APLICAÇÕES DE HEURÍSTICAS À SOBREVIVÊNCIA DE REDES

Marcelo Lisboa Rocha
Fundação UNIRG
Alameda Madrid Nº 545, Jardim Sevilha, CEP 77410-470, Gurupi – TO – Brasil
marcelolisboarocha@yahoo.com.br

Fabiano Vieira de Alvarenga, Rosemir Barbosa Silva
Alameda Madrid Nº 545, Jardim Sevilha, CEP 77410-470, Gurupi – TO – Brasil
fabianovieiraa@yahoo.com.br, rosemirbs@hotmail.com

Resumo

Muitos problemas de sobrevivência de rede têm sido formulados como problemas de otimização combinatória. Neste trabalho, uma das principais questões quando do projeto de redes de telecomunicações é computar topologias de rede de custo mínimo que forneçam proteção contra falhas de equipamentos de rede e, conseqüentemente, aumente a sobrevivência da rede. Neste trabalho, são propostas três heurísticas para o problema: o método guloso, guloso aleatorizado e GRASP. Este trabalho propõe encontrar boas topologias de rede minimizando seus custos e satisfazendo as restrições de conectividade impostas.
Palavras-Chaves: Sobrevivência de rede; Método guloso; Método guloso aleatorizado; GRASP.

Abstract

Several problems of network survability have been seen as combinatorial optimization problems. In this work, the main question about telecommunication network design is to compute minimum cost network topologies that give protection against equipments failures and consequently, increase the survivability of the network. In this work, are proposed three heuristics to the problem: a greedy method, a randomized greedy method and GRASP. This work intends to find good network topologies minimizing their costs and satisfying imposed connectivity restrictions.
Keywords: Network survability; Greedy method; Randomized Greedy method; GRASP.

1. INTRODUÇÃO

A partir do início da década de 70, houve um grande crescimento da complexidade do sistema de telecomunicações brasileiro. Isso tem exigido das empresas maneiras mais econômicas e seguras para o fornecimento desses serviços e conseqüentemente, a aplicação adequada de ferramentas computacionais e técnicas de otimização no planejamento dessas redes [06].

Problemas de otimização combinatória aparecem freqüentemente em vários setores da economia. Como exemplos desses problemas podem-se citar o projeto de redes de telecomunicação, a construção de agendas para tripulações e a produção de rotas eficientes para coleta de lixo, entre outros [10].

Um dos principais fatores para o baixo desempenho destas tecnologias de rede é o processamento eletrônico de pacotes, que insere um atraso a cada nó da rede, e o uso ineficiente dos recursos da rede, que acarreta em uma indisponibilidade de recursos [01].

Como conseqüência, as redes tornaram-se muito vulneráveis e a gerência de todos os seus elementos constituintes tornou-se bastante complexa [06].

Segundo [07] a sobrevivência pode ser conceituada como a capacidade da rede sobreviver, total ou parcialmente, na ocorrência de falha. Sobreviver significa manter os serviços em certos níveis. Em geral, as falhas são provocadas por eventos não desejados, tais

como: raio, furacão, terremoto, tornado, incêndio inundação, erro não detectado no software, corte de cabo causado pela construção civil e etc. Elas podem ser classificadas em:

- Falha de nó – quando ocorre num centro de fios.
- Falha de arco – quando ocorre num cabo. Para efeito do nosso estudo, consideraremos somente falhas nos arcos.

Há muitas maneiras de avaliar a integridade da rede após a ocorrência de uma catástrofe; pode-se; por exemplo, calcular o volume de tráfego sobrevivente, o número de assinantes conectados ou a receita preservada [11].

Custos crescem com o aumento do valor desejado de sobrevivência, bem como com o volume de tráfego que desejamos proteger. Por isso, é importante proceder a uma pré-classificação seja dos centros de fios seja das demandas que circulam na rede rotulando-os como especiais ou comuns. Uma demanda será classificada como especial se possui importância destacada; por exemplo, o destaque pode ser conferido pela receita gerada [07].

As entradas do algoritmo são os nos e arcos formando uma rede e sua topologia. As saídas será a topologia da rede formada.

Heurística é uma palavra grega, que deriva do verbo *heurisken* que significa "encontrar". As heurísticas ou algoritmos heurísticos foram desenvolvidos com a finalidade de se resolver problemas de elevado nível de complexidade em tempo computacional razoável [03].

As heurísticas procuram encontrar soluções próximas da otimalidade em um tempo computacional razoável, sem, no entanto, conseguir definir se esta é a solução ótima, nem quão próxima ela está da solução ótima [10].

A heurística colaborara na sobrevivência da rede ampliando a sua sobrevivência de maneira sensível, garantindo assim serviços essenciais ou de alta rentabilidade para o setor [02].

O termo metaheurística deriva da composição de duas palavras gregas: heurística deriva do verbo *heurisken* que significa "encontrar", enquanto o sufixo "meta" significa "além de, em um nível superior" [03].

Um procedimento GRASP -- *Greedy Randomized Adaptive Search Procedure* (em português, Procedimento de Busca Adaptativo, Aleatório e Guloso) é uma metaheurística híbrida de construção seguida de melhoramento para Problemas de Otimização Combinatorial (POC). O princípio é análogo ao chamado método de múltiplas partidas aleatórias (*random multi-start*): repete-se várias vezes a aplicação de busca local em soluções iniciais diferentes geradas aleatoriamente [03].

