# Um Novo Algoritmo Dual-Ascent Distribuído para Multicast Dinâmico

**Rafaelli de Carvalho Coutinho, Ubiratam Carvalho de Paula Junior, Diego Passos, Célio Albuquerque, Lúcia M. A. Drummond**

Instituto de Computação – Universidade Federal Fluminense

{rcoutinho,upaula,dpassos,celio,lucia}@ic.uff.br

***Abstract.*** *Dynamic Multicast can be modeled by the On-Line Steiner Problem, which consists in finding a minimum cost tree that reaches a subset of nodes of a graph, called terminals, from a root node $r$, where the terminals set can change over time, with nodes entering or leaving this set. Most of the known algorithms for this problem apply simple approaches for inclusion and exclusion operations, and known static algorithms to reorganize the tree periodically. This paper presents a new heuristic for the On-Line Steiner Problem, based on the previously proposed distributed Dual-Ascent algorithm, that triggers reorganizations more opportunistically than the main known heuristics for the problem. In order to analyze the proposed algorithm, another heuristic frequently employed in the literature as a baseline for comparison with other works, Aries, was also implemented. Experimental results showed that the new On-Line distributed version of Dual-Ascent outperformed Aries in terms of quality of solutions and computational effort.*

***Resumo.*** *Multicast dinâmico pode ser modelado pelo Problema de Steiner On-Line, que consiste em encontrar uma árvore de custo mínimo que atinja um subconjunto de nós, ditos terminais, de um grafo a partir de um nó raiz $r$, onde o conjunto de terminais sofre alterações com o passar do tempo, com nós podendo entrar ou sair deste conjunto. A maioria dos algoritmos conhecidos para este problema usa técnicas simples para inclusão e exclusão de terminais, utilizando periodicamente algoritmos conhecidos para o problema estático para reorganizar a árvore. Este artigo apresenta uma nova heurística para o Problema de Steiner On-Line, baseada no algoritmo Dual-Ascent distribuído previamente proposto, que dispara reorganizações de forma mais oportuna do que os principais algoritmos conhecidos. A fim de analisar o algoritmo proposto, uma outra heurística frequentemente empregada na literatura como base de comparação com outros trabalhos, Aries, foi também implementada. Resultados experimentais mostraram que a nova versão On-Line distribuída do Dual-Ascent superou os resultados do Aries em termos de qualidade de solução e esforço computacional.*

## 1. Introdução

Algumas aplicações atuais requerem distribuição de dados a um subconjunto específico de nós de uma rede, como por exemplo distribuição de vídeo sob demanda e teleconferência.

Este tipo de aplicação utiliza *broadcast seletivo* ou *multicast*, que é frequentemente modelado como o Problema de Steiner em Grafos Direcionados (*Steiner Problem in Directed Graphs - SPDG*) [Novak et al. 2001, Oliveira e Pardalos 2005].

Em situações práticas das aplicações exemplificadas, terminais podem entrar e sair do grupo *multicast*. Isto é ilustrado com a solicitação do mesmo vídeo sob demanda ou participação na teleconferência por novos usuários, assim como o cancelamento de ambas as aplicações por usuários participantes. Neste caso, o problema de Steiner é chamado de Problema de Steiner On-Line e é definido da seguinte forma. Sejam um 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 $a \in A$ e um conjunto $T_0 \subseteq V$ de terminais (conjunto inicial de terminais). Seja ainda $T = (T_1, T_2, ..., T_n)$ uma sequência de conjuntos de terminais determinada por uma sequência de operações $(o_1, o_2, ..., o_n)$ onde cada operação $o_i$ pode ser uma inclusão ($T_i = T_{(i-1)} \cup \{v\}, v \in V, v \notin T_{(i-1)}$) ou uma exclusão ($T_i = T_{(i-1)} - \{t\}, t \in T_{(i-1)}$). Encontrar uma sequência $S = (G'_0, G'_1, G'_2, ..., G'_n)$ de sub-árvores de $G_d$, 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.

É possível aproveitar uma árvore existente, podando os ramos desnecessários após exclusões de terminais ou unindo-a a um novo terminal através do caminho mínimo entre eles. Entretanto, após várias operações de inclusão e exclusão, a qualidade da árvore tende a se distanciar do ótimo.

Uma reorganização da árvore pode ser utilizada visando uma aproximação do custo ótimo. No entanto, este pode ser um processo computacionalmente caro, nem sempre compensado pelo ganho obtido na solução. Para este problema, sistemas tentam disparar reorganizações apenas periodicamente, sendo que a maioria dispara reorganizações após um determinado número de alterações indicado como limite (Aries [Bauer e Varma 1997], EBA [Imase e Waxman 1991]). O número de alterações nem sempre implica a queda do custo da solução para o problema de Steiner. É possível continuar com boas soluções após muitas alterações ou comprometer seriamente a qualidade da solução após uma única alteração.

Neste trabalho é apresentada uma nova adaptação do algoritmo Dual-Ascent (DA) [Drummond et al. 2009, Santos et al. 2007, Santos et al. 2009] para a versão *On-Line* do Problema de Steiner. Esta adaptação, denominada DA_Mod, tornou o algoritmo competitivo em relação a alguns algoritmos clássicos para este problema. O DA_Mod oferece como saída, além do custo da solução, um limite inferior, que é uma boa aproximação para o ótimo do problema de Steiner. Este limite inferior é utilizado para avaliar a qualidade da solução atual. Quando a distância entre o custo da solução e o limite inferior supera um determinado valor, o algoritmo dispara uma reorganização.

