seu desejo.
- S5(30): Cada funcionário que tem como preferência trabalhar no fim de semana, deve ser designado para trabalhar no sábado e no domingo, ou não trabalhar no fim de semana. Os custos dessa restrição serão aplicados para cada enfermeiro que trabalhar apenas um dia do final de semana.

2.3.2 Restrições de todo o horizonte de planejamento

Todas as restrições desse tipo são fracas e avaliadas somente no final do horizonte de planejamento.

- S6(20): Para cada enfermeiro, o número de dias trabalhados deve respeitar o mínimo e o máximo de dias estipulados no seu tipo de contrato. A diferença para mais ou para menos é multiplicada pelo custo dessa restrição.
- S7(30): Para cada enfermeiro, o número de finais de semana trabalhado deve ser menor ou igual ao máximo permitido. O número em excesso de finais de semana trabalhados é multiplicado pelo custo da restrição.

2.4 *Benchmark*

Foi disponibilizado pela competição uma ferramenta de *benchmark* para contabilizar o tempo que a máquina do competidor está levando para gerar cada solução semanal. Para cada tamanho de problema diferente, definido pelo número de enfermeiros que o arquivo de cenário possui, o programa retorna um valor indicando por quanto tempo o competidor pode executar seu *solver* para cada estágio do problema. Em geral, o tempo disponibilizado pelo *benchmark* é relativamente pequeno, tornando assim a competição ainda mais intrigante e desafiadora.

2.5 Heurística e Meta-heurística

Hartog (2016) analisou em seu trabalho diversas restrições de problemas de escalonamento de enfermagem, por fim, classificou se cada uma das restrições presentes no estudo continha uma complexabilidade NP-Difícil ou poderia ser resolvida em tempo polinomial aceitável. Por meio de comparações com o trabalho citado, foi possível concluir que a combinação de restrições propostas pelo INRC-II classificam o problema da competição como NP-Difícil.

Visto a complexabilidade do problema deste estudo, torna-se necessário a utilização de métodos heurísticos para a geração de soluções sobre as instâncias do INRC-II. Segundo Shanno (1985), heurísticas podem ser definidas como métodos baseados em modelos não exatos, como a experiência e julgamento e são utilizadas para gerar boas soluções em determinados problemas, contudo, não se comprometem a alcançar uma solução ótima.

Em complemento as heurísticas existem meta-heurísticas, as quais segundo Gonçalves e Resende (2011) são procedimentos de alto nível responsáveis por coordenar heurísticas mais simples, tornando assim possível alcançar soluções melhores do que as encontradas por uma simples heurística. Tal vantagem é justificada pelo fato das meta-heurísticas

conterem estratégias com a intenção de evitar que uma solução fique presa em ótimos locais, e também possivelmente distante do ótimo global (SOUZA, 2008).

As meta-heurísticas podem ser divididas em duas categorias: busca local e busca populacional. Se diferenciam basicamente pela forma como exploram o espaço de solução, na tentativa de escapar dos ótimos locais, e podem ser definidas ainda segundo Souza (2008) como:

- Busca Local: O espaço de soluções é percorrido através de movimentações, feitas a cada passo sobre a solução atual, por meio desse procedimento é gerada uma nova solução pertencente a vizinhança.
- Busca Populacional: Encontram e armazenam um conjunto de boas soluções, em sequência passam por um procedimento de combinação, na tentativa de produzir soluções ainda mais próximas do ótimo global.

2.6 Algoritmo Genético

O algoritmo genético (AG) é uma meta-heurística de busca populacional que foi proposto pela primeira vez por Holland (1962) com a intenção de utilizar os conhecimentos da biologia evolutiva para a solução de problemas da computação. Para que um processo de evolução ocorra é necessário primeiramente a existência de uma população. Nos AG'S a população pode ser definida como um conjunto de soluções do problema, onde cada solução recebe o nome de indivíduo. Cada indivíduo contém um cromossomo, representado como um vetor com o número de entradas igual ao tamanho da instância do problema. Um cromossomo é representado como um conjunto de genes, para os algoritmos genéticos um gene é definido como cada uma das entradas do vetor, sendo que o valor contido nessa entrada representa um alelo (MORO et al., 2017). A Figura 4 representa a relação entre cada uma das definições explicadas.

2.6.1 Etapas de um algoritmo genético

Todas as etapas citadas nessa subseção possuem como referência os trabalhos de Goldberg e Holland (1988), Souza (2008), Moro et al. (2017).

A meta-heurística tem seu início marcado pela busca em uma população, geralmente escolhida de forma aleatória. Em sequência é chamada uma função de *fitness* (aptidão), responsável por quantificar a adaptação ao meio de cada indivíduo da população atual, sendo que quanto maior for a aptidão de um indivíduo, mais adaptado ao meio ele está.

O próximo passo é composto por uma verificação no critério de parada dos AG's, caso seja satisfeito, o algoritmo chega a seu fim e são listados os melhores indivíduos da

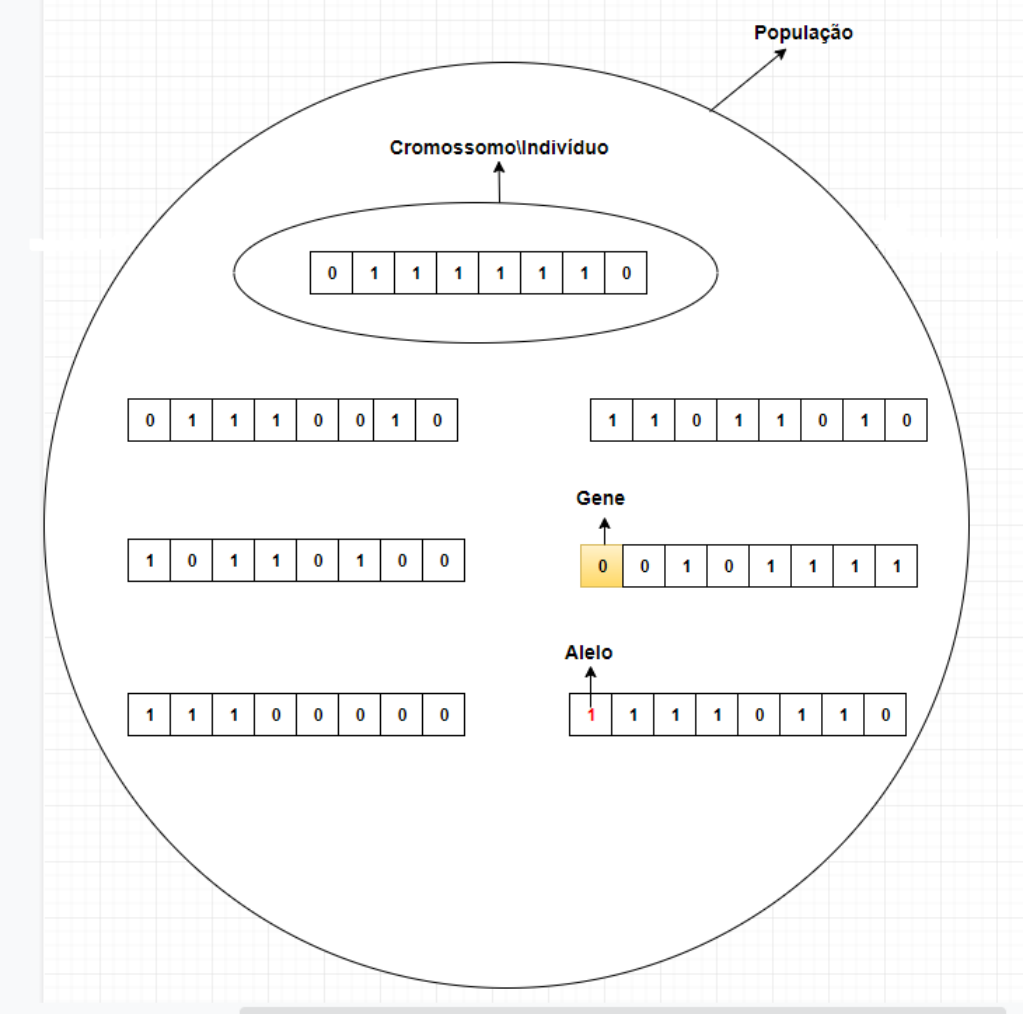

Figura 4 – Relação das definições básicas de um algoritmo genético, baseado em Zäpfel, Braune e Bögl (2010).

população (melhores soluções). Caso contrário, é iniciado um conjunto de processos para a geração de uma nova população. Os critérios de parada mais adotados pela meta-heurística são 3:

- Quando um certo número de novas populações geradas alcança uma constante predeterminada.
- Quando a qualidade da solução não apresenta melhora após um certo número de iterações do algoritmo.
- Quando é verificado um alto grau de homogeneidade da população, isso acontece quando o desvio padrão da população é menor que um determinado valor.

Verificado que o critério de parada do algoritmo não foi satisfeito, devem ser iniciados os procedimentos necessários para a geração da próxima população, sendo os operadores genéticos *crossover* (cruzamento) e mutação responsáveis por essa função. Durante

o cruzamento, dois indivíduos devem receber o título de pai, em sequência os genes de dois cromossomos pais devem ser interpolados. Ao fim desse processo são gerados geralmente outros dois novos indivíduos com um conjunto de genes de cada pai. A escolha dos pais pode ser feita de forma aleatória ou por meio de uma função que utiliza o grau de aptidão como parâmetro. A Figura 5 representa um operador de *crossover* por um ponto de corte, onde é inicialmente definido um único ponto de corte para dividir duas soluções pais. Em sequência é gerado um novo descendente, sendo seu lado esquerdo do ponto de corte preenchido pelos genes do Pai 1, enquanto seu lado direito contém os genes do Pai 2.

Figura 5 – Exemplo de *Crossover* por um ponto de corte, baseado em (MORO et al., 2017).

O próximo passo é conhecido como mutação e pode ser aplicada nos AG's por meio da modificação dos genes de determinado indivíduo. Comumente uma entrada do vetor é escolhida de forma aleatória para sofrer o processo de mutação. Em seguida, o valor contido na entrada (alelo) é alterado. A Figura 6 representa um processo de mutação, onde o gene modificado pela mutação está representado pela cor amarelo, tendo seu alelo trocado de 1 para 0.

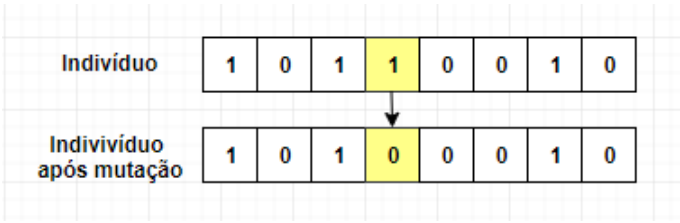

Figura 6 – Exemplo de mutação, baseado em (MORO et al., 2017).

Os processos de mutação e *crossover* são de grande importância pois possibilitam um aumento no espaço de busca, diminuindo assim a probabilidade das soluções ficarem presas a máximos locais. A operação de *crossover* é normalmente implementada com uma

probabilidade muito maior do que a de mutação, por exemplo: 80% e 2% respectivamente. Após as duas etapas citadas, é gerada uma nova população, a qual deve passar por um filtro, que por meio da aptidão estipula critérios a fim de definir quais indivíduos devem sobreviver dentro da nova população gerada.

Os cromossomos sobreviventes podem ser escolhidos de diversas formas diferentes, alguns exemplos são: por meio de uma seleção aleatória, através de uma roleta (onde a chance de um cromossomo sobreviver é proporcional ao seu nível de aptidão), ou por um critério que mescla os dois citados anteriormente. Vale ressaltar que todos esses métodos não excluem a sobrevivência de indivíduos com menor aptidão (dois deles apenas diminuem a chance dos mesmos serem mantidos na população). A Figura 7 representa o fluxograma de um algoritmo genético, contendo todas as etapas recém descritas neste trabalho.

