Roteamento *Multicast* com Múltiplas Sessões Sujeito a Limite de Orçamento

Romerito Campos
Programa de Pós-graduação em Sistemas e Computação, UFRN
Campus Universitário, Lagoa Nova, Natal, RN
romerito.campos@gmail.com

Marco C. Goldbarg
Departamento de Informática e Matemática Aplicada, UFRN
Campus Universitário, Lagoa Nova, Natal, RN
marcocgold@gmail.com

Elizabeth F. G. Goldbarg
Departamento de Informática e Matemática Aplicada, UFRN
Campus Universitário, Lagoa Nova, Natal, RN
beth@dimap.ufrn.br

RESUMO

Este trabalho apresenta um novo modelo matemático para o problema de roteamento *multicast* com múltiplas sessões. O modelo proposto tem como função objetivo a maximização da capacidade residual, sujeito às restrições de capacidades das arestas, limite de orçamento e requisito de uma árvore de comunicação para cada sessão. Também é proposto um algoritmo genético que possui um passo de refinamento das soluções através da substituição de arestas congestionadas. Experimentos computacionais foram conduzidos em 120 instâncias. São apresentados resultados para o modelo com e sem limite de orçamento. Tais resultados mostram que o modelo proposto foi capaz de gerar boas soluções para o problema de modo consistente.

PALAVRAS CHAVE. Roteamento *multicast* com múltiplas sessões, Modelo matemático, Algoritmo genético.

Área Principal: Outras aplicações em PO

ABSTRACT

This paper presents a new mathematical model to the multi session multicast routing problem. The proposed model has as its objective function the maximization of the residual capacity, subject to constraints due to edges capacities, budget and the requirement of one communication tree to each session. A genetic algorithm with a procedure to refine solutions substituting congested edges is also proposed. Computational experiments were performed on 120 instances. Results are presented to the models with and without budget constraints. Those results showed that the proposed model was able to generate consistently good solutions to the problem.

KEYWORDS. Multi session multicast routing. Mathematical model. Genetic algorithm.

Main Area: Other applications in OR

1. Introdução

Atualmente, vários serviços relacionados à área de telecomunicações são oferecidos. Tais serviços utilizam os recursos da rede de modo simultâneo, levando à concorrência por recursos. Como exemplo, pode-se citar o IPTV, videoconferência, jogos on-line (Wu, 2005), *software delivery* (Han and Shahmehri, 2000), entre outros. Estas aplicações compartilham um interesse comum: uma maneira eficiente de comunicação entre emissor (nó fonte – fornece o serviço) e receptores (conjunto de nós de destino – clientes do serviço). Basicamente, há duas maneiras de estabelecer a comunicação entre emissor e receptores: *unicast* e *multicast*. A comunicação *unicast* consiste em uma forma simples de conectar os participantes da aplicação. No entanto, seu principal problema está relacionado ao envio de cópias de um mesmo pacote usando as mesmas arestas da rede (considerando múltiplos destinos). Isto causa uma sobrecarga de tráfego nas arestas. Por outro lado, o *multicast* supera tal limitação beneficiando-se do compartilhamento de pacotes através das arestas e tem um papel importante na utilização dos recursos da rede. O envio de pacotes dá-se através de arestas compartilhadas entre os nós de destino, evitando-se cópias desnecessárias durante o trajeto. Há várias maneiras de conectar os nós fonte/destino em uma transmissão *multicast*: métodos baseado em Árvores de Steiner (Hwang et al., 1992), *Center-Based trees* e roteamento baseado em anel (Oliveira, 2004). Chen et al. (2000) mostraram que Árvores de Steiner utilizam menos arestas para montar uma árvore de comunicação em comparação à abordagem baseada em anel. *Center-Based trees* adicionam a necessidade de encontrar o centro da rede, o qual deve estar próximo dos nós receptores (tal proximidade deve ter como base alguma métrica – por exemplo, custo). A partir de tais observações, neste trabalho escolheu-se a abordagem baseada em árvores de Steiner para representação das soluções.

O problema de encontrar uma maneira de conectar apenas um grupo *multicast* (considerando grupo *multicast* a união entre nó fonte e nós de destinos) tem sido estudado sob diferentes abordagens. Resende and Pardalos (2006) abordam o problema sob o ponto de vista da otimização de custo. Cui et al. (2003) consideram o problema sob uma perspectiva multiobjetivo - especificamente quatro objetivos: custo da solução; utilização das arestas (razão entre a utilização da aresta e sua capacidade - expressa o congestionamento da aresta); perda de pacotes e *jitter* (diferença entre o *delay* máximo e o *delay* mínimo nos caminhos entre fonte/destinos). Xu (2011) apresenta uma abordagem que considera cinco objetivos: custo, *delay* máximo (nos caninhos entre fonte/destinos); utilização das arestas; *delay* médio da árvore (a soma do *delay* dos caminhos entre fonte/destino pelo número de destinos) e *jitter*.

Quando mais de um grupo *multicast* deve ser configurado simultaneamente na rede, deve-se dar alguma garantia referente à alocação dos recursos necessários. Uma maneira de resolver este problema é projetar a rede para suportar um número finito de grupos *multicast* simultaneamente. A solução é um conjunto de árvores de roteamento *multicast* (uma para cada grupo), este problema é denominado *Multicast* Packing Problem(MPP) (Chen et al., 2000).

