INSTITUTO FEDERAL DA PARAÍBA – IFPB
DEPARTAMENTO DE ENSINO SUPERIOR
COORDENAÇÃO DO CURSO DE BACHARELADO EM
ENGENHARIA ELÉTRICA

CALEBE OLIVEIRA DE FIGUEIRÊDO

HIPER-HEURÍSTICA DE GERAÇÃO APLICADA AO PROBLEMA DO POSICIONAMENTO AUTOMÁTICO DE PONTOS DE ACESSO DE REDES SEM FIO EM AMBIENTES INTERNOS

JOÃO PESSOA – PB
2023

CALEBE OLIVEIRA DE FIGUEIRÊDO

HIPER-HEURÍSTICA DE GERAÇÃO APLICADA AO PROBLEMA DO POSICIONAMENTO AUTOMÁTICO DE PONTOS DE ACESSO DE REDES SEM FIO EM AMBIENTES INTERNOS

Trabalho de Conclusão de Curso apresentado à Coordenação do Curso Superior de Bacharelado em Engenharia Elétrica, no Instituto Federal da Paraíba – IFPB, como requisito parcial à obtenção do grau de Bacharel em Engenharia Elétrica.

JOÃO PESSOA – PB

2023

Dados Internacionais de Catalogação na Publicação (CIP)
Biblioteca Nilo Peçanha do IFPB, *campus* João Pessoa

F475h Figueirêdo, Calebe Oliveira de.

Hiper-heurística de geração aplicada ao problema do posicionameto automático de pontos de acesso / Calebe Oliveira de Figueirêdo . - 2023.

48 f. : il.

TCC (Graduação - Curso Superior de Bacharelado em Engenharia Elétrica) - Instituto Federal de Educação da Paraíba / Unidade Acadêmica de Processos Industriais, 2023.

Orientação : Profº DSc. Thiago Gouveia.

1.Heurística. 2. Algoritmos. 3. Posicionamento de pontos de acesso. 4. Redes sem fio. I. Título.

CDU 004.023(043)

Lucrecia Camilo de Lima
Bibliotecária - CRB 15/132

CALEBE OLIVEIRA DE FIGUEIRÊDO

HIPER-HEURÍSTICA DE GERAÇÃO APLICADA AO PROBLEMA DO POSICIONAMENTO AUTOMÁTICO DE PONTOS DE ACESSO DE REDES SEM FIO EM AMBIENTES INTERNOS

Trabalho de Conclusão de Curso apresentado à Coordenação do Curso Superior de Bacharelado em Engenharia Elétrica, no Instituto Federal da Paraíba – IFPB, como requisito parcial à obtenção do grau de Bacharel em Engenharia Elétrica.

Trabalho Aprovado em 10 de fevereiro de 2023.

BANCA EXAMINADORA

gov.br Documento assinado digitalmente
Thiago Gouveia da Silva
Data: 15/03/2023 14:22:20-0300
Verifique em https://validar.iti.gov.br

Prof. Dsc. Thiago Gouveia (Orientador)
Instituto Federal da Paraíba – IFPB

gov.br Documento assinado digitalmente
PATRIC LACOUTH DA SILVA
Data: 15/03/2023 17:39:14-0300
Verifique em https://validar.iti.gov.br

Prof. DSc. Patric Lacouth da Silva
Instituto Federal da Paraíba – IFPB

gov.br Documento assinado digitalmente
VALERIA MARIA BEZERRA CAVALCANTI MAC
Data: 15/03/2023 18:05:13-0300
Verifique em https://validar.iti.gov.br

Prof. MSc. Valéria Maria Bezerra Cavalcanti Maciel
Instituto Federal da Paraíba – IFPB

À Deus e à minha família.

AGRADECIMENTOS

Agradeço primeiramente à Deus, porque sem Ele, nada disto seria possível.

Agradeço à minha família, representada pelo meu pai Moisés Veríssimo de Figueirêdo, minha mãe Maria do Disterro Oliveira de Figueirêdo, minha irmã Quézia Figueirêdo de Barros, meu irmão Jeriel Oliveira de Figueirêdo e minha esposa Andressa Theotônio Alves de Figueirêdo. Sem todos os sacrifícios que fizeram até hoje, eu não seria a pessoa que sou. Tenho sentimentos de grande privilégio ao fazer parte desta família que me moldaram para ser a melhor pessoa que eu pudesse ser.

Agradeço aos amigos que tive a oportunidade de conhecer durante o curso, como Ana Beatriz, Arthur Barôncio, Bianca e Jessly.

Agradeço aos amigos que conheci antes do curso e me ajudaram a passar por toda a jornada, como Allan Alves, Armando Bonifácio, Gabrielle Cassol, Esther Paula, Edson Mesquita, Matheus Petrúcio e Pedro Moraes.

Agradeço ao corpo integrante do Projeto Olímpico de Programação (POP), representados pela professora Valéria Maria Bezerra Cavalcanti Maciel, Thiago Gouveia, Kerven Maciel Monteiro de Albuquerque, Victor Herbert e Mateus de Lima Melo. Foi neste grupo que comecei a minha jornada na programação, ganhei amigos e mentores. Sou muito feliz pelo POP e espero continuar ajudando nos próximos anos.

Agradeço ao corpo docente do curso de Bacharelado em Engenharia Elétrica do Instituto Federal da Paraíba, em especial para os professores Patric Lacouth da Silva, Lincoln Machado de Araújo e Jefferson Costa e Silva. Obtive excelentes ensinamentos e experiências de aprendizagem diferentes neste curso, nas quais levarei comigo daqui em diante como formadores de caráter.

Novamente agradeço à minha esposa Andressa, que me acompanha de perto desde antes do curso, e compartilhou da mesma jornada de autoconhecimento. Juntos, nós nos apoiamos e seguimos em frente, sempre gratos por tudo o que temos. É incrível ver o que nos tornamos ao olharmos para trás.

Por fim, agradeço novamente ao professor Thiago Gouveia, pela paciência e atenção enquanto orientador. É possível observar claramente sua paixão pelo que faz e seu interesse por fazer trabalhos de qualidade. És um grande exemplo de pessoa e profissional, que nos inspiram a ser pessoas cada vez melhores.

Tudo posso naquele que me fortalece.

(Filipenses 4:13)

RESUMO

O projeto de redes sem fio tem se tornado cada vez mais relevante nos últimos anos. Com o advento da popularização de dispositivos de Internet das Coisas e 5G, este assunto tem tomado ainda mais relevância, devido aos custos relacionados à instalação deste tipo de projeto. Ao modelar o projeto destas redes como o Problema de Posicionamento de Pontos de Acesso, pode-se verificar que é um problema de natureza NP-Difícil e, portanto, ainda não possui um algoritmo que resolva o problema em tempo polinomial. Este trabalho propõe uma solução para o problema baseada em uma hiper-heurística de geração, na qual cada solução é acompanhada de uma heurística nova que pode ser reutilizada para novas instâncias. São utilizadas instâncias randômicas para redes sem fio com dimensões relevantes para o trabalho. É realizada uma comparação dos resultados das execuções destas instâncias com uma abordagem de meta-heurística. Os experimentos computacionais apontam excelentes resultados para dimensões maiores, de maneira a demonstrar que esta abordagem consegue ser mais escalável e produzir soluções razoáveis para o problema.

Palavras-chave: Posicionamento de Pontos de Acesso; Hiper-heurísticas; Redes Sem Fio.

ABSTRACT

Wireless network design has become increasingly relevant in recent years. With the advent of popularization of Internet of Things and 5G devices, this subject has become even more relevant, due to the costs related to the installation of this type of project. By modeling the design of these networks as the Access Point Positioning Problem (APPP), it can be seen that it is a NP-Hard problem and, therefore, does not yet have an algorithm that solves the problem in polynomial time. This work proposes a solution to the problem based on a generation hyper-heuristic, in which each solution is accompanied by a new heuristic that can be reused for new instances. Random instances are used for wireless networks with dimensions relevant to the work. A comparison of the execution results of these instances is carried out using a meta-heuristic approach. Computational experiments point to excellent results for higher dimensions, in order to demonstrate that this approach can be more scalable and produce reasonable solutions to the problem.

Keywords: Access Point Positioning; Hyper-heuristics; Wireless Networks.

LISTA DE FIGURAS

Figura 1 – Classificação de hiper-heurísticas. 21
Figura 2 – Sequência de *bits* de uma solução. 32
Figura 3 – Exemplo de uma discretização de uma planta-baixa. 34
Figura 4 – Exemplo de uma instância de teste e a diferença entre seus pontos. . . 36

LISTA DE TABELAS

Tabela 1 – Modelo de atenuação para determinados anteparos. 35
Tabela 2 – Tipos de roteadores utilizados. 37
Tabela 3 – Execução de testes com instâncias de dimensões 8x14. 39
Tabela 4 – Execução de testes com instâncias de dimensões 13x13. 39
Tabela 5 – Execução de testes com instâncias de dimensões 11x23. 39
Tabela 6 – Execução de testes com instâncias de dimensões 25x25. 40
Tabela 7 – Execução de testes com instâncias de dimensões 30x30. 40
Tabela 8 – Execução de testes com instâncias de dimensões 50x25. 40
Tabela 9 – Execução de testes com instâncias de dimensões 50x50. 41
Tabela 10 – Execução de testes com instâncias de dimensões 100x50. 41

LISTA DE ALGORITMOS

Algoritmo 1 – Procedimento de um *Iterated Local Search* (ILS) 19
Algoritmo 2 – Procedimento de um Algoritmo Genético (AG) 20

Algoritmo 3 – Heurística de *AllChangeLink()* 27
Algoritmo 4 – Heurística de *AllDryRouter()* 28
Algoritmo 5 – Heurística de *BestAddRouter()* 29
Algoritmo 6 – Heurística de *AllBestRadiusType()* 29
Algoritmo 7 – Heurística de *BestLink()* 30
Algoritmo 8 – Heurística de *AllBestCapacityType()* 30
Algoritmo 9 – Heurística de *AllClosestMoveRouter()* 31
Algoritmo 10 – Hiper-heurística de construção 31
Algoritmo 11 – Critério de *crossover* 33
Algoritmo 12 – Critério de mutação 33

LISTA DE ABREVIATURAS E SIGLAS

ACO	*Ant Colony Optimization*
AG	Algoritmo Genético
AP	*Access Point*
APPP	*Access Point Positioning Problem*
CPU	*Central Processing Unit*
GRASP	*Greedy Randomized Adaptive Search Procedures*
ILS	*Iterated Local Search*
MA	*Memetic Algorithm*
NP	*Non-deterministic Polynomial Time*
RAM	*Random-Access Memory*
SA	*Simulated Annealing*
TS	*Tabu Search*
WLAN	*Wireless Local Area Networks*

SUMÁRIO

1 Introdução 13

2 Fundamentação Teórica 15

2.1 ALGORITMOS 15

2.2 HEURÍSTICAS 16

2.2.1 Heurísticas Construtivas 16

2.2.2 Heurísticas de Refinamento 17

2.3 META-HEURÍSTICAS 18

2.3.1 *Iterated Local Search* 18

2.3.2 Programação Genética 19

2.4 HIPER-HEURÍSTICAS 20

2.4.1 Hiper-heurísticas de seleção 21

2.4.2 Hiper-heurísticas de geração 22

3 Revisão da literatura 23

4 Metodologia 25

4.1 FORMULAÇÃO MATEMÁTICA UTILIZADA 25

4.2 HIPER-HEURÍSTICA DE GERAÇÃO PROPOSTA 26

4.2.1 Heurísticas de construção selecionadas 26

4.2.2 Heurísticas de busca local selecionadas 27

4.2.3 Estrutura da hiper-heurística 30

4.2.3.1 Método de *crossover* 32

4.2.3.2 Método de mutação 33

4.3 INSTÂNCIAS DE TESTE UTILIZADAS 33

5 Resultados e experimentos computacionais 38

6 Considerações Finais 43

REFERÊNCIAS 44

1 INTRODUÇÃO

As redes sem fio têm se tornado cada vez mais populares nos últimos anos, devido à sua flexibilidade e facilidade de uso. No entanto, a implementação de redes sem fio em ambientes internos pode ser desafiadora devido às limitações físicas do espaço e às interferências de outros dispositivos eletrônicos. O posicionamento automático de pontos de acesso (APs) é uma tarefa importante para garantir que as redes sem fio funcionem de forma eficiente e confiável. Neste contexto, surgem problemas de otimização combinatória da classe NP-Difícil que podem ser modelados matematicamente e/ou tratados por abordagens exatas e heurísticas.