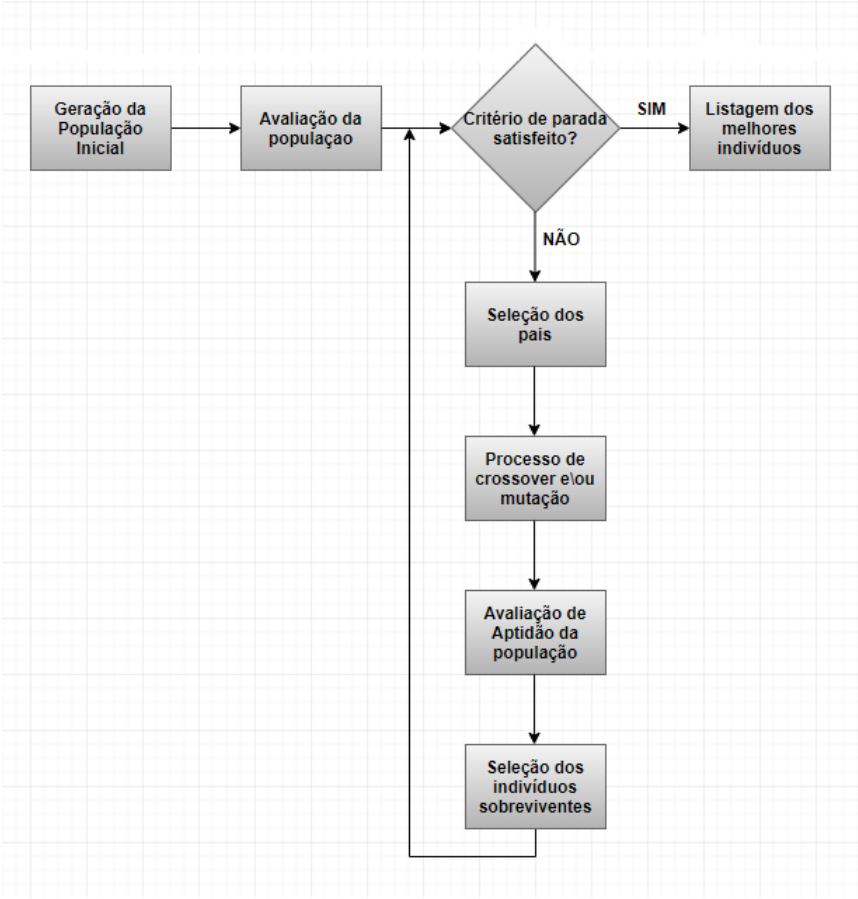

Figura 7 – Etapas de um algoritmo genético, baseado em Souza (2008).

2.7 Algoritmo RKGA e BRKGA

Uma variação dos algoritmos genéticos foi proposta por Bean (1994) sendo nomeada de Algoritmo Genético de Chaves Aleatórias (RKGA). Diferentemente da imple-

mentação comum de algoritmos genéticos, essa meta-heurística representa cada solução de um problema de otimização como uma chave aleatória, onde cada chave é composta por um número real gerado de forma aleatória pertencente ao intervalo [0,1). Após uma população com p vetores de chaves aleatórias ser gerada é aplicado um decodificador sobre eles. O resultado disso é um mapeamento, onde cada chave aleatória é associada a uma solução dentro de toda a população.

O procedimento do decodificador é específico de cada problema (BRITO; SEMAAN; BRITO, 2014), por exemplo, o proposto em Bean (1994) realiza uma simples ordenação das chaves aleatórias de um vetor e em sequência gera uma permutação com as posições dos elementos que tiveram que ser reordenados. A Figura 8 oferece uma representação do decodificador citado. Vale frisar que os operadores genéticos de *crossover* e mutação são realizados sobre o vetor de chaves aleatórias, com isso é reduzida a incidência de soluções que não cumprem as restrições do problema (MORO et al., 2017).

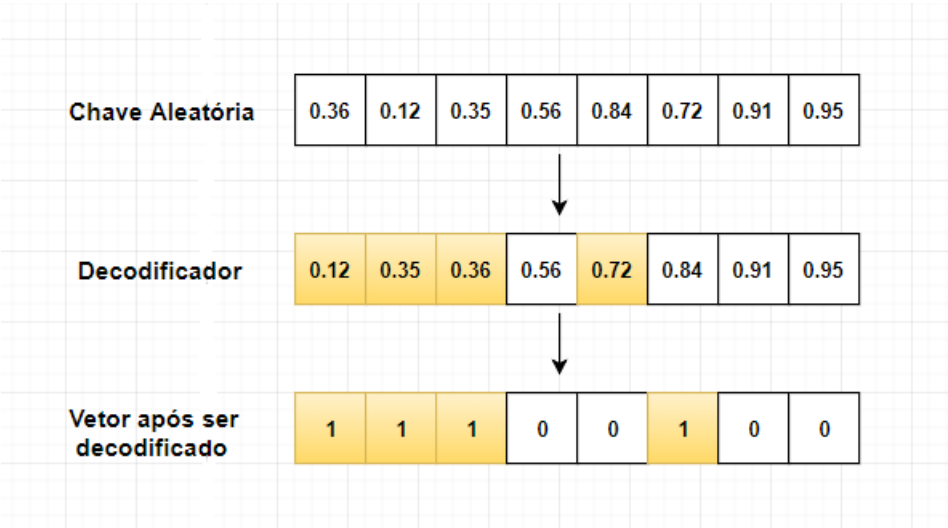

Figura 8 – Decodificador de Bean (1994), baseado em (MORO et al., 2017).

O RKGA inicia-se com a geração de uma população inicial e seus vetores já representados por meio de chaves aleatórias. Em sequência, é aplicada uma função *fitness* sobre cada indivíduo da população, sendo esta uma medida de eficiência do que se deseja maximizar ou minimizar no problema, podendo ter algum tipo de relação com a função objetivo do problema. (GOLDBERG; HOLLAND, 1988)

A próxima etapa utiliza a função de *fitness* para dividir a população em dois grupos, denominados elitistas e não elitistas. O primeiro recebe esse nome justamente por possuir uma boa avaliação de *fitness*, já o grupo restante é denominado de não elitista pelo fato de suas soluções não fazerem parte do primeiro grupo. A divisão citada deve ser feita de forma que o grupo de soluções elitistas seja sempre menor do que a não elite, sendo esta regra de extrema importância para que o algoritmo não fique estagnado em

máximos locais logo nas primeiras iterações (BEAN, 1994).

Feita a classificação, inicia-se os passos para a geração da nova população. Inicialmente todas as soluções pertencentes ao grupo elitista são escolhidas para fazerem parte da nova população. O próximo passo é a escolha de um número de indivíduos que deverão passar pelo processo de mutação e o último é completar os indivíduos da nova população com a operação de *crossover*. Sendo k uma iteração do algoritmo, é importante ressaltar que o número de soluções presentes na iteração $(k+1)$ deve ser igual ao da iteração k. O procedimento para geração de novas populações é mantido até que se alcance um resultado homogêneo, onde um cromossomo representa praticamente toda a população, sendo este por sua vez, retornado como o resultado do algoritmo (MORO et al., 2017).

Assim como o RKGA foi criado com base nos algoritmos genéticos, Gonçalves e Resende (2011) construíram um novo algoritmo baseado no modelo RKGA de Bean (1994), o mesmo foi chamado de Algoritmo Genético de Chaves Aleatórias Viciadas (BRKGA). Ambos seguem os mesmo princípios já citados e utilizam o cruzamento uniforme parametrizado de Spears e Jong (1991) para a geração de novos filhos. Contudo, se diferem pela maneira como os pais são selecionados para o processo de *crossover* e pela probabilidade de um filho herdar um alelo do pai elitista. Enquanto o RKGA escolhe os pais com base em todos os indivíduos da população, o BRKGA estabelece que um pai deve ser sorteado dentro da população elitista e o outro da população não elitista. Definido o par de pais, é lançada uma moeda viciada onde a chance do filho gerado herdar os genes do pai elitista é maior que 50% (GONÇALVES; RESENDE, 2011).

É possível observar na Figura 9 que o número de iterações que algoritmo BRKGA levou para alcançar uma solução alvo foi muito menor em comparação com o RKGA. Fica claro que o fato do BRKGA propagar os genes dos indivíduos elitistas em uma taxa maior que o RKGA, faz com que quase sempre o novo algoritmo seja superior ao seu predecessor (GONÇALVES; RESENDE, 2011). Como as regras do INRC–II estipulam um limiar de tempo para que o *solver* do competidor retorne uma solução, a utilização do BRKGA se torna uma boa estratégia para que seja possível alcançar bons resultados nas instâncias do problema.

2.8 *Variable Neighborhood Descent*

O *Variable Neightborhhod Descent* (VND) é um algoritmo de busca local proposto por Mladenović e Hansen (1997). O método busca por meio de trocas sistemáticas entre diferentes estruturas de vizinhança explorar o espaço de soluções de um determinado problema. O algoritmo inicia com a exploração de uma vizinhança inicial até que não seja mais possível alcançar melhorias, após este ponto, uma nova estrutura deve ser explorada. Ao fim da exploração da nova estrutura, o algoritmo possuí dois caminhos diferentes a

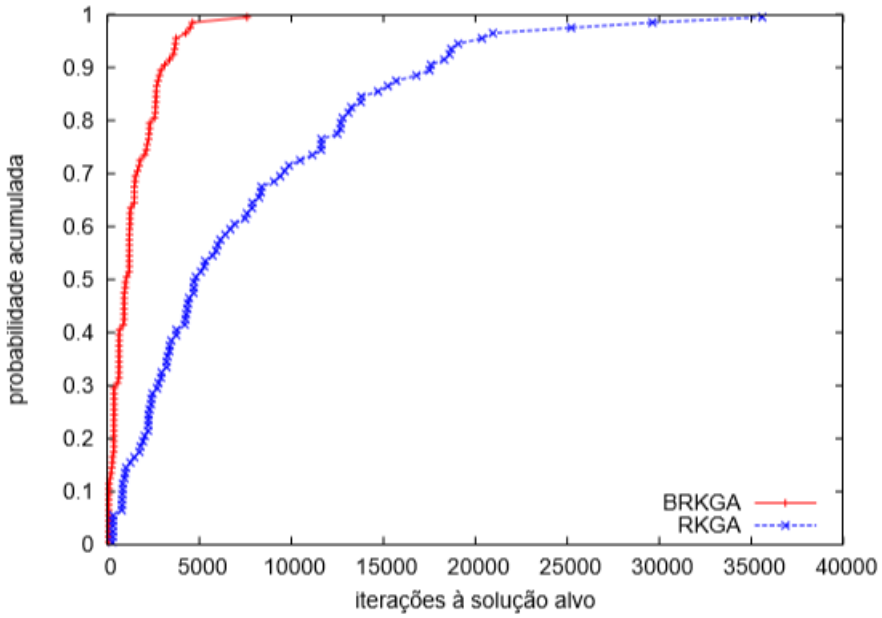

Figura 9 – Comparação da meta-heurística RKGA e BRKGA dada uma solução alvo em um problema de otimização (GONÇALVES; RESENDE, 2011).

seguir: caso seja encontrado alguma melhora, o método retorna para a estrutura de vizinhança inicial, e caso não haja melhoras, a próxima estrutura de vizinhança deve ser explorada. O algoritmo termina quando não é mais possível encontrar melhoras após visitar todas as estruturas de vizinhança implementadas no algoritmo. O resultado no fim de todo o processo é uma nova solução pertencente a um ótimo local em relação as estruturas que foram exploradas. A descrição do algoritmo está presente no Pseudocódigo 1.

Algoritmo 1 Pseudocódigo VND

```
    Seja s_0 uma solução inicial e r o número de estruturas de vizinhança corrente
 2: s ← s_0;                                              ▷ Solução corrente
    k ← 1;                                                ▷ Tipo de estrutura de vizinhança
 4: enquanto (k <= r) faça
        Encontre o melhor vizinho s'∈ N^k(s);
 6:     se f(s) < (s') então
            s ← s';                                       ▷ Atualiza a solução corrente
 8:         k ← 1;
        senão k ← k + 1;                                  ▷ Passa para a próxima estrutura de vizinhaça
10:     fim se
    fim enquanto
        devolve s
```

Existe uma variação do VND conhecida como RVND (*Random Variable Neighborhood Descent*). Consiste basicamente em escolher aleatoriamente quais estruturas de vizinhança devem ser exploradas. A aleatoriedade na execução dessas estruturas, em com-

paração com a abordagem clássica, faz com que a primeira vizinhança escolhida no VND não seja visitada sempre mais vezes do que as outras.(SOUZA et al., 2010)

3 Trabalhos Relacionados

Algumas das equipes finalistas do INRC-II publicaram artigos sobre as estratégias utilizadas em seus *solvers* (os mesmos serão explicados no decorrer deste capítulo). Nesses artigos são encontradas as descrições dos algoritmos utilizados para o *solver*, além das dificuldades encontradas para cumprir as restrições do problema.

