GRASP APLICADO AO PROBLEMA DO CONJUNTO MÍNIMO DE OBSERVADORES PARA MONITORAMENTO DE REDES

[1]Diego Giacomelli Cardoso, [1]Fernando Ávila Fossi Silveira
[1,2]Renato Elias Nunes de Moraes, [1,2]Rodolfo da Silva Villaça
[1]Programa de Pós-Graduação em Informática (PPGI)
[2]Departamento de Tecnologia Industrial (DTI)
Universidade Federal do Espírito Santo (UFES) – Vitória/ES
dgiacomellic@gmail.com, fernandofossi@gmail.com,
renato.moraes@ufes.br, rodolfo.villaca@ufes.br

RESUMO

O monitoramento de redes de computadores consiste em observar o tráfego nos enlaces da rede para auxiliar a tomada de decisões de gerenciamento. Dentre diversas estratégias de monitoramento destaca-se a geração da Matriz de Tráfego, que consiste em contabilizar a quantidade de *Bytes* (ou pacotes) entre todos os pares de nós da rede. Para construção da Matriz de Tráfego é necessária a instalação de observadores nos nós roteadores da rede, o que consome recursos de armazenamento e processamento. Dessa forma, encontrar o Conjunto Mínimo de Observadores para Geração de Matrizes de Tráfego (CMO-MT) é uma importante tarefa para redução de custos de monitoramento. O problema CMO-MT consiste em, dado um conjunto de nós e enlaces de rede, determinar conjunto mínimo de observadores necessários para monitorar a rede e auxiliar a geração da Matriz de Tráfego. Neste contexto, este artigo propõe uma heurística gulosa aleatória, combinada com uma busca local, para formar um algoritmo GRASP que soluciona o problema CMO-MT. Extensos experimentos computacionais são apresentados para validar a eficácia da solução proposta.

PALAVRAS CHAVE. Matriz de tráfego, GRASP, Traversal Mínima.

Tópicos: TEL&SI – PO em Telecomunicações e Sistemas de Informações; TAG – Teoria e Algoritmos em Grafos; OC – Otimização Combinatória.

ABSTRACT

Network monitoring consists on the observation of the data traffic that is present in the links of a network to support management decisions. Among several monitoring strategies, it is worth mentioning the generation of the Traffic Matrix, which consists of counting the amount of Bytes (or packets) carried between each pair of nodes in the network. To construct the Traffic Matrix, it is necessary to install observers on the network routers, which consumes storage and processing resources. Thus, finding the Minimum Set of Observers for generating Traffic Matrices (CMO-MT) is an important task for cost reduction in network monitoring. The CMO-MT problem consists of, given a set of network nodes and links, determining the minimum set of observers required to monitor the network and support the generation of the Traffic Matrix. This paper proposes a random greedy heuristic, combined with a local search, to form a GRASP algorithm that solves the CMO-MT problem. Extensive computational experiments are presented to validate the effectiveness of the proposed solution.

KEYWORDS. Traffic matrix, GRASP, Minimum Hitting Set.

Topics: TEL&SI – OR in Telecommunications and Information Systems; GTA – Graph Theory and Algorithms; CO – Combinatorial Optimization.

1. Introdução

Problemas de monitoramento de redes de computadores consistem em observar uma grande fração do tráfego de uma rede. Para alcançar esse objetivo é necessário observar vários enlaces ao mesmo tempo, uma vez que apenas uma parte do tráfego pode ser vista por um único ponto de medição de uma grande rede. Uma maneira de atingir esse objetivo é selecionar alguns roteadores (ou elementos que se comportem como um roteador) na rede para realizar o monitoramento do tráfego. Os nós selecionados são chamados de observadores.

O posicionamento de um observador incorre em custos de implantação, hardware/software, armazenamento e manutenção. Este trabalho lida com o problema de localização de observadores, que consiste em selecionar o menor número de posições onde instalar os observadores na rede [Cantieni et al., 2006; Suh et al., 2006; Breitbart et al., 2009].

Identificar os locais estratégicos para os observadores de tráfego é um problema difícil que tem atraído interesse significativo na literatura [Cantieni et al., 2006]. Várias soluções têm sido propostas para diferentes contextos. Por exemplo, em [Jamin et al., 2000], os autores propõem a inserção de dispositivos de medição para a construção de mapas de distância. Assume-se que quanto maior a distância entre dois nós, maior será a latência de rede, daí a necessidade desse mapa. Outros trabalhos têm abordado o problema de posicionamento dos observadores para monitoramento ativo (aquele cuja medição interfere no tráfego da rede) da infraestrutura para medir atrasos e detectar falhas de enlace [Horton e López-Ortiz, 2003; Nguyen e Thiran, 2004; Bejerano e Rastogi, 2006].

Considerando monitoramento passivo (aquele cuja medição não interfere no tráfego da rede), Suh et al. [2006] tratam o problema de posicionar os observadores e definir taxas de amostragem de pacotes (*sampling rate*) a fim de maximizar a fração de fluxos a ser amostrado em função dos custos de operação e instalação. Cantieni et al. [2006] também determinam o conjunto mínimo de observadores e suas taxas de amostragem a fim de obter uma medição de alta precisão com baixo consumo de recursos.

Este trabalho trata o problema do Conjunto Mínimo de Observadores para Geração de Matrizes de Tráfego (CMO-MT). A geração da Matriz de Tráfego de uma rede é uma estratégia de gerenciamento que consiste em contabilizar a quantidade de informação trafegada entre todos os pares de nós da rede, podendo ser medido em *Bytes* ou pacotes. Uma discussão mais aprofundada sobre Matrizes de Tráfego foge ao escopo deste trabalho. Para mais informações sobre o assunto consultar [Zhao et al., 2005; Paul Tune, 2013; Silveira et al., 2016].