Por sua vez, hiper-heurísticas são abordagens recentes para resolver problemas de otimização combinatória. Esta técnica combina diferentes métodos de heurísticas para encontrar soluções melhores e mais precisas. As hiper-heurísticas têm sido aplicadas com sucesso a diversos problemas, como otimização de rotas (OLGUN; KOÇ; ALTIPARMAK, 2021), alocação de recursos (ZHUANG; ZHOU, 2020) e planejamento de manutenção (DAHAL; ALDRIDGE; GALLOWAY, 2007).

Dado que a solução exata de grandes instâncias de problemas NP-Difíceis é intratável, este trabalho propõe uma hiper-heurística de geração de heurísticas para o Problema do Posicionamento de Pontos de Acesso - do inglês, *Access Point Positioning Problem* (APPP) - em ambientes internos. O objetivo é desenvolver uma abordagem hiper-heurística que possa ser usada para gerar heurísticas personalizadas para o problema do posicionamento de APs, levando em conta fatores como a cobertura, a capacidade e a interferência.

Para alcançar este objetivo, será utilizada uma técnica de geração de heurísticas baseada em Programação Genética (KOZA, 1994). A partir de uma série de heurísticas de construção e heurísticas de busca local de baixo nível, estas técnicas serão combinadas para gerar heurísticas personalizadas para o problema do posicionamento de APs.

Este trabalho irá avaliar a eficácia da abordagem hiper-heurística de geração de heurísticas proposta comparando-a com outra abordagem existente na literatura. A avaliação será realizada em ambientes simulados, usando diferentes cenários de teste.

Apos esta introdução, o restante do trabalho está organizado como segue. O Capítulo 2 discute pontos cruciais para o entendimento do trabalho, de maneira a definir algoritmos, heurísticas, meta-heurísticas e hiper-heurísticas, além de descrever exemplos

destes.

O Capítulo 3 revisita a literatura sobre APPP ao focar especificamente em abordagens com heurísticas. Por sua vez, o Capítulo 4 apresenta a formulação matemática utilizada para resolver o problema, além da exposição, em detalhes, da hiper-heurística construída, bem como as instâncias de testes a serem utilizadas. Já o Capítulo 5 expõe os resultados obtidos da hiper-heurística elaborada, ao realizar experimentos computacionais em instâncias de teste, e comparando com uma abordagem de meta-heurística da literatura. Por fim, o Capítulo 6 fornece considerações finais sobre este trabalho.

Finalmente, apresentaremos as conclusões do trabalho e sugestões para futuras pesquisas no campo do projeto de redes sem fio. Este trabalho tem como objetivo fornecer uma visão geral dos desafios e soluções existentes no projeto de redes sem fio, e contribuir para a compreensão das melhores práticas no desenvolvimento de redes sem fio. Além disso, sugeriremos possíveis áreas de pesquisa futura, como possíveis melhorias na criação de heurísticas para a resolução do problema.

2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo descreve conceitos essenciais para a construção e compreensão deste trabalho. A Seção 2.1 define termos importantes de algoritmos. A Seção 2.2 explica o que são heurísticas e qual é o seu papel, bem como suas categorias. Por sua vez, a Seção 2.3 define meta-heurísticas, seus tipos e apresenta exemplos. O capítulo termina com uma apresentação sobre hiper-heurísticas, na Seção 2.4, e categoriza as suas maneiras de atuação.

2.1 ALGORITMOS

Um algoritmo pode ser definido como um procedimento computacional em que, dado um valor ou um conjunto de valores de **entrada**, é retornado um valor ou um conjunto de valores de **saída**. Assim, um algoritmo é uma sequência de passos computacionais que transformam a entrada em uma saída (CORMEN *et al.*, 2009).

Um algoritmo também pode ser visto da perspectiva de uma ferramenta para resolver um problema computacional bem especificado, de maneira que o enunciado do problema deve descrever em termos gerais a relação esperada entre a entrada e a saída. O algoritmo, então, deve descrever um procedimento específico para alcançar esta relação entre entrada e saída (CORMEN *et al.*, 2009).

Uma **instância** de um problema é definida como uma entrada que satisfaça os requerimentos para construir uma solução para o problema em questão. Um algoritmo é definido como **correto** se, para cada instância de entrada, este retornará uma saída esperadamente correta. Costumeiramente é dito que um algoritmo resolveu um determinado problema computacional. Um algoritmo também pode não resolver um problema para determinadas instâncias, sendo incorreto por não prover respostas corretas sempre, mas não inútil, já que outras variáveis podem ser mais importantes para uma determinada situação (CORMEN *et al.*, 2009). Por exemplo, instâncias de magnitudes elevadas podem resultar em respostas incorretas do algoritmo, mas estas serem irrelevantes para uma aplicação que apenas precisam de instâncias de magnitudes menores.

2.2 HEURÍSTICAS

Heurísticas são definidas como critérios, métodos ou princípios para decidir quais sequências de ações serão efetivas para atingir um objetivo. Muitos problemas complexos requerem a avaliação de uma série de possibilidades para atingir uma solução exata.

Entretanto, o tempo requerido para encontrar uma solução exata pode atingir um tempo significativo, como décadas ou centenas de anos. Desta forma, para problemas desta magnitude de complexidade, heurísticas possuem o papel de indicar uma forma de reduzir o número de procedimentos e avaliações a fim de obter soluções com limites de tempos razoáveis (PEARL, 1984). O fato de heurísticas entregarem um caminho para a resolução de problemas, de maneira a não considerar todas as alternativas, implica na possibilidade de não obter resultados ótimos, mas apenas razoáveis no intervalo de tempo adotado para a execução dos procedimentos determinados pela heurística.

Assim, heurísticas realizam buscas no espaço de solução de um problema, com uma sequência de passos que são determinados por um certo critério, mas este não garante a sua otimalidade. A construção de uma heurística frequentemente é realizada a partir de palpites refinados para o problema, geralmente parecidos com táticas gulosas de resolução de problemas. Heurísticas podem ser dividas em dois tipos: heurísticas construtivas e heurísticas de refinamento.

2.2.1 Heurísticas Construtivas

Uma heurística construtiva tem como objetivo a construção de uma solução, elemento por elemento. A maneira pela qual cada elemento será inserido a cada etapa do procedimento irá variar de acordo com a função de avaliação específica ao problema abordado. Em métodos clássicos de heurísticas de construção, os elementos candidatos são geralmente ordenados seguindo uma função gulosa, de forma a selecionar o “melhor” elemento a cada etapa, sem se importar com efeitos desta escolha nas próximas etapas (SOUZA, 2022).

Outra forma comum de construir uma solução inicial é a seleção aleatória de elementos candidatos. Então, ao contrário da seleção gulosa, cada elemento inserido na solução final não é o “melhor”, mas pode ser qualquer candidato dentre o conjunto de elementos ainda não selecionados. Estas são mais fáceis de serem desenvolvidas, devido a

sua simplicidade de implementação, mas tendem a produzir soluções de baixa qualidade que podem exigir mais esforço computacional em etapas de refinamento a seguir.

2.2.2 Heurísticas de Refinamento

As heurísticas de refinamento - também chamadas de heurísticas de busca local - são uma família de técnicas baseadas na noção de vizinhança. Esta classe de heurísticas parte de uma solução inicial qualquer, gerada por uma heurística construtiva, e caminha, de vizinho a vizinho, de acordo com a definição de vizinhança determinada.

Um método de busca local pode ser interpretado como um procedimento que percorre um caminho em um grafo não-orientado $G = (S, E)$, no qual S representa o conjunto de soluções s do problema e E o conjunto de arestas (s, s') com $s' \in N(s)$ (HERTZ; WIDMER, 2003).

A definição de vizinhança é importantíssima para uma heurística de refinamento, de maneira que, a partir de cada solução s do espaço de soluções, sempre deve ser possível atingir qualquer outra solução viável em um número finito de etapas.

Entretanto, em muitos problemas combinatórios é difícil até mesmo encontrar uma solução viável. Nessas situações, pode ser prejudicial caminhar apenas no espaço das soluções viáveis do problema considerado. Nestes problemas, o espaço de busca pode incluir soluções inviáveis, obtidas a partir de um relaxamento de certas restrições do problema original. Desta forma, bastaria acrescentar à função objetivo (a que avaliaria a solução) componentes que penalizam violações às restrições (SOUZA, 2022).

Um exemplo comum no qual é difícil de encontrar uma solução viável é o Problema de Programação de Horários de Cursos Universitários (BABAEI; KARIMPOUR; HADIDI, 2015). A restrição principal deste problema exige que as aulas oferecidas pelo mesmo professor para diferentes turmas não sejam realizadas no mesmo horário (ou seja, não se sobreponham), visto que este não pode estar em mais de um lugar ao mesmo tempo. Se a mudança das aulas de um curso de um horário para outro for considerado como movimento m, seria improvável não gerarmos quadros de horários sem sobreposições de aulas. Para isso, seria possível relaxar estas restrições, penalizando-as (com algum valor de alta magnitude) na função de avaliação, de maneira que a exploração do espaço de busca será mais eficiente (HERTZ, 1992).

2.3 META-HEURÍSTICAS

Meta-heurísticas, assim como as heurísticas, são métodos de aproximação para encontrar uma boa solução em um tempo razoável.As heurísticas costumam ter soluções específicas em relação aos requisitos do problema. Já as meta-heurísticas possuem a capacidade de produzir uma estrutura genérica para aplicar a diversos problemas de otimização (ABDEL-BASSET; ABDEL-FATAH; SANGAIAH, 2018).

O termo "meta-heurística" foi cunhado para ilustrar um método heurístico que não possui características específicas de um problema (GLOVER, 1986). As meta-heurísticas são munidas de mecanismos para evitar o aprisionamento de soluções em ótimos locais possivelmente distantes dos ótimos globais. Esta classe de procedimentos se divide em duas categorias: busca por trajetória e busca populacional, dependendo com o princípio utilizado para realizar a exploração do espaço de busca (SOUZA, 2022).

Nas meta-heurísticas baseadas em busca por trajetória, o espaço de soluções é explorado por meio de movimentos aplicados a atual solução, de maneira a gerar soluções promissoras em vizinhanças. Este tipo de meta-heurística também é chamado de busca em trajetória. *Iterated Local Search* (ILS), *Simulated Annealing* (SA) e *Tabu Search* (TS) são exemplos desta categoria.

Já os métodos baseados em busca populacional buscam manter conjuntos de soluções boas o suficiente para combiná-las e produzir melhores soluções. Algoritmo Genético (AG), *Memetic Algorithm* (MA) e *Ant Colony Optimization* (ACO) são exemplos deste tipo de procedimento, indicando uma forte inspiração de processos biológicos a fim de construir resultados interessantes.

Para ilustrar ainda mais as diferenças entre estes dois tipos de meta-heurísticas, podemos apresentar uma meta-heurística de cada referenciada na literatura.

2.3.1 *Iterated Local Search*

O método *Iterated Local Search* (ILS) é baseado na ideia de um procedimento de busca local que consegue ser melhorado ao gerar novas soluções de partida a partir de métodos perturbativos na solução de ótimo local.