O problema tem sido estudado com base em diferentes requisitos. No trabalho de Chen et al. (2000), o problema foi modelado e solucionado considerando a minimização da congestão na rede, o modelo inclui uma penalidade associada ao custo (explicitamente não definido) e uma restrição que limita o crescimento de cada árvore (usada como meio para garantir QoS). Lee and Cho (2004) consideram a maximização da capacidade residual sujeito ao limite de capacidade das arestas e uma restrição que controla o crescimento das árvores como medida de QoS.. Kang et al. (2009) consideram a minimização do custo de construção das árvores *multicast* sujeito a duas restrições: limite da capacidade das arestas e limite no número de *hops* no caminho entre nó fonte e nó de destino como medida de QoS. Chen et al. (2013) abordam o problema de maximizar a capacidade residual da rede onde um grupo *multicast* pode ser atendido por mais de uma fonte (cada fonte cria uma árvore disjunta das outras fontes).

Alguns modelos levam em consideração a otimização do custo da solução, que é um parâmetro interessante para avaliação de solução entre diferentes abordagens (Jia and Wang, 1997;

Low and Wang, 1999; Yan-lin, 2010). Por outro lado, há modelos que consideram a otimização de recursos tal como reserva de banda (por exemplo, capacidade residual ou congestão). Os modelos que tratam de custo geralmente incorporam restrições associadas à capacidade das arestas, embora não haja nenhum controle no que diz respeito à distribuição de carga entre as arestas. Em contrapartida, os modelos que otimizam a capacidade residual não empregam controle sobre o custo da solução.

Neste trabalho, é apresentado um novo modelo para o problema de roteamento *multicast* com múltiplas sessões (MMRBP). O modelo em questão visa maximizar a capacidade residual da rede sujeito a duas restrições: capacidade das arestas e custo. O modelo proposto é resolvido sem e com restrição de orçamento. Experimentos computacionais mostram que o modelo com restrições de orçamento obtém soluções com capacidade residual ótima, na maioria das vezes, com o custo da solução mais baixo. Também, é proposto um algoritmo genético que possui um passo de refinamento das soluções através da substituição de arestas congestionadas.

Este trabalho está dividido em quatro seções além desta. Na seção 2 é apresentado o modelo matemático proposto. A seção 3 apresenta o algoritmo genético criado para o problema. A seção 4 apresenta os dados obtidos dos experimentos computacionais realizados. Por fim, são apresentadas algumas conclusões.

2. Modelo Matemático

O principal objetivo do MMRBP é encontrar uma floresta de árvores *multicast* que maximize a capacidade residual da rede sujeito a limitações de orçamento e capacidade das arestas. A formulação proposta tem como base as definições do *multi-commodity flow problem*, que tem como objetivo a entrega de *commodities* envolvendo diferentes emissores e receptores.

Maximizar: $$Z = min_{(i,j)\in E}\{z_{ij}\} \quad (1)$$

Sujeito a:

$$\sum_{i\in V} x_{ir_k}^{kd} - \sum_{i\in V} x_{r_ki}^{kd} = -1 \quad \forall k \in K, \forall d \in D^k \quad (2)$$

$$\sum_{(i,j)\in E} x_{ij}^{kd} - \sum_{(i,j)\in E} x_{ji}^{kd} = 0 \quad \forall k \in K \text{ e } \forall d \in D^k \quad (3)$$

$$\sum_{\substack{(i,j)\in E \\ j\neq r_k,d}} x_{ij}^{kd} - \sum_{\substack{(i,j)\in E \\ j\neq r_k,d}} x_{ji}^{kd} = 1 \quad \forall k \in K, \forall d \in D^k \quad (4)$$

$$x_{ij}^{kd} \leq y_{ij}^{k} \quad \forall k \in K, d \in D^k \text{ e } \forall (i,j) \in E \quad (5)$$

$$\sum_{d\in D^k} x_{ij}^{kd} - y_{ij}^{k} \leq 0 \quad \forall k \in K \quad (6)$$

$$b_{ij} \geq \sum_{k\in K} y_{ij}^{k} t^k \quad \forall (i,j) \in E \quad (7)$$

$$b_{ij} - \sum_{k\in K} y_{ij}^{k} t^k = z_{ij} \quad \forall (i,j) \in E \quad (8)$$

$$\sum_{(i,j)\in E} y_{ij}^{k} c_{ij} \leq B, \quad \forall k \in K \quad (9)$$

$$x_{ij}^{kd} \in \{0,1\}, y_{ij}^{k} \in \{0,1\} \quad \forall k \in K, \forall d \in D^k, \forall (i,j) \in E \quad (10)$$

As seguintes notações são consideradas: $G = (V, E)$ é um grafo, c_{ij} o custo da aresta $(i, j) \in E$ e b_{ij} a capacidade da aresta $(i, j) \in E$, K é o conjunto das sessões *multicast*, $k \in K$ denota uma sessão *multicast* envolvendo um emissor r_k (fonte) e um grupo de receptores D^k (destinos). Para cada sessão $k \in K$ é considerado uma demanda de tráfego t^k.

As variáveis binárias x_{ij}^{kd} assumem valor 1 quando há fluxo na aresta (i, j) vindo da fonte r_k para o destino $d \in D^k$ e y_{ij}^{k} assumem valor 1 quando (i, j) é aresta da árvore referente à k-ésima sessão *multicast*, $k \in K$. O modelo do MMRBP consiste na otimização da capacidade residual da rede sujeito às restrições de orçamento (custo) e capacidade das arestas. A variável Z corresponde à capacidade residual da rede e deve ser maximizada. O valor de Z é calculado na equação 1.

Os conjuntos de restrições 2, 3 e 4 garantem que haja fluxo passando para cada receptor na sessão *multicast* $k \in K$. Logo, para cada sessão *multicast* haverá uma árvore *multicast* enraizada em r_k. Os conjuntos de restrições 5 e 6 forçam a variável x_{ij}^{kd} a assumir valor 1 quando houver fluxo passando pela aresta (i, j) para algum destino $d \in D^k$. A capacidade b_{ij} de cada aresta $(i, j) \in E$ deve ser respeitada quando da acomodação das árvores *multicast* à rede, conforme o conjunto de restrições 7. O conjunto de equações 8 calcula a capacidade residual de cada aresta denotada por z_{ij}.

