Dual-Ascent Distribuído para Multicast com Restrição de Saltos

Marcelo Santos[1], **Lúcia M. A. Drummond**[1], **Eduardo Uchoa**[1], **Luidi Simonetti**[2]

[1]Universidade Federal Fluminense

[2]Universidade Federal do Rio de Janeiro

{mpinto,lucia }@ic.uff.br, uchoa@producao.uff.br

Abstract. *The directed Steiner Problem consists in finding a minimum cost tree that reaches a subset of nodes of a graph from a root node r. It is frequently used to model the multicast routing problem. The number of hops between two nodes of a network can be used as an approximation for the communication latency between the nodes. So we have practical interest in the construction of a tree with limited height. In this case, we call the problem "Steiner Problem with Hop Constraint". This paper introduces a distributed heuristic to the Steiner Problem with Hop Constraint based on the algorithm "Dual Ascent" proposed by Wong [Wong 1984] and in the layered graph introduced by Gouveia and Uchoa [Gouveia et al. 2007]. We show experimental results where Dual Ascent gives better results than the best known algorithm for the problem.*

Resumo. *O Problema de Steiner direcionado consiste em minimizar o custo de uma arborescência que atinja um subconjunto de nós de um grafo a partir de um nó raiz r. É freqüentemente utilizado para modelar o problema de roteamento multicast. O número de saltos entre dois nós de uma rede pode ser utilizado como uma aproximação para a latência de comunicação entre estes nós, portanto, é de interesse prático a construção de uma arborescência com altura limitada. Neste caso, temos o Problema de Steiner com Restrição de Saltos. Este trabalho apresenta uma heurística distribuída para o problema com restrição de saltos, baseada no algoritmo "Dual Ascent" proposto por Wong [Wong 1984] e no grafo em camadas introduzido por Gouveia e Uchoa [Gouveia et al. 2007]. São exibidos resultados experimentais onde o Dual Ascent apresenta melhores soluções que o melhor algoritmo conhecido para o problema.*

1. Introdução

Várias aplicações atuais, como por exemplo teleconferência e distribuição de vídeo sob demanda, requerem a distribuição de grandes quantidades de dados a um subconjunto de nós de uma rede, o que é denominado de "broadcast seletivo" ou "multicast". Este problema é freqüentemente modelado como o Problema de Steiner em Grafos Direcionados (*Steiner Problem in Directed Graphs- SPDG*)[Novak et al. 2001, Oliveira e Pardalos 2005].

A variação do problema de Steiner onde devemos, além de minimizar o custo da solução, manter uma segunda métrica abaixo de um determinado limite, chamada "Problema de Steiner com Restrições", é de grande interesse prático, pois a distribuição de

multimídia em "multicast" freqüentemente é melhor modelada desta forma. Desejamos minimizar o custo da arborescência, mas a aplicação não tolera que a latência fique acima de determinado valor. Mais formalmente, o problema com restrições pode ser definido da seguinte forma. Dados o grafo $G = (V, E)$, onde V é o conjunto vértices e E o conjunto de arestas, um terminal raiz $r \in V$, dois números reais positivos c e d associados a cada aresta, um número real l e um conjunto $T \subseteq V$ de terminais, encontrar $G' = (V', E')$, tal que existam caminhos de r a todo $t \in T$, sendo $path(r, t)$ o conjunto de cada um destes caminhos; $\sum_{e \in E'} c_e$ seja mínimo; e, para todo $t \in T, \sum_{d_p \in path(r,t)} d_p < l$.

Para a maioria das redes utilizadas atualmente o processamento local nos roteadores domina o crescimento da latência de comunicação, portanto, uma simplificação comumente realizada na prática é associar a latência de comunicação ao número de conexões ponto a ponto necessárias para a transmissão completa, da origem ao destino final. Abordamos o problema como se cada canal de comunicação intermediário acrescentasse um valor constante igual a um à latência de comunicação completa, origem-destino, ou seja, $d = 1$ e l igual ao número máximo de saltos permitidos, que chamamos de H. Denominaremos o problema de Steiner em digrafos com restrição de saltos de hcSPDG.

Várias heurísticas distribuídas para o Problema de Steiner com restrições são conhecidas. O DCSP (*Delay-Constrained, Shortest Path*) [Mokbel et al. 1999] propaga a partir do nó raiz *tokens* em inundação, que carregam informações sobre as métricas relevantes para otimização durante seu trajeto até os nós terminais. De posse dos diversos caminhos possíveis, o terminal elege o de menor custo que não viole a restrição e o encaminha para a raiz, para que possa ser unido ao eleito dos outros terminais e formar a solução do problema. Os autores propõem o uso de uma estimativa do diâmetro do grafo para limitação da propagação dos tokens, que é representada pelo valor K. O algoritmo apresenta complexidade de tempo $O(d.K^2.|V|^2)$, sendo d o grau médio dos nós. Liang Guo e Ibrahim Matta introduziram o QDMR (*QoS Dependent Multicast Routing*)[Guo e Matta 1999], baseado no DDMC (*Destination Driven Multicast Routing*) [Shaikh e Shin 1997]. Este algoritmo altera a probabilidade de um terminal ser escolhido para minimizar o custo de uma rota até a fonte dos dados, tornando-a inversamente proporcional à proximidade de violação de restrição da rota passando pelo terminal. Desta forma, caminhos que passem por terminais são preferidos, minimizando-se o custo da solução, enquanto existir folga na restrição. À medida que se aproxima do limite, o algoritmo progressivamente desconsidera o custo, buscando caminhos que pesem menos na restrição. A complexidade em tempo do QDMR é $O(|E|.log|V|)$. Uma versão do PrimSPH [Bauer e Varma 1996] adaptada para trabalhar com restrições pode ser encontrada em [Jia 1998]. Iremos nos referir a este algoritmo por "hcSPH". Inicia-se a solução parcial com um nó raiz. A cada iteração, o terminal mais próximo da solução parcial é incluído a ela utilizando o caminho mínimo que não viole a restrição, entre a solução parcial e o terminal. Todos os nós do caminho entre o terminal e a solução parcial são também incluídos na solução. Este processo se repete até que todos os terminais estejam contidos na solução, que neste ponto deixará de ser parcial. Como o PrimSPH, este algoritmo necessita das distâncias mínimas entre cada par de nós do grafo, o que leva sua complexidade a $O(|V|)$ em tempo e $O(|V|^2)$ para número de mensagens.

