Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Estudo Comparativo de Meta-Heurísticas: Otimização do Modelo de Alocação na Agência Nacional de Aviação Civil

Hélio Santana da Silva Júnior

Monografia apresentada como requisito parcial
para conclusão do Bacharelado em Ciência da Computação

Orientador
Prof. Dr. Marcelo Antonio Marotta

Brasília
2021

Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Estudo Comparativo de Meta-Heurísticas: Otimização do Modelo de Alocação na Agência Nacional de Aviação Civil

Hélio Santana da Silva Júnior

Monografia apresentada como requisito parcial
para conclusão do Bacharelado em Ciência da Computação

Prof. Dr. Marcelo Antonio Marotta (Orientador)
CIC/UnB

Prof. Dr. Luís Paulo Faina Garcia
CIC/UnB

Prof. Dr. Marcos Fagundes Caetano
CIC/UnB

Prof. Dr. Marcelo Grandi Mandelli
Coordenador do Bacharelado em Ciência da Computação

Brasília, 12 de Maio de 2021

Dedicatória

Dedico esse trabalho à minha família e amigos.

Agradecimentos

Ao meu orientador Professor Marcelo Antonio Marotta , agradeço pela dedicação, profissionalismo e boa vontade, além das palavras de apoio e motivação.

Agradeço aos Professores Silvia Araujo dos Reis e Victor Rafael Rezende Celestino pela orientação e coordenação do projeto Safety Oversight na Agência Nacional de Aviação Civil, bem como aos meus colegas e amigos Beatriz Simões, Camila Mayerhofer, Gabriel Guth, Henrique Daminelli, Hugo Frota, Ingrid Silva, Olivia Ziller e Thiago Dias, cujo trabalho auxiliou a produção dessa monografia.

À Agência Nacional de Aviação Civil Agência Nacional de Aviação Civil (ANAC) agradeço pelo apoio e suporte para a realização desse estudo.

À minha família, por todo o amor, apoio e estrutura proporcionados para que essa jornada fosse mais agradável e possível.

Agradeço a Universidade de Brasília pelas oportunidades e incríveis experiencias vivenciadas durante todos esses anos de estudo.

O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES), por meio do Acesso ao Portal de Periódicos.

Resumo

Órgãos públicos de caráter fiscalizatório, como agências reguladoras, tem a necessidade de atender uma grande demanda de missões com um número limitado de recursos humanos. Esse tipo de problema pode ser categorizado como um problema de *High School Timetibling Problem*(HSTP) um sub-problema da classe de problemas de alocação e agendamento. É possível observar na literatura um grande número de trabalhos que visam solucionar os problemas de otimização, destacando-se as soluções que envolem as meta-heurísticas. Além do uso desses algoritmos para a resolução de problemas de agendamento, modelos de hibridização de meta-heurísticas têm se mostrado promissores na maximização e minimização de funções objetivo em problemas de agendamento, obtendo resultados sub-ótimos muito próximos aos resultados ótimos. Nesse contexto, o trabalho presente tem como objetivo apresentar e comparar três meta-heurísticas, Busca Tabu, Algoritmo Genético e *Simulated Annelaning* e dois modelos de hibridização, multi-estágio e sequencial, utilizando como caso de uso, o modelo de alocação das missões da Agência Nacional de Aviação Civil (ANAC). Além disso, foi desenvolvido uma plataforma para uso de não especialistas como aplicação de rede, utilizando a arquitetura REST, onde conectam-se os dados das missões com o algoritmo de otimização que obteve o melhor desempenho, com uma interface amigável e simplificada ao usuário final. Resultados comparativos entre os algoritmos apresentados mostraram que o algoritmo híbrido multi-estágio combinando a abordagem de Busca Tabu e o Algoritmo Genético (HMS) obteve melhor desempenho na maioria dos testes. Já, para os métodos sem hibridização, o algoritmo genético destacou-se com os melhores resultados dentre as demais técnicas comparadas.

Palavras-chave: Meta-heurísticas Híbridas, Algoritmos Evolucionários, Busca Tabu, HighSchool Timetibling Problem

Abstract

Public organs of supervisory nature, such as regulatory agencies, have the need to attend a high demand for inspection missions with a limited number of human resources. This type of problem can be categorized as a *High School Timetibling Problem* (HSTP), a sub-problem of the class of allocation and scheduling problems. It is possible to observe in the literature a large number of works that aim to solve the optimization problems, highlighting the solutions that involve the meta-heuristics. In addition to the use of these algorithms to solve scheduling problems, meta-heuristic hybridization models are showing promising in the maximizing and minimizing objective functions in scheduling problems, obtaining sub-optimal results very close to the optimal results. In this context, the present work aims to present and compare three meta-heuristics: tabu search, genetic algorithm and simulated annelaning and two hybridization models: multi-stage and sequential, using the mission allocation model from the National Civil Aviation Agency of Brazil (ANAC), as well as the development of a web platform, using the REST architecture, where it connects the data of the missions and the inspectors to the optimization algorithm that obtained the best performance, with a friendly and simplified interface to the end user. Results of the comparison between the presented algorithms showed that the hybrid multi-stage algorithm between the tabu search and the genetic algorithm (HMS) obtained better performance in most of the test cases. The genetic algorithm also obtained reasonable results, showing itself to be indistinguishable from the HSM algorithm in some cases.

Keywords: Hybrid Metaheuristics, Evolutionary algorithms, Tabu Search, HighSchool Timetibling Problem

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas e Siglas

ANAC Agência Nacional de Aviação Civil.

GEN Algoritmo Genético.

HMS Híbrido multi-estágio.

HSEQ Híbrido Sequencial.

HSTP High School Timetabling Problem.

SA Simulated Annealing.

TABU Busca Tabu.

UCTP University Course Timetabling Problem.

Capítulo 1

Introdução

É comum observar em órgãos públicos de caráter fiscalizatório, como as agências reguladoras federais, a necessidade de atender uma grande demanda de fiscalização com um número limitado de servidores. Órgãos como a ANAC possuem missões de fiscalização em todo o território nacional e uma quantidade reduzida de postos onde seus servidores estão alocados. Considerando as limitações de recurso humanos e a crescente demanda de missões, surge a necessidade de se otimizar tais recursos. Nesse contexto, técnicas de maximização e minimização de funções, heurísticas em grafos e soluções em logística são exemplos de ferramentas que podem auxiliar nesse tipo de problema [1].

O problema de otimização de recursos da ANAC, onde existe um número determinado de missões de fiscalização e uma quantidade reduzida de pessoas aptas a realizar essas atividades, pode ser encaixado como um problema de agendamento [2]. Problemas de agendamento são muito comuns de serem encontrados na sociedade de um modo geral. O modelo base de problema de agendamento é o agendamento de salas de aulas em universidades, ou *High School Timetibling Problem* (HSTP) [3], onde disciplinas e professores são alocados em salas de aulas e horários determinados. O HSTP é um problema NP-difícil onde a medida que as variáveis vão crescendo em número é necessário mais combinações e cálculos para encontrar a melhor solução, se encaixando como problemas combinatórios. Softwares proprietários de solução de problemas de otimização como o LINGO e AIMMS, utilizam algoritmos de solução ótima, como Simplex e *Branch Bound*, para resolver essa classe de problemas.

Embora os softwares proprietários consigam resolver os problemas de agendamento de forma ótima e, por consequência, o problema de otimização de recursos da ANAC [2], estes tendem a ser pouco eficiente em termos de custo computacional e sua compreensão limitada aos especialistas da área. Em outras palavras, o grande número de combinações entre missões e recursos humanos necessária para solucionar o problema de otimização de recursos da ANAC gera um custo computacional muito alto, implicando em um tempo

de execução inviável para utilização cotidiana tendo sua compreensão e uso limitados aos técnicos da ANAC. Para contornar o alto custo computacional, diversas soluções heurísticas e meta-heurísticas podem ser utilizadas para resolver o problema de agendamento em um tempo reduzido explorando soluções semi-ótimas aceitáveis [1].

As meta-heurísticas são técnicas de otimização de classe probabilística. Essas técnicas combinam funções objetivo e heurísticas para resolver uma série de classes de problemas de otimização, sendo eficiente, tanto em tempo de execução quanto em alcançar uma solução sub-ótima. Essas técnicas podem ser personalizadas, ou seja, podem ser implementada em diversas linguagens de programação utilizando estruturas de dados a depender do contexto [4]. Entre as técnicas meta-heurísticas disponíveis, destacam-se as soluções utilizadas para resolver o problema de HSTP, como por exemplo a utilização do Algoritmo Genético (GEN) [5], *Simulated Annealing* (SA)[6] e até mesmo soluções hibridas misturando a Busca Tabu (TABU) com algoritmos evolucionários e algoritmos de programação inteira [7].

Ainda que as meta-heurísticas possam ser utilizadas para resolver o problema de HSTP, as mesmas precisam ser adequadas e comparadas em relação aos seus desempenhos para solucionar o problema específico de otimização de recursos da ANAC. Além disso, o uso de meta-heurísticas também requisita um alto e restritivo conhecimento técnico, *i.e.,* essas técnicas normalmente são personalizadas em nível de código cuja compreensão é restrita aos especialistas em pesquisa operacional e seus programadores, tornando sua utilização inviável para não especialistas, *e.g.,* corpo técnico administrativos e gestores, sendo estes os mais interessados. Dessa forma, existe a necessidade de uma solução que consiga combinar técnicas meta-heurísticas sem que o seu uso seja restritivo aos especialistas da área.

Neste documento propõe-se apresentar um sistema remoto de otimização do agendamento de missões da ANAC, alocando de forma a minimizar o gasto em descolamento aéreo o número reduzido de inspetores a quantidade total de missões da Agência. O sistema desenvolvido utilizando a técnica de engenharia de software REST conectando a base de dados da ANAC com o usuário final de modo que a partir da meta-heurística escolhida a escala de missões é gerada automaticamente. Para a escolha das meta-heurísticas presentes no sistema web, foi realizado também uma comparação entre cinco algoritmos e o modelo ótimo desenvolvido na ferramenta LINGO, com dados reais da agência. Deste modo são utilizadas duas métricas de comparação: perceber que o algoritmo X, por exemplo, é melhor que o algoritmo Y e diferenciável com Z% de confiança e o algoritmo X é melhor que o algoritmo Y em média, mas com intervalos de confiança com Z% não foi possível diferenciá-los.

