Dual Ascent Distribuído para o Problema de Steiner *On-Line*

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

[1]Universidade Federal Fluminense

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

Abstract. *The On-Line Steiner Problem consists in finding a minimum cost tree that reaches a subset of nodes of a graph from a root node r. In the On-Line variation of the Steiner problem, the terminals set can change over time, with nodes entering in the terminals set and/or terminals getting out of this set, turning into common nodes. The major part of the known algorithms for the problem uses simple approachs for inclusion and exclusion operations, and uses known static algorithms periodically to reorganize the tree, leading to better solution quality. This work presents a heuristic for the On-Line problem, based on the "Dual Ascent" algorithm proposed by [Wong 1984] that triggers reorganizations more opportunistically than the main known heuristics for the problem.*

Resumo. *O Problema de Steiner consiste em minimizar o custo de uma arborescência que atinja um subconjunto de nós, ditos "terminais", de um grafo a partir de um nó raiz r. No Problema de Steiner On-Line, o conjunto de terminais sofre alterações com o passar do tempo, podendo nós comuns entrar no conjunto de terminais e/ou terminais saírem deste conjunto, se transformando em nós comuns. A maioria dos algoritmos conhecidos para este problema atualmente usa técnicas simples para inclusão ou exclusão de terminais, utilizando algoritmos conhecidos para o problema estático periodicamente para reorganizações na árvore, levando a qualidades de solução superiores. Este trabalho apresenta uma heurística distribuída para o problema On-Line, baseada no algoritmo "Dual Ascent" proposto por [Wong 1984], que dispara reconstruções na árvore de distribuição de forma mais oportuna do que os principais algoritmos conhecidos atualmente.*

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 frequentemente modelado como o Problema de Steiner em Grafos Direcionados (*Steiner Problem in Directed Graphs - SPDG*) [Novak et al. 2001b, Oliveira e Pardalos 2005].

Em várias situações práticas pode acontecer de terminais entrarem e saírem do conjunto de terminais de forma dinâmica, por exemplo, em uma distribuição de vídeo sob demanda, usuários finais podem solicitar sua inclusão no grupo de multicast depois desse iniciado, ou usuários podem desligar seus receptores antes do final da transmissão. Neste caso, o problema de Steiner é chamado de Problema de Steiner *On-Line* e é definido

da seguinte forma. Dados o grafo $G_d = (V, A)$ onde V é o conjunto de vértices e A o conjunto de arcos, um terminal raiz $r \in V$, um custo positivo c_a associado a cada arco $a \in A$, um conjunto $T_0 \subseteq V$, de terminais (conjunto inicial de terminais), uma sequência de conjuntos de terminais $T = (T_1, T_2, ..., T_n)$ determinada por uma sequência de operações $(o_1, o_2, ..., o_n)$ onde cada operação o_i pode ser: uma inclusão de um terminal, $T_i = T_{(i-1)} \cup \{v\}, v \in V, v \notin T_{(i-1)}$ ou uma exclusão de um terminal, $T_i = T_{(i-1)} - \{t\}, t \in T_{(i-1)}$, encontrar uma sequência de grafos $S = (G'_0, G'_1, G'_2, ..., G'_n)$, tal que $G'_i = (V'_i, A'_i)$, onde existam caminhos de r a todo $t \in T_i$, $T_i \subseteq V'_i$ e $\sum_{a \in A'_i} c_a$ seja mínimo.

Podemos sempre aproveitar a árvore existente, podando ramos que não sejam mais necessários quando ocorrerem exclusões de terminais ou unindo à árvore antiga o caminho mínimo dela até um novo nó que seja incluído no conjunto de terminais, no entanto, com o acúmulo destas operações, pode ocorrer que a qualidade da árvore caia muito.

Atualizar a árvore de distribuição pode ser um processo computacionalmente caro, e nem sempre o ganho obtido com a melhoria da árvore de distribuição compensa esse custo. Sistemas para esse problema então, tentam disparar reorganizações apenas periodicamente, a maioria deles dispara reorganizações quando o número de alterações sofrida pela árvore de distribuição ultrapassa um determinado limite (ARIES [Bauer e Varma 1997], EBA [Imase e Waxman 1991]). Um dos problemas desta abordagem, é que o número de alterações nem sempre implica na queda da qualidade da solução para o problema de Steiner. Pode ocorrer de termos boas soluções mesmo após muitas alterações ou o inverso, uma única alteração pode comprometer seriamente a qualidade da solução.

Apresentamos neste trabalho uma adaptação do algoritmo Dual Ascent [Drummond et al. 2008, Santos et al. 2007] para o problema *On-Line*. O Dual Ascent oferece como saída um limite inferior, que é uma boa aproximação para o ótimo do problema de Steiner (tipicamente menos 3% de distância). Utilizamos este limite inferior para avaliar a qualidade da solução atual para o problema, disparando reorganizações quando detectamos que a distância entre a qualidade da solução e o limite inferior supera um determinado limite.

A abordagem foi implementada para testes e, quando comparada com o disparo de reorganizações baseado no número de alterações, mostrou-se mais eficiente, disparando menos reorganizações e apresentando um ganho por reorganização superior, enquanto a qualidade média da solução é mantida em níveis semelhantes.