O problema CMO-MT consiste em, dado o conjunto de nós (roteadores) da rede de computadores e um conjunto de enlaces entre os pares de nós, encontrar os caminhos entre os nós e o conjunto mínimo de observadores para monitorar todos os caminhos encontrados. O problema CMO-MT foi inicialmente proposto e tratado em Silveira et al. [2016]. Naquela solução consideram-se caminhos fixos entre os pares de nós, definidos pelo algoritmo de menor caminho de Dijkstra [Dijkstra, 1959]. Foi mostrado, ainda, que é possível tratar o CMO-MT como um problema de Travessal Mínima (*Minimum Hitting Set - MHS*) buscando minimizar o conjunto de observadores para o monitoramento de cada caminho. O *MHS* é equivalente ao Problema da Cobertura de Conjuntos (*Set Cover Problem - SCP*) [Ausiello et al., 1980; Moshkovitz, 2012], sendo ambos importantes membros da classe NP-Difícil [Garey e Johnson, 1979; Caprara et al., 1999]. Por causa dessas características, Silveira et al. utilizaram a heurística gulosa proposta por Chvatal [1979] para minimizar a quantidade de observadores necessários.

Diferente de Silveira et al. [2016], este trabalho não restringe o conjunto de caminhos aos menores caminhos calculados pelo algoritmo de Dijkstra. Na solução proposta para o problema CMO-MT, além do conjunto mínimo de observadores, os algoritmos implementados encontram os caminhos entre os nós capazes de serem cobertos pelo conjunto mínimo de observadores. O trabalho propõe uma heurística gulosa aleatória combinada com uma busca local para formar um algoritmo GRASP que soluciona o CMO-MT e reduz ainda mais o número de observadores necessários para geração da Matriz de Tráfego da rede.

O restante deste artigo está organizado na seguinte forma: a Seção 2 descreve o modelo da rede e formaliza a definição do CMO-MT como um *MHS*. A Seção 3 apresenta uma proposta do algoritmo GRASP para o problema CMO-MT. Extensos resultados computacionais são apresentados na Seção 4 enquanto na Seção 5 conclui-se o artigo e faz-se o apontamento de trabalhos futuros.

2. Modelo de Rede e Formulação do Problema

A definição de topologia de rede de computadores utilizada neste trabalho é detalhada nesta seção. Uma topologia de rede pode ser definida como um grafo conexo não direcionado $G = (V, A)$ onde V é o conjunto de vértices (roteadores) e A o conjunto de arestas (u, v), com $u, v \in V$ (isto é, enlaces entre roteadores). No contexto deste artigo, a cada aresta $(u, v) \in A$ é atribuido um custo unitário.

A unidade básica de informação trafegando na rede é chamada de pacote. Todo pacote está associado a um par de vértices $[v_i, v_e]$, representado como $p_{[v_i,v_e]}$, tal que $v_i \in V$ é o vértice origem (ingresso) e $v_e \in V$ é o vértice destino (egresso) do pacote. Um pacote se propaga na rede partindo do vértice de ingresso v_i e chegando ao vértice de egresso v_e, trafegando pelo conjunto de arestas $A_{[v_i,v_e]} \subset A$ formado entre os vértices v_i e v_e. O conjunto de arestas $A_{[v_i,v_e]}$ determina o caminho entre os vértices v_i e v_e. Como uma aresta é representada pelo par de vértices (u, v) pode-se representar também um caminho pela sequência de vértices que formam as arestas da seguinte forma: $C_{[v_i,v_e]} = \{u, v, w \in V \mid (u, v) \text{ e } (v, w) \in A_{[v_i,v_e]}\}$. Como exemplo, a Figura 1 mostra um grafo com os vértices $V = \{0, 1, ..., 7\}$ conectados pelas arestas exibidas como linhas entre eles. Um pacote $p_{[0,7]}$ passa pelo conjunto de arestas $A_{0,7} = \{(0, 3), (3, 4), (4, 7)\}$, consequentemente, pela sequência de nós $C_{[0,7]} = \{0, 3, 4, 7\}$.

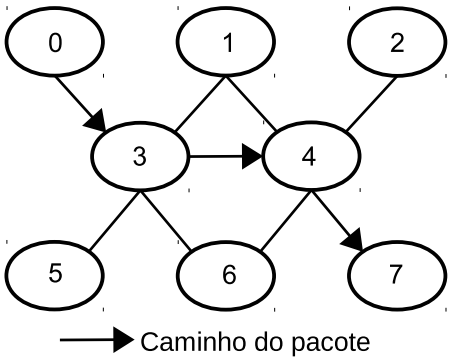

Figura 1: Exemplo de um pacote viajando em um caminho na rede.

Para a construção de uma Matriz de Tráfego é necessário monitorar o tráfego da rede. Esse monitoramento consiste em contar quantos pacotes trafegaram entre todos os pares de vértices ingresso-egresso $[v_i, v_e]$. Quando um pacote $p_{[v_i,v_e]}$ viaja de v_i para v_e, ele passa por todos os vértices no caminho $C_{[v_i,v_e]}$. Portanto, para contar quantos pacotes trafegaram entre um determinado par de vértices ingresso-egresso $[v_i, v_e]$, pode-se instalar um único observador em um vértice $u \in C_{[v_i,v_e]}$. Para monitorar toda a rede, deve-se instalar observadores em um conjunto de vértices $O \subseteq V$, tal que o conjunto O cubra todos os caminhos $C_{[v_i,v_e]}$ da rede. Como exemplo, na Figura 1, o conjunto mínimo de observadores é $O = \{3, 4\}$, pois estes nós cobrem todos os caminhos presentes na rede, logo todos os pacotes passam por eles.