Os bons resultados alcançados na utilizazação do Dual-Ascent com o SPDG [Drummond et al. 2007, Santos et al. 2007] nos motivaram a explorar o algoritmo para

o hcSPDG. [Gouveia et al. 2007] introduz uma forma de utilizarmos algoritmos para o MSTna solução do hcMST por meio de uma transformação que aplicada ao grafo original, não direcionado, gera um grafo transformado direcionado, com uma estrutura em camadas que encerra a limitação de saltos, permitindo a utilização de algoritmos para o MST na resolução do hcMST. Neste artigo demonstramos como a idéia pode ser utilizada também para a resolução do hcSPDG e como a transformação pode ser utilizada em situações reais de roteamento multicast. Não é de nosso conhecimento a existência de um algoritmo distribuído específico para o Problema de Steiner com Restrição de Saltos e não consta da literatura comparações de desempenho entre as heurísticas citadas para o hcSPDG. Como o PrimSPH é a heurística que apresenta melhores resultados para o SPDG, com a qualidade da solução suplantada apenas pelo Dual Ascent [Drummond et al. 2007, Santos et al. 2007], implementamos o hcSPH, com as correções posteriormente sugeridas por [Huang e Lee 2005], para fins de balizamento e comparações com o Dual Ascent com restrições de saltos.

2. Dual Ascent Seqüencial

O SPDG considera um grafo direcionado $G_D = (V, A)$, um conjunto $T \subseteq V$ de terminais e uma raiz $r \in T$. O Dual Ascent (DA), proposto por Wong[Wong 1984], é um algoritmo para o SPDG. No algoritmo, cada arco a possui um "custo reduzido" não negativo $\bar{c}_a$ associado, inicialmente com valor igual ao custo original do arco. Os custos reduzidos decrescem durante a execução do algoritmo até atingirem zero quando são ditos "saturados". Estes arcos induzem um "grafo de saturação" $G_S = (V, A_r(\bar{c}))$ onde $A_r(\bar{c}) = \{a \in A : \bar{c}_a = 0\}$. Como todos os custos originais são positivos, este grafo inicialmente não possui arcos. Todas as operações do DA são definidas sobre este grafo de saturação. Seja R um conjunto de nós de um componente fortemente conexo de G_S, este conjunto será um "componente raiz" se (i) R contiver pelo menos um terminal, (ii)R não contiver r, (iii) não existir terminal $t \notin R$ alcançando um R por um caminho em G_S. Dado um componente raiz R, $W(R) \supseteq R$ é o conjunto de nós que atingem R por um caminho em G_S e $\delta^-(W)$ é o corte direcionado composto pelos arcos incidentes a W. Segue o algoritmo:

Dual Ascent
$LI \leftarrow 0$;
$\bar{c}_a \leftarrow c_a$, for all $a \in A$;
Enquanto (existirem componentes raízes em G_S) {
    Escolha um componente raiz R;
    $W \leftarrow W(R)$;
    $\Delta \leftarrow \min_{a \in \delta^-(W)} \bar{c}_a$;
    $\bar{c}_a \leftarrow \bar{c}_a - \Delta$, para todo $a \in \delta^-(W)$;
    $LI \leftarrow LI + \Delta$;
}
Saída: LI e $\bar{c}$

Inicialmente, cada terminal exceto o raiz corresponde a um componente raiz. Em cada rodada, um componente raiz R é selecionado e o custo reduzido de todos os arcos incidentes $W(R)$ são diminuídos de Δ, o menor destes custos reduzidos. O limite inferior parcial é aumentado do mesmo valor. Pelo menos um arco é saturado em cada rodada.

Algumas saturações reduzem o número de componentes raízes, até que não existam mais componentes raízes. Neste ponto G_S contém pelo menos um caminho direcionado de r até cada um dos demais terminais.

Como saída, o DA retorna um limite inferior assim como os custos reduzidos finais. Para conseguirmos uma solução viável para o SPDG, Wong propõe uma heurística adicional sobre G_S. Outros autores [Polzin e Vahdati 2001, de Aragão et al. 2001] descobriram que executar o PrimSPH sobre G_S é muito rápido, por este grafo ser tipicamente muito esparso e produz soluções de melhor qualidade do que o método proposto por Wong. De qualquer forma, a saída final obtida depois de executarmos "DA + PrimSPH sobre G_S"é uma árvore de Steiner. LI provê uma garantia da qualidade da solução. Apesar da qualidade da garantia obtida com o DA ser muito boa na prática, este não é um algoritmo aproximativo.

3. O Dual Ascent Distribuído

A versão distribuída do Dual Ascent foi apresentada em [Drummond et al. 2007, Santos et al. 2007]. Todos os terminais iniciam o crescimento em paralelo de fragmentos. Fragmentos correspondem aos componentes raízes da versão seqüencial. Cada ciclo de crescimento é constituído da difusão no fragmento do valor de Δ a ser subtraído dos custos reduzidos dos arcos incidentes. Saturações podem ocorrer, aumentando o número de nós no fragmento. Informações sobre menores custos reduzidos de arcos incidentes são propagadas pelo fragmento em uma operação de "convergecast", de tal forma que o terminal líder do fragmento saiba o valor do novo Δ que deve ser propagado para que um novo ciclo possa ser iniciado. Estes ciclos de crescimento ocorrem até que uma saturação inclua o terminal raiz r. Neste momento, o fragmento cessa o seu crescimento, informando a r seu limite inferior parcial. Quando r receber o limite inferior de todos os demais fragmento, ele pode calcular o limite inferior global, e o algoritmo termina. A informação sobre o conjunto de arcos do grafo de saturação G_S fica distribuída no sistema e pode ser obtida inspecionando-se os custos reduzidos dos arcos incidentes em cada nó.

Podemos observar na Figura 1 uma ilustração sobre um ciclo de crescimento típico do algoritmo distribuído, onde na parte (a) temos as mensagens de difusão de Δ ("Broad") e inclusão de um novo nó ao componente raiz, através da mensagem "Include", e na parte (b) o "convergecast" com menores valores incidentes locais. O terminal é representado pelo quadrado e os demais nós por círculos.

Os nós recém incluídos em um fragmento devem informar o custo do menor arco incidente ao fragmento. Pode ocorrer que um vizinho já pertença ao fragmento e, portanto, o arco incidente com origem nele não seja incidente ao fragmento. Ao ser incluído em um fragmento, o nó envia uma mensagem "Check" para cada um de seus vizinhos locais e aguarda o recebimento de mensagens "Ack" carregando a informação sobre a pertinência ou não do emissor no fragmento. Com base na informação recebida, o nó incluído sabe quais arcos devem ser considerados incidentes.

Os nós podem pertencer a mais de um fragmento simultaneamente, mas o algoritmo não poderia calcular com segurança o menor valor de arco incidente a um fragmento se os custos reduzidos estivessem sendo alterados concorrentemente por mais de um fragmento. Portanto, quando dois fragmentos possuem um nó em comum, um deles suspende seu crescimento até que o outro termine e ele possa retomar sua operação normal.

