UMA ABORDAGEM EVOLUCIONÁRIA PARA O PROJETO DE REDES EIXO-RAIO COM ALOCAÇÃO SIMPLES

BRUNO NONATO GOMES

UMA ABORDAGEM EVOLUCIONÁRIA PARA O PROJETO DE REDES EIXO-RAIO COM ALOCAÇÃO SIMPLES

Dissertação apresentada ao Programa de Pós-Graduação em Engenharia Elétrica da Escola de Engenharia da Universidade Federal de Minas Gerais como requisito parcial para a obtenção do grau de Mestre em Engenharia Elétrica.

ORIENTADOR: JAIME ARTURO RAMIREZ
CO-ORIENTADOR: RICARDO SARAIVA DE CAMARGO

Belo Horizonte
Agosto de 2011

Agradecimentos

Impossível chegar até aqui e não agradecer a quem contribuiu para a concretização de mais uma vitória!

Primeiramente agradeço a **DEUS** por estar sempre presente em minha vida, me abençoando e dando forças nesta caminhada.

Aos meus amados pais e irmãos que sempre estiveram ao meu lado em minhas decisões; aos demais familiares e amigos que, de alguma forma, fizeram parte dessa caminhada; à todos aqueles que, realmente, torceram por mais essa benção em minha vida.

À minha namorada, Thais, pela compreensão, carinho e amor a mim dedicados, e principalmente por ter tolerado minhas ausências.

Aos professores e funcionários do programa de pós-graduação em engenharia elétrica; aos colegas do GOPAC, em especial minha "mãe substituta", Luciana, pelos momentos de descontração e amizade; aos amigos da república onde morei.

Ao meu co-orientador Ricardo Saraiva de Camargo por sua orientação, direcionamento desse trabalho e, principalmente, por sua sincera amizade.

Ao meu orientador Jaime Arturo Ramirez, por sua orientação, conselhos sábios e sugestões valorosas que fizeram este momento tornar-se realidade.

Ao CNPq pelo apoio financeiro, sem o qual esse trabalho não teria sido possível.

"Não mostre ao seu Deus o tamanho do seu problema,
mostre ao seu problema o tamanho do seu Deus."
(Autor desconhecido)

Resumo

O projeto de redes eixo-raio com alocação simples é foco desta dissertação. Esse é um problema muito importante na área de otimização discreta, possuindo diversas aplicações em diferentes contextos, tais como sistemas de telecomunicação e informação, redes de transporte de carga e passageiros, entre outras. Para resolução desse problema, propõe-se três abordagens evolucionárias diferentes: algoritmo genético (AG), nomeado GGA; GGA com busca local; e GGA com descida em vizinhança variável (VND). O GGA possui uma fase de construção muito eficiente, a qual provê indivíduos de alta qualidade para a população inicial, e os operadores desenvolvidos, especificamente para o problema, são capazes de melhorar as soluções durante o processo evolucionário. Tendo em vista que os indivíduos promissores da população não passavam por um procedimento de refinamento de soluções, é proposta a implementação de 4 buscas locais para o problema. Porém, cada uma das buscas locais propostas explora uma determinada solução somente em uma vizinhança específica, assim, propõe-se também uma técnica, conhecida como VND, que explora o espaço de soluções através de trocas sistemáticas de estruturas de vizinhança. Dessa forma, as diferentes buscas locais propostas podem ser exploradas em uma solução. Para avaliação de desempenho dos métodos propostos são realizados experimentos computacionais utilizando a base de dados do serviço postal australiano (AP). Inicialmente, o GGA proposto é comparado com três outros AGs considerados estado da arte na literatura. Os resultados mostram que o GGA claramente supera os outros AGs estudados, tanto em qualidade das soluções, quanto em tempo para obtenção de uma solução alvo. Posteriormente, o GGA é comparado com o GGA combinado com as buscas locais. Percebe-se que, a combinação do AG com as buscas locais propostas proporcionam melhores soluções que o GGA sozinho, e requisita menor tempo de processamento para obtenção de uma solução alvo. Os melhores resultados apresentados são proporcionados pelo GGA com o VND, o qual superou o GGA com a busca local de re-alocação em todas as métricas avaliadas, e é mais rápido para obtenção de uma solução alvo. Além disso, o GGA-VND também é mais eficiente para encontrar a solução ótima das instâncias teste.

Palavras-chave: Otimização combinatória, redes eixo-raio, algoritmos evolucionários.

Abstract

The designing of hub-and-spoke networks with single allocation is the focuss of this work. This is a very important problem in the discrete location research, and it has many applications in several contexts such as telecommunication and information systems, and cargo and passengers transportation networks, amongst others. To tackle this problem, 3 evolutionary approaches are investigated: genetic algorithm (GA), named GGA; GGA with local search; e GGA with variable neighborhood descend (VND). The GGA has a very efficient construction phase that provides high quality individuals for the initial population. In addition, the developed operators, specific for the problem, are able to improve the solutions over the evolutionary process. As the promising individuals of the population are not submitted to a solution refinement procedure, this work proposes 4 local searches to the problem. Considering that each proposed local search explores a determined solution only in one specific neighborhood, this dissertation proposes one technique, known as VND, that explores the solution space trough systematics exchanges of neighborhood structure, and enables the different local searches be exploited in one solution. For the performance evaluation of the proposed methods, computational experiments using the data set of the Australian post service (AP) are carried out. Initially, the proposed GGA is compared with three others GAs considered to be state-of-the-art in the literature. The results show that the GGA clearly outperforms the others studied GAs both in solution quality and CPU time to obtained a target solution. Then, the GGA is compared with the GGA combined with the local searches. It is noticed that the combination of the GA with the proposed local searches provide better solutions than the GGA alone, and requires less CPU time to obtained a target solution. The best solutions presented are provided by the GGA combined with VND that outperforms the GGA with local search of re-allocation in all evaluate metrics, and it is faster to achieve a target solution. Furthermore, the GGA-VND is also more efficient to achieve the optimal solution of the test instances.

Keywords: Combinatorial optimization, hub-and-spoke networks, evolutionary algorithms.

Lista de Figuras

Lista de Tabelas

Sumário

Capítulo 1

Introdução

1.1 Introdução

Redes estão presentes no cotidiano das pessoas de diversas maneiras, como por exemplo: redes de telecomunicações, transporte, computadores, sociais, entre outras. De uma maneira geral, as redes possuem pontos de origem e destino que comunicam entre si. A topologia de uma rede depende das características de demanda, assim como dos custos fixos de instalação e operação da mesma. Em consequência dessa característica, vários tipos de redes surgiram para atender os requisitos de fluxo de cada configuração.

Uma configuração de rede possível é a do tipo malha (*mesh*), na qual todos os nós são interligados formando um grafo completo. Devido a inviabilidade de custo e operação da rede do tipo malha, as redes do tipo eixo-raio (E-R), ou em inglês *hub-and-spoke networks*, são uma alternativa de configuração mais econômica.

Nas redes E-R, a comunicação entre os nós não acontece de forma direta, mas através de nós concentradores. Dependendo da aplicação, os concentradores podem se apresentar de diversas maneiras, como por exemplo, depósitos, pontos de agregação e/ou distribuição de tráfego em redes de computadores, entre outras. Assim sendo, essas redes são formadas por concentradores, nós escolhidos estrategicamente, que agregam o fluxo de tráfego dos nós não concentradores, sendo responsáveis pelo roteamento e distribuição desse fluxo entre os diferentes pontos de origem e destino. Tal configuração diminui a quantidade de ligações diretas entre os nós, conforme evidenciado pela Figura 1.1. Nessa Figura, (a) mostra a disposição dos nós no espaço, (b) apresenta uma rede do tipo malha e (c) uma rede do tipo E-R com todos os concentradores interconectados, alocação simples dos nós não concentradores aos concentradores instalados e sem permissão de ligação direta entre nós não concentradores.

Diversas aplicações possibilitam a utilização das redes E-R, como por exemplo: companhias aéreas de passageiros [1, 2, 3], empresas de entrega de pacotes expressos [4], redes de

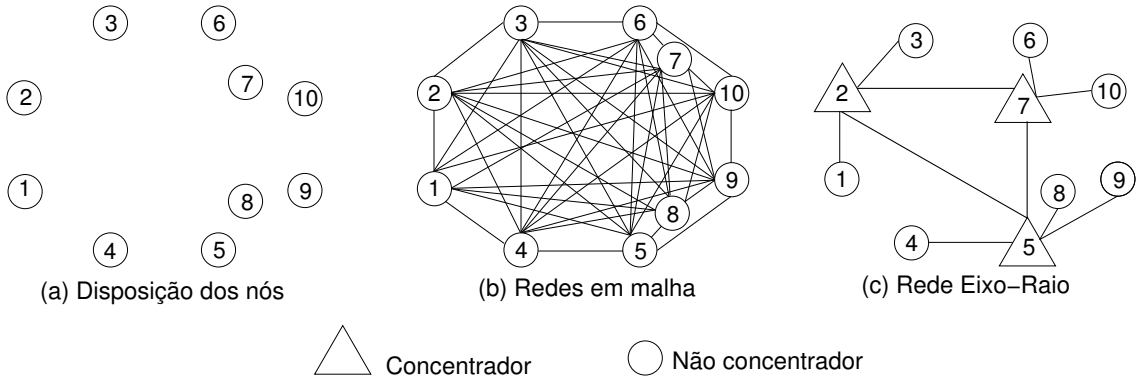

Figura 1.1. Topologias: Malha e Eixo-Raio.

entregas de mensagens [5], indústrias de transporte por caminhões [6, 7, 8], sistemas de telecomunicações [9], cadeia de lojas tais como *Wal-Mart* [10], entre outras [11, 12]. Assim, as redes E-R se tornaram um importante campo de pesquisa na área de otimização discreta, onde facilidades, aqui chamadas de concentradores, são responsáveis por agregação, roteamento e distribuição de tráfego entre diferentes pontos de origem/destino.

Em redes de telecomunicações, a aplicação desse tipo de redes pode ser exemplificada pela parte (a) da Figura 1.2. Considere uma pessoa localizada no prédio 2 que deseja fazer uma ligação para outra pessoa localizada no prédio 6. Assim, a requisição da ligação é enviada para a central em que a origem está alocada, nesse caso a central 1, que por sua vez determina o encaminhamento da ligação para um prédio alocado a ela mesma ou a outra central. Como o destino está associado a outra central, a requisição é enviada da central 1 para a central 2. Finalmente, a central 2 repassa a ligação ao prédio 6. Vale ressaltar que a tecnologia das ligações entre as centrais transmite mais informações por unidade de tempo do que a utilizada entre as centrais e os prédios.

Outra aplicação das redes E-R pode ser demonstrada pela utilização das mesmas em sistemas logísticos de transporte, conforme mostrado na parte (b) da Figura 1.2. Nessa Figura, considere uma empresa que presta serviço de transporte rodoviário de cargas em que a demanda de um cliente não é suficiente para ocupar a capacidade do veículo em uma viagem. Por esse motivo, cargas de diferentes clientes são agregadas, formando um grupo de cargas, e transportadas em conjunto. Para isto, as empresas que trabalham dessa maneira possuem instalações físicas, concentradores ou depósitos, localizadas em diversas regiões para agregar as cargas vindas das diferentes origens. Portanto, a realização do serviço compreende as operações de coleta, de um cliente até concentrador de origem; transferência, do concentrador de origem para um concentrador de destino; e distribuição, do concentrador de destino até o cliente final. Assim, um bom planejamento da rede de transporte, com boa localização dos concentradores, pode implicar em ganhos financeiros significativos para uma empresa que trabalha com esse

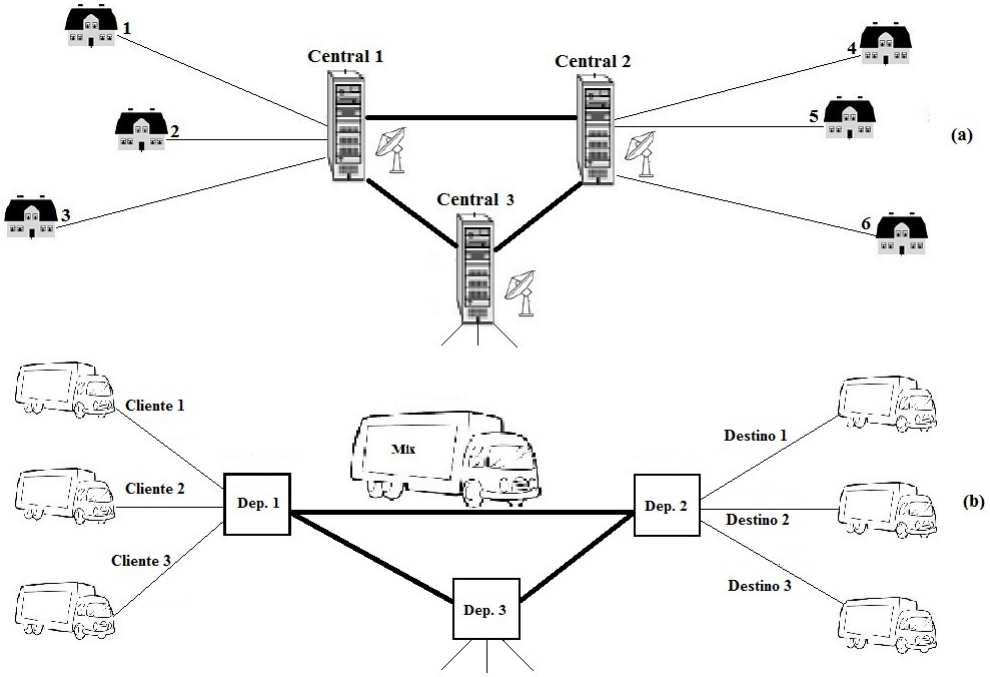

Figura 1.2. Aplicação das redes eixo-raio em telecomunicação e sistemas logísticos.

tipo de serviço.

De maneira semelhante, os serviços dos correios também podem ser vistos como uma rede E-R. Considere, por exemplo, que as diversas correspondências vindas de diferentes origens são recolhidas e concentradas em uma central. Por sua vez, essa central realiza uma triagem e separa tais correspondências por semelhança de destino, encaminhando-as para a central que será responsável pela distribuição das mesmas aos diferentes destinatários.

Um benefício trazido pela configuração das redes E-R é o fluxo agregado nos concentradores que permite a utilização de meios de forma mais eficiente, sendo transmitido maior volume de tráfego nas conexões entre concentradores, resultando assim em menor custo de transporte por unidade [13]. Assim, economia de escala é obtida no transporte de grande quantidade de fluxo nas ligações entre os nós concentradores. Essa economia de escala geralmente é representada por um fator constante de desconto no custo de transporte ($0 < \alpha < 1$).

É valido lembrar que a redução do custo de transporte obtido pela economia de escala, a diminuição dos custos de instalação, aumento da eficiência logística e do desempenho do sistema são as principais vantagens obtidas pela utilização desse tipo de rede [11].

Geralmente, no projeto de redes E-R é assumido que: todos concentradores são interconectados; nós não concentradores não são diretamente conectados; toda demanda é roteada por um ou no máximo dois concentradores. Além disso, conforme as características consideradas, diferentes suposições devem ser contempladas incluindo:

- Alocação simples ou múltipla dos nós aos concentradores: são duas possíveis configurações de alocação dos nós aos concentradores. Tais configurações se diferem na quantidade de concentradores que um nó não concentrador pode estar alocado: na alocação simples os nós devem estar alocados a exatamente um concentrador, e na alocação múltipla os nós podem estar alocados a mais de um concentrador;

- Número de concentradores pré-determinado ou variável: essas duas variantes se diferem em relação ao número de concentradores presentes na rede. Na primeira variante, a rede deve possuir p nós concentradores, sendo p um número pré-determinado. Na outra variante o número de nós que devem ser concentradores é variável de decisão do problema;

- Permissão ou não de ligação direta entre nós não concentradores: nessas variantes, a ligação direta entre alguns nós não concentradores pode ou não ser permitida. Ou seja, quando essa ligação não é permitida todo fluxo deve ser roteado via concentradores. Porém, quando existe a permissão, o fluxo pode ser trocado diretamente entre nós não concentradores;

- Projeto com ou sem restrição de capacidade: existência ou não de restrições de capacidade em relação ao total de tráfego suportado por um concentrador. No projeto com restrição de capacidade, um concentrador possui um valor máximo de tráfego que é capaz de suportar, dessa forma o fluxo agregado à ele é limitado por esse valor. Já no projeto sem restrição de capacidade, a quantidade de tráfego que pode ser agregada em um concentrador é ilimitada.

- Congestionamento: nessas variantes, efeitos de congestionamento em um concentrador podem ou não ser considerados. Quando esses efeitos são considerados, existe um custo de congestionamento que pode ser representado por restrições ou expresso na função objetivo do problema. Já na outra variante, caso de não consideração desses efeitos, um concentrador opera normalmente inclusive em casos de sobrecarga de fluxo imposta ao mesmo.

A Figura 1.3 apresenta diferentes variantes das redes eixo-raio: alocação simples ou múltipla; e permissão ou não de ligação direta entre nós não concentradores. Nessa Figura, a parte (a) representa uma rede com alocação simples e sem permissão de ligação direta entre nós não concentradores; já a parte (b) exemplifica a rede com alocação múltipla e com permissão de ligação direta entre nós não concentradores.

Este trabalho abordará o projeto de redes E-R com alocação simples, que é um problema de difícil resolução muito encontrado na área de telecomunicações. Neste projeto, o objetivo

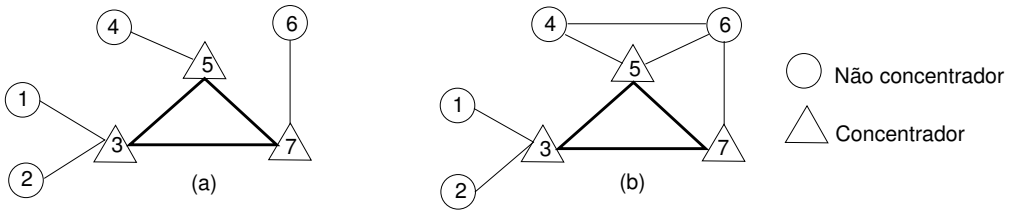

Figura 1.3. Variantes das redes eixo-raio.

é encontrar o número de concentradores e a localização dos mesmos na rede, bem como atribuir os nós não concentradores aos concentradores instalados. As seguintes características são encontradas no referido problema:

- O número de concentradores não é conhecido previamente;
- Os nós são atribuídos a exatamente um concentrador;
- Todos nós concentradores são interconectados;
- Impossibilidade de ligação direta entre nós não concentradores;
- Pelo menos um e no máximo dois concentradores devem ser responsáveis pelo roteamento e transmissão do fluxo de informações;
- Não há restrição de capacidade nem efeitos de congestionamento nos concentradores;
- Economia de escala é obtida nas conexões entre concentradores.

Considerando que cada nó não concentrador é alocado a somente um concentrador, a formulação matemática pode ser dada da seguinte maneira [14]:

$$\textbf{Minimize} \sum_{k=1}^{n} f_k z_{kk} + \sum_{i=1}^{n} \sum_{j=i+1}^{n} \sum_{k=1}^{n} \sum_{m \neq k}^{n} (w_{ij} c_{ijkm} + w_{ji} c_{ijmk}) z_{ik} z_{jm} \tag{1.1}$$

$$\textbf{Sujeito a:} \sum_{k=1}^{n} z_{ik} = 1 \qquad \forall\, i \in N \tag{1.2}$$

$$z_{ik} \leq z_{kk} \qquad \forall i \neq k \in N \tag{1.3}$$

$$z_{ik} \in \{0, 1\} \qquad \forall\, i,\ k \in N \tag{1.4}$$

Onde N é o conjunto de n nós, w_{ij} é o fluxo de demanda originado em i com destino à j (i, $j \in N : i \neq j$) o qual é roteado por um ou dois concentradores instalados. Essa demanda originada em i destinada a j (i, $j \in N : i \neq j$), e roteada pelos concentradores k e m (k, $m \in N : k \neq m$), passa por três segmentos: coleta do nó i para concentrador k; transferência entre os concentradores k e m; distribuição do concentrador m para nó j. O custo por unidade de demanda ao longo desse caminho é dado por $c_{ijkm} = c_{ik} + \alpha c_{km} + c_{mj}$, sendo $0 \leq \alpha \leq 1$ um fator de desconto que representa a economia de escala nas conexões entre os concentradores. Se somente um concentrador é utilizado em alguma rota, o fator de desconto não é aplicado. Além disso, o custo fixo para instalação de um concentrador no nó k é dado por f_k. A variável $z_{ik} \in \{0, 1\}$ indica a localização dos concentradores e a alocação dos nós aos concentradores instalados. Assim, se $z_{ik} = 1$ o nó i é atribuído ao concentrador k, e $z_{ik} = 0$ caso contrário. Adicionalmente, se um concentrador é localizado no nó k, $z_{kk} = 1$; senão $z_{kk} = 0$.

A equação (1.1) objetiva a minimização dos custos variáveis de transporte e custos fixos de instalação. As restrições descritas por (1.2) garantem que cada nó pode estar conectado a somente um concentrador. As restrições representadas por (1.3) permitem a alocação do nó i ao nó k, somente se *k* for um concentrador. As equações (1.4) são restrições de integralidade das variáveis z_{ik}.

O projeto de uma rede E-R pode ser exemplificado pela rede da Figura 1.4. Essa rede é constituída de 7 nós, N = {1 2 3 4 5 6 7}, sendo que 2 deles (2 e 5) foram escolhidos como concentradores. A atribuição dos nós se deu da seguinte maneira: os nós 1 e 3 são associados ao concentrador instalado em 2, e o restante dos nós (4, 6 e 7) estão alocados ao concentrador instalado em 5. Dessa forma, o fluxo de demanda dos nós dessa rede, representado por w_{ij},

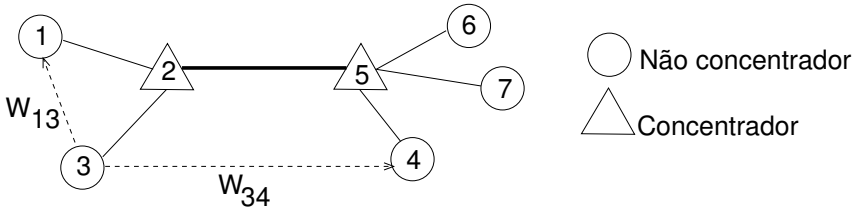

Figura 1.4. Comunicação nas redes eixo-raio.

deve ser roteado pelos concentradores 2 e/ou 5. Portanto, se $z_{ik}z_{jm} = 1$, então o fluxo será transmitido pelo caminho originado em i com destino a j, sendo roteado pelos concentradores k e m. Assim, a maneira como acontece a comunicação pode ser exemplificada através dos nós 3 e 4: a demanda w_{34} primeiramente é coletada do nó origem pelo concentrador 2 por um custo de c_{32}; após isso, o encaminhamento da mesma é feito para o concentrador 5 pelo custo de c_{25}; finalmente, o nó 5 envia a demanda ao destino por um custo c_{54}. Portanto, o custo total de transporte de uma unidade de demanda entre os nós 3 e 4, roteada pelos concentradores 2 e

5, é composto por $c_{3425} = c_{32} + \alpha c_{25} + c_{54}$. A comunicação entre os nós 1 e 3 acontece de forma semelhante, exceto pela inexistência de transmissão entre concentradores, já que ambos nós são associados ao mesmo concentrador (2). Assim sendo, o custo total de transporte da demanda entre os nós 1 e 3 é representada por $c_{321} = c_{32} + c_{21}$.

Existem duas possíveis abordagens para resolução desse problema: métodos exatos e métodos heurísticos. A primeira, refere-se ao procedimento que encontra a solução ótima para o problema, e a segunda é um procedimento que não garante a solução ótima do problema, mas garante resultado viável [15]. Métodos heurísticos buscam por soluções viáveis para os problemas de otimização nos casos em que a complexidade do problema ou a limitação de tempo de processamento inviabiliza a utilização dos algoritmos exatos [16].

A complexidade de um problema, geralmente, é representada pelo tempo de processamento requisitado para a resolução do mesmo. Assim, os problemas podem pertencer à duas classes de problemas diferentes: P e NP. A classe P possui todos os problemas que podem ser resolvidos por algoritmos determinísticos em tempo polinomial; e a classe NP, *Non-deterministic Polynomial time*, é o conjunto de problemas que podem ser resolvidos por algoritmos não-determinísticos em tempo polinomial processando em uma máquina não-determinística de Turing. Além disso, deve existir uma transformação polinomial do problema para outro problema pertencente à classe NP [17].

Como o projeto de redes E-R possui grande complexidade combinatória, pertencendo à classe NP-Difícil [12], uma abordagem evolutiva é proposta para a resolução do mesmo. Os algoritmos evolutivos têm mostrado bons resultados em problemas combinatórios como QAP [18], roteamento de veículos [19], entre outros.

Os algoritmos evolucionários (AEs) são uma classe particular de métodos que utilizam conceitos desenvolvidos a partir do processo natural de evolução das espécies vinda do paradigma da Teoria Darwiniana [20]. O uso dos modelos e princípios do *Neodarwinismo* se tornou suporte teórico e fonte de inspiração para desenvolvimento da maioria dos diferentes métodos evolutivos existentes. Esses algoritmos abstraem o processo evolucionário redesenhando-o em uma linguagem computacional. Eles funcionam segundo uma estratégia de populações e contêm, em geral, características semelhantes, sendo o processo evolucionário composto basicamente de três fases: seleção, variação e atualização.

1.2 Objetivos do trabalho

1.2.1 Objetivo geral

O presente trabalho tem como principal objetivo a investigação do comportamento dos AEs na resolução do projeto de redes do tipo E-R, mais especificamente o projeto de redes E-R

sem restrição de capacidade com alocação simples (*PRERSCAS*). Dessa forma, a implementação de um algoritmo genético (AG) é proposta nesse trabalho.

A investigação aborda a aplicabilidade do método implementado na resolução do *PRERSCAS*, assim como trata questões de resultados e desempenho em comparação com outros métodos da literatura.

1.2.2 Objetivos específicos

Para atingir o objetivo geral do trabalho os seguintes objetivos específicos são definidos:

- Codificar adequadamente os indivíduos possibilitando um bom desempenho dos operadores;
- Determinar vizinhanças a serem exploradas pelos operadores e buscas locais desenvolvidas;
- Desenvolver operadores específicos para o problema, já que os operadores convencionais do AG não se aplicam ao problema em questão;
- Buscar e implementar trabalhos baseados em AEs considerados estado da arte na literatura, de forma que os mesmos sirvam como comparação para o método implementado.

1.3 Contribuições

A principal contribuição desta dissertação é a proposta de um método baseado em princípios evolucionários que soluciona o projeto de redes E-R de forma eficaz e eficiente. Ou seja, este trabalho propõe um AG capaz de projetar uma rede E-R com custo total de operação reduzido em tempo viável, sendo aplicável a problemas de grande dimensão.

Primeiramente sugere-se um AG, no qual propõe-se uma fase de construção que gera indivíduos de alta qualidade para a população inicial. Além disso, os operadores de cruzamento e mutação, específicos para o problema, são capazes de evoluir as soluções durante o processo evolutivo, fazendo com que o método consiga bons resultados em tempo computacional viável.

Posteriormente, é também proposta a combinação desse AG com métodos de refinamento: buscas locais e descida em vizinhança variável. Esse refinamento é realizado em indivíduos promissores da população, e tem o objetivo de melhorar uma determinada solução explorando a vizinhança da mesma.

Como pode ser observado nos próximos capítulos, a comparação das abordagens propostas frente a métodos conhecidos como estado da arte na literatura revela ganhos consideráveis em relação a tempo computacional e qualidade das soluções obtidas.

1.4 Organização do trabalho

Esta dissertação está organizada em 5 capítulos. O presente capítulo faz uma introdução do problema a ser abordado, além de descrever os objetivos e contribuições do trabalho. O capítulo 2 apresenta o exame da literatura e descreve os diferentes tipos de redes E-R, assim como métodos para resolução dos mesmos. Além disso, o capítulo também contempla uma contextualização histórica dos AEs, descrevendo e discutindo detalhadamente os principais métodos baseados em AG para resolução do *PRERSCAS* encontrados na literatura, bem como a metodologia de apresentação de resultados desse trabalho. No capítulo 3, o AG implementado para o *PRERSCAS* é descrito com detalhes, evidenciando as características e operadores utilizados, assim como as busca locais e o método de descida em vizinhança variável desenvolvidos são apresentados. O capítulo 4 apresenta os resultados obtidos pela abordagem proposta em comparação com 3 AGs conhecidos da literatura, bem como mostra os resultados das melhorias realizadas no algoritmo implementado quando combinado com os métodos de refinamento. As conclusões, comentários finais e trabalhos futuros são apresentados no capítulo 5.

Capítulo 2

Exame da Literatura

Este capítulo apresenta alguns conceitos importantes sobre o projeto de redes E-R (*PRER*), fornecendo uma contextualização histórica e descrevendo as diferentes variantes desse problema, assim como métodos para resolução do mesmo encontrados na literatura. Além disso, uma revisão bibliográfica dos AEs é apresentada, descrevendo com detalhes os principais algoritmos da literatura baseados em AGs para resolução do problema em questão.