O *solver* da equipe *NurseOptimizers* publicado por Römer e Mellouli (2016), utiliza um *framework*, proposto por Powell (2014), tanto para a modelagem do problema como também para gerar as soluções das instâncias. O *framework* foi elaborado especificamente para problemas de otimização estocástica multi-estágios (INRC-II) e contém diferentes métodos de otimização que podem ser combinados para criar poderosas estratégias híbridas a fim de resolver problemas de grande complexidade. A prova da eficiência dessa ferramenta pôde ser comprovada dentro da competição, visto que com sua utilização a equipe *NurseOptimizers* foi a grande vencedora, conseguindo alcançar para grande parte das instâncias soluções iguais ou próximas dos melhores resultados conhecidos.

O *solver* da equipe *Polytechnique Montreal* foi proposto por Legrain, Omer e Rosat (2018) e obteve a segunda colocação do INRC-II. A implementação foi baseada em um algoritmo estocástico *online* composto por algumas técnicas que já foram aplicadas de forma bem sucedida em problemas de otimização dinâmica, mas que nunca foram colocadas para trabalharem juntas. Entre as técnicas utilizadas ganham destaque o uso de um algoritmo *primal-dual* para otimização *online* e de um aproximador de média das amostras (SSA), sendo o primeiro responsável por gerar as escalas candidatas de cada semana, enquanto o SSA valida cada uma delas e guarda a melhor de todas.

O *solver* da equipe *SSHH* proposto em Kheiri et al. (2016) conseguiu a terceira colocação na competição e utilizou uma seleção baseada em sequência hiper-heurística. Neste método, os autores propuseram a utilização de 9 heurísticas de baixo nível, onde todas devem ser aplicadas a uma solução candidata, e possuem como maior desafio encontrar um método eficaz para determinar qual a melhor sequência de heurísticas a serem empregadas. A dificuldade foi solucionada por meio da elaboração de uma fórmula encarregada de calcular a probabilidade de determinada heurística ser selecionada. Apesar do *solver* não ter conseguido a primeira colocação na competição, no fim de todas as instâncias, acabou se mostrando o mais eficiente quando se trata de gerar de soluções factíveis.

Os próximos estudos não pertencem às equipes que participaram da competição, mas de alguma forma se relacionam com o tema do presente trabalho, ou tentaram resolver as instâncias do INRC-II após o término da competição.

Portella (2017) propôs para o problema do INRC-II um algoritmo baseado na metaheurística *Late Acceptance Hill Climbing* (LAHC) e um conjunto de movimentações para compor as soluções vizinhas. Apesar do LAHC ser relativamente simples em comparação aos algoritmos utilizados na competição, o método se mostrou promissor, visto que seria capaz de alcançar a sexta colocação geral da competição. Uma segunda contribuição do trabalho foi a proposta de um modelo matemático utilizando programação inteira não linear, o qual permite uma compreensão mais fácil das restrições em comparação com o modelo proposto em Ceschia et al. (2015b).

Com o mesmo objetivo do trabalho anterior, Gomes, Toffolo e Santos (2017) utilizaram a heurística *Variable Neighborhood Search* (VNS) junto a um algoritmo de geração de colunas, assim como uma heurística *relax-and-fix* para a geração de soluções factíveis. As escalas geradas aprimoraram em pelo menos 10%, 29 das melhores soluções conhecidas, sendo todas referentes a instâncias de 4 semanas. Contudo, as regras impostas pelo INRC-II não foram totalmente satisfeitas, visto que o *solver* dos autores continha os conceitos de programação paralela e se beneficiava de um tempo maior do que o disponível pela ferramenta de *benchmark* da competição. Além do que foi citado, o problema foi resolvido de forma global (sem gerar soluções para cada semana), quebrando assim a proposta multi-estágio da competição.

Existe um *survey* elaborado por Cheang et al. (2003) com diversos trabalhos sobre o problema de escalonamento de enfermagem. Por meio dessa contribuição é possível observar a diversidade de restrições que podem existir para o problema e também concluir quais métodos possuem os melhores resultados para as diferentes elaborações possíveis do problema. Burke et al. (2004) com o objetivo de compreender melhor as abordagens presentes na literatura para o PEE e auxiliar novas pesquisas, montaram uma coletânea de artigos da área e buscaram compreender os pontos fortes e fracos de cada abordagem. Entre esses estudos, 8 utilizavam algoritmos genéticos ou algum tipo de variação, e as conclusões sobre eles são de grande importância, pois demonstram em quais aspectos o algoritmo se torna eficiente e em quais acaba por prejudicar a qualidade da solução. Outro ponto interessante, é que um bom número de artigos presentes no estudo fizeram uso do mesmo conjunto de instâncias, permitindo assim uma sólida comparação entre os resultados de cada abordagem.

Sperandio (2015) fez uso de alguns algoritmos para resolver o problema de escalonamento de cirurgias eletivas nos hospitais portugueses, e os comparou a fim de entender melhor seus resultados, sendo o BRKGA um dos objetos de comparação. O autor concluiu que o BRKGA apresentou resultados muito acima da maioria dos algoritmos propostos (levando em consideração a qualidade das soluções e o tempo computacional gasto), contudo obteve resultados piores em comparação com a nova abordagem multiobjetivo proposta no trabalho. Segundo o autor, umas das possíveis causas do desempenho inferior ao novo algorítimo foi que o BRKGA apontou ser um algoritmo muito eficiente para cenários com

baixo nível de incertezas, o que não ocorre em um ambiente hospitalar, onde uma cirurgia pode levar várias horas e utilizar uma variedade de recursos diferentes. Outra conclusão importante sobre o BRKGA foi que o mesmo se mostrou extremamente dependente de uma heurística de refinamento, visto que a soberania dos resultados encontrados em relação aos demais métodos só foi possível com a implementação de uma busca local em cada solução gerada pelo algoritmo genético.

Ceschia e Schaerf (2018) utilizou um método de busca local baseado em uma longa vizinhança e guiada pela meta-heurística *Simulated Annealing* para gerar soluções ao problema proposto no INRC-II. Para a geração da solução inicial é utilizada uma estratégia para garantir que a restrição H2, referente ao requerimento mínimo seja respeitada. Foram utilizadas duas estruturas de vizinhança para a busca local, denominados de mudança de turnos/qualificações e trocas entre enfermeiros. Todas as regras impostas pela competição foram levadas em consideração durante os experimentos computacionais. Os resultados encontrados classificariam o algoritmo em primeiro lugar na competição, além disso, também foram encontradas melhores soluções conhecidas para um bom número de instâncias.

4 Metodologia

O principal objetivo deste capítulo é demonstrar como o Algoritmo Genético Hibrído de Chaves Aleatórias Viciadas (AGH) foi aplicado para o problema proposto no INRC-II. Para permitir um entendimento completo da metodologia, todas as funções presentes no AGH serão brevemente mencionadas e interligadas em sequência por meio de um fluxograma, por fim, as etapas de maior relevância e complexabilidade são descritas de forma mais detalhada e seu funcionamento representado por figuras.

4.1 Funções do AGH

- Gerar População: Cria um conjunto de matrizes, cada uma representa uma solução para o problema de escalonamento de enfermagem. Cada posição de uma matriz recebe dois números reais pertencentes ao intervalo de [0,1). Esse dois números, denominados de chaves aleatórias são responsáveis por representar o turno e qualificação que cada enfermeiro deve exercer em um determinado dia. Ao fim do processo é obtido um conjunto de matrizes de chaves aleatórias.

- Decodificação: Transforma a matriz de chaves aleatórias em uma solução real do problema. Esse procedimento é feito convertendo cada par de chaves da matriz em um turno e qualificação.

- Avaliar Fitness: Função responsável por avaliar o nível de aptidão de cada indivíduo. É implementada por meio do somatório das restrições descumpridas por uma solução, em sequência cada somatório é multiplicado pelo peso referentes as restrição não respeitadas.

- Ordenar Soluções: As soluções são ordenadas de forma crescente, levando em consideração o custo de cada solução (calculado pela função de avaliação).

- Dividir População: As soluções são divididas em dois conjuntos: elitista e não elitista. O conjunto elitista representa as melhoras soluções da população, e o não elitista o restante dos indivíduos.

- Copiar Soluções Elitistas: As soluções pertencentes ao grupo elitista são copiadas para a próxima geração do algoritmo.

- Mutação: Indivíduos de toda a população são selecionados aleatoriamente para o operador genético de mutação. Dependendo do número de iterações sem melhora do algoritmo é aplicada uma busca local nos indivíduos selecionados.

- Seleção dos Pais para o cruzamento: Um pai elitista e outro não elitista são selecionados aleatoriamente para o operador genético de *crossover*/cruzamento.
- Cruzamento: Os genes dos pais são combinados para a geração de um novo indivíduo. O processo é feito pelo método de um ponto de corte.

Ao fim do tempo disponibilizado pela ferramenta de *benchmark* o algoritmo gera como saída o melhor indivíduo encontrado após todas as iterações, ou seja, a solução com o menor custo possível. As etapas do AGH citadas acima são ilustradas pelo fluxograma da Figura 10.

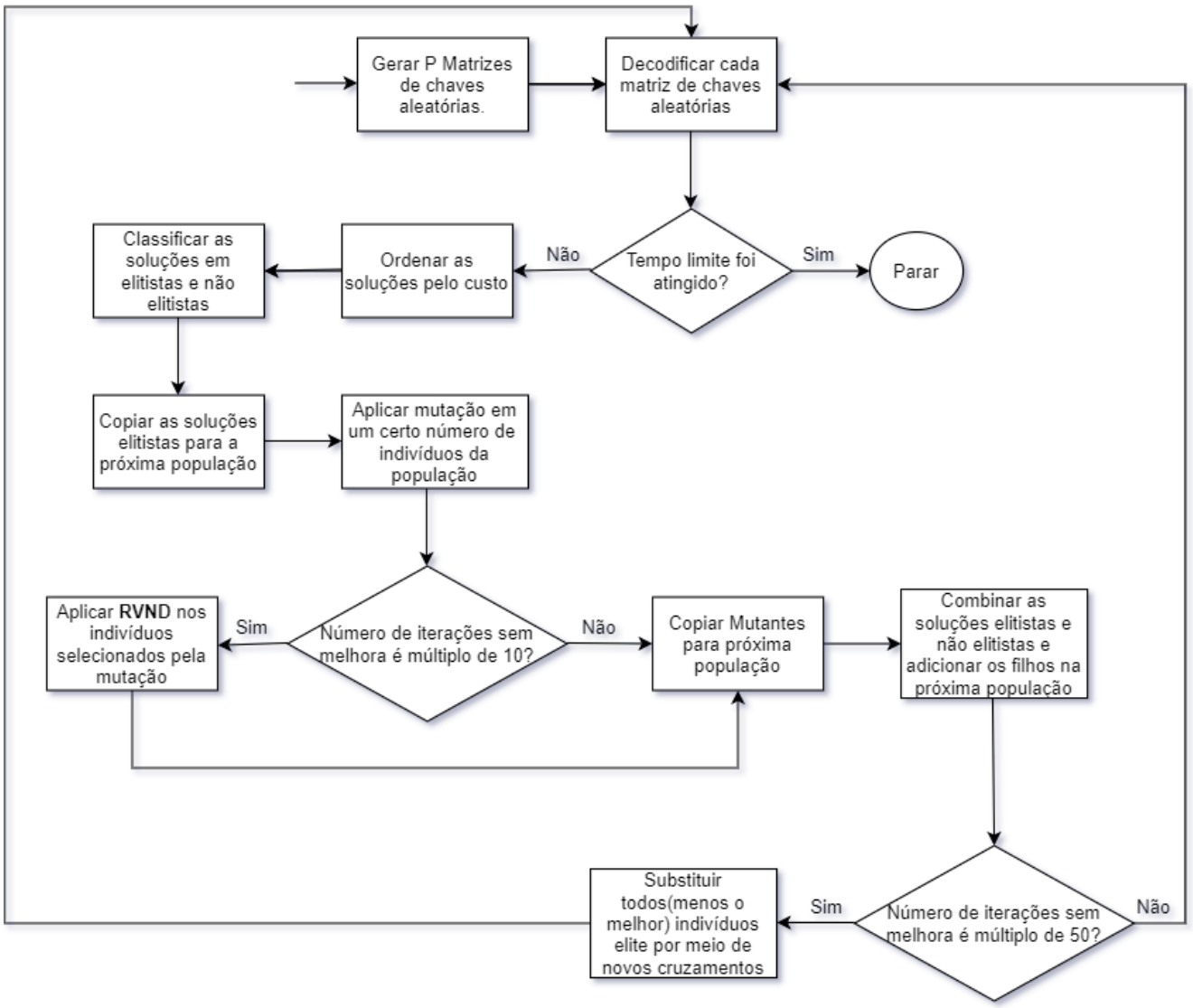

Figura 10 – Fluxograma AGH.