O seguinte trabalho foi desenvolvido em seis capítulos. No primeiro capítulo é apresentado toda a fundamentação teórica dos algoritmos técnicas utilizados neste trabalho. Já no segundo capítulo são apresentados os trabalhos relacionados que inspiraram toda a metodologia utilizada nesse trabalho. No terceiro capítulo é apresentado a modelagem do problema de agendamento utilizada pelos algoritmos de otimização. A proposta e os módulos que a compõem bem como a definição das meta-heurísticas e aplicação web são apresentados no quarto capítulo. Os resultados obtidos a partir da comparação das meta-heurísticas são apresentadas no quinto capítulo. Por fim, no sexto capítulo, as conclusões e considerações finais são apresentadas.

Capítulo 2

Fundamentação Teórica

Este capítulo apresenta os principais conceitos que amparam o desenvolvimento de uma solução web de um modelo de otimização baseado em HSTP utilizando meta-heurísticas. As bases teóricas sobre as classes de problemas NP-Difíceis em modelos de agendamentos são conceituadas, bem como o funcionamento de algoritmos de meta-heurísticas. Por fim, são apresentadas as ferramentas utilizadas no desenvolvimento do sistema web de otimização.

2.1 Modelos de Agendamento

Segundo [8] um problema de agendamento genérico pode ser definido utilizando 4 parâmetros: T, sendo um conjunto finito de intervalos de tempo; R, conjunto finito de recursos; M, conjunto finito de encontros; C, conjunto finito de restrições. O objetivo é combinar intervalos de tempos e recursos com encontros, de forma que as restrições sejam satisfeitas. Problemas desse tipo podem ser dividas em várias categorias, podemos destacar *University Course Timetabling Problems* (UCTP), HSTP e *Examination Timetabling Problems* [9]. Como demonstrado em [2] o modelo utilizado neste documento se aproxima melhor do HSTP, onde os intervalos de tempo T são vistos como sessões, os recursos R são os alunos, professores, disciplinas e as salas de aula e os encontros são as combinações dos recursos e sessões, respeitando as restrições impostas [3].

2.1.1 Modelo Desenvolvido no LINGO

O modelo desenvolvido na ANAC é um modelo de programação linear binária mista. Seu objetivo é alocar com o menor custo possível os inspetores da agência nas missões de fiscalização. Cada missão possui uma atividade, um destino e uma duração em dias para ser realizada. Cada inspetor está previamente alocado em uma unidade da federação e

possui além de uma quantidade de dias disponíveis para realizar as missões uma lista de atividades que é capaz de realizar.Todas as missões devem ser atendidas pelos inspetores.

A relação entre origem e destino é chamada de arco origem-destino. Este arco é formado por uma origem sendo uma unidade da federação e um destino sendo um aeroporto. Também é atrelado a esse arco os valores de passagem área média e o tempo de viagem discretizado entre meio e um dia. O calculo do valor médio de passagem área leva em conta o menor valor médio de passagem área em uma cidade com mais de um aeroporto, ou seja, em uma cidade como São Paulo que possui três aeroportos comerciais, Congonhas, Guarulhos e Viracopos, é considerado apenas o menor valor médio de passagem dentre essas três origens, em relação a um destino desejado.

O tempo de cada missão varia de acordo com o inspetor escolhido para realizá-la, ou seja, é o somatório de duas vezes o tempo da viagem entre a origem do inspetor escolhido e o destino da missão somado ao tempo de realização da missão. Quando o inspetor é escolhido para realizar determinada missão, sua disponibilidade, em dias, é decrescida desse somatório de tempo de viagem. Cada missão é realizada por apenas um inspetor, porém caso a disponibilidade desse inspetor ainda for maior que zero, ele poderá realizar outras missões caso seja apto do ponto de vista da atividade da missão. A função objetivo então visa minimizar o custo total quando todas as missões tiverem inspetores alocados para realizar elas.

A saída do modelo gera uma lista de missões com inspetores, custo de viagem e tempo gasto para realização da missão atrelado. Quando um servidor realiza uma missão onde sua cidade de origem é a mesma que a de destino, o custo da missão é zero, pois não existe deslocamento aéreo e o tempo da missão será apenas o tempo de realização da atividade. Um servidor também não pode fazer reaproveitamento de viagem, ou seja, na mesma viagem realizar duas missões no destino, caso exista. Cada missão é realizada em uma viagem única.

2.2 Meta-heurísticas

Em problemas NP-Difíceis, de caráter combinatório, surge o desafio de achar a melhor solução, ou a maximização da solução e nesse contexto surgem técnicas que aliam a possibilidade de encontrar soluções próximas de um máximo (ou mínimo) global com estados de paradas bem definidos. Meta-heurísticas são estratégias que guiam o processo de busca, unindo simples técnicas de busca local com complexos processos de aprendizagem, são estratégias de alto nível aplicadas na exploração de espaço de busca [10].

A classe desses algoritmos incluem os algoritmos evolucionários, destacando o Algoritmo Genético (GEN), algoritmos de busca local e algoritmos baseados em processos

físicos, como por exemplo o Simulated Annealing (SA). É possível classificar também como algoritmos baseados em trajetória e populações [10]. Além dos algoritmos citados acima, um método promissor de construção de meta-heurísticas é a hibridização, ou seja, a junção de dois ou mais tipos de meta-heurísticas formando um algoritmo mais robusto. Dentre as diversas formas de hibridização, destacam-se a forma sequencial e multi-estágio [11].

Na abordagem multi-estágio a busca pela solução é iniciada por um otimizador global, podendo ser um algoritmo evolucionário por exemplo, em seguida concluído com uma busca local. Segundo [12], algoritmos evolucionários possuem melhor desempenho nos estágios iniciais da busca, enquanto algoritmos de busca locais possuem desempenho melhor nos estágios finais. No método sequencial ambos algoritmos, independendo da sua classe, são executados alternativamente, possivelmente com mesmo número de iterações, chegando por fim na convergência esperada [12].

Além das abordagens de hibridização citadas acima é possível encontrar uma solução paralela, onde ambos os algoritmos são executados juntos sobre a população. Existe também a forma integrada de hibridização, onde um algoritmo tem um peso maior que o outro. Nessa abordagem a população pode ser integralmente utilizada ou apenas uma parte fracionada, para que uma porcentagem da população seja executada com um algoritmo de busca local, por exemplo, enquanto o restante da população seja executada normalmente com o algoritmo base. A ideia é acelerar a otimização em uma parte da população e por fim utilizar o algoritmo base para encontrar a solução sub-ótima [12].

2.2.1 Busca Tabu

A Busca Tabu (TABU) é um algoritmo de busca local clássico, onde tem como principal característica a estrategia de penalização [13]. Algumas soluções são colocadas em uma lista, chamada lista tabu, de modo que quando o algorítimo encontra alguma solução que já esteja na lista, essa solução é ignorada. O objetivo dessa estratégia é evitar ciclos, ou seja, desencorajar mínimos (ou máximos) locais. O método de busca desse algoritmo se dá pelos vizinhos imediatos.

O Algoritmo 21 apresenta um pseudo-código genérico da Busca Tabu. Primeiramente é gerado uma solução inicial *S0* que será escolhida como melhor solução e também melhor candidata, além de ser inserida na lista tabu. O Algoritmo então entrará em um laço até que uma condição de parada seja alcançada. Dentro do laço, o primeiro passo tomado é a geração de vizinhos a partir da melhor solução atual e um desses vizinhos será tomado arbitrariamente como o melhor candidato. Cada vizinho será testado entre si, utilizando a função *fitness* desenvolvida, para determinar de fato o melhor candidato. Em seguida o melhor candidato é comparado a melhor solução atual para ser atualizada, caso necessário.

Algorithm 1: Busca Tabu

```
   Result: Resultado sub-ótimo
 1 sBest ← s0;
 2 bestCandidate ← s0;
 3 tabuList ← [];
 4 tabuList.push(s0);
 5 while not stoppingCondition() do
 6 |   sNeighborhood ← getNeighbors(bestCandidate);
 7 |   bestCandidate ← sNeighborhood[0];
 8 |   for sCandidate in sNeighborhood do
 9 |   |   if ((not tabuList.contains(sCandidate)) and (fitness(sCandidate) >
   |   |    fitness(bestCandidate))) then
10 |   |   |   bestCandidate ← sCandidate;
11 |   |   end
12 |   end
13 |   if fitness(bestCandidate) > fitness(sBest) then
14 |   |   sBest ← bestCandidate;
15 |   end
16 |   tabuList.push(bestCandidate);
17 |   if tabuList.size > maxTabuSize then
18 |   |   tabuList.removeFirst();
19 |   end
20 end
21 return sBest;
```

Algorithm 2: Algoritmo Genético

Result: Resultado sub-ótimo

```
generateInitialPopulation();
while (i < maxIteration) and (bestFitness < maxFitness) do
    fitnessCalculationForEachchromosome();
    selection();
    crossover();
    mutation();
end
return bestChromosome;
```

A lista tabu é atualizada com o melhor candidato. Note que a lista tabu possui um tamanho finito determinado, então quando seu tamanho é preenchido, deve-se excluir a solução mais antiga.

Note que as funções de geração da primeira solução, geração de vizinhos e *fitness* são geradas a partir da modelagem do contexto especifico, bem como as estruturas de dados escolhidas para a implementação da modelagem do problema. Ao final do algoritmo a melhor solução atual é a solução mais próxima do mínimo (ou máximo) global do conjunto de soluções.

2.2.2 Algoritmo Genético

O Algoritmo Genético (GEN) pertence a classe dos algoritmos evolucionários. Sua implementação é inspirada no processo de seleção natural biológica. Cada etapa do algoritmo é uma abstração de um processo biológico, assim como suas estruturas, como por exemplo os cromossomos, genes e populações [14]. As etapas do algoritmo são: geração da população inicial, *fitness*, seleção, *crossover* e mutação.

A primeira abstração que deve ser feita quando se utiliza desse algoritmo é a implementação de um cromossomo, ou seja, cada solução deve estar na forma de uma lista binária finita, onde cada posição da lista representa um gene. O conjunto de soluções representa a população, formada por cada cromossomo. O Algoritmo 8 apresenta o pseudo-código do GEN. Podemos observar que inicialmente uma população inicial deve ser iniciada, ou seja, um conjunto finito de soluções aleatórias na forma de cromossomos.