A diferença crucial entre GRASP e Busca Local com Múltiplas Partidas Aleatórias (BLMP) é que no primeiro método a aleatoriedade é utilizada em conjunto com informação heurística que leva em conta critérios de otimização relativos à solução parcial; na segunda, a aleatoriedade é usada somente como critério direto de geração de combinações, arranjos ou seqüencias iniciais sem nenhuma preocupação com o valor de função objetivo [10].

Neste trabalho, só será abordado o problema de computação da topologia da rede de custo mínimo, onde uma rede é representada por uma coleção de nós (switches, roteadores, hubs, satélites, rádio-bases, etc.) e a conexão entre eles se dá por arestas/links (fibra ótica, fios elétricos, etc.), e a robustez da topologia da rede vem de sua confiabilidade [05].

A confiabilidade depende da confiabilidade do equipamento (links ou nós), mas também do fato de como os nós estão interligados. Por isto, a confiabilidade da rede pode ser caracterizada por diversos parâmetros distintos. Neste trabalho, a confiabilidade da rede foi baseada na presença de caminhos alternativos entre nós, ou seja, o parâmetro de conectividade [07].

Dado que a probabilidade de falha dos componentes de uma rede (links e nós) de telecomunicação é independente, o ideal é que a mesma seja imune a falhas dos mesmos. Este conceito de sobrevivência de redes permite a uma rede se manter funcional, mesmo quando

links são quebrados ou nós falham, ou seja, os serviços da rede podem ser recuperados mesmo na ocorrência de falhas catastróficas [04].

Desta forma, uma das principais questões quando do projeto de redes de telecomunicações é computar topologias de rede que forneçam proteção contra falhas de equipamentos de rede e, conseqüentemente, aumente a sobrevivência. Assim, o problema de computação de topologia de custo mínimo que será focado neste trabalho, consiste em selecionar links de modo que a soma de seus custos seja minimizada e a exigência do número de caminhos entre cada par de nós seja satisfeita [05].

2. O PROBLEMA DE SOBREVIVÊNCIA DE REDES

Segundo [07] a sobrevivência pode ser conceituada como a capacidade da rede sobreviver, total ou parcialmente, na ocorrência de falha. Sobreviver significa manter os serviços em certos níveis. Em geral, as falhas são provocadas por eventos não desejados, tais como: raio, furacão, terremoto, tornado, incêndio inundação, erro não detectado no software, corte de cabo causado pela construção civil e etc. Elas podem ser classificadas em:

- Falha de nó – quando ocorre num centro de fios.
- Falha de arco – quando ocorre num cabo. Para efeito do nosso estudo, consideraremos somente falhas nos arcos.

Há muitas maneiras de avaliar a integridade da rede após a ocorrência de uma catástrofe; pode-se; por exemplo, calcular o volume de tráfego sobrevivente, o número de assinantes conectados ou a receita preservada [11].

A confiabilidade das redes de telecomunicações tem despertado interesse maior dentro do setor devido à importância crescente dos serviços prestados, ao nível de exigência da qualidade dos serviços prestados aos usuários e à severidade das falhas em sistemas de alta hierarquia [07].

Como conseqüência, as redes tornaram-se muito vulneráveis e a gerência de todos os seus elementos constituintes tornou-se bastante complexa [06].

Custos crescem com o aumento do valor desejado de sobrevivência, bem como com o volume de tráfego que desejamos proteger. Por isso, é importante proceder a uma pré-classificação seja dos centros de fios seja das demandas que circulam na rede rotulando-os como especiais ou comuns. Uma demanda será classificada como especial se possui importância destacada; por exemplo, o destaque pode ser conferido pela receita gerada [07].

Demandas especiais deverão ter conectividade maior ou igual a dois, ou seja, garantiremos a existência de pelo menos dois caminhos arcos-disjuntos para roteá-las. Assim, tais demandas classificadas como comuns terão à disposição apenas um caminho para o seu roteamento [11]. As entradas do algoritmo são os nos e arcos formando uma rede e sua topologia. As saídas será a topologia da rede formada.

O conceito de redes robustas está entre os mais freqüentes nos problemas de projetos de redes de telecomunicações. Existem várias formas de expressar a robustez de uma rede. Uma delas pode ser definida como a habilidade de a rede continuamente executar suas funções na presença de danos e interrupções [08].

O processo de projeto de redes é extremamente complexo, pois ele gerencia o tráfego, o desempenho e os recursos da rede conjuntamente desta forma, não pode ser considerado um problema de otimização simples [06].

Neste trabalho, só será abordado o problema de computação da topologia da rede, onde uma rede é representada por uma coleção de nós (switches, roteadores, hubs, satélites, rádio-bases, etc.) e a conexão entre eles se dá por arestas/links (fibra ótica, fios elétricos, etc.), e a robustez da topologia da rede vem de sua confiabilidade [05].

A confiabilidade depende da confiabilidade do equipamento (links ou nós), mas também do fato de como os nós estão interligados. Por isto, a confiabilidade da rede pode ser caracterizada por diversos parâmetros distintos. Neste trabalho, a confiabilidade da rede foi

baseada na presença de caminhos alternativos entre nós, ou seja, o parâmetro de conectividade [07].

Desta forma, uma das principais questões quando do projeto de redes de telecomunicações é computar topologias de rede que forneçam proteção contra falhas de equipamentos de rede e, conseqüentemente, aumente a sobrevivência. Assim, o problema de computação de topologia que será focado neste trabalho, consiste em selecionar links de modo que a soma de seus custos seja minimizada e a exigência do número de caminhos entre cada par de nós seja satisfeita [05].