4.2 Construção de soluções por meio de chaves aleatórias

A implementação do AGH para o problema proposto no INRC-II tem como início a atribuição de duas chaves aleatórias a todas as posições das matrizes geradas pelo algoritmo, onde as linhas representam o número de enfermeiros definidos pela instância analisada e as colunas são referentes aos 7 dias da semana. A primeira chave aleatória corresponde aos turnos que um enfermeiro pode trabalhar, enquanto a segunda chave representa as competências que o enfermeiro está apto a exercer. A Figura 11 é referente a uma instância de 3 enfermeiros, e representa uma solução semanal do problema por meio de chaves aleatórias.

	D1	D2	D3	D4	D5	D6	D7
N0	< 0.52 0.37 >	< 0.62 0.33 >	< 0.02 0.83 >	< 0.92 0.57 >	< 0.14 0.39 >	< 0.78 0.62 >	< 0.25 0.23 >
N1	< 0.50 0.89 >	< 0.01 0.03 >	< 0.99 0.03 >	< 0.72 0.18 >	< 0.22 0.46 >	< 0.59 0.66 >	< 0.48 0.44 >
N2	< 0.91 0.73 >	< 0.24 0.09 >	< 0.58 0.96 >	< 0.21 0.31 >	< 0.84 0.29 >	< 0.55 0.88 >	< 0.05 0.62 >

Figura 11 – Representação de uma solução semanal do INRC-II com chaves aleatórias.

4.3 Decodificação

O decodificador do BRKGA é um algoritmo determinístico responsável por transformar um vetor ou matriz de chaves aleatórias em uma solução de um determinado problema de otimização. O decodificador deste estudo atua sobre cada posição da matriz, até que chegue ao seu fim, e pode ser dividido em duas partes: a primeira é responsável por definir em qual turno o enfermeiro irá trabalhar. A segunda define qual a competência que o enfermeiro será escalado para trabalhar. É importante ressaltar que caso o decodificador atribua o turno de determinado enfermeiro como um dia de folga, o processo para definir a competência do enfermeiro não é executado, visto que não faz sentido atribuir uma competência para um enfermeiro que não irá trabalhar naquele dia.

4.3.1 Decodificação dos turnos

Visando melhorar as restrições referentes aos dias consecutivos de folga e turnos trabalhados (S2 e S3), o processo de decodificação dos turnos leva em conta o número de dias máximo e mínimo que um determinado enfermeiro deve trabalhar, sendo esses limites definidos pelos tipos de contratos presentes nas instâncias da competição. Para exemplificar o processo de decodificação dos turnos será levado em conta o par de chaves aleatórias $< 0.52\ 0.37 >$ referentes ao enfermeiro N0 para do dia 1 da Figura 10 e (17,25)

como número mínimo e máximo de dias a serem trabalhados em um horizonte de planejamento de 4 semanas (28 dias). Inicialmente é realizada uma média dos dias mínimo e máximo a serem trabalhados 17+25/2 = 21, o resultado indica que o enfermeiro N0 deve trabalhar em média 21 dos 28 dias possíveis, o que equivale a 75% do número total de dias da semana (28). Para calcular a média de dias a serem atribuídos como folga basta subtrair a média dos dias trabalhados pelo total, 1 - 0.75 = 0.25, ou seja, em 25% dos dias o enfermeiro N0 deve receber uma folga.

Feito os cálculos, o processo de decodificação dos turnos pode efetivamente começar. Primeiramente é verificado se a chave referente ao turno (0.52) é menor ou igual a porcentagem de dias médios a serem atribuídos como folga (0.25), caso seja, o enfermeiro N0 para o dia 1 deverá receber uma folga. Caso a primeira verificação seja falsa deve ser realizado o seguinte cálculo: dividir a porcentagem média de dias a serem trabalhados (0.75) pelo número total de turnos presentes no vetor de turnos representado na Figura 11. Sendo que não deve ser contabilizado o dia de folga presente na posição 0 do vetor, somando assim 4 possibilidades diferentes de turnos.

A conta a ser feita então é 0.75/4, com resultado igual a 0.1875. O próximo passo é somar iterativamente o valor de 0.1875 a porcentagem de dias médios de folga (0.25) até que o valor seja maior ou igual ao valor contido na chave referente aos turnos (0.52). Simulando os passos, na primeira iteração a conta seria 0.25 + 0.1875 = 0.4375, já na segunda iteração, 0.4375 + 0.1875 = 0.625. Como 0.625 é maior que 0.52, o processo é encerrado, sendo necessárias duas iterações até que esse resultado fosse obtido. Por fim, é comparado o número de iterações necessárias do processo anterior com as posições do vetor de turnos da Figura 12. Como foram necessárias duas iterações, é verificado qual o turno contido na posição 2 do vetor de turnos. A posição contém o turno *Day*, então o enfermeiro N0 para o dia 1 deve ser escalado para trabalhar no turno *Day*.

	0	1	2	3	4
Turnos	-	Early	Day	Late	Night

Figura 12 – Distribuição de turnos dos enfermeiros.

4.3.2 Decodificação das competências

O processo de decodificação de competências funciona de maneira mais simples que a dos turnos. A exemplificação também será realizada levando em consideração o par de chaves aleatórias < 0.52 0.37 > referente ao enfermeiro N0 para do dia 1 da Figura 11. O valor pertencente a competência (0.37) deve ser multiplicado pelo número de competências que o enfermeiro pode exercer, o qual varia entre 1, 2 ou 3, visto que em

nenhuma das instâncias da competição há a presença de um enfermeiro podendo exercer 4 funções diferentes. A Figura 13 representa uma matriz de funções, a qual contém cada uma das competências que determinado enfermeiro pode exercer. Como o valor do par de chaves aleatórias utilizado para exemplificar a decodificação é referente ao enfermeiro N0, ao analisar a Figura 13 é possível concluir que o enfermeiro está capacitado a exercer 3 diferentes competências, sendo assim, o valor da chave de competências (0.37) deve ser multiplicado por 3 (0.37 x 3 = 1.11) , como o arrendondamento para baixo do resultado da operação é 1, a competência a ser atribuída ao enfermeiro é aquela contida na coluna 1 e linha refente ao enfermeiro N0 da matriz de funções, ou seja, *Nurse*.

	0	**1**	**2**
N_ 0	Headnurse	Nurse	Caretaker
N_1	Nurse	Caretaker	
N_2	Nurse		

Figura 13 – Matriz de funções dos enfermeiros.

A Figura 14 representa uma solução semanal fictícia encontrada após o processo de decodificação, onde as linhas representam os enfermeiros e as colunas os 7 dias da semana.

	D1	**D2**	**D3**	**D4**	**D5**	**D6**	**D7**
N0	D nurse	L headnurse	-	N nurse	-	L nurse	E headnurse
N1	D caretaker	-	N nurse	L nurse	E nurse	D caretaker	D nurse
N2	N nurse	-	D nurse	-	N nurse	D nurse	-

Figura 14 – Solução encontrada pelo decodificador.

Como já foi mostrado durante a descrição do problema deste trabalho, as restrições fortes impostas pelo INRC-II devem ser respeitadas para que a solução encontrada seja válida. A implementação do par de chaves aleatórias para cada enfermeiro e dia da semana, aliado ao mecanismo implementado no decodificador, garantem que algumas restrições fortes do problema sejam sempre respeitadas. A restrição H1 é garantida pela implementação de somente uma chave aleatória de turnos para cada dia da semana, dessa forma um enfermeiro nunca será alocado para trabalhar em mais de um turno por dia.

A restrição H4 é garantida durante a decodificação, onde o valor da chave aleatória referente as competências é multiplicado somente pelo número de competências que cada enfermeiro pode exercer.

As restrições H2 e H3 não podem ser garantidas com a representação de soluções em chaves aleatórias ou pelo decodificador, a solução para esse problema foi durante a execução da função de avaliação, penalizar fortemente as ocorrências dessas restrições nas escalas encontradas. Dessa forma, as soluções que não respeitam as restrições H2 e H3, tendem, conforme o número de iterações for aumentando, a não fazerem mais parte das novas populações geradas pelo algoritmo.

4.4 Função de Avaliação

A função de avaliação foi elaborada baseando-se no modelo matemático proposto em Portella (2017). As Tabelas 1, 2 e 3 desta seção são referentes, respectivamente, aos conjuntos, parâmetros e variáveis utilizadas na função de avaliação.

Tabela 1 – Notação dos conjuntos utilizados para a função de avaliação.

Conjuntos	**Definição**
$d \in D$	Conjunto de estágios (semanas).
$s \in S$	Conjunto de turnos.
$s \in S'$	Conjunto de turnos, excluindo o turno referente a folga.
$k \in K$	Conjunto de competências de um enfermeiro.
$n \in N$	Conjunto de enfermeiros.
$(s1, s2) \in F$	Conjunto de sucessões proibidas, s1 não pode ser precedido por s2.
$(i, j) \in B$	Conjunto de tuplas (i,j) tal que $u \in D, v \in D$ onde $u \leq v$. Cada par representa um possível bloco de alocações consecutivas em um estágio.
$P_{nds} \in \{0, 1\}$	Assume o valor 1 caso o enfermeiro n prefira não ser alocado no turno s do dia d.
$P_{ndsk} \in \{0, 1\}$	Assume o valor 1 caso o enfermeiro n não queira ser alocado em nenhum turno do dia d, ou seja, ele deseja uma folga no dia d.

Fonte: (PORTELLA, 2017).

Tabela 2 – Notação dos parâmetros utilizados para a função de avaliação.

Parâmetros	**Definição**
W^{H_2}, W^{H_3}	Peso das restrições H2 e H3.
$W^{S_1}, ..., W^{S_7}$	Peso das restrições S1 a S7.
$V_n \in \{0, 1\}$	Assume o valor 1 caso o enfermeiro n precise trabalhar todo o fim de semana.
$\delta^e \in \mathbb{N}^*$	Custo das alocações consecutivas no final do estágio e - 1.
$L_n^+ \in \mathbb{N}^*$	Número máximo de dias consecutivos trabalhados para o enfermeiro n.
$L_n^- \in \mathbb{N}^*$	Número mínimo de dias consecutivos trabalhados para o enfermeiro n.
$L_{ns}^+ \in \mathbb{N}^*$	Número máximo de alocações consecutivas do turno s para o enfermeiro n.
$L_{ns}^- \in \mathbb{N}^*$	Número mínimo de alocações consecutivas do turno s para o enfermeiro n.
$G_n^+ \in \mathbb{N}^*$	Número máximo de folgas consecutivas para o enfermeiro n.
$G_n^- \in \mathbb{N}^*$	Número mínimo de folgas consecutivas para o enfermeiro n.
$Q_n^+ \in \mathbb{N}^*$	Número máximo de dias trabalhados levando em conta todos os estágios para o enfermeiro n.
$Q_n^- \in \mathbb{N}^*$	Número mínimo de dias trabalhados levando em conta todos os estágios para o enfermeiro n.
$R_n^+ \in \mathbb{N}^*$	Número máximo de fins de semana trabalhados por enfermeiro.