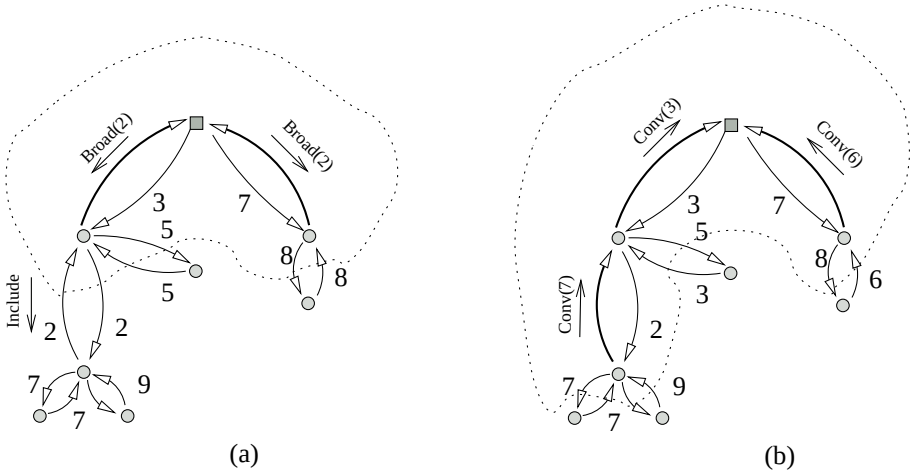

Figura 1. Crescimento de Fragmentos

4. Transformação para o Grafo em Camadas

Um problema semelhante ao hcSPDG é o da árvore geradora mínima com restrições de saltos (hcMST) que consiste em dados o grafo $G = (V, E)$ onde V é o conjunto vértices e E o conjunto de arestas, um custo positivo c associado a cada aresta, um número natural H (número máximo de saltos permitido) e um nó raiz 0, encontrar uma árvore geradora T com custo total mínimo tal que o caminho entre 0 e todos os demais nós não contenha mais do que H arestas.

[Gouveia et al. 2007] sugerem a solução do problema aplicando qualquer algoritmo para solução do SPDG, sobre um grafo transformado em camadas, que encerra a limitação de saltos em sua topologia. O grafo transformado é composto por uma camada inicial contendo a origem 0, e H camadas adicionais, contendo cada uma todos os demais nós diferentes de 0. Arcos direcionados, correspondentes as arestas do grafo original conectam as camadas, sempre da camada de nível superior para a camada de nível inferior. Desta forma o grafo transformado possui uma fonte (o nó 0) e vários sumidouros (os nós da última camada).

Supondo $G = (V, E)$ o grafo original, para o qual a MST é desejada, geramos o grafo transformado $G_t = (V_t, A_t)$ como segue:

- $V_t = \{0\} \cup \{(i, h) : 1 \leqslant h \leqslant H, i \in V \setminus \{0\}\}$
- $A_t = A_0 \cup A_1 \cup A_2$ sendo:
  $A_0 = \{(0, (j, 1)) : j \in V \setminus \{0\}$ e $(0, j) \in E\}$
  $A_1 = \{((i, h), (j, h + 1)) : (i, j) \in E, i \neq 0, 1 \leqslant h \leqslant H - 1\}$
  $A_2 = \{((i, h), (i, H)) : i \in V \setminus \{0\}, 1 \leqslant h \leqslant H - 1\}$

Temos que o hcMST é equivalente ao SPDG sobre o grafo G_t desde que o terminal raiz do SPDG seja o nó raiz do hcMST ($r = 0$) e os terminais do SPDG sejam os nós da última camada do hcMST: $T = \{(i, H) : i \in V_t\}$. O grande mérito desta transformação, é que podemos utilizar todos os algoritmos já conhecidos para o SPDG na resolução do hcMST, inclusive o Dual Ascent.

Uma estratégia semelhante pode ser adotada para resolvermos o hcSPDG com algoritmos originalmente desenvolvidos para o SPDG. Neste caso, o grafo original será orientado, composto por arcos e não arestas, e o sentido dos arcos deve ser levado em consideração quando da geração do grafo transformado.

Mais formalmente, a transformação seria a seguinte. Dados o grafo $G = (V, A)$, onde V é o conjunto vértices e A o conjunto de arcos, um terminal raiz $r \in V$, um conjunto $T \subseteq V$ de terminais e um número natural H (número máximo de saltos permitido), construir $G_t = (V_t, A_t)$ tal que:

- $V_t = \{(i, h) : 1 \leqslant h \leqslant H, i \in V\}$
- $A_t = A_0 \cup A_1$ sendo:
  $A_0 = \{((i, h), (j, h + 1)) : (i, j) \in A, 1 \leqslant h \leqslant H - 1\}$
  $A_1 = \{((i, h), (i, h + 1)) : i \in V, \leqslant h \leqslant H - 1\}$

Desta forma, podemos aplicar algoritmos para o SPDG sobre o grafo transformado. A solução sobre o grafo transformado será uma solução para o hcSPDG sobre o grafo original. A fim de evitarmos ambigüidades, acrescentamos o número da camada à identificação do nó no grafo transformado. Desta forma, o nó (1,2) representa o nó 1 da camada 2, (1,3) o nó 1 na camada 3, e assim por diante. Veja a Figura 2 para um exemplo.

5. Dual Ascent Distribuído sobre o Grafo em Camadas

Considerando o problema seqüencial, qualquer algoritmo para o SPDG pode ser utilizado para solução do hcSPDG, sem qualquer alteração, trocando-se apenas o grafo de entrada do problema para o grafo em camadas, e convertendos-se novamente para o formato original a saída do algoritmo. Para o caso distribuído, o grafo na realidade é uma representação de uma rede de computadores. A geração do grafo em camadas não pode ser feita de forma tão explícita, no entanto a idéia básica ainda pode ser aplicada.