A partir do conjunto inicial, com seu *fitness* calculado o laço de execução é iniciado. Enquanto a população não convergir para o mínimo (ou máximo) global é realizado o processo de seleção dos genes. Neste processo os melhores cromossomos são escolhidos e os piores descartados. Os melhores pais são cruzados de modo a obter o conjunto de soluções filhas. Mutações podem ocorrer com uma taxa pré determinada. Todo o processo é repetido até as condições de parada sejam alcançadas.

Algorithm 3: Simulated Annealing

```
   Result: Resultado sub-ótimo
 1 initialSolution ← generateInitialSolution();
 2 while not stoppingCondition() do
 3 |   fitnessInitialSolution ← fitness(initialSolution);
 4 |   NeighborsSolutions ← generateNeighbors(initialSolution);
 5 |   for candidate in NeighborsSolutions do
 6 |   |   fitnessCandidate ← fitness(candidate);
 7 |   |   if fitnessCandidate > fitnesInitialSolution then
 8 |   |   |   initialSolution ← candidate;
 9 |   |   end
10 |   |   if isConstant(fitnesInitialSolution) then
11 |   |   |   shakeWithProbability(initialSolution);
12 |   |   end
13 |   end
14 end
15 best ← initialSolution;
16 return best;
```

2.2.3 Simulated Annealing

O Algoritmo 16 Simulated Annealing (SA), também é da classe dos algoritmos de busca local, porém probabilístico, sua motivação vem da termodinâmica, na capacidade térmica dos materiais, onde um material é aquecido e em seguida o mesmo é resfriado gradualmente até uma temperatura muito baixa, com o intuito de observar o estado fundamental do sólido [15]. Inicialmente é gerado uma solução inicial válida, que atende todas as restrições. No laço principal são realizados operações de corte na solução, ou seja, cada pedaço é deslocado de modo a encontrar novos vizinhos que terão seu *fitness* comparado. A melhor solução válida dentre essas é escolhida para que uma nova serie de cortes seja realizado. Note que em cada iteração o tamanho do corte é diminuído pela metade. Quando não for possível mais realizar cortes é reiniciado o tamanho do corte para metade do tamanho da solução. O corte é exemplificado como a geração de novos vizinhos, apresentado na linha 4 do Algoritmo 16. Caso a solução atual se mantenha constante, é aplicado uma função de agitação para buscar um outro vizinho que pode sair de um mínimo local. Ao fim do algoritmo é salvo a melhor solução encontrada.

2.2.4 Métodos de Hibridização

Existem diversos métodos de hibridização de algoritmos de otimização destacando-se a hibridização colaborativa e integrativa [11]. Nessa sessão iremos apresentar duas técnicas, apresentadas na figura 2.1 colaborativas que foram utilizadas nesse trabalho: multi-estágio

e sequencial. A hibridização multi-estágio funciona a partir da mistura de duas meta-heurísticas, preferencialmente uma que performa melhor globalmente sobre o espaço de busca e outra especializada em busca local. O primeiro algoritmo gera uma população inicial e realiza um corte mais amplo nessa população, após esse processo o algoritmo de busca local penetra nessa população selecionada de modo a encontrar a melhor solução. No método sequencial, cada algoritmo atua alternativamente um número de iterações iguais até que os critérios de parada sejam alcançados.

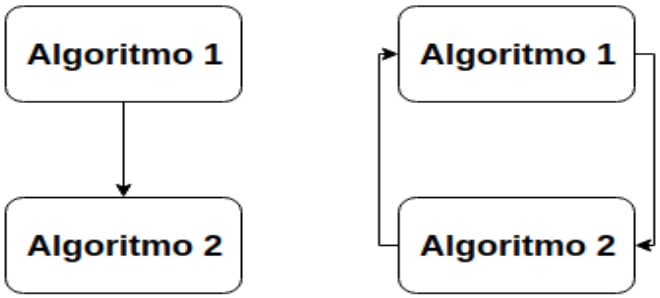

Figura 2.1: Hibridização multi-estágio e sequencial;

2.3 Sistema web de alocação

O sistema web que une o modelo de alocação utilizando a meta-heurística de melhor desempenho e o banco de dados de entradas foi desenvolvido utilizando a arquitetura de software Representational State Transfer (REST). Os recursos do *web service* são requisitados sem estado, tornando o processo mais dinâmico.

A arquitetura REST utiliza de várias convenções para extrair o melhor do protocolo HTTP para prover um CRUD (Create, Read, Update, and Delete). O CRUD é mapeado utilizando as ações: POST, GET, PUT e DELETE. Essa arquitetura aliada ao gerenciamento de URL permite a criação de uma API para lidar com as requisições do usuário e servidor. Cada ação é mapeada em uma URL diferente, permitindo uma assincronicidade.

Este capitulo apresentou toda a fundamentação necessária para o entendimento do problema de alocação abordado por esse trabalho bem como as meta-heurísticas e técnicas de hibridização que serão utilizadas para resolver esse problema. O próximo capitulo irá apresentar diversos trabalhos que utilizaram essas técnicas para as resoluções de problemas de agendamento similares.

Capítulo 3

Trabalhos Relacionados

Diversos projetos de pesquisa já utilizaram e compararam o comportamento de meta-heurísticas e suas derivações em problemas de alocação e agendamento. Este capítulo apresenta uma serie de trabalhos da literatura que apresentam soluções do problema de agendamento utilizando as técnicas usadas neste documento, além de um trabalho de alocação realizado na ANAC similar a proposta apresentada por este documento.

3.1 Caracterização de problemas de agendamento

O trabalho apresentado por Babaei *et al* [1] é um *survey* que apresenta diversos métodos para a resolução do problema UCTP. Dentre os métodos apresentados desacatam-se: soluções utilizando pesquisa operacional, meta-heurísticas e métodos inteligentes e sistemas multi-agentes distribuídos. Além de apresentar os diferentes métodos, o trabalho investiga a escalabilidade de algumas soluções e apresenta alguns *data sets* para testar os diferentes algoritmos.

O primeiro grupo de métodos apresentados são soluções envolvendo conceitos de pesquisa operacional como por exemplo a teoria de coloração de grafos, programação linear inteira, que gera sempre um resultado ótimo em problemas desse tipo, além do método de programação de satisfação de restrições. O segundo grupo de métodos consiste nas meta-heurísticas populacionais, como por exemplo o algoritmo genético e colônia de formigas, e buscas locais como o TABU e SA. O terceiro grupo inclui métodos multi-objetivos e multi-critérios e o quarto grupo engloba métodos de inteligencia hibrida, logica fuzzy e algoritmos de clusterização.

Dos resultados os autores destacam que os métodos utilizando técnicas baseadas de pesquisa operacional não possuem uma boa eficiência, apesar de serem fáceis de implementação. Contudo os métodos utilizando algoritmos exploratórios e técnicas de inteligencia artificial performam de maneira mais eficiente embora esses métodos não encontrem as

soluções ótimas. A utilização de métodos exploratórios com métodos multi-critério e multi-agente obtém ganho no ponto de vista da escalabilidade e consequentemente um ganho na procura da solução sub-ótima.

Em resumo o trabalho de Babei *et al* [1] apresenta diversas formas de resolver o problema de agendamento e cabe ao leitor escolher a melhor solução, tendo em vista que existem soluções que são mais flexíveis, mais precisas e outras que exigem menos custo computacional. A escolha dessas soluções está atrelada ao tipo de restrições duras e flexíveis que o modelo possui e como a saída do modelo depende dessas restrições. Neste projeto será testado também a eficacia e eficiência de algoritmos sub-ótimos em relação à técnicas ótimas, buscando uma alternativa menos custosa computacionalmente mas razoável do ponto de vista da otimização.

3.2 Soluções utilizando meta-heurísticas em problemas de agendamento

No trabalho de Al-Jarrah *et al* [16] é apresentado um sistema de alocação de cursos e salas de aula utilizando o GEN. O problema pode ser enquadrado no caso UCTP. O sistema depende dos dados enviados pelo departamento e em seguida é realizado a alocação. O algoritmo foi testado em sete benchmarks e obteve resultado satisfatório em todos os casos.

Nesse trabalho o GEN implementado não possui muitas alterações em relação a sua forma clássica. A diferença se dá, além das estruturas de dados, na geração de uma população inicial aleatória. Essa população é afinada a partir das restrições duras e ai sim iniciada no algoritmo. O gene escolhido para a mutação e crossover também é aleatório.

O algoritmo foi testado utilizando sete datasets da International Timetabling Competition (ITC-2007) e comparado com outros três autores que utilizaram dessa mesma base. Apenas em duas bases o GEN proposto tem desempenho pior, mas na maioria possui desempenho melhor. O trabalho realizado neste documento visa utilizar o GEN em problemas de agendamento semelhantes ao proposto por Al-Jarrah *et al* [16], porém com melhorias como por exemplo a hibridização.

3.3 Soluções híbridas em problemas de agendamento

O trabalho de Ting *et al* [11] mostra diversos métodos de hibridização de meta-heurísticas. A motivação da busca entre algoritmos híbridos parte do problema de convergência prema-

tura, onde a solução converge rapidamente para um máximo ou mínimo global tornando a solução final distante de uma solução ótima e a convergência lenta, onde o algoritmo demora a chegar numa solução razoável, podendo precisar de um poder computacional muito maior que o comum. Deste modo hibridização de algoritmos pode mitigar esses problema e aumentar significativamente o desempenho de meta-heurísticas.

A classe dos algoritmos híbridos pode ser dividida em duas categorias: proposito único ou multi proposito. Na categoria dos algoritmos de proposito único, um algoritmo pode usar um sub-algoritmo em diversas etapas da busca, de modo a encontrar a melhor solução. Em contra partida algoritmos de multi proposito utiliza sub-algoritmos para ajustar alguns parâmetros, como a taxa de mutação em algoritmos genéticos.

Podemos também dividir os algoritmos híbridos de outra forma: colaborativos e integrativos. Dentro da categoria dos algoritmos colaborativos temos o método multi-estágio (HMS), sequencial (HSEQ) e paralelo. No método multi-estágio um algoritmo inicia sua busca no universo de soluções e após um número determinado de passos o segundo algoritmo continua a busca a partir das soluções obtidas pelo algoritmo anterior. No método sequencial os algoritmos fazem a busca de maneira alternada e possivelmente com o mesmo número de passos. No método paralelo ambos algoritmos realizam a busca sobre o mesmo espaço de busca compartilhado.