2. Dual Ascent Sequencial

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 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, Poggi de Aragão et al. 2001] descobriram que executar o Prim-SPH 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. 2008, Santos et al. 2007]. Todos os terminais iniciam o crescimento em paralelo de fragmentos. Fragmentos correspondem aos componentes raízes da versão sequencial. Cada ciclo de crescimento é constituído do "broadcast" 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. Os ciclos de "broadcast" e "convergecast" são realizados com o auxílio de "árvores de saturação" construídas sobre o grafo de saturação e enraizadas no terminal criador do fragmento. Os ciclos continuam 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.

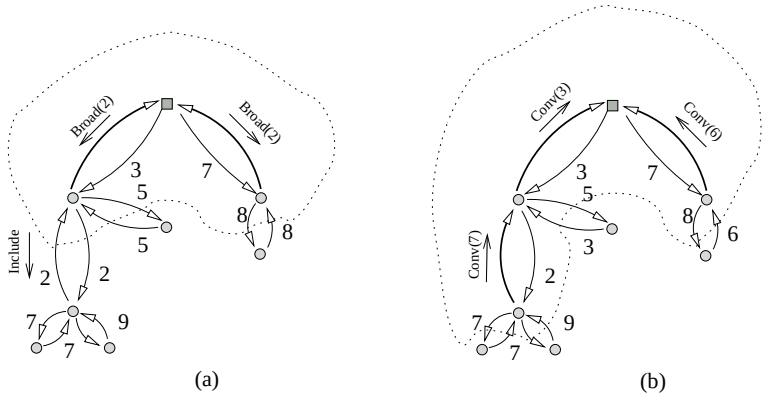

Figura 1. Crescimento de Fragmentos

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.

4. Algumas Heurísticas Distribuídas para o Problema de Steiner *On-Line*

O problema dinâmico é equivalente a uma sequência de problemas estáticos, portanto, podemos utilizar, para sua solução, sucessivas aplicações de algoritmos para o problema estático. No entanto, essa abordagem pode apresentar custos desnecessariamente altos já que, na maioria dos casos, estruturas construídas no problema anterior continuam válidas e podem ser aproveitadas para a construção da solução após uma operação de inclusão ou exclusão de um nó do conjunto de terminais.

Não é de nosso conhecimento qualquer algoritmo que trate especificamente da versão direcionada desse problema, mas propostas existem para a versão não direcionada

e algumas delas podem ser facilmente adaptadas para grafos direcionados. A primeira e mais direta abordagem do tipo gulosa é proposta por [Waxman 1988]. Inclusões são feitas sempre com a utilização dos caminhos mínimos entre o nó incluído e a solução atual, e as exclusões são feitas apenas quando o terminal excluído possui grau um. Exclusões de terminais podem gerar nós não terminais de grau um, que são também excluídos da solução. Essa abordagem é bastante simples e pode ser adaptada facilmente para o caso direcionado. No entanto, vários trabalhos posteriores apresentaram soluções mais sofisticadas. O EBA (*Edge-Bounded Algorithm*) [Imase e Waxman 1991] mantém as estratégias para inclusão e exclusão de nós do método guloso, mas após cada inclusão, verifica-se a possibilidade de baixar o custo da árvore de distribuição, verificando se o custo de conexão de algum nó antigo pode ser diminuído utilizando-se, mesmo que parcialmente, o caminho adicionado à árvore para a inclusão do novo nó.

O ARIES [Bauer e Varma 1997] é provavelmente o trabalho mais influente na área, estando entre os mais frequentemente citados e fornecendo base outros trabalhos publicados. O autor utiliza a mesma estratégia do EBA para inclusão e exclusão de nós, mas observa que o EBA tenta melhorar a solução somente quando ocorrem inclusões, o que pode baixar sua qualidade após exclusões sucessivas. A fim de contornar esse ponto fraco do EBA, o ARIES sugere monitorar o número de alterações processadas (inclusões e exclusões) e disparar "reorganizações" visando melhorar a qualidade da solução quando o número de alterações ultrapassa um determinado limite. No ARIES quando um terminal t é incluído ou excluído, uma mensagem *counter_update* é enviada aos terminais vizinhos. Considera-se um terminal t_1 vizinho a outro t_2, quando não existem outros terminais no caminho entre t_1 e t_2 na solução atual. O algoritmo define região como um subgrafo $G_i = (V_i, E_i)$ da instância $G = (V, E)$ para o problema, onde V_i possui pelo menos um nó v incluído ou excluído do conjunto de terminais que não tenha passado por reorganizações e todos os vizinhos de v, e E_i é o conjunto das arestas (v_j, v_k) tais que $v_j, v_k \in V_i$ e $(v_j, v_k) \in E$. Terminais mantêm contadores associados a cada região de que façam parte. Ao receber uma mensagem *counter_update*, o terminal incrementa o contador correspondente à região associada. Quando o contador ultrapassa um determinado limite, a região passa por uma reorganização. A reorganização consiste em excluir todas as arestas da região a ser reorganizada, desconectando terminais da árvore e reconectando-os utilizando o algoritmo K-SPH [Bauer e Varma 1996], heurística não adaptável para a versão orientada do problema de Steiner. A versão original do ARIES não permite operações de inclusão, exclusão ou reorganização simultâneas. [Adelstein et al. 2003] aprimora o algoritmo para permitir operações simultâneas.