O CMO-MT, que consiste em encontrar o conjunto de observadores O de menor tamanho $|O|$ que cubra todos os caminhos $C_{[v_i,v_e]}$ da rede, pode ser modelado como um *MHS*. A definição do *MHS* é a seguinte: dados um conjunto S de m elementos e uma coleção $\mathcal{K}$ de n conjuntos destes elementos, encontrar o subconjunto $S^* \subseteq S$, de cardinalidade $|S^*|$ mínima, tal que $S^* \cap K \neq \emptyset$, para todo $K \in \mathcal{K}$. Isto é, cada subconjunto $K \in \mathcal{K}$ deve possuir, pelo menos, um elemento $s \in S^*$.

Dessa maneira, a definição do CMO-MT como um *MHS* é a seguinte: dado uma rede, definida como um grafo $G = (V, A)$, definir: (1) o conjunto de caminhos $\mathcal{C} = \{C_1, C_2, \ldots, C_n\}$ tal que n é a quantidade de pares de nós na rede e $n \leq |V|(|V| - 1)$, e (2) o conjunto mínimo de observadores $O \subseteq V$ de cardinalidade $|O|$ mínima, tal que $O \cap C_j \neq \emptyset$, $j = 1, 2, \ldots, n$. Como entre um par de nós $[u, v]$ em um grafo podem existir vários caminhos, a escolha do conjunto de observadores O depende dos caminhos escolhidos. Como exemplo, a Figura 2 mostra um grafo com os nós $V = \{0, 1, ..., 4\}$ conectados pelas arestas exibidas como linhas entre eles. Se for utilizado um conjunto de caminhos contendo os menores caminhos entre os pares de vértices (menor quantidade de saltos, no caso de arestas com peso unitário), o conjunto mínimo de observadores será $O = \{0, 2, 1\}$ (ou $O = \{1, 2, 3\}$), com cardinalidade $|O| = 3$. Por outro lado, se o conjunto de caminhos for gerado partindo-se de uma árvore centrada no vértice 2, o conjunto mínimo de observadores passa a ser $O = \{2\}$ com cardinalidade $|O| = 1$.

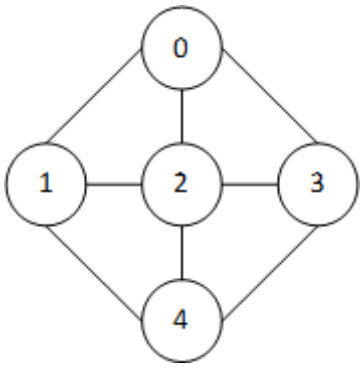

Figura 2: Exemplo da escolha do conjunto de caminhos

3. GRASP para o CMO-MT

Um GRASP (*Greedy Randomized Adaptive Search Procedure*) [Feo e Resende, 1989], procedimento de busca gulosa, aleatória e adaptativa, pode ser visto como uma metaheurística que captura boas qualidades de um algoritmo guloso e de procedimentos de construção aleatórios [Martins et al., 1999].

O GRASP é um método iterativo de múltiplas partidas no qual cada iteração consiste de dois passos: um passo de partida e um passo de melhoramento. No passo de partida, utiliza-se um algoritmo construtivo aleatório para se obter uma solução viável, não necessariamente a melhor, e a cada iteração do GRASP, o algoritmo construtivo aleatório é capaz de obter uma nova solução diferente da anterior. No passo de melhoramento, utiliza-se uma estratégia de busca local para melhorar a qualidade da solução obtida no passo de partida. A melhor solução encontrada em todas as iterações do GRASP é retornada como resultado. A proposta de uma estratégia aleatória para solucionar o CMO-MT é feita na Subseção 3.1. Na Subseção 3.2 é discutida uma proposta de busca local para o problema.

O Algoritmo 1 mostra o pseudocódigo do GRASP para solucionar o problema CMO-MT. O algoritmo recebe como entrada um Grafo $G = (V, A)$ que representa a topologia da rede. Na linha 2, o conjunto de observadores O^* é inicializado como o conjunto de vértices do grafo e na linha 3, o conjunto de caminhos $\mathcal{C}^*$ é inicializado como um conjunto vazio.

O laço da linha 4 até a linha 8 implementa o procedimento iterativo de busca aleatorizada. O laço é executado enquanto o critério de parada $crit\_parada$ não é alcançado. Na linha 5, a estratégia de construção aleatória computa o conjunto de caminhos $\mathcal{C}$, onde existe um único caminho $C_{[u,v]}$ para cada par de vértices $u, v \in V$. Tais caminhos não são necessariamente os menores, podem ser quaisquer caminhos que liguem os pares de vértcies. Além disso, a estratégia aleatória computa também o conjunto de observadores O que cubram todos os caminhos de $\mathcal{C}$. Uma discussão de como a estratégia aleatória funciona ocorre na Seção 3.1.

Na linha 6 do Algoritmo 1, é executada a estratégia de busca local com o objetivo de encontrar um conjunto de observadores O' menor do que o conjunto O encontrado pelo passo anterior. Na linha 7, atualiza-se a melhor solução O^* e $\mathcal{C}^*$ caso a cardinalidade do conjunto de observadores O' calculado seja menor do que a cardinalidade do conjunto da melhor solução O^* obtida pelo GRASP. Por fim, o GRASP retorna a melhor solução obtida e o respectivo conjunto de caminhos na linha 9.