A solução do problema proposto tem um fator importante que limita a maximização de Z. Tendo em vista que cada aresta (i, j) tem um custo de utilização, a construção da solução está sujeita ao orçamento B, conforme o conjunto de restrições 9.

3. Algoritmo Genético

Nesta seção é apresentado um algoritmo genético para o problema definido na seção 2. São apresentados os algoritmos referentes à criação, recombinação e mutação. Além disso, é apresentado um algoritmo de refinamento das soluções.

3.1. Representação

O cromossomo definido consiste do conjunto de todos os caminhos entre uma fonte e um nó destino. Cada posição do cromossomo, isto é cada gene, representa um caminho entre a fonte r_k e um destino $d \in D^k$. A figura 1 ilustra uma parte de um cromossomo. Supondo $k = 1$ e $D^1 = \{10, 5, 15\}$, a figura 1 mostra na primeira posição do cromossomo o caminho entre o terminal 15 e a fonte $r_k = 1$. Após a representação dos caminhos dos terminais de $k = 1$, o cromossomo contém os caminhos entre a fonte e os terminais de $k = 2$, e daí por diante. Portanto, o número de genes em um cromossomo é dado por $SIZE = \sum_{k \in K} |D^k|$.

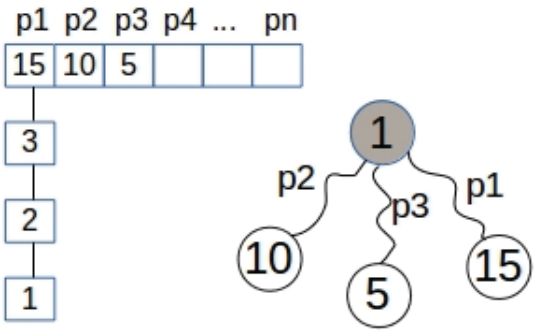

Figura 1: Representação de uma solução.

3.2. Construção de uma Solução

O processo de criação de uma solução leva em consideração o uso de caminhos mais curtos. Tais caminhos são gerados com o algoritmo de Dijkstra (Ahuja et al., 1993; Dijkstra, 1959) para cada par (r_k,$d \in D^k$) de cada grupo *multicast* $k \in K$. O algoritmo 1 cria uma solução. Ele inicia definindo um novo indivíduo, $\_novo\_$. Os caminhos entre uma fonte r_k e um destino $d \in D^k$ são construídos no laço entre as linhas 2-5. A função $shortest\_path$ na linha 3 recebe dois argumentos: r_k e $dest(i)$. O valor de r_k é deduzido pelo valor de i, assim como o destino $d \in D^k$ apropriado em relação ao gene i. Estruturas de dados auxiliares são utilizadas para garantir que uma árvore está sendo montada a partir de caminhos da raiz r_k para um destino $d \in D^k$.

Algoritmo 1 Algoritmo de Construção de Solução.
Entrada: Grafo G, K, $SIZE$
Saída: Indivíduo _novo_
1: $Individuo \quad \_novo\_$;
2: **para** $i \leftarrow 0$ **até** $SIZE - 1$ **faça**
3: $\quad Path \quad p \leftarrow shortest\_path(r_k, dest(i))$
4: $\quad \_novo\_[i] \leftarrow p$
5: **fim para**

3.3. Seleção e Recombinação

Operadores de recombinação permitem que a população de indivíduos seja melhorada gradativamente a cada iteração. Isto se dá pela combinação de genes de indivíduos da população para gerar novos indivíduos. Para tanto, deve-se considerar o processo de seleção para reprodução. Neste trabalho, optou-se por utilizar seleção por torneio binário (Goldberg, 1989). São selecionados quatro indivíduos aleatoriamente com equiprobabilidade. Os dois melhores indivíduos são recombinados gerando um terceiro conforme mostrado no algoritmo 2. O critério de adequação dos melhores indivíduos é definido pela capacidade residual. Quanto maior a capacidade residual (função Z), melhor é o indivíduo.

Algoritmo 2 Recombinação
Entrada: Indivíduo P1, Indivíduo P2, K, $SIZE$
Saída: Indivíduo _novo_
1: $Individuo \quad \_novo\_$;
2: $point \leftarrow rand(SIZE)$
3: **para** $i \leftarrow 0$ **até** $point$ **faça**
4: $\quad \_novo\_[i] \leftarrow P1[i]$
5: **fim para**
6: **para** $i \leftarrow point + 1$ **até** $SIZE - 1$ **faça**
7: $\quad \_novo\_[i] \leftarrow P2[i]$
8: **fim para**

O procedimento recebe como entrada dois indivíduos, $P1$ e $P2$, o conjunto de grupos multicast K e o tamanho do indivíduo, $SIZE$. Ao final do procedimento, um novo indivíduo será criado com base nos genes de $P1$ e $P2$. Inicialmente, define-se um ponto de corte (linha 1). Os laços são utilizados para percorrer todos os genes e atribuí-los ao novo indivíduo (linhas 3-8), o qual conterá genes(caminhos) de ambos os pais, $P1$ e $P2$.

3.4. Mutação

O operador de mutação permite que o conjunto de indivíduos da população seja diversificado. Este operador é útil para evitar que a população convirja prematuramente para soluções viáveis com baixa qualidade.