2.1 Projeto de redes eixo-raio

2.1.1 Introdução

Existem diversos trabalhos presentes na literatura que abordam o *PRER*. Esses trabalhos incluem até problemas que consideram o espaço contínuo, onde os concentradores podem ser instalados em qualquer lugar na região definida. Pesquisas abordando esse assunto podem ser encontradas nos trabalhos de Aykin [21], O'Kelly [22], O'Kelly e Miller [23], entre outros. Discussões detalhadas dessa variante fogem do escopo dessa dissertação.

As duas primeiras revisões sobre *PRER* foram realizados por Campbell [24] e O'Kelly e Miller [25] em 1994. No trabalho realizado por Campbell, o autor forneceu uma revisão do crescimento de pesquisas sobre localização de concentradores em redes e apresentou um esquema de classificação para os diferentes modelos e problemas considerados. Já O'Kelly e Miller focaram mais nas alternativas topológicas disponíveis em redes com concentradores, com abordagens para localização contínua [26] e discreta [14]. Porém, talvez o modelo mais antigo para o *PRER* pode ser considerado o proposto por Goldman [27] *apud* Alumur e Kara [11], o qual estendeu a propriedade de otimalidade do nó elaborada por Hakimi [28, 29] *apud* Alumur e Kara [11], e obteve o que é praticamente o projeto de rede E-R com concentradores medianos.

Depois disso, com o aumento nas pesquisas sobre *PRER*, outras revisões sobre a área foram apresentadas. A primeira delas foi a realizada por Klincewicz [5] que focou em trabalhos na área de telecomunicação, incluindo a topologia das redes. Posteriormente, Bryan e O'Kelly [30] revisaram pesquisas no contexto de transporte aéreo e sugeriram direções para trabalhos futuros. Recentemente, Alumur e Kara [11] classificaram e revisaram os diversos trabalhos e modelos para o *PRER*.

O *PRER* possui diversas semelhanças com os problemas de localização de facilidades clássicos. Para cada um dos problemas de localização de facilidades clássicos existem análogos *PRER* estudados e formulados:

- Problema de localização de p-facilidades medianas $\rightarrow$ *PRER* com localização de p-concentradores medianos;
- Problema de localização de p-facilidades centrais $\rightarrow$ *PRER* com localização de p-concentradores centrais,
- Problema de localização de facilidades não capacitadas $\rightarrow$ *PRER* com localização de concentradores não capacitados;
- Problema de localização com cobertura de facilidades $\rightarrow$ *PRER* com localização concentradores cobertos.

Porém, é necessário destacar uma diferença importante entre os *PRER* e os problemas de localização de facilidade clássicos, existem versões de alocação simples e alocação múltipla para cada variante do *PRER*.

Os *PRER* também possuem versões com restrição de capacidade, de maneira semelhante aos problemas de localização de facilidades clássicos. Contudo, as limitações de capacidade são diferentes para as duas classes de problemas. Devido à natureza da demanda dos *PRER*, pode haver diferentes limitações de capacidade: nos concentradores; no fluxo entre concentradores; ou entre concentradores e nós não concentradores.

Conforme descrito, existem diversas variantes para o *PRER*. As subseções seguintes descrevem quatro dessas variantes, que se diferem em relação ao número de concentradores instalados e à forma de alocação dos nós aos mesmos [11]: *PRER* com localização de p-concentradores medianos (p-*PRER* mediano), *PRER* com localização de p-concentradores centrais (p-*PRER* centrais), *PRER* com localização de concentradores cobertos e *PRER* com localização de concentradores com custos fixos de instalação.

Respeitando o escopo do trabalho, o exame da literatura será focado nas versões de alocação simples e sem restrição de capacidade das referidas variantes do *PRER*.

2.1.2 *PRER* com localização de *p*-concentradores medianos - *p-PRER* medianos

De acordo com Alumur e Kara [11], os *p-PRER* medianos têm o objetivo instalar exatamente *p* concentradores, minimizando o custo de transporte e atendendo todos os pontos de demanda. A primeira formulação linear inteira para esse problema foi elaborada por Campbell [31]. A formulação proposta possui $(n^4 + n^2 + n)$ variáveis, da quais $(n^2 + n)$ são variáveis binárias; além disso, possui $(n^4 + 2n^2 + n + 1)$ restrições lineares. Nesse trabalho, o autor propôs um modelo com limiares de fluxo, em que é definido um fluxo mínimo necessário para permitir a alocação do nós de demanda aos concentradores.

Após mostrar que a relaxação Lagrangeana para o modelo proposto em [31] obtinha soluções altamente fracionárias, Skorin-Kapov et al. [32] propuseram uma nova formulação inteira mista para o *p-PRER* medianos. Essa formulação contém $(n^4 + n^2)$ variáveis, sendo (n^2) delas binárias, e também possui $(2n^3 + n^2 + n + 1)$ restrições lineares. Os autores mostraram que essa formulação possui uma relaxação linear acurada.

Uma formulação linear inteira diferente, contendo menor número de variáveis ($n^3 + n^2$, sendo n^2 binárias) e restrições $(2n^3 + n + 1)$ foi proposta por Ernst e Krishnamoorthy [33]. Os autores trataram a transferência inter-concentradores como problema de fluxo de *multicommodities*, sendo que cada *commodity* representa o tráfego originado de um determinado nó. Além disso, os autores dividiram o fluxo em diferentes partes: coleta, transporte e distribuição. Os autores modelaram o problema com base em estudos do serviço postal australiano que utiliza diferentes fatores de desconto para coleta e distribuição.

Outra formulação, apresentada por Ebery [34], requer O(n^2) variáveis e O(n^2) restrições. Essa formulação utiliza menos variáveis que todas as formulações previamente encontradas na literatura. De acordo com Alumur e Kara [11], a resolução dessa formulação requer maior tempo computacional que a formulação de Ernst e Krishnamoorthy [33].

As primeiras heurísticas para resolução do *p-PRER* medianos foram propostas por O'Kelly [14]. Nesse trabalho foram desenvolvidas duas abordagens que enumeravam todas as possibilidades de escolha dos *p* concentradores. Na primeira heurística, os nós não concentradores são alocados ao concentrador mais próximo; já na segunda, escolhe-se o melhor, em termos de objetivo, entre o primeiro e o segundo concentrador mais próximo.

Vários métodos heurísticos foram propostos por Klincewicz [35, 36] para o referido problema. No primeiro trabalho, o autor desenvolveu uma heurística de troca baseada em busca local, considerando movimentos de troca simples e dupla. No segundo, dois algoritmos foram propostos, um baseado em busca tabu e outro em *Greedy Randomized Search Procedure* (GRASP); em ambos métodos os nós não concentradores são alocados ao concentrador mais próximo. As duas abordagens desenvolvidas obtiveram melhores resultados que a heurística de

O'Kelly [14].

A heurística busca tabu proposta por Skorin-Kapov e Skorin-Kapov [37] superou os resultados de O'Kelly [14] e Klincewicz [36], porém, devido a maior ênfase dada à fase de alocação, tal abordagem requer maior tempo computacional que os demais métodos.

Um método heurístico baseado em recozimento simulado, ou em inglês *Simulated Annealing* (SA), foi proposto no trabalho de Ernst e Krishnamoorthy [33]. Os resultados obtidos pelo algoritmo se comparam, em tempo computacional e qualidade de solução, com o método de Skorin-Kapov e Skorin-Kapov [37], porém se limita a resolver instâncias de até 50 nós.

Uma eficiente relaxação Lagrangeana foi elaborada por Pirkul e Schilling [38]. Esse método, baseado no algoritmo do subgradiente, proveu os menores erros em relação às soluções ótimas até então, variando de 0,0048% até 1%. Já o trabalho realizado por Smith et al. [39] mapeou o *p-PRER* medianos em uma rede neural modificada. O método foi comparado com o SA de Ernst e Krishnamoorthy [33] tendo resultados competitivos com o mesmo.

Outra heurística baseada em SA foi apresentada por Abdinnour-Helm [10], porém tal método obteve resultados inferiores ao método de Ernst e Krishnamoorthy [33].

2.1.3 *PRER* com localização de *p*-concentradores centrais - *p-PRER* centrais

O *p-PRER* centrais é um tipo de problema de *min-max*, ou seja, consiste em localizar p concentradores e alocar os nós não concentradores aos concentradores instalados minimizando o máximo de tempo, ou distância, entre qualquer par de nós origem/destino. A primeira formulação para esse problema foi elaborada por Campbell [31]. Nesse trabalho o autor discute o problema e define três tipos de *p-PRER* centrais:

- Minimizar o custo máximo entre qualquer par de nós origem/destino: sistemas que envolvem items perecíveis em que o custo é o tempo;

- Minimizar o custo máximo entre qualquer ligação disponível (origem-concentrador, concentrador-concentrador, concentrador-destino): sistemas que os items precisam de alguma fase de preservação ou transformação, aquecimento ou resfriamento, que são feitas nos concentradores;

- Minimizar o custo máximo entre o concentrador e o par de nós origem/destino: sistemas em que as ligações inter-concentradores possuem atributos especiais.

Outra formulação linear para o problema foi proposta por Kara e Tansel [40]. Essa formulação obteve resultados superiores às linearizações dos modelos elaborados por Camp-

bell [31]. Essa nova modelagem matemática possui $(n^2 + 1)$ variáveis, sendo (n^2) binárias, e $(n^3 + n^2 + n + 1)$ restrições lineares.

Uma nova formulação foi desenvolvida por Ernst et al. [41], a qual provê a inserção de uma variável que limita o custo máximo entre um concentrador e os nós alocados a ele. Experimentos computacionais mostraram que essa nova formulação superou os resultados de Kara e Tansel [40]. Em termos de número de variáveis e restrições, ela possui $(n^2 + n + 1)$ variáveis das quais (n^2) são binárias e $(3n^2 + n + 1)$ restrições lineares.

O primeiro método heurístico para o *p-PRER* centrais foi desenvolvido por Pamuk e Sepil [42]. Nesse trabalho, foi proposta uma heurística de localização-alocação, utilizando a busca tabu para fugir de ótimos locais. Além disso, uma busca local gulosa é empregada para melhorar as alocações dos nós não concentradores aos concentradores instalados. Os autores consideraram satisfatórios os resultados obtidos.

Ernst et al. [43] e Campbell et al. [44] focaram os estudos no subproblema de alocação do *p-PRER* centrais. Em ambas pesquisas, os autores apresentaram formulações lineares para o problema de alocação simples das variantes capacitadas e não capacitadas. No primeiro trabalho, Ernst et al. [43] propuseram 5 métodos heurísticos, analisaram o pior caso dos mesmos, e obtiveram bons resultados. Na segunda pesquisa, Campbell et al. [44] apresentaram resultados de complexidade do problema em questão.

2.1.4 *PRER* com localização de concentradores cobertos

Neste problema, o intuito é localizar os concentradores atendendo toda a demanda minimizando os custos de instalação dos concentradores. Porém, esse problema possui uma característica específica, pois a demanda de um determinado nó é considerada coberta, ou atendida, se a distância do mesmo à um concentrador capaz de atendê-lo não exceder um valor pré-estabelecido. Uma variante para o referido problema, encontrada na literatura, tem o objetivo de maximizar a demanda atendida dado um número de concentradores a serem localizados.

Formulações inteiras mista para ambas variantes do problema foram apresentadas por Campbell [31]. Nesse mesmo trabalho, o autor determinou 3 critérios de cobertura para os concentradores: o custo de transmissão do nó *i* para *j* distribuído pelos concentradores *k* e *m* não excede o valor máximo especificado; o custo de cada ligação do caminho *i-k-m-j* não ultrapassa o máximo especificado; cada uma das ligações origem-concentrador e concentrador-destino satisfazem, separados, os valores especificados.

Nos estudos feitos por Kara e Tansel [45] foram apresentadas linearizações para a formulação quadrática original, além disso foi proposto um novo modelo linear que se mostrou superior aos modelos lineares encontrados até então. Outras formulações foram propostas por Wagner [46], Ernst et al. [47] e Campbell [48]. Comparações de diversas formulações fo-

ram feitas no trabalho de Hamacher e Meyer [49], que analisaram a viabilidade do poliedro e identificaram desigualdades válidas.

2.1.5 *PRER* com localização de concentradores com custos fixos de instalação

Nesta versão do *PRER*, o custo fixo para instalação de um concentrador é agregado na função objetivo do problema. Nessa variante, o objetivo é encontrar o número e localização dos concentradores a serem instalados, assim como alocar cada nó não concentrador aos concentradores instalados, minimizando os custos fixos de instalação e os custos variáveis de transmissão. Uma formulação inteira quadrática para o referido problema foi elaborada por O'Kelly [50], na qual o número de concentradores é variável de decisão do problema. Já Campbell [31] propôs a primeira formulação linear para as diferentes versões do problema: alocação múltipla/simples e capacitado/não capacitado.

Uma nova formulação inteira quadrática, baseada na ideia de fluxo de *multicommodities*, foi apresentada por Abdinnour-Helm e Venkataramanan [51]. Os autores ainda propuseram um método *branch-and-bound* que utiliza a estrutura de fluxo *multicomodities* para obter limites inferiores e um AG para encontrar bons limites superiores. Os resultados encontrados foram considerados satisfatórios. Aykin [2] propôs um método baseado em *branch-and-bound* e SA, e obteve resultados competitivos.

Uma heurística híbrida baseada em AG e busca tabu foi proposta por Abdinnour-Helm [52]. O AG determina o número e localização dos concentradores, enquanto a busca tabu trabalha na alocação dos nós não concentradores aos concentradores instalados. Os resultados encontrados por esse método superaram o AG de Abdinnour-Helm e Venkataramanan [51].

Outro método baseado em AG, que obteve melhor performance que Abdinnour-Helm e Venkataramanan [52], foi proposto por Topcuoglu et al. [53]. O AG desenvolvido é responsável por determinar o número e localização dos concentradores, bem como alocar os demais nós aos concentradores instalados.

Cunha e Silva [8] desenvolveram uma eficiente combinação das heurísticas AG e SA. Nesse método, todas as soluções passam por uma busca local após as fases de cruzamento e mutação, objetivando melhorar a alocação dos nós não concentradores aos concentradores instalados. Além disso, os autores fizeram um estudo de caso com uma empresa de transporte no Brasil.

Chen [54] apresentou uma heurística híbrida baseada em SA, busca tabu e procedimentos de melhoria. Esse método híbrido obteve melhor performance que o AG de Topcuoglu et al. [53].

Mais recentemente, Silva e Cunha [55] desenvolveram 3 variantes da heurística busca tabu com *multi-start*, obtendo resultados satisfatórios que superam os demais métodos desenvolvidos até então.

Naeem e Ombuki-Berman [56] propuseram um AG, com duas formas de codificação das soluções e diferentes operadores de cruzamento, obtendo melhores resultados que os métodos propostos em [52], [53], e desempenho competitivo com os métodos de Chen [54] e Silva e Cunha [55].

Abordagens exatas também são consideradas para o referido problema. A pesquisa feita por Labbé e Yaman [57] trabalha com duas formulações e desigualdades válidas que podem ser separadas em tempo polinomial, mas os autores não apresentaram resultados computacionais. O método de decomposição de Benders foi aplicado por Castro et al. [58] e Castro [59] obtendo bons resultados. No primeiro trabalho, os autores apresentaram três implementações do método e resolveram instâncias de até 60 nós. No segundo trabalho, o autor apresenta resultados para instâncias de até 200 nós.

Outra abordagem utilizando a decomposição de Benders foi proposta por Camargo et al. [60] para a resolução do problema de localização de concentradores sem restrição de capacidade com alocação múltipla. O algoritmo foi testado com instâncias benchmark da literatura, sendo capaz de resolver problemas com até 81 nós. Além disso, os resultados obtidos pelo método superam softwares de programação inteira não-linear, como XPRESS-MP.

Outras pesquisas para as versões de alocação múltipla e/ou sem restrição de capacidade também são amplamente encontradas na literatura. O exame detalhado desses métodos foge do escopo do trabalho, podendo ser encontrado na revisão de Alumur e Kara [11].

2.2 Algoritmos evolucionários

2.2.1 Introdução

Nas últimas três décadas, o interesse em métodos inspirados por fenômenos naturais vem crescendo de forma estrondosa, resultando no desenvolvimento de uma classe de algoritmos conhecida como algoritmos evolucionários (AEs).

A ideia de que o sistema biológico era fixo e imutável foi predominante até o início do século XIX, ou seja, o pensamento era que o sistema não possuia dinamismo e a reprodução dos seres vivos geravam frutos que não sofriam mudanças com o passar do tempo. Porém, com a ciência empírica muito presente no renascimento, foi tomando forma a ideia de que novas formas de vida poderiam surgir das já existentes [61].

O pioneirismo dessa área pode ser dado à Lamarck com sua *Philosophie Zoologique*. Os contemporâneos da época combatiam Lamarck pela idealização de evolução existente nesse

trabalho. Mesmo com tantas dificuldades, a ideia vinha ganhando força, necessitando apenas de mais contribuições para se formalizar. Foi então que Darwin, no ano de 1831, embarcou em uma viagem de quase 5 anos pelo hemisfério sul com intuito de pesquisar as questões e ideias de evolução. Essa longa pesquisa culminou em um princípio capaz de sintetizar o processo evolutivo: a seleção natural baseada na luta pela existência, ou seja, a disputa pela sobrevivência é um mecanismo que leva à preservação de características favoráveis e eliminação das desfavoráveis, sendo que a sobrevivência do mais apto é predominante [61].

Tendo em vista a dinâmica da evolução, três fatores evolutivos atuam sobre as gerações de indivíduos de uma dada espécie [62]:

- Cruzamento ou recombinação genética: genes de indivíduos distintos são agrupados para geração de novos indivíduos;
- Mutação: perturbações na estrutura genética que podem alterar a informação nela codificada;
- Seleção: processo de competição entre os fenótipos dos indivíduos da população;

Assim, o processo evolucionário descrito pode ser redesenhado em uma linguagem computacional, em que os AEs representam um conjunto de técnicas desenvolvidas a partir do paradigma da Teoria Darwiniana [20]. Portanto, o uso dos modelos e princípios do *Neodarwinismo* se tornou suporte teórico e fonte de inspiração para desenvolvimento da maioria dos diferentes métodos evolutivos existentes.

Embora a origem da computação evolucionária tenha sido originalmente a partir da década de 50, esse campo passou a receber mais destaque por volta dos anos 70, quando um grande crescimento no número de publicações foi iniciado, começando pelos trabalhos de Holland, Fogel, Rechenberg, Schwefel, entre outros. Posteriormente, as pesquisas das décadas de 80 e 90, tais como Goldberg, Fogel, Mitchell, e vários outros contribuíram para popularização e amadurecimento dessa família de métodos, constituindo-se atualmente uma área estabelecida e em crescente desenvolvimento. A ideia de aplicação da computação evolutiva na resolução de problemas de otimização se desenvolveu em múltiplas vias. Inicialmente, três subgrupos de métodos foram reconhecidos como principais representantes dessa classe, são eles:

- Programação evolucionária, tendo como pioneiro Fogel, Owens e Walsh;
- Algoritmos genéticos, iniciado pelos trabalhos de Holland e Goldberg;
- Estratégias evolutivas, idealizados por Rechenberg e Schwefel.

Outras subclasses de métodos desenvolvidas foram os algoritmos baseados em Sistemas Imunológicos Artificiais. Os pioneiros na investigações dessa área são Farmer, Bersini e Varela.

Devido ao funcionamento similar da maioria dos AEs, eles podem ser descritos por uma estrutura genérica. O Algoritmo 1 apresenta um esquema unificado para representação dessa classe de métodos.

Algoritmo 1: Estrutura Genérica dos Algoritmos Evolucionários [63].

Entrada: Tamanho da População (μ), Espaço de Busca (χ), Funções Objetivo e Restrições $(f(.), g(.), h(.))$, Critérios de Parada (Q)
Saída: Estimativa(s) de (x^*)
$\mathcal{P}(n) \rightarrow$ População da geração *n*
$\Phi(n) \rightarrow$ Valor da função da população na geração *n*
$\mathcal{S}(n) \rightarrow$ População selecionada na geração *n*
$\mathcal{Q}(n) \rightarrow$ População alterada na geração *n*
início
  $\mathcal{P}(0) \leftarrow$ Inicialização da População (μ, χ)
  $n \leftarrow 0$
  enquanto *não* Q **faça**
    $\Phi(n) \leftarrow$ Avaliação $(\mathcal{P}(n), f(.), g(.), h(.))$
    $\mathcal{S}(n) \leftarrow$ Seleção $(\mathcal{P}(n), \Phi(n))$
    $\mathcal{Q}(n) \leftarrow$ Variação $(\mathcal{S}(n))$
    $\mathcal{P}(n+1) \leftarrow$ Atualização $(\mathcal{P}(n), \mathcal{Q}(n))$
    $n = n + 1$
  fim enquanto
fim

Após a definição do problema, ou seja, determinar a função objetivo $f(.)$ e as restrições $g(.)$ e $h(.)$, um AE é inicializado a partir da configuração dos parâmetros de estratégia, tamanho da população, probabilidade de cruzamento e mutação, entre outros, que direcionam o funcionamento do método. Assim que é realizada a determinação desses parâmetros, é necessário a geração da população inicial por meio da distribuição aleatória dos indivíduos no espaço de busca. Assim, até que um critério de parada seja atendido, essa população passará pelas fases principais do processo evolutivo que correspondem à:

- Avaliação - os indivíduos da população corrente são avaliados em relação a função objetivo do problema;
- Seleção - competição entre os indivíduos da população, em que os mais aptos possuem maior probabilidade de serem selecionados para participar da fase de variação;
- Variação - operadores responsáveis pela diferenciação da população, tendo como objetivo exploração do espaço de busca;
- Atualização - atualização da população com novos indivíduos gerados.

Cada um dos diferentes AEs possuem características particulares, sendo que essas estão presentes nessas fases principais do processo evolucionário.

Conforme já dito, os AGs são uma classe particular dos AEs que têm suas origens nos trabalhos de Holland [64]. Esse método de busca, como a maioria dos algoritmos dessa família, são baseados nos mecanismos de seleção natural e genética [65]. Os AGs compreendem o campo dos AEs com maior número de aplicações e extensões. A forma padrão desses algoritmos surgiu na tentativa de Holland [64] explicar os processos adaptativos dos sistemas naturais e construir sistemas artificiais baseados em tais processos [65].

A robustez dos AGs representa um dos campos de pesquisa nessa área, sendo que, diferentemente dos métodos clássicos de otimização, essa classe de algoritmos tem características que os tornam extremamente robustos, entre as quais destacam-se [65]:

- Por conceito, os AGs adotam dois espaços diferentes: o espaço de busca e o espaço de soluções. O espaço de busca representa as soluções codificadas, ou genótipos, para o problema. Já o espaço de soluções denota as soluções reias, ou fenótipos, as quais serão avaliadas pela função de aptidão do problema. Assim sendo, tais algoritmos trabalham com a codificação de um conjunto de parâmetros de estratégia, não com os parâmetros em si;

- Uma população de soluções candidatas é responsável pela busca, não apenas um único ponto;

- A utilização de conhecimento prévio, tal como gradiente da função objetivo, não se faz necessário, o método é baseado no critério de desempenho ou aptidão;

- Regras de transição probabilística e não determinística são utilizadas;

Os AGs vem sendo aplicados com sucesso em uma gama de problemas, como por exemplo *bin-packing* [66], *job shop* [67], roteamento de veículos [19], localização de facilidades [68], entre outros. Similarmente a outras metaheurísticas, a obtenção da solução ótima não é garantida, porém o método é capaz de alcançar boas aproximações em um tempo computacional viável.

O Algoritmo 2 apresenta uma estrutura genérica para os AGs. Por se tratar de um AE, nota-se a semelhança existente entre a estrutura genérica dos AEs e dos AGs, sendo que a diferença é observada apenas na fase de variação da população. Assim, o funcionamento do mesmo é similar ao descrito anteriormente para os AEs, a única diferença é que a fase de variação dos AGs é composta por dois operadores: cruzamento e mutação.

Um AG tipicamente inicia o processo de otimização com uma população inicial que, geralmente, é escolhida aleatoriamente. Essa população é submetida a um processo iterativo

no qual a qualidade dos indivíduos é representada por uma função de aptidão. Essa função é fundamental na seleção de indivíduos que participarão dos processos de cruzamento e mutação, uma vez que o processo de seleção é estocástico e os indivíduos mais aptos possuem maior probabilidade de serem selecionados. Portanto, a definição inadequada da função de aptidão pode levar ao fracasso do algoritmo. Após os procedimentos de variação, responsáveis pela exploração do espaço de busca, a população é atualizada com os novos indivíduos gerados. Até que um critério de parada seja atendido, a população é submetida ao processo evolucionário.

Algoritmo 2: Estrutura Genérica dos Algoritmos Genéticos.

Entrada: Tamanho da População (μ), Espaço de Busca (χ), Funções Objetivo e Restrições $(f(.), g(.), h(.))$, taxa de cruzamento (p_c), taxa de mutação (p_m), Critérios de Parada (Q)
Saída: Estimativa(s) de (x^*)
$\mathcal{P}(n) \rightarrow$ População da geração *n*
$\Phi(n) \rightarrow$ Valor da função da população na geração *n*
$\mathcal{S}(n) \rightarrow$ População selecionada na geração *n*
$\mathcal{Q}(n) \rightarrow$ População alterada na geração *n*
início
  $\mathcal{P}(0) \leftarrow$ Inicialização da População (μ, χ)
  $n \leftarrow 0$
  enquanto *não* Q **faça**
    $\Phi(n) \leftarrow$ Avaliação $(\mathcal{P}(n), f(.), g(.), h(.))$
    $x^* \leftarrow$ Melhor $(\mathcal{P}(n), f(.))$
    $\mathcal{S}(n) \leftarrow$ Seleção $(\mathcal{P}(n), \Phi(n))$
    para *(*$i = 1$*; (*μ*); i++)* **faça**
      se *(*$p_c \geq$ *rand[0,1])* **então**
        $\mathcal{Q}(n) \leftarrow$ Cruzamento $(\mathcal{S}(n))$
      fim se
      se *(*$p_m \geq$ *rand[0,1])* **então**
        $\mathcal{Q}(n) \leftarrow$ Mutação $(\mathcal{S}(n))$
      fim se
    fim para
    $\mathcal{P}(n+1) \leftarrow$ Atualização $(\mathcal{P}(n), \mathcal{Q}(n))$
    $n = n + 1$
  fim enquanto
fim

2.2.2 Métodos evolucionários para o projeto de redes eixo-raio

Esta seção descreve as principais abordagens utilizadas para resolução do *PRERSCAS* baseadas em AEs, as quais serão comparadas com o AG proposto nesse trabalho. Assim, os

AGs de Topcuoglu et al. [53], Cunha e Silva [8] e Naeem e Ombuki-Berman [56] são detalhados a seguir.

2.2.2.1 AG proposto por Topcuoglu et al. - GAI

Nesta seção, o AG proposto no trabalho de Topcuoglu et al. [53] é descrito em detalhes, e um pseudo-código do mesmo é apresentado.

A representação das soluções desse método é feita por dois vetores de tamanho igual ao número de nós da rede: vetor de concentradores e vetor de atribuição. O vetor de concentradores é composto por 0's e 1's, onde o valor 1 indica que o nó é um concentrador e 0 denota que o nó não é concentrador. Já o vetor de atribuição representa a alocação dos nós, assim, se o nó i é associado ao concentrador k, o valor do mesmo no vetor de atribuição é igual à k. É importante lembrar que cada concentrador é associado a ele mesmo.

A geração da população inicial é composta por 3 fases distintas: determinação do número de concentradores; seleção dos nós que serão concentradores; alocação do nós ao concentrador mais próximo.

- Determinação do número de concentradores: para 75% da população, o número de concentradores é escolhido aleatoriamente entre [1 ,..., $\lfloor |N|/4 \rfloor$], já nos 25% de indivíduos restante o número de concentradores é determinado no intervalo [$\lceil |N|/4 \rceil$,..., $\lfloor |N|/2 \rfloor$];

- Seleção de quais nós serão concentradores: após a determinação do número de concentradores, é necessário escolher quais serão os nós onde os mesmos serão instalados. A escolha desses nós leva em consideração o fluxo de demanda total originado e destinado por cada nó. Dessa forma, para 75% dos indivíduos os concentradores são escolhidos entre $\lfloor 2/3 \rfloor$ dos nós com maior fluxo; já para o restante da população, todos nós podem ser escolhidos como concentradores;

- Alocação dos nós aos concentradores instalados: os nós não concentradores são alocados ao concentrador mais próximo.

A seleção dos indivíduos para participar das fases de cruzamento e mutação é feita por meio da estratégia roleta, ressaltando que o algoritmo trabalha com aplicação de elitismo.

A operação de cruzamento é feita com um ponto de corte no vetor de concentradores e no vetor de atribuição. Os filhos são gerados pela troca dos cromossomos dos pais, seguido por uma fase de correção, caso necessário. A fase de correção é requisitada quando um determinado nó estiver associado a outro nó que deixou de ser concentrador devido ao cruzamento; assim, tais nós são re-associados ao concentrador mais próximo.

O algoritmo propõe duas formas de mutação a serem aplicadas na atribuição dos nós não concentradores: deslocamento de alocação e troca de alocação. Na mutação deslocamento de

alocação, é feita a troca de alocação de um nó não concentrador, escolhido aleatoriamente, para outro concentrador, também selecionado de forma aleatória. O operador troca de alocação faz a escolha aleatória de dois nós não concentradores, sendo as alocações dos mesmos trocadas entre si. O Algoritmo 3 apresenta o pseudo-código do método descrito.