Algoritmo 1: Algoritmo GRASP para solucionar o problema CMO-MT

Entrada: Grafo $G = (V, A)$
Saída: Conjunto de nós de observadores O^*, Conjunto de caminhos $\mathcal{C}^*$

```
1  início
2      O* ← V
3      C* ← ∅
4      enquanto crit_parada faça
5          O, C ← SMV-A(G)
6          O' ← BuscaLocal(O, C)
7          Atualiza(O', C, O*, C*)
8      fim enqto
9      retorna O*, C*
10 fim
```

3.1. Algoritmo Construtivo Aleatório

O algoritmo construtivo aleatório baseia-se na geração de caminhos considerando diferentes árvores contidas em um grafo. A execução do algoritmo pode ser divididida em dois passos: o primeiro para gerar os caminhos, e o segundo para calcular os observadores utilizando os caminhos do primeiro passo. Neste trabalho, a construção aleatória é nomeada como SMV-A e pode ser vista no Algoritmo 2. Ele recebe como entrada um grafo $G = (V, A)$ e tem como saída, o conjunto de observadores O e o conjunto de caminhos $\mathcal{C}$.

Na linha 2, o SMV-A constrói o conjunto de caminhos $\mathcal{C}$ a partir de G. Na linha 3, o SMV-A obtém o conjunto de observadores O utilizando o conjunto de caminhos $\mathcal{C}$ obtidos no passo anterior. Por fim, o algoritmo retorna $\mathcal{C}$ e O na linha 4.

O primeiro passo do SMV-A, visto na linha 2 do Algoritmo 2, é o Algoritmo 3 nomeado como EscolheCaminho. Sua entrada é um grafo G e sua saída é um conjunto de caminhos $\mathcal{C}$. Na linha 2 computa-se uma árvore aleatória $T = (V_t, A_t)$, onde $V_t = V$ e $A_t \subseteq A$. É utilizado uma variação aleatória do algoritmo de Dijkstra [Dijkstra, 1959] para calcular T. O algoritmo de Dijkstra original escolhe, dentre os vértices candidatos, o vértice com a menor distância até a raíz. Um vértice é considerado candidato se ele ainda não pertencer à algum caminho partindo da raíz. A distância de um vértice pode ser denominado como $d(v)$, que representa a quantidade de saltos da raiz até o vértice v.

No Dijkstra aleatório, acrescenta-se uma Lista de Candidatos Restrita (LCR). A LCR é formada por todos os vértices candidatos tais que

$$d(v) \leq d(v)_{min} + \alpha(d(v)_{max} - d(v)_{min}) \tag{1}$$

onde $\alpha \in [0, 1]$, $d(v)_{min} = \min\{d(v) : v \in V \text{ e } v \text{ é um vértice candidato}\}$ e $d(v)_{max} = \max\{d(v) : v \in V \text{ e } v \text{ é um vértice candidato}\}$. Além da LCR, o vértice raíz também é escolhido aleatoriamente.

Após calcular uma árvore, o algoritmo EscolheCaminho constrói um conjunto de caminhos $\mathcal{C}$ para todo par de vértices $(u, v) \in V$. Isso ocorre no algortimo 3 da linha 3 até a linha 5. A estratégia `MontaCaminho(`T, u, v`)` funciona da seguinte maneira: constrói-se um caminho

$C_{[u,s]}$ partindo do vértice u até a raíz s de T. Depois, a estratégia monta um segundo caminho $C_{[v,w]}$ partindo de v até um vértice w qualquer pertence ao caminho $C_{[u,s]}$. Finalmente a estratégia une os dois caminhos gerados $C_{[u,w]}$ e $C_{[w,v]}$, formando o caminho $C_{[u,v]}$. Por fim, o algoritmo EscolheCaminho retorna o conjunto de caminhos $\mathcal{C}$ na linha 6.

O segundo passo da construção aleatória, visto na linha 3 do Algoritmo 2, é o algoritmo 4. Foi proposto por [Silveira et al., 2016] e é nomeado como SMV. A entrada do algoritmo é o conjunto de caminhos $\mathcal{C}$ calculado pelo algoritmo EscolheCaminho e a sua saída é o conjunto de observadores O necessários para cobrir todos os caminhos de $\mathcal{C}$.

O SMV funciona como segue. Na linha 2 o algoritmo inicia o conjunto O como vazio. O laço da linha 3 até a linha 7 é executado enquanto houver caminhos não cobertos por algum observador. Na linha 4, um novo observador obs é escolhido. A escolha de um observador é sempre o vértice que cobrir a maior quantidade de caminhos naquela iteração. Na linha 5 o conjunto de caminhos $\mathcal{C}$ é atualizado, removendo os caminhos cobertos pelo observador escolhido obs. Na linha 6, o SMV atualiza a lista de observadores O adicionando obs. Por fim, o algoritmo retorna o conjunto de observadores na linha 8.