O algoritmo 3 descreve o operador de mutação. Ele recebe quatro parâmetros: o indivíduo $P1$ que sofrerá mutação, o grafo G, o parâmetro $LIST\_SIZE$ ($0 < LIST\_SIZE < 1$), que representa um valor percentual utilizado para calcular o tamanho da lista de arestas do indivíduo $P1$ que será utilizada na mutação e o tamanho do indivíduo, $SIZE$. O algoritmo inicia obtendo uma lista de arestas do indívudo $P1$(linha 1). Esta lista contém todas as arestas do indivíduo $P1$ e é ordenada pelo valor da capacidade residual das arestas - ordem crescente. A variável $size$ armazena o número de arestas de $P1$ (linha 2) e uma lista vazia de arestas é criada na linha 3. O próximo passo é marcar algumas arestas utilizadas por $P1$ como não disponíveis. Para tanto, o laço das linhas 4-7 é utilizado. Neste laço, a lista de arestas de $P1$ é percorrida da aresta mais congestionada

Algoritmo 3 Mutação

Entrada: Indivíduo P1, Grafo G, $LIST\_SIZE$, $SIZE$
Saída: Indíviduo _novo

1: $List \leftarrow P1.get\_used\_list()$ ▷ ordem crescente por capacidade residual
2: $size \leftarrow List.size()$
3: $rem\_edges \leftarrow \emptyset$
4: **para** $i \leftarrow 0$ **até** $size * LIST\_SIZE$ **faça**
5: $G.remove(List[i])$
6: $rem\_edges \leftarrow rem\_edges \cup List[i]$
7: **fim para**
8: **para** $i = 0$ **até** $SIZE - 1$ **faça**
9: **se** $test(P1[i], rem\_edges)$ **então** $\_novo[i] \leftarrow shortest\_path(P1[i].source, P1[i].dest)$
10: **senão**$\_novo[i] \leftarrow P1[i]$
11: **fim se**
12: **fim para**

(posição 0) até a posição $size * LIST\_SIZE$. Em cada iteração uma aresta é removida do grafo G(linha 5) e adicionada à lista de arestas rem_edges (linha 6).

O processo de recriação do indivíduo $P1$ é iniciado (laço das linhas 8-12). Todos os genes do indivíduo $P1$ são verificados (os caminhos de $(r_k, d \in D^k)$ para $k \in K$). Em cada caminho é realizado um teste (linha 9), o qual verifica se o caminho possui alguma aresta marcada como removida. Caso o teste seja verdadeiro, um novo caminho é construido entre r_k e $d \in D^k$ referente ao gene i. Observa-se que o algoritmo mantém o controle sobre o inicío e o fim de cada árvore no cromossomo, de modo que o caminho seja computado corretamente. Por outro lado, se o caminho referente ao gene i não sofreu alteração, então o mesmo é copiado para o novo indivíduo, como mostrado na linha 10.

3.5. Operador de Refinamento

As operações de recombinação e mutação permitem que o espaço de busca seja bastante explorado. No entanto, devido à complexidade do problema em questão, uma operação de refinamento da solução é aplicada. Este algoritmo é semelhante ao algoritmo proposto por Lee and Cho (2004).

O procedimento de refinamento proposto tem como base a ideia de remoção de arestas muito congestionadas. Porém, não se constrói caminhos alternativos que possam conectar as duas árvores provenientes da remoção de uma aresta como em Lee and Cho (2004). Ao contrário, quando uma aresta é removida, obtém-se um conjunto de arestas que possuem um vértice na árvore T_1^k e outro vértice na árvore T_2^k. Este procedimento possui a mesma complexidade que informada por Lee and Cho (2004), porém permite substituir arestas sem aumentar exageradamente o valor de custo da solução, que poderia violar a restrição de orçamento.

O algoritmo 4 mostra o pseudocódigo do operador de refinamento. Ele inicia com a criação da lista $Edges$. Esta lista contém as arestas utilizadas na criação do indivíduo $P1$. Cada aresta $e \in Edges$ possui o valor de capacidade residual disponível. Em seguida, todas as arestas de $Edges$ são verificadas (linhas 2-10). Todas as árvores do indivíduo $P1$ são verificadas no laço entre as linhas 3-9 . Um teste é feito considerando uma árvore t e uma aresta e. Caso seja verificado que a aresta e está na árvore t (linha 4), então uma lista de arestas que podem substituir e é criada - definida como $list$ (linha 5). Tal lista é criada considerando as arestas que conectam as duas árvores resultantes da remoção da aresta e da árvore t. Portanto, toda aresta que reconectar as árvores resultantes da divisão de t e que possuir capacidade residual superior à mínima capacidade residual da solução representada por P1 serão consideradas para compor a lista $list$. Uma aresta $e^{'} \in list$ é escolhida aleatoriamente para substituir e e o indivíduo $P1$ é atualizado.

Algoritmo 4 Refinamento

Entrada: Indivíduo P1
Saída: Indivíduo _novo_
1: $Edges \leftarrow P1.getEdges()$
2: **para** $e \in Edges$ **faça**
3: **para** $Treet \in P1.getTrees()$ **faça**
4: **se** $t.contains(e)$ **então**
5: $List \quad list \leftarrow getAvailableEdges(t, e)$
6: $Edge \quad e^{'} \leftarrow rand(list)$
7: $P1.update(e^{'})$
8: **fim se**
9: **fim para**
10: **fim para**

Refinamento de Custo

A mesma ideia aplicada para melhorar a capacidade residual de um indivíduo $P1$ ilustrado no algoritmo 4 pode ser utilizada para redução de custo de um indivíduo. Para tanto, basta substituir a chamada da linha 1 do algoritmo 4 por uma chamada que retorne as arestas ordenadas pelo custo - ordenação decrescente. Além disso, a chamada da linha 5 do algoritmo 4 deve considerar as arestas com custo menor que e e que após a utilização fiquem com capacidade residual pelo menos igual a da aresta e.

3.6. Pseudocódigo

O algoritmo 5 mostra o pseudocódigo do algoritmo genético proposto.

Algoritmo 5 Algoritmo Genético proposto para o MMRBP.