Algoritmo 3: AG proposto por Topcuoglu et al. [53]

Entrada: Tamanho da População (μ), taxa de cruzamento (p_c), taxa de mutação (p_m), Critérios de Parada (Q)
Saída: Estimativa(s) de (x^*)
$\mathcal{P}(n) \rightarrow$ População da geração *n*
$\Phi(n) \rightarrow$ Valor da função da população na geração *n*
$\mathcal{S}(n) \rightarrow$ População selecionada na geração *n*
$\mathcal{Q}(n) \rightarrow$ População alterada na geração *n*
início
  $\mathcal{P}(0) \leftarrow$ Inicialização da População (μ)
  $n \leftarrow 0$
  enquanto *não* Q **faça**
    $\Phi(n) \leftarrow$ Avaliação $(\mathcal{P}(n))$
    $x^* = \max$(aptidão)
    $\mathcal{S}(n) \leftarrow$ Seleção $(\mathcal{P}(n), \Phi(n))$
    para $i = 1$*;* μ*;* $i++$ **faça**
      $\mathcal{Q}(n) \leftarrow$ Cruzamento $(\mathcal{S}(n), p_c)$
      $\mathcal{Q}(n) \leftarrow$ Mutação $(\mathcal{S}(n), p_m)$
    fim para
    $\mathcal{P}(n+1) \leftarrow$ Atualização com elitismo $(\mathcal{P}(n), \mathcal{Q}(n))$
    $n = n + 1$
  fim enquanto
  retorne x^*
fim

2.2.2.2 AG proposto por Cunha e Silva - GAII

O trabalho de Cunha e Silva é descrito em detalhes nessa seção, assim como um pseudo-código do mesmo é apresentado.

A forma de representação das soluções é similar ao trabalho de Topcuoglu et al. [53], descrito anteriormente.

A população inicial é gerada aleatoriamente, sendo que cada nó tem 15% de probabilidade de se tornar concentrador da solução. Caso nenhum dos nós seja escolhido, um nó é sorteado aleatoriamente para ser concentrador da solução.

A seleção dos indivíduos para cruzamento e mutação é realizada pelo torneio binário, no operador implementado o melhor indivíduo é selecionado. O algoritmo trabalha com a ideia de elitismo.

Na operação de cruzamento, dois pontos de corte são efetuados no vetor de concentradores, sendo dois filhos gerados pela troca dos cromossomos centrais dos pais.

A mutação também ocorre no vetor de concentradores, nesse caso cada nó da solução tem uma probabilidade baixa de trocar de função, ou seja, deixar de ser concentrador ou se tornar concentrador.

Algoritmo 4: AG proposto por Cunha e Silva [8].

Entrada: Tamanho da População (μ), taxa de cruzamento (p_c), taxa de mutação (p_m), Critérios de Parada (Q)
Saída: Estimativa(s) de (x^*)
$\mathcal{P}(n) \rightarrow$ População da geração *n*
$\Phi(n) \rightarrow$ Valor da função da população na geração *n*
$\mathcal{S}(n) \rightarrow$ População selecionada na geração *n*
$\mathcal{Q}(n) \rightarrow$ População alterada na geração *n*
início
  $\mathcal{P}(0) \leftarrow$ Inicialização da População (μ)
  $n \leftarrow 0$
  enquanto *não* Q **faça**
    para $i = 1$*;* μ*;* $i++$ **faça**
      $\mathcal{Q}(n) \leftarrow$ Cruzamento ($\mathcal{P}(n)$, p_c)
      $\mathcal{Q}(n) \leftarrow$ Mutação ($\mathcal{P}(n)$, p_m)
      $\mathcal{P}(n) \leftarrow$ Alocação ao mais próximo ($\mathcal{Q}(n)$)
    fim para
    $\Phi(n) \leftarrow$ Avaliação ($\mathcal{P}(n)$)
    $\mathcal{P}(n) \leftarrow$ busca local ($\mathcal{P}(n)$)
    $x^* = \max$(aptidão)
    $\mathcal{S}(n) \leftarrow$ Seleção ($\mathcal{P}(n), \Phi(n)$)
    $\mathcal{P}(n+1) \leftarrow$ Atualização com elitismo ($\mathcal{P}(n), \mathcal{Q}(n)$)
    $n = n + 1$
  fim enquanto
  retorne x^*
fim

Após as fases de variação, os nós são alocados ao concentrador mais próximo e a aptidão dos indivíduos é calculada. Esse procedimento é seguido por uma busca local combinada com o SA. Nessa busca, todos os movimentos deslocamento de alocação e troca de alocação são realizados. Os movimentos de melhora são aceitos e, em caso de movimentos de piora, a aceitação dependerá da probabilidade calculada com base no critério de Metropolis [69]. Assim, o procedimento é realizado até que não haja re-alocações de melhoria e os movimentos de piora não sejam permitidos. O Algoritmo 4 apresenta o pseudo-código do AG descrito.

2.2.2.3 AG proposto por Naeem e Ombuki-Berman - GAIII

Esta seção descreve o AG de Naeem e Ombuki-Berman [56], assim como apresenta um pseudo-código do mesmo.

Nesse trabalho, Naeem e Ombuki-Berman propõem duas formas de codificação dos indivíduos: lista e grupos. A representação em lista é similar às descritas nas seções anteriores, exceto pela ausência do vetor de concentradores. Na outra forma de codificação, existem m grupos, onde m é o número de concentradores da solução. Esses grupos são compostos por um concentrador e os nós alocados ao mesmo.

A população inicial é gerada de forma aleatória, similarmente ao método de Topcuoglu et al. [53], sendo a criação de um indivíduo constituída de 3 fases: determinação do número de concentradores; escolha dos concentradores e alocação dos nós aos concentradores mais próximos. Ressalta-se que nessa estratégia o número máximo e mínimo de concentradores de uma solução é, respectivamente, $\lfloor |N|/2 \rfloor$ e 2.

Os indivíduos são selecionados pelo procedimento de torneio com 4 soluções, sendo os 2 melhores escolhidos para participarem da fase de variação. Cabe destacar que a forma de elitismo aplicada substitui 4 indivíduos escolhidos aleatoriamente pelos 4 melhores indivíduos da população corrente.

Os autores propõem 3 operações de cruzamento: troca de múltiplos grupos; troca de dois grupos e cruzamento com rota de melhor custo. Todos cruzamentos necessitam de uma fase de correção, pois as soluções geradas podem ser inviáveis, como por exemplo: alguns nós podem pertencer a mais de um grupo; alguns nós podem não estar em nenhum grupo; solução contendo dois grupos com mesmo concentrador. Nesses operadores, os filhos substituem os pais em caso de melhora na função objetivo.

Além das mutações já descritas, deslocamento de alocação e troca de alocação, o AG propõe o operador troca de função. Nesse operador, um concentrador e um nó não concentrador são escolhidos para trocarem de função, ou seja, o concentrador será rebaixado à não concentrador e o nó não concentrador será promovido à concentrador. O Algoritmo 5 apresenta o pseudo-código do método descrito.

Algoritmo 5: AG proposto por Naeem e Ombuki-Berman [56].

Entrada: Tamanho da População (μ), taxa de cruzamento (p_c), taxa de mutação (p_m), Critérios de Parada (Q)
Saída: Estimativa(s) de (x^*).
$\mathcal{P}(n) \rightarrow$ População da geração *n*
$\Phi(n) \rightarrow$ Valor da função da população na geração *n*
$\mathcal{S}(n) \rightarrow$ População selecionada na geração *n*
$\mathcal{Q}(n) \rightarrow$ População alterada na geração *n*
início
  $\mathcal{P}(0) \leftarrow$ Inicialização da População (μ)
  $n \leftarrow 0$
  enquanto *não* Q **faça**
    $\Phi(n) \leftarrow$ Avaliação ($\mathcal{P}(n)$)
    $x^* = \max$(aptidão)
    $\mathcal{S}(n) \leftarrow$ Seleção ($\mathcal{P}(n), \Phi(n)$)
    para $i = 1$*;* μ*;* $i++$ **faça**
      $\mathcal{Q}(n) \leftarrow$ Cruzamento ($\mathcal{S}(n)$, p_c)
      $\mathcal{Q}(n) \leftarrow$ Mutação ($\mathcal{S}(n)$, p_m)
    fim para
    $\mathcal{P}(n+1) \leftarrow$ Atualização com elitismo ($\mathcal{P}(n), \mathcal{Q}(n)$)
    $n = n + 1$
  fim enquanto
  retorne x^*
fim

2.3 Conclusão

Esse capítulo apresentou importantes conceitos sobre o *PRER* e forneceu uma contextualização histórica do mesmo. Uma descrição das diferentes variantes do problema, assim como dos métodos encontrados na literatura para resolução do mesmo foi realizada. Esse capítulo também contemplou uma revisão bibliográfica dos AEs, a qual contém o detalhamento dos principais métodos evolutivos da literatura para resolução do problema em questão.

As diferentes abordagens propostas para resolução do problema são descritas no próximo capítulo.

Capítulo 3

Algoritmos Genéticos para o Projeto de Redes Eixo-Raio

O presente capítulo apresenta o AG proposto para a resolução do *PRERSCAS*. Uma estratégia eficiente para geração da população inicial, assim como novos operadores de mutação e cruzamento específicos para o problema serão descritos. Além disso, discute-se as buscas locais, bem como o método de decida em vizinhança variável implementados para o problema.

3.1 Introdução

A utilização de métodos para resolução de problemas combinatórios é restringida de forma considerável. Por exemplo, a utilização de algoritmos determinísticos voltados para otimização não-linear de espaços contínuos é impraticável, pois os mesmos dependem do cálculo da derivada da função que, por definição, não existe no espaço discreto onde tal problema é definido. Técnicas como *branch-and-bound* [70] ou *branch-and-cut* [71] garantem a obtenção da solução ótima, porém se torna inviável em grande parte dos casos práticos, problemas de grande dimensão, pelo alto esforço computacional requerido.

Devido a inaplicabilidade desses e de outros métodos clássicos, um AG eficaz e eficiente na resolução do *PRERSCAS* é proposto, sendo essa escolha justificada pelas características favoráveis de tal abordagem:

- Não dependem de premissas matemáticas básicas fortes como linearidade, diferenciabilidade, convexidade ou unimodalidade;
- São aplicáveis em grande parte dos problemas práticos, pois permitem adaptações estruturais nos casos em que a estrutura tradicional é inaplicável;

- São eficientes na busca de ótimos globais em problemas multimodais [72].

Além disso, os AGs são famosos pelo alto grau de flexibilidade presente em sua estrutura, podendo ser aplicado a diversos tipos de problemas conforme já citado nos capítulos anteriores.

Contudo, a forma tradicional de tais algoritmos é incapaz de ser aplicado a problemas combinatórios sem adaptações. Isso porque é extremamente difícil encontrar soluções viáveis para problemas definidos em espaço discreto, uma vez que os operadores tradicionais, mutação e cruzamento, geram muitas soluções que não são interessantes para o problema.

Formas para tratamento dessas restrições, como penalização ou substituição das soluções inviáveis, reduzem o desempenho do algoritmo. A penalização resulta em propagação de soluções inviáveis concentrando a busca em uma parcela pequena da população que não mapeia bem o espaço de soluções; já a substituição das soluções inviáveis insere novos indivíduos que não passaram por processos de melhoria, transformando o método em uma busca aleatória.

A utilização de codificação e operadores específicos para o problema é uma das forma de atenuar as limitações descritas acima. Porém, não é tarefa fácil encontrar uma codificação para os indivíduos de forma que possibilite o desenvolvimento de operadores eficientes. Essa dificuldade de representação implica diretamente no desenvolvimento de operadores específicos para o problema que sejam fiéis ao seu papel no processo evolutivo.

Outro ponto crítico dos AGs é a preservação da diversidade da população durante a evolução; pois, com ao decorrer das gerações os indivíduos tendem a ser alocados aos espaços promissores pela difusão dos genes dos indivíduos pertencentes a tais subespaços. Porém, nem sempre as regiões aparentemente mais promissoras da fase inicial contém o ótimo global. Esse fato leva o algoritmo a perder informações relevantes de outras áreas do espaço de busca, causando convergência prematura.

Por outro lado, a estratégia de busca adotada pelos AGs necessita de equilíbrio entre dois objetivos, de certa forma, conflitantes: manutenção das melhores soluções e a exploração de novas regiões do espaços de busca. Esses objetivos são representados no método através das fases de seleção e variação, cruzamento e mutação, respectivamente. A primeira implica em pressão seletiva por busca de qualidade resultando em diminuição da exploração do espaço, e a segunda é responsável por inserir diversidade na população.

O AG implementado nesse trabalho busca amenizar as dificuldades encontradas na utilização dos métodos evolutivos para problemas combinatórios, adotando uma codificação específica para problemas em redes e propondo operadores próprios para o problema.

Nas seções seguintes, as características básicas do AG implementado são descritas: representação da solução; função de aptidão; geração da população inicial; operador de seleção; operador de cruzamento; operador de mutação; busca locais; descida em vizinhança variável.

3.2 Representação da solução

Inicialmente os AGs utilizavam codificação binária para denotar a região de busca, sendo as soluções representadas por cadeias binárias de tamanho n, onde n é tão grande quanto necessário para a precisão requerida. Entretanto, outras formas de codificação podem ser utilizadas em função do tipo de problema. Em problemas combinatórios, por exemplo, é comum a utilização de vetores inteiros, e em espaço contínuo tem-se o uso de variáveis reais. Há também problemas que podem ser representados por uma codificação mesclada, combinando duas formas de codificação para representar as variáveis.

A escolha de uma boa codificação da solução é fator primordial para o sucesso dos AGs, e uma definição inadequada dos cromossomos pode causar problemas de convergência. Portanto, essa é uma das etapas críticas na implementação do método.

Neste trabalho, de forma semelhante ao AG de Topcuoglu et al. [53], o indivíduo é representado através de dois vetores que possuem tamanho igual ao número de nós da rede. Tem-se um vetor de concentradores utilizando a codificação binária e um vetor de atribuição que utiliza a codificação inteira. No vetor de concentradores, o valor 1 indica que o nó é um concentrador e o valor 0 significa que o nó não é concentrador. A codificação inteira é responsável por representar a atribuição dos nós não concentradores aos concentradores instalados; assim, se nó $i \in N$ é atribuído ao concentrador $k \in N$, então a posição i do vetor de atribuição tem o valor k. A Figura 3.1 exemplifica a representação das soluções.

Concentradores	0 1 0 0 0 1 0 0 0
Alocação	6 2 6 2 2 6 6 2 2

Figura 3.1. Representação dos indivíduos.

Nesse exemplo, o indivíduo possui 9 nós, tendo os nós 2 e 6 como concentradores e os demais como nós não concentradores. Como um concentrador é associado a ele mesmo, a associação dos nós não concentradores aos concentradores instalados se dá seguinte forma: os nós 1, 3 e 7 são atribuídos ao concentrador 6, e os nós 4, 5, 8 e 9 são associados ao concentrador 2.

3.3 Função de aptidão

A função de aptidão deve representar o quão aptos são os indivíduos, dados os requisitos do problema a ser otimizado. Algumas ressalvas devem ser feitas sobre tal função:

- Em etapas posteriores, tal como seleção, pode ser muito útil que a função de aptidão não assuma valores negativos. Por isso, ao se projetar um AG deve-se analisar a conveniência da transformação da função real [65];

- A ideia empregada na evolução sugere uma maximização dessa função de aptidão. Assim sendo, pode ser necessário aplicar uma transformação que converta um problema de minimização num problema de maximização.

De um modo geral, as funções de aptidão podem ser mono ou multiobjetivo, uni ou multimodais, discretas ou contínuas, suaves ou ruidosas, estáticas ou variantes com o tempo. Qualquer que seja o perfil dessa função, o fato é que a definição da função de aptidão constitui um ponto importante para obtenção de uma solução de qualidade.

A função de aptidão adotada no AG implementado é a própria função objetivo do problema representada pela equação (1.1), já que o operador de seleção permite a utilização da mesma obtendo bom desempenho.

3.4 Geração da população inicial

Após a definição da codificação e da função de aptidão, o processo evolutivo pode ser iniciado. Primeiramente, é necessário a geração da população inicial, ou seja, o conjunto de indivíduos a partir dos quais se dará o trabalho de evolução. Essa população tipicamente é inciada de forma aleatória no espaço de busca. Contudo, quando existe algum conhecimento *a priori* sobre o problema, este pode ser explorado de forma a gerar soluções mais eficientes. Além disso, em alguns casos, é possível introduzir soluções já conhecidas, geralmente vindas de outros métodos de busca, na população inicial.

No AG implementado, os indivíduos da população inicial são gerados com base na estratégia de construção do GRASP, que é uma metaheurística introduzida por Feo e Resende [73, 74]. De uma forma geral, duas fases fazem parte desse método: na primeira fase uma solução viável é gerada, podendo ser construída de forma aleatória, semi-gulosa ou gulosa; na segunda fase, a vizinhança dessa solução é explorada, por um algoritmo de busca local, com objetivo de encontrar melhores soluções. O método retorna a melhor solução encontrada ao longo das iterações, sendo interrompido assim que algum dos critérios de parada seja atendido. Uma revisão das aplicações de sucesso dessa metodologia pode ser encontrada no trabalho de Resende e Ribeiro [75].

Basicamente, na estratégia de geração da população inicial do AG proposto, que se baseia na fase de construção do GRASP, todos os indivíduos são construídos, inicialmente, com somente um concentrador instalado. Essa solução montada é submetida a um procedimento de

melhoria, em que um novo concentrador é adicionado à solução até que o custo total (equação (1.1)) dessa mudança não resulte em melhoria.

Considere a geração de um conjunto *S* de soluções viáveis, de forma que $s \in S$ possua estrutura de alocação $s(j) = k$, ou seja, nó $j \in N$ é atribuído ao concentrador k. Assim sendo, um indivíduo é composto por dois conjuntos O_s e C_s que representam, respectivamente, os concentradores instalados e os nós não concentradores, em que $O_s \bigcap C_s = \emptyset$ e $O_s \bigcup C_s = N$. Esse conjunto S de soluções é submetido ao procedimento de melhoria.

Nesse procedimento de melhoria, para cada nó não concentrador $j \in C_s$ de uma solução $s \in S$, calcula-se o custo marginal de redução no custo total β_j quando esse nó é configurado como concentrador e os demais nós são re-alocados ao concentrador mais próximo. O custo marginal é computado pela diferença de custo da solução depois e antes do procedimento. Percebe-se que, caso $\beta_j \geq 0$, a configuração do nó j como concentrador resulta em aumento no custo total, assim o nó j é adicionado à lista NL, que contém os nós descartados como candidatos a concentrador da solução em questão.

Após o cálculo de cada β_j, o custo marginal de redução máximo (β_{max}) e mínimo (β_{min}) são determinados entre os nós $j \notin NL$. O próximo passo é a determinação de uma lista restrita L de nós candidatos a concentrador segundo a seguinte regra: $L = \{j \in C_s, j \notin NL : \beta_j \leq \beta_{min} + \lambda(\beta_{max} - \beta_{min})\}$, onde $\lambda \in [0, 1]$ representa um parâmetro que controla o nível de aleatoriedade no processo de construção e o tamanho de L. Note que, se $\lambda = 0$ tem-se um procedimento guloso que constrói uma lista menor de candidatos, apenas os nós que possuem $\beta_j = \beta_{mim}$ são considerados; caso $\lambda = 1$ esse processo é totalmente aleatório formando uma lista maior de candidatos, todos nós que possuem $\beta_j \leq \beta_{max}$ são considerados; valores intermediários para λ determinam o grau de aleatoriedade do processo e o tamanho da lista de candidatos. Se existe candidato em L, um nó $l \in L$ é escolhido de forma aleatória para se tornar concentrador ($O_s \leftarrow O_s + \{l\}$ e $C_s \leftarrow C_s - \{l\}$), e os demais nós não concentradores são re-alocados ao concentrador mais próximo.

Portanto, para cada solução $s \in S$, nós não concentradores são selecionados através de um processo aleatório iterativo até que $L = \emptyset$. Ressalta-se que, em cada iteração, são computados, novos valores de β_j para todo nó não concentrador $j \in C_s, j \notin NL$, β_{max} e β_{min}, bem como monta-se novas listas L e NL. O Algoritmo 6 apresenta um pseudo-código do procedimento de melhoria.

A Figura 3.2 exemplifica a estratégia de construção implementada. Nessa Figura, na parte (a) tem-se uma solução construída inicialmente com somente um concentrador, sendo $O_s = \{3\}$ e $C_s = \{1\ 2\ 4\ 5\ 6\}$, que será submetida ao procedimento de melhoria. Assim, os custos marginais de redução, β_j, de configurar os nós $j \in C_s$ e $j \notin NL$ como concentrador foram calculados e são representados por $\beta = \{$-9; 5; inf; -1; -8; -7,5$\}$. É importante ressaltar que, $\beta_j =$ inf denota que o nó $j \in N$ é um concentrador da solução ou pertence à lista de

Algoritmo 6: Procedimento de melhoria do método de construção.

Entrada: Solução (s), Conjunto de concentradores (O_s), Conjunto de não concentradores (C_s), Parâmetro de controle (λ)
Saída: Solução (s)
L lista de nós candidatos a concentrador
NL lista de nós não candidatos a concentrador
βj custo marginal de redução de configurar j como concentrador
β_{max} e βmin, respectivamente, custos marginais máximo e mínimo
d_{ij} distância entre nós i e j
início
  enquanto $L \neq \emptyset$ **faça**
    para $j \in C_s$ *e* $j \notin NL$ **faça**
      Compute β_j
      se $\beta_j > 0$ **então**
        $NL \leftarrow NL + j$
      fim se
    fim para
    Determine β_{max}, β_{min}, considerando $j \in C_s, j \notin NL$
    $L = \{j \in C_s e j \notin NL : \beta_j \leq \beta_{min} + \lambda(\beta_{max} - \beta_{min})\}$
    se $L \neq \emptyset$ **então**
      Selecione aleatoriamente $l \in L$
      Faça $O_s \leftarrow O_s + \{l\}$ e $C_s \leftarrow C_s - \{l\}$
      para $j \in C_s$ **faça**
        $s(j) \leftarrow argmin_{k \in O_s}\{d_{jk}\}$
      fim para
    fim se
  fim enquanto
  retorne s
fim

nós descartados como concentrador da solução, $j \in NL$. Parte (b) mostra a configuração desses nós como concentradores, note que quando nó 2 é promovido à concentrador o custo total da solução não é reduzido, portanto o mesmo será adicionado à lista de nós não candidatos a concentrador $NL = \{2\}$ e não será testado na próxima iteração. Além disso, os valores de $\beta_{max} = -1$ e $\beta_{min} = -9$ são determinados, portanto, utilizando $\lambda = 0,2$, a lista restrita de candidatos é construída respeitando a restrição $\beta_j \leq -7,4$. Dessa forma, a lista restrita de candidatos é composta por $L = \{1\ 5\ 6\}$. Considerando a escolha aleatória do nó 5, tem-se uma nova solução com $O_s = \{3\ 5\}$ e $C_s = \{1\ 2\ 4\ 6\}$. Após a configuração do nó 5 como concentrador e a alocação dos nós ao concentrador mais próximo, o procedimento de melhoria é repetido. Assim, na parte (c) novos valores para β_j são calculados, $\beta = \{1;$ inf; inf; 3; inf; $2\}$, conforme percebido nenhum dos nós $j \in C_s$ e $j \notin NL$ reduzem a função objetivo da solução quando transformados em concentradores, logo $NL = \{1\ 2\ 4\ 6\}$. Consequentemente, a lista

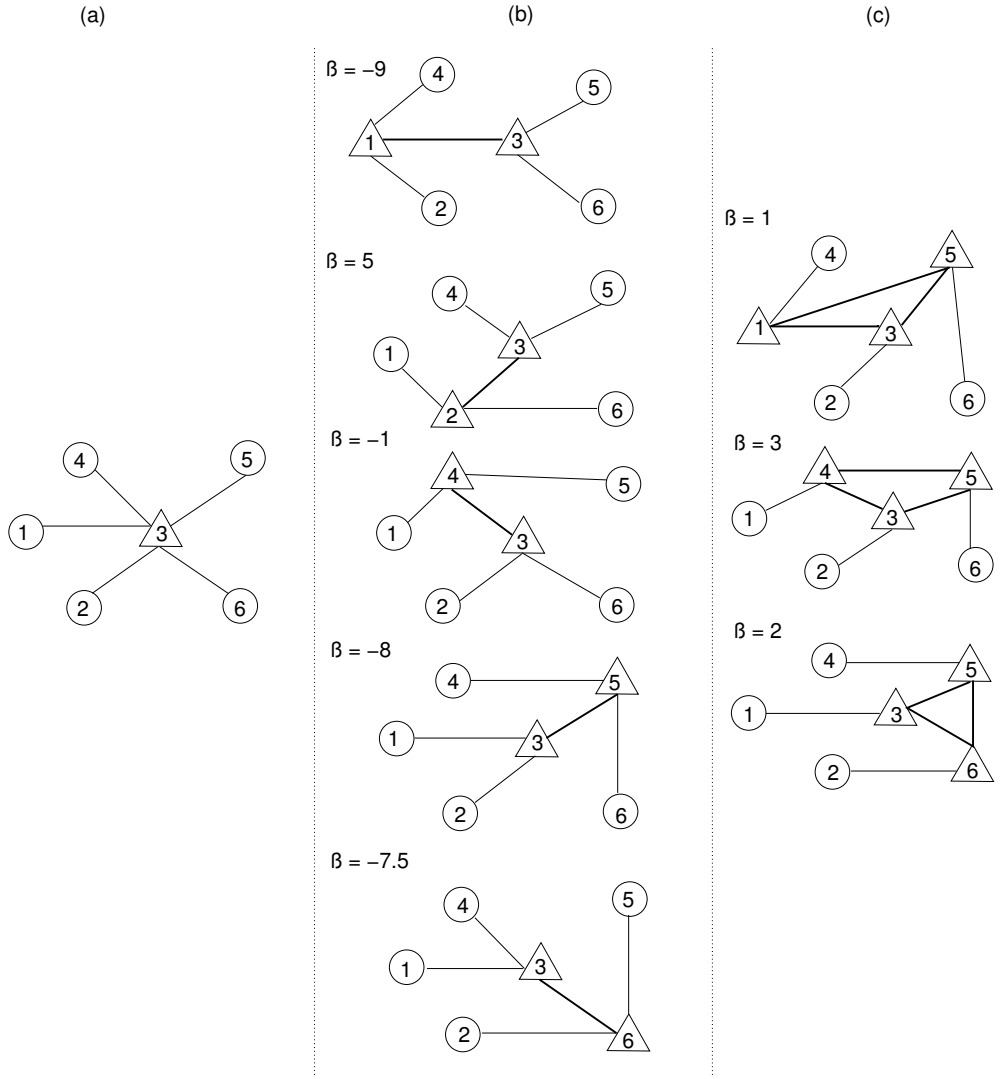

Figura 3.2. Procedimento de construção de soluções.

$L = \emptyset$ e não possuirá nós candidatos a se tornarem concentradores, portanto o procedimento é interrompido e a solução construída, $O_s = \{3\ 5\}$ e $C_s = \{1\ 2\ 4\ 6\}$, é retornada.

3.5 Operador de seleção

A teoria Darwiniana estabelece que os indivíduos mais bem adaptados têm maior possibilidade de reprodução e sobrevivência. Isso se reflete nos AGs através dos operadores de seleção e atualização.

O operador de seleção é o principal elemento que impulsiona os AGs na direção de regiões de busca promissoras, ou seja, é o responsável por efetuar a pressão seletiva na população de indivíduos.

A seleção é um operador estocástico que deve fornecer maior probabilidade de sobrevivência aos indivíduos que possuem maior desempenho. Para manter a diversidade da população e evitar convergência prematura, se faz necessário atribuir uma probabilidade não-nula de seleção aos indivíduos menos aptos. Existem várias possíveis estratégias de implementação do operador de seleção, algumas das mais conhecidas delas são descritas abaixo:

- Decimação - Nessa estratégia, após a determinação da aptidão dos indivíduos, aqueles que apresentam valores de desempenho superiores a um dado valor limite são selecionados, descartando-se os demais. Esse limite representa um valor intermediário entre o valor de aptidão mínimo e máximo, que é ponderado por um parâmetro que regula a pressão seletiva aplicada ao conjunto de pontos distribuídos no espaço de busca;

- Roleta - A roleta é uma estratégia bastante difundida nos AGs, nesse operador os indivíduos recebem uma probabilidade de serem selecionados de acordo com seus valores de aptidão. O nome roleta se justifica pelo fato que cada probabilidade associada aos indivíduos pode ser representada como uma seção em um gráfico de pizza. Assim, a seleção pode ser realizada através da geração de um número aleatório, $0 \leq n \leq 1$, com distribuição uniforme, possibilitando determinar qual região do gráfico ou indivíduo será sorteado. Devido a seu processo de construção, essa estratégia apresenta uma forte tendência de favorecer *superindivíduos*, isto é, indivíduos que possuem valores de aptidão extremamente elevados em relação aos demais. Essa fato pode levar à convergência prematura do método;

- Torneio - Na seleção por torneio, k indivíduos são escolhidos aleatoriamente na população. As funções de aptidão de cada integrante desse grupo são comparadas entre si e aquele mais apto será selecionado. Porém, outra forma de implementar tal operador é possibilitar a escolha dos indivíduos menos aptos, dando aos mesmos menor probabilidade de serem escolhidos;

- Outras formas de implementação podem ser encontradas em literatura especializada [65].