Fonte: (PORTELLA, 2017).

Tabela 3 – Notação das variáveis utilizadas para a função de avaliação.

Variáveis	**Definição**
$x^{e}_{ndsk} \in \{0,1\}$	Assume o valor 1 quando o enfermeiro n está alocado no turno s com a competência k no estágio e.
$C^{H_2}_{dsk} \in \mathbb{N}^*$	Número de enfermeiros faltando no turno s do dia d para atingir a cobertura mínima.
$C^{H_3}_{nds_1s_2} \in \mathbb{N}^*$	Número de turnos com sucessões inválidas na solução.
$C^{S_1}_{dsk} \in \mathbb{N}^*$	Número de enfermeiros faltando no turno s do dia d para atingir a cobertura ótima.
$C^{S_{2a}}_{nij} \in \mathbb{N}^*$	Número de dias excedendo o limite L^+_n no bloco de trabalho $(i,j) \in B$ para um enfermeiro n no estágio e.
$C^{S_{2b}}_{nij} \in \mathbb{N}^*$	Número de dias faltando para atingir o limite L^-_n no bloco de trabalho $(i,j) \in B$ para um enfermeiro n no estágio e.
$C^{S_{2c}}_{nij} \in \mathbb{N}^*$	Número de dias excedendo o limite L^+_{ns} no bloco de trabalho $(i,j) \in B$ para um enfermeiro n considerando apenas o turno s no estágio e.
$C^{S_{2d}}_{nij} \in \mathbb{N}^*$	Número de dias faltando para atingir o limite L^-_{ns} no bloco de trabalho $(i,j) \in B$ para um enfermeiro n considerando apenas o turno s no estágio e.
$C^{S_{3a}}_{nij} \in \mathbb{N}^*$	Número de dias de folga excedendo o limite G^+_n no bloco de trabalho $(i,j) \in B$ para um enfermeiro no estágio e.
$C^{S_{3b}}_{nij} \in \mathbb{N}^*$	Número de dias de folga faltando para atingir o limite G^-_n no bloco de trabalho $(i,j) \in B$ para um enfermeiro n no estágio e.
$C^{S_{6a}}_{n} \in \mathbb{N}^*$	Número de dias de trabalho excedendo o limite Q^+_n para um enfermeiro n para todos os estágios.
$C^{S_{6b}}_{n} \in \mathbb{N}^*$	Número de dias de trabalho faltando para atingir o limite Q^-_n para um enfermeiro n para todos os estágios.
$C^{S_{7}}_{n} \in \mathbb{N}^*$	Número de fins de semana trabalhados excedendo o limite R^+_n para um enfermeiro n para todos os estágios.

Fonte: (PORTELLA, 2017).

Uma solução no PEE é avaliada conforme a função 4.1. O cálculo é feito por meio do somatório do número de vezes em que cada restrição é desrespeita. Em sequência o resultado do somatório de cada uma das restrições é multiplicado pelo seu devido peso. Como as restrições fortes H2 e H3 não são garantidas, foi definido para elas um peso muito maior que as fracas. O valor foi empiricamente configurado para 200, e se mostrou suficientemente forte para guiar o AGH a não encontrar soluções inviáveis.

$$\min H_2 + H_3 + S_1 + S_2 + S_3 + S_4 + S_5 + S_6 + S_7 \tag{4.1}$$

$$H_2 = W^{H_2} \sum_{d \in D} \sum_{s \in S} \sum_{k \in K} C^{H_2}_{dsk} \tag{4.2}$$

$$H_3 = W^{H_3} \sum_{n \in N} \sum_{d \in D} \sum_{(s_1, s_2) \in F} C^{H_3}_{nds_1s_2} \tag{4.3}$$

$$S_1 = W^{S_1} \sum_{d \in D} \sum_{s \in S} \sum_{k \in K} C^{S_1}_{dsk} \tag{4.4}$$

$$S_2 = W^{S_{2ab}} \sum_{n \in N} \sum_{(i,j) \in B} C^{S_{2a}}_{nij} + C^{S_{2b}}_{nij} \quad + \quad W^{S_{2cd}} \sum_{n \in N} \sum_{(i,j) \in B} \sum_{s \in S'} C^{S_{2c}}_{nijs} + C^{S_{2d}}_{nijs} \tag{4.5}$$

$$S_3 = W^{S_3} \sum_{n \in N} \sum_{(i,j) \in B} C^{S_{3a}}_{nij} + C^{S_{3b}}_{nij} \tag{4.6}$$

$$S_4 = W^{S_4} \sum_{n \in N} \sum_{d \in D} \sum_{s \in S'} \sum_{k \in K} x^e_{ndsk} \times P_{nds} + x^e_{ndsk} \times P_{nd} \tag{4.7}$$

$$S_5 = W^{S_5} \sum_{n \in N} V_n \times \left(1 - \prod_{d \in W} \sum_{s \in S'} \sum_{k \in K} x^e_{ndsk} \right) \tag{4.8}$$

$$S_6 = W^{S_6} \sum_{n \in N} C^{S_{6a}}_{n} + C^{S_{6b}}_{n} \tag{4.9}$$

$$S_7 = W^{S_7} \sum_{n \in N} C^{S_7}_{n} \tag{4.10}$$

4.5 Mutação

O operador genético de mutação é de extrema importância para que as soluções não fiquem presas a ótimos locais. O primeiro passo para sua aplicação é o sorteio de uma solução entre toda a população presente no algoritmo (a única solução não passível de ser

escolhida é a melhor solução de toda a população). Em sequência é definido um número de enfermeiros da solução (linhas) que devem ter seus alelos alterados. O número varia de acordo com o número de iterações sem melhora, sendo que quanto mais iterações sem melhora o algoritmo tiver maior deve ser o número de linhas a serem alteradas durante a mutação.

O próximo passo é definir em quais colunas da matriz a operação de mutação deve ser realizada. Como o número de colunas é fixo para todas as instâncias (7 dias da semana), esse procedimento pode ser feito através do sorteio de um número inteiro entre 0 e 6. Após a definição das linhas e colunas a mutação pode ser aplicada e funciona de forma muito simples: devem ser geradas duas novas chaves aleatórias para as linhas e colunas sorteadas, possibilitando assim que o turno e competência dos enfermeiros em um determinado dia sejam alterados. A Figura 15 é referente a operação de mutação para um cenário com 3 enfermeiros, sendo os 3 escolhido para passarem pelo operador genético de mutação. Para cada linha escolhida, é sorteado um dos sete dias da semana (representado pela cor vermelho), sendo em sequência gerado um novo par de chaves aleatórias para a posição sorteada da matriz.

Solução escolhida para operação de mutação

	D1	D2	D3	D4	D5	D6	D7
N0	< 0.52 0.37 >	< 0.62 0.33 >	< 0.02 0.83 >	< 0.92 0.57 >	< 0.14 0.39 >	< 0.78 0.62 >	< 0.25 0.23 >
N1	< 0.50 0.89 >	< 0.01 0.03 >	< 0.99 0.03 >	< 0.72 0.18 >	< 0.22 0.46 >	< 0.59 0.66 >	< 0.48 0.44 >
N2	< 0.91 0.73 >	< 0.24 0.09 >	< 0.58 0.96 >	< 0.21 0.31 >	< 0.84 0.29 >	< 0.55 0.88 >	< 0.05 0.62 >

Solução após operação de mutação

	D1	D2	D3	D4	D5	D6	D7
N0	< 0.52 0.37 >	< 0.62 0.33 >	< 0.02 0.83 >	< 0.21 0.83 >	< 0.14 0.39 >	< 0.78 0.62 >	< 0.25 0.23 >
N1	< 0.50 0.89 >	< 0.61 0.28 >	< 0.99 0.03 >	< 0.72 0.18 >	< 0.22 0.46 >	< 0.59 0.66 >	< 0.48 0.44 >
N2	< 0.91 0.73 >	< 0.24 0.09 >	< 0.58 0.96 >	< 0.21 0.31 >	< 0.84 0.29 >	< 0.95 0.54 >	< 0.05 0.62 >

Figura 15 – Aplicação do operador de mutação do AGH para uma solução sorteada.

4.6 Cruzamento

A operação de cruzamento utilizada neste trabalho foi a por um ponto de corte. Inicialmente devem ser escolhidos dois indivíduos pais para que a operação de *crossover*

possa ser realizada, sendo um pertencente a população elitista e outra da não elitista. O passo seguinte é sortear para cada linha da matriz um número entre 0 e 6 (referentes aos dias da semana) para ser utilizado como ponto de corte. Após o sorteio é verificado se o resultado é maior ou igual a 3, caso seja, os alelos do indivíduo elitista são mantidos até o valor sorteado para o ponto de corte, e as posições do ponto de corte para frente são copiadas do indivíduo não elitista para o elitista. Contudo, caso o valor sorteado seja menor que 3, os alelos da solução não elitista são copiados para o elitista até que seja alcançado o resultado do sorteio, e os alelos do indivíduo elitista da posição sorteada para frente são mantidos.

A implementação do cruzamento é feita da maneira descrita para que o novo indivíduo filho gerado sempre contenha mais características do indivíduo elitista do que do não elitistas. Ao fim do processo somente o indivíduo elitista é modificado, sendo este agora um novo filho gerado pelo cruzamento. A Figura 16 representa uma operação de cruzamento para uma mesma linha de duas soluções diferentes e com ponto de corte igual a 3. É possível observar que existem mais alelos do indivíduo elite a esquerda do ponto de corte, logo esses alelos devem ser repassados para o indivíduo filho. O restante da solução filho é preenchida com os alelos contidos a direita do ponto de corte da solução não elitista.

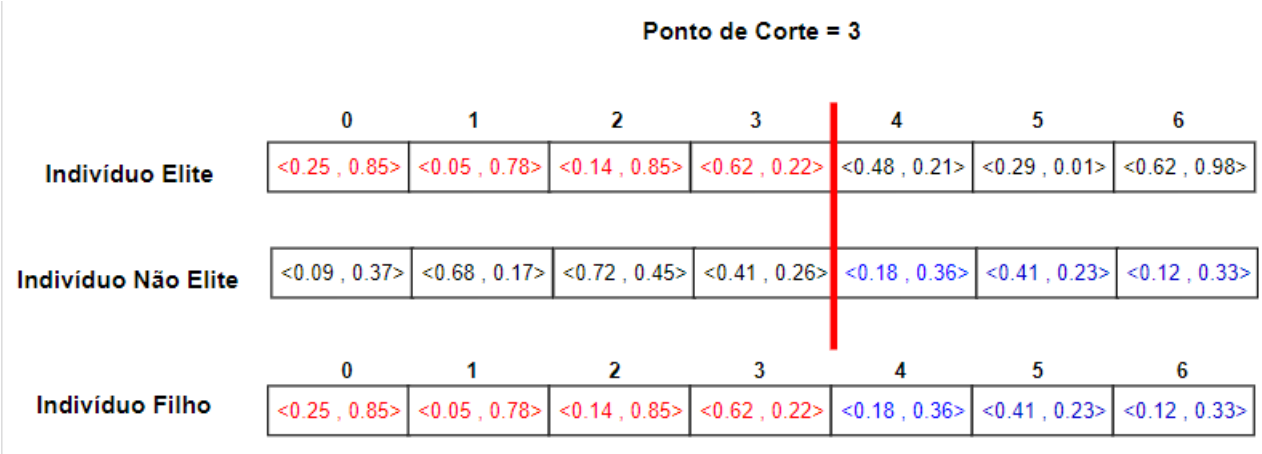

Figura 16 – Representação do cruzamento para um ponto de corte de tamanho 3.

Já na figura 17 o ponto de corte é igual a 2, sendo possível observar que existem mais alelos do indivíduo elite a direita do ponto de corte, logo esses alelos devem ser repassados para o indivíduo filho. O restante da solução filho é preenchida com os alelos contidos a esquerda do ponto de corte da solução não elitista.