A fim de analisar o algoritmo proposto, uma outra heurística, que é frequentemente empregada na literatura como base de comparação com outros trabalhos, Aries, foi também implementada.

A versão anterior do algoritmo Dual-Ascent não era competitiva com o Aries em termos de tempo de execução. Durante as operações de exclusão, terminais descendentes do nó excluído na árvore eram também removidos para uma posterior re-inserção (processo denominado de reativação). Este comportamento prejudica o desempenho do DA.

Por este motivo, o novo algoritmo difere do anterior basicamente em relação às exclusões. Neste, as exclusões não geram mais exclusões e consequentes reativações de outros terminais. Com isso, o novo algoritmo gasta menos tempo e mensagens, e pode ser comparado ao Aries. Os resultados do DA_Mod são melhores em termos de tempo e número de mensagens e as reorganizações são mais oportunas, melhorando, em média, a qualidade das soluções (em relação ao DA e também ao Aries).

O texto está organizado da seguinte forma. Na Seção 2, o algoritmo Dual-Ascent Distribuído é brevemente descrito . Na Seção 3, alguns trabalhos relacionados são comentados, em especial o Aries. Na Seção 4, é descrito o Novo Algoritmo Dual-Ascent On-Line, destacando o funcionamento da nova operação de exclusão. Na Seção 5, são feitas as análises de complexidades para o algoritmo proposto, DA_Mod, e para o trabalho relacionado mais influente, o Aries. Na Seção 6, são expostos os resultados experimentais para os algoritmos analisados na seção anterior. Finalmente, na Seção 7, uma conclusão é apresentada, resumindo as contribuições deste trabalho.

## 2. O Dual-Ascent Distribuído

A versão distribuída do Dual-Ascent foi apresentada em [Drummond et al. 2009, Santos et al. 2009]. Nesta, foram considerados um grafo direcionado $G_d = (V, A)$, um conjunto $T \subseteq V$ de terminais e uma raiz $r \in V$, onde cada arco $a$ possui um *custo reduzido* não negativo $\bar{c}_a$ associado, inicialmente com valor igual ao custo original do arco (pois este ainda não é membro da árvore de *multicast*). Durante a execução do algoritmo estes custos reduzidos decrescem até atingirem zero, quando são chamados de *saturados*. Estes arcos formam um *grafo de saturação* $G_S = (V, A_r(\bar{c}))$ onde $A_r(\bar{c}) = \{a \in A : \bar{c}_a = 0\}$.

No DA distribuído, todos os terminais iniciam o crescimento de partes da árvore de *multicast*, chamadas fragmentos, em paralelo. Estes fragmentos crescem em ciclos, que são constituídos da difusão no fragmento do valor do custo do menor arco incidente, a ser subtraído dos custos reduzidos de todos os arcos incidentes a estes fragmentos. 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* (mensagens de muitos nós origem para um mesmo destino), de tal forma que o terminal líder do fragmento que iniciou o crescimento saiba o novo valor do custo do menor arco incidente 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 . Neste momento, o fragmento cessa o seu crescimento, informando ao raiz seu limite inferior parcial. Quando o raiz recebe o limite inferior de todos os fragmentos, ele calcula 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.

Na Figura 1 é ilustrado um ciclo de crescimento de fragmento típico do algoritmo distribuído. Terminais são representados por quadrados enquanto não terminais pertencentes à arvore são círculos e os demais nós são triângulos. Na Figura 1(a) são enviadas as mensagens de difusão do valor do custo do menor arco incidente (em *broadcast*) obtido no ciclo anterior. Ao receber estas mensagens, o custo reduzido do arco é diminuido do valor recebido. Desta forma, saturações podem ocorrer e quando ocorrem, o nó em

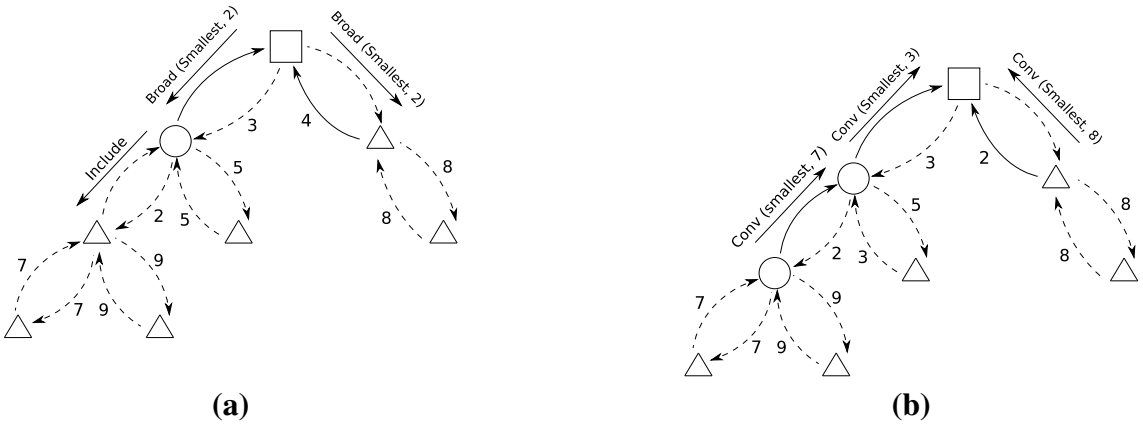


**Figura 1. Crescimento de Fragmentos.**

questão envia uma mensagem *include* pelo arco que foi saturado. A Figura 1(b) ilustra o *convergecast* dos menores valores das arestas incidentes locais. No próximo ciclo, o terminal líder irá difundir o menor valor recebido. Este processo se repete até que o terminal raiz seja incluído.