O operador de seleção utilizado no trabalho é o torneio binário. Nesse operador, dois indivíduos são selecionados para competir entre si, sendo que aquele mais bem adaptado tem maior probabilidade de ser escolhido para fazer cruzamento e mutação. Devido à aleatoriedade do processo, a manutenção na diversidade da população não é deixada de lado. Além disso, o algoritmo implementado trabalha com o conceito de elitismo, sendo o melhor indivíduo de cada geração inserido na população da geração seguinte. O Algoritmo 7 apresenta o procedimento de seleção por torneio binário.

Algoritmo 7: Seleção por torneio binário.

Entrada: Tamanho da população (μ), Soluções (S), Função Objetivo ($f(.)$), Probabilidade de escolher melhor indivíduo (p_{melhor})
Saída: Soluções selecionadas (Sel)
início
  para $i = 1$ *até* μ **faça**
    Selecionar s1 e s2 de S
    se $f1(.) < f2(.)$ **então**
      $s_{melhor} \leftarrow s1$
      $s_{pior} \leftarrow s2$
    fim se
    senão
      $s_{melhor} \leftarrow s2$
      $s_{pior} \leftarrow s1$
    fim se
    se $rand \leq p_{melhor}$ **então**
      $Sel_i \leftarrow s_{melhor}$
    fim se
    senão
      $Sel_i \leftarrow s_{pior}$
    fim se
  fim para
  retorne Sel
fim

3.6 Operadores de cruzamento

Em um processo evolucionário, a etapa de cruzamento é possivelmente a mais importante. O operador de cruzamento utilizado nos AGs é abstraído de uma metáfora biológica da recombinação, esse fenômeno é o responsável pelo embaralhamento ou variabilidade da informação genética durante a meiose de uma célula.

Após o processo de seleção da população, os indivíduos selecionados passarão pela etapa de cruzamento, ou recombinação, que irá gerar novas soluções a partir do material genético dos mesmos. Ou seja, o operador de cruzamento fica responsável pela combinação das características dos pais, permitindo que as próxima gerações herdem tais características.

Assim como na seleção, existem diversas formas de implementação do operador de cruzamento, principalmente para problemas definidos em espaço contínuo: cruzamento com 1 ponto de corte, cruzamento uniforme, entre outros que podem ser encontrados em [65]. Porém, como o problema abordado é de natureza combinatória definido em espaço discreto, as formas de cruzamento são restringidas. Por exemplo, operadores que trabalham com variáveis reais, como cruzamento convexo, não são adequados para tal problema. Portanto, existe a necessidade de

desenvolvimento de operadores específicos para o problema.

O operador de cruzamento depende da eficiência na propagação das boas características entre os indivíduos da população. No *PRERSCAS* as características de um indivíduo são representadas pela combinação dos concentradores e a atribuição dos nós aos mesmos. Com base nessas observações, três operadores de cruzamento, específicos para o problema, foram desenvolvidos: cruzamento multi-pais; cruzamento GRASP e cruzamento de grupos.

O algoritmo 8 mostra um pseudo-código para os operadores de cruzamento.

Algoritmo 8: Pseudo-código do operador de cruzamento.

Entrada: Soluções S, probabilidade de cruzamento multi-pais (p_{multi}), probabilidade de cruzamento de grupos (p_{grupos}), probabilidade de cruzamento GRASP (p_{grasp})
Saída: Soluções (S)
início
  $choose = rand[0, 1]$
  se $choose \leq p_{multi}$ **então**
    cruzamento multi-pais (S)
  fim se
  senão se $choose > p_{multi}$ *e* $choose \leq p_{grupos}$ **então**
    cruzamento grupos (S)
  fim se
  senão se $choose > p_{grupos}$ *e* $choose \leq p_{grasp}$ **então**
    cruzamento GRASP (S)
  fim se
  retorne S
fim

3.6.1 Cruzamento multi-pais

No algoritmo proposto, implementa-se o operador de cruzamento multi-pais utilizando três pais. Esse operador é desenvolvido com intuito de proporcionar maior variabilidade genética na população. Para isso, três pais (P1, P2 e P3) são selecionados para gerar três filhos (C1, C2 e C3). No cruzamento, dois pontos de corte, $r \in \{1, ..., \lfloor |N|/2 \rfloor\}$ e $t \in \{\lceil |N|/2 \rceil, ..., |N|\}$ gerados aleatoriamente, são aplicados nos vetores de concentradores e alocação. Assim, a construção dos filhos é feita da seguinte forma: C1, C2 e C3 recebem os primeiros r genes de P2, P1 e P3; enquanto os próximos $t - r$ genes são herdados de P1, P3 e P2; e finalmente o restante $|N| - t$ genes são obtidos de P3, P2 e P1, respectivamente. Esse processo é seguido por uma fase de ajuste em que indivíduos inviáveis são corrigidos. Nessa fase de ajuste, se um nó é atribuído a um nó não concentrador, então o mesmo é re-alocado ao concentrador mais próximo. A Figura 3.3 ilustra esse operador.

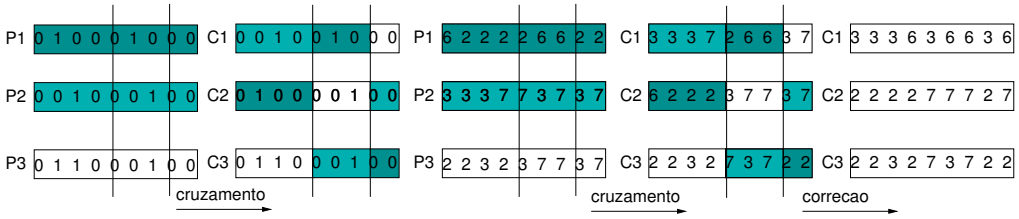

Figura 3.3. Cruzamento multi-pais.

Nessa Figura, os pais P1, P2 e P3 possuem como concentradores, respectivamente, os nós {2 6}, {3 7} e {2 3 7}. Já os filhos gerados C1, C2 e C3 possuem como concentradores, respectivamente, os nós {3 6}, {2 7} e {2 3 7}. Porém, nota-se que os filhos C1 e C2 precisaram passar pelo procedimento de ajuste, pois após o cruzamento possuiam nós associados a nós não concentradores. Como exemplo, considere o filho C1, os nós 4, 5 e 9 estão atribuídos aos nós 2 e/ou 7 que não são concentradores da solução em questão. Fato semelhante acontece com o filho C2, em que os nós 1, 5 e 8 estão alocados aos nós 3 e/ou 6 que não são concentradores desse indivíduo. Portanto, foi necessário que esses filhos gerados passassem pelo processo de correção descrito anteriormente.

O algoritmo 9 apresenta um pseudo-código para o cruzamento multi-pais.

Algoritmo 9: Pseudo-código do cruzamento multi-pais.

Entrada: Soluções (S)
Saída: Filhos C1, C2 e C3
início
  Selecione aleatoriamente os pais P1, P2 e P3 de S
  Determine $r \in \{1, ..., \lfloor |N|/2 \rfloor\}$ e $t \in \{\lceil |N|/2 \rceil, ..., |N|\}$
  para $i = 1$ *to* r **faça**
    C1(i) $\leftarrow$ P2(i)
    C2(i) $\leftarrow$ P1(i)
    C3(i) $\leftarrow$ P3(i)
  fim para
  para $i = r$ *to* t **faça**
    C1(i) $\leftarrow$ P1(i)
    C2(i) $\leftarrow$ P3(i)
    C3(i) $\leftarrow$ P2(i)
  fim para
  para $i = t$ *to* $|N|$ **faça**
    C1(i) $\leftarrow$ P3(i)
    C2(i) $\leftarrow$ P2(i)
    C3(i) $\leftarrow$ P1(i)
  fim para
  retorne C1, C2 e C3
fim

3.6.2 Cruzamento de grupos

No cruzamento de grupos, proposto por Naeem e Ombuki-Berman [56], os filhos são gerados pela troca de múltiplos grupos entre os pais, sendo que um grupo é composto por 1 concentrador e os nós alocados à ele. É importante lembrar que, podem ser selecionados de 1 à k - 1, k é o número de concentradores, grupos para serem trocados entre as soluções. Esse operador de cruzamento é seguido por uma fase de re-alocação em que indivíduos inviáveis são corrigidos, tal como descrito na seção anterior. As inviabilidades que podem ser geradas são: herdar um mesmo concentrador de ambos pais; nós com alocação duplicada e nós sem alocação. Se o filho gerado herdar um mesmo concentrador de ambos pais, as alocações desse concentrador são unidas no filho. Caso um nó possua alocação duplicada ou esteja sem alocação, o mesmo é atribuído ao concentrador mais próximo. A Figura 3.4 exemplifica o cruzamento de

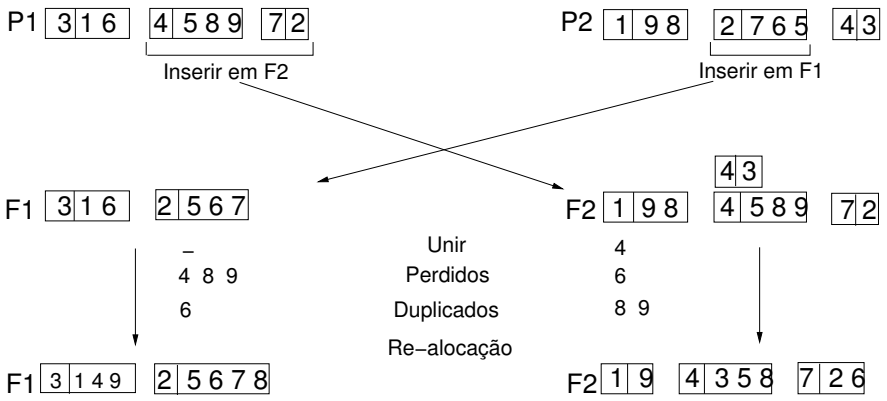

Figura 3.4. Cruzamento de grupos.

grupos. Nessa operação de cruzamento, os pais selecionados, P1, $O_{P1} = \{3\ 4\ 7\}$ e $C_{P1} = \{1\ 2\ 5\ 6\ 8\ 9\}$, e P2, $O_{P2} = \{1\ 2\ 4\}$ e $C_{P2} = \{3\ 5\ 6\ 7\ 8\ 9\}$, geram dois filhos trocando os grupos, concentrador e alocações, entre si. Assim, o filho F1 recebe os grupos do concentrador 3 de P1 e do concentrador 2 de P2; já F2 recebe os grupos dos concentradores 4 e 7 de P1 e dos concentradores 1 e 4 de P2. Percebe-se que inviabilidades são geradas nessa operação. A solução F2 herda grupos com mesmo concentrador, 4, de ambos pais, então as alocações desse concentrador são unidas no filho. Além disso, tem-se a duplicidade e/ou perca de alocação de alguns nós não concentradores. Por exemplo, em F1 os nós 4, 8 e 9 estão sem alocação e o nó 6 possui alocação duplicada, assim esses nós são re-alocados ao concentrador mais próximo; fato semelhante acontece em F2, onde o nó 6 não possui alocação e os nós 8 e 9 estão com alocação duplicada, portanto esses nós são re-alocados ao concentrador mais próximo. Dessa forma, gera-se os filhos F1, $O_{F1} = \{2\ 3\}$ e $C_{F1} = \{1\ 4\ 5\ 6\ 7\ 8\ 9\}$, e F2, $O_{F2} = \{1\ 4\ 7\}$ e $C_{F2} = \{2\ 3\ 5\ 6\ 8\ 9\}$.

3.6.3 Cruzamento GRASP

No cruzamento GRASP, dois pais são selecionados para geração de 1 filho. Nesse operador, os concentradores de ambas soluções são unidos no filho e os demais nós são atribuídos ao concentrador mais próximo. Essa etapa é seguida por uma fase de remoção de concentradores da solução. Nessa fase de remoção, o procedimento realizado é similar ao método de construção de soluções, porém, ao invés de testar a inserção de um concentrador na solução, no operador testa-se a retirada de concentradores.

Na etapa de remoção de concentradores, para cada nó concentrador $j \in O_s$ da solução, calcula-se o custo marginal de redução no custo total β_j quando esse nó deixa de ser concentrador e os nós alocados à ele são re-alocados ao concentrador mais próximo. Percebe-se que, caso $\beta_j \geq 0$, a remoção do concentrador j resulta em aumento no custo total, assim o nó j é adicionado à lista ND, que contém os concentradores descartados para a remoção da solução em questão.

Após calculados os custos marginais de redução, β_j, quando cada nó concentrador $j \in O_s$ é configurado como não concentrador da solução, determina-se o custo marginal de redução máximo (β_{max}) e mínimo (β_{min}) entre os nós $j \notin ND$. A determinação de uma lista restrita D que contém os concentradores candidatos a serem removidos é realizada de forma similar ao procedimento de construção de soluções, e segue a seguinte regra: $D = \{j \in O_s, j \notin ND : \beta_j \leq \beta_{min} + \lambda(\beta_{max} - \beta_{min})\}$, onde $\lambda \in [0, 1]$ representa um parâmetro que controla o nível de aleatoriedade no processo de construção e o tamanho de D. Se existe candidato em D, um concentrador é escolhido de forma aleatória para ser removido, e os nós alocados à ele são re-alocados ao concentrador mais próximo.

Portanto, nós concentradores são selecionados através de um processo aleatório iterativo até que $D = \emptyset$. Ressalta-se que, em cada iteração, são computados, novos valores de β_j para todo nó concentrador $j \in O_s, j \notin NL$, β_{max} e β_{min}, bem como monta-se novas listas D e ND. O Algoritmo 10 apresenta um pseudo-código desse cruzamento.

A Figura 3.5 exemplifica a fase de união dos concentradores no cruzamento GRASP. Nessa Figura, o filho é gerado pela junção dos concentradores de dois pais e alocação dos nós não concentradores ao concentrador mais próximo. A operação de cruzamento é seguida pelo procedimento de remoção dos concentradores.

O procedimento de remoção é ilustrado pela Figura 3.6. Nessa Figura, na parte (a) tem-se a solução gerada pelo cruzamento, O_s = {1 3 4 5} e $C_s = \{2\ 6\}$, que será submetida ao procedimento de remoção de concentradores. Assim, os custos marginais de redução, β_j, de remover cada concentrador $j \in O_s$ e $j \notin ND$ foram calculados e são representados por $\beta =$ {-8; inf; -7; -2; -6; inf }. É importante ressaltar que, $\beta_j =$ inf denota que o nó $j \in N$ não é um concentrador da solução ou pertence à lista de nós descartados para remoção, $j \in ND$. Parte

Algoritmo 10: Cruzamento GRASP.

Entrada: Soluções (S), Conjunto de concentradores (O_s), Conjunto de não concentradores (C_s), Parâmetro de controle (λ)
Saída: Solução (s)
D lista de concentradores candidatos à remoção
ND lista de concentradores descartados para a remoção
βj custo marginal de redução de remover concentrador j
β_{max} e βmin, respectivamente, custos marginais máximo e mínimo
d_{ij} distância entre nós i e j
início
  Selecione s1 e s2 de S
  $O_s \leftarrow O_{s1} + O_{s2}$
  $C_s \leftarrow C_{s1} + C_{s2} - O_s$
  enquanto $D \neq \emptyset$ **faça**
    para $j \in O_s$ *e* $j \notin ND$ **faça**
      Compute β_j
      se $\beta_j > 0$ **então**
        $ND \leftarrow ND + j$
      fim se
    fim para
    Determine β_{max}, β_{min}, considerando $j \in O_s, j \notin ND$
    $D = \{j \in O_s e j \notin ND : \beta_j \leq \beta_{min} + \lambda(\beta_{max} - \beta_{min})\}$
    se $D \neq \emptyset$ **então**
      Selecione aleatoriamente $l \in D$
      Faça $O_s \leftarrow O_s - \{l\}$ e $C_s \leftarrow C_s + \{l\}$
      para $j \in C_s$ *e* $s(j) = l$ **faça**
        $s(l) \leftarrow argmin_{k \in O_s}\{d_{lk}\}$
      fim para
    fim se
  fim enquanto
  retorne s
fim

(b) mostra a remoção de cada um dos concentradores da solução. Assim sendo, os valores de $\beta_{max} = -2$ e $\beta_{min} = -8$ são determinados, portanto, utilizando $\lambda = 0,2$, a lista restrita de candidatos é construída respeitando a restrição $\beta_j \leq -5$. Dessa forma, a lista é composta por $L = \{1\ 3\ 5\}$. Considerando a remoção aleatória do concentrador 5, tem-se uma nova solução com $O_s = \{1\ 3\ 4\}$ e $C_s = \{2\ 5\ 6\}$. Após a configuração do nó 5 como não concentrador e a alocação dos nós ao concentrador mais próximo, o procedimento de remoção é repetido. Assim, na parte (c) novos valores para β_j são calculados, $\beta = \{$-4; inf; -1; -1; inf; inf $\}$. Portanto, uma nova lista de candidatos à remoção é montada, $L = \{\ 1\ \}$. Como a lista contém somente um candidato, o mesmo é escolhido para remoção, os nós não concentradores são alocados ao concentrador mais próximo e uma nova lista é montada. Conforme pode ser visto, na parte (d)

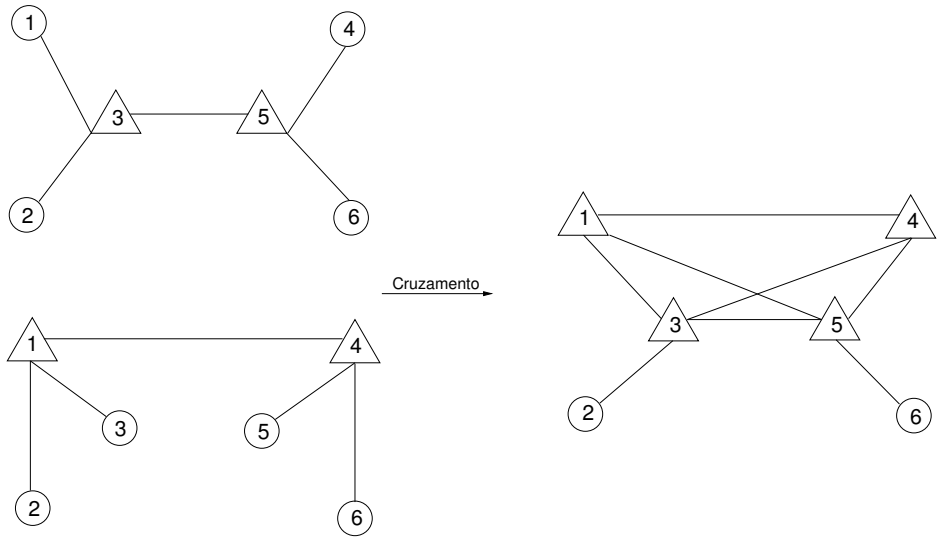

Figura 3.5. Cruzamento GRASP.

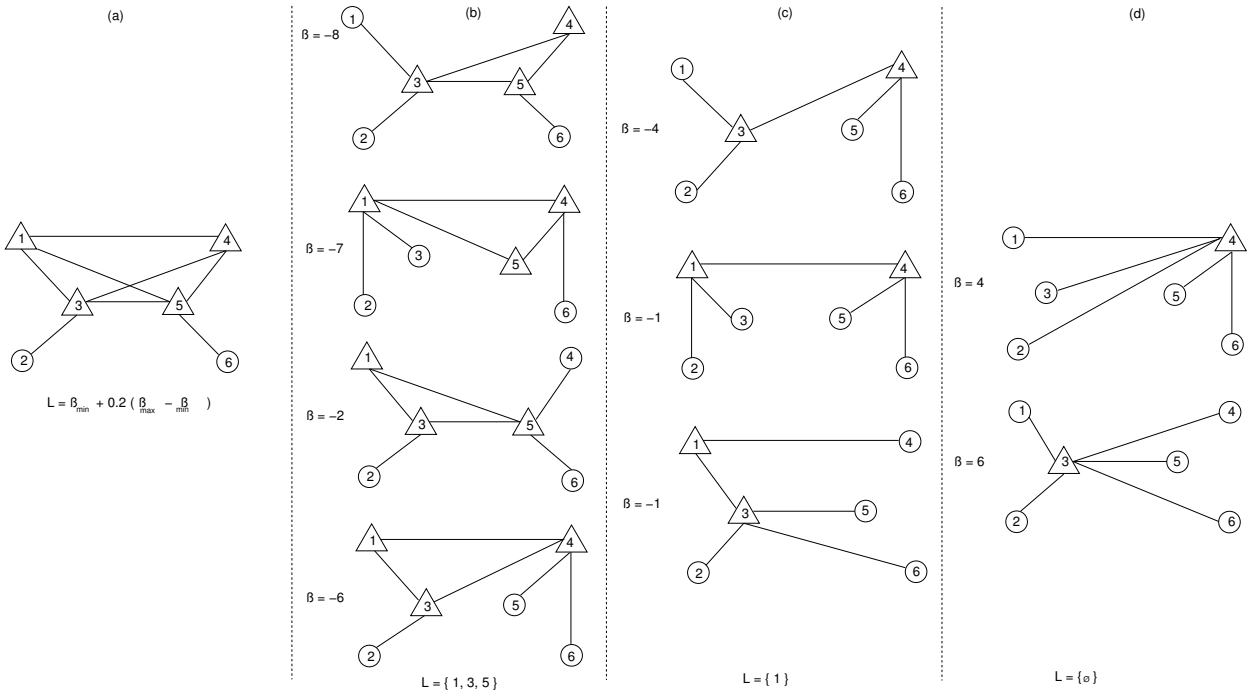

Figura 3.6. Fase de remoção.

nenhum dos concentradores $j \in O_s$ e $j \notin ND$ reduzem a função objetivo da solução quando removidos, logo $ND = \{3\ 4\}$. Consequentemente, a lista $L = \emptyset$ e não possuirá concentradores candidatos à remoção, portanto o procedimento é interrompido e a solução construída, $O_s = \{3\ 4\}$ e $C_s = \{1\ 2\ 5\ 6\}$, é retornada.

3.7 Operador de mutação

O operador de mutação tem o papel de fazer transformações esporádicas nos indivíduos através da inserção de novas informações nos indivíduos. Esse operador assegura a exploração de novas regiões no espaço de busca, possibilitando ao algoritmo escapar de bacias locais de atração.

De forma semelhante aos demais operadores, existem diversas estratégias de implementação para a mutação, principalmente para problemas com variáveis reais: mutação uniforme, mutação gaussiana, outros operadores podem ser encontradas em [65].

As mutações dos indivíduos do AG implementado são empregadas por meio de 5 operadores diferentes: deslocamento de alocação, troca de alocação, troca de função, inserir concentrador e remover concentrador.

Esses operadores, desenvolvidos especificamente para o problema, trabalham em 3 objetivos: alterar a alocação dos nós não concentradores aos concentradores instalados; alterar quais nós são concentradores; e alterar o número de concentradores. A Figura 3.7 exemplifica esses operadores.

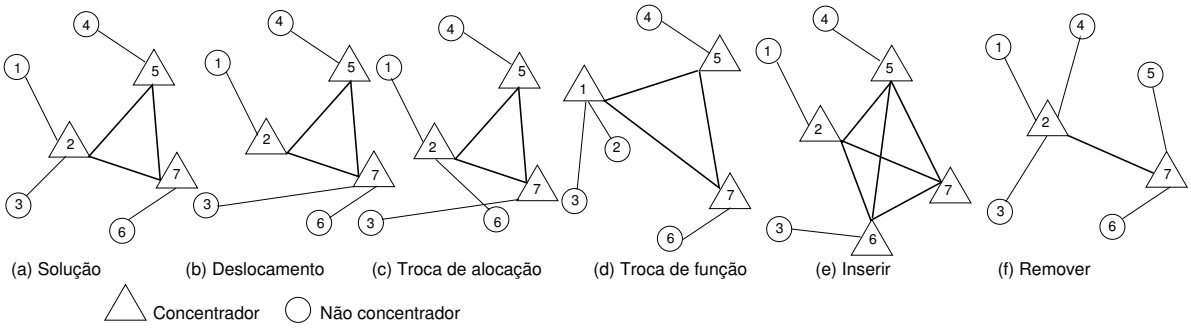

Figura 3.7. Operadores de mutação.

O algoritmo 11 mostra um pseudo-código para os operadores de mutação.

3.7.1 Mutação deslocamento de alocação

A mutação deslocamento de alocação trabalha com a ideia de trocar as alocações dos nós aos concentradores. Para isso, um nó não concentrador é escolhido aleatoriamente e realocado a outro concentrador, também escolhido de forma aleatória. Nota-se que, o operador necessita que o indivíduo tenha mais de um concentrador.

Para ilustrar a ideia de tal operador, considere as soluções, parte (a) e (b) da Figura 3.7. O nó não concentrador $i = \{3\}$ foi selecionado aleatoriamente para ser re-alocado a outro concentrador. Assim, como pode ser visto, o nó 3 associado ao concentrador 2 na parte (a),

Algoritmo 11: Pseudo-código do operador de mutação.

Entrada: Soluções (S), probabilidade de mutação deslocamento de alocação (p_s), probabilidade de mutação troca de alocação (p_e), probabilidade de mutação troca de função (p_{troca}), probabilidade de mutação remover concentrador (p_r), probabilidade de mutação inserir concentrador (p_{ins})
Saída: Solução mutada s_{mut}
início
  Selecione aleatoriamente s_{mut} of S
  $choose = rand[0, 1]$
  se $choose \leq p_s$ **então**
    mutação deslocamento de alocação (s_{mut})
  fim se
  senão se $choose > p_s$ *e* $choose \leq p_e$ **então**
    mutação troca de alocação (s_{mut})
  fim se
  senão se $choose > p_e$ *e* $choose \leq p_{troca}$ **então**
    mutação troca de função (s_{mut})
  fim se
  senão se $choose > p_{troca}$ *e* $choose \leq p_r$ **então**
    mutação remover (s_{mut})
  fim se
  senão se $choose > p_r$ *e* $choose \leq p_{ins}$ **então**
    mutação inserir (s_{mut})
  fim se
  retorne s_{mut}
fim

foi atribuído ao concentrador 7 na parte (b), após a mutação. O Algoritmo 12 apresenta o pseudo-código do procedimento descrito.

Algoritmo 12: Mutação deslocamento de alocação.

Entrada: Solução s
Saída: Solução s
início
  se $|O_s| > 1$ *e* $|C_s| \geq 1$ **então**
    Selecione aleatoriamente $j \in C_s$
    Selecione aleatoriamente $i \in O_s$ e $i \neq s(j)$
    $s(j) \leftarrow i$
  fim se
  retorne s
fim

3.7.2 Mutação troca de alocação

O objetivo da mutação troca de alocação é o mesmo do operador deslocamento de alocação, ou seja, trocar as alocações dos nós aos concentradores. Porém, na mutação troca de alocação os nós são re-alocados por meio de uma estratégia diferente da utilizada pelo operador descrito anteriormente. Assim, nesse operador dois nós não concentradores são escolhidos aleatoriamente e a alocação dos mesmos é trocada entre si. Para isso, o indivíduo deve possuir mais de um concentrador e no mínimo um nó não concentrador associado a cada um deles.

A estratégia de mudança de alocações adotada por esse operador é exemplificada pelas partes (a) e (c) da Figura 3.7. Nessa Figura, a solução selecionada sofrerá mutação na alocação dos nós não concentradores 3 e 6. Assim, na parte (a) os nós 3 e 6 atribuídos, respectivamente, aos concentradores 2 e 7, trocaram suas alocações. Dessa forma, na parte (c) o nó 3 passou a ser associado ao concentrador 7 e o nó 6 foi alocado ao concentrador 2. O Algoritmo 13 apresenta o pseudo-código do procedimento descrito.

Algoritmo 13: Mutação troca de alocação.

Entrada: Solução s
Saída: Solução s
início
  se $|O_s| > 1$ *e* $|C_s| > 1$ **então**
    Selecione aleatoriamente $j \in C_s$
    Selecione aleatoriamente $i \in C_s$ e $s(i) \neq s(j)$
    $aux \leftarrow s(i)$
    $s(i) \leftarrow s(j)$
    $s(j) \leftarrow aux$
  fim se
  retorne s
fim

3.7.3 Mutação troca de função

A ideia da mutação troca de função é mudar quais são os concentradores da rede, fazendo uma troca de função entre um concentrador e um nó não concentrador alocado ao mesmo. Para isso, dois nós da rede, um concentrador e um não concentrador, são escolhidos de forma aleatória para trocarem de função. Ou seja, o concentrador será rebaixado à não concentrador e o não concentrador será promovido à concentrador. Assim, é necessário que a rede tenha pelo menos um concentrador e um nó não concentrador.

Esse procedimento pode ser exemplificado pelas partes (a) e (d) da Figura 3.7. Esse exemplo mostra um indivíduo em que os nós 1 e 2 trocaram de função. Nota-se que, na parte (a)

o nó 2 é um concentrador e o nó 1 é um nó não concentrador, porém em (d), após sofrer mutação, o nó 1 passou a ser concentrador e 2 foi transformado em não concentrador. O Algoritmo 14 apresenta o pseudo-código do procedimento descrito.

Algoritmo 14: Mutação troca de função.