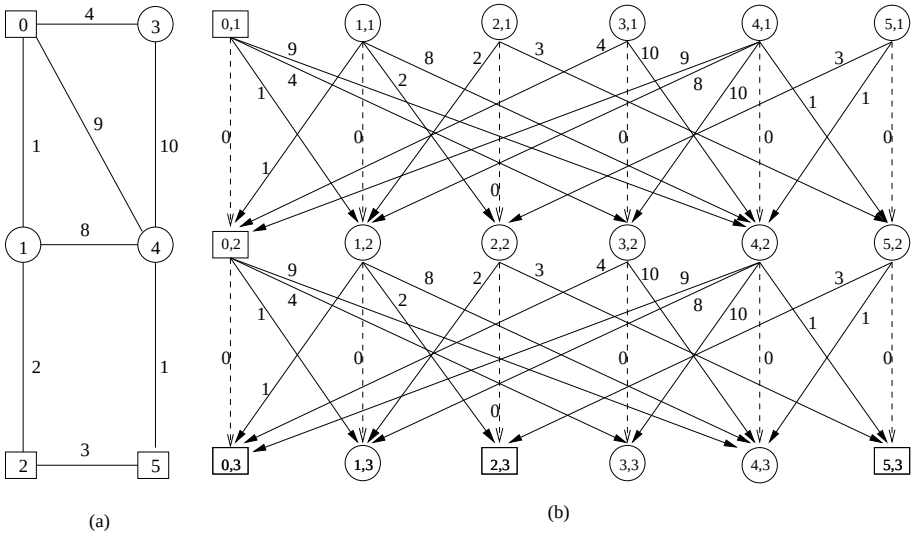

Figura 2. Transformação com limite de saltos H=2 (a) grafo original (b) grafo em camadas

No caso distribuído, para cada nó i do grafo original existe um processo que faz o papel das instâncias $(i, 1)$, $(i, 2)$, ... ,(i, H). Todas as mensagens devem carregar a

camada de destino w de forma a permitir que, ao receber uma mensagem, o processo possa reagir adequadamente, fazendo o papel do nó (i, w). No Dual Ascent sobre o grafo transformado, as mensagens em "broadcast" e de inclusão são sempre para um outro nó em camada igual à camada do emissor menos um. De forma análoga, as mensagens em "convergecast" serão sempre para a camada igual à camada do emissor mais um. Os terminais da última camada iniciam ciclos de crescimento com mensagens de "broadcast".

Duas diferenças topológicas em relação à proposta original de [Gouveia et al. 2007] podem ser observadas: (i) criamos arcos de custo zero entre cada nó e seu correspondente na camada imediatamente superior; (ii) a raiz aparece em todos os níveis, como os demais nós. Estas alterações, simplificam a implementação, pois todos os arcos de entrada e saída das diversas instâncias de um nó se tornam idênticos. Apesar da diferença (i), cada nó continua a possuir um caminho de custo zero até sua instância na camada H. O possível aumento no número de saltos entre um nó e sua instância na última camada é irrelevante, já que na realidade mensagens entre eles não são realmente enviadas pois trata-se do mesmo "nó real". Apesar de existirem diversas instâncias do nó raiz em nossa implementação, todas elas se comportam de maneira idêntica, independente da camada. Ao serem incluídas em um fragmento, respondem com uma mensagem que causa o término do crescimento do fragmento e a difusão do limite inferior parcial até então calculado por seu terminal líder, para que seja calculado o limite inferior global. Pode ocorrer que um fragmento atinja mais de uma instância da raiz em um mesmo ciclo de crescimento. Neste caso, o terminal líder do fragmento receberá duas mensagens de terminação, o que não constitui problema. A raiz receberá duas cópias do limite inferior parcial, devendo ter o cuidado de não calcular duas vezes a terminação do fragmento, o que não é difícil, pois todas as mensagens sempre carregam a identificação do terminal que originou o ciclo de crescimento a que ela pertence.

5.1. Análise do Algoritmo

A corretude da estratégia apresentada baseia-se fortemente nas provas de corretude do Dual-Ascent distribuído apresentada em [Drummond et al. 2007], e do grafo em camadas introduzida em [Gouveia et al. 2007].

Estamos interessados em limites superiores para o número de mensagens trocadas durante a execução do algoritmo e para o tempo global. A complexidade de tempo global assume que o processamento local não consome tempo e também que o tempo de envio de uma mensagem é O(1). A complexidade então é o número de mensagens na maior cadeia causal do tipo "receber uma mensagem e enviar outra em conseqüencia" ocorrida em todas as execuções do algoritmo e sobre todas as variações possíveis na estrutura do grafo [Barbosa 1996]. Como o grafo em camadas é sempre maior do que o grafo original, ($V_t = |V|.|H|$ e $E_t = |E|.|H|$), poderíamos esperar custos de execução do Dual Ascent iguais a $O(|T|.|V_t|^2)$ para número de mensagem e tempo, que são as complexidades do Dual Ascent distribuído proposto em [Drummond et al. 2007]. No entanto, como a topologia do grafo transformado é muito particular, alguma análise extra é necessária.

Para a complexidade de mensagens, o pior caso ocorrerá em um grafo completo, onde todos os terminais, incluindo a raiz, são conectados pelos arcos de maior custo do grafo, desta forma, estes arcos só serão saturados quando todos os nós não terminais tiverem sido incluídos em todos os fragmentos. Quando um nó é incluído em um fragmento, ele envia mensagens "Check" para todos os seus vizinhos que, no caso do grafo

em camadas será igual ao número de nós do grafo original V. Portanto, o crescimento de um fragmento não enviará mais que $|V_t|.|V|$ mensagens "Check". Como existirá um fragmento para cada terminal, o número de mensagens "Check" enviadas durante toda execução do algoritmo será da $O(|T|.|V_t|.|V|)$. Cada rodada de crescimento demanda $O(|V_t|)$ mensagens, como podem existir $O(|V_t|)$ rodadas por fragmento, $O(|T|.|V_t|^2)$ é um limite superior para o número de mensagens de broadcast enviadas. Portanto, este é um limite superior válido para o número total de mensagens enviadas. Não houve melhora no limite teórico em função da topologia especial do grafo transformado para o número de mensagens enviadas.

A pior situação em termos de tempo global no caso do Dual Ascent ocorre quando um único fragmento pode crescer de cada vez, porque todos os demais fragmentos que ainda não tenham incluído a raiz estão suspensos. O crescimento de um único fragmento pode consumir $O(|V_t|.|H|)$ tempo. Cada rodada de crescimento requer uma cadeia de mensagens de broadcast e convergecast proporcional ao maior caminho de arcos saturados na árvore do fragmento. Este caminho mais longo pode ter $O(|H|)$ em comprimento e o crescimento completo deste fragmento pode gastar $O(|V_t|)$ rodadas. Como existem $|T|$ fragmentos a complexidade em tempo fica $O(|T|.|V_t|.|H|)$. A topologia especial do grafo em camadas torna possível um limite superior mais baixo quando comparado com a execução do algoritmo em um grafo qualquer de mesmo tamanho.