Os nós recém incluídos em um fragmento devem informar o novo custo do menor arco incidente. No entanto, o vizinho associado a este arco pode já pertencer ao fragmento e, portanto, o arco não é incidente ao fragmento. Para evitar esta situação, 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*. Uma mensagem *Ack* informa sobre a pertinência ou não do emissor no fragmento. Com base na informação recebida, o nó incluído sabe quais arcos devem ser considerados incidentes.

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

O algoritmo distribuído constrói sobre o grafo de saturação uma árvore de saturação enraizada em cada terminal. Tal árvore é utilizada nas operações de *broadcast* e *convergecast*.

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

A versão dinâmica do problema de Steiner pode ser vista como uma sequência de instâncias do problema estático. Com isso, é possível resolvê-lo através execuções de algoritmos para o problema estático a cada operação de inclusão ou exclusão realizada. Entretanto, essa prática leva a custos elevados e desnecessários, uma vez que informações obtidas na solução anterior podem ser aproveitadas para encontrar a solução atual.

A primeira abordagem para este problema é a totalmente gulosa, proposta por [Waxman 1988]. Um novo terminal é incluído através do caminho mais curto entre ele e qualquer nó pertencente a árvore atual. A exclusão também é bastante simples, bastando verificar se o nó a ser excluído tem filhos na árvore. Caso tenha, o nó é mantido

na árvore. Caso contrário, o nó é removido e esta mesma avaliação é realizada para o nó pai.

O EBA (*Edge-Bounded Algorithm*) [Imase e Waxman 1991] é uma heurística mais sofisticada. Inclusões e exclusões são efetuadas, a princípio, da mesma forma que na heurística puramente gulosa. No entanto, após cada inclusão, o EBA tenta reduzir o custo da árvore através de uma simples verificação. Para cada terminal já existente na árvore, o EBA analisa se é vantajoso reconectá-lo através de um dos novos nós, resultantes da última operação gulosa de inserção.

A heurística VTDM (*Virtual Trunk Dynamic Multicast*) [Lin e Lai 1998] estima a probabilidade de cada nó da rede ser utilizado na conexão entre dois outros nós quaisquer. Os nós com maior probabilidade são separados em um conjunto denominado *Virtual Trunk*. A heurística mantém também uma árvore com todos os componentes deste conjunto. A inclusão de novos nós é realizada através do seu caminho mínimo até a árvore de nós do *Virtual Trunk*. O processo de exclusão é idêntico ao da heurística gulosa. Embora o VTDM possa obter bons resultados em redes relativamente pequenas, a necessidade de concentração dos dados de probabilidades de cada nó pode inviabilizar seu uso em redes grandes.

A heurística apresentada por [Gatani et al. 2006] utiliza o ADH (*Average Distance Heuristic*) para construir uma solução inicial. À medida que inserções e remoções são realizadas em regiões específicas da árvore, a heurística avalia (através do número de modificações) se deve realizar uma reorganização dos nós da região. A reorganização de uma região, baseada na *stirring technique* [di Fatta e Re 1998], consiste em avaliar para cada terminal se este pode ser conectado com um custo menor a um novo *grafting point*, que é um terminal ancestral de grau maior ou igual a 2 na árvore. Para utilizar o conceito de nó ancestral, a heurística supõe a existência de um terminal especial denominado *source*.

### 3.1. A Heurística Aries

Uma das heurísticas distribuídas mais influente nesta área é, sem dúvida, o Aries, proposto em [Bauer e Varma 1997]. Por este motivo, ela foi escolhida como base de comparação para a avaliação de desempenho da heurística proposta neste trabalho.

Em sua proposta original, o Aries trabalha com grafos não direcionados (diferentemente do DA, por exemplo). Seu funcionamento básico é relativamente simples. Ele utiliza a mesma estratégia de inclusão e exclusão de nós adotada pelo algoritmo guloso. Ou seja, nós são adicionados através do caminho mais curto até a árvore e removidos apenas se forem folhas. A grande diferença do Aries para a heurística gulosa está na funcionalidade de reorganização da solução. O algoritmo monitora a qualidade da solução a medida que novas operações são realizadas e, caso julgue apropriado, um procedimento de re-conexão da árvore é disparado.

O Aries mantém para cada nó da rede um estado. Sempre que um nó se torna um terminal, seu estado passa a ser *modificado*. O estado *modificado* é atribuído também a nós removidos do grupo *multicast* que não são retirados da árvore por não serem folhas. O monitoramento da qualidade da solução é realizado justamente através do número de nós *modificados*. Quando este número ultrapassa um determinado limiar (parâmetro do algoritmo), o Aries entende que muitas inserções e remoções foram realizadas de ma-

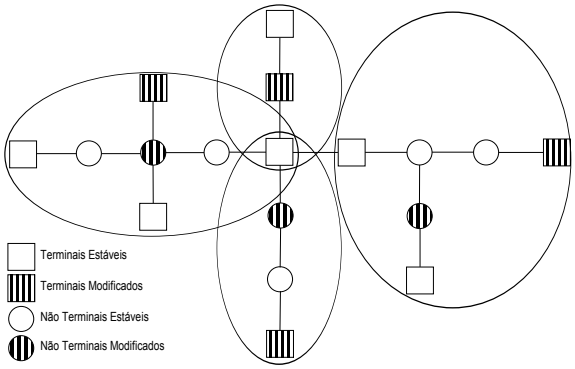


**Figura 2. Exemplo de uma possível árvore em um determinado momento da execução da heurística Aries.**

neira gulosa e, por isso, a solução tem grandes chances de poder ser melhorada. Nesta situação, o Aries desconecta a árvore atual e executa uma heurística para a versão estática do problema de Steiner, o algoritmo K-SPH [Bauer e Varma 1996].