Entrada: Solução (s)
Saída: Solução (s)
d_{ij} distância entre nós i e j
início
  Selecione aleatoriamente $j \in O_s$
  Selecione aleatoriamente $i \in C_s$ e $s(i) = j$
  $O_s \leftarrow O_s - j$
  $C_s \leftarrow C_s + j$
  $O_s \leftarrow O_s + i$
  $C_s \leftarrow C_s - i$
  para $z \in C_s$ **faça**
    $s(z) \leftarrow argmin_{k \in O_s}\{d_{zk}\}$
  fim para
  retorne s
fim

3.7.4 Mutação inserir concentrador

O objetivo da mutação inserir concentrador é aumentar o número de nós que são concentradores da rede. Nesse processo, um nó não concentrador é escolhido aleatoriamente para se tornar concentrador. Para isso, é necessário que o número de concentradores existentes na rede seja menor que o número de nós da mesma.

As partes (a) e (e) da Figura 3.7 ilustram a operação realizada. Na parte (a) desse exemplo o indivíduo possui 3 concentradores, k ={2 5 7}, sendo que na parte (e) o nó 6 passou a ser concentrador da referida solução. Cabe ressaltar que, após a inserção desse concentrador os nós não concentradores são alocados ao concentrador mais próximo, considerando o concentrador inserido e o concentrador atual. O Algoritmo 15 apresenta o pseudo-código do procedimento descrito.

3.7.5 Mutação remover concentrador

O objetivo da mutação remover concentrador é o mesmo do operador descrito na seção anterior, ou seja, alterar o número de concentradores de uma solução. Porém, ao invés de adicionar um concentrador na solução, essa alteração é feita pela remoção do mesmo da solução.

Algoritmo 15: Mutação inserir concentrador.

Entrada: Solução (s)
Saída: Solução (s)
início
  se $|O_s| < |N|$ **então**
    Selecione aleatoriamente $j \in C_s$
    $C_s \leftarrow C_s - j$
    $O_s \leftarrow O_s + j$
    para $z \in C_s$ **faça**
      $s(z) \leftarrow argmin_{k \in O_s}\{d_{zk}\}$
    fim para
  fim se
  retorne s
fim

Esse operador é exemplificado pelas partes (a) e (f) da Figura 3.7. Nessa Figura, a parte (a) possui 3 concentradores, $k = \{2\ 5\ 7\}$, e após sofrer a mutação o nó 5 passou a ser nó não concentrador da rede, conforme pode ser visto na parte (f). É importante lembrar que o concentrador removido e os nós associados ao mesmo são atribuídos ao concentrador mais próximo. O Algoritmo 16 apresenta o pseudo-código do procedimento descrito.

Algoritmo 16: Mutação remover concentrador.

Entrada: Solução (s)
Saída: Solução (s)
início
  se $|O_s| > 1$ **então**
    Selecione aleatoriamente $j \in O_s$
    $O_s \leftarrow O_s - j$
    $C_s \leftarrow C_s + j$
    para $z \in C_s$ *e* $s(z) = j$ **faça**
      $s(z) \leftarrow argmin_{k \in O_s}\{d_{zk}\}$
    fim para
  fim se
  retorne s
fim

3.8 Algoritmo genético implementado

O Algoritmo 17 apresenta um pseudo-código do AG implementado. O AG proposto foi nomeado de GGA para efeito de comparação com os demais métodos da literatura.

Algoritmo 17: Pseudo-código do GGA.

Entrada: Tamanho da População (μ), probabilidade de escolher melhor indivíduo na seleção (p_{melhor}), taxa de cruzamento (p_{cruz}), probabilidade de cruzamento multi-pais (p_{multi}), probabilidade de cruzamento grupos (p_{grupos}), probabilidade de cruzamento GRASP (p_{grasp}), taxa de mutação (p_{mut}), probabilidade de mutação deslocamento de alocação (p_s), probabilidade de mutação troca de alocação (p_e), probabilidade de mutação troca de função (p_{troca}), probabilidade de mutação remover (p_r), probabilidade de mutação inserir (p_{ins}), critérios de parada (Q)
Saída: melhor indivíduo (s^*)
início
  Inicialize a população com μ indivíduos
  enquanto *não* Q **faça**
    Avalie a aptidão da população
    Determine $s^* = \max$ (aptidão)
    Selecione μ indivíduos por torneio binário
    para $i = 1$ *até* μ **faça**
      se *random* $\leq p_{cruz}$ **então**
        Aplique cruzamento (multi-pais p_{multi}, grupos p_{grupos}, GRASP p_{grasp})
      fim se
      se *random* $\leq p_{mut}$ **então**
        Aplique mutação (deslocamento de alocação p_s, troca de alocação p_e, troca de função p_{troca}, remover p_r, inserir p_{ins})
      fim se
    fim para
    Atualize a população com elitismo
  fim enquanto
  retorne s^*
fim

3.9 Buscas locais para o projeto de redes eixo-raio

Como os indivíduos do GGA não passam por um processo rigoroso de refinamento de soluções, como por exemplo aplicar busca local em indivíduos promissores da população, foram desenvolvidas diferentes buscas locais para o projeto de redes E-R que são apresentadas a seguir.

Técnicas de busca local em problemas de otimização são estratégias de refinamento baseadas em noção de vizinhança. Em linhas gerais, esses procedimentos partem de uma solução qualquer e exploram uma vizinhança definida com objetivo de encontrar melhores resultados.

As próximas subseções apresentam as buscas locais desenvolvidas para o projeto de redes E-R, explorando 4 estruturas de vizinhança diferentes: deslocamento de alocação, troca de função, remover concentrador e inserir concentrador.

3.9.1 Busca local deslocamento de alocação

Na busca local deslocamento de alocação, somente movimentos de re-alocação de nó não concentradores são permitidos. Portanto, o número de concentradores e os concentradores instalados não são alterados nesse procedimento. Essa busca local objetiva melhorar os custos de transmissão em uma rede. Nesse procedimento, todas as possíveis re-alocações de cada nó não concentrador $j \in C_s$ aos concentradores instalados $k \in O_s$ são testadas. Todos movimentos de re-associação que melhoram a solução são efetuados. Como essa busca local é computacionalmente custosa, ela é aplicada somente no melhor indivíduo de cada geração. Nos casos em que o melhor indivíduo permaneça o mesmo durante duas ou mais gerações, outra solução é sorteada aleatoriamente para ser submetida à busca local, substituindo a melhor solução, até então, se obtiver menor valor de função objetivo que a mesma. A Figura 3.8 apresenta todas os possíveis movimentos deslocamento de alocação do nó não concentrador 3.

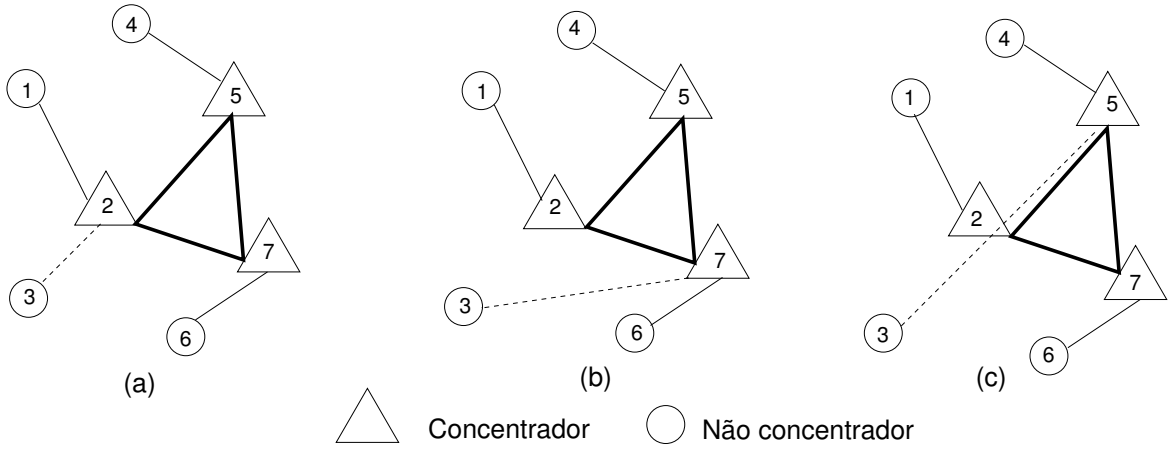

Figura 3.8. Busca local deslocamento de alocação.

Nessa Figura, o nó não concentrador 3 foi alocado aos diferentes concentradores da solução (2, 5 e 7). Na parte (a), o nó 3 está associado ao concentrador 2, em (b) o nó é atribuído ao concentrador 7, e a parte (c) ilustra a alocação desse nó ao concentrador 5. O Algoritmo 18 apresenta o pseudo-código do procedimento descrito.

3.9.2 Busca local troca função

Neste procedimento de busca local, o objetivo é alterar quais nós são concentradores da rede. Assim, para cada nó concentrador da solução $k \in O_s$, efetua-se a troca de função do mesmo com todos nós não concentradores alocados a ele $j \in C_s, s(j) = k$, e todos movimentos de melhora são efetuados. Pelo alto esforço computacional requerido, esse processo é realizado somente no melhor indivíduo de cada geração. Nos casos em que o melhor indivíduo

Algoritmo 18: Procedimento de busca local deslocamento de alocação.

Entrada: Solução (s), Função objetivo ($f(.)$)
Saída: Solução (s)
início
  para $j \in C_s$ **faça**
    $s^{'} \leftarrow s$
    para $i \in O_{s^{'}}$ **faça**
      $s^{'}(j) \leftarrow i$
      se $f(s^{'}) < f(s)$ **então**
        $s \leftarrow s^{'}$
      fim se
    fim para
  fim para
  retorne s
fim

permaneça o mesmo durante duas ou mais gerações, outra solução é sorteada aleatoriamente para ser submetida à busca local, substituindo a melhor solução, até então, se obtiver menor valor de função objetivo que a mesma. A Figura 3.9 mostra todas os possíveis movimentos de troca de função do nó concentrador 2 com os nós não concentradores alocados à ele.

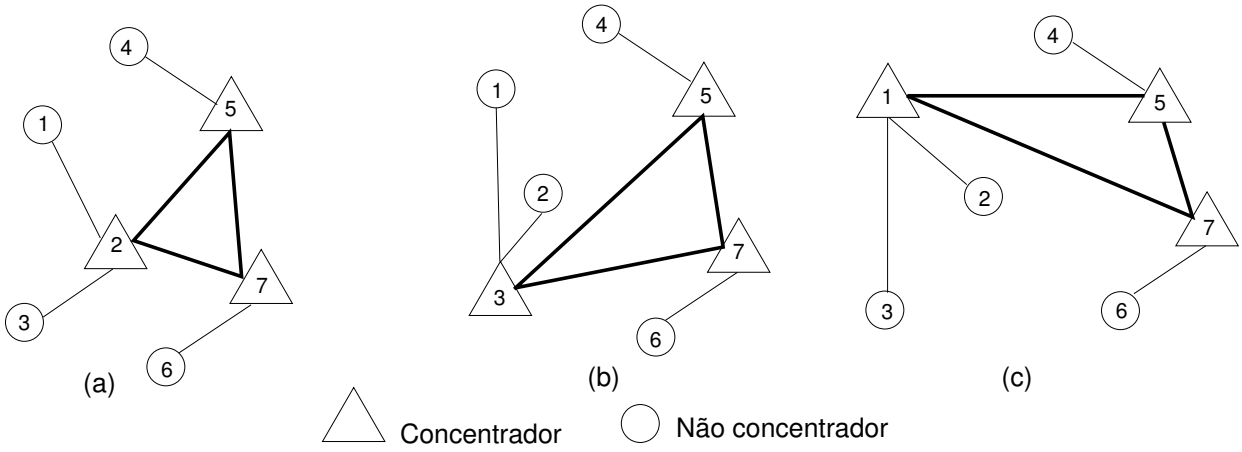

Figura 3.9. Busca local troca de função.

Nessa Figura, a parte (a) representa a solução em questão. Na pate (b), o nó 2 foi rebaixado como nó não concentrador e o nó 3 foi promovido à concentrador. Já na parte (c), a transformação em concentrador foi do nó 1. O Algoritmo 19 apresenta o pseudo-código do procedimento descrito.

Algoritmo 19: Procedimento de busca local troca de função.

Entrada: Solução (s), Função objetivo ($f(.)$)
Saída: Solução (s)
d_{ij} distância entre nós i e j
início
  para $j \in O_s$ **faça**
    para $i \in C_{s'}$ **faça**
      $s^{'} \leftarrow s$
      $O_{s'} \leftarrow O_{s'} - j$
      $C_{s'} \leftarrow C_{s'} + j$
      $O_{s'} \leftarrow O_{s'} + i$
      $C_{s'} \leftarrow C_{s'} - i$
      para $z \in C_{s'}$ **faça**
        $s^{'}(z) \leftarrow argmin_{k \in O_{s}{'}}\{d_{zk}\}$
      fim para
      se $f(s^{'}) < f(s)$ **então**
        $s \leftarrow s^{'}$
      fim se
    fim para
  fim para
  retorne s
fim

3.9.3 Busca local remover concentrador

O procedimento de busca local remover concentrador busca alterar o número de concentradores da rede, explorando a vizinhança de uma solução através da retirada de concentradores do indivíduo. Para isso, testa-se a remoção de cada um dos nós concentradores da solução $k \in O_s$, alocando os nós não concentradores ao concentrador mais próximo. Todas as remoções que melhorem o custo final são executadas no indivíduo em questão. Pelo alto esforço computacional requerido, esse processo é realizado somente no melhor indivíduo de cada geração. Nos casos em que o melhor indivíduo permaneça o mesmo durante duas ou mais gerações, outra solução é sorteada aleatoriamente para ser submetida à busca local, substituindo a melhor solução, até então, se obtiver menor valor de função objetivo que a mesma. A Figura 3.10 apresenta todas as possíveis remoções dos nós concentradores da solução.

Nessa Figura, a parte (a) representa a solução em questão. Na parte (b), o concentrador 2 foi transformado em não concentrador. Já na parte (c), o concentrador 5 foi rebaixado como não concentrador. Em (c), o concentrador 7 foi transformado em não concentrador. O Algoritmo 20 apresenta o pseudo-código do procedimento descrito.

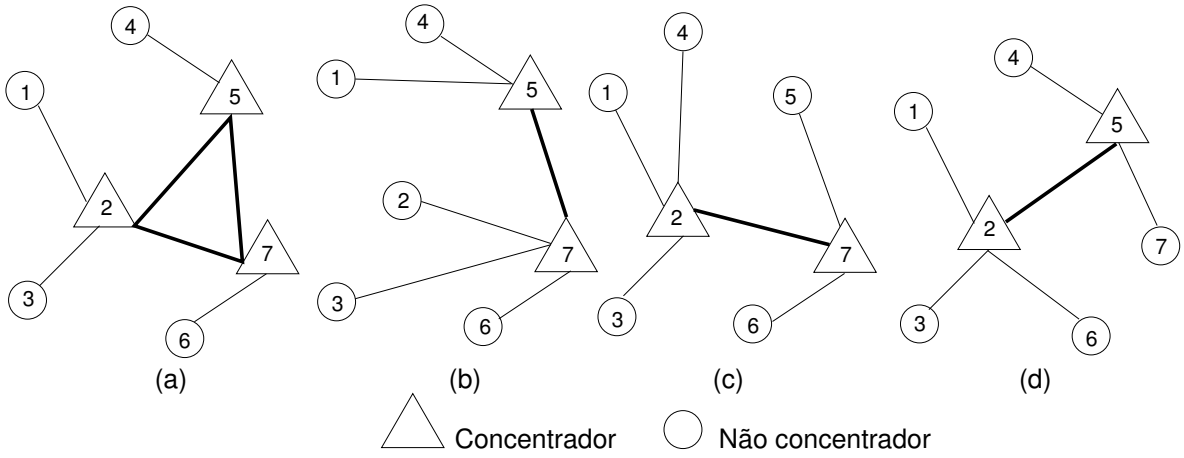

Figura 3.10. Busca local remover concentrador.

Algoritmo 20: Procedimento de busca local remover concentrador.

Entrada: Solução (s), Função objetivo ($f(.)$)
Saída: Solução (s)
início
  para $j \in O_s$ **faça**
    se $O_s > 1$ **então**
      $s^{'} \leftarrow s$
      $O_{s'} \leftarrow O_{s'} - j$
      $C_{s'} \leftarrow C_{s'} + j$
      para $z \in C_{s'}$ *e* $s^{'}(z) = j$ **faça**
        $s^{'}(z) \leftarrow argmin_{k \in O_{s'}}\{d_{zk}\}$
      fim para
      se $f(s^{'}) < f(s)$ **então**
        $s \leftarrow s^{'}$
      fim se
    fim se
  fim para
  retorne s
fim

3.9.4 Busca local inserir concentrador

O procedimento de busca local inserir concentrador também objetiva alterar o número de concentradores da rede, porém a exploração da vizinhança se dá pela inserção de concentradores no indivíduo. Assim, testa-se a inserção de cada um dos nós não concentradores $j \in C_s$ como concentradores da solução, alocando os nós não concentradores ao concentrador mais próximo. Todas as inserções que melhorem o custo final são executadas no indivíduo em questão. Pelo alto esforço computacional requerido, esse pro-

cesso é realizado somente no melhor indivíduo de cada geração. Nos casos em que o melhor indivíduo permaneça o mesmo durante duas ou mais gerações, outra solução é sorteada aleatoriamente para ser submetida à busca local, substituindo a melhor solução, até então, se obtiver menor valor de função objetivo que a mesma. A Figura 3.11 mostra todas as possíveis transformações dos nós não concentradores em concentradores da solução.

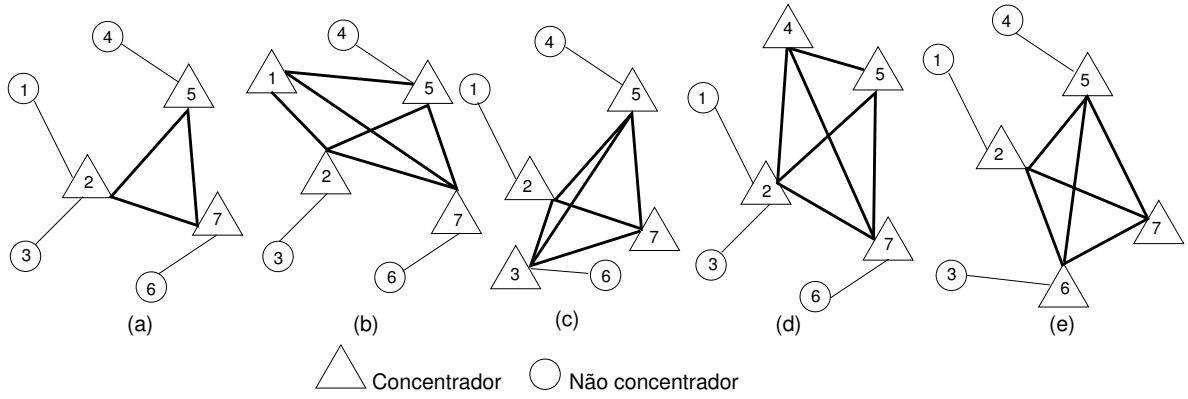

Figura 3.11. Busca local inserir concentrador.

Algoritmo 21: Procedimento de busca local inserir concentrador.

Entrada: Solução (s), Função objetivo ($f(.)$)
Saída: Solução (s)
início
  para $j \in C_s$ **faça**
    se $|O_s| < |N|$ **então**
      $s^{'} \leftarrow s$
      $O_{s'} \leftarrow O_{s'} + j$
      $C_{s'} \leftarrow C_{s'} - j$
      para $z \in C_{s'}$ **faça**
        $s^{'}(z) \leftarrow argmin_{k \in O_{s'}}\{d_{zk}\}$
      fim para
      se $f(s^{'}) < f(s)$ **então**
        $s \leftarrow s^{'}$
      fim se
    fim se
  fim para
  retorne s
fim

Nessa Figura, a parte (a) representa a solução em questão. A parte (b) exemplifica a inserção do nó 1 como concentrador da solução. Em (c) o nó 3 foi promovido a concentrador da

solução. Na parte (d), o nó 4 foi transformado em concentrador. Finalmente, a parte (e) mostra a promoção do nó 6 à concentrador. O Algoritmo 21 apresenta o pseudo-código do procedimento descrito.

3.10 Algoritmo genético com busca local implementado

O Algoritmo 22 apresenta um pseudo-código para o GGA combinado com busca local.

Algoritmo 22: Pseudo-código do GGA-busca local.

Entrada: Tamanho da População (μ), taxa de cruzamento (p_{cruz}), probabilidade de cruzamento multi-pais (p_{multi}), probabilidade de cruzamento grupos (p_{grupos}), probabilidade de cruzamento GRASP (p_{grasp}), taxa de mutação (p_{mut}), probabilidade de mutação deslocamento de alocação (p_s), probabilidade de mutação troca de alocação (p_e), probabilidade de mutação troca de função (p_{troca}), probabilidade de mutação remover (p_r), probabilidade de mutação inserir (p_{ins}), critérios de parada (Q)
Saída: melhor indivíduo (s^*)
início
  Inicialize a população com μ indivíduos
  enquanto *não* Q **faça**
    Avalie a aptidão da população
    Determine $s^* = \max$ (aptidão)
    Aplique busca-local (s^*)
    Selecione μ indivíduos por torneio binário
    para $i = 1$ *até* μ **faça**
      se *random* $\leq p_{cruz}$ **então**
        Aplique cruzamento (multi-pais p_{multi}, grupos p_{grupos}, GRASP p_{grasp})
      fim se
      se *random* $\leq p_{mut}$ **então**
        Aplique mutação (deslocamento de alocação p_s, troca de alocação p_e, troca de função p_{troca}, remover p_r, inserir p_{ins})
      fim se
    fim para
    Atualize a população com elitismo
  fim enquanto
  retorne s^*
fim

3.11 Descida em vizinhança variável para o projeto de redes eixo-raio

Cada uma das diferentes buscas locais propostas anteriormente exploram as soluções somente em uma vizinhança específica. Assim, com o objetivo de contornar essa questão, propõe-se uma técnica que explora o espaço de soluções através de trocas sistemáticas de estruturas de vizinhança: descida em vizinhança variável. Dessa forma, as diferentes buscas locais implementadas poderão ser exploradas.

Algoritmo 23: Pseudo-código do procedimento VND.

```
Entrada: Solução (s), Função objetivo (f(.)), estruturas de vizinhança (V)
Saída: Solução (s)
início
    v ← 1
    enquanto v ≤ r faça
        s* ← busca-local(s, V^v)
        se f(s*) < f(s) então
            s ← s*
            v ← 1
        fim se
        senão
            v ← v + 1
        fim se
    fim enquanto
    retorne s
fim
```

O procedimento de descida em vizinhança variável (VND), proposto por Mladevinoc e Hansen [76], é uma técnica de refinamento que consiste em explorar diferentes vizinhanças de uma solução com objetivo de reduzir o risco de ficar preso em ótimos locais de uma vizinhança específica. Durante essa exploração, somente movimentos de melhora são aceitos e, sempre que uma solução melhor é encontrada, o método recomeça a exploração na primeira estrutura de vizinhança considerada.

O algoritmo 23 apresenta um pseudo-código geral para VND, onde $V_s = \{V^1, V^2, ..., V^r\}$ é o conjunto das diferentes estruturas de vizinhança consideradas ($|V_s| = r$)).

A implementação desse procedimento para o projeto de redes E-R é justificada pelo fato das buscas locais aprimorarem uma solução somente por meio da exploração de uma vizinhança. Assim, o VND busca explorar todas estruturas de vizinhanças descritas: busca local deslocamento de alocação (V^1), busca local troca de função (V^2), busca local remover concentrador (V^3) e busca local inserir concentrador (V^4).

De forma semelhante às buscas locais descritas, o VND requer alto esforço computacional, assim, esse processo é realizado somente no melhor indivíduo de cada geração. Nos casos em que o melhor indivíduo permaneça o mesmo durante as duas gerações, outra solução é sorteada aleatoriamente para ser submetida ao VND, substituindo a melhor solução, até então, se obtiver menor valor de função objetivo que a mesma. O Algoritmo 24 apresenta um pseudo-código para o GGA-VND.

Algoritmo 24: Pseudo-código do GGA-VND.

Entrada: Tamanho da População (μ), taxa de cruzamento (p_{cruz}), probabilidade de cruzamento multi-pais (p_{multi}), probabilidade de cruzamento grupos (p_{grupos}), probabilidade de cruzamento GRASP (p_{grasp}), taxa de mutação (p_{mut}), probabilidade de mutação deslocamento de alocação (p_s), probabilidade de mutação troca de alocação (p_e), probabilidade de mutação troca de função (p_{troca}), probabilidade de mutação remover (p_r), probabilidade de mutação inserir (p_{ins}), critérios de parada ((Q))
Saída: melhor indivíduo (s^*)
início
  Inicialize a população com μ indivíduos
  enquanto *não* Q **faça**
    Avalie a aptidão da população
    Determine $s^* = \max$ (aptidão)
    Aplique VND (s^*)
    Selecione μ indivíduos por torneio binário
    para $i = 1$ *até* μ **faça**
      se *random* $\leq p_{cruz}$ **então**
        Aplique cruzamento (multi-pais p_{multi}, grupos p_{grupos}, GRASP p_{grasp})
      fim se
      se *random* $\leq p_{mut}$ **então**
        Aplique mutação (deslocamento de alocação p_s, troca de alocação p_e, troca de função p_{troca}, remover p_r, inserir p_{ins})
      fim se
    fim para
    Atualize a população com elitismo
  fim enquanto
  retorne s^*
fim

3.12 Conclusão

Ao longo desse capítulo realizou-se uma explanação das abordagens propostas para resolução do projeto de redes E-R com alocação simples. Portanto, as principais contribuições dessa dissertação estão presentes no mesmo. De uma forma geral, foi realizada uma discussão

detalhada a cerca do funcionamento dos métodos implementados, assim como foi apresentado o pseudo-código de cada um deles.

Primeiramente, foi apresentado o AG simples, o qual contém uma estratégia eficiente para geração da população inicial, e novos operadores de cruzamento e mutação capazes de evoluir a solução durante o processo evolucionário. O capítulo também contempla uma descrição das buscas locais, bem como do VND desenvolvidos para o projeto de redes E-R.

Os resultados obtidos pelas abordagens implementadas são descritos no próximo capítulo.

Capítulo 4

Resultados Computacionais

4.1 Introdução

O presente capítulo apresenta um conjunto de experimentos computacionais que foram realizados com o objetivo de obter o desempenho do AG proposto, nomeado GGA, e suas extensões (GGA com busca local e GGA com descida em vizinhança variável). De uma forma geral, a avaliação de desempenho é realizada por um estudo comparativo com outros três AGs considerados estado da arte na literatura, Topcuoglu et al. [53] (GAI), Cunha e Silva [8] (GAII), e Naeem e Ombuki-Berman [56] (GAIII), em relação a fase de construção, melhor solução encontrada e tempo de processamento para obtenção de uma solução alvo. Todos os algoritmos foram implementados em C++, e os experimentos foram executados em um computador com processador Core 2 Duo 2.2 GHz, 8 GB de memória RAM e sistema operacional M S Windows 7.

Após uma série de testes, apresentados no anexo A, os parâmetros do GGA foram configurados com: tamanho da população $\mu = 200$ indivíduos; parâmetro de controle da fase de construção $0,05 \leq \lambda \leq 0,2$; probabilidade de cruzamento $p_{cruz} = 0,9$; probabilidade de cruzamento multi-pais $p_{multi} = 0,5$; probabilidade de cruzamento de grupos $p_{grupos} = 0,4$; probabilidade de cruzamento GRASP $p_{grasp} = 0,1$; probabilidade de mutação $p_{mut} = 0,4$; probabilidade de mutação inserir $p_{ins} = 0,1$; probabilidade de mutação remover $p_r = 0,1$; probabilidade de mutação troca de função $p_{troca} = 0,3$; probabilidade de mutação troca de alocação $p_e = 0,25$; e probabilidade de mutação deslocamento de alocação $p_s = 0,25$. Além disso, os critérios de parada utilizados são: tempo de processamento limite de $|N|$ segundos ou obtenção de uma solução alvo.

4.2 Metodologia para apresentação de resultados

4.2.1 Introdução

Um problema comum em pesquisa operacional envolve a comparação de diversos métodos para determinar qual deles obtém desempenho superior na resolução de determinado problema. Duas possíveis formas de avaliar a efetividade de um algoritmo, cada uma fornecendo informações diferentes para auxiliar na tomada de decisão, são: análises teóricas e testes empíricos.

- Análise teórica: fornece características teóricas do algoritmo (pior e médio caso para tempo; pior e médio caso para limites de qualidade da solução);
- Testes empíricos: envolve implementação do método e avaliação de tempo de processamento, número de operações, qualidade da solução em problemas teste.

De acordo com Coffin e Saltzman [15], a literatura voltada para comparação de algoritmos em pesquisa operacional é muito esparsa e não existe um padrão a ser utilizado nesses casos. Trabalhos que comparam métodos são de alta variabilidade em relação a qualidade, e, muitas vezes, apresentam resultados obscuros.

Nessa seção é descrita a metodologia utilizada para apresentação de resultados desse trabalho. Tal abordagem é baseada no trabalho de Ribeiro et al. [77], que vem sendo amplamente utilizada na literatura para comparação de métodos heurísticos.