Ao aplicar o procedimento de ILS, quatro componentes devem ser especificados:

- Procedimento **GeraSoluçãoInicial()**: gera-se uma solução inicial para o problema;
- Procedimento **BuscaLocal()**: retorna-se uma solução possivelmente melhorada para

a solução corrente;

- Procedimento **Perturbação()**: modifica a solução corrente ao guiá-la a uma solução intermediária;
- Procedimento **CritérioAceitação()**: escolhe qual solução sofrerá determinada perturbação;

Algoritmo 1: Procedimento de um *Iterated Local Search* (ILS)

Procedimento *ILS()*
*s*0 ← *GeraSoluçãoInicial()*;
s ← *BuscaLocal(s0)*;
repita
  s′ ← *Perturbação(histórico, s)*;
  s″ ← *BuscaLocal(s')*;
  s ← *CritérioAceitação(s, s", histórico)*;
até *(critérios de parada serem satisfeitos)*;
Retorne s;

2.3.2 Programação Genética

Algoritmo Genético (AG) é uma meta-heurística inspirada pelo processo biológico de evolução (MICHALEWICZ; SCHOENAUER, 1996). Proposto por (HOLLAND, 1992), AGs são inspirados na Teoria Darwiniana de Seleção Natural (DARWIN, 1859). Os elementos básicos de uma AG são representações em cromossomos, seleção de aptidão (*fitness*) e outros operadores inspirados na biologia, como mutações e *crossover*. O Algoritmo 2 demonstra a sequência de decisões de um algoritmo genético.

Os cromossomos são definidos como uma sequência de números binários. Cada elemento (também chamado de *locus*) possui diversos valores possíveis (geralmente definidos como 0 ou 1), sendo chamados de alelos, representando uma variedade genética. Um cromossomo pode significar diversas abstrações, como uma possível escolha de pontos dentro de uma solução. Uma função de *fitness* é utilizada para avaliar o cromossomo dependendo de um critério.

Os operadores inspirados pela biologia são a seleção, mutação e cruzamento, sendo este último chamado comumente de *crossover*. Na seleção, os cromossomos são selecionados com base no valor retornado pela função de *fitness*. Na operação de *crossover*, um *locus* é escolhido aleatoriamente e é realizada uma troca entre subsequências entre cromossomos, a fim de criar uma nova geração de cromossomos. Por fim, na mutação,

alguns *locus* (comumente abstraídos por *bits*) são invertidos de valor aleatoriamente baseada em uma probabilidade escolhida (KATOCH; CHAUHAN; KUMAR, 2020).

Algoritmo 2: Procedimento de um Algoritmo Genético (AG)

Procedimento *AG()*
$t \leftarrow 0$;
Seja $P(t)$ a população inicial;
Avalie $P(T)$;
repita
  $t \leftarrow t + 1$;
  Gere $P(t)$ a partir de $P(t-1)$;
  Avalie $P(t)$;
  Defina a população sobrevivente;
até *(critérios de parada serem satisfeitos)*;
Retorne $P(t)$

2.4 HIPER-HEURÍSTICAS

Hiper-heurísticas são um conjunto de abordagens motivadas, principalmente, pela meta de automatizar a construção de métodos heurísticos para resolver problemas de busca computacional considerados difíceis. A ideia, aqui, é buscar soluções mais genéricas possíveis, focando em uma classe de problemas, a fim de apenas alguns parâmetros serem alterados ao aplicar em diversos setores.

A principal diferença de uma hiper-heurística para heurísticas ou meta-heurísticas é que, aqui, o objetivo é trabalhar na busca dentro de um espaço de busca de heurísticas ou componentes de heurísticas, ao invés de apenas operar no espaço de busca das soluções dos problemas adereçados.

O termo hiper-heurística é relativamente novo, e foi utilizado primeiramente nos anos 2000 para descrever heurísticas que selecionam outras heurísticas com a finalidade de resolver problemas de otimização combinatória. Entretanto, a ideia de automatizar a construção de heurísticas não é necessariamente nova, mas foi estendida recentemente para referir-se a um método de pesquisa ou mecanismo de aprendizado para selecionar ou gerar heurísticas a fim de resolver problemas de busca computacional.

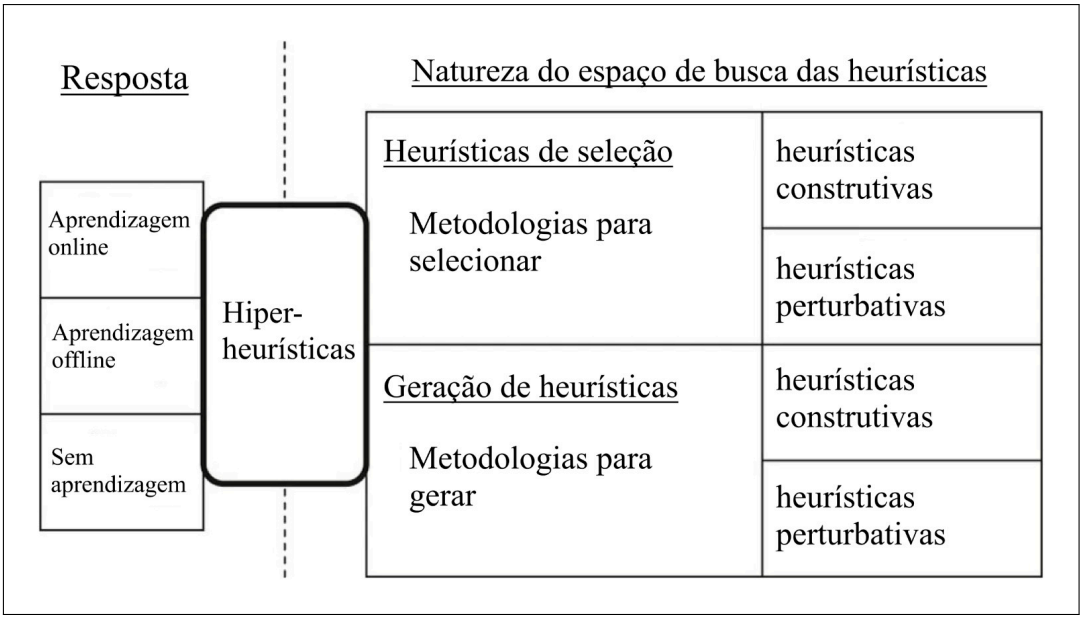
Burke *et al.* (2013) definem duas categorias de hiper-heurísticas: as **hiper-heurísticas de seleção** e as **hiper-heurísticas de geração**. A Figura 1 ilustra essa classificação com mais detalhes, considerando a natureza do espaço de busca das heurísticas

e as diferentes fontes de informação de resposta ou retorno.

Em outras palavras, uma hiper-heurística é um algoritmo de aprendizado de máquina quando usa um retorno no processo de busca. A partir do tipo de busca durante o aprendizado, pode-se distinguir entre aprendizagem *online* e aprendizagem *offline*. Em heurísticas de aprendizagem *online*, o aprendizado ocorre enquanto um algoritmo está resolvendo uma instância de um problema. Já em heurísticas de aprendizagem *offline*, a ideia é receber informações de um conjunto de instâncias e, em seguida, tentar generalizar para instâncias não vistas anteriormente. Por fim, também existem as hiper-heurísticas que não possuem nenhum aprendizado durante o processo de busca.

Dentro das metodologias para selecionar heurísticas, diferentes formas de selecionar heurísticas de construção e heurísticas de perturbação podem ser elaboradas. Os métodos de construção são responsáveis por prover uma solução inicial factível para o problema para, então, os métodos de pertubação ou de busca local iterarem sobre aspectos desta solução a fim de melhorá-los dada uma função objetivo.

Figura 1 – Classificação de hiper-heurísticas.



FONTE: (BURKE *et al.*, 2013). Tradução livre.

2.4.1 Hiper-heurísticas de seleção

As hiper-heurísticas de seleção são metodologias para selecionar heurísticas iterativamente a fim de construir uma solução factível (BURKE *et al.*, 2013). Diversas abordagens foram propostas, na literatura, para a construção de soluções iniciais ótimas,

utilizando heurísticas construtivas.

2.4.2 Hiper-heurísticas de geração

As hiper-heurísticas de geração possuem como objetivo a criação de heurísticas ou meta-heurísticas novas, a partir de métodos já existentes. Ao contrário das hiper-heurísticas de seleção, as hiper-heurísticas de geração possuem uma heurística nova como saída proposta para determinada solução, com procedimentos a serem realizados. Assim, esta heurística pode ser potencialmente reutilizada em instâncias novas do problema.

A Programação Genética (KOZA, 1994) é uma técnica de computação evolutiva que se destaca na literatura como metologia de gerar heurísticas. A população de programas, na Programação Genética, são heurísticas ou composições de heurísticas, nas quais evoluem para uma sequência de etapas a serem realizadas e produzirem uma solução viável.

3 REVISÃO DA LITERATURA

Este capítulo visa revisitar a literatura ao passar por trabalhos que abordam o APPP por meio de métodos heurísticos.

Matematicamente, o APPP pode ser modelado como um problema de Programação Linear Inteira e é baseado no problema de localização de facilidades. Megiddo e Tamir (1982) provam que o problema é NP-Difícil, sendo pelo menos tão difíceis quanto os problemas mais difíceis em NP, quando é proposto minimizar os custos dos APs e penalizando a cobertura de pontos proibidos da área externa.

Pelo fato do APPP ser definido como NP-Difícil, não é possível garantir que a solução seja encontrada em tempo polinomial, então surgem alternativas de heurísticas para resolução do problema em tempo viável, de maneira a obterem resultados razoáveis quando comparados com métodos exatos.

Diversos trabalhos tratam sobre tratam sobre a resolução do APPP. Entretanto, a definição do problema tem uma alta variabilidade, visto que os métodos precisam ser adequados aos projetos reais. Isto proporciona uma dificuldade na estruturação das pesquisas desta área. Sendo assim, o foco deste trabalho é tratar de métodos heurísticos para o tratamento do problema.

Kamenetsky e Unbehaun (2002) propuseram uma combinação de duas abordagens utilizando um algoritmo de remoções sucessivas chamado *Pruning* para construir uma solução inicial viável. Em seguida, melhorias são realizadas a partir desta solução utilizando a meta-heurística *Simulated Annealing* (SA). A fim de avaliar o método proposto, uma função objetivo foi desenvolvida em um espaço discreto, de maneira a considerar a área de cobertura e qualidade do sinal obtido.

Arroyo (2006) propuseram dois métodos baseados na meta-heurística *Greedy Randomized Adaptive Search Procedures* (GRASP)(RESENDE; RIBEIRO, 2018) para resolver o problema de alocação de antenas de transmissão em uma região. Nesse trabalho, a função objetivo do modelo buscou maximizar a cobertura do sinal ao utilizar um número mínimo de antenas e instalando-as o mais próximo possível dos pontos de demanda, de maneira a levar em consideração as restrições de alcance do sinal de transmissão e a presença de obstáculos.

Lu *et al.* (2006) propuseram um método baseado na meta-heurística *Tabu Search* (TS) (GLOVER, 1989) para a definição da quantidade de APs e o seu posicionamento em

relação aos critérios de cobertura, interferências e qualidade de serviço. Uma cadeia de Markov é utilizada para avaliar a largura de banda em cada célula da área. Por fim, a cobertura é estimada por um simulador de propagação de rádio de muti-resolução.

Vanhatupa, Hannikainen e Hamalainen (2007) apresentam um novo algoritmo baseado em AG para o desenvolvimento de redes WLAN de acordo com os requisitos desejados, além de realizar uma estimativa da qualidade de serviço dividida entre: cobertura, custos de implantação e capacidade da rede. Assim, essas estimativas fornecem um retorno (*feedback*) para o próprio algoritmo e o construtor da rede. Outros requisitos são levados em conta neste método, como o consumo de energia, possíveis pontos de posicionamento de APs e quantidade de APs permitidos.

Farkas K. (2013) propuseram um método baseado em SA com o objetivo de encontrar o posicionamento e o numero ideal de pontos de acesso Wi-Fi de tal forma que cada ponto na área seja coberto por pelo menos 3 APs.