O número de caminhos entre cada par de nós é determinado da seguinte forma: a cada nó u, é associado um inteiro r(u), chamado de tipo da conexão que representa sua importância de comunicação de e para ele (quanto maior este número, mais importante é este nó); e o número de caminhos entre dois nós s e t é dado por r(s,t) = min{r(s), r(t)} [11].

3. MÉTODO GULOSO

O método guloso foi escolhido por ser uma das estratégias mais utilizadas para resolver Problemas de Otimização Combinatória. A estratégia deste método é construir iterativamente uma única combinação de solução possível, sem testar outras possibilidades, através de um critério de otimização definido por uma função gulosa. É importante notarmos que tal método pode gerar algoritmos exatos ou de forma mais abrangente algoritmos heurísticos, onde a otimalidade não é garantida [03].

O Método Guloso é uma forma de se resolver problemas que constrói a solução através de uma seqüência de passos, com um conjunto de opções (escolhas) a cada passo. Um Algoritmo Guloso sempre faz a escolha que parece ser melhor no momento, sem nunca reconsiderar esta decisão [09]. Esta escolha é feita com base nas informações locais, ou seja, pode-se dizer que o algoritmo faz uma escolha ótima para as condições locais, na esperança de que essa escolha leve a uma solução ótima para a situação global. Referimos à "esperança" de se chegar à solução ótima. Isso porque o Método Guloso não garante o retorno de uma solução ótima [05]. O pseudocódigo descrito pela Figura 1 ilustra um Procedimento Guloso.

```
1  função guloso(C: conjunto) : conjunto
2     {C é o conjunto de todos os candidatos}
3     S := {} {Conjunto que constitui a solução, inicialmente vazio}
4     Enquanto S não é uma solução e C não vazio
5        x := Melhor elemento de C
6        C := C - {x}
7        Se x é viável então S := S U {x}
8     Se S é uma solução
9        retornar S
10    Senão
11       retornar Não existe uma solução
```

Figura 1. Procedimento Guloso.

Neste código, inicialmente o conjunto solução de candidatos escolhidos é vazio. A seguir, a cada passo, o melhor candidato restante ainda não tentado é considerado, sendo o critério de escolha ditado pela função Seleciona. Este é então adicionado à solução e esta é então verificada se é viável até acabarem os itens [03].

4. GRASP

A metaheurística GRASP foi desenvolvida como uma tentativa de obter bons resultados para problemas difíceis de otimização combinatória. Tentativa esta que vem obtendo grande sucesso, porque atualmente esta metaheurística de construção e melhoria possui um grande destaque na literatura devido aos bons resultados obtidos nas aplicações já realizadas [10].

O GRASP é basicamente um procedimento iterativo, onde cada iteração consiste em uma fase de construção de uma solução inicial de forma gulosa, aleatória e adaptativa seguida

da aplicação de uma heurística de melhoramento, tipicamente um procedimento de busca local. A melhor dentre todas as iterações será então o resultado final [05].

O GRASP combina as características de heurísticas de construção e melhoramento (busca local). A estratégia deste método é acelerar o procedimento de busca local através da construção de soluções próximas, ou ainda, vizinhas a ótimos locais razoáveis e eventualmente ao ótimo global [09]. O pseudocódigo descrito pela Figura 2 ilustra um procedimento GRASP.

```
Procedimento GRASP
1       f(S) = +∞;
2       Para i de 1 até MaxIter faça
3       .   Aplicar o procedimento de construção para obter uma solução viável S ;
4       .   Aplicar busca local em S gerando uma nova solução S';
5       .   Se custo de f (S') <= f (S*) então
6       .       S* ← S' ;
7       .   Fim Se
8       Fim Para
9   Retornar S* ;
Fim Procedimento
```

Figura 2. Procedimento GRASP.

5. ALGORITMOS PROPOSTOS

Nesta Seção é apresentada uma descrição das principais ferramentas e representações dos algoritmos propostos neste trabalho para solução do problema de sobrevivência de redes. Os algoritmos foram implementados na linguagem C.

5.1. ALGORITMO GULOSO

O Método Guloso é uma forma de se resolver problemas que funcionam através de uma seqüência de passos, com um conjunto de opções (escolhas) a cada passo. Um Algoritmo Guloso sempre faz a escolha que parece ser melhor no momento, sem nunca reconsiderar esta decisão [09].

Esta escolha é feita com base nas informações locais, ou seja, pode-se dizer que o algoritmo faz uma escolha ótima para as condições locais, na esperança de que essa escolha leve a uma solução ótima para a situação global. Refere-se à "esperança" de se chegar à solução ótima porque o Método Guloso não garante o retorno de uma solução ótima [05].

Como proposta deste trabalho, tem-se uma heurística gulosa simples, aqui denominada HGS, como também uma heurística gulosa aleatoriazada, aqui denominada HGA. O objetivo de se utilizar aleatorização em um algoritmo guloso é permitir que o mesmo explore novas soluções no espaço de busca. Obtendo-se assim, uma maior diversidade de soluções e conseqüentemente, aumentando assim, a possibilidade de obtenção de melhores soluções que a heurística gulosa simples. Contudo, devido à característica de ser *multi-start* (múltiplos-reinícios), a HGA possui maior tempo computacional que a HGS. A seguir, nas Figuras 3 e 4, são apresentados respectivamente, os pseudo-códigos principais de HGS e HGA.

```
1. Inicializar S = Ø ; //conjunto de arestas da solução
2. A' = Ordenar as arestas a_ij do grafo G, em ordem crescente de custo c_ij
3. v = 0; //onde v é o valor da solução
4. Para cada aresta de A' Fazer
5.          Verificar Se o número de ligações mínimas de ambos os vértices i e j da aresta a_ij ainda não foi
            atingido então
6.              S = S ∪ a_ij;
7.              v = v + c_ij;
8.      FimSe
9. FimPara
```