O resultado deste procedimento é uma nova árvore, com custo possivelmente mais baixo que o da árvore original, contendo todos os terminais atuais. Após a execução do K-SPH, todos os terminais passam a ter o estado *estável*, incluindo os que tinham estado *modificado*. Os demais nós da rede cujo estado era anteriormente *modificado* passam a ser apenas não terminais.

O algoritmo K-SPH é uma heurística para versão estática do problema de Steiner. Se comparada aos procedimentos gulosos de inserção e remoção, a execução do K-SPH é bastante custosa, mas apresenta soluções melhores. Para evitar que o custo das reorganizações influenciasse de maneira contundente na sua complexidade total, o Aries tenta realizar apenas reorganizações localizadas. A ideia é dividir a árvore em regiões. Caso o número de nós de estado *modificado* dentro de uma região ultrapasse o limiar estabelecido, apenas as arestas da árvore internas à região são desconectadas. Assim, o processamento do K-SPH já é iniciado em uma fase mais adiantada, tendo sua complexidade reduzida.

A Figura 2 ilustra um exemplo da divisão feita pelo Aries da árvore em regiões. Na ilustração, são mostrados apenas os nós que compõem a árvore. Eles são divididos em 4 conjuntos: terminais cujo estado é *estável*, terminais cujo estado é *modificado*, não terminais cujo estado é *modificado* (nós recentemente removidos de maneira gulosa), além dos demais não terminais (nós auxiliares na formação da árvore). As regiões são demarcadas pelas elipses envolvendo os nós.

As regiões da árvore são delimitadas pelos terminais cujo estado atual é *estável*. Dois nós da árvore pertencem à mesma região se existe um caminho na árvore entre eles que não passe por algum terminal *estável*. Qualquer não terminal da árvore ou nó cujo estado atual seja *modificado* pertence a uma única região. Os terminais estáveis, por outro lado, podem pertencer a tantas regiões quanto seu grau dentro da árvore. São justamente estes nós que realizam o monitoramento da quantidade de nós modificados em cada região. Quando um nó da árvore muda seu estado de ou para *modificado*, ele envia uma mensagem para os terminais estáveis da sua região solicitando uma alteração nos contadores.

É interessante notar que a remoção de um terminal estável pode causar alterações na disposição das regiões. Por exemplo, na árvore apresentada na Figura 2, há um terminal *estável* pertencente a três regiões distintas. Se este nó fosse removido do grupo *multicast*, ele não seria retirado da árvore por não ser uma folha (outros nós dependem dele para se conectar à árvore). Neste caso, seu estado passaria a ser *modificado*, o que transformaria as três regiões em uma única. Quando este tipo de evento ocorre, os terminais estáveis precisam detectar a existência de possíveis novos nós modificados em suas regiões.

Uma limitação importante do Aries é a impossibilidade de execução de operações concorrentes. Se operações forem disparadas de forma concorrente, é possível que ocorram inconsistências entre os valores dos contadores armazenados pelos vários nós estáveis de uma região. Neste caso, um conjunto de nós da região pode tentar disparar uma reorganização, enquanto os demais não estão cientes desta execução.

## 4. Novo Algoritmo Dual-Ascent para o Problema de Steiner *On-Line*

O novo algoritmo Dual-Ascent para o Problema de Steiner *On-Line*, DA_Mod, utiliza as mesmas rotinas da versão anterior [Santos et al. 2009] para as inclusões e para as reorganizações, apresentando diferença apenas nas operações de exclusão. Por este motivo, nesta seção apresentamos apenas os detalhes da operação de exclusão.

### 4.1. Novo Processo de Exclusão de Terminais

A operação de exclusão representa, em uma situação prática, a saída de um terminal do conjunto de membros de *multicast*. Nesta subseção, a operação de exclusão descrita em [Santos et al. 2009] será comparada com a nova, proposta neste artigo.

No DA, esta operação começa com o pedido de autorização ao terminal raiz. Ao receber esta autorização, o nó a ser excluído deve cancelar as saturações realizadas por ele. Para isto, este envia uma mensagem em *broadcast* informando ao seu fragmento a intenção de sair do conjunto de terminais. É válido observar que um terminal excluído pode continuar na árvore de *multicast* como um nó comum intermediário no caminho entre o raiz e um outro terminal.

Cada nó, ao ser atingido pelo *broadcast* de exclusão, cancela os valores reduzidos pelo terminal sainte em seus arcos locais. Um arco saturado pode então deixar de ter este estado. Nesse caso, o nó envia mensagens *React* aos líderes dos fragmentos (os terminais que iniciam o crescimento de seus 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. A reconstrução da árvore e grafos de saturação é feita com a exclusão do terminal líder do fragmento, seguida de sua re-inserção (reativação). A inclusão e a exclusão decorrentes da reativação também necessitam de autorização do nó raiz.

A nova operação de exclusão do DA_Mod também começa com o pedido de autorização ao terminal raiz. Entretanto, ao receber esta autorização, além de enviar uma mensagem em *broadcast* informando ao seu fragmento a intenção de sair do conjunto de terminais, o nó sainte deve achar um outro terminal para ser seu substituto nos custos reduzidos dos arcos saturados por ele. Desta forma, o terminal que será excluído só irá cancelar as saturações realizadas por ele nos arcos que não são utilizados por nenhum outro terminal.