Entrada: $cross, iter, mut, ref, pop, LIST\_SIZE$
Saída: Indivíduo $best$
1: $populacao \quad P$
2: $Individuo \quad best \leftarrow best(P)$
3: $inicializacao(P, pop)$ ▷ Algoritmo 1
4: **para** $i \leftarrow 0$ ate $iter - 1$ **faça**
5: $recombinacao(P, cross)$ ▷ Algoritmo 2
6: $mutacao(P, mut, LIST\_SIZE)$ ▷ Algoritmo 3
7: $refinamento(P, ref)$ ▷ Algoritmo 4
8: $best \leftarrow best(P)$
9: **fim para**

O algoritmo 5 inicia definindo uma população P(linha 1). Em seguida, define-se um indivíduo $best$ (linha 2). Este indivíduo manterá a melhor solução durante o processo de busca. Após as inicializações necessárias, tem início o laço (linhas 4-8) que realiza o processo de evolução da população P. Este laço é executado $iter$ vezes(parâmetro passado ao algoritmo). Os operadores de recombinação, mutação e refinamento (capacidade residual e custo) são aplicados à população. Os parâmetros considerados são: $cross$ - taxa de recombinação; mut - taxa de mutação; $LIST\_SIZE$ - parâmetro necessário ao operador de mutação; ref - taxa de aplicação do operador de refinamento. No fim de cada iteração a melhor solução é atualizada.

4. Experimentos Computacionais

Os testes foram realizados utilizando o sistema operacional Ubuntu 15.04 com processador Intel(R) Core(TM) i7-2600K CPU 3.40GHz. Os algoritmos foram implementados utilizando

a linguagem de programação C++ e compilados utilizado a *flag* -O3. Além disso, utilizou-se a biblioteca *heap* do conjunto de bibliotecas *boost* versão 1.54.0 (Polukhin, 2013; Siek et al., 2001).

Um conjunto de 120 instâncias foram utilizadas para realização de experimentos. Elas foram criadas utilizando o framework BRITE (Medina et al., 2001), o qual utiliza o modelo de Waxman (1988). A tabela 1 ilustra os detalhes das instâncias.

Tabela 1: Instâncias

Vértices(V)	Grupos	Vértice/Grupo	Nº Arestas(E)
30	5,10,15,20,25	20-30 % V	2V
60	5,10,15,20,25	15-30 % V	2V
120	5,10,15,20,25	10-30 % V	2V
240	5,10,15,20,25	10-20 % V	2V

A tabela 1 ilustra a composição do conjunto de instâncias. A coluna Vértices ilustra o número de vértices de cada instâncias. A coluna Grupos ilustra o número de grupos *multicast* por instâncias. A coluna Vértice/Grupo ilustra o número de membros por grupo e é definido com base em um percentual do número de nós da rede (V). Por fim, a coluna Nº Arestas(E) ilustra o número de arestas que a rede possui.

Além dos grupos de instâncias definidos pelo tamanho em número de vértices. Cada um dos conjuntos é dividido em conjuntos menores com base no número de grupos. O conjunto de instâncias de tamanho V=30 possui 5 grupos de instâncias com base no número de grupos *multicast*. Cada um destes subconjuntos possui 6 instâncias. Por exemplo, o conjunto de instâncias de tamanho V=30 possui 6 instâncias com 5 grupos *multicast* cada uma, 6 instâncias com 10 grupos *multicast* cada uma e daí por diante. A mesma divisão aplica-se às instânicas com $V = \{60, 120, 240\}$. As instâncias são identificadas por um identificador e o número de grupos que ela possui. Por exemplo, no conjunto com $V = 30$ a primeira instância é a 30_1_5: o valor 30 representa o número de vértices, o valor 1 é o seu identificador e o valor 5 representa o número de grupos *multicast*.

Para os experimentos relatados neste trabalho, o valor de capacidade das arestas é definido como sendo o número de grupos no caso de teste. Por exemplo, um caso de teste com 5 grupos tem as arestas da rede com capacidade $b_{ij} = 5$. Além disso, cada grupo *multicast* tem uma demanda $t^k = 1$. Logo, sempre que uma aresta (i, j) for utilizada por uma árvore, terá sua capacidade diminuída de 1.

O afinamento dos parâmetros do algoritmo genético foi realizado mediante a utilização do procedimento *Iterated Racing* implementado no pacote IRACE (López-Ibáñez et al., 2011). A implementação permite obter uma configuração automática para os parâmetros. Como resultado da utilização do IRACE, os parâmetros foram definidos como: $pop = 24$, $cross = 0,65$, $mut = 0,1$, $iter = 25$, $list = 0,13$ e $ref = 0,3$.

4.1. Resultados

Nesta seção são apresentados resultados da solução do modelo com o solver Gurobi (Gurobi Optimization, 2015) e com o algoritmo genético proposto. Foram resolvidos dois modelos com o Gurobi: modelo sem a restrição de orçamento e com a restrição de orçamento. Os resultados do algoritmo genético foram obtidos de 50 execuções independentes por caso teste. O valor relatado é a mediana do valor objetivo (capacidade residual) e a mediana do tempo computacional. As tabelas 2 e 3 apresentam os resultados. A primeira coluna de cada uma das tabelas indica o tamanho da instância $V = \{30, 60, 120, 240\}$. As colunas *Opt* indicam o valor objetivo obtido pelo método correspondente, sendo tal valor a capacidade residual da rede. As colunas *Custo* mostram o custo da solução. No modelo com limite, considerou-se o orçamento restrito a 80% do valor do custo da solução do modelo sem restrição. As colunas *T* indicam os tempos de processamento, em segundos, dos métodos correspondentes. Em relação ao algoritmo genético, as colunas *Z* e *T*

mostram as medianas do valor da função objetivo para 50 execuções do e dos tempos de execução, respectivamente.