4.2.2 Instâncias teste

Uma das principais questões na condução de experimentos computacionais é a escolha das instâncias teste. Essas instâncias podem ser obtidas de diversas maneiras dependendo do problema abordado:

- Conjunto de dados reais: possuem dados originados de aplicações reais. Infelizmente, essa classe de dados dificilmente é obtida devido a diversos fatores, e mesmo quando são obtidos, muitas vezes, não representam todas as características de interesse do problema a ser abordado;
- Variações aleatórias de conjunto de dados reais: consiste em expandir, aleatoriamente, uma pequena base de dados real mantendo a macro estrutura atual da mesma;
- Base de dados online ou dados publicados: são dados benchmark publicados online. Esse tipo de dados é típico de problemas clássicos como caixeiro viajante (TSPLIB), sequenciamento de tarefas, programação de quadro de horários, entre outros;

- Dados gerados aleatoriamente: quando nenhum dos recursos citados anteriormente é fornecido, a alternativa restante é gerar dados de forma aleatória. Esse tipo de dados pode muitas vezes induzir à erros na investigação e/ou levar a algumas armadilhas [16].

4.2.2.1 Base de dados *Australian Post (AP)*

A base de dados AP, introduzida por Ernst et al. [33], é baseada em aplicações reais do serviço postal na Austrália. Instâncias de até 200 nós são derivadas desses dados, o que possibilita o estudo de problemas reais de grande escala. Muitos trabalhos da literatura [8, 34, 53, 56] utilizaram essa base de dados para medir o desempenho dos métodos propostos.

Diferentes valores de custo fixo para instalação de um concentrador em cada nó da rede são considerados, o que pode ser justificado devido a fatores geográficos, comerciais ou outros. Além disso, com objetivo de representar mais fielmente problemas reais, os fluxos de demanda entre os nós são asssimétricos ($w_{ij} \neq w_{ji}$).

É importante lembrar que os dados originais possuem apenas os primeiros 50 custos de instalação. Em função dessa limitação, custos fixos foram gerados para todos os problemas testes através de uma distribuição normal com média igual a f_o e variância de 40 % desse valor. Isso foi feito para representar a variação dos custos de instalação em problemas reais. O valor de f_o foi calculado como sugerido por Ebery [34], e representa a diferença normalizada entre um cenário que possui apenas um concentrador imaginário localizado no centro de demanda e outro considerando todos os pontos de demanda como concentradores. Além disso, conforme sugerido nesse mesmo trabalho, foi atribuído maior custo de instalação aos pontos que possuem maior demanda, o que dificulta a seleção desses pontos como concentradores.

As instâncias tratadas nesse trabalho são $|N| = \{10, 20, 30, ..., 100, 130, 150, 170, 200\}$ e fatores de desconto $\alpha = \{0,2; 0,4; 0,6; 0,8\}$ (economia de escala). Portanto, os 56 problemas teste gerados são representados por $APN{-}\alpha$, onde N representa o número de nós de demanda e α o fator de desconto. Cada um dos problemas teste foi executado 30 vezes por cada algoritmo utilizando diferentes sementes geradoras de números aleatórios, totalizando 1680 execuções. Entretanto, em cada execução, todos algoritmos utilizaram a mesma semente.

4.2.3 Métricas de desempenho

Diferentemente de algoritmos exatos, em que a principal métrica de desempenho é tempo e eficiência de convergência para solução ótima, existem duas questões fundamentais na avaliação de heurísticas: eficiência, em relação a tempo computacional, para obtenção de soluções; e qualidade das soluções encontradas. Existem diversas formas para avaliação da qualidade de soluções e tempo de processamento [15, 16].

Nos experimentos computacionais realizados nesse trabalho, as seguintes métricas de qualidade foram computadas: $BestValue$, $DevMed$, $DevMin$, $\#Best$ e $Score$. $BestValue$ representa a melhor solução encontrada entre todos os métodos considerados para uma dada instância. Dessa forma, calcula-se para cada método, a porcentagem relativa de desvio entre a melhor solução obtida pelo mesmo e o $BestValue$ da instância em questão. Considerando todas as execuções de cada método para todas instâncias, $DevMin$ e $DevMed$ representam, respectivamente, a média dos mínimos e a média das médias desse desvio. Além disso, $\#Best$ e $Score$ denotam, respectivamente, o número de instâncias em que o $BestValue$ foi encontrado pelo método e o número de métodos que obtiveram melhor resultado que o algoritmo em questão.

Considere, por exemplo, um experimento que visa comparar três métodos (A, B e C) na resolução de 3 instâncias ($I1$, $I2$ e $I3$). Para isso, cada método foi executado 3 vezes para cada instância. A Tabela 4.1 apresenta os resultados obtidos.

Tabela 4.1. Resultados dos métodos A, B e C.

Instâncias	A	B	C
	10	**9**	**9**
$I1$	12	11	**9**
	15	10	12
	14	**13**	**13**
$I2$	21	15	16
	18	14	15
	11	12	14
$I3$	13	15	**11**
	16	**11**	13

Nota-se que, o $BestValue$ das instâncias $I1$, $I2$ e $I3$ são, respectivamente, 9, 13 e 11. Portanto, o $\#Best$ dos algoritmos também pode ser determinado. Como essa métrica representa o número de instâncias em que o método encontrou o $BestValue$, os algoritmos A, B e C possuem $\#Best$ iguais à, respectivamente, 1, 3 e 3; pois, A só alcançou o $BestValue$ da instância $I3$, e os demais métodos obtiveram o $BestValue$ de todas as instâncias. Similarmente, o $Score$ de cada um dos métodos também é determinado. Essa métrica representa o número de métodos que obtém resultado melhor que o algoritmo em questão considerando todas instâncias, assim o $Score$ dos métodos A, B e C são, nessa ordem, 4, 0 e 0; pois, a melhor solução do algoritmo A foi pior que as de B e C nas instâncias $I1$ e $I2$, e os demais métodos não foram piores em nenhuma instância.

Tendo o $BestValue$ de cada instância, o desvio relativo de cada execução para o

$BestValue$ pode ser calculado de acordo com:

$$Dev = \frac{(FO - BestValue)}{BestValue} \tag{4.1}$$

Dessa forma, os desvios mínimos (DevMinInst) e médios (DevMedInst) dos métodos são calculados para cada instância. Os resultados são reportados na Tabela 4.2.

Tabela 4.2. Desvios mínimos e médios por instância.

	A		B		C	
Instância	DevMinInst	DevMedInst	DevMinInst	DevMedInst	DevMinInst	DevMedInst
I1	0,1111	0,3704	0,0000	0,1111	0,0000	0,1111
I2	0,0769	0,3590	0,0000	0,0769	0,0000	0,1282
I3	0,0000	0,2121	0,0000	0,1515	0,0000	0,1515

Assim, as médias dos desvios mínimos ($DevMin$) e médios ($DevMed$) são computadas para cada método. Os valores de $DevMin$ e $DevMed$, assim como as demais métricas são reportados na Tabela 4.3.

Nota-se que, os métodos B e C obtiveram resultados bem parecidos, e por sua vez superam os resultados de A.

Para medir o desempenho dos métodos em relação a tempo de processamento e convergência, realizou-se um experimento que mede a distribuição de probabilidade acumulada de um algoritmo alcançar uma determinada solução alvo. Para isso, uma instância é escolhida e uma solução alvo é determinada. Após 200 execuções do método, obtém-se a probabilidade acumulada do mesmo de alcançar o alvo dado um intervalo de tempo. Como exemplo, a Figura 4.1 ilustra a distribuição de probabilidade acumulada dos algoritmos B e C para a instância $I1$.

Como pode ser visto, o algoritmo B é capaz de alcançar o alvo em menor tempo de processamento. Aproximadamente no tempo de $0,5$ segundos, o algoritmo B tem mais de 90% de probabilidade de alcançar o alvo, enquanto o algoritmo C tem probabilidade quase nula. Esse fato demonstra que B converge em menor tempo de processamento para o alvo determinado da instância $I1$.

Tabela 4.3. Comparação final dos métodos.

Métricas	A	B	C
$Score$	4	0	0
$\#Best$	1	3	3
$DevMin$	0,0627	0,0000	0,0000
$DevMed$	0,3138	0,1132	0,1303

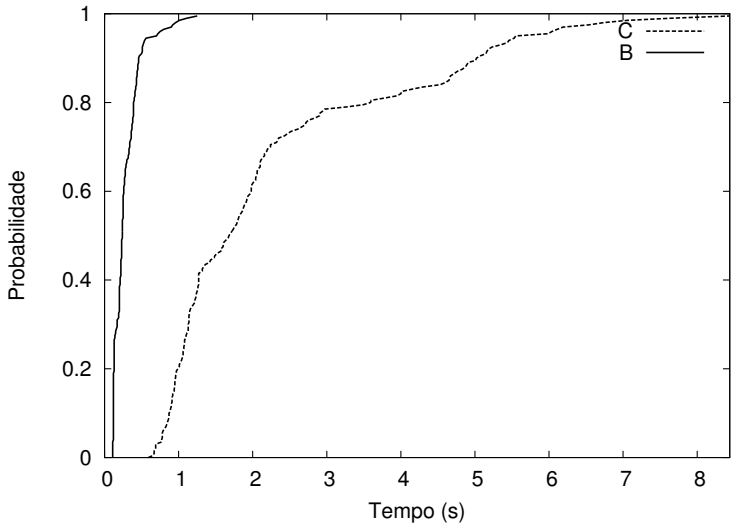

Figura 4.1. Probabilidade acumulada de encontrar solução alvo em determinado tempo para a instância $I1$- B x C.

4.3 Resultados GGA x AGs

No primeiro experimento, a qualidade da fase de construção dos diferentes AGs é comparada. Dessa forma, nesse teste o $BestValue$ é considerado a melhor solução inicial fornecida entre todos os métodos para cada uma das instâncias.

Tabela 4.4. Qualidade da fase de construção

	Instâncias AP			
Método	GAI	GAII	GAIII	GGA
$Score$	87	126	62	0
$\#Best$	9	11	10	56
$DevMin$	0,0895	0,2108	0,0797	0,0000
$DevMed$	0,1646	0,3806	0,1493	0,0029

A Tabela 4.4 apresenta a qualidade da fase de construção dos AGs. Note que o GGA obteve melhor resultado que os demais métodos em todas as métricas computadas, e sempre fornece o $BestValue$ da fase de construção para todas instâncias. Além disso, o método construção do GGA obtém $DevMed$ menor que $DevMin$ dos outros métodos, comprovando a robustez e qualidade da estratégia de construção proposta. A Figura 4.2 reporta o erro percentual médio da melhor solução inicial de cada método para o $BestValue$. Como pode ser visto, a fase de construção do GGA claramente supera GAI, GAII e GAIII fornecendo um erro médio pequeno em relação ao $BestValue$.

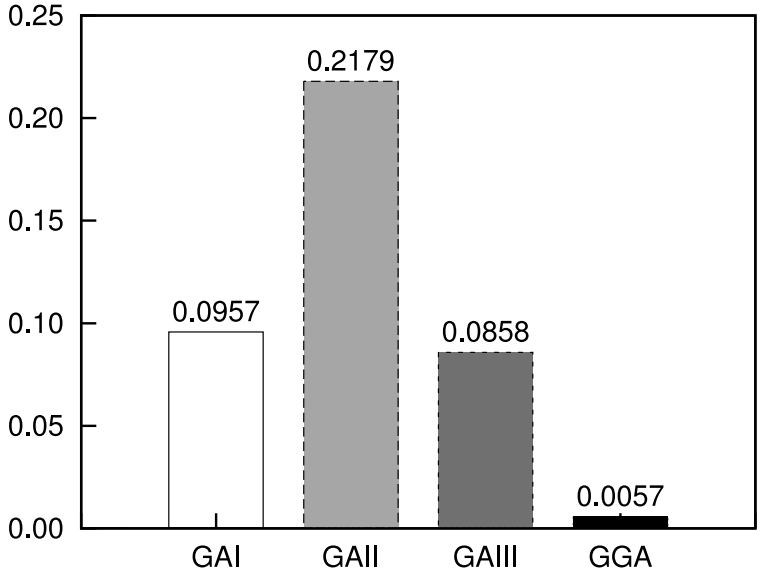

Figura 4.2. Erro percentual médio da melhor solução para o $BestValue$.

No próximo conjunto de testes, os métodos foram comparados em relação a melhor solução encontrada durante o processo evolucionário.

Tabela 4.5. Eficiência dos métodos - GGA x AGs

	Instâncias AP			
Method	GAI	GAII	GAIII	GGA
$Score$	76	135	32	6
$\#Best$	17	11	24	50
$DevMin$	0,0162	0,2136	0,0011	0,0001
$DevMed$	0,0514	0,3774	0,0023	0,0013
Ganho Médio	0,0737	0,0039	0,0846	0,0056

A Tabela 4.5 reporta os resultados obtidos, e, como pode ser visto, o GGA obteve soluções superiores entre os métodos comparados, apresentando melhores valores para todas as métricas consideradas. O AG implementado é capaz de encontrar o $BestValue$ em 50 das 56 instâncias, tendo também $DevMin$ e $DevMed$ muito menores que os outros AGs. O menor valor de ganho médio do GGA é justificado pela qualidade da fase de construção do mesmo, a qual fornece soluções com erro pequeno em relação ao $BestValue$. Na Figura 4.3 é apresentado o número de vezes em que o $BestValue$ é encontrado pelo método considerando as 30 execuções das 56 instâncias teste. Nota-se que, o GGA é mais robusto que os demais métodos, sendo capaz de alcançar o $BestValue$ em mais vezes.

Finalmente, a Figura 4.4 ilustra a distribuição de probabilidade acumulada para os algoritmos GGA e GAIII de encontrar uma solução alvo, média das médias das soluções encontradas por ambos algoritmos, para a instância $|N| = 100$ com $\alpha = 0,2$, depois de 200 execuções de cada método. Percebe-se que o tempo de processamento que o GGA leva para encontrar o

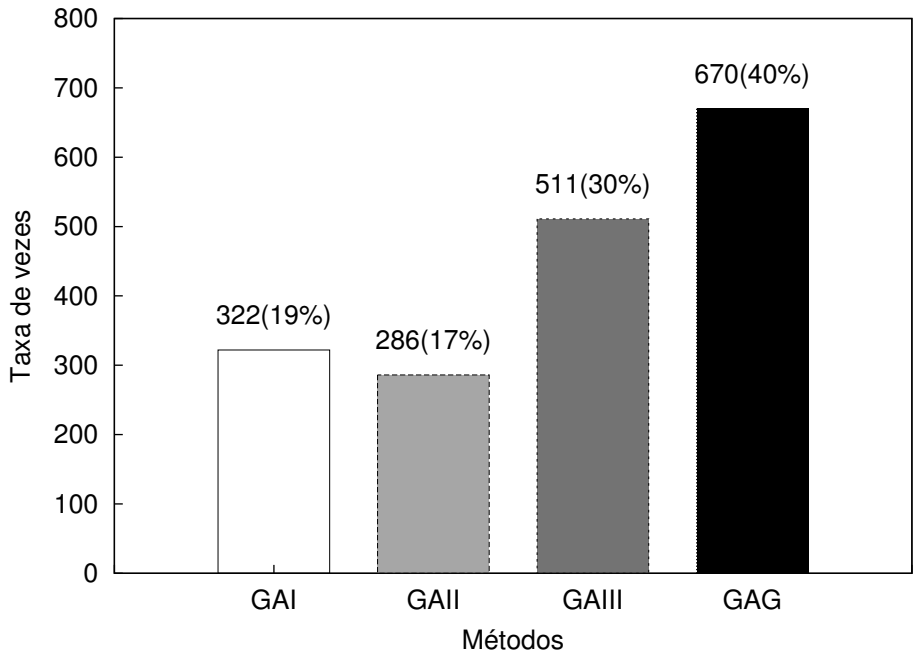

Figura 4.3. Número de vezes que o $BestValue$ é encontrado durante os experimentos - GGA x AGs.

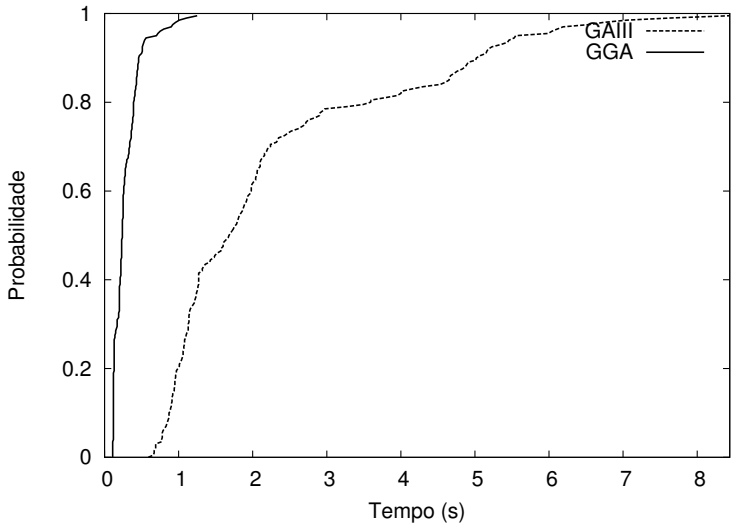

Figura 4.4. Probabilidade acumulada de encontrar solução alvo em determinado tempo para a instância $Ap100 - 2$ - GGA x GAIII.

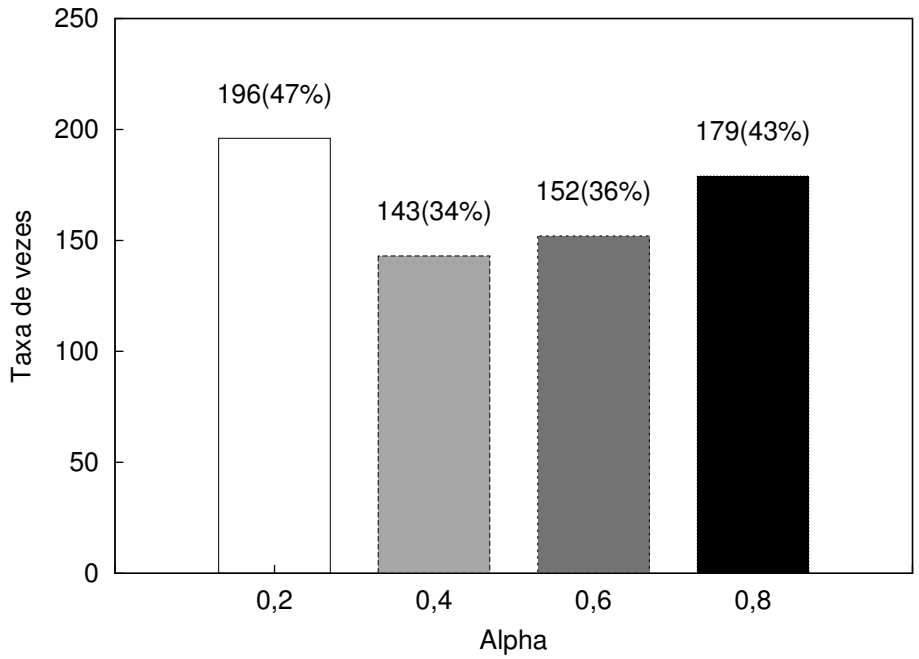

Figura 4.5. Número de vezes que o GGA encontra o $BestValue$ para os diferentes valores de α.

alvo é muito menor que o GAIII, mostrando que o algoritmo proposto, além de obter soluções de melhor qualidade, também é mais eficiente em relação ao tempo de processamento para encontrar uma determinada solução alvo. Nesse exemplo, enquanto o GAIII tem probabilidade quase nula de encontrar a solução alvo em aproximadamente $0,5$ segundos, as chances do GGA alcançar tal valor são maiores que 90%, provando a eficiência do algoritmo proposto.

Na Figura 4.5 é reportado o número de vezes que o GGA encontrou o $BestValue$ para os diferentes fatores de desconto (α). Note que, o GGA encontrou maior dificuldade para $\alpha = 0,4$ e $\alpha = 0,6$, pois o número de vezes que obtém-se o $BestValue$ é menor nesses casos.

Operadores GGA x GAIII

Esta subseção apresenta uma comparação dos operadores do GGA com os operadores do AG que obteve melhor resultado entre os AGs da literatura estudados. Para isso, a fase de construção proposta nesse trabalho foi inserida no GAIII. Dessa forma, os algoritmos trabalharam em condições iguais em relação à população inicial, pois, como pôde ser visto anteriormente, a estratégia de construção implementada fornece melhores indivíduos para a população inicial. É importante lembrar que a metodologia adotada nesses experimentos é a mesma da seção ante-

rior.

Tabela 4.6. Operadores GGA x GAIII

Instâncias AP		
Métricas	GGA	GAIII
$DevMin$	0,00010	0,00121
$DevMed$	0,00125	0,00251
$\#Best$	53	23
$Score$	3	33

Os resultados reportados na Tabela 4.6 mostram que os operadores do GGA são mais eficientes que os do GAIII, uma vez que os algoritmos possuem a mesma fase de construção e, mesmo assim, o GGA obteve melhores valores em todas as métricas computadas. Além disso, a Figura 4.6 apresenta o número de vezes que os algoritmos convergiram para o $BestValue$ de cada instância, sendo que mais uma vez o método proposto superou o GAIII, pois encontra o $BestValue$ em mais execuções.

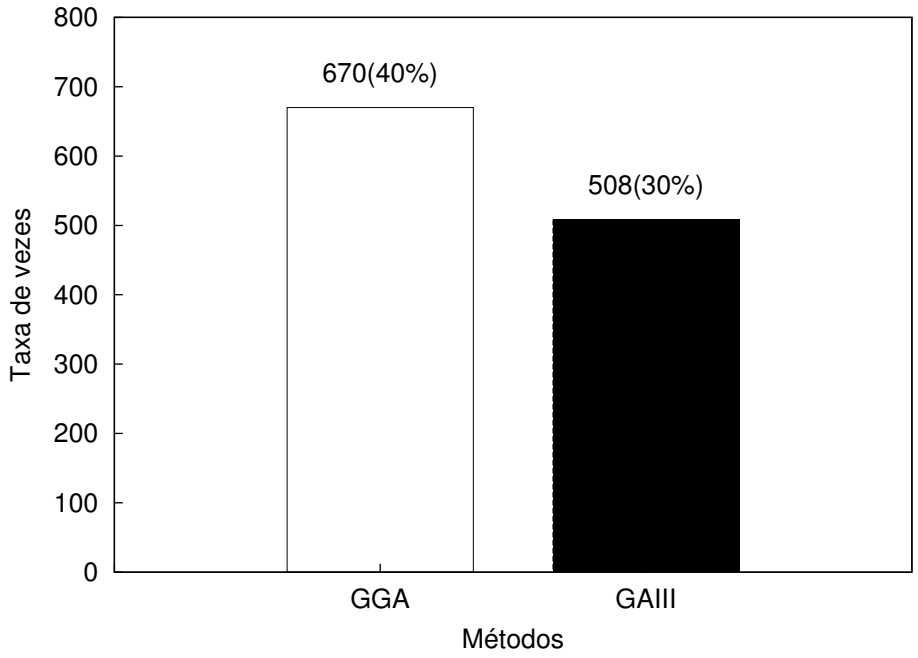

Figura 4.6. Número de vezes que o $BestValue$ é encontrado durante os experimentos - GGA x GAIII.

Os resultados apresentados mostram que o algoritmo proposto claramente supera outros 3 AGs bem conhecidos da literatura, tanto em relação a qualidade de soluções obtidas quanto

em tempo de processamento para alcançar uma solução alvo.

4.4 Resultados GGA com busca local x GGA

Nesta seção serão reportados os resultados obtidos pelo GGA combinado com as buscas locais propostas nesse trabalho: GGA com busca local deslocamento de alocação (GGA-S); GGA com busca local troca de função (GGA-T); GGA com busca local remover concentrador (GGA-R); e GGA com busca local inserir concentrador (GGA-I).

Tabela 4.7. GGA x GGA combinado com buscas locais

	Instâncias AP				
Métricas	GGA	GGA-S	GGA-T	GGA-R	GGA-I
$DevMin$	0,00179	0,00004	0,00139	0,00150	0,00139
$DevMed$	0,00294	0,00075	0,00229	0,00223	0,00219
$\#Best$	26	53	30	29	27
$Score$	87	3	45	68	61

Os resultados reportados na Tabela 4.7 mostram a eficiência do GGA combinado com as buscas locais propostas. Nessa comparação, o GGA-S obteve melhores resultados em todas as métricas computadas, fornecendo o $BestValue$ em 53 das 56 instâncias teste. Além disso, o $DevMed$ do GGA-S é muito menor que o $DevMin$ dos demais métodos, provando a robustez do AG proposto quando combinado com a busca local de re-alocação dos nós não concentradores aos concentradores instalados. Nota-se ainda que, as demais buscas não oferecem uma melhora significativa em relação ao GGA, já que os resultados do mesmo são, de certa forma, competitivos com o GGA-T, GGA-R e GGA-I. Na Figura 4.7 o número de vezes que o $BestValue$ foi encontrado pelos métodos é apresentado, considerando as 30 execuções das 56 instâncias teste. Os resultados comprovam ainda mais a eficiência e robustez do GGA combinado com a busca local deslocamento de alocação, pois o mesmo alcança o $BestValue$ em aproximadamente 70% das execuções, quase o dobro de vezes que os demais métodos.

Como o GGA-S obteve os melhores resultados, na Figura 4.8 é ilustrada a distribuição de probabilidade acumulada dos algoritmos GGA e GGA-S encontrarem uma determinada solução alvo, média das médias das soluções encontradas por ambos algoritmos, para a instância $|N| = 100$ com $\alpha = 0,2$, depois de 200 execuções de cada método. É importante ressaltar que as soluções alvo das Figuras 4.8 e 4.4 são diferentes. Percebe-se que, o GGA-S é capaz de alcançar a solução alvo em menor tempo de processamento que o GGA sem a busca local. Observa-se também que, no melhor tempo de processamento do GGA para obtenção do alvo (aproximadamente 55 segundos) onde a probabilidade é quase nula, o GGA-S possui aproximadamente 90% de probabilidade de encontrar tal solução. Essa fato mostra que a combinação

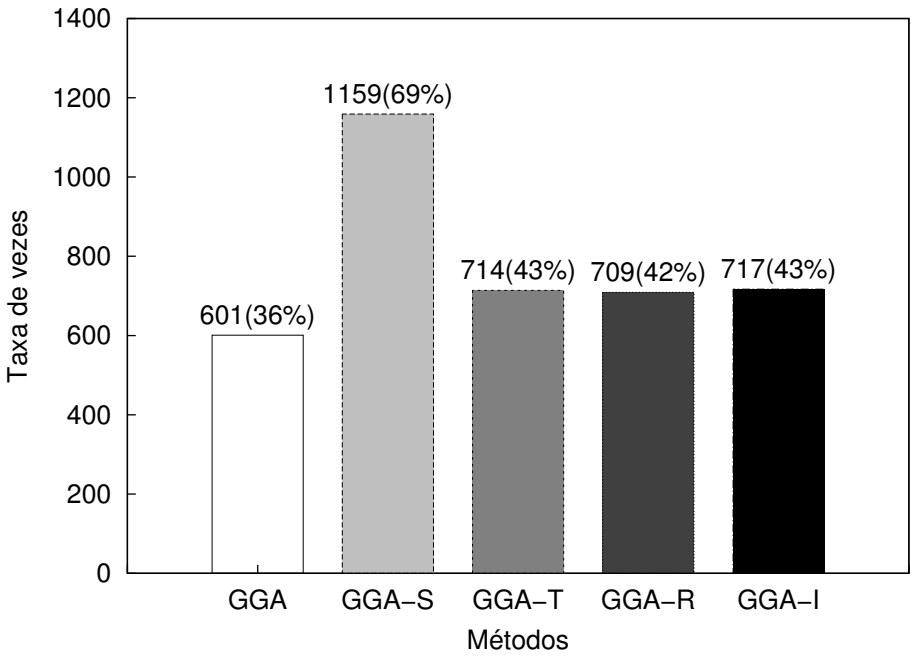

Figura 4.7. Número de vezes que o $BestValue$ é encontrado durante os experimentos - GGA x GGA com busca local.

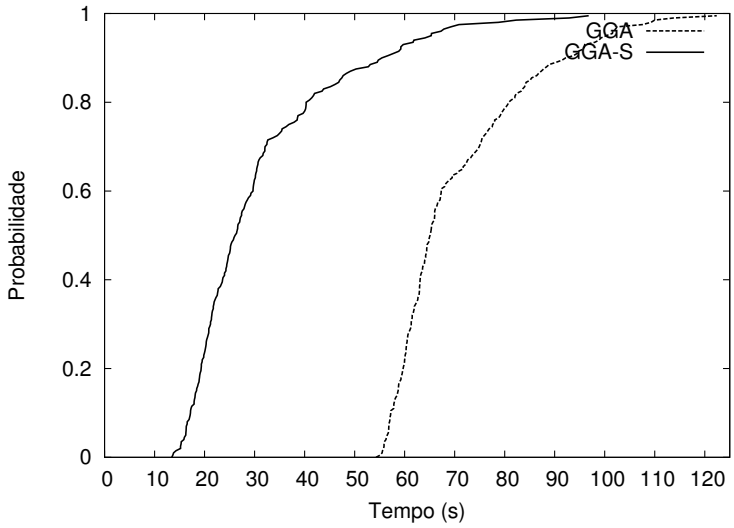

Figura 4.8. Probabilidade acumulada de encontrar solução alvo em determinado tempo para a instância $Ap100 - 2$ - GGA x GGA-S.

do AG proposto com a busca local deslocamento de alocação, além de obter soluções de melhor qualidade, também é mais eficiente em relação ao tempo de processamento para encontrar uma determinada solução alvo.