Cada nó, ao ser atingido pelo *broadcast* de exclusão, verifica se o terminal sainte diminuiu o custo reduzido de algum dos seus arcos incidentes. Em caso afirmativo, este nó deverá encontrar um terminal que também utilize este arco para ser seu substituto. Para este terminal escolhido, é enviada a mensagem *Subst*. Um terminal ao receber esta mensagem, atualiza seu limite inferior e custo da solução parciais. Em seguida, este envia ao terminal raiz uma mensagem *Atual* informando estes valores atualizados.

A nova operação de exclusão, como pode ser observada na descrição acima, não gera reativações de terminais como na exclusão do DA. Deste modo, não ocorre desperdício de tempo e de mensagens para excluir e incluir terminais participantes da solução, para reconstrução do grafo de saturação. Então, as reativações, embora ajudassem a manter a boa qualidade da solução, prejudicavam muito o desempenho do algoritmo.

### 4.2. Exemplo da Nova Exclusão - Antes e Depois

A Figura 3 exibe um exemplo da nova exclusão. Os terminais são representados por quadrados e os demais nós como círculos. O terminal raiz é representado pelo quadrado com a letra *R*, enquanto o terminal a ser excluído é representado pelo quadrado com a letra *E*.

Na Figura 3(a), é mostrada a configuração dos arcos. Os valores fora dos parenteses representam os custos dos arcos, e os de dentro, os custos que foram subtraídos para que os arcos fossem saturados. Note que o arco de custo 3 possui dois valores para o custo reduzido. Isto indica que este arco é utilizado por dois terminais. O primeiro valor dentro do parentese indica que o terminal mais a esquerda foi o responsável por sua saturação.

Na Figura 3(b), o terminal sainte inicia o envio em *broadcast* de sua exclusão. Com a propagação desta mensagem, os nós cancelam as reduções realizadas pelo terminal sainte, com exceção do segundo, que percebe que o arco de custo 3 é também utilizado por outro terminal. Assim, este nó escolhe arbitrariamente um terminal substituto que utilize o arco, e altera a configuração do custo reduzido, indicando agora que o substituto é o responsável pela saturação, ilustrado na Figura 3(c).

Na Figura 3(d), é enviada a mensagem de *Subst* ao terminal substituto, informando-o que ele é agora o responsável pela saturação do arco de custo 3, para que ele possa atualizar seu custo parcial da solução. Em resposta à mensagem *Subst*, na Figura 3(e), o terminal escolhido como substituto envia uma mensagem *Atual* ao terminal raiz, com seu custo parcial da solução atualizado.

Na operação de exclusão do DA para esta mesma situação exemplificada, a propagação da mensagem em *broadcast* da exclusão iria cancelar todas as saturações realizadas pelo terminal sainte. Desta forma, o arco de custo 3 teria zero subtraído do seu custo original. O nó que o possui como arco incidente, percebe que este também é utilizado por outro terminal. Com isso, é enviada uma mensagem de reativação, informando ao terminal que seu grafo de saturações foi danificado. Ao receber esta mensagem, o terminal inicia seu processo de exclusão, seguido por sua inclusão.

Este exemplo, embora simples, ilustra a diferença entre as operações de exclusão em relação ao custo. Embora para este exemplo a diferença de custo não seja tão alta, na prática foi possível observar nos experimentos realizados muitos casos em que di-

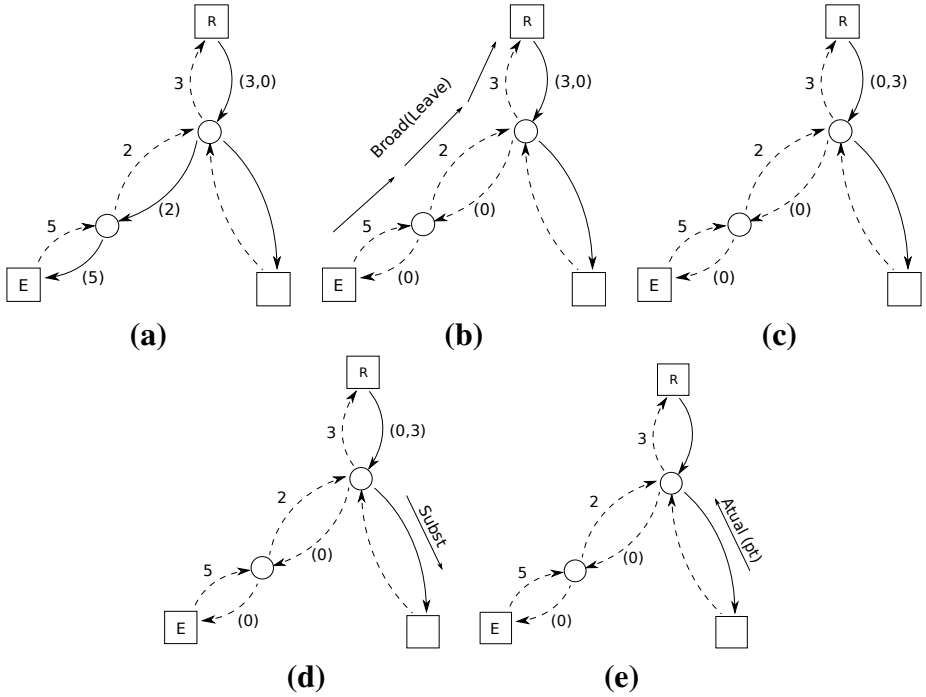


**Figura 3. Exemplo da operação de exclusão.**

versos terminais utilizam o mesmo arco saturado, resultando em um grande número de reativações.