Figura 3. Pseudo-código de HGS.

```
1. A' = Ordenar as arestas a_ij do grafo G, em ordem crescente de custo c_ij
2. Inicializar Sbest = Ø ; //conjunto de arestas da melhor solução
3. vbest = INFINITO;
4. Para k=1 até numiter Fazer
5.    Inicializar S = Ø ; //conjunto de arestas da solução
6.    v = 0; //onde v é o valor da solução
7.    Para cada aresta de A' Fazer
8.          Verificar Se o número de ligações mínimas de ambos os vértices i e j da aresta a_ij ainda não foi
            atingido então
9.             S = S ∪ a_ij;
10.            v = v + c_ij;
11.         FimSe
12.   FimPara
13.   Se v < vbest então
14.      vbest = v;
15.      Sbest = S;
16.   FimSe
17. FimPara
```

Figura 4. Pseudo-código de HGA.

5.2. ALGORITMO GRASP

A metaheurística GRASP (Greedy Rondomized Adaptative Search Procedure) que significa em português, "Procedimento de Busca Gulosa Adaptativa Aleatória", foi desenvolvida visando obter bons resultados para problemas difíceis de otimização combinatória.

Tentativa esta que vem obtendo grande sucesso, porque atualmente esta metaheurística de construção e melhoria possui um grande destaque na literatura devido aos bons resultados obtidos nas aplicações já realizadas [10].

O GRASP é basicamente um procedimento iterativo, onde cada iteração consiste em uma fase de construção de uma solução inicial de forma gulosa, aleatória e adaptativa seguida da aplicação de uma heurística de melhoramento, tipicamente um procedimento de busca local. A melhor dentre todas as iterações será então o resultado final [05].

O GRASP combina as características de heurísticas de construção e melhoramento (busca local). A estratégia deste método é acelerar o procedimento de busca local através da construção de soluções próximas, ou ainda, vizinhas a ótimos locais razoáveis e eventualmente ao ótimo global [09].

5.2.1. FASE DE CONSTRUÇÃO

A fase de construção é a responsável pela denominação do método, já que é nela que aparecem as características marcantes do GRASP: o método guloso (*greedy*), a aleatoriedade (*randomized*) e a adaptação da função gulosa (*adaptative*). Um pseudo-código da fase de construção adotada neste trabalho é apresentado na Figura 5. Esta fase é iterativa, já que a cada passo é adicionada à solução um novo elemento até que se tenha uma solução completa e válida. Cada iteração é composta por três subfases:

- Construção da Lista Restrita de Candidatos (LRC), a qual contém um conjunto reduzido de elementos candidatos a pertencer a solução, definidos de acordo com uma função gulosa e mecanismos de restrição;
- Escolha aleatória do elemento na LRC e inclusão de elemento na solução;
- Adaptação ou recálculo da função gulosa para os elementos ainda não pertencentes à solução [03].

```
Procedimento Construção
1  Inicializar S = Ø ; //conjunto de arestas da melhor solução
2  A' = Ordenar as arestas a_ij do grafo G, em ordem crescente de custo c_ij
3  v = 0; //onde v é o valor da solução
4  Enquanto (todos os vértices não tiverem os seus números de ligações mínimas satisfeito) faça
5        Calcular Valor ← min{c_ij : a_ij ∉ S};
6        Defina LRC ← { a_ij ∉ S : c_ij ≤ (1+α) x Valor};
7        Selecione aleatoriamente a_ij ∈ LRC;
8        Atualize S ← S ∪ a_ij;
9        v = v + c_ij;
10 Fim Enquanto;
11 Retorne S;
Fim Procedimento
```

Figura 5. Algoritmo de Construção.

5.2.2. FASE DE BUSCA LOCAL

Neste algoritmo *Busca_Local* parte-se de uma solução inicial gerada pelo algoritmo *Construção*, apresentado na Figura 5. A fase de busca local aproveita a solução inicial da fase de *Construção* e explora a vizinhança ao redor desta solução. Se um melhoramento é encontrado, a solução corrente é atualizada e novamente a vizinhança ao redor da nova solução é pesquisada. O processo se repete até nenhum melhoramento ser encontrado.

A o processo de pesquisa na vizinhança, aqui denominado *Viz(S)*, utilizada neste trabalho é realizada da forma descrita a seguir. São selecionados aleatoriamente 20% dos vértices do grafo e retiradas todas as arestas que estão conectadas a eles. O valor de 20% para o número de vértices a serem considerados foi determinado empiricamente e por apresentar boa relação custo-benefício. A seguir, um método semelhante ao construtivo descrito na Seção 5.2.1 é utilizado para reconectar o grafo satisfazendo as restrições de conectividade de cada vértice. Ao final, a melhor troca, entre aquelas que melhoraram o custo da solução satisfazendo as restrições de capacidade e satisfazendo as demandas dos clientes, é realizada. Na Figura 6 é apresentado o algoritmo *Busca_Local*.

```
Procedimento Busca_Local
1  Inicializar S' = Ø ; //melhor solução a ser armazenada
2  Defina Melhorou = TRUE;
3  Enquanto (Melhorou = TRUE) faça
4        Explore Viz(S);
5          Se (Solução S não for atualizada) Então
6            Melhorou = FALSE;
7          Senão
8            S' = Viz(S);
9        Fim Se
10 Fim Enquanto;
11 Retorne S;
Fim Procedimento
```

Figura 6: Algoritmo de busca local.