O próximo conjunto de experimentos visa obter o desempenho do GGA-S em relação à eficiência para alcançar a solução ótima de cada instância. Portanto, o $BestValue$ nesses experimentos computacionais é o valor ótimo de cada instância, obtido pelo algoritmo de decomposição de Benders proposto por Castro [59], como indicado no Apêndice A. Nesse conjunto de testes, também são reportados os valores NumÓtimo e T(s) que representam, respectivamente, o número de vezes que o método convergiu para a solução ótima de cada instância e a média do tempo de processamento (segundos) requisitado pelo método, considerando as 30 execuções do algoritmo para cada problema teste.

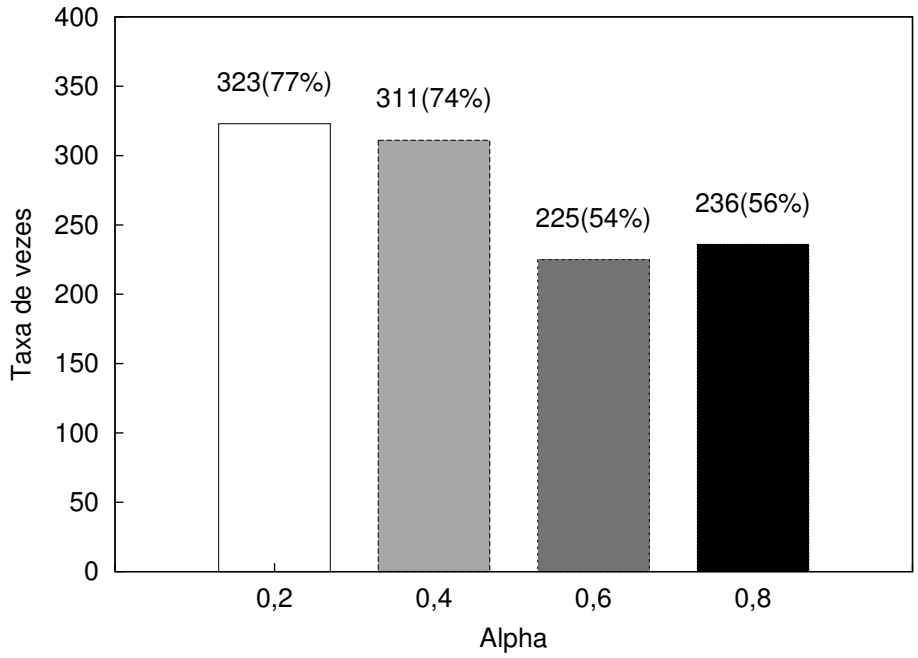

Figura 4.9. Número de vezes que o GGA-S encontra a solução ótima para os diferentes valores de α.

A Tabela 4.8 mostra a eficiência do GGA-S em relação à solução ótima das instâncias. Nota-se que, o algoritmo proposto também é eficiente na obtenção do ótimo dos problemas teste, não encontrando tal solução em 11 das 56 instâncias (AP30-2, AP70-8, AP90-8, AP100-6, AP100-8, AP130-6, AP150-6, AP150-8, AP170-6 e AP200-8), mas mesmo quando não encontra o ótimo, o método possui baixos valores de $DevMin$ e $DevMed$. O GGA-S tem pior

Tabela 4.8. Eficiência do GGA-S para obtenção da solução ótima considerando 30 execuções de cada instância

Instâncias AP				
Instância	$DevMin$	$DevMed$	NumÓtimo	T(s)
AP10-2	0,0000	0,0000	30	0,1657
AP10-4	0,0000	0,0000	30	0,0290
AP10-6	0,0000	0,0000	30	0,0083
AP10-8	0,0000	0,0000	30	0,0100
AP20-2	0,0000	0,0000	30	0,1133
AP20-4	0,0000	0,0000	30	0,0986
AP20-6	0,0000	0,0000	30	0,6919
AP20-8	0,0000	0,0000	30	0,0670
AP30-2	0,0011	0,0011	0	30,0270
AP30-4	0,0000	0,0000	30	1,0659
AP30-6	0,0000	0,0000	30	0,2268
AP30-8	0,0000	0,0000	30	0,2370
AP40-2	0,0000	0,0000	30	0,9263
AP40-4	0,0000	0,0000	30	0,6907
AP40-6	0,0000	0,0000	30	2,6197
AP40-8	0,0000	0,0000	30	0,4987
AP50-2	0,0000	0,0000	30	2,0100
AP50-4	0,0000	0,0000	30	4,3107
AP50-6	0,0000	0,0007	14	36,8580
AP50-8	0,0000	0,0000	30	6,3837
AP60-2	0,0000	0,0014	11	42,9537
AP60-4	0,0000	0,0000	30	2,6247
AP60-6	0,0000	0,0020	12	48,9653
AP60-8	0,0000	0,0000	30	6,7437
AP70-2	0,0000	0,0000	30	3,7970
AP70-4	0,0000	0,0000	30	3,7603
AP70-6	0,0000	0,0001	27	24,9817
AP70-8	0,0007	0,0023	0	70,0620
AP80-2	0,0000	0,0000	30	7,2797
AP80-4	0,0000	0,0000	30	6,8023
AP80-6	0,0000	0,0006	28	19,1627
AP80-8	0,0000	0,0005	26	33,1477
AP90-2	0,0000	0,0000	30	18,5167
AP90-4	0,0000	0,0000	29	16,7150
AP90-6	0,0000	0,0034	14	65,3703
AP90-8	0,0011	0,0029	0	90,0837
AP100-2	0,0000	0,0000	26	46,1830
AP100-4	0,0000	0,0000	30	13,9587
AP100-6	0,0001	0,0046	0	100,1160
AP100-8	0,0023	0,0027	0	100,1303
AP130-2	0,0000	0,0000	30	44,3567
AP130-4	0,0000	0,0010	3	122,8000
AP130-6	0,0002	0,0047	0	130,4013
AP130-8	0,0000	0,0006	3	122,6930
AP150-2	0,0000	0,0006	7	145,1557
AP150-4	0,0000	0,0007	3	143,3550
AP150-6	0,0022	0,0039	0	150,5773
AP150-8	0,0010	0,0064	0	150,5053
AP170-2	0,0000	0,0002	17	137,5857
AP170-4	0,0000	0,0003	7	151,3457
AP170-6	0,0021	0,0065	0	170,4273
AP170-8	0,0000	0,0000	27	85,9747
AP200-2	0,0000	0,0002	29	135,6357
AP200-4	0,0000	0,0003	1	200,2260
AP200-6	0,0000	0,0020	8	185,9157
AP200-8	0,0030	0,0038	0	200,6683

resultado na resolução da instância AP200-8, na qual possui $DevMin$ e $DevMed$, respectivamente, igual à 0,0030 e 0,0038. Além disso, a Figura 4.9 mostra que quando os fatores de desconto (α) são iguais à $0,6$ ou $0,8$ o método encontrou maior dificuldade para convergir para as soluções ótimas, pois a soma dos valores ótimos obtidos nesses casos são menores.

4.5 Resultados GGA com VND x GGA-S

Nesta seção serão reportados os resultados obtidos pela combinação do GGA com a descida em vizinhança variável (GGA-VND), especificamente a sequência de vizinhanças V^1 (deslocamento de alocação), V^4 (inserir concentrador), V^2 (troca de função) e V^3 (remover concentrador) que obteve melhores resultados que as demais combinações, conforme pode ser visto no Anexo C.

Tabela 4.9. GGA-S X GGA-VND

Instâncias AP		
Métricas	GGA-S	GGA-VND
$DevMin$	0,00025	0,00003
$DevMed$	0,00100	0,00022
$\#Best$	48	56
$Score$	8	0

Os resultados reportados na Tabela 4.9 mostram a eficiência do GGA-VND em comparação com o GGA-S. Nessa comparação, o GGA-VND obteve melhores resultados em todas as métricas computadas, fornecendo o $BestValue$ para todas instâncias testes. Além disso, o $DevMed$ do GGA-VND é menor que o $DevMin$ do GGA-S, provando a eficiência do método proposto. Adicionalmente, a Figura 4.10 reporta o número de vezes que o $BestValue$ foi encontrado pelos métodos, considerando as 30 execuções das 56 instâncias teste. Os resultados comprovam ainda mais a eficiência e robustez do GGA-VND, pois o mesmo alcança o $BestValue$ em mais de 85% das execuções (1433 das 1680 execuções) do algoritmo.

A Figura 4.11 ilustra a distribuição de probabilidade acumulada dos algoritmos GGA-S e GGA-VND encontrarem uma solução alvo, média das médias das soluções encontradas por ambos algoritmos, para a instância $|N| = 100$ com $\alpha = 0,2$, depois de 200 execuções de cada método. É importante ressaltar que as soluções alvo das Figuras 4.11 e 4.8 são diferentes. Percebe-se que, o GGA-VND é capaz de alcançar a solução alvo em menor tempo de processamento que o GGA-S. Observa-se também que, em aproximadamente 35 segundos o GGA-VND possui 99,9% de encontrar o alvo, por outro lado, o GGA-S possui aproximadamente 70% de probabilidade de alcançar tal solução. Esse fato mostra que o GGA quando combinado com

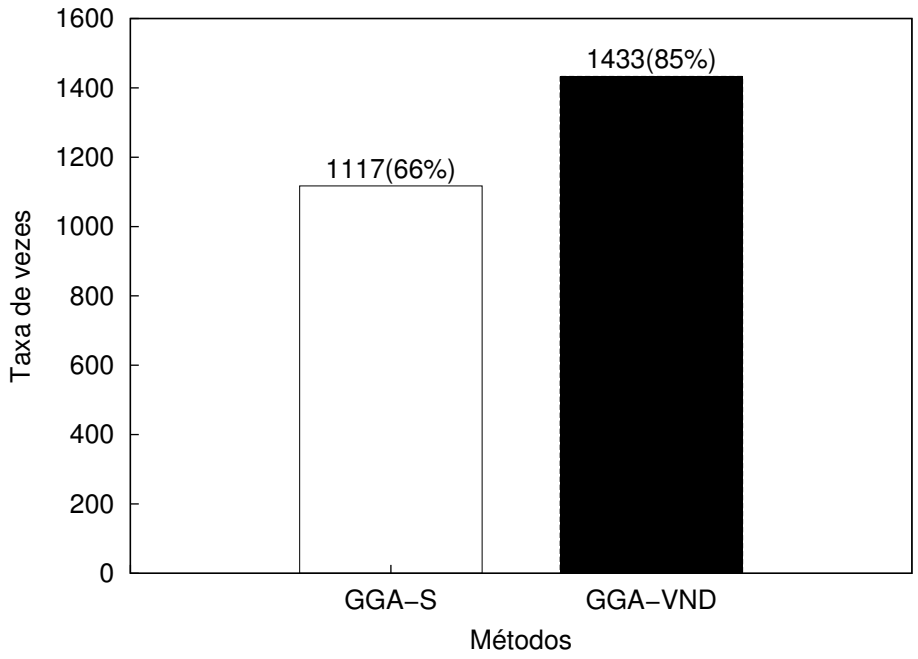

Figura 4.10. Número de vezes que o $BestValue$ é encontrado durante os experimentos- GGA-S x GGA-VND.

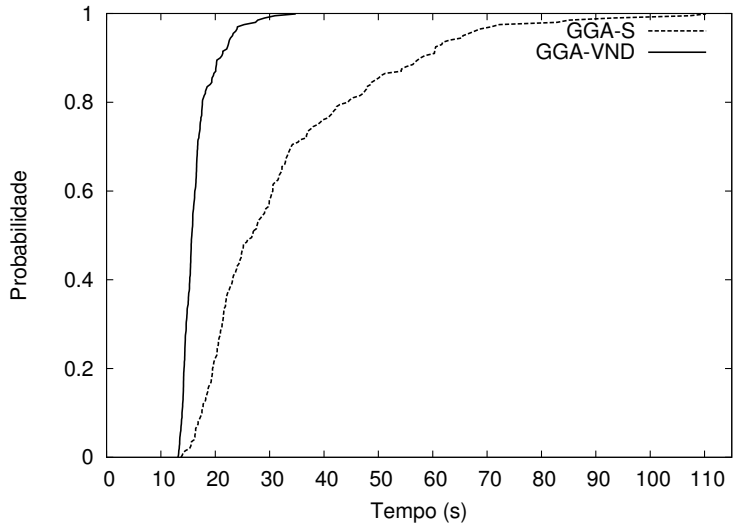

Figura 4.11. Probabilidade acumulada de encontrar solução alvo em determinado tempo para a instância $Ap100 - 2$ - GGA-S x GGA-VND.

o VND, além de obter soluções de melhor qualidade, também é mais eficiente em relação ao tempo de processamento para encontrar uma determinada solução alvo.

De forma similar à seção anterior, o próximo conjunto de teste mostra a eficácia do GGA-VND para obtenção da solução ótima das instâncias. As mesmas métricas são utilizadas como comparação.

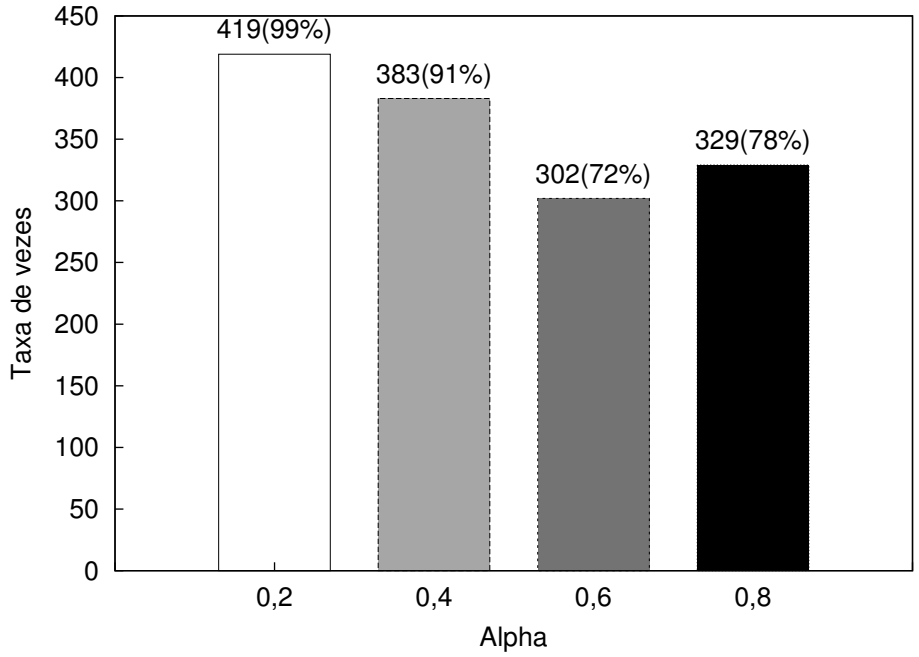

Figura 4.12. Número de vezes que o GGA-VND encontra a solução ótima para os diferentes valores de α.

A Tabela 4.10 mostra a eficiência do GGA-VND em relação à obtenção da solução ótima das instâncias. Nota-se que, o algoritmo proposto é muito eficiente para alcançar a solução ótima dos problemas teste, não encontrando tal solução somente em 3 das 56 instâncias (AP130-6, AP150-8 e AP170-6), melhorando os resultados do GGA-S que não obtinha o ótimo em 11 das 56 instâncias. Observa-se ainda que, mesmo quando o GGA-VND não encontra a solução ótima, os valores alcançados possuem baixo $DevMin$ e $DevMed$ para a solução ótima. O GGA-VND tem pior resultado na resolução da instância AP170-6, em que o método possui $DevMin$ e $DevMed$, respectivamente, igual à 0,0010 e 0,0024. Adicionalmente, a Figura 4.12 mostra que quando os fatores de desconto (α) são iguais à $0,6$ ou $0,8$ o algoritmo perde desempenho na obtenção das soluções ótimas, sendo que a soma de valores ótimos encontrados pelo método nesses casos são menores.

Tabela 4.10. Eficiência do GGA-VND para obtenção da solução ótima considerando 30 execuções de cada instância

Instâncias AP				
Instância	$DevMin$	$DevMed$	NumÓtimo	T (s)
AP10-2	0,0000	0,0000	30	0,1373
AP10-4	0,0000	0,0000	30	0,0363
AP10-6	0,0000	0,0000	30	0,0110
AP10-8	0,0000	0,0000	30	0,0113
AP20-2	0,0000	0,0000	30	0,0267
AP20-4	0,0000	0,0000	30	0,0133
AP20-6	0,0000	0,0000	30	0,0183
AP20-8	0,0000	0,0000	30	0,0177
AP30-2	0,0000	0,0001	28	8,8213
AP30-4	0,0000	0,0000	30	0,6837
AP30-6	0,0000	0,0000	30	0,2360
AP30-8	0,0000	0,0000	30	0,2450
AP40-2	0,0000	0,0000	30	0,5807
AP40-4	0,0000	0,0000	30	0,6843
AP40-6	0,0000	0,0000	30	0,9077
AP40-8	0,0000	0,0000	30	0,5047
AP50-2	0,0000	0,0000	30	1,8397
AP50-4	0,0000	0,0000	30	2,0980
AP50-6	0,0000	0,0001	30	9,3843
AP50-8	0,0000	0,0000	30	1,3193
AP60-2	0,0000	0,0000	30	3,8007
AP60-4	0,0000	0,0000	30	2,5977
AP60-6	0,0000	0,0000	30	10,9233
AP60-8	0,0000	0,0000	30	3,0120
AP70-2	0,0000	0,0000	30	0,8903
AP70-4	0,0000	0,0000	30	3,2593
AP70-6	0,0000	0,0000	30	10,6777
AP70-8	0,0000	0,0006	5	67,0347
AP80-2	0,0000	0,0000	30	6,5143
AP80-4	0,0000	0,0000	30	5,5770
AP80-6	0,0000	0,0000	30	7,2803
AP80-8	0,0000	0,0001	30	20,4490
AP90-2	0,0000	0,0000	30	13,8733
AP90-4	0,0000	0,0000	30	11,7617
AP90-6	0,0000	0,0006	28	50,3377
AP90-8	0,0000	0,0004	28	52,5990
AP100-2	0,0000	0,0000	30	18,9697
AP100-4	0,0000	0,0000	30	11,7853
AP100-6	0,0000	0,0005	13	85,8107
AP100-8	0,0000	0,0011	23	79,0427
AP130-2	0,0000	0,0000	30	37,1060
AP130-4	0,0000	0,0007	9	121,6803
AP130-6	0,0002	0,0008	0	130,4093
AP130-8	0,0000	0,0003	13	112,1477
AP150-2	0,0000	0,0000	30	75,5677
AP150-4	0,0000	0,0000	22	106,8353
AP150-6	0,0000	0,0018	3	150,0863
AP150-8	0,0005	0,0012	0	150,4957
AP170-2	0,0000	0,0000	30	85,7237
AP170-4	0,0000	0,0000	22	121,7677
AP170-6	0,0004	0,0029	0	170,5353
AP170-8	0,0000	0,0000	30	41,7417
AP200-2	0,0000	0,0000	29	129,3860
AP200-4	0,0000	0,0000	30	113,9017
AP200-6	0,0000	0,0005	9	185,8183
AP200-8	0,0000	0,0003	15	174,7670

Os resultados mostram que a combinação do AG proposto com a descida em vizinhança variável é extremamente eficiente, fornecendo soluções de qualidade com menor tempo de processamento.

4.6 Conclusão

Neste capítulo foi apresentado um conjunto de experimentos computacionais que teve como objetivo investigar o desempenho do GGA e suas extensões, bem como do GAI, GAII e GAIII, na resolução do projeto de redes eixo-raio com alocação simples.

De uma forma geral, a avaliação de desempenho foi realizada por meio de um estudo comparativo entre os métodos considerando as seguintes métricas: $BestValue$, $DevMed$, $DevMin$, $\#Best$ e $Score$. Para isso, utilizou-se problemas teste da base de dados AP que contém instâncias de até 200 nós.

Os testes foram realizados comparando os métodos da literatura, GAI, GAII e GAIII, GGA e suas extensões: GGA simples, GGA com VND e GGA com cada uma das buscas locais inclusas no VND exploradas separadamente.

Primeiramente, mediu-se o desempenho do GGA simples em comparação aos AGs da literatura estudados. Em relação ao GAI, o algoritmo proposto obteve desempenho superior em 33 das 56 instâncias testadas, bem como forneceu valores de $Score$, $DevMin$ e $DevMed$ significativamente menores. Quando comparado com o GAII, o GGA obteve resultados superiores em 39 das 56 instâncias, além de ter alcançado valores melhores para as métricas $Score$, $DevMin$ e $DevMed$. Dentre os métodos da literatura, o GAIII apresentou resultados mais competitivos com o GGA, mas também foi superado em todas as métricas computadas. O GGA foi superior em 26 das 56 instâncias, assim como obteve menores valores de $Score$, $DevMin$ e $DevMed$. Além disso, o GGA também foi mais eficiente que o GAIII para encontrar uma determinada solução alvo.

Após mostrar que o GGA foi mais eficiente que os métodos da literatura em todas as métricas computadas, no segundo conjunto de testes o GGA simples foi comparado ao GGA combinado com as diferentes buscas locais implementadas. Constatou-se que, o GGA com busca local superou os resultados do GGA simples, principalmente quando combinado com a busca local deslocamento de alocação. Essa versão proporcionou os ganhos mais significativos, conseguindo bom desempenho, inclusive, para alcançar a solução ótima de algumas instâncias. Além disso, o GGA com busca local deslocamento de alocação foi mais eficiente que o GGA simples para obter uma determinada solução alvo. Melhores resultados foram alcançados pela exploração das diferentes buscas locais propostas com a utilização do VND. Essa combinação superou os resultados do GGA com busca local de re-alocação, assim como foi mais eficaz para

encontrar a solução ótima de algumas instâncias. Além disso, o GGA-VND também foi mais eficiente que o GGA com busca local deslocamento de alocação para obter uma determinada solução alvo.

Portanto, pode ser concluído que a estratégia de construção e os operadores do GGA levaram o mesmo a superar os resultados dos AGs da literatura estudados. O refinamento de soluções promissoras da população, por meio da aplicação de busca local, proporcionou ganhos significativos nas soluções, principalmente a busca local deslocamento de alocação. Além disso, a estratégia de exploração de diferentes estruturas de vizinhança de uma solução, utilizada pelo GGA com VND, forneceu os melhores resultados entre os métodos considerados. Assim sendo, as três abordagens propostas, GGA, GGA com buscas locais e GGA com VND, são competitivas para projeto de redes eixo-raio.

Capítulo 5

Conclusões e Trabalhos Futuros

Este capítulo concluiu a presente dissertação de mestrado que teve como foco o projeto de redes E-R com alocação simples. Apresenta-se inicialmente as questões fundamentais contempladas em cada um dos capítulos, bem como as principais contribuições e resultados. Por fim, são enumeradas algumas sugestões de trabalho futuros.

No Capítulo 1, apresentou-se o projeto de redes E-R, considerado um problema muito importante na área de otimização discreta, possuindo diversas aplicações em diferentes contextos como sistemas de telecomunicação e informação, e redes de transporte de carga e passageiros. Uma breve descrição destacou as principais características e aplicações desse problema.

O Capítulo 2 delineou importantes conceitos sobre o projeto de redes E-R e forneceu uma contextualização histórica do mesmo. Além disso, descreveu-se as diferentes variantes do problema, assim como os métodos encontrados na literatura para resolução do mesmo. Esse capítulo também contemplou uma revisão bibliográfica dos AEs, a qual contém o detalhamento dos principais métodos evolutivos da literatura para resolução do problema em questão.

Uma explanação das abordagens propostas para resolução do projeto de redes E-R com alocação simples foi apresentada no Capítulo 3. Assim sendo, as principais contribuições dessa dissertação estão presentes no mesmo. De uma forma geral, foi realizada uma discussão detalhada a cerca do funcionamento dos métodos implementados, assim como foi apresentado o pseudo-código de cada um deles. Primeiramente, descreveu-se o AG simples, o qual contém uma estratégia eficiente para geração da população inicial, e novos operadores de cruzamento e mutação capazes de evoluir a solução durante o processo evolucionário. Como os indivíduos promissores da população não passavam por um procedimento rigoroso de refinamento de soluções, foram implementadas quatro buscas locais para o projeto de redes E-R, assim, o capítulo também contemplou a descrição dessas buscas, bem como do VND implementado. O VND, técnica que explora o espaço de soluções através de trocas sistemáticas de estruturas de vizinhança, foi proposto pelo fato de cada uma das buscas locais desenvolvidas explorarem as

soluções em somente uma vizinhança específica.

No Capítulo 4, foi apresentado um conjunto de testes computacionais que teve como objetivo investigar o desempenho dos métodos estudados na resolução do projeto de redes E-R com alocação simples. De uma forma geral, a avaliação de desempenho foi realizada por meio de um estudo comparativo entre os métodos considerando as seguintes métricas: $BestValue$, $DevMed$, $DevMin$, $\#Best$ e $Score$. Para isso, utilizou-se problemas teste da base de dados AP que contém instâncias de até 200 nós.

Os testes foram realizados comparando os métodos da literatura, GAI, GAII e GAIII, GGA e suas extensões: GGA simples, GGA com VND e GGA com cada uma das buscas locais inclusas no VND exploradas separadamente.

Primeiramente, mediu-se o desempenho do GGA simples em comparação aos AGs da literatura estudados. Em relação ao GAI, o algoritmo proposto obteve desempenho superior em 33 das 56 instâncias testadas, bem como forneceu valores de $Score$, $DevMin$ e $DevMed$ significativamente menores. Quando comparado com o GAII, o GGA obteve resultados superiores em 39 das 56 instâncias, além de ter alcançado valores melhores para as métricas $Score$, $DevMin$ e $DevMed$. Dentre os métodos da literatura, o GAIII apresentou resultados mais competitivos com o GGA, mas também foi superado em todas as métricas computadas. O GGA foi superior em 26 das 56 instâncias, assim como obteve menores valores de $Score$, $DevMin$ e $DevMed$. Além disso, o GGA também foi mais eficiente que o GAIII para encontrar uma determinada solução alvo.

Após mostrar que o GGA foi mais eficiente que os métodos da literatura em todas as métricas computadas, no segundo conjunto de testes o GGA simples foi comparado ao GGA combinado com as diferentes buscas locais implementadas. Constatou-se que, o GGA com busca local superou os resultados do GGA simples, principalmente quando combinado com a busca local deslocamento de alocação. Essa versão proporcionou os ganhos mais significativos, conseguindo bom desempenho, inclusive, para alcançar a solução ótima de algumas instâncias. Além disso, o GGA com busca local deslocamento de alocação foi mais eficiente que o GGA simples para obter uma determinada solução alvo. Melhores resultados foram alcançados pela exploração das diferentes buscas locais propostas com a utilização do VND. Essa combinação superou os resultados do GGA com busca local de re-alocação, assim como foi mais eficaz para encontrar a solução ótima de algumas instâncias. Além disso, o GGA-VND também foi mais eficiente que o GGA com busca local deslocamento de alocação para obter uma determinada solução alvo.

Portanto, pode ser concluído que a estratégia de construção e os operadores do GGA levaram o mesmo a superar os resultados dos AGs da literatura estudados. O refinamento de soluções promissoras da população, por meio da aplicação de busca local, proporcionou ganhos significativos nas soluções, principalmente a busca local deslocamento de alocação. Além

disso, a estratégia de exploração de diferentes estruturas de vizinhança de uma solução, utilizada pelo GGA com VND, forneceu os melhores resultados entre os métodos considerados. Assim sendo, as três abordagens propostas, GGA, GGA com buscas locais e GGA com VND, são competitivas para projeto de redes eixo-raio.

Embora bons resultados tenham sido obtidos pelas abordagens propostas, ainda existe possibilidade de melhoria. Portanto, possíveis pontos a serem explorados em trabalhos futuros são descritos a seguir:

- Combinar o critério de Metropolis do SA com estratégia de exploração das vizinhanças do VND. Assim, a primeira vizinhança de uma solução poderá ser explorada, em casos de piora na exploração das demais vizinhanças, dada uma probabilidade. Essa probabilidade é calculada com base no critério de Metropolis;
- Implementar hierarquia na população do AG, de forma que a operação de cruzamento aconteça somente entre indivíduos de uma mesma hierarquia;
- Expandir o algoritmo implementado para as diferentes variantes do projeto de redes E-R.

Apêndice A

Resultados do método de decomposição de Benders

Este apêndice tem como objetivo apresentar os resultados obtidos pelo método de decomposição de Benders proposto por Castro [59]. Os resultados reportados na Tabela A.1 serviram como forma de testar a eficiência dos métodos propostos nessa dissertação em relação a obtenção da solução ótima.

Tabela A.1. Resultados da decomposição de Benders.