Algoritmo 2: Algoritmo SMV-A

Entrada: Grafo $G = (V, A)$
Saída: Conjunto de Observadores O, Conjunto de caminhos $\mathcal{C}$
1 **início**
2 $\quad \mathcal{C} \leftarrow$ `EscolheCaminhos`(G)
3 $\quad O \leftarrow$ `SMV`$(\mathcal{C})$
4 $\quad$ **retorna** $O, \mathcal{C}$
5 **fim**

Algoritmo 3: Algoritmo EscolheCaminho

Entrada: Grafo $G = (V, A)$
Saída: Conjunto de caminhos $\mathcal{C}$
1 **início**
2 $\quad T \leftarrow$ `DijkstraAleatorio`(G)
3 $\quad$ **para todo** $(u, v) \in G.V$ **faça**
4 $\quad\quad \mathcal{C} \leftarrow$ `MontaCaminho`(T, u, v)
5 $\quad$ **fim para todo**
6 $\quad$ **retorna** $\mathcal{C}$
7 **fim**

Algoritmo 4: Algoritmo SMV

Entrada: Conjunto de caminhos $\mathcal{C}$
Saída: Conjunto de observadores O
1 **início**
2 $\quad O \leftarrow \emptyset$
3 $\quad$ **enquanto** $\mathcal{C} \neq \emptyset$ **faça**
4 $\quad\quad obs \leftarrow$ `EscolheObservador`$(V \setminus O, \mathcal{C})$
5 $\quad\quad \mathcal{C} \leftarrow$ `RemoveCaminhos`$(\mathcal{C}, obs)$
6 $\quad\quad O \leftarrow O \cup \{obs\}$
7 $\quad$ **fim enqto**
8 $\quad$ **retorna** O
9 **fim**

3.2. Busca Local

A busca local tem por objetivo melhorar uma solução. Para o problema proposto, a busca local tenta reduzir o número de observadores, isto é, encontrar um novo conjunto de observadores com tamanho menor do que a solução obtida pelo passo de partida.

O algoritmo de busca local utiliza a vizinhança de *1-flip* como descrita no trabalho de Bilal et al. [2013], onde, em cada movimento, um único observador pode ser inserido ou removido de acordo com a avaliação da função objetivo. A estratégia da busca local implementada é a melhor aprimorante, ou seja, avalia-se todas as soluções na vizinhança da solução corrente e retorna a melhor encontrada.

Neste trabalho, a busca local é nomeada como BL e pode ser vista no Algoritmo 5. Ele recebe como entrada o conjunto de vértices V, o conjunto de observadores O e o conjunto de caminhos $\mathcal{C}$ e retorna como saída, o conjunto O', cujo tamanho é menor ou igual ao tamanho de O.

O algoritmo BL funciona como segue. Na linha 2, o conjunto de observadores O' é inicializado. O laço da linha 3 até a linha 8 é executado até que o algoritmo encontre um ótimo local. Na linha 4 o algoritmo busca qual melhor vértice a ser feito um *flip* (melhor solução na vizinhança da solução corrente O'), isto é, qual vértice que, ao ser inserido ou removido do conjunto O', apresenta melhor ganho da função objetivo. O método `MelhorCandidato()` retorna um valor diferente de $null$ para a variável c_j quando uma solução melhor é encontrada na vizinhança. O condicional da linha 5 verifica a variável c_j. Se foi encontrada uma solução vizinha aprimorante, c_j não é nulo e é executado a linha 6, onde é feito o *flip* de c_j e o novo conjunto de observadores é armazenado em O'. Caso contrário, isto é, quando c_j for nulo, significa que foi encontrado um ótimo local e o loop finaliza. Finalmente, a melhor solução obtida é retornada na linha 9.

Algoritmo 5: Algoritmo BL

Entrada: Conjunto de Vértices V, Conjunto de observadores O, Conjunto de caminhos $\mathcal{C}$
Saída: Conjunto de Observadores O'

```
1  início
2      O' ← O
3      repita
4          c_j ← MelhorCandidato(V, O', C)
5          se c_j ≠ null então
6              O' ← FlipCandidato(V, O', c_j)
7          fim se
8      até c_j ≠ null;
9      retorna O'
10 fim
```

3.3. Ajustes dos parâmetros do GRASP

O parâmetro α é utilizado pelo GRASP para controlar a LCR dentro da construção aleatória (ver Seção 3.1). Ele pode assumir valores no intervalo $[0.0, 1.0]$. Se $\alpha = 0.0$, então a escolha dos elementos da LCR se torna puramente gulosa. Se $\alpha = 1.0$ então a escolha dos elementos da LCR se torna puramente aleatória.

A utilização de um único valor fixo para α dificulta encontrar soluções de boa qualidade que podem ser encontradas quando se utiliza outros valores para α. Isto é mostrado em [Prais e Ribeiro, 2000] e os autores propuseram uma estratégia reativa para o ajuste do α automaticamente.

Na estratégia proposta, o parâmetro α é selecionado dentro de um conjunto discreto de valores $A = \{\alpha_1, \alpha_2, \ldots, \alpha_n\}$. Probabilidades P_i são associadas a cada α_i com valores iniciais iguais a $P_i = 1/n$, $i = 1, \ldots, n$. Além disso, a distribuição de probabilidade é atualizada regularmente a cada bloco de repetição fixado em B iterações seguindo as seguintes equações:

$$P_i = q_i / \sum_{j=1}^{n} q_j \tag{2}$$

onde

$$q_i = (1/M_i)^{\delta} \tag{3}$$

O valor M_i é a média das soluções encontradas utilizando o parâmetro α com o valor α_i. O expoente $\delta > 0$ é utilizado para atenuar a atualização dos valores das probabilidades P_i. Usualmente, $\delta = 8$ [Prais e Ribeiro, 2000].