[Gatani et al. 2006] utiliza o ADH para construir uma solução inicial que é posteriormente mantida utilizando uma técnica semelhante a do ARIES, com contadores controlando o número de alterações ocorridas em determinadas regiões do grafo. Gatani utiliza a *stirring techinique* [di Fatta e Re 1998] para reorganização, técnica viável para grafos orientados e não orientados. Apesar de trabalhar com grafos não orientados, di Fatta assume que existe um terminal especial chamado *source* e define o conjunto de ancestrais de um terminal como sendo os nós da solução para o SPG no caminho entre este terminal e o nó *source*. Isto posto, podemos definir *grafting point* de um terminal como seu ancestral mais próximo que possua pelo menos dois filhos. A *stirring techinique* consiste em reavaliar todas possibilidades para *grafting points* alternativos, que levem a árvores de menor custo. Para cada terminal t, é buscado um nó v_k tal que a distância entre

t e v_k seja menor que a distância entre t e v_a sendo v_a o *grafting point* atual de t. Este procedimento é repetido até que não sejam conseguidas soluções de menor custo.

Apresentamos neste trabalho uma abordagem para o problema *On-Line* baseada na versão distribuída do Dual Ascent. Nossa heurística dispara reorganizações quando o custo da solução para o problema de Steiner se afasta do limite inferior obtido com o DA além de um determinado limite. Desta forma, esperamos que as reorganizações sejam mais oportunas do que as realizadas com base em mecanismos que avaliem o número de alterações realizadas em regiões do grafo, como a maioria das soluções atualmente conhecidas.

5. Algoritmo Dual Ascent para o Problema de Steiner *On-Line*

A ideia geral do algoritmo é realizar operações de exclusão e inclusão de terminais como no EBA e no ARIES, mantendo atualizadas dinamicamente, de forma distribuída, as árvores de saturação, os grafos de saturação, o limite inferior e a solução para o Problema de Steiner atual. Quando a diferença entre o limite inferior e o custo da solução ultrapassar uma determinada tolerância, dispara-se uma reorganização que tenta baixar o custo da solução utilizando um algoritmo para o problema estático sobre o grafo de saturação.

O nó raiz é o nó onde o custo da solução e o limite inferior global são mantidos durante a execução do algoritmo, portanto, é o nó mais adequado para disparar as reorganizações quando necessário. No nó raiz pode-se manter com maior facilidade o estado do sistema, que pode estar em reorganização, processando alterações no conjunto de terminais, ou estável.

Desejamos utilizar qualquer algoritmo distribuído para a solução do problema estático como rotina de reorganização, portanto, é nosso interesse fazer com que as rotinas de exclusão e inclusão de terminais sejam independentes da reorganização. Para isso, não serão permitidas alterações no conjunto de terminais durante reorganizações, nem reorganizações enquanto as estruturas mantidas atualizadas dinamicamente não estiverem estáveis.

As iniciativas de exclusão ou inclusão do conjunto de terminais partem espontaneamente dos terminais ou candidatos a terminal, respectivamente, mas a informação sobre o estado do sistema está no nó raiz, portanto, um terminal que deseja ser excluído do conjunto de terminais ou um nó que deseja ser incluído nesse conjunto, deve consultar o nó raiz sobre o estado do sistema antes de iniciar a rotina de exclusão ou inclusão. Este pedido é feito com o envio de mensagens *Leave* ou *Join*. Essa consulta funciona como um pedido de autorização para exclusão ou inclusão. Caso o sistema encontre-se em reorganização, o nó raiz envia a autorização apenas ao fim dessa rotina, evitando assim alterações no conjunto de terminais durante reorganizações. Assumimos que os nós que tenham solicitado exclusão ou inclusão no conjunto de terminais não farão novas solicitações enquanto a primeira não tenha sido completamente processada pelo sistema.

Serão permitidas que inclusões sejam processadas simultaneamente a outras inclusões e exclusões simultaneamente a outras exclusões. Não serão enviadas autorizações para inclusão enquanto exclusões estiverem sendo processadas. Essa última restrição evita *livelocks* [Santos 2008].

A **inclusão de terminais** é a situação onde um nó que não faça parte do conjunto de terminais é incluído nesse conjunto. Sob um ponto de vista prático, essa situação pode modelar o caso onde um nó deseja passar a fazer parte do conjunto de membros de *multicast*. Para ser incluído, o nó solicitará primeiramente autorização ao nó raiz. Ao receber esta autorização, o terminal a ser incluído inicia o crescimento de um fragmento na maneira usual do Dual Ascent.

No Dual Ascent distribuído estático, o último *convergecast*, após o "broadcast" de inclusão do nó raiz, é feito com uma mensagem *Conv(End)*. Esta mensagem já informa ao raiz a contribuição do terminal ao limite inferior. Na versão *On-Line*, a propagação desta mensagem calcula o custo acrescido a solução pela inclusão do terminal, sendo este valor também informado ao raiz. Para isto, durante a propagação do *Conv(End, Value)*, cada nó intermediário, caso ainda não participe da solução, passa a se considerar como participante e soma o custo de seu arco local de saída a *Value*. Esse valor vai sendo carregado em *convergecast* até o líder do fragmento. Além da contribuição do terminal ao limite inferior como já ocorria no Dual Ascent estático, também é enviado ao nó raiz, com a mensagem *Broad(End)*, a contribuição do terminal ao custo da solução.