## 5. Comparação Entre o Aries e o Novo Dual-Ascent *On-Line*

Os autores do Aries apontam como uma das vantagens desta heurística a pouca alteração realizada na estrutura da árvore após cada operação (mesmo no caso de reorganizações, dado que elas são localizadas). O DA_Mod também apresenta esta virtude. 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 pelas 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. O DA_Mod, por outro lado, dispara reorganizações somente quando o custo da solução torna-se muito alto, se comparado ao limite inferior fornecido pelo algoritmo.

Outra diferença importante entre o Aries e o DA_Mod 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 nas tabelas de roteamento nas redes práticas. O DA_Mod 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.

### 5.1. Complexidades para Inclusão e Exclusão de Terminais

A complexidade para inclusão no DA_Mod é a mesma da versão anterior. Entretanto, com as modificações realizadas na operação de exclusão, a complexidade desta foi reduzida significativamente.

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, por sua vez, enviarão mensagens *Check* para todos os demais. Portanto, a complexidade de mensagens é $O(|V|^2)$. Para o tempo, o pior caso ocorrerá quando os nós são incluídos um a um formando um caminho até a inclusão do nó raiz. 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 $O(|V|^2)$.

Considerando a exclusão de um terminal, no pior caso, será necessário encontrar um terminal substituto para cada arco do grafo de saturação entre o raiz e o terminal sainte pelo qual este seja responsável pelo custo reduzido. Ou seja, é necessário encontrar um outro terminal para substituir o terminal sainte e também responsável pelo custo reduzido do arco, sendo agora o terminal substituto o novo responsável pelo custo reduzido do arco. Como cada nó é responsável por armazenar as informações dos custos reduzidos de seus arcos incidentes, eles devem escolher um substituto e enviar uma mensagem de atualização, se necessário. Deste modo, a complexidade de tempo e mensagem da exclusão é $O(|V|)$.

As complexidades de tempo e de mensagens da heurística Aries dependem do número de reorganizações realizadas. Este número, por sua vez, depende do limiar de reorganização estipulado pelo usuário. No entanto, considerando apenas os mecanismos básicos de inserção e remoção, a análise de complexidade é bastante simples.

Para adicionar um novo terminal à árvore, no pior caso, será utilizado um caminho contendo $|V| - 2$ nós do grafo. O número de mensagens para este operação é, portanto, $O(|V|)$. Como cada nó do caminho envia sua mensagem apenas após a recepção da mensagem do nó anterior, a complexidade de tempo é também $O(|V|)$.

Para a operação de remoção, o nó que deseja sair do grupo *multicast* verifica se é uma folha da árvore. Em caso afirmativo, ele envia uma mensagem de desconexão para seu nó pai na árvore que, por sua vez, verifica se também pode deixar a árvore (caso ele não seja um terminal). No pior caso, este processo se repete até que a mensagem de desconexão passe por todos os nós do grafo, resultando em uma complexidade de mensagens e de tempo de $O(|V|)$. Caso o nó a ser removido não seja uma folha da árvore, basta que ele avise aos terminais estáveis da região sobre a sua saída (ou seja, que os contadores devem ser incrementados). Isto é feito através de difusão de uma mensagem por todas as arestas da árvore. No pior caso, novamente as complexidades de mensagens e tempo são iguais a $O(|V|)$.

O funcionamento do Aries necessita ainda do conhecimento do caminho de menor custo de cada nó para todos os outros da rede. Tal conhecimento depende da prévia execução de um algoritmo de caminho mínimo, o que adicionaria $O(|V|)$ à complexidade