6. Resultados Experimentais

A fim de avaliarmos a qualidade das soluções geradas e o custo computacional da execução do algoritmo proposto, o implementamos utilizando linguagem C e MPICH2-1.0.6 e executado em 15 processadores Athlon 1.8 GHz. Parte dos testes foi executada sobre instâncias de *benchmark* adaptadas das séries B, I080 e P4Z da SteinLib [Koch et al.]. A série B é composta por grafos aleatórios de diferentes densidades e custos entre 1 e 10. As instâncias da classe I080 também são também grafos aleatórios, com custos dos arcos escolhidos para tornar a resolução do problema de Steiner mais difícil. A série P4Z contêm somente grafos completos. Nenhuma das instâncias disponíveis na SteinLib é direcionada, como o Dual Ascent foi projetado para o caso mais geral de grafos direcionados, substituímos cada aresta original do grafo por dois arcos antiparalelos com custos iguais ao custo original multiplicado por um número aleatório uniformemente distribuído entre [0.5,1.5] e arredondado. Para os grafos completos, I080-021, I080-121, I080-221 e I080-321 e toda a classe P4Z, realizamos testes com 2, 3, 4 e 5 para limite de saltos. Para os demais grafos foi calculada a menor distância entre a raiz e todos os terminais, e a maior entre todas as distâncias calculadas foi utilizada como limite de saltos.

As instâncias das classes GLP e SW referem-se a grafos que tentam reproduzir as características da INTERNET. As instâncias GLP foram geradas utilizando o sistema BRITE ("Boston University Representative Internet Topology Generator") [Medina et al. 2001]. O BRITE gera diversos modelos de grafos aleatórios. Escolhemos o mais recentemente implementado que é o GLP ("Generalized Linear Preference"), pois estudos recentes indicam que este modelo reproduz melhor as características da INTERNET [Bu e Towsley 2002]. A classe SW foi construída segundo [Jin e Bestavros 2006], modelo ainda não implementado no BRITE, que ressalta as características "Small World" da Internet. Para estas classes escolhemos os terminais aleatoriamente entre os nós que estivessem entre 3, 4 e 5 saltos de distância do terminal raiz, e utilizamos estas distâncias

(3, 4 e 5) como limite de saltos para cada instância.

As colunas na Tabela 1 possuem o seguinte significado: $|V|$, $|A|$ e $|T|$ mostram o tamanho da instância; H o limite de saltos utilizado; Otm o valor da solução ótima calculada com o algoritmo "branch-and-bound" de [de Aragão et al. 2001]; SPHc o valor da solução obtida na execução do hcSPH sobre o grafo original; LI é o limite inferior obtido com o Dual Ascent; $(|Ar|/|A|)$ é a proporção de arcos que são saturados; DA+SPHcr é o valor da solução obtida na execução do hcSPH sobre o grafo de arcos saturados gerado pelo Dual Ascent. Os resultados referentes às execuções com 2, 3 e 4 para limite de saltos nas instâncias completas foram omitidas da tabela.

Observe que o Dual Ascent pode levar a resultados diferentes se os fragmentos crescerem em diferentes velocidades. A velocidade de crescimento dos fragmentos depende principalmente dos escalonadores dos diversos sistemas operacionais utilizados nos experimentos, incluindo um caráter aleatório às execuções do Dual Ascent. Portanto, este algoritmo foi executado 5 vezes e os resultados apresentados constituem a média aritmética das 5 execuções. Como durante nossos testes todos os computadores e a rede de comunicação foram de uso exclusivo, obtivemos um desvio padrão muito baixo: 0,6% para os valores das soluções.

Utilizamos *competitividade*, a razão entre o valor da solução obtida com a heurística e o ótimo para comparar a qualidade das soluções obtidas com e sem a utilização do Dual Ascent. Os gráfico das figuras 3 e 4 mostram o percentual cumulativo dos casos onde a competitividade é menor ou igual ao valor do eixo das abcissas, respectivamente para as instâncias da "Steinlib" e representativas da INTERNET. Claramente a utilização do Dual Ascent leva a resultados melhores, em média 4,1%. Os gráficos das figuras 3 e 4 mostram também uma linha referente ao melhor resultado entre os dois algoritmos testados. Em uma minoria dos casos, a execução do SPHc sem o Dual Ascent leva a melhores soluções, como pode ser observado na Tabela 6. Portanto, se executarmos as duas abordagens e tomarmos o melhor resultado, obteremos melhores soluções do que em ambos os casos separadamente. No entanto, existe uma abordagem mais interessante, que leva a resultados muito semelhantes. Se após a execução do DA+SPHcr, executarmos o SPHc sobre a instância completa apenas se a variação percentual entre a solução obtida e o limite inferior estiver acima de um determinado valor, conseguiremos praticamente a mesma qualidade de solução da execução dos dois algoritmos com menores custos, pois o SPHc sobre a instância completa será executado em menos de 50% dos casos.

Também comparamos o desempenho prático do Dual Ascent e do PrimSPH com respeito ao tempo, dado pelo tamanho da maior cadeia de mensagens com relação de causalidade e o número total de mensagens enviadas. Os resultados destas medidas podem ser observados nos gráficos das figuras 5, 6, 7 e 8.

Observamos que o Dual Ascent apresenta maior sensibilidade ao limite de saltos do que o SPHc. O grafo transformado é sempre H vezes maior do que o grafo original e o Dual Ascent emprega mensagens "Check" para as diversas instâncias de um nó nas diversas camadas e explora as diversas instâncias de um mesmo arco, o que explica os custos maiores. Podemos observar a sensibilidade do Dual Ascent para o limite de saltos principalmente nas instância B10 à B12, que possuem limites maiore de saltos e nas instâncias completas, quando variamos o limite de saltos (2, 3, 4 e 5). Nestes casos