4. Avaliações e Resultados

Os experimentos computacionais realizados para os algoritmos da Seção 3 são apresentados nesta seção. Especificamente, foram avaliados os algoritmos SMV (Algoritmo 4), SMV-A (Algoritmo 2), BL (Algoritmo 5) e GRASP (Algoritmo 1). Foram realizados dois experimentos, separados pelos conjuntos de entrada utilizados: o primeiro utilizando um conjunto de redes artificiais e o segundo um conjunto de redes reais. Todos os resultados foram gerados em um sistema com 8GB de memória RAM, processador Intel® Core™ i5-2400 de 3GHz e SO Ubuntu 16.04.

Para ambos os experimentos realizados, os algoritmos foram configurados e executados de maneira independente. O SMV é equivalente ao algoritmo SMV-A quando executado com parâmetro α igual a zero, isto é, foram geradas árvores utilizando o algoritmo de Dijkstra [Dijkstra, 1959] original. A busca local BL foi executada em cima dos resultados obtidos pelo SMV-A. Para o GRASP o critério de parada $crit\_parada$ foi definido como 10000 iterações sem melhorias com a atualização da distribuição de probabilidades sendo atualizadas a cada $B = 100$ iterações.

O primeiro experimento foi realizado com as redes artificiais. Foi criado um conjunto de redes com vértices V, $|V| \in \{100, 200, 300, 400, 500\}$, distribuídos aleatoriamente em um plano. Para cada tamanho de V, dez instâncias diferentes foram geradas, totalizando uma quantidade de cinquenta instâncias. Os resultados apresentados mostram a média, para cada tamanho de V, das dez execuções independentes de cada algoritmo avaliado. Os resultados obtidos para esse experimento são apresentados na Tabela 1 e no gráfico da Figura 3.

A Tabela 1 compara os resultados obtidos pelos algorimos SMV, SMV-A, BL e GRASP para redes artificiais. A primeira coluna mostra a quantidade de vértices das redes. A segunda coluna apresenta os resultados extraídos do SMV em quantidade média de observadores calculada pelo algoritmo. A terceira, quarta e quinta colunas apresentam os resultados obtidos pelos algoritmos SMV-A, BL e GRASP, respectivamente. Nestas últimas colunas, as informações são divididas em duas subcolunas, sendo a primeira subcoluna a quantidade média de observadores calculada pelos algoritmos e a segunda subcoluna a distância percentual média desses algoritmos em relação ao SMV.

Analisando a Tabela 1, percebe-se que todos os algoritmos propostos neste trabalho (SMV-A, BL e GRASP) obtiveram resultados melhores que o SMV, isto é, todos algoritmos propostos conseguiram reduzir a quantidade de observadores necessários para monitorar e gerar a Matriz de Tráfego em todas as redes artificiais, sendo que o GRASP conseguiu os melhores resultados. Esses resultados são exibidos graficamente na Figura 3. Nesta figura, o eixo horizontal representa o tamanho da rede em quantidade de vértices e o eixo vertical representa a quantidade média de observadores calculados pelos algoritmos.

Com os resultados obtidos nesse primeiro experimento é possível afirmar que o fator determinante para a melhora dos resultados foi a utilização dos caminhos baseados em árvore, visto que o algoritmo guloso SMV-A apresentou resultados melhores que o SMV em todas as baterias de testes. Além disso pode-se afirmar que a estratégia de busca local funciona, visto que

Tabela 1: Comparação dos resultados obtidos pelos algoritmos, SMV, SMV-A, BL e GRASP para as redes artificiais.

	SMV	SMV-A		BL		GRASP	
$\|V\|$	Obser.	Obser.	Dist(%)	Obser.	Dist(%)	Obser.	Dist(%)
100	58,90	31,70	-46,18	31,30	-46,86	**29,60**	**-49,75**
200	118,50	68,50	-42,19	68,40	-42,28	**64,80**	**-45,32**
300	175,30	103,70	-40,84	102,80	-41,36	**99,90**	**-43,01**
400	235,80	139,90	-40,67	138,30	-41,35	**135,00**	**-42,75**
500	295,00	174,40	-40,88	173,20	-41,29	**169,70**	**-42,47**

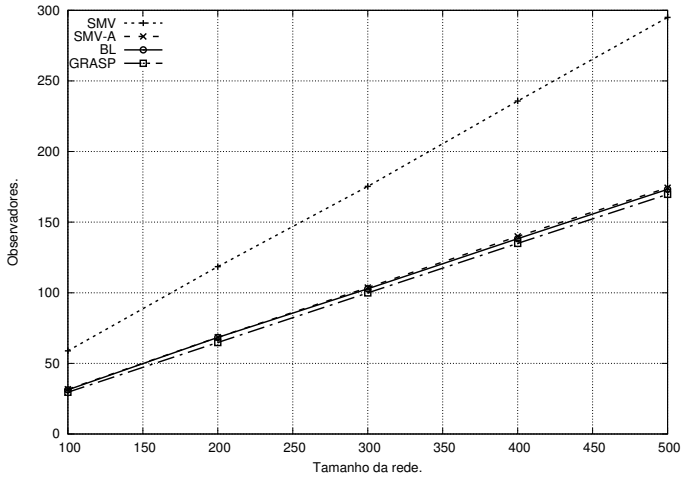

Figura 3: Gráfico dos resultados obtidos pelos algoritmos, SMV, SMV-A, BL e GRASP para as redes artificiais.

o GRASP consegue resultados melhores dos que obtidos pelos algoritmos SMV e SMV-A, que utilizam estratégias gulosas para geração dos caminhos.