Na categoria dos algoritmos integrativos existem dois métodos: manipulação total e parcial. Na manipulação total a população inteira é manipulada a cada iteração, utilizando o sub-algoritmo como uma sub-rotina, enquanto na manipulação parcial uma parte da população é acelerada utilizando algum algoritmo de busca local. Algumas desvantagens que pode ser encontrada na hibridização de algoritmos é a complexidade da implementação e da analise, além do aumento do tempo computacional. O trabalho finaliza sugerindo que algoritmos mais simples e melhor estruturados podem ser a melhor saída para a hibridização. Tomando esse estudo como base, o projeto deste documento irá utilizar as propostas de hibridização por multi-estágio e sequencial com algoritmos evolucionários e de busca local.

No trabalho de Feng *et al* [7] é utilizado uma extensão do problema UCTP, incluindo restrições de consecutividade e periodicidade em aulas multi-sessões. Foi criado então um modelo de programação mixa linear e inteira e em seguida em um *three-dimensional container packing problem* (3DCPP). Para a resolução do 3DCPP foi desenvolvido um algoritmo genético híbrido e comparado com o algoritmo de busca tabu clássico, onde o algoritmo híbrido obteve melhor desempenho.

A diferença no modelo UCTP apresentada é a possibilidade de ter aulas seguidas e aulas da mesma disciplina em diferentes dias toda semana, ou seja, uma periodicidade. Os autores formalizaram o problema utilizando programação linear e inteira e em seguida

converteram para um problema 3DCPP onde foi considerado que cada aula é compactada em um *container* de altura D, profundidade R e largura P sobre as restrições duras e flexíveis. O espaço de busca foi criado utilizando a lógica *layer-based bottom deepest left with fill* (LBDLF) representado em um gráfico de eixos: salas, período e dias. É utilizado também a técnica de troca de vizinhos, onde para cada item que compartilha os mesmos dias, salas ou períodos podem ser trocados sem desrespeitar as restrições.

A hibridização do GEN ocorre no afinamento dos cromossomos, onde é utilizado um algoritmo de busca local. Essa técnica de hibridização é categorizada como sequencial por Ting *et al* [11]. O algoritmo hibrido foi comparado a uma Busca Tabu com cargas de trabalho pequenas, médias e grandes. A diferença na função objetivo entre os algoritmos é pequena, com ganho do algoritmo híbrido, porém a solução híbrida tem um tempo de processamento consideravelmente maior.

Em resumo o trabalho proposto por Feng *et al* [7] apresenta uma eficiência do algoritmo híbrido em relação a uma meta-heurística clássica no problema de UCTP. Essa eficiência pode se dar pela forma em que as restrições e variáveis são tratadas, utilizando a estratégia LBDLF em um 3DCPP. Bem como o estudo realizado por Feng *et al* [7], este documento irá testar a eficiência de algoritmos híbridos baseados em problemas de programação inteira mista em HSTP.

No trabalho de Salman *et al* [12] é apresentado uma comparação entre o GEN, SA e uma hibridização entre esses dois algoritmos no problema UCTP. Foi utilizado um data set do *Royal Institute of Technology in Stockholm*. Dentre os algoritmos, o SA teve um desempenho melhor no quesito tempo de execução. Entretanto o algoritmo híbrido obteve uma solução mais razoável a medida que a base de dados escalava.

A construção dos GEN e SA seguem a implementação clássica. Segundo Dahl *et al* [6] algoritmos evolucionários possuem melhor desempenho nos estágio iniciais enquanto algoritmos de busca local desempenham melhor nos estágios finais. Desta forma a hibridização segue a estrategia multi-estágio, apresentada por Ting *et al* [11]. O GEN incia a busca na população até uma quantidade determinada de ciclos e a partir do resultado parcial o SA termina a busca na população refinada.

O GEN puro possui um desempenho em relação ao tempo de execução pior que os outros algoritmos, isso pode se dar pela forma com que os dados são tratados. O SA performa mais rápido que os outros algoritmos. O algoritmo híbrido possui melhor desempenho é achar a solução sub-ótima mas carrega as desvantagens de tempo de execução do GEN.

Em resumo o trabalho acima compara algoritmos clássico com poucas modificações na sua implementação. A escolha da estrategia de hibridização está diretamente relacionada com o desempenho nos estágios iniciais e finais das classes de meta-heurísticas escolhidas.

A estratégia multi-estágio funciona muito bem com a união de algoritmos evolucionários e busca local. A escolha assertiva desses algoritmos e da estrategia de representação de dados pode acarretar numa melhora significativa do desempenho. Esse documento também irá testar a estratégia de hibridização multi-estágio em problemas de agendamento semelhantes, buscando saber o desempenho em relação a outros algoritmos de otimização sub-ótimos.

3.4 Alocação na Agência Nacional de Aviação Civil

No trabalho de Silva *et al* [2] os autores mapearam o processo das missões de fiscalização na ANAC de modo a propor uma solução para dois problemas: otimizar o custo de deslocamento gasto para os inspetores da agência realizar as missões e equalizar as missões com a satisfação dos inspetores. Essa solução utiliza de uma heurística construtiva desenvolvida no software Excel utilizando a linguagem Visual Basic for Applications (VBA).

A heurística desenvolvida cria uma solução inicial que atrela cada missão a uma equipe, não respeitando necessariamente a satisfação homogênea dos inspetores e nem a minimização do custo. Em seguida o algoritmo realiza uma iteração para encontrar soluções que atendem as restrições de maior importância, essas restrições não podem ser flexibilizadas. Após esse passo é realizado mais uma iteração que busca soluções, a partir da solução anterior que atendem as soluções de menor importância, ou seja, as que podem ser flexibilizadas juntamente com a variável de satisfação homogênea, equalizando a medida que for possível a satisfação individual de cada inspetor nas suas missões. A variável de satisfação engloba tanto a satisfação do tipo de atividade a ser realizada quanto o destino da missão.

Os testes foram realizados utilizando 915 missões e 93 inspetores disponíveis e com o índice de satisfação de 10% e carga de trabalho de 10 dias. Foi observado que a medida que o algoritmo atendia as restrições a satisfação geral diminuía, chegando a entorno de 60% de satisfação e o valor final gasto sempre aumentava em relação a solução inicial, devido a regra de satisfação.

Em resumo o trabalho apresenta uma solução utilizando heurística construtiva para, além de otimizar o gasto da realização das missões levar em conta também, a satisfação dos inspetores, porém o trabalho não compara a heurística com uma solução ótima, além de não ser escalável, pois é desenvolvido em um software limitado e não uma linguagem de programação tradicional. Esse documento possui muitos pontos em comum com a modelagem propostas por Silva *et al* [2], porém a estrategia de otimização será utilizando algoritmos sub-ótimos para tornar a otimização em grande escala e remota mais factível.

Capítulo 4

Modelagem de Sistema

O modelo apresentado por este trabalho é uma simplificação do modelo utilizado na ANAC. O objetivo da simplificação foi podar as funcionalidades menos utilizadas e menos influentes na decisão do modelo. Este modelo é de programação inteira binária com o objetivo de alocar com o menor custo os servidores da agência, que iremos chamar de inspetores, nas missões. Deste modo o problema que este trabalho visa resolver é a minimização do custo na realização das missões previamente escolhidas, respeitando todas as restrições do modelo.

As missões são formadas pro três elementos: uma atividade a ser realizada, um aeroporto destino e um inspetor apto. A duração, em dias, da realização da missão é variável, pois depende de dois fatores, o tempo de realização da atividade e o tempo da viagem entre a origem do inspetor e o destino da missão. O custo da realização da missão é variável também, pois depende do origem do inspetor e do destino da missão e caso a origem e destino sejam iguais, o custo é igual à zero. Vale ressaltar que o custo da missão é única e exclusivamente a média da passagem aérea entre determinada origem e destino, qualquer custo de hospedagem ou deslocamento terrestre não é abordado neste modelo.

Em relação às origens e destinos, essas variáveis são descritas como arcos, onde cada origem é uma unidade da federação (UF), todos os destinos são aeroportos que possuem voos comerciais para essa origem, a média do preço das passagens aéreas e a duração discretizada entre meio e um dia. Nos casos das cidades do Rio de Janeiro e São Paulo, onde existem mais de um aeroporto com voos comerciais, é considerado no calculo do custo e do tempo a origem com a menor média do custo da passagem dos aeroportos. Caso um inspetor que tenha origem no estado de São Paulo, seja alocado para uma missão cujo destino é o Aeroporto de Congonhas, tanto o custo quanto o tempo de viagem serão iguais à zero, entretanto se esse mesmo inspetor for alocado em uma missão cujo o destino seja o Aeroporto de São José do Rio Preto, seu custo e tempo de viagem serão determinado pelo valor médio da passagem aérea e da discretização do tempo de viagem.

Tabela 4.1: Conjuntos utilizados no modelo

Conjuntos	**Descrição**	
Conjunto de Nós	N	Origens e Destinos do Brasil
Origem	$O \subset N$	Unidades Federativas (UF) do Brasil
Destinos	$D \subset N$	Aeroportos do Brasil
Missão	K	Atividades demandadas
Pessoas	P	Servidores da ANAC
Grupo	Gr	Capacitação necessária para Atividades

Tabela 4.2: Variáveis do Modelo

Variáveis	Descrição
$Atendida_{k,gr,d,p}$	Indica a Pessoa (P) que atendeu a Missão (K) do Grupo (GR), no destino (D);
$Alocado_{i(i\in N),j(j\in N),p,gr,k}$	Indica o fluxo da Pessoa (P) de um nó (I) (origem ou destino) para outro nó (J) (origem ou destino), capacitada no grupo (GR), que foi atender a missão (K);
$AlocadoGeral_{i,j,p}$	Indica o fluxo da Pessoa (P) de um nó (I) (origem ou destino) para outro nó (J) (origem ou destino);
$TempoUtilizado_p$	Tempo utilizado pela Pessoa (P) nas missões (duração das missões) e viagens (tempo no arco);

Cada inspetor está alocado em uma origem e cada vez que ele realiza uma missão, ele deve obrigatoriamente voltar a sua origem e não pode se descolar para outro destino para realizar uma nova missão. O inspetor também possui uma quantidade de dias disponíveis para realizar as missões, chamado de disponibilidade. Uma lista de certificações é atrelada a cada inspetor dizendo quais atividades ele é apto a realizar. A ordem quem algum inspetor irá realizar as missões não é relevante para o modelo, apenas a quantidade das missões, desde que respeite as restrições.