	Modelo Sem Limite		Modelo com Limite-B			Algoritmo Genético		
V=30	Opt	Custo	Opt	Custo	T(s)	Z	Custo	T(s)
30_1_5	3	21081	3	16865	0,69	3	16069	0,16
30_2_5	3	24412	3	19530	0,37	3	16566	0,20
30_3_5	3	23051	3	18441	0.79	3	16495	0,24
30_4_5	3	24869	3	19895	0,50	2	18670,5	0,25
30_5_5	3	25834	3	20667	0,31	2	17465,5	0,21
30_6_5	2	31327	2	25062	0,66	2	22047	0,27
30_7_10	7	45437	7	36350	2,69	6	35738	0,84
30_8_10	7	52358	7	41886	1,82	6	41508	1,04
30_9_10	7	54618	7	43694	1,58	6	39675	0,67
30_10_10	8	36192	7	28954	1,56	6	28225,5	0,45
30_11_10	8	44872	7	35898	3,13	6	35414	0,82
30_12_10	7	54834	7	43867	4,04	7	42037	1,13
30_13_15	11	77912	11	62330	3,30	10	60224,5	1,87
30_14_15	12	70259	10	56207	3,05	10	56050	1,06
30_15_15	11	73710	11	58968	4,15	10	52243	1,55
30_16_15	11	76859	11	61487	11,96	10	60962,5	2,24
30_17_15	11	83240	11	66592	2,68	10	58621,5	2,34
30_18_15	12	67857	11	54286	6,94	10	54095,5	1,52
30_19_20	14	114284	14	91427	4,40	14	88850	3,00
30_20_20	14	117613	14	94090	4,56	13	88968	2,56
30_21_20	15	102702	15	82162	7,50	14	80539	2,64
30_22_20	16	100562	15	80450	8,98	14	79859	3,21
30_23_20	16	82086	16	65669	6,95	15	63858,5	2,09
30_24_20	15	97198	15	77758	9,56	14	73899	2,85
30_25_25	20	106649	18	85319	7,33	17	84021,5	3,17
30_26_25	20	103482	19	82786	17,00	18	81003	4,70
30_27_25	19	136974	18	109579	21,84	17	109332,5	4,46
30_28_25	19	125196	18	100157	10,33	18	100068	4,33
30_29_25	19	119507	19	95606	9,25	17	95161	3,12
30_30_25	17	125162	17	100130	8,43	16	94894	4,20

	Modelo Sem Limite		Modelo com Limite-B			Algoritmo Genético		
V=60	Opt	Custo	Opt	Custo	T(s)	Z	Custo	T(s)
60_1_5	3	44444	3	35555,2	2,12	3	28962,5	0,46
60_2_5	4	29802	4	23841,6	1,32	3	21036	0,22
60_3_5	4	41826	3	33460,8	23,63	3	30263,5	0,71
60_4_5	3	38051	3	30440,8	6,95	3	29219	0,60
60_5_5	3	46246	3	36996,8	2,22	1	29695	0,41
60_6_5	3	41745	3	33396	6,06	2	28763	0,72
60_7_10	7	89774	7	71819,2	55,87	6	67580	3,01
60_8_10	7	98862	7	79089,6	8,48	6	60300,5	3,23
60_9_10	7	80436	7	64348,8	11,58	6	60389	1,66
60_10_10	7	78211	7	62568,8	6,20	6	52915	1,70
60_11_10	7	73133	7	58506,4	5,45	6	55016	1,11
60_12_10	7	85753	7	68602,4	19,32	6	64610	2,11
60_13_15	12	108559	12	86847,2	130,89	10	85115,5	2,96
60_14_15	11	132437	11	105949,6	17,73	10	105512	5,23
60_15_15	12	110902	12	88721,6	136,62	10	88496,5	4,80
60_16_15	12	123929	12	99143,2	32,93	10	88530,5	3,74
60_17_15	11	105230	11	84184	12,96	9	77750	3,82
60_18_15	11	113084	11	90467,2	38,57	10	89219,5	4,46
60_19_20	16	150439	16	120351,2	156,77	15	118695	5,56
60_20_20	15	174909	15	139927,2	103,73	13	132460,5	8,60
60_21_20	15	167379	15	133903,2	85,99	14	131542,5	9,73
60_22_20	16	142865	16	114292	211,51	14	104450	7,28
60_23_20	16	140864	16	112691,2	89,60	15	109700	7,04
60_24_20	15	159375	15	127500	79,62	14	119272	10,62
60_25_25	20	188566	20	150852,8	137,98	18	142010	12,19
60_26_25	20	203994	20	163195,2	156,38	17	154561	15,64
60_27_25	20	202395	20	161916	95,24	18	154935	12,05
60_28_25	20	209046	20	167236,8	155,96	17	166158	11,94
60_29_25	20	209735	20	167788	253,19	18	164355	11,35
60_30_25	20	182720	20	145097	86,53	19	143907	10,13

Tabela 2: Resultados para instâncias de tamanho V=30 e V=60.

Comparando a resolução do modelo com e sem restrição de orçamento, as tabelas mostram que uma redução de 20% no custo (passado como restrição de orçamento) não acarreta perda considerável no valor da capacidade residual da rede, sendo a perda média de 3%, 0,3%, 0% e 0% nas instâncias com 30, 60, 120 e 240 vértices, respectivamente. É interessante ressaltar que, no conjunto de instâncias do experimento, a perda na capacidade residual decresce com o aumento do tamanho da instância.