**Tabela 1. Comparação entre o DA_Mod com folga de 2% e o Aries com limite igual a 6.**

| Instâncias | Nº de Reog. | | Ganho Médio | | Distância Média | | Tempo Global | | Nº de Mensagens | |
|---|---|---|---|---|---|---|---|---|---|---|
| | DA_Mod | Aries | DA_Mod | Aries | DA_Mod | Aries | DA_Mod | Aries | DA_Mod | Aries |
| **glp_as_1** | 12.8 | 1 | **5.34** | -2.67 | **0.54** | 1.58 | **878.00** | 35148.00 | **46068.20** | 198020.86 |
| **glp_as_2** | 13 | 1 | **5.29** | -0.22 | **1.39** | 1.80 | **1952.80** | 46278.43 | **93227.40** | 241666.14 |
| **glp_as_3** | 12 | 1 | **3.97** | -0.96 | **3.95** | 4.69 | **1450.00** | 45820.29 | **63591.40** | 253733.29 |
| **glp_as_4** | 11 | 1 | **6.04** | 0.05 | 2.10 | **1.92** | **1959.40** | 53354.00 | **104335.60** | 317859.86 |
| **glp_as_5** | 11 | 1 | **3.02** | 0.01 | **0.99** | 1.75 | **1557.00** | 55781.43 | **69687.40** | 287659.00 |
| **glp_asht_1** | 6 | 1 | **3.63** | 0.36 | **0.37** | 0.44 | **382.60** | 35911.43 | **12904.60** | 169585.00 |
| **glp_asht_2** | 6 | 1 | **2.92** | -0.71 | 0.87 | **0.26** | **1644.80** | 99947.86 | **52932.00** | 455584.14 |
| **glp_asht_3** | 12 | 1 | **2.98** | -2.37 | 1.95 | **1.30** | **967.20** | 41984.43 | **37698.60** | 215563.57 |
| **glp_asht_5** | 13 | 1 | **4.96** | -0.44 | 1.89 | **1.82** | **1221.20** | 36903.57 | **103764.00** | 225460.43 |
| **glp_rt_1** | 2 | 1 | **1.96** | 0.28 | **0.84** | 0.92 | **290.40** | 69940.71 | **8775.40** | 396784.86 |
| **glp_rt_2** | 9 | 1 | **2.55** | -0.88 | **0.67** | 1.44 | **597.80** | 61257.43 | **26437.00** | 334225.57 |
| **glp_rt_3** | 12 | 1 | **5.14** | 0.30 | 1.34 | **1.10** | **745.20** | 35129.14 | **31944.80** | 202852.14 |
| **glp_rt_4** | 13.6 | 1 | **4.99** | 0.93 | **1.15** | 1.83 | **2081.00** | 53907.43 | **103586.00** | 299251.00 |
| **glp_rt_5** | 9 | 1 | **5.36** | 4.20 | **2.01** | 3.30 | **892.20** | 51392.86 | **76726.40** | 309744.43 |
| **glp_rtht_1** | 6.8 | 0 | **0.34** | 0.00 | 0.42 | **0.37** | **820.00** | 34330.14 | **25791.00** | 186069.14 |
| **glp_rtht_2** | 12 | 1 | **5.93** | -0.11 | **3.24** | 8.45 | **1039.60** | 92778.14 | **42656.00** | 511067.43 |
| **glp_rtht_3** | 12.6 | 1 | **6.87** | 3.93 | **3.24** | 7.87 | **802.60** | 43059.29 | **26428.80** | 216324.57 |
| **glp_rtht_4** | 12 | 0 | **10.33** | 0.00 | 5.04 | **4.00** | **3283.60** | 35993.57 | **86131.00** | 195609.86 |
| **glp_rtht_5** | 9 | 0 | **8.01** | 0.00 | **0.91** | 1.28 | **1508.40** | 27666.29 | **70072.20** | 178206.57 |
| **Média** | 10.25 | 0.84 | **5.14** | 0.09 | **1.73** | 2.43 | **1267.04** | 50346.55 | **56987.25** | 273435.15 |

de tempo e $O(|V| \cdot |E|)$ à complexidade de mensagens. No entanto, como apenas uma execução do algoritmo de caminho mínimo é necessária para todas as operações de adição e remoção, a relevância destes valores de complexidade cai com o aumento do número de operações. Logo, não seria justo adicionar tais custos às complexidades totais do Aries.

De acordo com a análise feita em [Bauer e Varma 1996], uma execução completa da heurística K-SPH tem, no pior caso, complexidade de mensagens de $O(|T| \cdot |V|)$ e complexidade de tempo de $O(D \cdot |T|)$, onde $|T|$ denota o número de terminais e $D$ o diâmetro do grafo. No caso de uma execução parcial (limitada a uma região), como utilizada pelo Aries, as complexidades caem para $O(C \cdot |V|)$ e $O(D \cdot C)$, onde $C$ denota o número de componentes resultantes da desconexão da árvore.

## 6. Resultados Experimentais

A implementação do DA_Mod foi realizada com base na versão anterior do Dual-Ascent dinâmico, implementada em ANSI C e MPICH2. As execuções foram realizadas em um único computador Intel Core2Quad Q6600 com 2 GBytes de memória. As instâncias, que modelam características da Internet, foram as mesmas utilizadas anteriormente, geradas com o BRITE (*Boston University Representative Internet Topology Generator*) [Medina et al. 2001]. Cada instância possui 100 nós e, em média, 175 arestas.

Dois tipos de instâncias foram avaliadas. As do tipo AS (*autonomou systems*), nas quais cada nó do grafo representa um sistema autônomo, e as do tipo RT (*Router*), nas quais cada nó é um único roteador e os arcos representam enlaces reais. As instâncias podem ainda apresentar a variação HT (*heavy tailed*), relacionada à distribuição de probabilidade utilizada para gerar os arcos entre os nós. Na prática, esta variação implica em mais nós com grau muito mais alto que a média. Cada instância define também um grupo *multicast* inicial (contendo 15 nós) e uma sequência de 15 operações de inserção e remoção. Mais detalhes sobre as instâncias são encontrados em

[Drummond et al. 2009, Santos et al. 2009].

A Tabela 1 mostra os resultados do DA_Mod com folga de 2% (*i.e.*, 2% é a distância máxima tolerada entre uma solução e o limite inferior dado pelo algoritmo, sem uma reorganização ser disparada) e do Aries com limite igual a 6 (*i.e.*, mais de 6 nós modificados em uma região disparam uma reorganização) para 19 instâncias. Estes parâmetros foram utilizados pois: a folda de 2% apresentou melhores resultados com base no tempo de execução e, o valor 6 foi classificado pelos seus autores como o melhor limite inferior. Os valores são relativos à média de 5 execuções, apresentando os desvios padrões máximos de 13.3% para o Aries e 6.3% para o DA_Mod em relação ao tempo global, 12.4% para o Aries e 4.7% para o DA_Mod em relação ao número de mensagens, e 0% para o Aries e 8.7% para o DA_Mod em relação ao ganho médio por reorganização. A tabela contém o número de reorganizações, o ganho médio por reorganização, a distância média do ótimo, o tempo global (definido em 5.1) e o número de mensagens enviadas, no total da execução.