A Tabela 4.1 apresenta os conjuntos utilizados no modelo. O conjunto de nós é também chamado de arcos. O conjunto dos grupos é a lista de certificações de cada inspetor.A tabela 4.2 apresenta as variáveis do modelo. A tabela 4.3 apresenta os parâmetros do modelo. A equipe neste trabalho terá sempre um único inspetor. A equação a seguir é a função objetivo a ser minimizada neste modelo.

$$min \Sigma_{i,j,p} AlocadoGeral_{i,j,p} * Custo_{i,j} \tag{4.1}$$

O modelo possui seis restrições que serão listas a seguir:

Tabela 4.3: Parâmetros do Modelo

Parâmetros	**Descrição**
$Custo_{i,j}$	Custo da passagem aérea para o trecho (UF-Aeroporto ou Aeroporto- Aeroporto);
$Tempo_{i,j}$	Tempo de Viagem para o trecho (UF-Aeroporto ou Aeroporto- Aeroporto);
$Duração_k$	Duração da Missão (K).
$Equipe_k$	Equipe necessária para a Missão (K).
$Disponibilidade_p$	Disponibilidade em dias da Pessoa (P);
$Demanda_{k,gr,d}$	Existência da Missão K, que requer a capacitação do Grupo (GR), no Destino (D);
$Oferta_{p,gr,o}$	Local da Origem da Pessoa- Servidor da ANAC (P), com capacitação GR. Possibilitará restringir número máximo de missões por servidor.

1. Somente pessoas capacitadas no grupo Gr e com origem O sairão para qualquer destino D;
2. Nenhum fluxo deve ser gerado nos nós de destino, ou seja, não existe arco de origem-destino com mais de um destino;
3. Cada equipe é formada por um único inspetor;
4. O tempo utilizado é o somatório de todos os tempos de missões de cada inspetor;
5. O tempo utilizado de cada inspetor deve ser menor ou igual a sua disponibilidade;
6. Cada inspetor alocado nas diferentes missões gera diferentes variáveis, permitindo que o mesmo possa ser atrelado a missões diferentes, baseado nas restrições de disponibilidade, tempo e custo;

A Restrição 1 é referente a variável *Alocado*, onde só pode ser alocado na missão com destino D o individuo na origem O que possui capacitação no grupo Gr. A Restrição 2 diz que não é possível existir um arco de três destinos, ou seja, não pode existir baldeação entre as missões. A Restrição 3 diz que cada equipe só pode ser formada por um único inspetor, deste modo a variável *Atendida* atrela apenas um único inspetor a uma ou mais missões. A Restrição 4 diz que o tempo disponível para cada inspetor vai diminuindo a medida que o mesmo realiza as missões que geram deslocamento. Na Restrição 5 diz que o tempo de realização de missões e deslocamento não deve exceder a disponibilidade total do inspetor, ou seja, a variável *Tempo Utilizado*, não pode ser maior que o parâmetro *disponibilidade*. Por fim a Restrição 6 diz que cada inspetor pode realizar mais de uma missão, desde que atenda as outras restrições apresentadas, ou seja, a variável *Alocado Geral* deve ser maior ou igual a variável *Alocado*.

Este capitulo apresentou a modelagem do problema de agendamento no contexto da ANAC, demonstrando as regras, restrições, parâmetros e variáveis. O próximo capitulo irá apresentar os algoritmos utilizar para resolver esse problema de agendamento, bem como o sistema web desenvolvido que une os dados da Agência com uma interface amigável ao usuário final.

Capítulo 5

Proposta

Neste capítulo serão apresentadas as implementações dos algoritmos de otimização: SA, GEN, TABU, HMS e HSEQ. Será apresentado o sistema web responsável por fazer a conexão entre os algoritmos de otimização e o banco de dados as missões com o usuário final.

5.1 Meta-heurísticas

A estrutura de dados utilizada, para todos os algoritmos, para representar a escala das missões, ou seja, qual inspetor está alocado em quais missões, é uma matriz onde cada linha representa um inspetor diferente e cada coluna uma missão. Primeiramente, é gerado uma filtragem nas soluções possíveis, eliminando apenas os servidores aptos a realizar as missões escolhidas, ou seja, apenas os servidores que possuem certificação para as missões determinadas. É realizado também uma poda nos arcos de origem e destino, pois somente os arcos serão de interesse. Os arcos possuem as origens nos servidores aptos e conectam com os destinos das missões. Por fim, é criada uma matriz de custo e de tempo a fim de facilitar na comparação no decorrer da execução, contendo apenas as informações de custo e tempo dos inspetores aptos com as missões.

5.1.1 Simulated Annealing

O Algoritmo SA, apresentado no algoritmo 16, segue o padrão de algoritmos de busca local, tal como o Busca Tabu. Esse algoritmo tem motivação baseada na capacidade térmica dos materiais, onde é aquecido o material e em seguida o mesmo é resfriado gradualmente até uma temperatura muito baixa, com o intuito de observar o estado fundamental do sólido. Inicialmente é gerado uma solução inicial válida, como apresentado na linha 1, que atende todas as restrições tanto de disponibilidade quanto de missões. Na

sessão do laço de execução, linhas 2 à 14, é realizado quatro operações diferentes, dentro da função *generateNeighbors*. A matriz da função objetivo é cortada nas quatro direções e os bits que representam o relacionamento de uma missão com um servido são deslocados. A melhor solução válida dentre essas é escolhida para que uma nova serie de cortes seja realizado.

Note que em cada iteração o tamanho do corte é diminuído pela metade, ou seja, para cada vizinho metade da matriz de função objetivo tem seus valores permutados, para cada permutação é gerado um numero equivalentes de vizinhos, no primeiro corte dois vizinhos são gerados, no segundo corte três vizinhos e assim sucessivamente. A matriz da função objetivo tem duas dimensões, onde cada coluna representa uma missão e cada linha representa um inspetor e além disso, quando um ponto da matriz está com o valor 1, significa que o servidor da linha atende a missão da coluna. Quando não for possível mais realizar cortes, ou seja, o tamanho do corte for igual a um, é reiniciado o tamanho do corte para metade do tamanho dos eixos da matriz. Observe que cada corte gera um quadrante na matriz e apenas esse quadrante é deslocado, ou seja, quanto menor o corte, menor a área de deslocamento. Ao fim do algoritmo é salvo a melhor solução encontrada.

5.1.2 Algoritmo Genético

O GEN implementado nesse trabalho segue o pseudo-código 8 onde em um primeiro momento a população inicial utilizada é gerada de forma aleatória obedecendo as restrições do modelo. São gerados quatro cromossomos, chamados de cromossomos pais. A partir desses cromossomos são gerados seis cromossomos filhos. Na linha 3 do algoritmo 8 é indicado que para cada cromossomo a função fitness é calculada, nesse caso o algoritmo implementado calcula a função para os seis cromossomos. A ideia é permanecer com quatro cromossomos a cada geração, deste modo são escolhidos os quatro cromossomos filhos com menor valor de *fitness*, como indicado na linha 4 do algoritmo 8. Caso nenhum cromossomo filho atenda as restrições, é escolhido aleatoriamente três filhos e o pai com menor valor de *fitness* para compor a próxima geração. Caso a quantidade de filhos aptos seja menor que quatro, os cromossomos pai com menor valor de *fitness* são escolhidos para compor a próxima geração.

Todas as gerações possuem pelo menos um cromossomo apto, garantindo que ao final da execução a última geração possua algum cromossomo apto com valor afinado. A cada geração é possível que ocorra uma mutação, como indicado na linha 6 do algoritmo 8 em algum gene de algum cromossomo. Após o termino dos laços definidos, o algoritmo é capaz de fornecer uma combinação satisfatória da escala das missões.

5.1.3 Busca Tabu

O Algoritmo TABU implementado segue o pseudo-código 21 onde inicialmente é gerado um individuo *S0* que cumpre todas as restrições, independente do custo. Essa solução é tida como a melhor solução até o momento e também é inserida na lista tabu, que é uma lista de 60 posições que guarda os melhores resultados até o momento, de modo que se outro individuo igual aparecer, o algoritmo não vai considera-lo como individuo apto. A medida que o algoritmo é executado, ocorre a busca dos vizinhos aptos da solução inicial, como indicado nas linhas 8 a 12, até um número determinado de passos, onde o ultimo individuo encontrado é o individuo otimizado.

5.1.4 Algoritmo Híbrido Multi-Estágio

Na estratégia de hibridização multi-estágio apresentada por [11], um algoritmo evolucionado inicial a população nos estágios iniciais da busca e em seguida um algoritmo de busca local é executado até encontrar um individuo sub-ótimo. Neste trabalho a hibridização foi realizada utilizando o GEN e TABU. No momento de transição dos algoritmos, o cromossomo apto com menor *fitness* é escolhido para iniciar a busca local, como apresentado na Figura 2.1.

5.1.5 Algoritmo Híbrido Sequencial

Na estrategia sequencial um algoritmo realiza a busca em seguida do outro [11]. Neste trabalho também foram escolhidos os Algoritmos Genéticos e Busca Tabu. Inicialmente o GEN começa a otimização e para cada cromossomo o algoritmo TABU realiza sua busca, com listas tabu individuais. Ao final o número de passos executados por essa estrategia é bem maior que as outras estratégias apresentadas acima, como apresentado na Figura 2.1.

Considerando cada um dos algoritmos implementados para esse trabalho, uma pessoa não especialista na área teria dificuldades para reproduzi-los e utilizá-los. Dessa forma. a seguir, a implementação de um sistema web é apresentada para simplificar o acesso e uso dos algoritmos propostos para uso cotidiano por não especialistas.