O segundo experimento foi realizado para redes reais. Foram utilizadas oito topologias de Sistemas Autônomos (ASes) da Internet, obtidas diretamente do CAIDA [2008] (*Center for Applied Internet Data Analysis*). As topologias possuem tamanho que variam no intervalo $[11, 115]$ e seus respectivos nomes são: 1) Abilene; 2) AS-1221; 3) AS-1239; 4) AS-2914; 5) AS-3257; 6) AS-3356; 7) AS-7018 e 8) Geant. Os resultados obtidos para esse experimento são apresentados na Tabela 2 e no gráfico da Figura 4.

A Tabela 2 compara os resultados obtidos pelos algorimos SMV, SMV-A, BL e GRASP para redes reais. A primeira coluna mostra o nome e a quantidade de vértices de cada uma das redes. A segunda coluna apresenta os resultados extraídos do SMV em quantidade de observadores calculadas pelo algoritmo. A terceira, quarta e quinta colunas apresentam os resultados obtidos pelos algoritmos SMV-A, BL e GRASP, respectivamente. Nestas últimas colunas, as informações são divididas em duas subcolunas, sendo a primeira subcoluna a quantidade de observadores calculada pelos algoritmos e a segunda subcoluna a distância percentual dos algoritmos em relação ao SMV.

Analisando a Tabela 2, percebe-se novamente que todos os algoritmos propostos neste trabalho (SMV-A, BL e GRASP) obtiveram melhores resultados comparados ao SMV, exceto para a rede AS-1221, isto é, os algoritmos conseguiram reduzir a quantidade de observadores necessários para monitorar e gerar a Matriz de Tráfego em quase todas as redes reais avaliadas. Os resultados obtidos são exibidos graficamente na Figura 4. Nesta figura, o eixo horizontal representa o tamanho da rede em quantidade de vértices e o eixo vertical representa a quantidade de observadores calculados pelos algoritmos.

Tabela 2: Comparação dos resultados obtidos pelos algoritmos, SMV, SMV-A, BL e GRASP para as redes reais.

ASes		SMV	SMV-A		BL		GRASP	
Nome	$\|V\|$	Obser.	Obser.	Dist(%)	Obser.	Dist(%)	Obser.	Dist(%)
Abilene	11	7	**4**	**-42,86**	**4**	**-42,86**	**4**	**-42,86**
Geant	22	12	7	-41,67	7	-41,67	**6**	**-50,00**
AS-3257	41	16	**10**	**-37,50**	**10**	**-37,50**	**10**	**-37,50**
AS-1221	44	**6**	**6**	**0,00**	**6**	**0,00**	**6**	**0,00**
AS-1239	52	22	**15**	**-31,82**	**15**	**-31,82**	**15**	**-31,82**
AS-3356	63	20	**13**	**-35,00**	**13**	**-35,00**	**13**	**-35,00**
AS-2914	70	29	23	-20,69	23	-20,69	**21**	**-27,59**
AS-7018	115	21	**15**	**-28,57**	**15**	**-28,57**	**15**	**-28,57**

Com os resultados obtidos no segundo experimento pode-se afirmar que o GRASP proposto funciona também para as redes reais. Além disso, os resultados reforçam a ideia que o fator determinante para a melhora dos resultados foi a utilização dos caminhos baseados em árvore, pricipalmente pelo resultado obtido com a rede AS-1221. Conforme explicado na Seção 3, todos os algoritmos propostos dependem dos Algoritmos 3 e 4. Como a rede AS-1221 é uma árvore, o Algoritmo 3 gera sempre a própria topologia da rede AS-1221, fazendo com que o conjunto de caminhos de entrada para o Algoritmo 4 não varie e, portanto, o resultado obtido pelo algoritmo SMV foi equivalente aos resultados obtidos pelos algoritmos SMV-A, BL e GRASP.

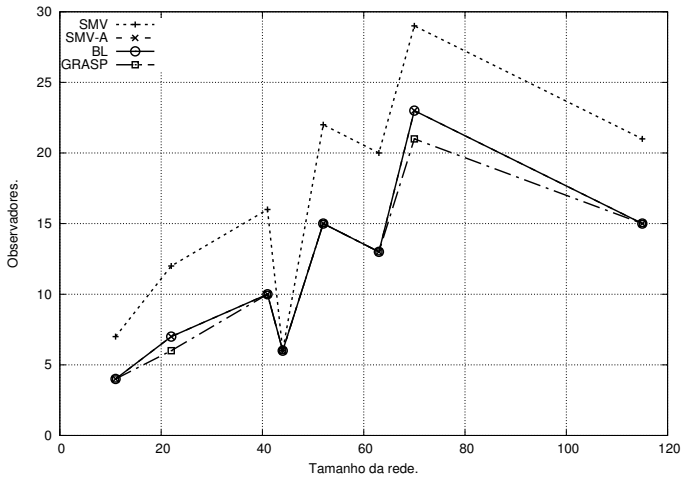

Figura 4: Comparação dos resultados obtidos pelos algoritmos, SMV, SMV-A, BL e GRASP para as redes reais.

5. Conclusões e Trabalhos Futuros

Neste trabalho foi estudado o problema do Conjunto Mínimo de Observadores para geração de Matrizes de Tráfego (CMO-MT). Esse problema pode ser modelado como um problema NP-completo de Traversal Mínima (*MHS - Minimum Hitting Set*).

Para resolver esse problema, foram propostos três algoritmos: um algoritmo para geração de caminhos aleatórios, uma busca local e uma metaheurística GRASP. Todos os algoritmos propostos mais um algoritmo proposto da literatura foram implementados, testados e comparados computacionamente utilizando um conjunto de redes artificias e um conjunto de redes reais de sistemas autônomos da Internet, obtidos no CAIDA.