A situação é ilustrada pela Figura 2, onde linhas cheias indicam arcos pertencentes a solução do problema e linhas pontilhadas arcos ainda não participantes. Na parte "a" da figura, um terminal atinge o nó raiz em um ciclo de crescimento; a parte "b" mostra a propagação da mensagem *Conv(End, Value)*; na parte "c", o terminal informa esse acréscimo na mensagem *Broad(End)*. Note que na parte "b" da figura, a primeira mensagem *Conv(End, Value)* carrega zero para *Value* pois o arco de custo 11 já pertencia à solução do problema antes da inclusão do terminal.

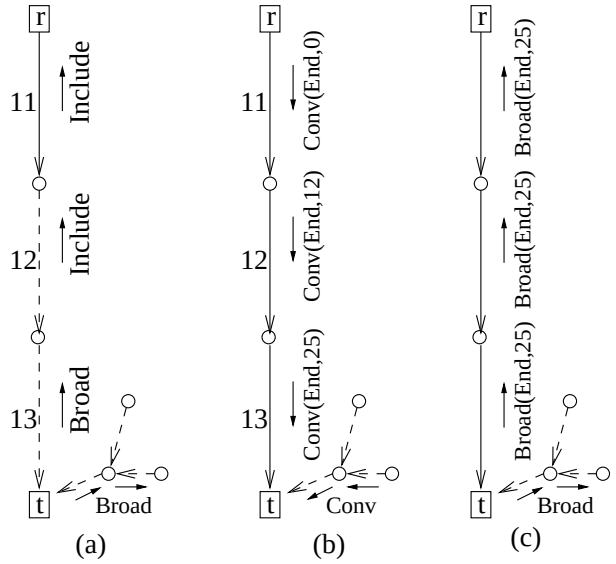

Figura 2. Inclusão de um terminal

A **exclusão de terminais** é a situação onde um nó que faça parte do conjunto de terminais é excluído desse conjunto, tornando-se um nó comum. Sob um ponto de vista prático, essa situação pode modelar o caso onde um membro deixa de ter interesse na participação em um grupo de *multicast*. Como na inclusão, um terminal que deseje ser excluído deve primeiramente solicitar autorização ao raiz.

Quando um terminal sai do conjunto de terminais, devemos subtrair sua contribuição do limite inferior global e do custo da solução. Portanto, no nó raiz, devemos armazenar separadamente a contribuição de cada terminal a cada um desses valores.

As contribuições de cada terminal para o valor do custo reduzido dos arcos locais de todos os nós também devem ser canceladas por ocasião de uma exclusão, logo, esses valores também devem ser armazenados separadamente em cada nó. Devemos cancelar as saturações feitas por um terminal que esteja sendo excluído do conjunto de terminais. Essas saturações ocorreram nos arcos da árvore de saturação associada ao terminal, portanto, ao receber a autorização para exclusão, o terminal inicia um *broadcast* em sua árvore de saturação, informando ao seu fragmento a intenção de sair do conjunto de terminais.

A exclusão de um terminal não implica necessariamente em sua saída da solução. Um ex-terminal, pode estar sendo utilizado como nó intermediário no caminho entre o nó raiz e um nó atualmente componente do conjunto de terminais, portanto, a saída da solução só ocorre quando o terminal excluído for uma folha dessa árvore. Nesse caso, outros nós que se tornem folhas e não estejam no conjunto de terminais também podem ser excluídos da solução.

Cada nó mantém a informação de quais terminais o utilizam para a conexão com o nó raiz e se considera fora da solução somente quando não é utilizado por nenhum terminal. Um nó sabe que é necessário para a conexão de um terminal quando é utilizado para propagação da mensagem *Conv(End)* no último *convergecast* de crescimento do fragmento, quando o nó raiz é incluído. Ao receber uma mensagem de *broadcast* de exclusão, o nó passa a saber qual terminal foi excluído do conjunto de terminais e pode então avaliar se faz parte ou não da solução do problema de Steiner.

Cada nó, ao ser atingido pelo *broadcast* de exclusão, cancela os valores baixados pelo terminal em seus arcos locais. Um arco saturado pode então deixar esse estado. Nesse caso, o nó envia mensagens aos líderes dos fragmentos que utilizam o arco cujo estado foi alterado em suas árvores de saturação, informando que suas árvores e grafos de saturação foram danificados e devem ser reconstruídos, por meio da mensagem *React*. A reconstrução da árvore e grafos de saturação é feita com a exclusão do nó líder do fragmento do grupo de terminais seguida de sua reinclusão. Chamaremos este processo de exclusão e posterior inclusão de **reativação** do terminal. A inclusão e a exclusão decorrentes da reativação também necessitam de autorização da raiz.

Com a estabilização do sistema, após a inclusão e/ou exclusão de terminais, o nó raiz pode comparar o limite inferior e o custo da solução para o problema de Steiner atuais. Caso a diferença esteja acima de um determinado limite, uma reorganização é disparada. Uma reorganização consiste na alteração completa ou parcial da solução atual. Isso pode ser conseguido por qualquer algoritmo distribuído para o problema de Steiner. Independente do algoritmo escolhido para a reorganização, podemos limitar seu grafo de entrada ao grafo de saturação obtido com o DA, diminuindo custos de execução e podendo, mesmo assim, esperar qualidade superior para a solução.