5.2 Sistema web

A Figura 5.1 apresenta de forma abrangente o funcionamento do sistema web proposto. Primeiramente o usuário final acessa o sistema por algum dispositivo e através dos dados disponíveis pelo banco de dados ele escolhe as pessoas e as missões. Alguns parâmetros

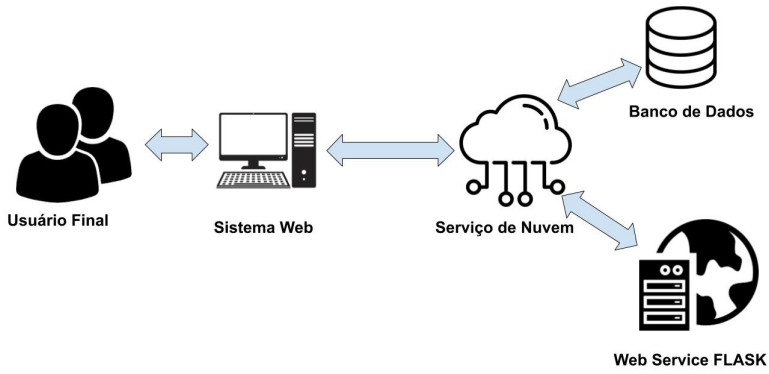

Figura 5.1: Visão geral do sistema web

são customizáveis, como as atividades das missões, bem como suas durações, todas as informações sobre os inspetores e as informações sobre os arcos de origem e destino, como o custo e tempo do arco.

O sistema web foi desenvolvido com o objetivo de ser utilizado em toda a rede da Agência, deste modo o servidor do sistema fica na nuvem da Agência e assim que uma requisição é feita pelo usuário final, o servidor da nuvem faz uma requisição para o servidor FLASK. Assim que o servidor FLASK recebe a solicitação do servidor da nuvem, ele envia as informações novas a respeito das missões, arcos e inspetores e chama a função de otimização. Quando a otimização é terminada o servidor FLASK retorna a nuvem a lista otimizada de inspetores e missões e por fim o usuário recebe na tela de seu dispositivo a escala otimizada.

O sistema FLASK utiliza a arquitetura REST e suas requisições são assíncronas. Essa arquitetura facilita a atualização e inserção de novos dados no banco de dados da aplicação. Antes da otimização em si o usuário pode realizar todas as alterações possíveis e assim que a configuração dos dados for feita o sistema chama a função de otimização passando os dados via JSON. O Servidor FLASK retorna para a aplicação web o resultado como uma tabela JSON.

A Figura 5.2 apresenta a arquitetura da aplicação web. Na construção dessa aplicação foram utilizados um servidor FLASK com gerenciador de URL SWAGGER e na parte visual foi utilizado o framework AJAX. Primeiramente o usuário realiza uma requisição do tipo CRUD (*Create, Read, Update, and Delete*) na interface gráfica e em seguida o controle do servidor AJAX é acionado a partir da função de clique. O controle chama o *model* passando os parametos colhidos na interface gráfica. Esses parametros são passados para o controle do servidor FLASK, que irá buscar informações a partir do model do servidor

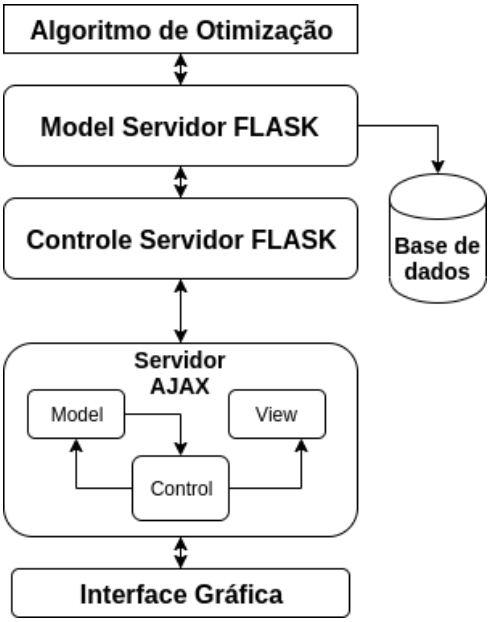

Figura 5.2: Arquitetura da aplicação web

FLASK na base de dados. Quando os dados forem retornados o *model* do servidor AJAX irá comunicar com o controlador para fazer alguma alteração na view.

Essa arquitetura permite uma modularidade tanto a nível de interface gráfica quanto a nível de base de dados. A respeito a base de dados, nesse caso são utilizados tabelas JSON pois a quantidade de dados não é suficientemente grande para justificar o uso de um banco de dados estruturado, além de não existir uma estrutura agregadora de dados no contexto da Agência. A aplicação funciona por meio de sessões, ou seja, quando um usuário troca a base de dados inicial, as informações da otimização anterior não é salva. Após o usuário realizar todas as configurações necessárias para a otimização, o model do servidor FLASK chama o algoritmo de otimização e aguarda até a sua execução ser completada e envia, passando por todo o fluxograma, para a interface gráfica a escala das missões.

A Figura 5.3 apresenta a tela inicial do sistema web. O primeiro input do sistema é referente a tabela de dados que será utilizada como base da otimização, essa tabela é padronizada na ANAC e todo o sistema foi construído a partir dela. Após o usuário fazer o upload da tabela, os campos de Pessoas, Missões, Arcos e Certificados são preenchidos automaticamente. Inicialmente é mostrado a tabela somente com a informação dos inspetores, sendo o nome do inspetor, sua origem e sua disponibilidade. É possível inserir novos inspetores, bem como excluir inspetores existentes e modificar alguma informação de um inspetor especifico. Na modificação e na exclusão dos inspetores da tabela, o usuário deve clicar duas vezes na linha do inspetor desejado.

A segunda tela do sistema é apresentada na Figura 5.4 onde é apresentado a tabela

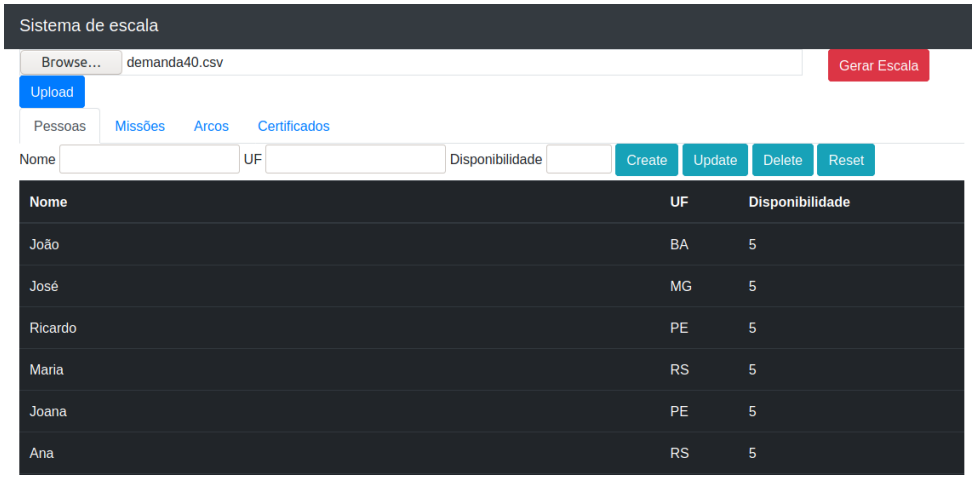

Figura 5.3: Tela inicial do sistema web

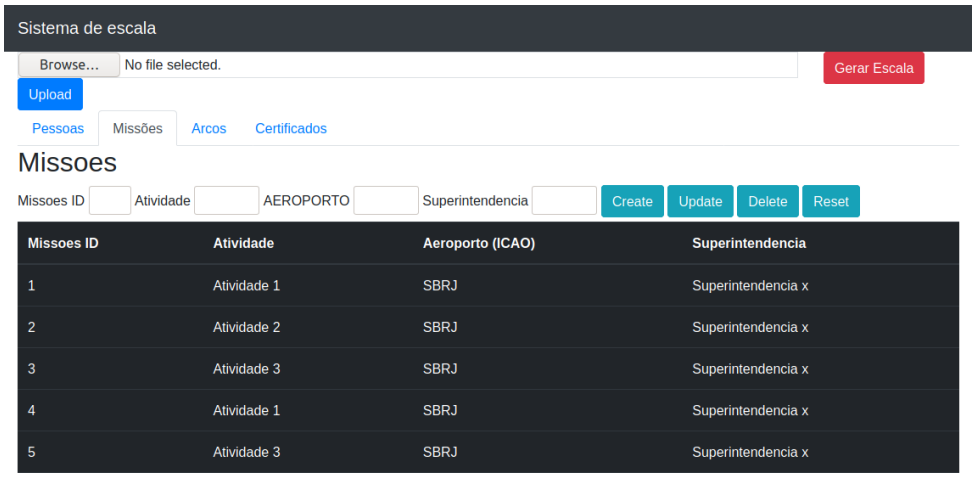

Figura 5.4: Tela de missões do sistema

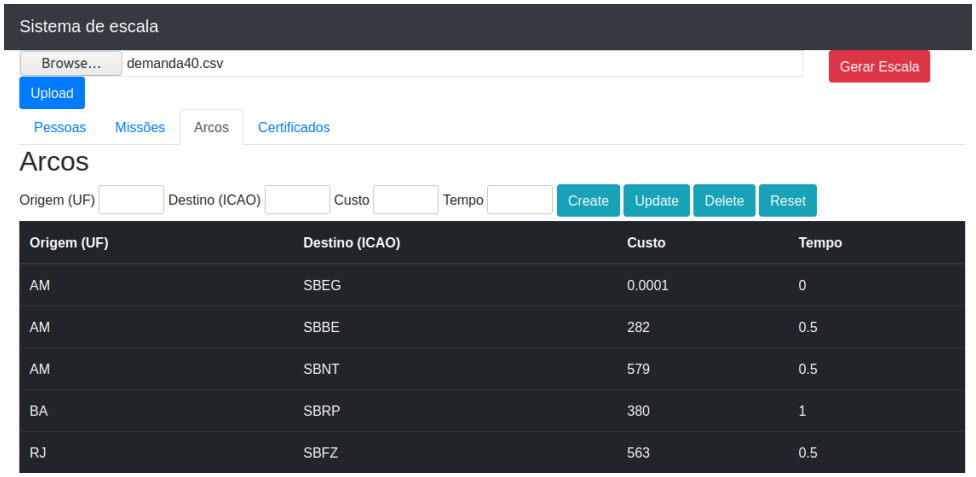

Figura 5.5: Tela de arcos do sistema

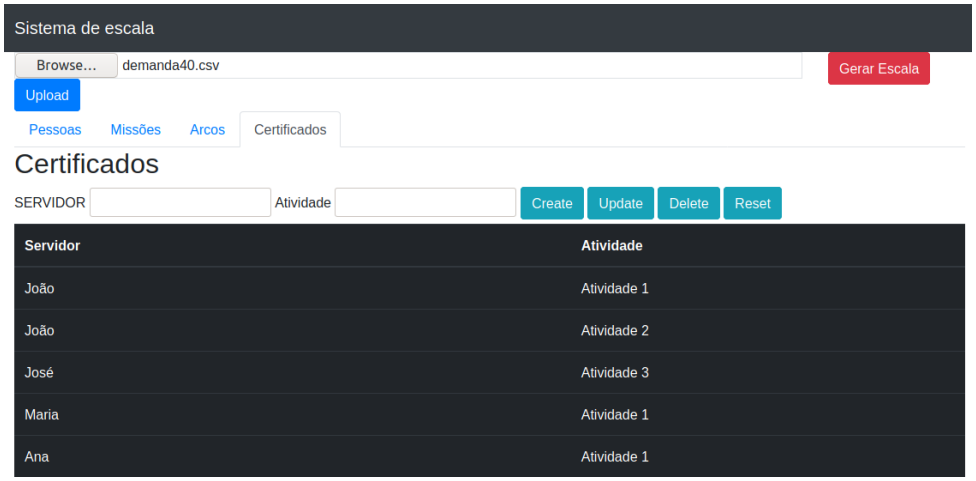

Figura 5.6: Tela de certificações do sistema

Nome	UF	Atividade	Aeroporto	Custo	Tempo
Ana	RJ	Atividade 1	SBRJ	0.0001	2
João	RJ	Atividade 2	SBRJ	0.0001	3
José	RJ	Atividade 3	SBRJ	0.0001	1
Maria	RJ	Atividade 1	SBRJ	0.0001	2