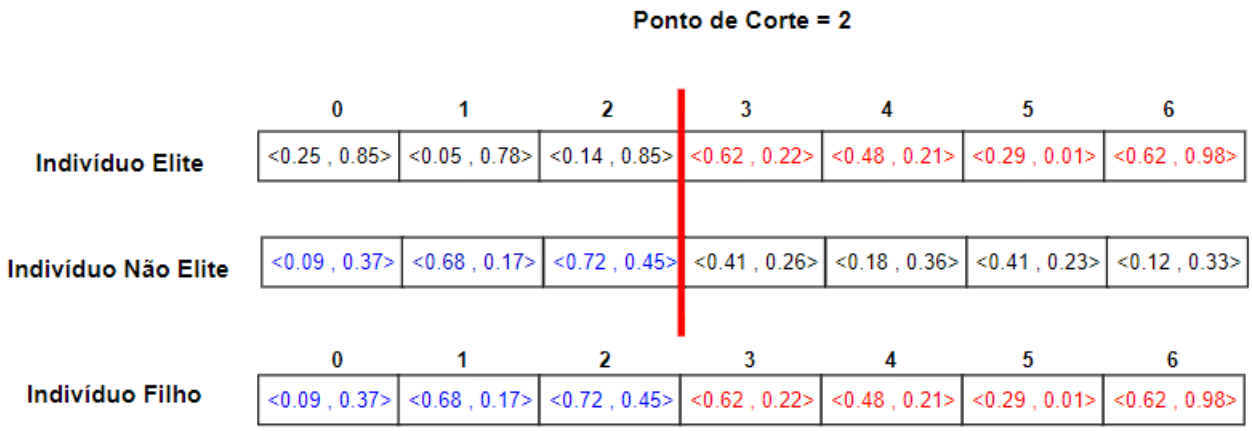

Figura 17 – Representação do cruzamento para um ponto de corte de tamanho 2.

4.7 Estimativas das Restrições S6 e S7

Diferentemente das outras restrições impostas pelo INRC-II, as restrições S6 e S7 só podem ser efetivamente avaliadas ao fim do horizonte de planejamento. Logicamente para a construção de *solvers* competitivos, os participantes tiveram que implementar estratégias com o objetivo de reduzir as ocorrências dessas restrições nas escalas semanais.

4.7.1 Estimativa S6

A restrição S6 estabelece um número mínimo e máximo de dias a serem trabalhados no período de 4 ou 8 semanas para cada tipo de contrato da instância. Para exemplificar a estratégia de estimativa utilizada neste trabalho será utilizado os valores $(12, 20)$, representando o número mínimo e máximo de dias a serem trabalhados por um enfermeiro com contrato do tipo *fulltime*, e um horizonte de planejamento de 4 semanas.

Sendo E o número de semanas do horizonte de planejamento, seu valor inicial é $E = 4$. A estimativa consiste então em dividir o número de dias máximo e mínimo permitidos pelo número de semanas E que ainda não foram processadas. Portanto para a primeira semana, os valores mínimos e máximos (12,20) são divididos respectivamente por 4. $12/4 = 3$ e $20/4 = 5$. Esses valores indicam que o enfermeiro com contrato *fulltime* deve trabalhar na semana 1 nó mínimo 3 dias e no máximo 5 dias. Caso o enfermeiro tenha trabalhado menos de 3 dias, é calculada a diferença do número de dias trabalhados para o número 3, e o resultado multiplicado pelo peso da restrição S6. O mesmo procedimento é aplicado caso o enfermeiro tenha trabalhado mais do que os 5 dias permitidos pela estimativa do número máximo de dias a serem trabalhados.

Para a semana seguinte são estipulados novos limites. Primeiramente é calculado o número de dias mínimo e máximo que ainda podem ser trabalhados em relação a todo

o horizonte de planejamento. Esse processo é feito pela subtração dos limites atuais pelo número de dias que efetivamente o enfermeiro trabalhou na semana anterior. Supondo que um enfermeiro com contrato *fulltime* tenha trabalhado 5 dias na semana inicial, os novos limites mínimo e máximo em relação a todo o horizonte de planejamento serão respectivamente, 12 - 5 = 7 e 20 - 5 = 15 (7,15). O próximo passo é dividir os novos valores encontrados pelo número de semanas restantes a serem processadas, neste caso, $E = 3$. O cálculo será $7/3 = 2.33$ e $15/3 = 5$, sendo que para o limite mínimo é retido como resultado e truncamento do cálculo e para o limite máximo o arredondamento para cima. Logo os novos limites mínimo e máximo para a semana 2 serão, respectivamente, (2,5). Todos as etapas descritas são repetidas até que o horizonte de planejamento de 4 semanas chegue ao seu fim.

4.7.2 Estimativa S7

A restrição S7 estabelece para cada tipo de contrato um número máximo de finais de semana a serem trabalhados durante todo o horizonte de planejamento. É considerado como final de semana trabalhado se ao menos em um dos dias, 5 ou 6, o enfermeiro está escalado para trabalhar. O cálculo dessa restrição é implementado inicialmente verificando se o enfermeiro trabalhou em algum dos dias do fim de semana. Caso tenha trabalhado é subtraído o valor 1 do número máximo de finais de semana trabalhados estipulados para o tipo de contrato que o enfermeiro possuí.

A partir do momento em que o número máximo de finais de semanas trabalhados para determinado tipo de contrato alcança o valor 0, qualquer enfermeiro que contenha esse contrato e venha a ser escalado para trabalhar em um fim de semana é penalizado com o peso estabelecido pela restrição S7.

4.8 Movimentos RVND

O RVND implementado neste trabalho contém duas vizinhanças diferentes, apesar deste número ser pequeno, ambas podem ser consideradas vizinhanças largas e portanto exploram com intensidade o espaço de soluções. Os movimentos utilizaram uma busca pela primeira melhora, visto que uma busca pela melhor melhora neste problema é computacionalmente muito cara, tornando inviável a sua aplicação no INRC-II, visto que para cada instância há um tempo limite de execução a ser respeitado pelos *solvers* desenvolvidos pelos competidores.

4.8.1 Mudança de turnos e qualificação

O primeiro movimento consiste em mudanças aleatórias de turno e função para cada posição da matriz solução. Para que as posições da matriz não sigam sempre a mesma

ordem toda vez que o movimento é realizado, é preenchido e posteriormente embaralhado um vetor contendo todas as posições possíveis da matriz solução. Para cada posição, é inicialmente feita uma mudança de turnos, em sequência devem ser testadas, para aquele turno modificado, todas as qualificações que o enfermeiro selecionado está apto a exercer. Desta forma são testadas todas as combinações de turno e qualificação possíveis para cada posição selecionada posição selecionada. A figura 18 contém um posição da matriz em vermelho, representando o sorteio de uma posição da matriz.

	D1	D2	D3	D4	D5	D6	D7
N0	D nurse	L headnurse	-	N nurse	-	L nurse	E headnurse
N1	D caretaker	-	N nurse	L nurse	E nurse	D caretaker	D nurse
N2	N nurse	-	D nurse	-	N nurse	D nurse	-

Figura 18 – Posição sorteada para mudança de turnos e qualificação.

Ao observar a figura 19 é possível perceber que a posição sorteada da Figura 18 teve seu turno e qualificação alterados, representando assim uma melhora após testar as combinações de turnos e qualificações.

	D1	D2	D3	D4	D5	D6	D7
N0	D nurse	L headnurse	-	N nurse	-	L nurse	E headnurse
|N1	D caretaker	-	N nurse	N caretaker	E nurse	D caretaker	D nurse
N2	N nurse	-	D nurse	-	N nurse	D nurse	-

Figura 19 – Melhora após troca de turnos e qualificação.

4.8.2 Troca de blocos entre enfermeiros

O segundo movimento consiste na troca de blocos de alocações entre dois enfermeiros. Inicialmente é sorteado uma posição inicial e um tamanho de bloco para que as trocas sejam feitas. O passo seguinte é a divisão dos enfermeiros em grupos de acordo com a compatibilidade das funções que podem exercer. Cada enfermeiro pertence a um grupo de enfermeiros que contém exatamente o mesmo número de funções a serem exercidas, além disso essas funções devem ser exatamente as mesmas. Essa divisão em grupos

é necessária para que após a troca de blocos entre os enfermeiros, a restrição H4, a qual determina que um enfermeiro só pode trabalhar em uma qualificação que está apto a exercer não seja descumprida. Por fim, para cada grupo, é realizada a troca de blocos entre dois enfermeiros para todas as combinações sem repetições de enfermeiros.

A figura 20 representa o processo descrito anteriormente. Dado dois enfermeiros compatíveis em relação as suas funções foi inicialmente sorteado o dia 3 como ponto de corte, sendo em sequência sorteado um tamanho de bloco igual a 3, portanto o trecho a ser trocado entre os dois enfermeiros é entre os dias 4 e 6. Após a troca é possível observar que, para o bloco definido no processo anterior, o enfermeiro N1 recebeu o conteúdo do enfermeiro N2, assim como o N2 recebeu o conteúdo do N1.

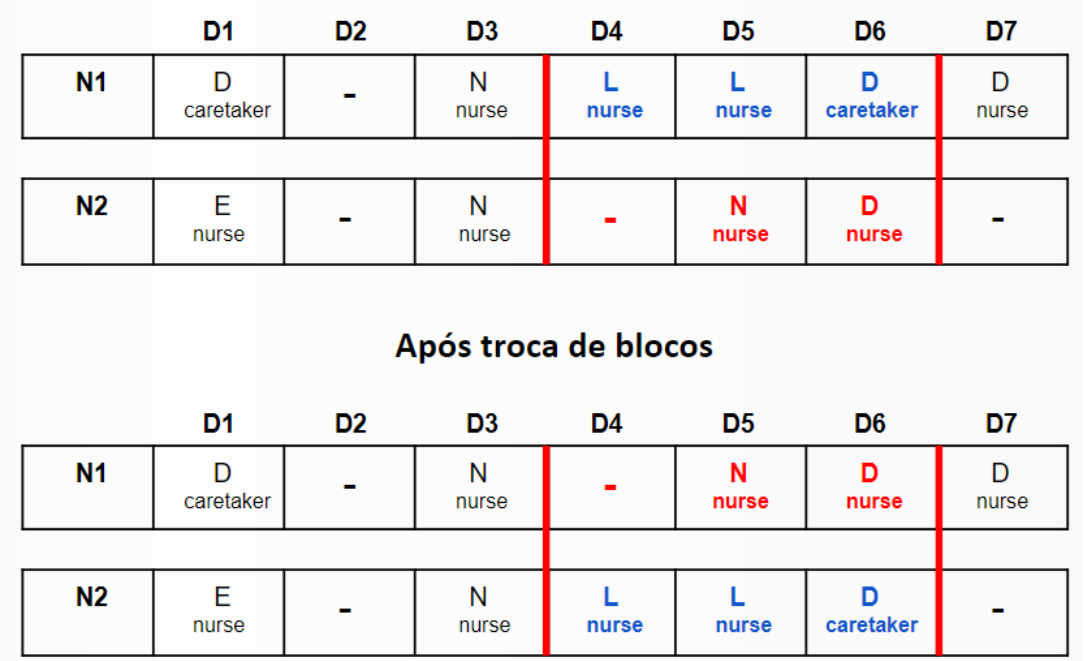

Figura 20 – Troca de blocos entre enfermeiros.

5 Resultados Computacionais

O algoritmo proposto neste trabalho foi aplicado no conjunto de instâncias *Late* disponibilizadas no site oficial do INRC-II. As instâncias desse tipo são divididas em cenários com 30, 40, 50, 60, 80, 100 e 120 enfermeiros, sendo que cada grupo de enfermeiros contém 4 instâncias (2 para uma escala de 4 semanas e 2 para 8 semanas), totalizando assim 28 instâncias.

As instâncias são nomeadas de forma que seja possível identificar o número de enfermeiros, tamanho do horizonte de planejamento (escala de 4 ou 8 semanas), arquivo de histórico e quais arquivos de dados da semana devem ser utilizados. A instância n03w4-2-9-9-2-1 indica a presença de 30 enfermeiros, um horizonte de planejamento referente a 4 semanas, arquivo de histórico número 2, e arquivos de dados da semana números 9, 9, 2, 1.