À medida em que sequências de inclusões vão sendo processadas, podemos esperar que os ganhos com o DA diminuam, pois terminais são incluídos a um sistema que já possui grandes grafos de saturação formados e essa situação leva a resultados piores segundo [Werneck 2001]. Entretanto, observe que as exclusões de terminais cau-

sam a exclusão e a reinclusão de outros terminais em uma região específica do grafo. As reinclusões não são autorizadas pelo nó raiz enquanto houver exclusões em processamento. Terminado o processamento de todas as exclusões, as autorizações para reinclusão serão enviadas pelo nó raiz. Os terminais reincluídos iniciam a recriação de seus grafos/árvores de saturação em paralelo, tendendo a formar grafos de saturação com tamanhos mais parecidos, como na versão estática do algoritmo. As exclusões funcionam como reorganizações dos grafos/árvores de saturação, diminuindo o efeito indesejado das inclusões de terminais a partir de um sistema com grandes grafos de saturação já formados.

6. Exemplos

Na Figura 3 exibimos uma possível execução da heurística para o problema *On-Line*. Na parte "a", um terminal t solicita autorização para inclusão no conjunto de terminais enviando a r uma mensagem *Join(Req)*. Como o sistema está estável, r autoriza a inclusão respondendo com a mensagem *Join(Ok)*. O terminal t inicia o crescimento de um fragmento e o processo continua como no DA estático, até que o terminal raiz seja incluído no fragmento. As reticências da figura representam a árvore de saturação que vai sendo criada durante o crescimento do fragmento, cujos nós são omitidos. O terminal t informa então no último *broadcast* o limite inferior parcial (pb) e o custo adicionado à árvore de distribuição (pt).

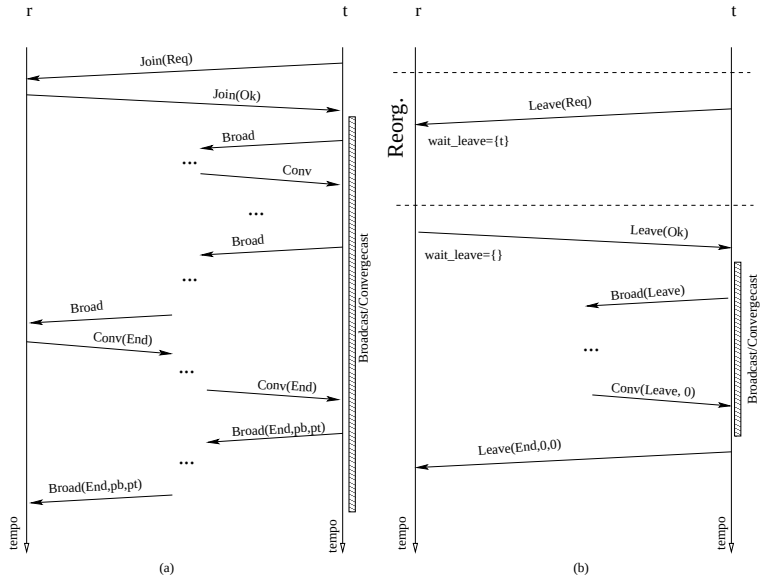

Figura 3. Possível execução para o problema *On-Line* - inclusão, reorganização e exclusão.

Na parte "b" da Figura 3, observamos a exclusão de um terminal. O terminal t pede autorização para exclusão enviando a mensagem *Leave(Req)*. Admitiremos que nesse ponto o sistema encontre-se em processo de reorganização. Nesse caso r registra que t aguarda por uma autorização para exclusão e envia a mensagem *Leave(Ok)* apenas após o término da reorganização. Recebendo o *Leave(Ok)*, t informa a seu fragmento a exclusão com um *broadcast* em sua árvore de saturação. O fragmento informa em *convergecast* que não foram reativados outros terminais (mensagem *Conv(Leave, 0)*). Neste

ponto t pode informar a r por meio da mensagem *Leave(End, 0, 0)* que terminou o processo de exclusão. Os dois zeros significam que não foram nem enviadas nem recebidas mensagens *React*.

7. Análise do Algoritmo

Todo o processamento de inclusão e exclusão de terminais pode ser feito sem qualquer perturbação à solução que esteja sendo utilizada para distribuição de mensagens em *multicast* no momento. Somente a reorganização altera rotas existentes para terminais. Ainda assim, pode-se continuar utilizando a solução antiga enquanto se calcula a nova, resumindo a perturbação ao mínimo, durante a troca de árvores.

O ARIES parte do pressuposto de que regiões da solução que tenham sofrido muitas alterações serão necessariamente regiões onde a qualidade da solução obtida por heurísticas se afastará do ótimo. Essa presunção é razoável, mas podem existir casos em que muitas alterações ocorram e as inclusões por caminho mínimo continuem gerando árvores de custo baixo, próximo ao ótimo ou, de forma inversa, uma única alteração gere uma árvore com custo muito superior ao ótimo. Neste trabalho disparamos reorganizações somente quando o custo da solução torna-se muito alto, se comparado ao limite inferior fornecido pelo Dual Ascent.