Instancia						SPHc	Dual Ascent		DA+SPHcr
Name	$\|V\|$	$\|A\|$	$\|T\|$	$\|H\|$	Otm	Solução	LI	$(\|Ar\|/\|A\|)\%$	Solução
b01d	50	126	9	5	79	**79**	79	64,98	**79**
b02d	50	126	13	5	136	155	136	82,51	**136**
b03d	50	126	25	8	175	179	171,6	86,31	**176**
b04d	50	200	9	4	63	73	63	44,45	**63**
b05d	50	200	13	3	82	84	82	46,47	**83**
b06d	50	200	25	6	144	155	131,4	62,55	**130**
b07d	75	188	13	6	126	132	126	70,57	**126**
b08d	75	188	19	6	140	163	139	77,41	**138**
b09d	75	188	38	7	240	246	239,4	83,36	**240**
b10d	75	300	13	5	109	139	105,4	50,88	**114**
b11d	75	300	19	5	138	155	123,2	59,75	**136**
b12d	75	300	38	5	207	307	198,2	62,56	**222**
b13d	100	250	17	6	201	219	196,4	76,04	**207**
b14d	100	250	25	7	288	296	285,2	87,54	**288**
b15d	100	250	50	9	370	391	364	95,52	**361**
b16d	100	400	17	6	167	218	142	58,35	**176**
b17d	100	400	25	6	160	167	154,2	57,18	**155**
b18d	100	400	50	5	287	327	271,2	61,96	**296**
i080-001d	80	240	6	3	2.052	**2.320**	2.320,00	55,28	**2320**
i080-101d	80	240	8	6	2.322	2.574	2.309,80	48,01	**2322**
i080-201d	80	240	16	6	4.408	4.744	4.337,60	63,35	**4725**
i080-301d	80	240	20	5	6.231	6.669	5.975,00	72,52	**6224**
i080-031d	80	320	6	4	1.646	**1.793**	1.628,00	42,97	1795
i080-131d	80	320	8	4	1.998	2.356	2.094,80	45,42	**1975**
i080-231d	80	320	16	4	4.686	5.186	4.382,20	53,92	**4923**
i080-331d	80	320	20	4	4.715	5.224	4.666,20	52,59	**5005**
i080-011d	80	700	6	2	1.744	**1.744**	1.744,00	20,14	**1744**
i080-111d	80	700	8	3	2.185	2.455	2.073,00	24,71	**1953**
i080-211d	80	700	16	3	3.411	4.143	3.128,00	27,62	**3967**
i080-311d	80	700	20	3	3.942	4.508	3.672,00	26,87	**4187**
i080-041d	80	1.264	6	2	953	1.026	940	10,82	**953**
i080-141d	80	1.264	8	2	2.382	4.008	2.382,00	12,81	**2470**
i080-241d	80	1.264	16	2	3.769	4.625	3.670,60	15,89	**3311**
i080-341d	80	1.264	20	2	4.064	4.621	3.910,40	15,36	**4390**
i080-021d	80	6.320	6	5	741	847	673	7,87	**834**
i080-121d	80	6.320	8	5	977	**1.097**	928	5,37	2695
i080-221d	80	6.320	16	5	1.982	2.293	1.816,00	7,93	**2291**
i080-321d	80	6.320	20	5	2.449	3.137	2.236,00	6,99	**2828**
P401d	100	9.900	5	5	158	**158**	158	1,51	**158**
P402d	100	9.900	5	5	104	**104**	104	1,42	**104**
P403d	100	9.900	5	5	181	**219**	181	1,73	220
P404d	100	9.900	10	5	334	349	300,8	1,87	**268**
P405d	100	9.900	10	5	276	325	274	1,78	**300**
P406d	100	9.900	10	5	346	454	332,8	2,05	**396**
P407d	100	9.900	20	5	673	836	596	2,19	**664**
P408d	100	9.900	20	5	576	633	552	2,1	**608**
glp01	100	358	5	3	3.043	**3.043**	2.959,00	46,2	3278
glp02	100	358	5	4	4.352	**4.496**	3.895,20	51,86	4786
glp03	100	374	5	5	3.671	3.705	4.026,00	52,78	**3671**
glp04	100	356	10	3	3.943	**3.943**	4.343,00	44,36	**3943**
glp05	100	336	10	4	7.526	**7.526**	6.964,00	50,76	7962
glp06	100	312	10	5	8.557	8.694	8.155,20	49,91	**8557**
glp07	100	372	15	3	9.870	10.214	10.424,00	45,16	**9870**
glp08	100	374	15	4	8.343	8.796	8.131,00	45,83	**8663**
glp09	100	400	15	5	5.945	6.122	6.066,60	48,07	**6069**
sw01	100	2.486	5	3	588.938	735.326	542.016,00	9,02	**684718**
sw02	100	2.148	5	3	130.563	**130.563**	130.563,00	7,18	**130563**
sw03	100	2.554	5	3	306.196	382.708	306.196,00	7,5	**306196**
sw04	100	2.300	5	4	105.341	110.257	105.341,00	7,42	**105341**
sw05	100	2.014	5	4	264.669	**269.626**	259.144,00	9,16	**269626**
sw06	100	1.784	5	4	239.683	267.859	235.847,00	10,5	**239683**
sw07	100	2.154	5	5	143.988	166.431	126.302,00	8,95	**163169**
sw08	100	2.052	5	5	99.168	113.539	98.989,40	9,45	**100477**
sw09	100	2.250	5	5	236.336	**262.207**	233.767,20	11,63	**262207**
sw10	100	3.106	10	3	383.049	510.486	337.111,00	6,75	**487943**
sw11	100	1.720	10	3	313.959	359.860	308.818,00	10,06	**350411**
sw12	100	3.010	10	3	1.096.275	**1.178.259**	1.096.275,00	10,68	**1178259**
sw13	100	1.866	10	4	362.021	620.420	362.021,00	11,04	**503299**
sw14	100	1.976	10	4	794.572	794.697	794.572,00	14,33	**794572**
sw15	100	2.490	10	4	501.390	655.357	412.620,00	10,98	**501407**
sw16	100	2.628	10	5	165.434	**183.654**	155.251,00	8,52	195700
sw17	100	2.486	10	5	230.683	354.632	200.117,00	10,13	**266769**
sw18	100	3.078	10	5	118.958	176.544	112.488,00	6,77	**145887**
sw19	100	2.056	15	3	639.718	647.029	636.710,00	12,09	**644513**
sw20	100	2.378	15	3	409.818	529.196	374.192,00	8,54	**442350**
sw21	100	2.896	15	3	438.285	561.050	420.087,00	8,78	**465200**
sw22	100	2.256	15	4	489.096	803.833	375.412,00	10,67	**580640**
sw23	100	1.920	15	4	385.990	690.621	385.990,00	11,64	**445032**
sw24	100	2.364	15	4	501.292	537.612	488.860,00	10,33	**536724**
sw25	100	2.214	15	5	364.148	510.276	313.994,00	11,6	**384652**
sw26	100	2.144	15	5	263.517	391.351	251.864,00	10,51	**307810**
sw27	100	2.088	15	5	883.258	1.095.476	759.095,00	17,39	**1044499**