Uma meta-heurística baseada no *Iterated Local Search* (ILS) (LOURENÇO; MARTIN; STüTZLE, 2010) foi proposta por (RUFINO A.; SILVA, 2016) para resolver instâncias sintéticas e reais das mais diversas dimensões. Para avaliar o método proposto, experimentos computacionais envolvendo adaptações de meta-heurísticas da literatura foram realizados e comparados com os demais resultados. Pelo nosso conhecimento, estes são os melhores resultados até esta presente data.

4 METODOLOGIA

Este capítulo apresenta a formulação matemática utilizada para definir a função objetivo que norteará os resultados de comparação com outros métodos. Além disso, a hiper-heurística desenvolvida é explicada em detalhes, bem como as instâncias de teste a que está será submetida a posteori.

4.1 FORMULAÇÃO MATEMÁTICA UTILIZADA

Nesta subseção há a formalização do APPP através de um modelo de programação linear inteira. A fim de realizar a discretização do problema, é considerada uma grade de pontos candidatos sobre a área. Dentre os pontos definidos, P é o conjunto de todos os pontos a serem cobertos, R é o conjunto de pontos onde é possível alocar um AP, T é o conjunto de diferentes tipos de APs e S é o conjunto dos pontos proibidos por serem incluídos na área externa.

Definindo p_k como o custo de utilizar o roteador do tipo $k \in T$; q_s a penalidade pelo ponto proibido s estar na área de cobertura de pelo menos um roteador ativo; c_k a capacidade do roteador do tipo k e d_i a demanda por largura de banda do i-ésimo ponto. A variável de decisão r_{jk} assume o valor 1 se, e somente se, o roteador j do tipo k está sendo escolhido para ser utilizado na solução. A variável de decisão s_h assume o valor 1 se, e somente se, o ponto proibido h está inserido na área de cobertura de algum roteador utilizado. Seja $cob(r_{jk})$ uma função cujo retorno são todos os pontos inseridos na área de cobertura do roteador r_{jk}, e $cob(r_{jk})$ uma função de mesma motivação para os pontos proibidos, a formulação matemática proposta é apresentada nas expressões matemáticas 3.1 à 3.6.

$$min \sum_{j \in R} \sum_{k \in T} p_k r_{jk} + \sum_{h \in S} q_h s_h \tag{4.1}$$

$$s.a. \sum_{j \in R} \sum_{k \in T} x_{ijk} = 1, \qquad \forall i \in P, \tag{4.2}$$

$$r_{jk} \geq x_{ijk}, \qquad \forall i \in cob(r_{jk}), \forall j \in R, \forall k \in T, \tag{4.3}$$

$$\sum_{i \in P} d_i x_{ijk} \leq c_k, \qquad \forall j \in R, \forall k \in T \tag{4.4}$$

$$s_h \geq r_{jk}, \qquad \forall h \in cobp(r_{jk}), \forall j \in R, \forall k \in T, \tag{4.5}$$

$$r_{ik}, x_{ijk}, s_h \in \{0, 1\}, \qquad \forall i \in P, \forall j \in R, \forall k \in T, \forall h \in S. \tag{4.6}$$

A função objetivo (4.1) procura minimizar o custo total da utilização dos roteadores, bem como busca minimizar a penalização de cobertura de pontos proibidos. As restrições (4.2) estabelecem que todos os pontos clientes estejam associados a um roteador. As restrições (4.3) definem que cada ponto apenas pode ser conectado a um roteador ativo. As restrições (4.4) definem que a demanda por largura de banda dos pontos conectados ao k-ésimo roteador não pode ultrapassar a capacidade deste. A restrição (4.5) indica se o ponto proibido s_h está dentro da área de cobertura de algum roteador. Enfim, as restrições (4.6) definem o domínio das variáveis de decisão, sendo este domínio binário.

4.2 HIPER-HEURÍSTICA DE GERAÇÃO PROPOSTA

Nesta seção será apresentada a proposta de hiper-heurística de geração de heurísticas. Inicialmente apresentamos as heurísticas construtivas e heurísticas de busca escolhidas para compôr as heurísticas a serem construídas. Além disso, o procedimento da hiper-heurística é exposto, bem como os operadores de *crossover* e mutação são definidos.

4.2.1 Heurísticas de construção selecionadas

Nesta subseção, as duas heurísticas de construção escolhidas são descritas. A primeira de heurística de construção é uma geração gulosa, na qual cada roteador é posicionado de forma a atingir a maior quantidade possível de pontos clientes cobertos dentro do raio de cobertura deste roteador. Se houverem empates, estes são quebrados com a mesma probabilidade para os pontos. Aqui, como a capacidade de largura de banda não é levada em consideração, a função objetivo penalizará a solução construída ao ter um alto valor final.

A segunda heurística de construção é uma geração aleatória de roteadores em pontos aleatórios. Neste caso, são associados pontos a roteadores aleatórios, bem como tipos de roteadores aleatórios, de maneira a construir uma solução passível de alta penalidade pela função objetivo proposta. A seleção de roteadores e tipos de roteadores é realizada de maneira equiprovável.

4.2.2 Heurísticas de busca local selecionadas

Nesta subseção, as heurísticas de busca local escolhidas são descritas. Foram utilizadas 5 heurísticas propostas por (RUFINO A.; SILVA, 2016), sendo expostas a seguir.
AllChangeLink(): Verifica-se, para todos os pontos clientes da solução que estão conectados por algum roteador, se este último está com a capacidade de banda passante acima do limite. Se for o caso, será realizada uma tentativa de reconectar o ponto corrente para qualquer outro roteador que ainda não tenha sua capacidade máxima de banda no momento. O Algoritmo 3 detalha a heurística proposta.

Algoritmo 3: Heurística de *AllChangeLink()*

```
Procedimento AllChangeLink()
    Seja PontosClientes ← conjunto de pontos clientes da solução que estão conectados
     por algum roteador;
     Seja RoteadoresDisponiveis ← conjunto de roteadores que não possuam sua
     capacidade máxima de banda;
     para cada ponto ∈ PontosClientes faça
        Seja roteador ← roteador do ponto cliente;
         Seja capacidadeDeBanda ← a capacidade de banda passante do roteador;
         Seja limiteDePassagem ← o limite de capacidade banda passante do roteador;
         Seja demanda ← a demanda do ponto por capacidade de banda passante;
         if capacidadeDeBanda > limiteDePassagem then
            para cada roteadorVizinho ∈RoteadoresDisponiveis faça
                Seja capacidadeBandaVizinho ← a capacidade de banda do
                 roteadorVizinho;
                 Seja limiteDePassagemVizinho ← o limite de capacidade de banda
                 passante do roteadorVizinho;
                if capacidadeBandaVizinho + demanda ≤ limitePassagemVizinho
                 then
                    Associe ponto ao roteadorVizinho;
                end
            fim
        end
    fim
end
```

AllDryRouter(): Como demonstrado no Algoritmo 4, nesta heurística, procura-se por

roteadores ativos que estejam com uma exigência de banda passante acima do limite suportado. Após o processo de identificação destes roteadores, distribui-se todos os pontos conectados ao atual roteador para outros roteadores ativos nos quais ainda não estão na sua capacidade máxima de banda.

Algoritmo 4: Heurística de *AllDryRouter()*

```
Procedimento AllDryRouter()
    Seja RoteadoresAcimaDoLimite ← conjunto de roteadores ativos que estejam com
     uma exigência de banda passante acima do limite suportado;
     Seja RoteadoresAbaixoDoLimite ← conjunto de roteadores ativos que estejam
     com uma exigência de banda passante abaixo do limite suportado;
     para cada roteador ∈ roteadoresAcimaDoLimite faça
        Seja pontosDoRoteador ← conjunto de pontos cliente associados ao roteador;
         para cada ponto ∈ pontosDoRoteador faça
            Seja demandaPonto ← demanda do ponto por capacidade de banda
             passante;
             para cada roteadorVizinho ∈ roteadoresAbaixoDoLimite faça
                Seja capacidadeRoteadorVizinho ← a capacidade do roteadorVizinho;
                 Seja limiteRoteadorVizinho ← limite de banda passante associado ao
                 roteadorVizinho;
                 if capacidadeRoteadorVizinho + demandaPonto ≤
                 limiteRoteadorVizinho then
                    Associe ponto ao roteadorVizinho;
                end
            fim
        fim
    fim
end
```

BestAddRouter(): Após obter os pontos clientes ainda não conectados a nenhum roteador da solução, procura-se a posição na qual, posicionando um roteador, este atenderá a maior quantidade possível de pontos clientes não conectados. Ao encontrar tal posição, um roteador é incrementado à solução e todos os pontos clientes dentro do raio de cobertura do roteador serão conectados ao mesmo, desde que estes não ultrapassem suas respectivas capacidades. O Algoritmo 5 detalha a heurística proposta.

AllBestRadiusType(): Analisa todos os roteadores da solução. Se um roteador possuir algum ponto cliente conectado a uma demanda por qualidade de sinal acima ou abaixo da necessária, este roteador será substituído por outro com um raio de cobertura menor ou maior, se disponível, com o objetivo de suprir a demanda por qualidade de sinal do ponto cliente com eficiência. O Algoritmo 6 detalha a heurística proposta.

Algoritmo 5: Heurística de *BestAddRouter()*

Procedimento *BestAddRouter()*
Seja *pontosNãoConectados* ← conjunto de pontos clientes ainda não conectados a nenhum roteador da solução;
Seja *pontoDeMaiorCobertura* ← ponto de maior cobertura de pontos clientes não conectados;
Seja *roteadorAleatorio* ← roteador aleatório que possua capacidade para a demanda que o ponto necessita;
Adicione *roteadorAleatorio* ao *pontoDeMaiorCobertura*;
end

Algoritmo 6: Heurística de *AllBestRadiusType()*

Procedimento *AllBestRadiusType()*
Seja *roteadoresAssociados* ← conjunto de roteadores associados a um ponto cliente na solução;
para cada *roteador* ∈ *roteadoresAssociados* **faça**
Seja *pontosConectados* ← conjunto de pontos clientes ainda já conectados ao roteador;
para cada *ponto* ∈ *pontosConectados* **faça**
if *ponto não está sendo satisfeito pela qualidade de sinal* **then**
Procure por um roteador de raio maior e associe a este ponto;
end
if *ponto está está com uma qualidade de sinal acima do necessário* **then**
Procure por um roteador de raio menor e associe a este ponto;
end
fim
fim
end

BestLink(): Verifica-se todos os pontos clientes ainda não conectados à corrente solução. Se o atual ponto cliente está dentro do raio de cobertura de um roteador ativo e este possui uma quantidade de banda suficiente para receber o ponto cliente, o procedimento fará a ligação do ponto ao roteador em questão. O Algoritmo 7 detalha a heurística proposta.

Além destas, mais 2 heurísticas de busca local são propostas neste trabalho, sendo detalhadas a seguir:

AllBestCapacityType(): Analisa-se todos os roteadores da solução com o objetivo de adicionar o tipo de roteador com melhor capacidade para os pontos clientes. Se ainda há possibilidade de trocar um tipo de roteador com maior ou menor capacidade em relação ao roteador corrente, este é adicionado. O Algoritmo 8 detalha a heurística proposta.

AllClosestMoveRouter(): Procura-se mover os roteadores da solução para os pontos mais próximos possíveis que não sejam proibidos ou possuam outro roteador. Se existir mais de um ponto possível para a locomoção do roteador, mova-o para o ponto que cobrirá

Algoritmo 7: Heurística de *BestLink()*

```
Procedimento BestLink()
    Seja pontosNãoConectados ← conjunto de pontos clientes ainda não conectados a
     nenhum roteador da solução;
     Seja roteadoresAssociados ← conjunto de roteadores associados a um ponto cliente
     na solução;
     para cada ponto ∈ pontosNãoConectados faça
         para cada roteador ∈ roteadoresAssociados faça
             Seja pontosDentroDoRaio pontos dentro do raio do roteador associado;
              if ponto ∈ pontosDentroDoRaio then
                 Associe o roteador ao ponto;
              end
         fim
    fim
end
```

Algoritmo 8: Heurística de *AllBestCapacityType()*

```
Procedimento AllBestCapacityType()
    Seja roteadoresAssociados ← conjunto de roteadores associados a um ponto cliente
     na solução;
     para cada roteador ∈ roteadoresAssociados faça
         Seja capacidadeUtilizada ← a capacidade utilizada do roteador no momento;
          Seja capacidadeTotal ← a capacidade total do roteador;
          if capacidadeUtilizada > capacidadeTotal then
             Associe todos os pontos deste roteador a um roteador com capacidade maior;
         end
         if capacidadeUtilizada < capacidadeTotal then
             Associe todos os pontos deste roteador a um roteador com capacidade menor;
         end
    fim
end
```

mais pontos. O Algoritmo 9 detalha a heurística proposta.