Outra diferença importante entre o ARIES e nossa abordagem é que o ARIES pressupõe o prévio conhecimento dos caminhos mínimos entre todos os pares de nós, sob a justificativa de que essa informação já consta das tabelas de roteamento nas redes práticas. O Dual Ascent *On-Line* calcula os caminhos mínimos durante sua execução, o que permite sua utilização mesmo quando a métrica cuja minimização é desejada seja diferente da utilizada pelos roteadores.

A seguir, analisamos a complexidade de pior caso para as operações de inclusão e exclusão de terminais, quanto ao número de mensagens enviadas e ao tempo, dado pelo tamanho da maior cadeia de mensagens que sustentem relação de causalidade do tipo enviar uma mensagem em decorrência do recebimento de outra.

Como veremos a seguir, reorganizações mais oportunas e a qualidade de solução superior do DA *On-Line* cobram seu custo. O DA é mais caro computacionalmente do que os algoritmos mais simples, baseados nos caminhos mínimos como o EBA e o ARIES. Esperamos que, na prática, parte desse custo nas operações de inclusão e exclusão de terminais seja compensada por um número menor de reorganizações desnecessárias e uma reorganização mais rápida por atuar sobre o grafo de saturação.

Considerando o procedimento de inclusão de um nó, quanto ao número de mensagens, no pior caso um terminal incluirá todos os nós do grafo que enviará mensagens *Check* para todos os demais. Portanto, a complexidade de mensagens $O(|V|^2)$. Para o tempo, o pior caso ocorrerá quando os nós são incluídos um a um formando um caminho até o nó raiz ser incluído. Nesse caso o número de mensagens da maior cadeia de eventos com relação de causalidade será dado pelo somatório $2 + 4 + 6 + 8 + ... + (V - 2)$, que é também $O(|V|^2)$.

A título de comparação, se considerarmos que os caminhos mínimos já estão calculados, a complexidade de pior caso para uma operação de inclusão no ARIES é $O(|V|)$ para tempo e mensagens. As complexidades para o cálculo dos caminhos mínimos são

$O(|E|.|V|)$ para mensagens e $O(|V|)$ para tempo. No caso *On-Line* não é justo que simplesmente acrescentemos este custo à execução do algoritmo, já que ele é cobrado apenas uma vez para todas as operações e, portanto, seu peso é tanto menor quanto maior for o número de operações.

Considerando a exclusão de um terminal, no pior caso, esta ocasionará a reativação de todos os demais terminais, elevando as complexidades de tempo e mensagens aos níveis do DA estático: $O(|T|.|V|^2)$. A complexidade em tempo e mensagem para exclusão de terminais no ARIES é $O(|V|)$.

8. Resultados Experimentais

Implementamos o Dual Ascent dinâmico em ANSI C e MPICH2 e executamos em um único computador Pentium Dual Core com 1 GByte de memória e MPICH2 versão 1.0.7 e com as instâncias simuladoras da Internet geradas com o sistema BRITE (*Boston University Representative Internet Topology Generator*) [Medina et al. 2001]. Dentre os vários modelos para geração de grafos disponíveis no BRITE, escolhemos o "Generalized Linear Preference" [Bu e Towsley 2002] para gerar os grafos da série GLP. Esse modelo utiliza leis de potência para determinação do grau dos nós e também características *Small World*. Utilizamos os parâmetros padrões para a geração de redes para simulação da Internet em nível de roteador. As instâncias geradas pelo BRITE não são orientadas. As instâncias da série SW foram geradas segundo o modelo sugerido para grafos simuladores da Internet no nível de Sistemas Autônomos (*AS - Autonomous Systems*) em [Jin e Bestavros 2006].

Para cada grafo, foi gerada uma sequência de 15 inserções de terminais escolhidos aleatoriamente no conjunto de nós. Os 15 nós escolhidos constituem o conjunto inicial de terminais e foram incluídos todos no instante 0 da simulação, utilizando a operação de inserção dinâmica, descrita anteriormente. A seguir, foram geradas sequências de operações de inclusão ou exclusão. Para cada operação, sorteou-se inicialmente se seria uma operação de inclusão ou exclusão. Para o caso de uma operação de exclusão, sorteou-se um terminal e para uma operação de inclusão, um nó não terminal. Foram utilizadas funções de distribuição de probabilidade homogênea em ambos os sorteios.

Para cada grafo, geraram-se duas sequências de operações. A primeira com 16 operações, separadas por tempos suficientes para que o sistema se estabilizasse e permitisse uma reorganização, caso necessário. Na segunda, foram geradas 60 operações, agrupadas de 6 em 6. As operações de um grupo são disparadas em paralelo. Entre cada grupo de 6 operações, guardou-se um tempo suficiente para a estabilização do sistema. Foi utilizado o limite de 40% de distância tolerada entre o custo da solução e o limite inferior obtido com o Dual Ascent. Ultrapassado esse limite, uma reorganização seria disparada.

Com a finalidade de comparação, executamos o mesmo algoritmo para inserção e exclusão de nós, porém implementando os contadores do ARIES como critério para disparo de reorganizações. Neste caso, desprezamos o limite inferior e executamos reorganizações quando um contador ultrapassava o limite de 6, valor indicado em [Bauer e Varma 1997] como ideal. Utilizamos a mesma rotina para reorganização em ambos os casos: o Prim-SPH descrito em [Novak et al. 2001a], principalmente por este algoritmo ser adaptável para o problema direcionado e apresentar excelentes resultados práticos.