Tabela 1. Alguns resultados práticos obtidos

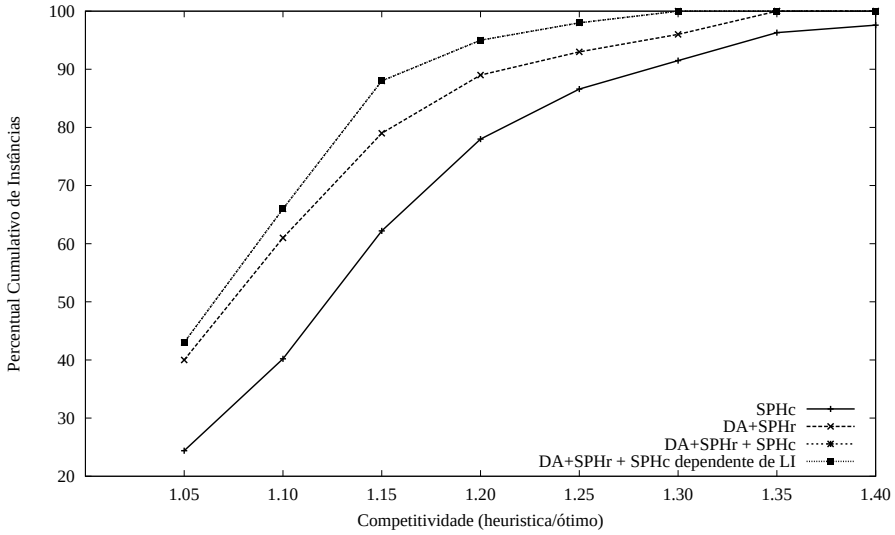

Figura 3. Qualidade das soluções para as instâncias da SteinLib

podemos observar claramente o rápido crescimento do custo do Dual Ascent em função do limite de saltos pela aparência de "serra" apresentada pelos gráficos. Apesar disto, para as instâncias de grande densidade, a aplicação do SPHc sobre a instância reduzida compensa o aumento do custo decorrente do aumento do grafo, principalmente quanto ao número de mensagens enviadas.

O gráfico da Figura 9 foi construído utilizando-se apenas os grafos completos, últimas instâncias da série I080 e toda a série P4Z, ordenados por número de terminais. Nestas instâncias, executamos os algoritmos diversas vezes, com limites de saltos diferentes de forma a poder estudar mais detalhadamente o comportamento do aumento do número de saltos sobre paralelismo conseguido na execução do Dual Ascent. O SPHc apresentou baixa sensibilidade ao número de saltos, portanto, traçamos uma linha única para a média dos valores obtidos com este algoritmo para todos os limites utilizados (2, 3, 4 e 5). No caso do Dual Ascent, traçamos uma linha para cada limite de saltos. Podemos observar por estas figuras que para limites de saltos baixos, a execução do Dual Ascent é comparável ao do SPHc e, mais importante, o ritmo de crescimento dos custos no Dual Ascent em função do crescimento da instância não é significativamente diferente para o Dual Ascent e o SPHc.

7. Conclusão

Foi mostrado como a versão distribuída do Dual Ascent apresentada em [Drummond et al. 2007, Santos et al. 2007] pode ser utilizada para o problema de Steiner com restrição de saltos. Esta variação do problema é de grande interesse prático, pois modela muito bem a transmissão de multimídia em multicast pela internet, quando as aplicações tipicamente não toleram uma latência acima de determinados limites.

A melhor qualidade da solução obtida com a utilização do novo algoritmo implica em aumento de custos, principalmente caso o limite de saltos tolerado for alto, mas não detectamos diferenças significativas no ritmo de crescimento destes custos.

O algoritmo foi testado em instâncias que mimetizam a internet (BRITE-GLP,

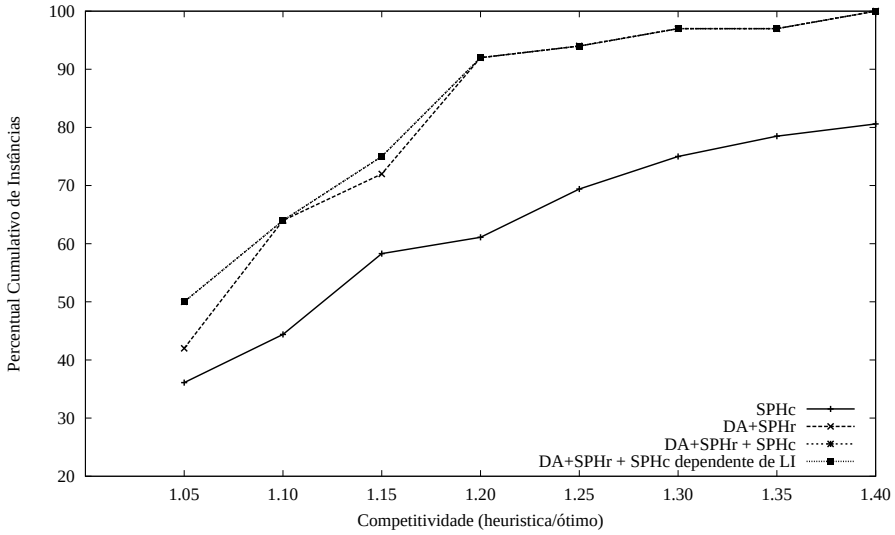

Figura 4. Qualidade da solução para as instancias simuladoras da INTERNET

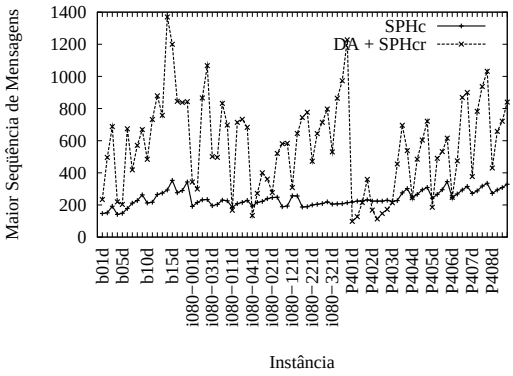

Figura 5. Tempo global nas instâncias da SteinLib

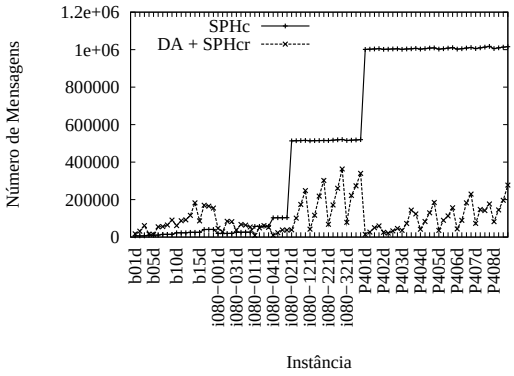

Figura 6. Número de mensagens nas instâncias da SteinLib

SW) com resultados muito semelhantes aos obtidos em instâncias geradas para avaliação do problema de um ponto de vista mais teórico (SteinLib), o que reforça nossa crença de o Dual Ascent ser uma alternativa a ser considerada para a distribuição de dados em multicast pela INTERNET.

Referências

Barbosa, V. C. (1996). *An introduction to distributed algorithms*. The MIT Press, Cambridge, MA, USA.

Bauer, F. e Varma, A. (1996). Distributed algorithms for multicast path setup in data networks. *IEEE/ACM Transactions on Networking*, pages 181–191.