4.2.3 Estrutura da hiper-heurística

A hiper-heurística de construção desenvolvida realiza buscas no espaço das heurísticas de maneira *online*. Ou seja, a aprendizagem da hiper-heurística ocorre no *feedback* em tempo de execução do programa. Além disso, a estrutura é baseada em um AG, como se descreve no Algoritmo 10.

Algoritmo 9: Heurística de *AllClosestMoveRouter()*

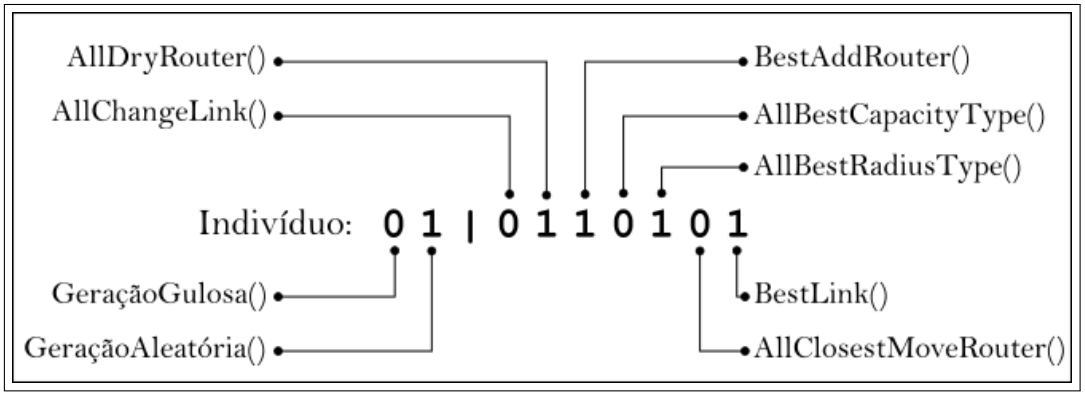
Procedimento *AllClosestMoveRouter()*
Seja $roteadoresAssociados \leftarrow$ conjunto de roteadores associados a um ponto cliente na solução;
para cada $roteador \in roteadoresAssociados$ **faça**
Seja $pontosNãoConectados \leftarrow$ conjunto de pontos clientes não conectados a um roteador;
Seja $pontosMaisProximos \leftarrow$ conjunto de pontos mais próximos pertencentes a $pontosNãoConectados$;
Seja $pontoMaiorCobertura \leftarrow$ ponto de maior cobertura pertencente a $pontosMaisProximos$;
Mova $roteador$ para o ponto $pontoMaiorCobertura$;
fim
end

Algoritmo 10: Hiper-heurística de construção

Procedimento Hiper-heurística()
Seja $t \leftarrow 0$;
Seja $P(t)$ a população inicial com 14 soluções;
Avalie $P(t)$ de acordo com o *fitness*;
repita
$t \leftarrow t + 1$;
Gere $P(t)$ a partir de $P(t-1)$, utilizando *Crossover* e mutação;
Avalie $P(t)$ de acordo com o *fitness*;
Defina a população sobrevivente a partir das melhores soluções;
até *tempo limite ser excedido*;
Retorne $P(t)$;
end

Cada cromossomo da população é uma solução, representada por uma sequência de 9 *bits*. Os 2 primeiros *bits* são as heurísticas de construção utilizadas na solução, enquanto os demais são as heurísticas de busca local escolhidas. Isto representa que, para cada *bit* (também chamado de *locus* pela literatura) ativado (possui valor 1) nesta sequência, a i-ésima heurística foi utilizada para construir a solução. A Figura 2 ilustra a composição de bits de uma solução. Neste exemplo, a segunda heurística de construção foi escolhida para compor a solução. Bem como as segundas, terceiras, quintas e sétimas heurísticas de busca locais foram utilizadas. Por conseguinte, também haverá o valor da função objetivo (descrita em 4.1) alcançada pela solução.

Figura 2 – Sequência de *bits* de uma solução.



FONTE: autoria própria.

Durante a fase de criação da população inicial de soluções são geradas exatamente 14 soluções. Isto se deve pois utilizamos 2 heurísticas de construção e 7 heurísticas de busca local, e cada solução desta população inicial possui exatamente 1 heurística de construção e 1 heurística de busca local, sendo 14 a quantidade total destas diferentes combinações.

Enquanto o tempo limite estipulado não é excedido, geram-se novas gerações de soluções. É realizado, para cada par de solução da geração atual, o *crossover* entre os pares. É gerado, para cada par que participa do *crossover*, mais um par de soluções. Estas soluções passam por um processo de mutação. Ao final de cada geração, 50% das soluções são descartadas, sendo estas as que pior obtiveram uma função objetivo interessante.

4.2.3.1 Método de *crossover*

O método de *crossover* escolhido foi descrito no Algoritmo 11. Tomando as soluções A e B como o par de soluções a serem realizados o *crossover*. Também considerando que C e D sejam as soluções resultantes do *crossover* do par recebido. Compara-se cada i-ésimo *bit* da solução A, e o j-ésimo *bit* da solução B. Se o valor de i for diferente do valor de j, o k-ésimo e o l-ésimo *bit* de C e D serão 1 ou 0, com 50% de probabilidade para cada. Se i-ésimo e o j-ésimo *bit* de A e B forem iguais, C e D serão este valor, mas poderão ser o inverso com 2% de probabilidade.

Algoritmo 11: Critério de *crossover*

Procedimento *Crossover* (A, B)
Seja $A \leftarrow$ a sequência de *bits* da primeira solução;
Seja $B \leftarrow$ a sequência de *bits* da segunda solução;
Seja $C \leftarrow \emptyset$ a sequência de *bits* do primeiro filho sem nenhuma heurística escolhida;
Seja $D \leftarrow \emptyset$ a sequência de *bits* do segundo filho sem nenhuma heurística escolhida;
para cada $i \in A$, $j \in B$, $k \in C$, $l \in D$ **faça**
if $i \neq j$ **then**
Faça $k \leftarrow 1$ *com 50% de probabilidade;*
Faça $l \leftarrow 1$ *com 50% de probabilidade;*
end
if $i = j$ **then**
Faça $k \leftarrow 1 - a$ *com 2% de probabilidade;*
Faça $l \leftarrow 1 - b$ *com 2% de probabilidade;*
end
fim
Retorne $\{C, D\}$;
end

4.2.3.2 Método de mutação

Após a realização de um *crossover*, cada solução gerada passa por um procedimento de mutação, descrito no Algoritmo 12. Aqui, cada *bit* da solução passa por uma chance de 2% de ser substituído pelo seu valor oposto, de maneira que pode se tornar 1 ou 0. Isto nos permite criar uma maior diversidade de soluções e não nos prender em ótimos locais.

Algoritmo 12: Critério de mutação

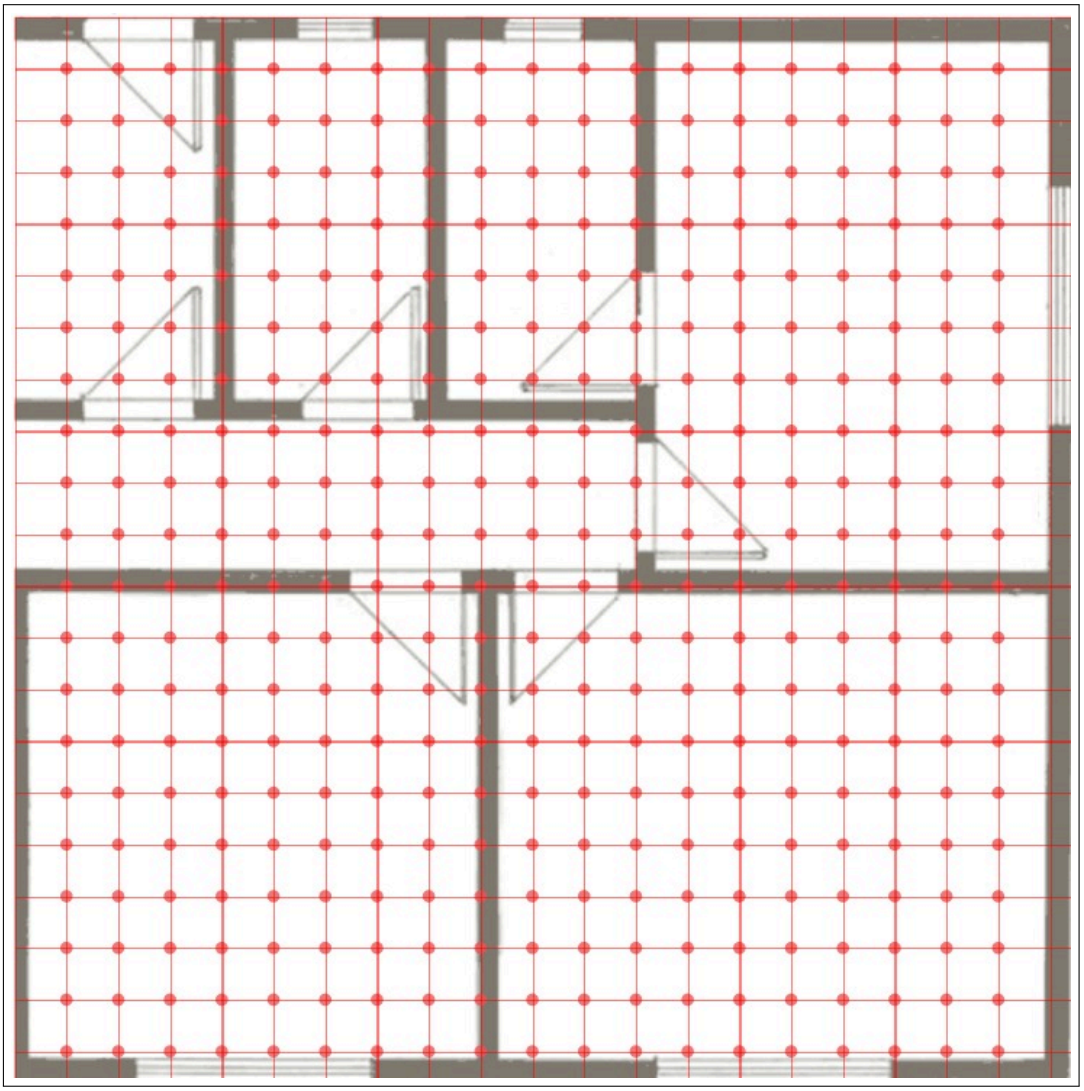
Procedimento Mutação (A)
Seja $A \leftarrow$ a sequência de *bits* desta solução;
para cada $i \in A$ **faça**
Faça $i \leftarrow 1 - i$ com 2% de probabilidade;
fim
Retorne A;
end

4.3 INSTÂNCIAS DE TESTE UTILIZADAS

Foram utilizadas instâncias criadas por (RUFINO A.; SILVA, 2016), já que estas atendem perfeitamente a formulação matemática do problema. A ideia por trás da geração destas instâncias gira em torno da discretização do espaço - no nosso caso, plantas baixas - em uma grade de pontos, a fim de melhor acomodar-se à formulação do

problema. Cada um destes pontos é posicionado horizontal e verticalmente (90º entre si) há exatamente um metro de distância. A Figura 3 exemplifica esta discretização ao ilustrar pontos de acesso em uma planta baixa.