Instância		Número de Reorganizações		Ganho Médio por Reorganização (%)		Distância Média para Ótimo (%)	
		DA	ARIES	DA	ARIES	DA	ARIES
GLP (1 oper. por vez)	1	**0**	8	-	1,4	11,7	**6,1**
	2	**2**	5	**32,7**	12,4	16,2	**15,2**
	3	**0**	6	-	0,3	3,9	**3,4**
	4	**1**	7	**38,0**	7,3	6,3	**5,0**
	5	**4**	6	**24,7**	11,3	**12,0**	21,5
GLP (6 oper. por vez)	1	**0**	8	-	2,4	9,3	**6,7**
	2	**4**	5	**27,2**	20,1	**30,4**	46,3
	3	**0**	6	-	0,3	0,7	**0,6**
	4	**2**	6	**25,2**	14,0	34,6	**21,8**
	5	**4**	7	**28,8**	8,6	**11,7**	22,8
SW (1 oper. por vez)	1	**5**	9	**32,8**	26,6	**33,3**	41,1
	2	**4**	5	**27,5**	23,6	**15,7**	18,0
	3	**3**	6	**41,6**	20,7	**13,5**	27,8
	4	**1**	5	**20,6**	2,3	**20,0**	24,3
	5	**0**	6	-	2,7	7,4	**3,6**
SW (6 oper. por vez)	1	**5**	7	**42,3**	24,9	**12,0**	21,9
	2	**2**	7	**24,7**	6,4	24,7	**10,7**
	3	**5**	8	**16,6**	12,5	24,5	**19,6**
	4	**1**	8	**19,9**	6,0	15,5	**7,9**
	5	**0**	8	-	1,7	4,5	**3,1**
MÉDIA		2,2	6,7	28,8	10,3	15,4	16,4

Tabela 1. Comparação entre o critério para disparo de reorganizações baseado em contadores do ARIES e o critério baseado no limite inferior oferecido pelo Dual Ascent (DA).

Como podemos observar pela Tabela 1, utilizando o limite inferior do Dual Ascent como critério para disparo de reorganizações, reduziu-se o número de disparos a praticamente um terço, mantendo-se a distância média entre a solução heurística e ótima após cada reorganização em um patamar ligeiramente inferior. Isso foi possível porque realizando as reorganizações no momento em que realmente são necessárias, conseguindo um ganho médio por reorganização muito superior: 28,4% utilizando o Dual Ascent contra 10,3% utilizando o método do ARIES.

Nas figuras 4 e 5 podemos observar curvas com o ótimo (calculado com o algoritmo *branch-and-bound* descrito em [Poggi de Aragão et al. 2001]), limite inferior e soluções heurísticas durante um período de tempo em que o conjunto de terminais sofre inclusões ou exclusões. Reorganizações são disparadas com base no limite inferior ou nos contadores do ARIES. Círculos próximos ao eixo das ordenadas indicam reorganizações disparadas pelo método do ARIES e quadrados indicam reorganizações disparadas pelo método do limite inferior (DA). Em ambos os casos, as reorganizações foram feitas utilizando o algoritmo Prim-SPH [Novak et al. 2001a].

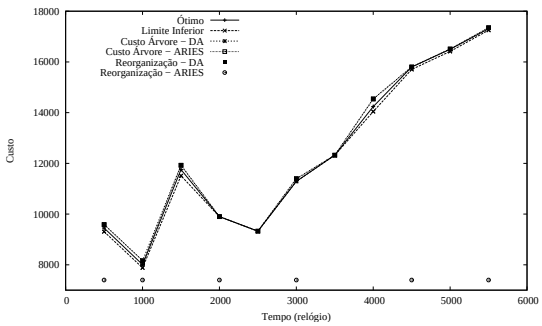

Figura 4. Instância GLP-3 com operações em grupos de seis.

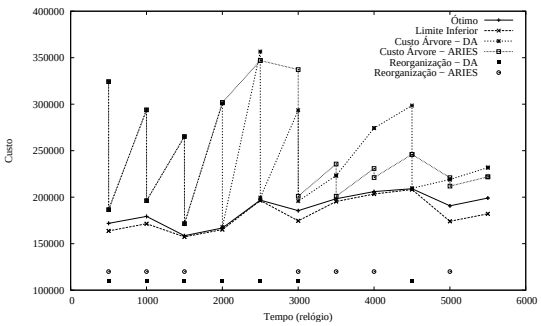

Figura 5. Instância SW-1 com operações em grupos de seis.

Para algumas instâncias, o método guloso produz resultados muito bons, como

na instância GLP-3, com operações realizadas em grupos de seis. Podemos observar no gráfico da Figura 4, que solução ótima, limite inferior e soluções heurísticas apresentam diferenças muito pequenas, sendo as reorganizações perfeitamente dispensáveis.

Para outras instâncias, no entanto, as soluções obtidas pelo método guloso não são tão boas. Este é o caso da instância SW-1 com operações feitas em grupos de seis. Como podemos observar pela Figura 5, as operações degradam muito rapidamente a qualidade da solução do problema. No início da simulação, ambas as estratégias disparam reorganizações que aproximam a qualidade da árvore de distribuição do ótimo. No entanto, a partir do instante 1.744, o método dos contadores não dispara reorganizações necessárias e o sistema fica com uma solução pobre por um grande período de tempo.