Figura 5.7: Tela de resultado do sistema

de missões. Todas as operações de criação, exclusão e modificação apresentadas na tela anterior se aplica nessa e nas telas seguintes. A tabela de missões possui o identificador da missão, a atividade, o destino da missão e a superintendência da missão. A Figura 5.5 apresenta parte da tabela de arcos do sistema. Os arcos são todas as combinações da origem com o destino juntamente com seu custo de deslocamento e tempo discretizado de deslocamento. A Figura 5.6 apresenta todas as atividades que os inspetores possui. A Figura 5.7 apresenta a ultima tela do sistema, quando o usuário clica no botão "Gerar Escala"no canto superior direito da tela. Após o usuário clicar nesse botão é calculado a escala utilizando o algoritmo hibrido multi-estágio, e a tabela da escala é mostrada na tela com as informações referentes as missões que cada pessoa foi alocada. Dessa forma, usuários não especialistas podem utilizar as soluções meta-heurísticas e ótimas desconhecendo os detalhes técnicos das soluções.

No próximo capítulo, diferentes resultados são apresentados para comparar as soluções meta-heurísticas propostas nesse trabalho para resolver o problema de alocação de servidores e missões (atividades). Para realizar tal comparação, um caso de uso baseado na base de dados fornecida pela ANAC será considerada.

Capítulo 6

Resultados

Na execução dos experimentos foram criados 3 cenários baseados em uma base de dados da ANAC. A base de dados utilizada foi anonimizada por ser de cunho privado e não é disponibilizada abertamente para uso. Dessa forma, os parâmetros como missões, servidores, custo de movimentação e etc, são valores reais extraídos dessa base de dados privada e somados a dados fictícios para anonimização dos mesmos. Cada um dos algoritmos propostos nesse trabalho foram comparados de acordo com os cenários gerados baseados nas bases de dados da ANAC.

Os cenários foram criados baseados na variação de dois fatores: (*i*) número de servidores; (*ii*) quantidade de missões; e (*iii*) variação de servidores e missões disponíveis, no intuito de achar qual a melhor variável para medir o desempenho entre os algoritmos. No cenário onde a quantidade de missões é variada, tem-se a faixa de {15, 45, 50, 60, 65, 70, 75} missões utilizada. Já, para o cenário com variação na quantidade de servidores, o fator é variado na faixa de {5, 100, 115, 120, 130, 140, 145, 150} servidores removidos de um montante de 187 servidores disponíveis, provenientes da base de dado da ANAC. É importante mencionar que a retirada acima de 150 pessoas torna a solução infactível. No terceiro cenário, uma combinação das variações de ambos, missões e servidores são avaliados comparativamente, com os pares de variações (missões, servidores) variados na faixa de {(50,100), (55,100), (50, 150), (55, 150)}. Em cada cenário são realizadas 40 repetições com combinações aleatórias para obter um intervalo de confiança de 80% de nível de confiança. Independentemente do cenário selecionado, todos os inspetores e missões escolhidas foram amostrados aleatoriamente, nenhum tipo de seleção controlada foi realizada. A solução ótima de [2] foi implementada no software LINGO e foi utilizada como linha base para os experimentos desenvolvidos e doravante referenciada apenas por "LINGO".

6.1 Algoritmos meta-heurísticos sem hibridização

A Figura 6.1 mostra a comparação no cenário onde a quantidade de missões é fixada em 10 e o número de pessoas disponíveis é variada. No eixo x pode-se verificar a variação da quantidade de servidores retirados. No eixo y, por sua vez, pode-se observar o Custo em R$ alcançado para deslocar os servidores para executarem as missões selecionadas de acordo com os preços tabelados no conjunto de dados extraído da ANAC. Os custos variam na faixa de R$ 7.500,00 à 15.000,00 no pior caso, conforme pode ser observado na Figura 6.1

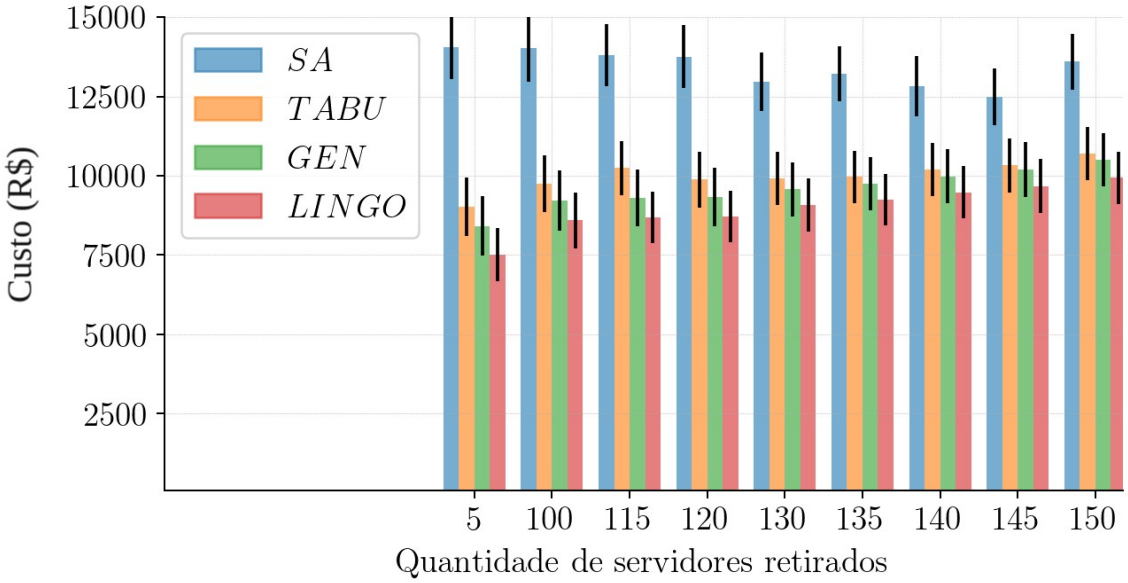

Figura 6.1: Variação da quantidade de pessoa quando o número de missões é fixo;

No primeiro cenário, pode-se perceber que a solução que mais se aproximou da linha-base foi o GEN. Já, o algoritmo com o pior desempenho foi o SA. Ao observar as variações do número de servidores removidos, percebe-se que a quantidade de servidores não altera dramaticamente o valor de custo das associações com as missões. Entretanto, quando o número de servidores decresce para abaixo de 37 (187-150), a solução torna-se infactível. Dessa forma, percebe-se que o número de servidores adiciona uma restrição operacional binária do tipo possível ou não possível ao problema, diferente do que se era esperado, conforme resultados em [2]. Para se aprofundar a investigação, na Figura 6.2, tanto o número de servidores quanto missões são variados.

A Figura 6.2 mostra a comparação no segundo cenário, onde a quantidade de missões é variada na faixa de {50,55} missões e a quantidade de inspetores removidos é também variado na faixa de {50,55} inspetores. No eixo x pode-se verificar a variação de missões e inspetores retirados, observando-se que os dois primeiros conjuntos de dados a quantidade

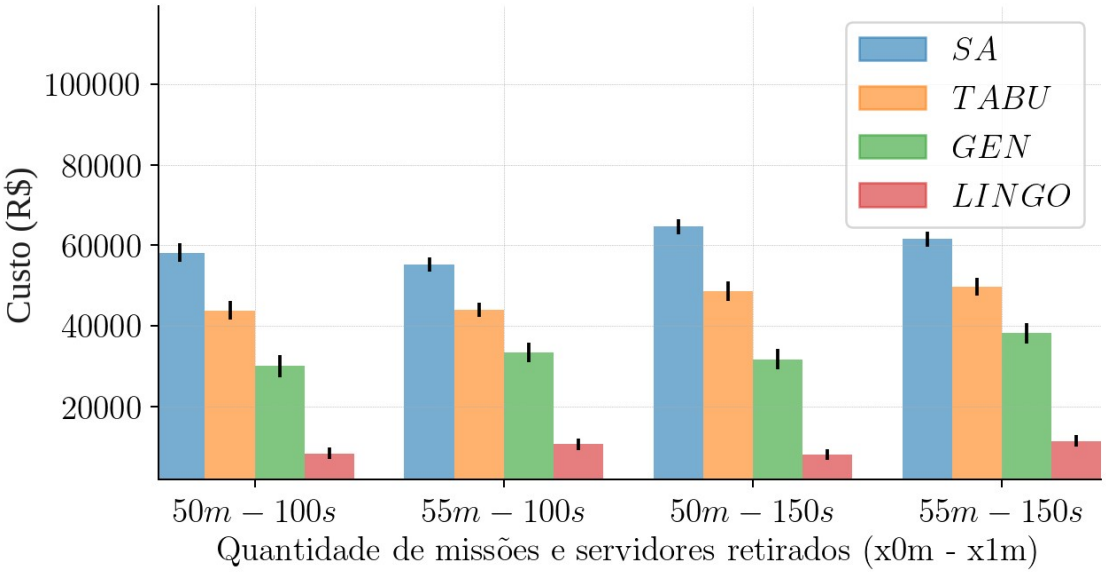

Figura 6.2: Visão geral do sistema web

de missões foi variada entre 50 e 55 enquanto a quantidade de inspetores retirados foi fixado em 100, no segundo conjunto de dados a quantidade de missões também foi variada entre 50 e 55 porém a quantidade de inspetores retiradas foi fixada em 150. O eixo y, por sua vez , pode-se observar o Custo em R$ alcançado para deslocar os inspetores para executarem as missões selecionadas de acordo com os preços tabelados no conjunto de dados extraídos da ANAC. Os custos variam na faixa de R$ 300,00 à R$ 60.500,00 no pior caso, conforme observado na figura 6.2.