O algoritmo genético apresenta um ganho expressivo em termos de custo. Quando comparado ao modelo com restrição de orçamento, a melhoria no custo é, em média, 2,8%, 4,4%, 6,3% e 9,1% nas instâncias com 30, 60, 120 e 240 vértices, respectivamente. Aqui, também, é interessante ressaltar que o custo da solução melhora com o aumento do tamanho das instâncias. Por outro lado as perdas médias em relação à capacidade residual são de 7,1%, 12,4%, 14,9% e 14,8%, respectivamente, nas instâncias com 30, 60, 120 e 240 vértices. Pode-se observar que enquanto os ganhos em relação a custo entre as menores e as maiores instâncias é de um fator aproximado de 3, a perda em relação à capacidade residual é de um fator aproximado de 2. Os tempos de processamento do algoritmo genético são, em média 2,9 vezes menores que os do *solver* para instâncias com 30 vértices. A razão de melhoria do tempo de processamento para as instâncias com 60, 120 e 240 vértices são de 13, 74 e 148 vezes.

5. Conclusão

Neste trabalho foram apresentados um novo modelo no contexto de roteamento *multicast* envolvendo múltiplas sessões, assim como um algoritmo genético para resolvê-lo. Também foram apresentados os resultados ótimos obtidos através da resolução do modelo por um *solver*. Foram consideradas situações com e sem limite de orçamento.

O modelo, ao considerar uma restrição de orçamento adicional aos modelos da literatura, permitiu que soluções significativamente mais econômicas (20% menores) fossem alcançadas com

	Modelo Sem Limite		Modelo com Limite-B			Algoritmo Genético		
V=120	Opt	Custo	Opt	Custo	T(s)	Z	Custo	T(s)
120_1_5	4	65944	4	52755,2	372,73	3	42572	1,91
120_2_5	3	73108	3	58486,4	23,20	3	52151	2,82
120_3_5	3	67368	3	53894,4	51,02	2	45740	2,22
120_4_5	3	60273	3	48218,4	26,57	2	42844,5	1,68
120_5_5	3	79061	3	63248,8	62,65	3	48997	2,13
120_6_5	4	50416	4	40332,8	72,46	3	39196	1,10
120_7_10	8	137375	8	109900	464,69	6	99466	8,05
120_8_10	7	120694	7	96555,2	213,61	6	83898	7,03
120_9_10	8	122811	8	98248,8	256,55	7	96149	6,96
120_10_10	8	131708	8	105366,4	243,38	6	94212,5	7,66
120_11_10	8	122316	8	97852,8	681,64	7	84061,5	5,98
120_12_10	7	124818	7	99854,4	75,72	7	85413	6,37
120_13_15	11	197480	11	157984	222,93	10	145233	17,01
120_14_15	12	182750	12	146200	1512,16	10	132282	16,11
120_15_15	12	196204	12	156963,2	580,79	11	142023	13,60
120_16_15	12	183424	12	146739,2	1560,96	10	136155	15,44
120_17_15	12	188018	12	150414,4	1122,65	10	138060,5	19,95
120_18_15	12	201234	12	160987,2	1307,37	10	156287,5	20,33
120_19_20	16	274885	16	219908	400,65	15	204621,5	34,03
120_20_20	16	275088	16	220070,4	1018,06	14	212463,5	39,63
120_21_20	16	270065	16	216052	2321,50	12	198924	31,72
120_22_20	16	262614	16	210091,2	4614,09	13,5	201466	43,47
120_23_20	16	271677	16	217341,6	1386,53	14	200384,5	32,95
120_24_20	17	245326	17	196260,8	16516,32	13,5	194540	27,82
120_25_25	20	346827	20	277461,6	1541,86	16,5	248842	66,30
120_26_25	21	300480	21	240384	4970,35	18	231388	39,058
120_27_25	20	323297	20	258637,6	1064,78	18	254306	50,55
120_28_25	21	328731	21	262984,8	4613,80	17	248082	47,16
120_29_25	20	326828	20	261462,4	492,88	18	238551,5	53,21
120_30_25	21	297924	21	238339,2	1977,19	18	232003,5	44,82

	Modelo Sem Limite		Modelo com Limite-B			Algoritmo Genético		
V=240	Opt	Custo	Opt	Custo	T(s)	Z	Custo	T(s)
240_1_5	4	74116	4	59293	566,12	3	54009,5	3,30
240_2_5	4	80270	4	64216	158,42	3	53209	2,45
240_3_5	4	79645	4	63716	229,21	3	62115	2,39
240_4_5	3	87426	3	69941	47,53	3	59274,5	4,20
240_5_5	4	83240	4	66592	137,63	3	53130	2,90
240_6_5	4	83915	4	67132	199,37	3	52924	2,53
240_7_10	8	164705	8	131764	496,54	7	106883,5	8,65
240_8_10	8	158398	8	126718	687,79	7	113321,5	9,53
240_9_10	8	172206	8	137765	1588,69	7	117819	12,60
240_10_10	8	171843	8	137474	2811,19	6	118442,5	13,83
240_11_10	8	163357	8	130686	3744,81	7	118580	12,62
240_12_10	7	177406	7	141925	252,71	6	125381,5	13,77
240_13_15	13	220747	13	176598	1274,63	11	164653	20,94
240_14_15	13	238206	13	190565	934,17	11	164821	20,60
240_15_15	13	231363	13	185090	9881,42	11	168042,5	21,09
240_16_15	13	227947	13	182358	4249,81	11	165174	17,52
240_17_15	13	233410	13	186728	5102,05	10	159097,5	22,22
240_18_15	13	219668	13	175734	4876,45	11	162627,5	25,10
240_19_20	17	310107	17	248086	1223,00	14	211989	40,69
240_20_20	17	287730	17	230184	2223,48	14	198157,5	29,95
240_21_20	17	315071	17	252057	897,88	15	222512	36,80
240_22_20	17	320122	17	256098	3946,29	14	230750,5	45,40
240_23_20	17	313767	17	251014	1499,74	15	236073,5	47,11
240_24_20	17	337920	17	270336	1661,71	14	240963	38,56
240_25_25	21	403140	21	322512	2346,13	18	292243,5	77,63
240_26_25	22	400258	22	320206	11841,16	19	317675,5	65,42
240_27_25	20	372828	20	298262	482,81	18	249232	58,36
240_28_25	21	412345	21	329876	2586,82	19	302051	72,23
240_29_25	20	440304	20	352243	746,38	18	332910,5	86,25
240_30_25	22	385065	22	308052	65378,42	19	287651	77,51

Tabela 3: Resultados para instâncias de tamanho V=120 e V=240.

uma degradação de congestão pouco significativa (em média 3%, 0,3%, 0% e 0% nas instâncias com 30, 60, 120 e 240 vértices, respectivamente).