As duas instâncias escolhidas para a construção dos gráficos das figuras 4 e 5 são casos extremos. As demais instâncias apresentaram resultados intermediários nos testes práticos.

9. Conclusão

Foi apresentada uma possível adaptação do Dual Ascent para o problema *On-Line*, onde nós podem ser incluídos e excluídos do conjunto de terminais com o passar do tempo, variação de grande interesse prático do Problema de Steiner em Grafos.

Exploramos a utilização do limite inferior fornecido pelo algoritmo como critério para realização de reconstruções parciais ou totais da solução quando o conjunto de terminais é alterado. Mostramos que este critério é mais eficiente do que os baseados em número de alterações sofridas em determinadas regiões do grafo, pois nem sempre o simples fato de haver ocorrido muitas alterações significa que a qualidade da solução tenha diminuído, justificando o custo da reconstrução e, muitas vezes, uma simples alteração pode degradar bastante a qualidade da solução, justificando a reconstrução. Como o limite inferior oferecido pelo Dual Ascent apresenta qualidade muito boa, apresentando valores muito próximos ao ótimo, sua utilização leva a reconstruções em situações muito mais oportunas do que o critério baseado em número de alterações.

Referências

Adelstein, F., Richard III, G. G., e Schwiebert, L. (2003). Distributed multicast tree generation with dynamic group membership. *Computer communications*, 26(10):1105–1128.

Bauer, F. e Varma, A. (1996). Distributed algorithms for multicast path setup in data networks. *IEEE/ACM Transactions on networking*, 4(2):181–191.

Bauer, F. e Varma, A. (1997). Aries: a rearrangeable inexpensive edge-based on-line steiner algorithm. *IEEE Journal on selected areas in communications*, 15(3):382–397.

Bu, T. e Towsley, D. F. (2002). On distinguishing between internet power law topology generators. In *Proceedings of IEEE annual joint conference of the IEEE computer and communications societies INFOCOM'02*, volume 2, pages 638–647.

di Fatta, G. e Re, G. L. (1998). Efficient tree construction for the multicast problem. In *Proceedings of the international telecommunications symposium ITS'98*, volume 2, pages 632–637.

Drummond, L., Santos, M., e Uchoa, E. (2008). A distributed dual ascent algorithm for Steiner problems in multicast routing. *Networks*. A ser publicado, disponível em http://www.ic.uff.br/∼lucia/dual-ascent.pdf.

Gatani, L., Re, G. L., e Gaglio, S. (2006). An efficient distributed algorithm for generating and updating multicast trees. *Parallel computing*, 32(11–12):777–793.

Imase, M. e Waxman, B. (1991). Dynamic steiner tree problems. *SIAM Journal of discrete mathematics*, 4(3):369–384.

Jin, S. e Bestavros, A. (2006). Small-world characteristics of internet topologies and implications on multicast scaling. *Computer networks: the international journal of computer and telecommunications networking*, 50(5):648–666.

Medina, A., Lakhina, A., Matta, I., e Byers, J. (2001). Brite: An approach to universal topology generation. In *Proceedings of the international symposium on modeling, analysis and simulation of computer and telecommunications systems MASCOTS'01*, pages 346–353.

Novak, R., Rugelj, J., e Kundus, G. (2001a). A note on distributed multicast routing in point-to-point networks. *Computers and operations research*, 28(12):1149–1164.

Novak, R., Rugelj, J., e Kundus, G. (2001b). Steiner tree based distributed multicast routing in networks. In *Steiner trees in industries, Combinatorial optimization*, volume 11, pages 327–351. Kluwer Academic Publishers, Dordrecht,.

Oliveira, C. A. S. e Pardalos, P. M. (2005). A survey of combinatorial optimization problems in multicast routing. *Computers and operations research*, 32(8):1953–1981.

Poggi de Aragão, M., Uchoa, E., e Werneck, R. (2001). Dual heuristics on the exact solution of large steiner problems. In *Eletronic notes in discrete mathematics GRACO'01*, volume 7.

Polzin, T. e Vahdati, S. (2001). Improved algorithms for the steiner problem in networks. *Discrete applied mathematics*, 112(1–3):263–300.

Santos, M. (2008). *Algoritmo Dual Ascent Distribuído Aplicado ao Problema de Steiner em Grafos*. PhD thesis, Universidade Federal Fluminense.

Santos, M., Drummond, L. M. A., e Uchoa, E. (2007). Distributed dual ascent algorithm for steiner problems in networks. In *Anais do simpósio brasileiro de redes de computadores e sistemas distribuídos SBRC'07*, pages 381–396.

Waxman, B. (1988). Routing of multipoint connections. *IEEE Journal on selected areas in communications*, 6(9):1617–1622.

Werneck, R. F. F. (2001). Problema de steiner em grafos: algoritmos primais, duais e exatos. Master's thesis, Pontifícia Universidade Católica do Rio de Janeiro.

Wong, R. (1984). A dual ascent approach for steiner tree problems on a directed graph. *Mathematical programming*, 28(3):271–287.