Todos os experimentos foram realizados serão executados em um *processador Ryzen 5 2600x com frequência base de 3.6 GHZ e 4.25 GHZ com Max Turbo, 16 GB de memória RAM* e uma maquina virtual *Linux Ubuntu 18.04 LTS de 64 bits*. O *solver* foi implementado na linguagem C++ e o compilador utilizado foi o *gcc (versão 7.40)*. Todas as gerações de números aleatórios foram realizadas utilizando o *Mersenne Twister* Matsumoto e Nishimura (1998). Para uma comparação justa dos resultados encontrados com os dos outros participantes do INRC-II todas as regras impostas pela competição serão respeitas. O problema foi resolvido por meio de escalas semanais, o *solver* foi executado levando em conta somente o tempo disponibilizado pela ferramenta de *benchmark*, e por fim, todos os resultados obtidos foram avaliados pelo validador da competição. Na tabela 4 são fornecidos, levando em conta o ambiente experimental descrito, os tempos limites de execução calculados pela ferramenta de *benchmark* . Os tempos estão representados em segundos e são válidos para todas as instâncias que contém o mesmo número de enfermeiros.

Tabela 4 – Tempo disponibilizado pela ferramenta de *benchmark*

Enfermeiros	30	40	50	60	80	100	120
Tempo	47.61(s)	83.31(s)	119.01(s)	154.72(s)	226.13(s)	297.53(s)	368.94(s)

Fonte: Do Autor

Ao total, o INRC-II teve 15 equipes participantes de diversos países diferentes, sendo que grande parte delas foram compostas por experientes pesquisadores na área da otimização combinatória. Entre as equipes também se encontravam as empresas *ORTEC* e *BulletSolutions*, e dois representantes brasileiros, a GOAL representando a Universidade Federal de Ouro Preto e a INF_UFRGS da Universidade Federal do Rio Grande do

Sul. Infelizmente o site oficial da competição não forneceu os nomes de todas as equipes participantes, deixando as equipes que não conseguiram chegar até a etapa final sem os nomes dos integrantes e da instituição que estavam vinculados. A tabela 5 fornece as informações encontradas sobre todas as equipes participantes, sendo as 7 primeiras ordenadas conforme a classificação final da competição.

Tabela 5 – Participantes do INRC-II

Id	**Equipe**	**Membros da equipe**	**Instituição**
E1	NurseOptimizers	M.Römer, T.mellouli	Martin Luther University
E2	Polytechnique Montreal	L.Antoine, O.Jérémy, R.Samuel	Polytechnique Montreal
E3	SSHH	A.Kheiri	University of Exeter
E4	Hust.Smart	Z.Su, Z. Wang, Z.Lü	Huazhong University
E5	ORTEC	H.Jin, G. Post, E. van der Venn	ORTEC
E6	LabGOL	(F. Nicolino *et al*)	Univ. degli Studi di Firenze
E7	ThreeJohns	T. Ioannis, S. loannis, B.Grigorios	University of Patras
E8	ScheduleNurse	Não informado	Não informado
E9	INF-UFRGS	T. Wickert, C. Sartori	UFRGS
E10	Bullet Solutions	Não informado	Bullet Solutions
E11	eatts-inrc2-2	Não informado	Não informado
E12	GOAL	Não informado	UFOP
E13	Team Puma	Não informado	Não informado
E14	in-optima-inrc-ii	Não informado	Não informado
E15	Nurse-Jugglers	Não informado	Não informado

Fonte: Do Autor

5.1 Configuração dos Parâmetros do AGH

O AGH implementado neste trabalho contém um série de parâmetros que devem ser definidos, sendo esta uma tarefa de grande importância, visto que diferentes combinações podem levar o algoritmo a alcançar soluções de boa ou má qualidade. Encontrar uma configuração de parâmetros que atenda de maneira eficiente todas as instâncias do conjunto *Late* é uma tarefa desafiadora e que exige uma grande quantidade de tempo para aplicação de um número exaustivo de testes. Infelizmente a configuração de parâmetros deste trabalho esbarrou na barreira relativa ao tempo, como resultado, a qualidade das soluções se mostraram muito sensíveis ao tamanho e configurações das instâncias. A Tabela 6 apresenta a configuração dos parâmetros utilizados pelo AGH proposto neste trabalho.

A busca local RVND se mostrou eficiente no processo de melhoria das soluções, contudo, houve uma grande dificuldade em encontrar qual o melhor momento para sua execução dentro do AGH. Empiricamente, a melhor maneira encontrada foi utiliza-la nos indivíduos mutantes somente após um certo número de iterações sem melhora. Um grande problema encontrado durante a execução do *solver* foi que para as instâncias com um grande número de enfermeiros (80,100,120) o custo computacional do RVND se tornou

Tabela 6 – Parâmetros do AGH

Parâmetros	**Descrição**	**Valor**
P	tamanho da população	*35*
Pe	tamanho da população elite	*0.20P*
Pm	tamanho da população mutante	*0.20P*
Ng	número de gerações	*20000*
STOP	critério de parada do algoritmo	*tempo limite da instância.*
TPE	critério para troca da população elitista	*a cada 50 iterações sem melhora.*
AVND	aplicação do RVND nos mutantes	*a cada 10 iterações sem melhora.*

demasiado grande, sendo necessário não executar o movimento de troca entre blocos de enfermeiros para que fosse possível encontrar soluções factíveis dentro do conjunto de instâncias mencionados.

5.2 Comparação dos resultados encontrados

Para as instâncias do tipo *Late* foi permitido que as equipes participantes da competição realizassem 10 execuções diferentes para cada uma das 28 instâncias do conjunto, sendo as soluções de melhor qualidade encontradas para cada instância consideradas como resultado final. Para cada execução, o validador disponibilizado pelo INRC-II permitia a passagem, por linha de comando, das sementes responsáveis pela geração de números aleatórios. Os resultados deste trabalho fizeram uso de apenas uma execução e de uma semente fixa para todas as instâncias *late*.

A tabela 7 compara os resultados encontrados pelo AGH com o de todas as equipes participantes da competição. Para uma visão geral dos resultados foi realizado um cálculo de *Gap(%)* relativo entre todas as soluções para a solução de menor custo de cada instância. Sendo E_n a solução de um competidor, C_{best} a melhor solução encontrada para cada instância e n = {1,...,15,AGH} o cálculo de *Gap(%)* é realizado da seguinte forma $100 \cdot ((E_n - C_{best})/C_{best})$. Cada linha da tabela é referente a uma instância do conjunto *Late*, as colunas de E_1 até E_{15} estão ordenadas de acordo com a classificação geral da competição. A coluna AGH é referente ao *Gap(%)* do algoritmo proposto para a melhor solução encontrada em cada instância. Os resultados destacados pela cor vermelha indicam as soluções em que o AGH conseguiu encontrar um resultado melhor do que o reportado por esses competidores. A última linha representa a classificação final da competição incluindo o AGH entre os competidores.

Tabela 7 – Tabela de Resultados do INRC-II

Instância	AGH	E1	E2	E3	E4	E5	E6	E7	E8	E9	E10	E11	E12	E13	E14	E15
n030w4_1_6-2-9-1	30.95%	0.00%	2.01%	10.60%	15.19%	14.61%	18.62%	18.34%	36.68%	34.96%	42.69%	87.97%	54.15%	89.11%	225.50%	464.47%
n030w4_1_6-7-5-3	24.29%	0.00%	0.78%	10.08%	10.34%	8.53%	11.63%	15.25%	24.81%	36.43%	38.50%	69.25%	53.23%	72.09%	229.20%	448.06%
n030w8_1_2-7-0-9-3-6-0-6	64.49%	0.00%	1.31%	28.10%	28.32%	13.73%	34.20%	36.17%	56.21%	72.77%	75.82%	94.99%	79.74%	115.47%	307.63%	823.09%
n030w8_1_6-7-5-3-5-6-2-9	77.37%	0.00%	4.74%	25.26%	36.05%	24.74%	38.95%	38.42%	49.47%	81.05%	82.63%	90.26%	112.89%	169.47%	359.21%	1012.9%
n040w4_0_2-0-6-1	41.74%	0.00%	2.32%	9.86%	8.41%	20.29%	22.90%	31.88%	25.22%	48.12%	59.42%	87.25%	71.88%	135.65%	391.01%	751.0%
n040w4_2_6-1-0-6	34.82%	0.00%	5.24%	13.35%	8.90%	17.02%	22.51%	23.82%	35.08%	41.88%	61.26%	77.23%	72.51%	102.09%	374.61%	657.07%
n040w8_0_0-6-8-9-2-6-6-4	59.10%	0.00%	6.76%	20.93%	21.26%	30.43%	49.76%	45.09%	46.05%	59.10%	92.11%	94.20%	118.68%	142.35%	421.26%	1027.5%
n040w8_2_5-0-4-8-7-1-7-2	51.60%	1.18%	0.00%	22.02%	16.64%	25.55%	44.37%	39.16%	28.40%	59.83%	80.84%	90.76%	103.70%	149.92%	404.54%	1009.2%
n050w4_0_0-4-8-7	74.43%	0.00%	6.89%	20.98%	13.44%	23.93%	33.77%	40.00%	60.98%	73.77%	94.75%	120.33%	111.48%	160.66%	537.38%	1063.6%
n050w4_0_7-2-7-2	81.42%	0.00%	11.15%	26.69%	20.61%	32.09%	44.59%	47.64%	64.53%	84.80%	106.08%	101.01%	113.51%	167.91%	542.57%	969.59%
n050w8_1_1-7-8-5-7-4-1-8	44.51%	0.00%	3.06%	9.80%	22.12%	19.24%	25.90%	29.95%	44.51%	43.62%	52.70%	80.13%	65.11%	77.25%	362.05%	674.10%
n050w8_1_9-7-5-3-8-8-3-1	41.19%	0.00%	2.28%	6.03%	21.28%	21.10%	30.41%	29.77%	46.67%	44.84%	57.17%	78.81%	66.58%	80.18%	339.82%	681.10%
n060w4_1_6-1-1-5	48.59%	0.00%	0.00%	6.18%	6.36%	22.08%	17.67%	21.55%	17.31%	42.93%	47.00%	66.43%	61.66%	74.91%	334.98%	579.51%
n060w4_1_9-6-3-8	49.49%	2.20%	0.00%	7.12%	9.83%	20.00%	24.41%	25.93%	27.97%	39.66%	65.08%	76.10%	61.19%	105.76%	291.86%	591.53%
n060w8_0_6-2-9-9-0-8-1-3	104.58%	0.00%	3.17%	32.75%	30.81%	41.20%	50.88%	66.37%	72.36%	107.39%	137.15%	111.80%	155.11%	194.37%	753.70%	1453.9%
n060w8_2_1-0-3-4-0-3-9-1	91.25%	0.00%	2.50%	21.41%	30.78%	40.78%	53.91%	63.44%	75.47%	100.63%	119.84%	105.16%	125.94%	166.25%	664.69%	1288.4%
n080w4_2_4-3-3-3	50.50%	0.00%	4.03%	9.93%	16.69%	18.85%	29.35%	29.06%	43.45%	53.96%	60.86%	62.01%	57.55%	88.92%	512.81%	675.11%
n080w4_2_6-0-4-8	44.55%	0.00%	3.25%	11.88%	13.01%	16.83%	25.60%	28.15%	50.50%	49.65%	51.91%	66.20%	66.34%	71.43%	461.95%	669.73%
n080w8_1_4-4-9-9-3-6-0-5	72.55%	0.00%	4.02%	20.95%	10.94%	39.01%	41.28%	41.18%	52.94%	72.86%	96.80%	73.37%	96.39%	145.30%	643.03%	1239.8%
n080w8_2_0-4-0-9-1-9-6-2	71.69%	0.00%	6.07%	23.90%	17.63%	32.52%	45.54%	46.52%	42.21%	74.14%	102.25%	82.57%	101.08%	136.14%	621.84%	1202.9%
n100w4_0_1-1-0-8	135.64%	6.23%	0.00%	16.96%	36.33%	62.63%	66.09%	76.47%	93.77%	134.60%	165.74%	149.48%	190.31%	276.82%	947.75%	2234.9%
n100w4_2_0-6-4-6	96.38%	0.00%	1.45%	11.35%	17.87%	40.82%	42.75%	49.28%	57.73%	97.34%	112.80%	95.65%	143.00%	183.82%	657.73%	1516.7%
n100w8_0_0-1-7-8-9-1-5-4	119.06%	3.23%	0.00%	14.70%	33.76%	65.27%	58.80%	70.92%	57.84%	120.03%	173.51%	111.47%	195.32%	271.73%	929.89%	2654.8%
n100w8_1_2-4-7-9-3-9-2-8	140.51%	0.00%	7.02%	32.85%	37.48%	75.60%	78.31%	78.47%	71.29%	134.93%	172.73%	119.14%	207.50%	295.85%	911.64%	2689.3%
n120w4_1_4-6-2-6	98.58%	0.00%	1.82%	3.44%	25.30%	37.04%	34.41%	47.77%	90.28%	100.00%	121.86%	114.37%	112.75%	140.69%	725.91%	1367.0%
n120w4_1_5-6-9-8	93.28%	0.00%	0.20%	13.24%	22.13%	35.77%	38.74%	46.25%	105.53%	101.98%	113.83%	118.77%	111.86%	156.52%	774.31%	1335.6%
n120w8_0_0-9-9-4-5-1-0-3	140.65%	0.00%	13.36%	14.06%	37.41%	72.86%	60.34%	77.64%	120.39%	141.77%	159.77%	130.24%	175.25%	240.79%	1218.4%	2251.3%
n120w8_1_7-2-6-4-5-2-0-2	154.15%	0.00%	15.72%	23.14%	54.59%	83.84%	83.99%	75.84%	154.73%	135.23%	165.50%	179.18%	184.43%	308.15%	1197.4%	2291.4%
Gap Médio	76.23%	0.46%	3.90%	16.70%	22.27%	34.16%	40.35%	44.30%	59.01%	78.15%	96.81%	97.65%	109.61%	154.27%	576.52%	1200.85%
Classificação	9	1	2	3	4	5	6	7	8	10	11	12	13	14	15	16