O algoritmo genético comprovou um caminho de solução viável para problemas de grande porte em que o tempo de processamento do algoritmo exato se mostre proibitivo. A eficiência do algoritmo genético também se comprovou aceitável na medida em que seu afastamento médio da solução ótima exibe pequena degradação adicional no valor da congestão, todavia adicionalmente compensada por um efeito de redução ainda maior no custo final da rede. Soluções de maior degradação no valor da congestão alcançada pelo algoritmo evolucionário se mostram sempre de menor custo, o que não necessariamente é garantido para soluções viáveis desse problema.

O experimento sugere que é possível e significativo o estudo do *trade off* entre a minimização do valor da congestão e a minimização do investimento necessário à realização da tarefa de comunicação desejada. Pode ser possível encontrar soluções de menor custo que não reduzam a congestão da rede ou a reduzam em valores pequenos. Sabendo-se que o citado *trade off* pode ser significativamente promissor, sugere-se que modelos futuros examinem o formato multiobjetivo desse problema.

6. Agradecimentos

A pesquisa foi parcialmente suportada pela CAPES, através de uma bolsa para o primeiro autor e pelo CNPq, projetos 301845/2013-1 e 308062/2014-0.

Referências

Ahuja, R., Magnanti, T., and Orlin, J. (1993). *Network flows: theory, algorithms, and applications*. Prentice Hall.

Chen, S., Gunluk, O., and Yener, B. (2000). The multicast packing problem. *IEEE/ACM Transactions on Networking*, 8:311–318.

Chen, Y.-R., Radhakrishnan, S., Dhall, S., and Karabuk, S. (2013). On multi-stream multi-source multicast routing. *Computer Networks*, 57(15):2916 – 2930.

Cui, X., Lin, C., and Wei, Y. (2003). A multiobjective model for qos multicast routing based on genetic algorithm. In *Proceedings of the 2003 International Conference on Computer Networks and Mobile Computing*, ICCNMC '03, pages 49–, Washington, DC, USA. IEEE Computer Society.

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

Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1st edition.

Gurobi Optimization, I. (2015). Gurobi optimizer reference manual.

Han, L. and Shahmehri, N. (2000). Secure multicast software delivery. In *Enabling Technologies: Infrastructure for Collaborative Enterprises, 2000. (WET ICE 2000). Proeedings. IEEE 9th International Workshops on*, pages 207 –212.

Hwang, F., Richards, D. S., and Winter, P. (1992). *Steiner Tree Problem*. Elsevier Science Publishers.

Jia, X. and Wang, L. (1997). A group multicast routing algorithm by using multiple minimum steiner trees. *Computer Communications*, 20(9):750 – 758.

Kang, J., Park, K., and Park, S. (2009). Optimal multicast route packing. *European Journal of Operational Research*, 196:351–359.

Lee, C. Y. and Cho, H. K. (2004). Multiple multicast tree allocation in ip network. *Comput. Oper. Res.*, 31:1115–1133.

López-Ibáñez, M., Dubois-Lacoste, J., Stützle, T., and Birattari, M. (2011). The irace package, iterated race for automatic algorithm configuration. Technical Report TR/IRIDIA/2011-004, IRIDIA, Université Libre de Bruxelles, Belgium.

Low, C. P. and Wang, N. (1999). An efficient algorithm for group multicast routing with bandwidth reservation. In *Proceedings of the 7th IEEE International Conference on Networks*, ICON '99, pages 43–, Washington, DC, USA. IEEE Computer Society.

Medina, A., Lakhina, A., Matta, I., and Byers, J. (2001). Brite: Universal topology generation from a user"s perspective. Technical report, Boston, MA, USA.

Oliveira, C. A. S. (2004). *Optimization problems in telecommunications and the internet*. PhD thesis, Gainesville, FL, USA. AAI3146236.

Polukhin, A. (2013). *Boost C++ Application Development Cookbook*. Packt Publishing.

Resende, M. G. C. and Pardalos, M. P. (2006). *Handbook of Optimization in Telecommunications*. Springer, 1 edition.

Siek, J., Lee, L., and Lumsdaine, A. (2001). *Boost Graph Library: User Guide and Reference Manual, The*. Pearson Education.

Waxman, B. M. (1988). Routing of multipoint connections. *IEEE Journal of Selected Areas in Communications*, 6(9):1617–1622.

Wu, Z. (2005). Performance modeling of multicast groups for multiplayer games in peer-to-peer networks. In *Distributed Simulation and Real-Time Applications, 2005. DS-RT 2005 Proceedings. Ninth IEEE International Symposium on*, pages 105 – 112.

Xu, Y. (2011). *Metaheuristic Approaches for QoS Multicast Routing Problems*. PhD thesis, University of Nottingham.

Yan-lin, W. (2010). Based qos constrained group multicast routing for multimedia communication. In *Computer, Mechatronics, Control and Electronic Engineering (CMCE), 2010 International Conference on*, volume 6, pages 296 –299.