O GRASP combina as características de heurísticas de construção e de melhoramento. A estratégia deste método é acelerar o procedimento de busca local através da construção de soluções próximas, ou ainda, vizinhas a ótimos locais razoáveis e eventualmente ao ótimo global. A fase de construção melhora, portanto o desempenho médio da fase de melhoria, sendo o GRASP por este motivo mais rápido e eficiente que os métodos de multi-partida (busca local com soluções inicias construídas de forma totalmente aleatória) [10].

6. RESULTADOS COMPUTACIONAIS

Nesta Seção, serão apresentados os testes e resultados computacionais realizados sobre os algoritmos propostos neste trabalho. A seguir, serão apresentadas as instâncias

utilizadas para testes, o desempenho dos algoritmos, tanto sobre a qualidade das soluções fornecidas quanto pelo tempo de execução.

6.1. TESTES

Os testes foram realizados com instâncias que consistem de grafos criados com o gerador de instâncias, que foi desenvolvido pelo autor para o problema de sobrevivência de redes. Foram atribuídos valores de r(u), para cada vértice u do grafo, sorteado entre {1, 4}. O valor 4 foi escolhido como limite para r(u) por proporcionar uma sobrevivência satisfatória. Isto foi feito, dado a inexistência de uma biblioteca pública na internet que contenha instâncias para o problema abordado neste trabalho. Vale a pena ressaltar que o autor do trabalho entrou em contato com autores de trabalhos na área, solicitando as instâncias utilizadas em seus respectivos trabalhos e até o momento da conclusão deste trabalho não recebeu retorno dos mesmos.

Para realização dos testes, foram criadas 25 instâncias. Cada uma das 25 instâncias é um grafo conexo, onde o número de nós varia de 10 a 1000; e por fim o número de arestas varia de 17 a 5629. Cada aresta é atribuída um peso (custo, valor ou distância). Na Tabela 1 são apresentadas todas as instâncias e seus parâmetros.

Tabela 1. Instâncias utilizadas para teste.

Instâncias	Nº. vértices	Nº. arestas	Instâncias	Nº. Vértices	Nº. Arestas
ahsr1	10	17	ahsr14	75	99
ahsr2	10	20	ahsr15	75	79
ahsr3	10	33	ahsr16	100	100
ahsr4	20	31	ahsr17	100	116
ahsr5	20	30	ahsr18	100	201
ahsr6	20	49	ahsr19	250	249
ahsr7	30	49	ahsr20	250	249
ahsr8	30	73	ahsr21	250	298
ahsr9	30	79	ahsr22	500	999
ahsr10	50	75	ahsr23	500	501
ahsr11	50	100	ahsr24	1000	3008
ahsr12	50	73	ahsr25	1000	5629
ahsr13	75	75			

6.2. TESTES E RESULTADOS COMPUTACIONAIS DOS ALGORITMOS PROPOSTOS

As heurísticas gulosa simples (HGS), gulosa aleatorizada (HGA) e GRASP foram executados em um computador com a seguinte configuração: Processador AMD Sempron de 2.2 Ghz (nome-código *Thoroughbred* e 256Kbytes de cache L2), 256Mbytes de RAM e Sistema Operacional: Microsoft Windows XP Professional Service Pack 2.

Na Tabela 2, são apresentadas as soluções (custos das sobrevivências) encontradas pelo algoritmo guloso respectivamente, os tempos de execução (em segundos de CPU).

Para cada instância, o HGS foi executado da seguinte forma: esse algoritmo tem como critério guloso a prioridade de cada solicitação de intervenção, que é definida de acordo com critérios como valores de r(u), para cada vértice u do grafo, sorteado entre {1, 4}. Antes de iniciar sua execução, o algoritmo define as prioridades das solicitações e as ordenam de forma crescente.

Tabela 2. Desempenho da HGS.

Desempenho da HGS					
Instâncias	**Solução**	**Tempo**	**Instâncias**	**Solução**	**Tempo**
ahsr1	32	3.37	ahsr14	61	19.25
ahsr2	31	3.57	ahsr15	110	21.65
ahsr3	45	4.35	ahsr16	99	24.06
ahsr4	63	5.09	ahsr17	81	26.81
ahsr5	66	7.32	ahsr18	146	28.62
ahsr6	83	10.59	ahsr19	67	30.03
ahsr7	54	7.87	ahsr20	86	45.92
ahsr8	76	11.31	ahsr21	107	57.57
ahsr9	107	15.31	ahsr22	138	74.68
ahsr10	98	15.48	ahsr23	110	83.23
ahsr11	146	14.84	ahsr24	255	197.50
ahsr12	91	16.35	ahsr25	288	447.14
ahsr13	96	17.54			

O desempenho da HGA é apresentado na Tabela 3, mostrando a solução encontrada pelo mesmo e respectivamente o seu tempo de execução em segundos de CPU.

Para a execução das 25 instancias (ahsr1, ahsr2,..., ahsr25) utilizadas, a HGA foi executada utilizando os seguintes parâmetros:

- Critério de parada: 10000 iterações;

Tabela 3. Desempenho da HGA.