Pode-se observar que o AGH, produziu resultados melhores para todas as instâncias em relação aos competidores E_{12} a E_{15}. Se for levado em consideração somente as instâncias que contém 30, 40, 60 e 80 enfermeiros o AGH vence o competidor E_{11} e perde em somente uma vez para o competidor E_{10}. É nítida a queda de desempenho do algoritmo proposto para as instâncias com horizonte de planejamento de 8 semanas. Uma provável explicação para esse fato é que as estimativas das restrições S6 e S7 não estão suficientemente bem implementadas, sendo essa deficiência escancarada com o aumento do número de semanas a serem processadas. Outra justificativa para a queda de desempenho seria a utilização do mesmo conjunto de parâmetros para as instâncias com horizonte de planejamento de 4 ou 8 semanas.

Ao observar a Tabela 8 é possível perceber que o *Gap* médio para as melhores soluções não é tão grande até que o número de enfermeiros seja maior que 50. Já quando o número de enfermeiros ultrapassa esse valor, o *Gap* médio praticamente dobra de valor. Consideramos esse comportamento das soluções como um forte indício de que a configurações dos parâmetros utilizados no AGH necessitam de um estudo mais profundo, possibilitando assim uma melhor calibragem dos valores adotados.

Tabela 8 – *Gap* Médio para dois conjuntos de número de enfermeiros

Número de enfermeiros das instâncias	**30,40,50**	**60,80,100,120**
AGH (*Gap* Médio)	48.04%	94.47%

O método de classificação utilizado pelo INRC-II é bastante simples. É feito para cada competidor uma média da classificação em cada uma das instâncias presentes no conjunto *Late* e por fim, os valores da média são ordenados do menor para o maior. O método proposto neste trabalho alcançaria a nona posição, não conseguindo atingir o objetivo de se classificar entre as 7 melhores equipes e assim disputar a etapa final da competição. Contudo, a nossa avaliação final para o resultado é positiva devido ao fato do método ser inovador em relação aos dos outros competidores e dos trabalhos realizados posteriormente a competição. Além do mencionado, o algoritmo é claramente passível de uma série de melhorias que serão discutidas mais a frente. Também não se pode deixar de levar em consideração que os testes computacionais foram realizados com uma única execução e uma única semente, o que sem dúvida alguma deixa o algoritmo em grande desvantagem perante os outros competidores.

6 Conclusão e Trabalhos futuros

O desempenho operacional dos hospitais está diretamente ligado a qualidade dos serviços prestados pela equipe de enfermagem. A partir deste ponto surgiu o Problema de Escalonamento de Enfermagem (PEE), o qual tem como objetivo, dado um conjunto de restrições diferentes, criar escalas de enfermagem que minimize o máximo possível a ocorrência dessas restrições. Este trabalho utilizou a meta-heurística BRKGA junto a um RVND para resolver o PEE proposto pelo INRC-II. Durante a pesquisa bibliográfica desta monografia não foi encontrado nenhum trabalho que utilizasse da meta-heurística BRKGA para um PEE, portanto a proposta deste trabalho é inovadora e espera-se que sirva como porta de entrada para mais pesquisas no contexto mencionado.

O algoritmo proposto ficaria em 9 lugar de um total de 15 participantes inscritos no INRC-II. Apesar do resultado não estar entre as 7 melhores equipes, o que era um dos objetivos deste trabalho, a avaliação final é considerada positiva. O algoritmo mostrou ter um grande potencial para melhora, visto primeiramente que o *Gap(%)* das soluções encontradas para a equipe que se classificou em sétimo lugar não é muito grande. Outro ponto importante é que para cada instância foi possível realizar apenas uma execução com uma única semente, sendo que todos os outros resultados da competições possuem dez execuções com sementes diferentes para obtenção da melhor solução. A configuração de parâmetros utilizados para algoritmo pode claramente ser melhorada, visto a inconsistência na qualidade dos resultados com o crescimento do tamanho das instâncias.

Com a melhoria de somente alguns dos pontos citados acima, a probabilidade da qualidade das soluções alcançarem resultados mais competitivos é muito grande. Possibilitando assim a aplicação do algoritmo para um ambiente hospital real. Entre as possibilidades de trabalhos futuros encontram-se:

- Utilização de uma ferramenta, como o *irace*, para a calibração de parâmetros
- Ajuste dos parâmetros de acordo com o tamanho da instância.
- Implementar estratégias mais robustas para as estimativas das restrições S6 e S7.
- Testes do algoritmo proposto em conformidade com a competição, isto é, executar para cada instância 10 vezes com sementes distintas e armazenar a melhor solução.

Referências

BEAN, J. C. Genetic algorithms and random keys for sequencing and optimization. **ORSA journal on computing**, INFORMS, v. 6, n. 2, p. 154–160, 1994.

BRITO, J. A. de M.; SEMAAN, G. da S.; BRITO, L. R. Resolução do problema dos k-medoids via algoritmo genético de chaves aleatórias viciadas. **Blucher Marine Engineering Proceedings**, v. 1, n. 1, p. 50–61, 2014.

BURKE, E. K. et al. The state of the art of nurse rostering. **Journal of scheduling**, Kluwer Academic Publishers, v. 7, n. 6, p. 441–499, 2004.

CESCHIA, S. et al. Second international nurse rostering competition (inrc-ii)—mathematical models—. **arXiv preprint arXiv:1501.04177**, 2015.

CESCHIA, S. et al. Second international nurse rostering competition (inrc-ii)—problem description and rules—. **arXiv preprint arXiv:1501.04177**, 2015.

CESCHIA, S.; SCHAERF, A. Solving the inrc-ii nurse rostering problem by simulated annealing based on large neighborhoods. In: PATAT. [S.l.], 2018.

CHEANG, B. et al. Nurse rostering problems—-a bibliographic survey. **European Journal of Operational Research**, Elsevier, v. 151, n. 3, p. 447–460, 2003.

GAIDZINSKI, R. R. Dimensionamento de pessoal de enfermagem em instituições hospitalares. 1998.

GOLDBERG, D. E.; HOLLAND, J. H. Genetic algorithms and machine learning. **Machine learning**, Springer, v. 3, n. 2, p. 95–99, 1988.

GOMES, R. A.; TOFFOLO, T. A.; SANTOS, H. G. Variable neighborhood search accelerated column generation for the nurse rostering problem. **Electronic Notes in Discrete Mathematics**, Elsevier, v. 58, p. 31–38, 2017.

GONÇALVES, J. F.; RESENDE, M. G. Biased random-key genetic algorithms for combinatorial optimization. **Journal of Heuristics**, Springer, v. 17, n. 5, p. 487–525, 2011.

HARTOG, S. den. **On the Complexity of Nurse Scheduling Problems**. Dissertação (Mestrado) — Utrecht University, 2016.

HASPESLAGH, S. et al. First international nurse rostering competition 2010 (august 10-13, 2010, belfast, uk). In: **PATAT 2010-Proceedings of the 8th International Conference on the Practice and Theory of Automated Timetabling, Belfast, Northern-Ireland, UK**. [S.l.: s.n.], 2010.

HOLLAND, J. H. Outline for a logical theory of adaptive systems. **Journal of the ACM (JACM)**, ACM, v. 9, n. 3, p. 297–314, 1962.

KHEIRI, A. et al. A sequence-based selection hyper-heuristic: A case study in nurse rostering. In: **11th International conference on the practice and theory of automated timetabling (PATAT'16)**. [S.l.: s.n.], 2016. p. 503–505.

LEGRAIN, A.; OMER, J.; ROSAT, S. An online stochastic algorithm for a dynamic nurse scheduling problem. **European Journal of Operational Research**, Elsevier, 2018.

MATSUMOTO, M.; NISHIMURA, T. Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator. **ACM Transactions on Modeling and Computer Simulation (TOMACS)**, ACM, v. 8, n. 1, p. 3–30, 1998.

MLADENOVIĆ, N.; HANSEN, P. Variable neighborhood search. **Computers & operations research**, Elsevier, v. 24, n. 11, p. 1097–1100, 1997.

MORO, M. A. et al. Meta-heurísticas grasp e brkga aplicadas ao problema da diversidade máxima. [sn], 2017.

POLTOSI, M. R. Elaboração de escalas de trabalho de técnicos de enfermagem com busca tabu e algoritmos genéticos. Universidade do Vale do Rio do Sinos, 2007.

PORTELLA, V. S. Metaheurística late acceptance hill climbing aplicada ao problema de escalonamento de enfermagem. 2017.

POWELL, W. B. Clearing the jungle of stochastic optimization. In: **Bridging data and decisions**. [S.l.]: Informs, 2014. p. 109–137.

RÖMER, M.; MELLOULI, T. Future demand uncertainty in personnel scheduling: Investigating deterministic lookahead policies using optimization and simulation. In: **ECMS**. [S.l.: s.n.], 2016. p. 502–507.

ROSSETTI, A. C.; CARQUI, L. M. Implantação de sistema informatizado para planejamento, gerenciamento e otimização das escalas de enfermagem. **Acta Paulista de Enfermagem**, SciELO Brasil, v. 22, n. 1, 2009.

SHANNO, D. F. Combinatorial optimization for undergraduates (lr foulds). **SIAM Review**, Society for Industrial and Applied Mathematics, v. 27, n. 4, p. 606, 1985.

SOUZA, G. P. S. et al. A problemática da elaboração da escala mensal de enfermagem. **Acta paul enferm**, SciELO Brasil, v. 24, n. 1, p. 137–41, 2011.

SOUZA, M. et al. A hybrid heuristic algorithm for the open-pit-mining operational planning problem. **European Journal of Operational Research**, Elsevier, v. 207, n. 2, p. 1041–1051, 2010.

SOUZA, M. J. F. Inteligência computacional para otimizaçao. **Notas de aula, Departamento de Computaçao, Universidade Federal de Ouro Preto, disponıvel em http://www. decom. ufop. br/prof/marcone/InteligenciaComputacional/InteligenciaComputacional. pdf**, 2008.

SPEARS, W.; JONG, K. D. On the virtues of parametrized uniform crossover. 01 1991.

SPERANDIO, F. de R. Large scale elective surgery scheduling under uncertainty. 2015.

ZÄPFEL, G.; BRAUNE, R.; BÖGL, M. **Metaheuristic search concepts: A tutorial with applications to production and logistics**. [S.l.]: Springer Science & Business Media, 2010.