Figura 3 – Exemplo de uma discretização de uma planta-baixa.



FONTE: Autoria própria.

O modelo de propagação utilizado para a geração de instâncias foi baseado nos estudos de (NAKPRASIT; PHONGCHAROENPANICH, 2018) e (ANDRONE; PALADE, 2010), nos quais abordam o comportamento do sinal das redes Wi-Fi na frequência de 2,4 GHz. A Tabela 1 descreve os valores de atenuação utilizados nas instâncias geradas.

Cada instância possui as seguintes informações:

- Tr: o número de tipos diferentes de roteadores a serem considerados, de maneira que cada roteador possui as seguintes características:

Tabela 1 – Modelo de atenuação para determinados anteparos.

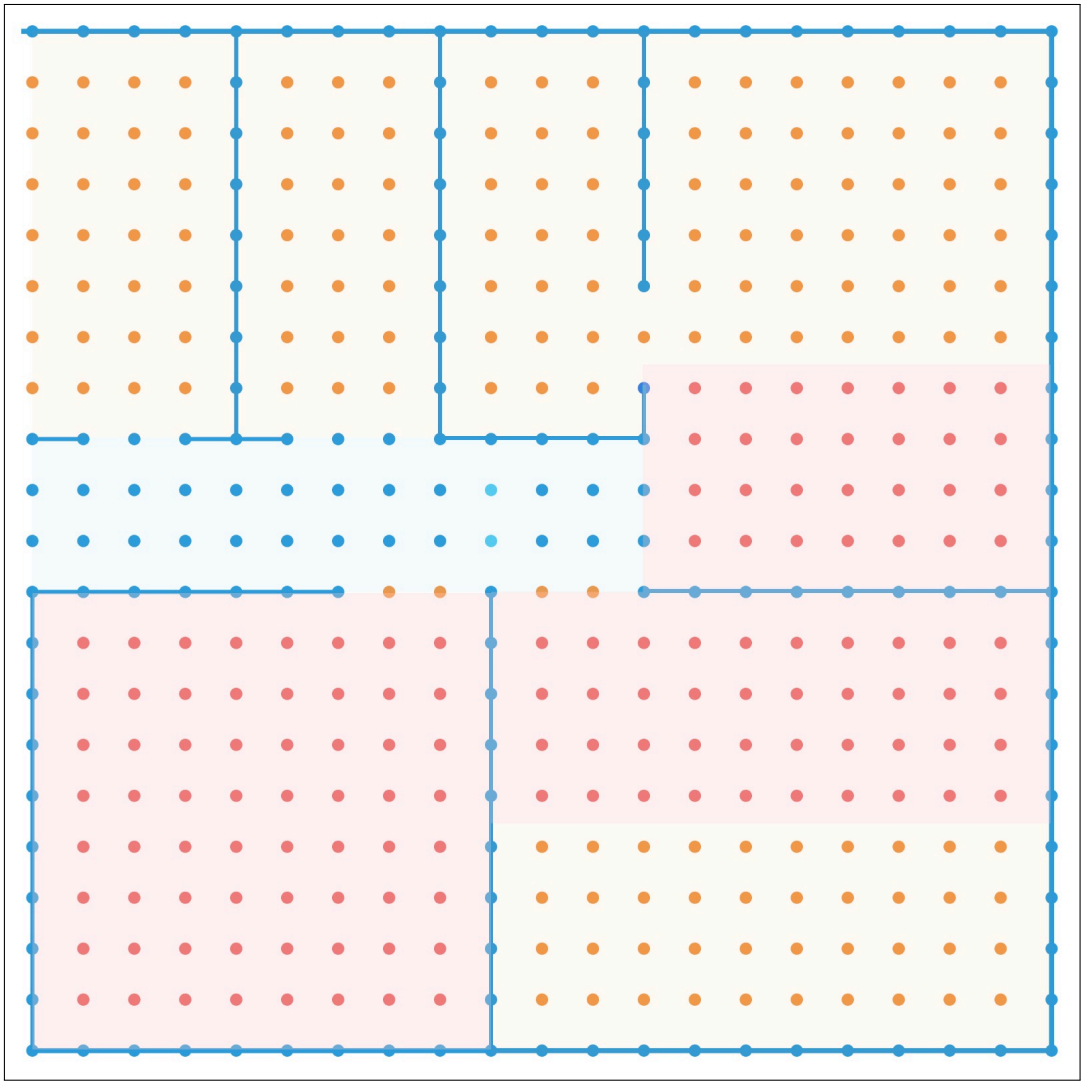
Tipo de Anteparo	Espessura (cm)	Atenuação (dB)
Divisória de Madeira	4	2
Divisória de Vidro	3	3
Placa de gesso (Drywall)	5 a 10	4
Parede de Tijolo	15	6
Parede de Concreto	15 a 20	9
Parede de Concreto	25 a 30	12

FONTE: (RUFINO A.; SILVA, 2016)

  - Raio: cobertura deste tipo de roteador;
  - Custo: indicando o preço deste roteador;
  - Capacidade do roteador.
- Np: o número de pontos considerados, possuindo as seguintes informações:
  - Tipo do ponto;
  - Posição x na grade;
  - Posição y na grade;
  - Demanda por qualidade de sinal: baixa, média ou alta;
  - Demanda por alta capacidade: proximidade do roteador;
- Na: o número de anteparos no caso teste, sendo cada anteparo sendo descrito da seguinte forma:
  - Coordenada x do ponto inicial do anteparo na grade;
  - Coordenada y do ponto inicial do anteparo na grade;
  - Coordenada x do ponto final do anteparo na grade;
  - Coordenada y do ponto final do anteparo na grade;
  - Fator de redução que o sinal sofre ao atravessar este anteparo, em %.

A Figura 4 ilustra uma instância de teste de exemplo. Neste caso, há paredes como anteparo. Os pontos vermelhos podem representar sinal com alta capacidade de rede. Os pontos em laranja podem representar que precisa-se de sinais com qualidade alta obrigatoriamente. Os pontos em azul podem representar que precisa-se apenas de ser coberto, com qualidade pelo menos baixa.

Figura 4 – Exemplo de uma instância de teste e a diferença entre seus pontos.



FONTE: Autoria própria.

As instâncias são compostas de dois tipos: randômicas e reais, sendo estas instâncias possuindo os três tipos de roteadores indicados na Tabela 2. 7 grupos de instâncias randômicas estão nesses casos teste, cada um contendo 10 casos de teste em ordem crescente de dificuldade. Cada grupo possui o mesmo conjunto de pontos e de anteparos, sendo o tipo de anteparo selecionado aleatoriamente, bem como a demanda por qualidade e por largura de banda de cada ponto. Além disso, cada grupo possui identificações de largura e altura, em metros, da área nas quais foram gerados os pontos. O grupo 8x14 é o menor e o 100x50 é o maior.

O conjunto de dados também possuem instâncias baseadas em plantas-baixas reais, com dimensões em metros:

- *me_100x40*: estação de metrô de 100x40;

Tabela 2 – Tipos de roteadores utilizados.

Tipo de Roteador	**Custo (R$)**	**Alcance (m)**	**Banda Passante (Mbps)**
Roteador de Entrada	150	100	150
Roteador Médio	200	100	300
Roteador de Alta Qualidade	300	400	400

FONTE: (RUFINO A.; SILVA, 2016)

- *ho_160x32*: andar de um hotel de 160x32;
- *of_1100x35*: referente a um escritório de1 100x32;
- *sc_170x45*: referente a uma escola de 170x45;
- *sh_1100x60*: referente a um pequeno *shopping center* de 1100x60.

5 RESULTADOS E EXPERIMENTOS COMPUTACIONAIS

Este capítulo apresenta os resultados obtidos a partir de experimentos computacionais realizados com as instâncias de teste como entrada, expostas em 4.3. A hiper-heurística de construção desenvolvida foi comparada com os resultados para a meta-heurística *ILS-RVND*(RUFINO A.; SILVA, 2016).

Os métodos estudados neste trabalho foram desenvolvidos em linguagem C++, utilizando o compilador clang (COMMUNITY, 2023) versão 14.0.0. Os experimentos computacionais foram executados em um computador com processador Intel Core i7 (64 bits), com CPU contendo 6 *cores* de 2,6 GHz, além de memória RAM DDR4 de 16 GB, 2667 MHz e executando o sistema operacional macOS Ventura versão 13.1 (APPLE, 2023).

Os conjuntos de testes utilizados podem ser agrupados em 7 grupos divididos por dimensões. As dimensões escolhidas foram: 8x14, 13x13, 11x23, 25x25, 30x30, 50x25 e 100x50. Cada conjunto de teste possui 10 instâncias distintas, indexadas a partir de 0, e são gerados aleatoriamente de acordo com os parâmetros de entrada expostos em 4.3.

Para fins de simplificação, denominamos a hiper-heurística de construção desenvolvida neste trabalho de *HH* (uma sigla para hiper-heurística) e a meta-heurística *ILS-RVND* de apenas ILS. Ambos os métodos foram submetidos a 10 execuções a cada instância, com um tempo limite de 30 segundos para cada execução.

Em todas as tabelas, a coluna **Instância** nomeia cada instância de entrada. As colunas **ILS-avg** e **ILS-best** expõem a média do resultado da função objetivo para as 10 execuções de cada instância, bem como a melhor função objetivo encontrada entre as execuções, respectivamente, para o método *ILS*. As colunas **HH-avg** e **HH-best** realizam o mesmo para os resultados da hiper-heurística de construção. A coluna **HH-best-sequence** ilustra a sequência (descrita na Figura 2) encontrada para a melhor função objetivo obtida para a *HH*.

As Tabelas 3 e 4 apresentam os resultados para a execução de testes com instâncias de tamanho 8x14 e 13x13, respectivamente. Neste casos, o *HH* obteve os melhores resultados em todas as instâncias de teste, empatando com o *ILS*, exceto por 1. Nas instâncias de 8x14, o *ILS* se sobressaiu nos resultados em média 3 vezes.

As Tabelas 5 e 6 apresentam os resultados para a execução de testes com instâncias de tamanho 11x23 e 25x25, respectivamente. Para as instâncias de 11x23, os melhores resultados da *HH* se sobressaem 3 vezes, contra apenas 1 vez do *ILS*. Para 25x25,

Tabela 3 – Execução de testes com instâncias de dimensões 8x14.

Instância	ILS-avg	ILS-best	HH-avg	HH-best	HH-best-sequence
r-(8x14)-0	339.0	339	339.0	339	1 0 | 0 0 0 0 1 0 0
r-(8x14)-1	181.0	181	181.0	181	0 1 | 0 0 1 0 1 0 0
r-(8x14)-2	181.0	181	181.0	181	0 1 | 0 0 0 0 0 0 0
r-(8x14)-3	**333.0**	333	333.8	333	0 1 | 0 0 0 0 0 0 0
r-(8x14)-4	326.0	326	326.0	326	0 1 | 0 0 0 0 0 0 0
r-(8x14)-5	338.0	338	338.0	338	1 0 | 1 0 0 0 0 0 0
r-(8x14)-6	222.0	222	222.0	222	0 1 | 0 0 0 0 0 0 0
r-(8x14)-7	**336.0**	336	336.2	336	0 1 | 0 0 1 1 0 0 1
r-(8x14)-8	229.0	229	229.0	229	1 0 | 0 0 1 0 0 0 0
r-(8x14)-9	**339.0**	339	339.1	339	0 1 | 0 0 0 0 0 1 0

Tabela 4 – Execução de testes com instâncias de dimensões 13x13.