Desempenho da HGA					
Instâncias	**Solução**	**Tempo**	**Instâncias**	**Solução**	**Tempo**
ahsr1	28	4.23	ahsr14	61	21.25
ahsr2	31	4.66	ahsr15	108	25.65
ahsr3	44	5.03	ahsr16	99	24.06
ahsr4	63	6.00	ahsr17	81	26.81
ahsr5	66	8.35	ahsr18	146	29.62
ahsr6	83	12.92	ahsr19	66	31.66
ahsr7	59	12.68	ahsr20	86	46.75
ahsr8	76	12.23	ahsr21	105	58.99
ahsr9	105	15.14	ahsr22	136	75.80
ahsr10	98	15.54	ahsr23	110	86.98
ahsr11	143	18.89	ahsr24	253	192.05
ahsr12	91	19.31	ahsr25	282	437.41
ahsr13	95	22.52			

O desempenho do algoritmo GRASP é apresentado na Tabela 4 mostrando a solução encontrada pelo mesmo e respectivamente o seu tempo de execução em segundos de CPU.

Para a execução das 25 instancias (ahsr1, ahsr2,..., ahsr25) utilizadas, o algoritmo GRASP foi executado utilizando os seguintes parâmetros:

- Alfa: sorteado aleatoriamente entre 0 e 1 a cada iterações;
- Critério de parada do GRASP: 10000 iterações;

Tabela 4. Desempenho do algoritmo GRASP.

Desempenho do algoritmo GRASP					
Instâncias	**Solução**	**Tempo**	**Instâncias**	**Solução**	**Tempo**
ahsr1	27	4.14	ahsr14	61	20.65
ahsr2	31	4.00	ahsr15	99	25.14
ahsr3	42	4.89	ahsr16	99	23.81
ahsr4	59	5.35	ahsr17	81	26.05
ahsr5	66	7.50	ahsr18	142	28.66
ahsr6	83	11.68	ahsr19	66	30.50
ahsr7	55	12.03	ahsr20	86	46.23
ahsr8	74	12.00	ahsr21	105	57.68
ahsr9	102	13.05	ahsr22	124	74.33
ahsr10	98	14.44	ahsr23	92	81.66
ahsr11	139	18.11	ahsr24	248	189.20
ahsr12	91	18.33	ahsr25	267	372.11
ahsr13	95	22.00			

Da análise das Tabelas 2 e 3, verifica-se a obtenção de melhores soluções da HGA em relação ao HGS. Contudo, a HGA tem um custo computacional maior, devido a ser um método multi-partida. Já as Tabelas 3 e 4, verifica-se a obtenção de melhores soluções pelo algoritmo GRASP em relação à HGA, mais especificamente em treze instâncias. Ressalta-se que os valores dos parâmetros tanto para a HGA como para o GRASP foram determinados empiricamente de modo a permitir a obtenção de soluções de boa qualidade com desempenho razoável.

Na Seção 6.3 a seguir, serão realizadas análises e comparações mais aprofundadas a respeito da HGS, HGA e GRASP.

6.3. ANÁLISES E COMPARAÇÕES DOS RESULTADOS

Nesta seção, serão feitas as análises e comparações dos resultados obtidos nos testes, mas não será possível fazer uma comparação com outro resultado da literatura. Apesar de existir um outro trabalho [08], o mesmo utilizou-se de instâncias próprias para testes, que não foram disponibilizadas publicamente, o que inviabilizou a comparação.

6.3.1. ANÁLISES

Nesta Seção, serão analisados os resultados obtidos pelas as três técnicas propostas aplicadas para uma das 25 instâncias (ahsr1, ahsr2,..., ahsr25) utilizadas para testes. A instância utilizada foi a ahsr1 que possui 10 nós e 17 arestas. A cada aresta é atribuída um peso (custo, valor ou distância). Na Figura 7 é apresentado o grafo completo da instância analisada e seus parâmetros, onde: o número interior a cada vértice (círculo) indica a sua ordem; o número adjacente a cada vértice é o seu valor de r(u) (requisito de conectividade); e o número adjacente a cada aresta é o custo da mesma.

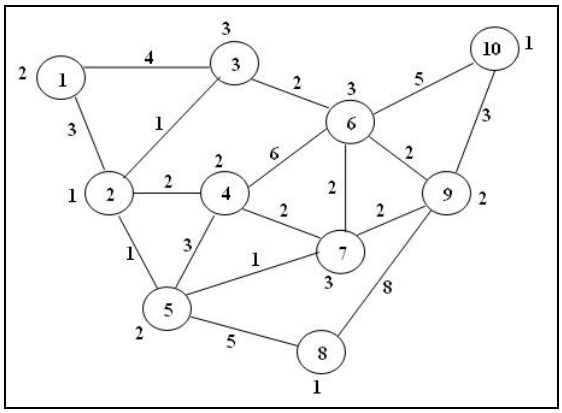

Figura 7. Grafo completo da instância analisada.

Através da execução da HGS, obteve-se uma topologia de rede satisfazendo as restrições impostas e com valor de solução 32, conforme pode ser observado na Figura 8, e com as respectivas arestas: 7-9; 6-10; 9-10; 5-8; 2-3; 3-6; 1-3; 4-5; 1-2; 4-7; 6-7. Analisando o grafo de sua topologia, tem-se os exemplos a seguir.

Exemplo 1:

- Tirar aresta 5-8;
- Colocar arestas 5-7;
- Como o vértice 8 ficou com o número de ligações menor que o mínimo, deve-se re-ligá-lo;

  A solução ficou maior em 4. Logo, esta troca não é interessante;

Exemplo 2:

- Tirar a aresta 4-5;
- Colocar a aresta 5-4;
- Como o vértice 4 ficou com o número de ligações menores que o mínimo, deve-se re-ligá-lo;
- Colocar aresta 2-4;
- A solução ficou com o mesmo valor (-3+1+2) = 0. Logo, troca.