Bu, T. e Towsley, D. F. (2002). On distinguishing between internet power law topology generators. *IEEE Infocom*, 2:638–647.

de Aragão, M. P., Uchoa, E., e Werneck, R. (2001). Dual heuristics on the exact solution of large Steiner problems. *Electronic Notes in Discrete Mathematics*, 7:46–51.

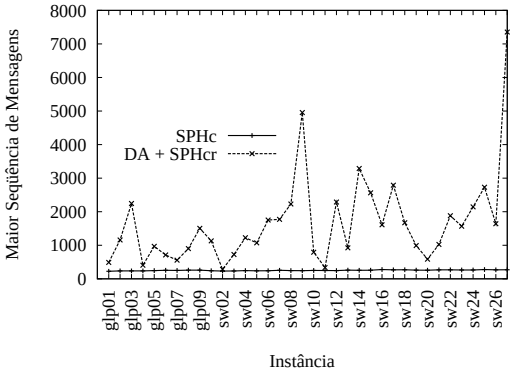

Figura 7. Tempo global nas instâncias da INTERNET

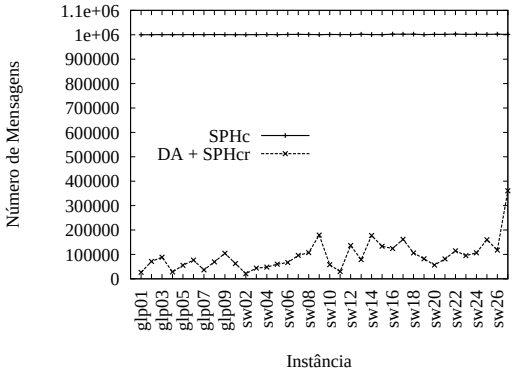

Figura 8. Número de mensagens nas instâncias da INTERNET

Drummond, L. M. A., Santos, M., e Uchoa, E. (2007). A distributed dual ascent algorithm for steiner problems in multicast routing. *Networks (aceito para publicação)*.

Gouveia, L., Simonetti, L., e Uchoa, E. (2007). Modelling the hop-constrained minimum spanning tree problem over a layered graph. In *Proceedings of the International Network Optimization Conference - INOC'07- in cd*.

Guo, L. e Matta, I. (1999). QDMR: An efficient QoS dependent multicast routing algorithm. In *Proceedings of the Fifth IEEE Real-Time Technology and Applications Symposium (RTAS '99)*, pages 213–223.

Huang, T.-L. e Lee, D. T. (2005). Comments and an improvement on "a distributed algorithm of delay-bounded multicast routing for multimedia applications in wide area networks". *IEEE/ACM Transactions on Networking*, 13:1410–1411.

Jia, X. (1998). A distributed algorithm of delay-bounded multicast routing for multimedia applications in wide area networks. *IEEE/ACM Transactions on Networking*, 6:828–837.

Jin, S. e Bestavros, A. (2006). Small-world characteristics of internet topologies and implications on multicast scaling. *Computer Networks Journal*, 50:648–666.

Koch, T., Martin, A., e Voss, S. Steinlib: an updated library on steiner problems in graphs. *Konrad-Zuse-Zentrum für Informationstechnik Berlin, ZIB-Report 00-37, http://elib.zib.de/steinlib*.

Medina, A., Lakhina, A., Matta, I., e Byers, J. (2001). Brite: An approach to universal topology generation. In *Proceedings of the International Workshop on Modeling, Analysis and Simulation of Computer and Telecommunications Systems - MASCOTS'01*.

Mokbel, M. F., Elhaweet, W. A., e Elderini, M. N. (1999). A delay-constrained shortest path algorithm for multicast routing in multimedia applications. In *IEEE Middle East Workshop on Networking*.

Novak, R., Rugelj, J., e Kandus, G. (2001). *Steiner tree based distributed multicast routing in networks*, volume 28, pages 327–352.

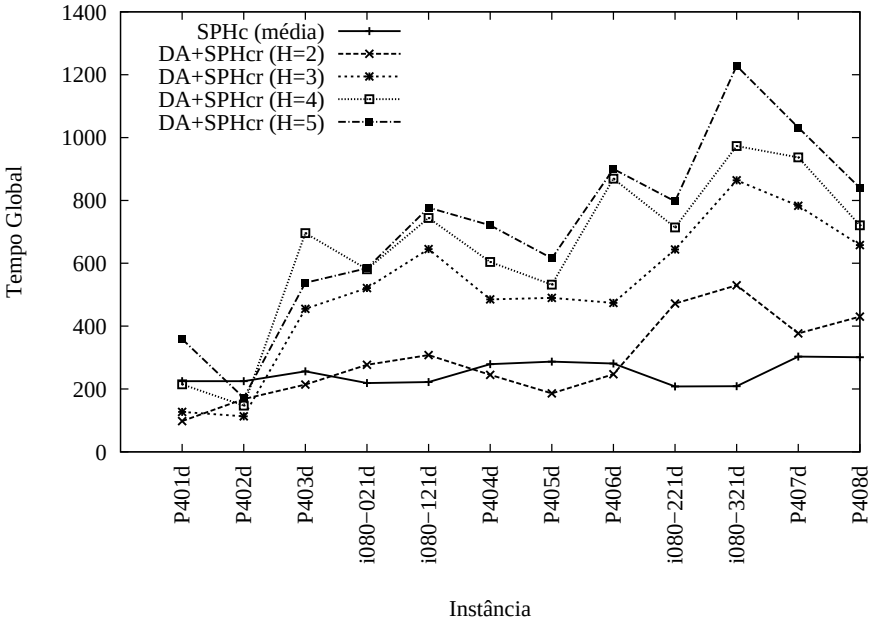

Figura 9. Tamanho da maior cadeia causal para os diversos limites de saltos em instâncias completas

Oliveira, C. A. S. e Pardalos, P. M. (2005). A survey of combinatorial optimization problems in multicast routing. *Computers & Operations Research*, 32:1953–1981.

Polzin, T. e Vahdati, S. (2001). Improved algorithms for the steiner problem in networks. *Discrete Applied Mathematics*, 112:263–300.

Santos, M., Drummond, L. M. A., e Uchoa, E. (2007). Distributed dual ascent algorithm for steiner problems in networks. In *25° Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*, pages 381–396.

Shaikh, A. e Shin, K. G. (1997). Destination-driven routing for low-cost multicast. *IEEE Journal on Selected Areas in Communications*, 15:373–381.

Wong, R. T. (1984). A dual ascent approach for steiner tree problems on a directed graph. *Mathematical Programming*, 28:271–287.