Instâncias AP		
Instância	Ótimo	T (s)
Ap10-2	90963539,4763	0,0300
Ap10-4	95079629,9069	0,0400
Ap10-6	95161467,5800	0,0200
Ap10-8	95161467,5800	0,0100
Ap20-2	91507336,6084	1,5500
Ap20-4	96673177,8591	0,6300
Ap20-6	98181949,7142	0,4200
Ap20-8	98181949,7142	0,3200
Ap30-2	83576754,7126	2,3400
Ap30-4	91209293,1037	4,5700
Ap30-6	95053569,4196	3,7100
Ap30-8	98520500,5833	2,3200
Ap40-2	80537210,1476	5,0300
Ap40-4	88042046,0930	9,4000
Ap40-6	94523086,4474	15,2400
Ap40-8	99180263,1029	21,3000
Ap50-2	71261044,6135	15,1400
Ap50-4	80325464,9026	13,5300
Ap50-6	89389885,1918	51,9200
Ap50-8	95205946,9629	52,7700
Ap60-2	64790967,0386	23,0100
Ap60-4	73074656,5752	27,7200
Ap60-6	80673823,6599	36,2100
Ap60-8	87285162,0911	38,1400
Ap70-2	74451085,6010	62,6500
Ap70-4	83272519,6325	67,0400
Ap70-6	91741977,1993	153,1500
Ap70-8	97109841,9802	96,1500
Ap80-2	70713485,9938	156,1800
Ap80-4	79704787,9527	150,2100
Ap80-6	88418089,7530	160,5600
Ap80-8	95798238,8757	271,8900
Ap90-2	69223173,9238	285,1400
Ap90-4	78931357,8060	303,4900
Ap90-6	87012179,9499	314,0500
Ap90-8	92780846,2542	340,2800
Ap100-2	67584119,7648	530,2000
Ap100-4	77545112,3194	583,4000
Ap100-6	86371515,8647	664,1800
Ap100-8	93184508,9692	1010,8900
Ap130-2	60901562,4908	1813,1400
Ap130-4	71560028,4291	2084,0000
Ap130-6	81332451,6280	3498,2000
Ap130-8	89960490,4486	4977,0300
Ap150-2	57397001,4468	3989,3200
Ap150-4	68083743,1244	5545,3500
Ap150-6	77856884,8290	5069,0800
Ap150-8	86068900,6009	5970,1900
Ap170-2	63888877,9054	7213,1900
Ap170-4	73373712,1241	7272,9100
Ap170-6	81744592,0692	8291,2500
Ap170-8	88839517,4393	10622,3000
Ap200-2	62774425,0872	19130,9800
Ap200-4	72333785,1444	25224,2000
Ap200-6	81236269,7266	26664,6300
Ap200-8	89181926,7904	52759,3700

Anexo A

Calibração dos parâmetros do algoritmo proposto

Este anexo tem como objetivo mostrar os resultados do estudo empírico para calibração dos parâmetros de cruzamento e mutação do AG implementado. Para isso foram escolhidas 24 das 56 instâncias teste, AP10, AP30, AP60, AP80, AP130 e AP170 com os diferentes valores de desconto $\alpha = \{0,2; 0,4; 0,6; 0,8\}$. Além disso, as mesmas métricas utilizadas como comparação foram consideradas.

Os parâmetros calibrados são: taxa de cruzamento p_{cruz}, probabilidade de cruzamento multi-pais p_{multi}, probabilidade de cruzamento de grupos p_{grupos}, probabilidade de cruzamento *GRASP* p_{grasp}, taxa de mutação p_{mut}, probabilidade de mutação inserir p_{ins}, probabilidade de mutação remover p_r, probabilidade de mutação troca de função p_{troca}, probabilidade de mutação troca de alocação p_e e probabilidade de mutação deslocamento de alocação p_s. Como diversas combinações dos diferentes parâmetros são possíveis, foram realizados 3 conjuntos de experimentos: o primeiro visa configurar os valores de p_{cruz} e p_{mut}; o segundo experimento as probabilidades de ocorrer diferentes tipos de cruzamento (p_{multi}, p_{grupos} e p_{grasp}) foram calibradas; e o terceiro conjunto de testes ajusta as probabilidade de aplicação dos diferentes operadores de mutação propostos (p_{ins}, p_r, p_{troca}, p_e e p_s).

No primeiro experimento 4 combinações de valores foram testadas:

- C9M1, versão com $p_{cruz} = 0,9$ e $p_{mut} = 0,1$;
- C9M4, versão com $p_{cruz} = 0,9$ e $p_{mut} = 0,4$;
- C8M2, versão com $p_{cruz} = 0,8$ e $p_{mut} = 0,2$;
- C8M3, versão com $p_{cruz} = 0,8$ e $p_{mut} = 0,3$.

Tabela A.1. Calibração dos parâmetros de cruzamento e mutação

	Instâncias AP			
Métricas	C9M1	C9M4	C8M2	C8M3
$DevMin$	0,00055	0,00020	0,00041	0,00036
$DevMed$	0,00163	0,00118	0,00159	0,00148
$\#Best$	14	17	15	15
$Score$	22	12	17	15

É importante lembrar que os demais parâmetros foram fixados.

A Tabela A.1 mostra que a configuração de $p_{cruz} = 0,9$ e $p_{mut} = 0,4$ obteve resultados superiores, pois o método retornou melhores valores em todas as métricas. Portanto, esse valor foi considerado como base para os demais testes e para o AG proposto.

No segundo teste, 4 combinações para os parâmetros foram consideradas:

- CM5CN4CG1, versão com $p_{multi} = 0,5$, $p_{grupos} = 0,4$ e $p_{grasp} = 0,1$;
- CM2CN4CG2, versão com $p_{multi} = 0,4$, $p_{grupos} = 0,4$ e $p_{grasp} = 0,2$;
- CM4CN5CG1, versão com $p_{multi} = 0,4$, $p_{grupos} = 0,5$ e $p_{grasp} = 0,1$;
- CM3CN3CG4, versão com $p_{multi} = 0,3$, $p_{grupos} = 0,3$ e $p_{grasp} = 0,4$.

É importante lembrar que os demais parâmetros foram fixados.

Tabela A.2. Calibração das probabilidades de cruzamento

	Instâncias AP			
Métricas	CM5CN4CG1	CM4CN4CG2	CM4CN5CG1	CM3CN3CG4
$DevMin$	0,00028	0,00049	0,00034	0,00085
$DevMed$	0,00128	0,00162	0,00144	0,00187
$\#Best$	17	15	16	11
$Score$	11	15	15	30

A Tabela A.2 mostra que quando os parâmetros foram configurados com $p_{multi} = 0,5$, $p_{grupos} = 0,4$ e $p_{grasp} = 0,1$, o algoritmo obteve resultados superiores, pois o método alcançou melhores valores em todas as métricas. Portanto, essa configuração dos parâmetros é considerada como base para o último experimento e para o AG proposto.

No último experimento, 4 combinações para os parâmetros foram consideradas:

- I1R1T3E25S25, versão com $p_i = 0,1$, $p_r = 0,1$, $p_{troca} = 0,3$, $p_e = 0,25$ e $p_s = 0,25$;
- I2R2T2E2S2, versão com $p_i = 0,2$, $p_r = 0,2$, $p_{troca} = 0,2$, $p_e = 0,2$ e $p_s = 0,2$;

Tabela A.3. Calibração das probabilidades de mutação

	Instâncias AP			
Métricas	I1R1T3E25S25	I2R2T2E2S2	I25R25T2E15S15	I2R2T1E25S25
$DevMin$	0,00019	0,00032	0,00044	0,00038
$DevMed$	0,00110	0,00133	0,00161	0,00152
$\#Best$	17	14	12	15
$Score$	12	16	17	16

- I25R25T2E15S15, versão com $p_i = 0,25$, $p_r = 0,25$, $p_{troca} = 0,2$, $p_e = 0,15$ e $p_s = 0,15$;
- I2R2T1E25S25, versão com $p_i = 0,2$, $p_r = 0,2$, $p_{troca} = 0,1$, $p_e = 0,25$ e $p_s = 0,25$.

A Tabela A.3 mostra que quando os parâmetros foram configurados com $p_i = 0,1$, $p_r = 0,1$, $p_{troca} = 0,3$, $p_e = 0,25$ e $p_s = 0,25$, o algoritmo obteve resultados superiores, pois o método alcançou melhores valores em todas as métricas. Portanto, essa configuração dos parâmetros é considerada como base para o AG proposto.

Assim, após esse conjunto de experimentos empíricos os parâmetros de cruzamento e mutação foram determinados.

Anexo B

Calibração de parâmetros dos métodos da literatura

De forma semelhante ao GGA, os AGs da literatura, GAI, GAII e GAIII, foram submetidos a um procedimento para calibração de parâmetros. Este anexo tem como objetivo mostrar os resultados dessa calibração, assim como reportar o desempenho dos métodos calibrados na resolução de instâncias benchmark da literatura.

Para calibração dos AGs, escolheu-se 24 das 56 instâncias teste da base de dados AP, AP10, AP30, AP60, AP80, AP130 e AP170, com diferentes valores de desconto $\alpha = \{0,2; 0,4; 0,6; 0,8\}$. Além disso, as mesmas métricas utilizadas como comparação foram consideradas.

Os parâmetros calibrados são: probabilidade de cruzamento, p_{cruz}, e probabilidade de mutação, p_{mut}. Como diversas combinações para os valores desses parâmetros são possíveis, 4 delas foram testadas para os diferentes métodos.

B.1 Calibração de parâmetros do GAI

Nesta seção são evidenciados os resultados da calibração de parâmetros do AG proposto por Topcuoglu et al. [53]. As versões resultantes das diferentes combinações de probabilidades são:

- C8M2, versão com $p_{cruz} = 0,8$ e $p_{mut} = 0,2$;
- C8M4, versão com $p_{cruz} = 0,8$ e $p_{mut} = 0,4$;
- C7M4, versão com $p_{cruz} = 0,7$ e $p_{mut} = 0,4$;
- C7M3, versão com $p_{cruz} = 0,7$ e $p_{mut} = 0,3$.

Tabela B.1. Calibração GAI

	Instâncias AP			
Métricas	C8M2	C8M4	C7M3	C7M4
DevMin	0,00131	0,00873	0,00356	0,00711
DevMed	0,03369	0,03820	0,03631	0,04128
#Best	17	7	13	9
Score	14	35	18	28

A Tabela B.1 mostra que a versão C8M2 do GAI, quando os parâmetros foram configurados com $p_{cruz} = 0,8$ e $p_{mut} = 0,2$, apresentou resultados superiores, pois o método alcançou melhores valores em todas as métricas computadas. Portanto, essa configuração dos parâmetros é considerada na seção B.4.

B.2 Calibração de parâmetros do GAII

Nesta seção são evidenciados os resultados da calibração de parâmetros do AG proposto por Cunha e Silva [8]. As versões resultantes das diferentes combinações de probabilidades são:

- C8M2, versão com $p_{cruz} = 0,8$ e $p_{mut} = 0,2$;
- C8M3, versão com $p_{cruz} = 0,8$ e $p_{mut} = 0,3$;
- C9M1, versão com $p_{cruz} = 0,9$ e $p_{mut} = 0,1$;
- C99M01, versão com $p_{cruz} = 0,99$ e $p_{mut} = 0,01$.

Tabela B.2. Calibração GAII

	Instâncias AP			
Métricas	C8M2	C8M3	C9M1	C99M01
DevMin	0,01171	0,01162	0,01467	0,00300
DevMed	0,07249	0,07236	0,07176	0,06228
#Best	17	17	13	18
Score	10	9	18	12

A Tabela B.2 mostra que a versão C99M01 do GAII, quando os parâmetros foram configurados com $p_{cruz} = 0,99$ e $p_{mut} = 0,01$, apresentou resultados superiores, pois o método alcançou melhores valores em todas as métricas computadas. Portanto, essa configuração dos parâmetros é considerada na seção B.4.

B.3 Calibração de parâmetros do GAIII

Nesta seção são evidenciados os resultados da calibração de parâmetros do AG proposto por Naeem e Ombuki-Berman [56]. É importante lembrar que, no GAIII os dois filhos gerados na operação de cruzamento possuem diferentes probabilidades de mutação. As versões resultantes das diferentes combinações de probabilidade são:

- C6M2-4, versão com $p_{cruz} = 0,6$, $p_{mut} = 0,2$ para o primeiro filho e $p_{mut} = 0,4$ para o segundo filho;
- C7M3-3, versão com $p_{cruz} = 0,7$, $p_{mut} = 0,3$ para o primeiro filho e $p_{mut} = 0,3$ para o segundo filho;
- C8M1-4, versão com $p_{cruz} = 0,8$, $p_{mut} = 0,1$ para o primeiro filho e $p_{mut} = 0,4$ para o segundo filho;
- C6M4-2, versão com $p_{cruz} = 0,6$, $p_{mut} = 0,4$ para o primeiro filho e $p_{mut} = 0,2$ para o segundo filho.

Tabela B.3. Calibração GAIII

	Instâncias AP			
Métricas	C6M2-4	C7M3-3	C8M1-4	C6M4-2
DevMin	0,00027	0,00045	0,00126	0,00016
DevMed	0,00130	0,00198	0,00276	0,00125
#Best	18	15	9	18
Score	8	15	42	8

A Tabela B.3 mostra que a versão C6M4-2 do GAIII, quando os parâmetros foram configurados com $p_{cruz} = 0,6$, $p_{mut} = 0,4$ para o primeiro filho e $p_{mut} = 0,2$ para o segundo filho, apresentou resultados superiores, pois o método alcançou melhores valores em todas as métricas computadas. Portanto, essa configuração dos parâmetros é considerada na seção B.4.

B.4 GGA x AGs calibrados

Esta seção tem como objetivo comparar o algoritmo proposto com os AGs da literatura após o procedimento de calibração de parâmetros. Para avaliar o desempenho dos métodos utilizou-se, além das instâncias AP já descritas, a base de dados introduzida por O'Kelly [14]. Essa base de dados é baseada no fluxo de passageiros de linhas aéreas entre 25 cidades dos Estados Unidos em 1970, conforme dados da *Civil Aeronautics Board* (CAB). A base de dados

CAB possui instâncias de 10, 15, 20 e 25 nós, e fatores de desconto, $\alpha = \{0,2; 0,4; 0,6; 0,8\}$, utilizados como economia de escala na conexão entre concentradores. Portanto, gerou-se 16 problemas teste dessa base de dados.

Tabela B.4. GGA x AGs calibrados - Instâncias CAB

	Instâncias CAB			
Métricas	GAI	GAII	GAIII	GGA
DevMin	0,01650	0,01182	0,00025	0,00016
DevMed	0,02524	0,01925	0,00028	0,00016
#Best	11	12	14	15
Score	13	8	4	1

Os resultados reportados na Tabela B.4 mostram que o GGA é mais eficiente que os AGs da literatura para resolução das instâncias da base de dados CAB, uma vez que obteve melhores valores em todas as métricas computadas. Além disso, o GGA possui *DevMed* menor que o *DevMin* dos demais métodos.

Comparou-se também o desempenho dos métodos na resolução das instâncias AP.

Tabela B.5. GGA x AGs calibrados - Instâncias AP

	Instâncias AP			
Métricas	GAI	GAII	GAIII	GGA
DevMin	0,01494	0,21366	0,00115	0,00018
DevMed	0,04722	0,37741	0,00229	0,00133
#Best	20	11	23	49
Score	70	135	42	9

Os resultados reportados na Tabela B.5 mostram que o GGA também obteve desempenho superior aos demais métodos na resolução das instâncias AP, fornecendo melhores valores para todas as métricas computadas.

Anexo C

Resultados das diferentes combinações de vizinhança do VND

O VND implementado para o projeto de redes E-R possui 4 estruturas de vizinhança diferentes: busca local deslocamento de alocação (V^1), busca local troca de função (V^2), busca local remover concentrador (V^3) e busca local inserir concentrador (V^4). Com o objetivo de definir a ordem de estruturas de vizinhança a ser explorada durante a execução do método, foram testadas 6 ordens diferentes: VND1234; VND1243; VND1324; VND1342; VND1423; e VND1432. O VND1234 representa a versão que a sequência de vizinhanças consideradas foi V^1 deslocamento de alocação, V^2 troca de função, V^3 remover concentrador e V^4 inserir concentrador. A nomeação das outras versões seguem a mesma ideia.

Para determinar a melhor sequência de vizinhanças, foram escolhidas 28 das 56 instâncias teste, AP10, AP30, AP50, AP70, AP90, AP130 e AP170 com os diferentes valores de desconto $\alpha = \{0,2; 0,4; 0,6; 0,8\}$, sendo que 30 execuções de cada método são realizadas para cada instância. Além disso, as mesmas métricas utilizadas como comparação foram consideradas, porém o $BestValue$ é considerado o valor ótimo de cada instância e $NumBest$ representa o número de vezes que o algoritmo convergiu para o ótimo nas 30 execuções de cada instância.

A Tabela C.1 mostra que os resultados obtidos por 3 versões (VND1243, VND1342 e

Tabela C.1. Resultados das diferentes versões do VND

	Instâncias AP					
Métricas	VND1234	VND1243	VND1324	VND1342	VND1423	VND1432
$DevMin$	0,00008	0,00004	0,00007	0,00003	0,00005	0,00011
$DevMed$	0,00040	0,00027	0,00037	0,00030	0,00023	0,00031
$\#Best$	26	26	24	25	25	25
$NumBest$	632	658	641	663	677	656
$Score$	4	2	9	2	3	6

VND1423) são bem parecidos. Porém, pelo menor valor de DevMed e maior número de obtenção da solução ótima, a versão VND1423 foi considerada melhor, e foi utilizada no AG proposto.

Referências Bibliográficas

[1] R. S. Toh, R. C. Higgins, The impact of hub and spoke network centralization and route monopoly on domestic airline profitability., Transportation Journal 24 (1985) 16–27.

[2] T. Aykin, Network policies for hub-and-spoke systems with applications to the air transportation system., Transportation Science 29 (1995) 201–221.

[3] N. Bania, P. Bauer, T. Zlatoper, Us air passenger service: a taxonomy of route networks, hub locations, and competition., Logistics and Transportation Review 34 (1998) 53–74.

[4] M. J. Kuby, R. G. Gray, Hub network design problem with stopovers and feeders: case of federal express., Transportation Research 27 (1993) 1–12.

[5] J. G. Klincewicz, Hub location in backbone/tributary network design: a review., Location Science 6 (1998) 3–7–335.

[6] T. Don, S. Harit, J. R. English, W. G., Hub and spoke networks in truckload trucking: configuration testing and operational concerns., Logistics and Transportation 31 (1995) 209–237.

[7] K. Lumsden, F. Dallari, R. Ruggeri, Improving the efficiency of the hub and spoke system for the skf european distribution network., International Journal of Physical Distribution & Logistics Management 29 (1999) 50–64.

[8] C. B. Cunha, M. R. Silva, A genetic algorithm for the problem of configuring a hub-and-spoke network for a LTL trucking company in Brazil., European Journal of Operational Research 179 (2007) 747–758.

[9] Y. Lee, B. Lim, J. Park, A hub location problem in designing digital data service networks: Lagrangian relaxation approach., Location Science 4 (1996) 185–194.

[10] S. Abdinnour-Helm, Using simulated annealing to solve the p-hub median problem., International Journal of Physical Distribution & Logistic Management 31 (3) (2001) 203–220.

[11] S. Alumur, B. Y. Kara, Network hub location problems: The state of the art., European Journal of Operational Research 190 (2008) 01–21.

[12] J. F. Campbell, A. T. Ernst, M. Krishnamoorthy, Facility location: Applications and theory, Drezner, Z., Hammacher, H., 2002.

[13] M. E. O'Kelly, A geographer's analysis of hub-and-spoke networks., Journal of Transport Geography 3 (6) (1998) 171–186.

[14] M. E. O'Kelly, A quadratic integer program for the location of interacting hub facilities, European Journal of Operational Research 32 (1987) 393–404.

[15] M. Coffin, M. J. Saltzman, Statiscal analysis of computacional tests of algorithms and heuristics, Journal on Computing 12 (2000) 24–44.

[16] R. L. Rardin, R. Uzsoy, Experimental evaluation of heuristics optimization algorithms: A tutorial, Journal of Heuristics 7 (2001) 261–304.

[17] M. R. Garey, D. S. Jonhson, Computers and Intractability: A Guide to the Theory of NP-Completeness., W. H. Freeman, 1979.

[18] D. D. Tate, A. E. Smith, A genetic approach to the quadratic assignment problem, Computers and Operations Research 1 (1995) 855–865.

[19] B. M. Ombuki-Berman, A. Runka, F. T. Hanshar, Waste collection vehicle routing problem with time windows using multi-objective genetic algorithms, Computational Intelligence 574 (2007) 91–97.

[20] H. Beyer, The theory of evolution strategies, Natural computing series, Springer, Berlin, 2001.

[21] T. Aykin, The hub location and routing routing problem., European Journal of Operational Research 83 (1995) 200–219.

[22] M. E. O'Kelly, A clustering approach to the planar hub location., Annals of Operations Research 40 (1992) 339–353.

[23] M. E. O'Kelly, H. J. Miller, Mode choice in a hub-and-spoke network: A zero-one linear programming approach., Geographical Analysis 23(4) (1991) 283–297.

[24] J. F. Campbell, A survey of network hub location., Studies in Locational Analysis 6 (1994) 31–49.

[25] M. E. O'Kelly, H. J. Miller, The hub network design problem., Journal of Transport Geography 2(1) (1994) 31–40.

[26] M. E. O'Kelly, The location of interacting hub facilities, Transportation Science 20(2) (1986) 92–106.

[27] A. J. Goldman, Optimal location for centers in a network, Transportation Science 3 (1969) 352–360.

[28] S. L. Hakimi, Optimum locations of switching centers and the absolute centers and medians of a graph., Operations Research 12 (1964) 450–459.

[29] S. L. Hakimi, Optimum distribution of switching centers in a communication network and some related graph theoretic problems., Operations Research 13 (1965) 462–475.

[30] D. L. Bryan, M. E. O'Kelly, Hub-and-spoke network in air transportation: an analytical review., Journal of Regional Science 39(2) (1999) 275–295.

[31] J. F. Campbell, Integer programming formulations of discrete hub location problems., European Journal of Operational Research 72 (1994) 387–405.

[32] D. Skorin-Kapov, J. Skorin-Kapov, M. E. O'Kelly, Tight linear programming relaxations of uncapacitated p-hub median problems., European Journal of Operational Research 94 (1996) 582–593.

[33] A. T. Ernst, M. Krishnamoorthy, Efficient algorithms for the uncapacitated single allocation p-hub median problem., Location Science 4 (1996) 139–154.

[34] J. Ebery, Solving large single allocation p-hub problems with two or three hubs., European Journal of Operational Research 128 (2) (2001) 447–458.

[35] J. G. Klincewicz, Heuristics for the p-hub location problem., European Journal of Operational Research 53 (1991) 25–37.

[36] J. G. Klincewicz, Avoiding local optima in the p-hub location problem using tabu search and grasp., Annals of Operations Research 40 (1992) 283–302.

[37] D. Skorin-Kapov, J. Skorin-Kapov, On tabu search for the location of interacting hub facilities., European Journal of Operational Research 73 (1994) 501–508.

[38] H. Pirkul, D. A. Schilling, An efficient procedure for designing single allocation hub-and-spoke systems, Management Science 44 (12) (1998) 235–242.

[39] K. Smith, M. Krishnamoorthy, M. Palaniswami, Neural versus traditional approaches to the location of interacting hub facilities., Location Science 4 (3) (1996) 155–171.

[40] B. Y. Kara, B. C. Tansel, On the single-assignment p-hub center problem., European Journal of Operational Research 125 (2000) 648–655.

[41] A. T. Ernst, H. Hamacher, H. Jiang, M. Krishnamoorthy, G. Woeginger, Uncapacitated single and multiple allocation p-hub center problems, CSIRO Mathematical an Information Sciences, 2002.

[42] F. Pamuk, C. Sepil, A solution to the hub center problem via a single-relocation algorithm with tabu search., IIE 33 (5) (2001) 399–411.

[43] A. T. Ernst, H. Hamacher, H. Jiang, M. Krishnamoorthy, G. Woeginger, Heuristic algorithms for the uncapacitated hub center single allocation problem., CSIRO Mathematical an Information Sciences, 2002.

[44] A. M. Campbell, T. J. Lowe, L. Zhang, The p-hub center allocation problem, European Journal of Operational Research 176 (2) (2007) 819–835.

[45] B. Y. Kara, B. C. Tansel, The latest arrival hub location problem., Management Science 47 (2003) 1408–1420.

[46] B. Wagner, Model formulations for hub covering problems, Institute of Operational Research, 2004.

[47] A. T. Ernst, H. Hamacher, H. Jiang, M. Krishnamoorthy, Reformulations and computacional results for single and multiple allocation hub covering problems, CSIRO Mathematical an Information Sciences, 2005.

[48] J. F. Campbell, A. T. Ernst, M. Krishnamoorthy, Hub arc location problems: Part i - introduction on results., Management Science 51 (10) (2005) 1540–1555.

[49] H. W. Hamacher, T. Meyer, Hub cover and hub center problems., in: Departament of Mathematics, University of Kaiserslautern, Germany, 2006.

[50] M. E. O'Kelly, Hub facility location with fixed costs., Papers in Regional Science 71 (3) (1992) 293–306.

[51] S. Abdinnour-Helm, M. A. Venkataramanan, Solution approaches to hub location problems., Annals of Operations Research 78 (1998) 31–50.

[52] S. Abdinnour-Helm, A hybrid heuristic for the uncapacitated hub location problem., European Journal of Operational Research 2-3(106) (1998) 489–499.

[53] H. Topcuoglu, F. Corut, M. Ermis, G. Yilmaz, Solving the uncapacitated hub location problem using genetic algorithms., Computers and Operations Research 32 (4) (2005) 967–984.

[54] J. F. Chen, A hybrid heuristic for the uncapacitated single allocation hub location problem, Omega 35 (2007) 211–220.

[55] M. R. Silva, C. B. Cunha, New simple and efficient heuristics for the uncapacitated single allocation hub location problem., Computers and Operations Research 36 (12) (2009) 3152–3165.

[56] M. Naeem, B. Ombuki-Berman, An efficient genetic algorithm for the uncapacitated single allocation hub location problem, in: IEEE Congress on Evolutionary Computation, 2010.

[57] M. Labbé, H. Yaman, Projecting flow variables for hub location problems., Networks 44 (2) (2004) 84–93.

[58] R. Castro, R. S. Camargo, G. Miranda, Método de decomposição de benders aplicado a localizacao de concentradores em redes do tipo eixo-raio com alocacao simples., in: Simpósio Brasileiro de Pesquisa Operacional, 2009.

[59] R. R. M. Castro, Sistemas eixo-raio de alocação simples: Modelos e algoritmos, Master's thesis, Universidade Federal de Minas Gerais (2010).

[60] R. Camargo, G. Miranda, H. Luna, Benders decomposition for the uncapacitated multiple allocation hub location problem., Computers and Operations Research 35 (2008) 1047–1064.

[61] D. J. Futuyma, Biologia Evolutiva, FUNPEC, 2003.

[62] C. Silva, S. Sasson, Biologia, Saraiva, 1991.

[63] J. A. Ramirez, F. Campelo, F. G. Guimarães, R. H. C. Takahashi, Notas de aula de otimização, Tech. rep., Universidade Federal de Minas Gerais (2009).

[64] J. H. Holland, Outline for a logical theory of adaptive systems, J. ACM 9 (3) (1962) 297–314.

[65] D. E. Goldberg, Genetic Algorithms in Search, Optimization, and Machine Learning., Addison-Wesley Professional, 1989.

[66] C. Reeves, Hybrid genetic algorithms for binpacking and related problems, Annals of Operation Research 63 (1993) 371–396.

[67] T. Namada, R. Nakano, Genetic algorithms for job-shop scheduling problems, Modern Heuristic for Decision Support 1 (1997) 474–479.

[68] T. Fang-Chih, H. Kuang-Han, C. Chun-Yuan, L. Chi-Shuan, Using hybrid genetic algorithms to solve discrete location allocation problems with rectilinear distance, Journal of the Chinese Institute of Industrial Engineers 24 (2007) 1–19.

[69] N. Metropolis, A. Rosenbluth, M. Rosenbluth, A. Teller, E. Teller, Equation of state calculations by fast computing machines, J. Chem. Phys. 21 (1953) 1087–1092.

[70] M. Grotschel, O. Holland, Solution of large scale travelling salesman problem., Mathematical Programming 52 (1991) 141–202.

[71] R. J. Vanderbei, Linear Programming: Foundations and Extensions., Springer, 2001.

[72] G. Duan, Y. Yu, Problem-specific genetic algorithms for power transmission system planning., Electrical Power and System Research 61 (2002) 41–50.

[73] T. A. Feo, M. G. C. Resende, A probabilistic heuristic for a computationally difficult set covering problem., Operations Research Letters 8 (1989) 67–71.

[74] T. A. Feo, M. G. C. Resende, Greedy randomized adaptive search procedures, Journal of Global Optimization 6 (1995) 109–133.

[75] M. G. C. Resende, C. C. Ribeiro, Greedy randomized adaptive search porocedures, G. Kochenberger (Eds.), Handbook of metaheuristics, 2003.

[76] N. Mladenovic, P. Hansen, A variable neighborhood search, Computers and Operations Research 24 (1997) 1097–1100.

[77] C. C. Ribeiro, E. Uchoa, R. F. Werneck, A hybrid grasp with pertubations for the steiner problem in graphs., Journal on Computing 14 (2002) 228–246.