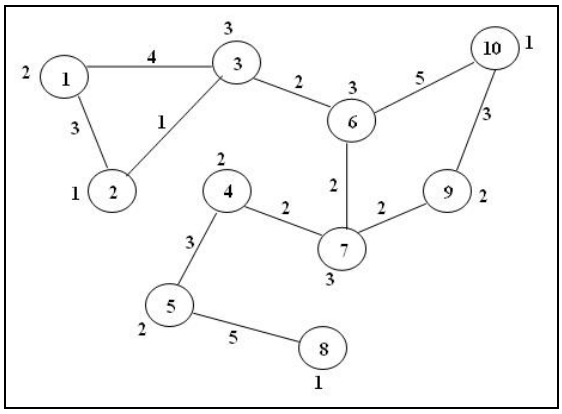

Figura 8. Solução da HGS.

Pela execução da HGA proposto, obteve-se para a instância de referência uma topologia de rede com valor de solução igual a 28, como pode ser observado na Figura 9, com as respectivas arestas: 2-3; 3-6; 2-5; 6-9; 2-4; 5-7; 5-8; 6-7; 4-7; 9-10; 1-3; 1-2. Comparando a topologia (grafo) da solução pela HGA com a solução da HGS, observou-se uma melhora da solução obtida pelo último, contudo, a um maior custo computacional.

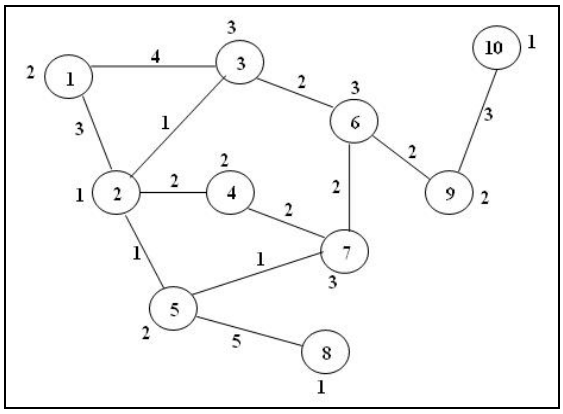

Figura 9. Solução da HGA.

O Algoritmo GRASP apresentou uma topologia com valor de solução 27 para a instância de referência, conforme pode ser visto na Figura 10, com as respectivas arestas: 1-3;

5-7; 3-6; 9-10; 2-3; 6-7; 6-9; 2-4; 5-8; 4-7; 1-2. Analisando o grafo da solução da HGA com o grafo da solução do algoritmo GRASP, observou-se uma melhora em sua topologia.

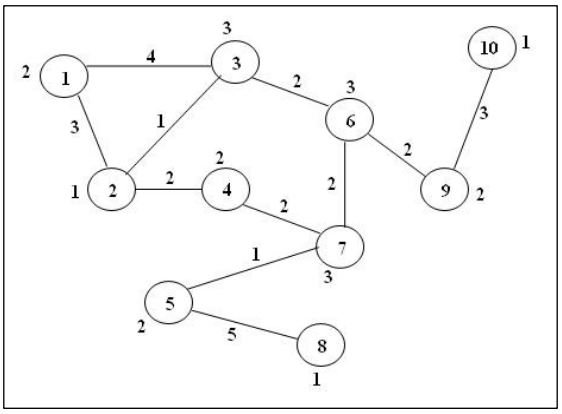

Figura 10. Solução do Algoritmo GRASP.

6.3.2. COMPARAÇÕES

Primeiramente, será realizada a comparação de desempenho entre a HGS e a HGA. No gráfico da Figura 11, é mostrado o número de melhorias da HGA sobre a HGS, em relação à solução, utilizando as 25 instâncias próprias para testes. As colunas representam os valores de soluções em menores, iguais e melhores em relação aos mesmos.

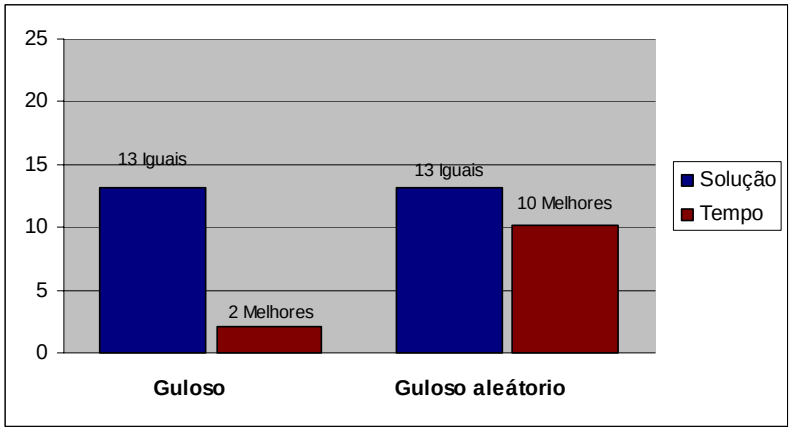

Figura 11. Comparação da HGS com a HGA.

Pelo resultado ilustrado no gráfico da Figura 11, verifica-se que a HGA superou a HGS na maioria das instâncias. A HGA obteve em 25 instâncias (descritas na Tabela 3), 10 melhores resultados, 13 resultados iguais e apenas 2 resultados inferiores quando comparados aos da HGS.