Instância	ILS-avg	ILS-best	HH-avg	HH-best	HH-best-sequence
r-(13x13)-0	300.0	300	300.0	300	0 1 | 0 0 0 1 0 0 0
r-(13x13)-1	305.0	305	305.0	305	0 1 | 0 0 0 0 0 0 1
r-(13x13)-2	513.0	513	513.0	513	0 1 | 0 0 0 0 0 0 0
r-(13x13)-3	**512.0**	**512**	513.0	513	0 1 | 0 0 0 0 1 0 0
r-(13x13)-4	613.0	613	613.0	613	0 1 | 0 0 0 0 0 0 0
r-(13x13)-5	613.0	613	613.0	613	0 1 | 0 0 0 0 0 0 1
r-(13x13)-6	703.0	703	703.0	703	0 1 | 0 1 0 0 1 0 1
r-(13x13)-7	812.0	812	812.0	812	0 1 | 0 0 0 0 0 0 0
r-(13x13)-8	963.0	963	963.0	963	0 1 | 0 0 0 0 0 0 0
r-(13x13)-9	1212.0	1212	1212.0	1212	0 1 | 0 0 0 0 0 0 0

Tabela 5 – Execução de testes com instâncias de dimensões 11x23.

Instância	ILS-avg	ILS-best	HH-avg	HH-best	HH-best-sequence
r-(11x23)-0	457.0	457	457.0	457	1 0 | 0 0 0 0 1 0 0
r-(11x23)-1	522.0	522	522.0	522	0 1 | 0 0 0 0 0 0 0
r-(11x23)-2	619.6	619	**618.1**	**618**	0 1 | 0 0 1 0 0 0 0
r-(11x23)-3	752.6	750	750.0	750	0 1 | 0 1 0 0 0 0 0
r-(11x23)-4	823.0	823	823.0	823	0 1 | 0 0 0 0 1 0 0
r-(11x23)-5	1103.3	**1019**	**1094.7**	1022	0 1 | 0 1 0 0 0 0 0
r-(11x23)-6	**1113.1**	1113	1115.0	1113	0 1 | 0 0 0 0 0 0 0
r-(11x23)-7	1156.2	1153	**1150.0**	**1150**	0 1 | 0 0 0 0 0 0 0
r-(11x23)-8	1322.8	1322	**1322.0**	**1322**	0 1 | 0 0 0 0 0 0 0
r-(11x23)-9	1523.0	1523	1523.0	1523	0 1 | 0 0 0 0 0 0 0

a *HH* supera os melhores resultados da função objetivo de *ILS* por 2 vezes, empatando em todas as outras.

Para as instâncias de 30x30 expressas na Tabela 7, o *ILS* foi superior por todas as melhores execuções. Já nas instâncias de 50x25 na tabela 8, o *HH* consegue superar o *ILS* por 3 vezes, mas possui valores de função objetivo piores por 4 momentos.

Tabela 6 – Execução de testes com instâncias de dimensões 25x25.

Instância	ILS-avg	ILS-best	HH-avg	HH-best	HH-best-sequence
r-(25x25)-0	1300.0	1300	1300.0	1300	0 1 | 0 0 0 0 1 0 0
r-(25x25)-1	1300.0	1300	**1270.0**	**1200**	0 1 | 0 1 1 0 0 0 0
r-(25x25)-2	1600.0	1600	1600.0	1600	0 1 | 0 0 0 0 0 0 1
r-(25x25)-3	1855.0	1800	1800.0	1800	0 1 | 0 0 0 0 0 0 0
r-(25x25)-4	2055.0	2000	2030.0	2000	0 1 | 0 0 0 0 0 0 0
r-(25x25)-5	2380.0	2300	**2325.0**	2300	0 1 | 0 0 0 0 0 0 0
r-(25x25)-6	2570.0	2500	2555.0	2500	0 1 | 1 0 1 0 0 0 0
r-(25x25)-7	**3100.0**	3100	3110.0	**3050**	0 1 | 0 0 1 0 1 0 0
r-(25x25)-8	3820.0	3800	3820.0	3800	0 1 | 0 0 0 0 1 0 0
r-(25x25)-9	**3930.0**	3900	3955.0	3900	0 1 | 0 1 0 0 0 0 0

Tabela 7 – Execução de testes com instâncias de dimensões 30x30.

Instância	ILS-avg	ILS-best	HH-avg	HH-best	HH-best-sequence
r-(30x30)-0	**1244.2**	**1208**	1293.0	1244	0 1 | 0 1 0 0 0 0 0
r-(30x30)-1	**1333.3**	**1333**	1346.5	1343	0 1 | 0 0 0 0 1 0 0
r-(30x30)-2	1804.9	1775	**1775.0**	1775	0 1 | 0 1 0 0 0 0 0
r-(30x30)-3	**1626.7**	**1600**	1730.9	1706	0 1 | 1 0 0 0 1 0 0
r-(30x30)-4	2247.6	2175	**2240.0**	2175	0 1 | 0 0 0 0 0 0 0
r-(30x30)-5	**2455.4**	**2452**	2512.6	2474	0 1 | 0 0 0 0 0 1 0
r-(30x30)-6	**3065.0**	**2975**	3075.0	3075	0 1 | 0 0 0 0 0 0 1
r-(30x30)-7	**2930.8**	**2890**	3064.8	3023	0 1 | 0 0 0 0 0 0 0
r-(30x30)-8	**4238.0**	**4175**	4290.0	4225	0 1 | 0 0 0 0 0 0 0
r-(30x30)-9	**4048.9**	**3971**	4131.0	4058	0 1 | 0 0 1 0 0 0 1

Tabela 8 – Execução de testes com instâncias de dimensões 50x25.

Instância	ILS-avg	ILS-best	HH-avg	HH-best	HH-best-sequence
r-(50x25)-0	2300.0	2300	**2290.0**	**2250**	0 1 | 0 0 0 0 0 0 0
r-(50x25)-1	2700.0	2700	**2620.0**	**2600**	0 1 | 0 0 0 0 0 0 0
r-(50x25)-2	3690.0	3600	**3655.0**	3600	0 1 | 0 0 0 0 1 0 0
r-(50x25)-3	4070.0	4000	**4060.0**	4000	0 1 | 0 0 1 0 0 0 0
r-(50x25)-4	4980.0	4900	4980.0	**4850**	0 1 | 0 0 0 0 0 0 0
r-(50x25)-5	**5185.0**	**5100**	5215.0	5150	0 1 | 0 0 1 0 0 0 0
r-(50x25)-6	**5985.0**	**5800**	6025.0	5900	0 1 | 0 0 1 0 0 0 0
r-(50x25)-7	**6605.0**	6550	6660.0	6650	0 1 | 0 0 1 0 0 0 1
r-(50x25)-8	**7765.0**	**7600**	7890.0	7750	0 1 | 0 0 0 0 1 0 0
r-(50x25)-9	**8725.0**	**8600**	8865.0	8800	0 1 | 0 0 0 0 0 0 1

Tabela 9 – Execução de testes com instâncias de dimensões 50x50.

Instância	ILS-avg	ILS-best	HH-avg	HH-best	HH-best-sequence
r-(50x50)-0	4950.0	4850	**4550.0**	**4300**	0 1 | 1 1 0 0 0 0 1
r-(50x50)-1	6185.0	6150	**5810.0**	**5450**	0 1 | 0 1 0 0 0 0 0
r-(50x50)-2	7265.0	7150	**6890.0**	**6700**	0 1 | 0 0 1 0 1 0 1
r-(50x50)-3	8338.0	8240	**8173.7**	**7847**	0 1 | 0 0 1 0 0 0 0
r-(50x50)-4	9650.0	9600	**9190.0**	**9000**	0 1 | 0 0 0 0 0 0 0
r-(50x50)-5	**11146.4**	11050	11170.0	**10750**	0 1 | 0 0 1 0 0 0 1
r-(50x50)-6	12605.0	11450	**11215.0**	**10900**	0 1 | 0 0 0 0 1 0 0
r-(50x50)-7	13262.7	12979	**12799.9**	**12250**	0 1 | 1 1 1 0 0 0 0
r-(50x50)-8	**16905.0**	16550	17030.0	**16500**	0 1 | 0 0 1 0 0 0 0
r-(50x50)-9	17435.0	17150	**17380.0**	**17000**	1 0 | 1 1 1 0 1 0 0

Tabela 10 – Execução de testes com instâncias de dimensões 100x50.

Instância	ILS-avg	ILS-best	HH-avg	HH-best	HH-best-sequence
r-(100x50)-0	26985.4	10266	**10331.7**	**9993**	0 1 | 0 0 1 0 1 0 0
r-(100x50)-1	12643.7	12415	**12548.9**	**12189**	0 1 | 0 0 1 0 0 0 0
r-(100x50)-2	57995.1	15500	**15263.3**	**14737**	0 1 | 0 0 1 0 0 0 0
r-(100x50)-3	51970.0	18880	**18968.3**	**18409**	0 1 | 0 1 0 0 0 0 0
r-(100x50)-4	44730.0	20100	**19775.0**	**19000**	0 1 | 0 0 0 0 0 0 0
r-(100x50)-5	115255.9	22450	**22056.5**	**21400**	0 1 | 0 0 0 0 0 0 0
r-(100x50)-6	131958.4	25400	**25074.8**	**23723**	0 1 | 0 1 0 0 0 0 0
r-(100x50)-7	164387.0	31422	**27734.5**	**26850**	0 1 | 1 1 0 0 0 0 0
r-(100x50)-8	239045.3	37400	**35120.2**	**34350**	0 1 | 0 1 1 0 0 0 0
r-(100x50)-9	454971.5	37731	**37322.8**	**36700**	1 0 | 1 1 1 0 0 0 0

Ao analisar as Tabelas 9 e 10, pode-se verificar que o *HH* consegue superar o método *ILS* em todos os melhores resultados de função objetivo. Além disso, o *HH* apenas possui média menor de função objetivo em 2 instâncias de 50x50.

Dito isto, o *HH* tende a encontrar resultados melhores para instâncias de dimensões maiores. No conjunto de instâncias construídas por (RUFINO A.; SILVA, 2016), o *HH* conseguiu encontrar resultados melhores no mesmo tempo limite mais vezes, principalmente impulsionado pelas instâncias de maior escala.

O *ILS*, mesmo sendo uma meta-heurística interessante para o problema, parece não conseguir explorar outros ótimos locais rapidamente. Isto se percebe ao verificar a magnitude alta das colunas **ILS-avg** em relação a **HH-avg**, chegando a serem quase 10 vezes de diferença em certos casos. O **HH-avg** possui valores próximos de **HH-best** também nas instâncias, indicando que o *HH* conseguiu chegar próximo no possível ótimo global em todas as execuções nesses casos.

Outro ponto interessante são as escolhas de heurísticas de baixo nível pela

HH: por diversas vezes na solução, as melhores execuções não precisaram ter mais que 4 heurísticas de busca local. Na maioria das vezes, precisaram apenas de 1 ou 2 destas heurísticas. Um ponto a se considerar é que apenas 2 instâncias, entre as Tabelas 9 e 10, possuíram a heurística de construção gulosa. Todas as demais tiveram a heurística de construção aleatória como parte da solução final, indicando que as heurísticas performaram melhor ao precisarem reajustar as posições de roteadores ou pontos de acesso.

6 CONSIDERAÇÕES FINAIS

O objetivo deste trabalho foi propor uma solução ao APPP utilizando uma hiper-heurística de construção de heurísticas, a fim de comparar com a literatura e expandir as abordagens de resolução deste problema.

Para construir a hiper-heurística, 5 heurísticas de busca local desenvolvidas por (RUFINO A.; SILVA, 2016) foram utilizadas como heurísticas de baixo nível para fazerem parte da estrutura. 2 outras heurísticas de refinamento são propostas neste trabalho, totalizando 7. A heurística de construção gulosa proposta por (RUFINO A.; SILVA, 2016) também foi utilizada, e combinada com uma heurística construtiva aleatória.

Os resultados computacionais demonstraram excelentes resultados para instâncias de dimensões 50x50 e 100x50, as maiores da base de dados consultada. Dentre as 20 instâncias destes dois grupos, a hiper-heurística obteve resultados melhores em todas, ao comparar com a meta-heurística *ILS-RVND* (RUFINO A.; SILVA, 2016).