Neste cenário, pode-se perceber que a solução que mais se aproximou da linha-base foi o GEN. Já o algoritmo com o pior desempenho foi o SA. Ao observar a variação conjunta de quantidade de missões e número de inspetores retirados, percebe-se que a quantidade de servidores também não altera drasticamente o valor de custo das associações com as missões. Também é observado que a pequena variação na quantidade de missões não altera drasticamente os valores de custo, porém um leve aumento é percebido. Desta forma o próximo cenário, com resultados mostrados na figura 6.3, aprofunda a análise quando a quantidade de missões é variada entre uma faixa de {15, 45, 50, 60, 65, 70, 75} missões com a presença dos algoritmos híbridos, pois a variável de quantidade de missões se mostra mais determinante na aferição do desempenho dos algoritmos.

6.2 Algoritmos meta-heurísticos híbridos

A figura 6.3 mostra a comparação no terceiro cenário, onde a quantidade de missões é variada na faixa de {15, 45, 50, 60, 65, 70, 75} missões e a quantidade de inspetores

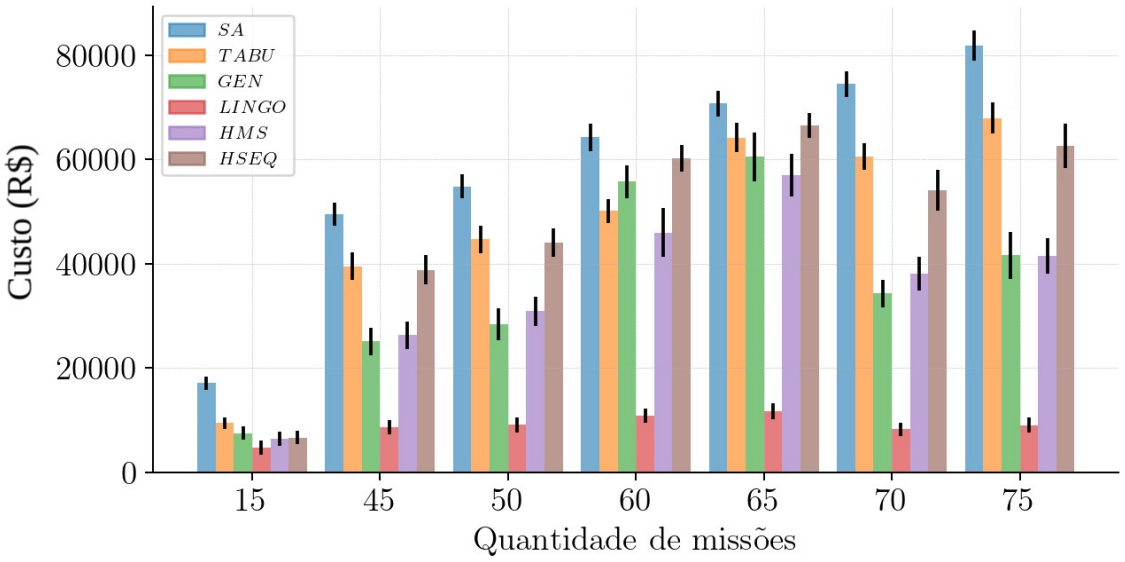

Figura 6.3: Variação do número de missões com a quantidade máxima de inspetores disponível;

removidos é fixada em zero, ou seja, nenhum inspetor foi removido. No eixo x pode-se verificar a variação de missões. O eixo y, por sua vez , pode-se observar o Custo em R$ alcançado para deslocar os inspetores para executarem as missões selecionadas de acordo com os preços tabelados no conjunto de dados extraídos da ANAC. Os custos variam na faixa de R$ 300,00 à R$ 80.100,00 no pior caso, conforme observado na figura 6.2. Neste cenário foram comparados os algoritmos TABU, GEN, SA, HMS e HSEQ.

Neste cenário, pode-se perceber que as soluções que mais se aproximaram da linha-base foram o GEN e o HMS. Já o algoritmo com o pior desempenho continuou sendo o SA. Ao observar a variação das missões percebe-se que em alguns momentos os algoritmos com melhor desempenho, GEN e HMS, se mostram indiferenciáveis entre si. Quando a quantidade de missões é 60, o HMS tem o melhor desempenho, em relação aos outros algoritmos comparados. É possível observar também que a tendência do algoritmo ótimo, LINGO, não é de aumentar seu custo enquanto a quantidade de missões aumenta, isso se dá pelo fato das missões escolhidas, independente da quantidade, ser aleatório, deste modo o aumento na quantidade de missões pode não influenciar o acréscimo do custo na solução ótima.

Este capitulo apresentou a comparação entre o desempenho de cada algoritmo nos três cenários propostos. O próximo capitulo irá finalizar esse trabalho com as considerações a respeito desses resultados e de possibilidades de trabalhos futuros.

Capítulo 7

Conclusão e Agradecimentos

O presente trabalho apresentou uma comparação entre cinco meta-heurísticas em um problema real de agendamento baseado em HSTP com uma linha base baseada na solução ótima de programação inteira mista desenvolvida no software LINGO. Foi observado que o algoritmo híbrido entre o GEN e TABU utilizando a estrategia multi-estágio se mostrou com melhor aproximação do resultado ótimo. Foi observado também que dentre os algoritmos sem hibridização, o SA obteve pior desempenho enquanto o GEN obteve o melhor desempenho. A escolha da ordem dos algoritmos na hibridização foi baseada nos trabalhos da literatura que definem que algoritmos evolucionários possuem melhor desempenho nos estágios iniciais da busca, enquanto algoritmos de busca local possui um desempenho melhor nos estágios finais.

A partir dos resultados descrito acima, foi realizado também o desenvolvimento de uma aplicação web, utilizando a arquitetura REST. Essa aplicação tem como objetivo agregar os dados referentes as missões e inspetores da ANAC e os algoritmos de otimização, de modo a facilitar o processo de geração de escala das missões. A aplicação foi construída de modo que o gargalo do tempo de execução do algoritmo de otimização não seja percebido pelo usuário final. Também foi levado em conta a interação do usuário com a aplicação, de modo que seja simples e fácil a sua utilização e personalização das variáveis e parâmetros.

Um próximo passo para esse trabalho é a comparação de outras técnicas de hibridização além da utilização de outras meta-heurísticas, tanto evolucionarias quanto de busca local. A utilização de técnicas de otimização de tempo de execução, como paralelismo e linguagens de baixo nível, também é um caminho a ser seguido, pois quanto maior o conjunto de dados maior o tempo gasto por um algoritmo para resolver o problema do agendamento. Em relação a aplicação web desenvolvida é sugerido a utilização de técnicas e arquiteturas de serviços web mais robustas para melhorar a conexão entre os dados e a execução dos algoritmos de otimização.

Referências

[1] Babei, H. Karimpour, J Hadidi A: *A survey of approaches for university course timetabling problem*, 2015. 1, 2, 11, 12

[2] Silva, Marco Antonio Diniz: *Escalamento de inspetores de aviação civil no brasil: Formalização do problema e metodologia de resolução.* 2018. 1, 4, 15, 27, 28

[3] Breslaw, Jon A: *A linear programming solution to the faculty assignment problem.* Socio-Economic Planning Sciences, 10(6):227–230, 1976. 1, 4

[4] Weise, Thomas: *Global Optimization Algorithms - Theory and Application.* Self-Published, second edição, 6 26, 2009. `http://www.it-weise.de/`, Online available at http://www.it-weise.de/. 2

[5] Yamazaki, V. H. Pertoft, J.: *Scalability of a genetic algorithm that solves a universitycourse scheduling problem inspired by kth*, 2014. 2

[6] DAHL, J. FREDRIKSON, R.: *A comparative study between a simulated annealing and a genetic algorithm for solving a university timetabling problem*, 2016. 2, 14

[7] Feng, Xuehao, Yuna Lee e Ilkyeong Moon: *An integer program and a hybrid genetic algorithm for the university timetabling problem.* Optimization Methods and Software, 32(3):625–649, 2017. 2, 13, 14

[8] Burke, EK, Jeffrey Kingston e D De Werra: *5.6: Applications to timetabling.* Handbook of graph theory, 445, 2004. 4

[9] Saviniec, Landir e Ademir Aparecido Constantino: *Effective local search algorithms for high school timetabling problems.* Applied Soft Computing, 60:363–373, 2017. 4

[10] Blum, Christian e Andrea Roli: *Metaheuristics in combinatorial optimization: Overview and conceptual comparison.* ACM computing surveys (CSUR), 35(3):268–308, 2003. 5, 6

[11] Ting, TO, Xin She Yang, Shi Cheng e Kaizhu Huang: *Hybrid metaheuristic algorithms: past, present, and future.* Recent advances in swarm intelligence and evolutionary computation, páginas 71–83, 2015. 6, 9, 12, 14, 22

[12] Salman, A. Hanna, R: *A comparative study between genetic algorithm, simulated annealing and a hybrid algorithm for solving a university course timetabling problem*, 2018. 6, 14

[13] Glover, F.: *Future paths for integer programming and links to artificial intelligence.* Comput. Oper. Res., 13(13):533–549, 1986. 6

[14] Mitchell, Melanie: *An introduction to genetic algorithms.* MIT press, 1998. 8

[15] Kirkpatrick, S., Gelatt Jr. C. Vecchi M. P.: *Optimization by simulated annealing.* Science, 220(220):671–680, 1983. 9

[16] Al-Jarrah, Mohammad A, Ahmad A Al-Sawalqah e Sami F Al-Hamdan: *Developing acourse timetable system for academic departments using genetic algorithm.* Jordanian Journal of Computers and Information Technology (JJCIT), 3(1), 2017. 12