Agora, serão apresentadas análises a respeito da comparação de desempenho entre a HGA e o GRASP. De acordo com os resultados mostrados nas Tabelas 3 e 4, observa-se que, à medida que as dimensões crescem, o desempenho do GRASP a se torna melhor que a HGA com melhores valores nas soluções. Em termos de tempos computacionais exigidos, o GRASP apresenta vantagens em relação a HGA, sendo sempre mais rápido. Estas comparações podem ser melhores observadas na Figura 12.

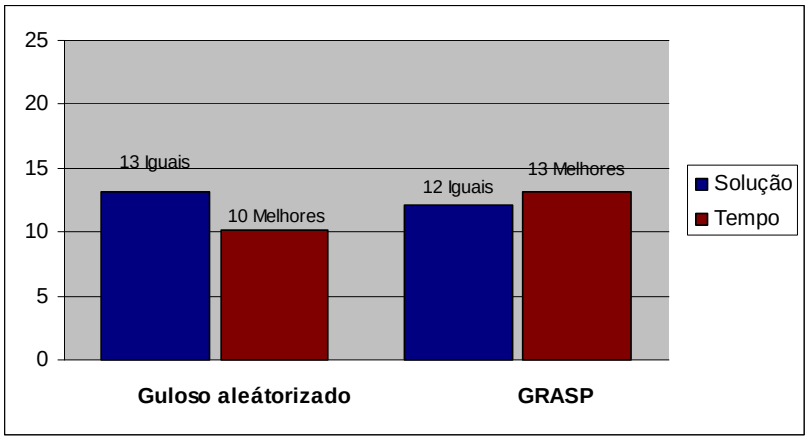

Figura 12. Comparação da HGA com o GRASP.

7. CONCLUSÕES E RECOMENDAÇÕES

Neste trabalho foi estudado o problema de sobrevivência de redes. Como contribuições, foram desenvolvidos algoritmos utilizando técnicas heurísticas e a metaheurística GRASP. O problema estudado neste trabalho está classificado como NP-Difícil, o que limita até o momento o uso de técnicas exatas para encontrar a solução para instâncias realísticas. Os algoritmos desenvolvidos neste trabalho foram testados em 25 instâncias de grafos completos, criados com o gerador de instâncias que foi desenvolvido pelos autores para o problema de sobrevivência de redes, onde o número de nós varia de 10 a 1000 e o número de arestas varia de 16 a 5629.

Neste trabalho foram realizados testes computacionais e comparações de desempenho, tanto no tocante a qualidade da solução quanto ao tempo de execução.

O objetivo deste trabalho é encontrar boas soluções num tempo computacional razoável. O objetivo foi plenamente alcançando, com a metaheurística GRASP sempre obtendo os melhores resultados.

As recomendações e trabalhos futuros para este trabalho são:

- realizar a implementação de outras técnicas ou heurísticas, de modo a verificar a possibilidade de obtenção de melhores soluções ou menores tempos computacionais.

- realizar a paralelização da heurística GRASP posposta com intuito de melhorar o desempenho da técnica.

REFERÊNCIAS

[01] BICUDO, M. D. D.; DUARTE, O. C. M. B. Um Mecanismo de Proteção em Redes WDM em Malha. Dissertação de Mestrado, Universidade Federal do Rio de Janeiro, Grupo de Teleinformática, Rio de Janeiro, 2005.

[02] BICUDO, M. D. D. Sobrevivência em Redes Ópticas Transparentes. Dissertação de Mestrado, Universidade Federal do Rio de Janeiro, Engenharia de Elétrica, Rio de Janeiro, 2005.

[03] BLUM, C.; ROLI, A. Metaheuristics in Combinatorial Optimisation: Overview and Conceptual Comparison. Technical Report TR/IRIDIA/2001-13, IRIDIA, Université Libre de Bruxelles, Belgium, 2001.

[04] CANCELA, H.; URQUHART, M. E.; RUBINO, G. HEIDI: Una Herramienta de apoyo para el diseño de Redes de comunicaciones. Research Report LIMOS/2005, Faculatad de Ingeniería, Uruguay, 2005.

[05] CANCELA, H.; ROBLEDO, F.; RUBINO, G. A GRASP Algorithm for Designing a Wide Area Network Backbone. Research Report LIMOS/2005, Faculatad de Ingeniería, Uruguay, 2005.

[06] DETONI, A. A. Sistema de Apoio à Decisão para Planejamento de Redes de Telecomunicações Baseado em Camadas. Dissertação de Mestrado, Engenharia Elétrica, Universidade Federal do Espírito Santo, Vitória, pp. 1-13, 2001.

[07] MELLO, O. D.; SILVA, M. C.; TAVARES, H. M. F. "Sobrevivenciabilidade em Redes de Telecomunicações'', XXVIII Simpósio Brasileiro de Pesquisa Operacional, 1, 1137-1142, 1996.

[08] ROBLEDO, F. Optimization and Survivability of telecommunication Networks. Research Report LIMOS/2005, Faculatad de Ingeniería, Uruguay, 2005.

[09] SILVEIRA, C. M. D. Análise de Métodos Heurísticos de Características Gulosa. Dissertação de Mestrado, Universidade Federal do Rio Grande do Sul, Instituto de Informática, Porto Alegre, 1999.

[10] VIANNA, D. S. Heurística Híbridas para o Problema da Filogenia. Tese de Doutorado, PUC-Rio, Departamento de Informática, Rio de Janeiro, 2004.

[11] KERIVIN, H.; MAHJOUB, A. R. Design of Survivable Networks: A Survey. Research Report LIMOS/2005/RR-05-04, Faculatad de Ingeniería, Uruguay, 2005.