Entretanto, para instâncias de dimensões menores, a hiper-heurística não obteve resultados melhores que o *ILS-RVND* com facilidade, levando a possibilidade de melhorias. Como o tempo limite da execução das instâncias foi configurado como 30 segundos, isto pode ter afetado a comparação entre a meta-heurística e a hiper-heurística de construção, visto que esta costuma necessitar de executar mais passos para conseguir resultados razoáveis.

Para trabalhos futuros, há a possibilidade de desenvolver uma hiper-heurística de construção de heurísticas ao APPP. Utilizando as mesmas heurísticas, ou dividindo em heurísticas menores, esta hiper-heurística pode ser treinada com as instâncias existentes, e obter resultados em tempos menores. Além disso, novas instâncias podem ser geradas, com dimensões maiores, a fim de comparar com maior eficácia a superioridade de soluções com hiper-heurísticas em dimensões de larga escala.

De forma geral, o método proposto cumpriu seu objetivo, de maneira a ter conseguido resultados animadores. Dito isto, diversas melhorias ainda podem ser realizadas para a estrutura exposta em trabalhos futuros, expandindo nosso conhecimento sobre o APPP ao modificar as abordagens do problema.

REFERÊNCIAS

ABDEL-BASSET, M.; ABDEL-FATAH, L.; SANGAIAH, A. K. Metaheuristic algorithms: A comprehensive review. In: **Computational Intelligence for Multimedia Big Data on the Cloud with Engineering Applications**. Elsevier, 2018. p. 185–231. Disponível em: <https://doi.org/10.1016/b978-0-12-813314-9.00010-4>.

ANDRONE, C.; PALADE, T. Radio coverage and performance analysis for local area networks. In: **2010 9th International Symposium on Electronics and Telecommunications**. IEEE, 2010. Disponível em: <https://doi.org/10.1109/isetc.2010.5679347>.

APPLE. **macOS Ventura**. 2023. Disponível em: <https://www.apple.com/br/macos/ventura/>. Acesso em 20/01/2023.

ARROYO, T. B. M. J. E. C. Heurísticas grasp aplicado ao problema de alocação de antenas de transmissão. **XXXVIII Simpósio Brasileiro de Pesquisa Operacional**, set. 2006.

BABAEI, H.; KARIMPOUR, J.; HADIDI, A. A survey of approaches for university course timetabling problem. **Computers &amp Industrial Engineering**, Elsevier BV, v. 86, p. 43–59, ago. 2015. Disponível em: <https://doi.org/10.1016/j.cie.2014.11.010>.

BURKE, E. K.; GENDREAU, M.; HYDE, M.; KENDALL, G.; OCHOA, G.; ÖZCAN, E.; QU, R. Hyper-heuristics: a survey of the state of the art. **Journal of the Operational Research Society**, Informa UK Limited, v. 64, n. 12, p. 1695–1724, dez. 2013. Disponível em: <https://doi.org/10.1057/jors.2013.71>.

COMMUNITY, C. **Clang: a C language family frontend for LLVM**. 2023. Disponível em: <https://clang.llvm.org/>. Acesso em 20/01/2023.

CORMEN, T. H.; LEISERSON, C. E.; RIVEST, R. L.; STEIN, C. **Introduction to Algorithms, Third Edition**. 3rd. ed. [S.l.]: The MIT Press, 2009. ISBN 0262033844.

DAHAL, K. P.; ALDRIDGE, C. J.; GALLOWAY, S. J. Evolutionary hybrid approaches for generation scheduling in power systems. **European Journal of Operational Research**, Elsevier BV, v. 177, n. 3, p. 2050–2068, mar. 2007. Disponível em: <https://doi.org/10.1016/j.ejor.2005.12.018>.

DARWIN, C. **On the Origin of Species**. Routledge, 1859. Disponível em: <https://doi.org/10.4324/9780203509104>.

FARKAS K., H. G. G. Optimization of wi-fi access point placement for indoor localization. **Journal IIT (Informatics IT Today)**, 2013.

GLOVER, F. Future paths for integer programming and links to artificial intelligence. **Computers &amp Operations Research**, Elsevier BV, v. 13, n. 5, p. 533–549, jan. 1986. Disponível em: <https://doi.org/10.1016/0305-0548(86)90048-1>.

GLOVER, F. Tabu search—part i. **ORSA Journal on Computing**, Institute for Operations Research and the Management Sciences (INFORMS), v. 1, n. 3, p. 190–206, ago. 1989. Disponível em: <https://doi.org/10.1287/ijoc.1.3.190>.

HERTZ, A. Finding a feasible course schedule using tabu search. **Discrete Applied Mathematics**, Elsevier BV, v. 35, n. 3, p. 255–270, mar. 1992. Disponível em: <https://doi.org/10.1016/0166-218x(92)90248-9>.

HERTZ, A.; WIDMER, M. Guidelines for the use of meta-heuristics in combinatorial optimization. **European Journal of Operational Research**, Elsevier BV, v. 151, n. 2, p. 247–252, dez. 2003. Disponível em: <https://doi.org/10.1016/s0377-2217(02)00823-8>.

HOLLAND, J. H. **Adaptation in Natural and Artificial Systems**. The MIT Press, 1992. Disponível em: <https://doi.org/10.7551/mitpress/1090.001.0001>.

KAMENETSKY, M.; UNBEHAUN, M. Coverage planning for outdoor wireless LAN systems. In: **2002 International Zurich Seminar on Broadband Communications Access - Transmission - Networking (Cat. No.02TH8599)**. IEEE, 2002. Disponível em: <https://doi.org/10.1109/izsbc.2002.991793>.

KATOCH, S.; CHAUHAN, S. S.; KUMAR, V. A review on genetic algorithm: past, present, and future. **Multimedia Tools and Applications**, Springer Science and Business Media LLC, v. 80, n. 5, p. 8091–8126, out. 2020. Disponível em: <https://doi.org/10.1007/s11042-020-10139-6>.

KOZA, J. Genetic programming as a means for programming computers by natural selection. **Statistics and Computing**, Springer Science and Business Media LLC, v. 4, n. 2, jun. 1994. Disponível em: <https://doi.org/10.1007/bf00175355>.

LOURENÇO, H. R.; MARTIN, O. C.; STüTZLE, T. Iterated local search: Framework and applications. In: **International Series in Operations Research &amp Management Science**. Springer US, 2010. p. 363–397. Disponível em: <https://doi.org/10.1007/978-1-4419-1665-5_12>.

LU, J.-L.; JAFFRES-RUNSER, K.; GORCE, J.-M.; VALOIS, F. Indoor wLAN planning with a QoS constraint based on a markovian performance evaluation model. In: **2006 IEEE International Conference on Wireless and Mobile Computing, Networking and Communications**. IEEE, 2006. Disponível em: <https://doi.org/10.1109/wimob.2006.1696372>.

MEGIDDO, N.; TAMIR, A. On the complexity of locating linear facilities in the plane. **Operations Research Letters**, Elsevier BV, v. 1, n. 5, p. 194–197, nov. 1982. Disponível em: <https://doi.org/10.1016/0167-6377(82)90039-6>.

MICHALEWICZ, Z.; SCHOENAUER, M. Evolutionary algorithms for constrained parameter optimization problems. **Evolutionary Computation**, MIT Press - Journals, v. 4, n. 1, p. 1–32, mar. 1996. Disponível em: <https://doi.org/10.1162/evco.1996.4.1.1>.

NAKPRASIT, K.; PHONGCHAROENPANICH, C. Investigation of received signal strength of IEEE 802.11n WLAN in coverage of nakhon ratchasima rajabhat university. In: **2018 15th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON)**. IEEE, 2018. Disponível em: <https://doi.org/10.1109/ecticon.2018.8620018>.

OLGUN, B.; KOÇ, Ç.; ALTIPARMAK, F. A hyper heuristic for the green vehicle routing problem with simultaneous pickup and delivery. **Computers &amp Industrial Engineering**, Elsevier BV, v. 153, p. 107010, mar. 2021. Disponível em: <https://doi.org/10.1016/j.cie.2020.107010>.

PEARL, J. **Heuristics**. New York, NY: Addison Wesley Longman Publishing, 1984. (The Addison-Wesley series in artificial intelligence).

RESENDE, M. G. C.; RIBEIRO, C. C. Greedy randomized adaptive search procedures: Advances and extensions. In: **International Series in Operations Research &amp Management Science**. Springer International Publishing, 2018. p. 169–220. Disponível em: <https://doi.org/10.1007/978-3-319-91086-4_6>.

RUFINO A.; SILVA, T. G. Q. E. V. C. N. S. C. L. A. F. Meta-heurística ils para o problema de posicionamento automático de pontos de acesso em ambientes internos. **XLVIII Simpósio Brasileiro de Pesquisa Operacional**, set. 2016.

SOUZA, M. J. F. **Inteligência Computacional para Otimização: metaheurísticas**. [S.l.]: Departamento de Computação, Universidade Federal de Ouro Preto, Ouro Preto, Minas Gerais, 2022. Disponível em: <http://www.decom.ufop.br/prof/marcone/Disciplinas/InteligenciaComputacional/InteligenciaComputacional.pdf>.

VANHATUPA, T.; HANNIKAINEN, M.; HAMALAINEN, T. D. Genetic algorithm to optimize node placement and configuration for WLAN planning. In: **2007 4th International Symposium on Wireless Communication Systems**. IEEE, 2007. Disponível em: <https://doi.org/10.1109/iswcs.2007.4392413>.

ZHUANG, Y.; ZHOU, H. A hyper-heuristic resource allocation algorithm for fog computing. In: **Proceedings of the 2020 the 4th International Conference on Innovation in Artificial Intelligence**. ACM, 2020. Disponível em: <https://doi.org/10.1145/3390557.3394321>.

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DA PARAÍBA
Campus João Pessoa
Av. Primeiro de Maio, 720, Jaguaribe, CEP 58015-435, João Pessoa (PB)
CNPJ: 10.783.898/0002-56 - Telefone: (83) 3612.1200

Documento Digitalizado Ostensivo (Público)

Trabalho de Conclusão de Curso

Assunto: Trabalho de Conclusão de Curso
Assinado por: Calebe Figueirêdo
Tipo do Documento: Tese
Situação: Finalizado
Nível de Acesso: Ostensivo (Público)
Tipo do Conferência: Cópia Simples

Documento assinado eletronicamente por:

- **Calebe Oliveira de Figueirêdo, ALUNO (20181610021) DE BACHARELADO EM ENGENHARIA ELÉTRICA - JOÃO PESSOA,** em 01/03/2023 10:45:34.

Este documento foi armazenado no SUAP em 01/03/2023. Para comprovar sua integridade, faça a leitura do QRCode ao lado ou acesse https://suap.ifpb.edu.br/verificar-documento-externo/ e forneça os dados abaixo:

Código Verificador: 761662
Código de Autenticação: 89e26bf582

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DA PARAÍBA
Campus João Pessoa
Av. Primeiro de Maio, 720, Jaguaribe, CEP 58015-435, João Pessoa (PB)
CNPJ: 10.783.898/0002-56 - Telefone: (83) 3612.1200

Documento Digitalizado Ostensivo (Público)

Trabalho de Conclusão de Curso com Folha de Aprovação

Assunto: Trabalho de Conclusão de Curso com Folha de Aprovação
Assinado por: Calebe Figueirêdo
Tipo do Documento: Tese
Situação: Finalizado
Nível de Acesso: Ostensivo (Público)
Tipo do Conferência: Cópia Simples

Documento assinado eletronicamente por:

- **Calebe Oliveira de Figueirêdo, ALUNO (20181610021) DE BACHARELADO EM ENGENHARIA ELÉTRICA - JOÃO PESSOA,** em 15/03/2023 22:37:03.

Este documento foi armazenado no SUAP em 15/03/2023. Para comprovar sua integridade, faça a leitura do QRCode ao lado ou acesse https://suap.ifpb.edu.br/verificar-documento-externo/ e forneça os dados abaixo:

Código Verificador: 780015
Código de Autenticação: 817a73e3e7