Com os experimentos realizados, mostrou-se que o GRASP conseguiu obter os melhores resultados (isto é, menores conjuntos de observadores) para as redes artificiais e reais, tendo

melhorado os resultados obtidos pelo algoritmo SMV [Silveira et al., 2016] na maioria dos casos. Além disso, os resultados obtidos permitem concluir que o fator determinante para a melhoria dos resultados foi a utilização de caminhos baseados em árvores, sendo que com a aleatorização destes caminhos o GRASP é capaz de conseguir bons resultados.

Como proposta de trabalhos futuros pretende-se avaliar os resultados do GRASP com a aleatorização do algoritmo SMV, a implementação de outras buscas locais com diferentes vizinhanças e analisar o impacto dos algoritmos em redes reais, com métricas reais.

Agradecimentos

Este trabalho foi parcialmente financiado por: CNPq (processos 461286/2014-9 e 449369/2014-5), FAPES (processo 524/2015), Projeto FUTEBOL (MCTIC/RNP e União Europeia, processo 688941) e ETAURE TI & AUTOMAÇÃO[1].

Referências

Ausiello, G., D'Atri, A., e Protasi, M. (1980). Structure preserving reductions among convex optimization problems. *Journal of Computer and System Sciences*, 21(1):136–153.

Bejerano, Y. e Rastogi, R. (2006). Robust monitoring of link delays and faults in ip networks. *IEEE/ACM Transactions on Networking (TON)*, 14:1092–1103.

Bilal, N., Galinier, P., e Guibault, F. (2013). A New Formulation of the Set Covering Problem for Metaheuristic Approaches. *ISRN Operations Research*, p. 1–10. ISSN 2314-6397.

Breitbart, Y., Dragan, F. F., e Gobjuka, H. (2009). Effective monitor placement in internet networks. *Journal of Networks*, 4:657–666.

CAIDA (2008). Skitter as links dataset. `https://www.caida.org/data/active/skitter_aslinks_dataset.xml`. Acessado em: 2017-06-14.

Cantieni, G. R., Iannaccone, G., Barakat, C., Diot, C., e Thiran, P. (2006). Reformulating the monitor placement problem: Optimal network-wide sampling. In *Proceedings of the 2006 ACM CoNEXT Conference*, p. 5:1–5:12.

Caprara, A., Fischetti, M., e Toth, P. (1999). A heuristic method for the set covering problem. *Operations Research*, 47(5):730–743. URL `http://pubsonline.informs.org/doi/abs/10.1287/opre.47.5.730`.

Chvatal, V. (1979). A greedy heuristic for the set-covering problem. *Mathematics of Operations Research*, 4:233–235.

Dijkstra, E. W. (1959). A note on two problems in connexion with graphs. *Numerische mathematik*, 1(1):269–271.

Feo, T. A. e Resende, M. G. (1989). A probabilistic heuristic for a computationally difficult set covering problem. *Operations research letters*, 8(2):67–71.

Garey, M. R. e Johnson, D. S. (1979). *Computers and Intractability: A Guide to the Theory of NP–Completeness*. W. H. Freeman and Company, San Francisco.

Horton, J. D. e López-Ortiz, A. (2003). On the number of distributed measurement points for network tomography. In *Proceedings of the 3rd ACM SIGCOMM conference on Internet measurement*, p. 204–209. ACM.

[1] http://www.etaure.com.br

Jamin, S., Jin, C., Jin, Y., Raz, D., Shavitt, Y., e Zhang, L. (2000). On the placement of internet instrumentation. In *INFOCOM 2000. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, volume 1, p. 295–304. IEEE.

Martins, S., Pardalos, P., Resende, M., e Ribeiro, C. (1999). Greedy randomized adaptive search procedures for the steiner problem in graphs. In *Proc. of the Randomization Methods in Algorithm Design: DIMACS Workshop*, p. 133–145. American Mathematical Soc.

Moshkovitz, D. (2012). The Projection games conjecture and the NP-hardness of ln n-approximating set-cover. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. ISBN 9783642325113.

Nguyen, H. X. e Thiran, P. (2004). Active measurement for multiple link failures diagnosis in ip networks. In *Passive and Active Network Measurement*, p. 185–194. Springer.

Paul Tune, M. R. (2013). Internet Traffic Matrices. In Haddadi, H. e Bonaventure, O., editors, *Recent Advances in Networking*, chapter 3. ACM SIGCOMM eBook.

Prais, M. e Ribeiro, C. C. (2000). Reactive GRASP: An application to a matrix decomposition problem in TDMA traffic assignment. *INFORMS Journal on Computing*, 12:164–176.

Silveira, F. A. F., Moraes, R. E. N., e Villaça, R. S. (2016). Conjunto mínimo de observadores para geração de matrizes de tráfego. In *Anais do XVLVIII Simpósio Brasileiro de Pesquisa Operacional*, SBPO '48, p. 3208–3219s, Vitória/ES, Brasil. SOBRAPO.

Suh, K., Guo, Y., Kurose, J., e Towsley, D. (2006). Locating network monitors: complexity, heuristics, and coverage. *Computer Communications*, 29:1564–1577.

Zhao, Q. G., Kumar, A., Wang, J., e Xu, J. J. (2005). Data Streaming Algorithms for Accurate and Efficient Measurement of Traffic and Flow Matrices. *SIGMETRICS Perform. Eval. Rev.*, 33(1): 350–361.