Como é possível observar, o DA_Mod faz mais reorganizações que o Aries. Entretanto, o ganho médio com as reorganizações no DA_Mod é maior, pois o critério de disparo é baseado na qualidade da solução a partir do limite inferior. Além disso, algumas reorganizações do Aries pioram a solução, resultando em um ganho médio negativo.

Em relação à distancia média do ótimo, em algumas instâncias o Aries consegue resultados ligeiramente melhores que o DA_Mod. Contudo, mesmo nestas instâncias a distância entre as soluções encontradas não ultrapassou 1%. Em outras palavras, o DA_Mod sempre apresentou resultados competitivos.

Uma diferença grande entre os dois algoritmos é nos tempos globais e no número de mensagens enviadas. Como mostrado na Tabela 1, o tempo global do DA_Mod é, em média, 40 vezes menor. Para as mensagens enviadas, os números do Aries são, na média, 4,8 vezes maiores.

Esta grande diferença entre o tempo global e o número de mensagens enviadas pelos dois algoritmos é justificada por três razões principais. A primeira é que o Aries necessita das informações das distâncias mínimas entre os nós e para isto utiliza um algoritmo de caminho mínimo. Além disso, o Aries necessita de uma árvore inicial, obtida através de uma execução completa do K-SPH (o procedimento mais caro da heurística). Embora o DA_Mod também utilize o K-SPH (nas suas reorganizações), suas execuções são feitas sobre o grafo de arcos saturados. Logo, a complexidade do K-SPH nas execuções do DA_Mod é bem menor que no caso do Aries.

Os gráficos das Figuras 4 e 5 mostram um comparativo entre os custos encontrados pelo Aries e pelo DA_Mod, além dos custos ótimo e do limite inferior dado pelo algoritmo Dual Ascent para duas instâncias distintas. Os círculos e quadrados plotados próximos ao eixo das abscissas indicam se houve ou não uma reorganização disparada após a operação correspondente. Os valores ótimos foram obtidos com a execução do algoritmo *branch-and-bound* descrito em [Poggi de Aragão et al. 2001].

Para a instância avaliada na Figura 4, os dois algoritmos têm resultados semelhantes até a oitava operação. A partir deste ponto, o custo da árvore do Aries se afasta do ótimo, permanecendo acima do custo da árvore do DA_Mod até a última operação. Este gráfico também mostra que, embora o DA_Mod tenha feito mais reorganizações, em ne-

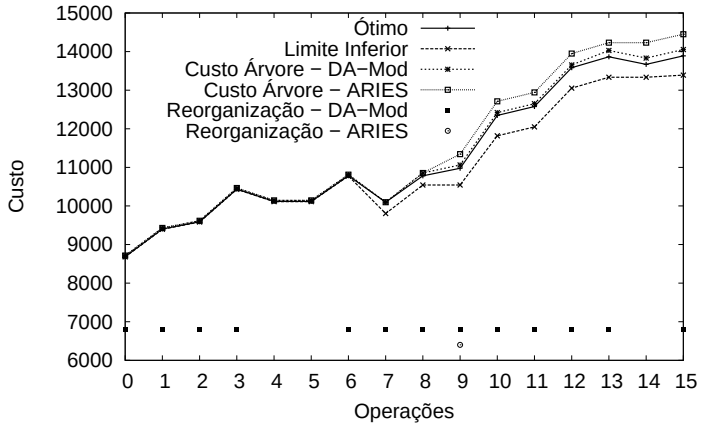


**Figura 4. Instância glp_as_1 com operações 15 operações.**

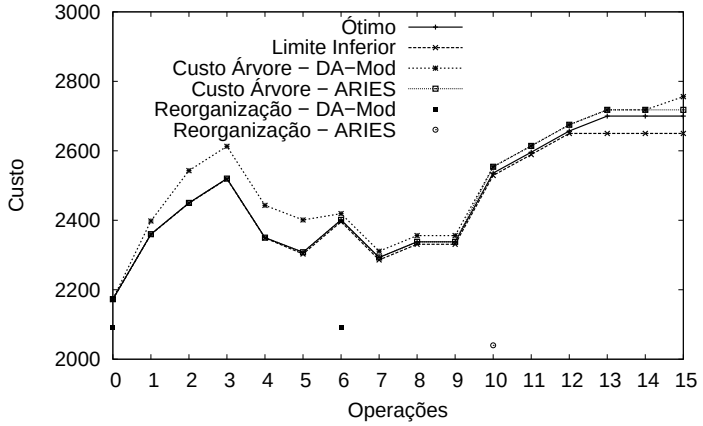


**Figura 5. Instância glp_asht_2 com 15 operações.**

nhum momento as reorganizações pioraram a solução. Entretanto, a única reorganização efetuada pelo Aries o afastou ainda mais do ótimo. Este resultado reforça o argumento de que a reorganização baseada no limite inferior é mais eficiente.

Já na instância ilustrada na Figura 5, o Aries obteve soluções sempre melhores ou iguais ao DA_Mod. Nas primeiras operações, o custo da árvore do DA_Mod permanece mais alto que o custo da árvore do Aries. A partir da sexta operação, na qual o DA_Mod realiza uma reorganização, a diferença entre estes custo é reduzida. Na décima operação, o Aries dispara a sua reorganização. No entanto, mesmo com esta reorganização o custo do Aries se mantém igual ao custo do DA_Mod.

## 7. Conclusão

Neste artigo, foi apresentada uma nova versão do Dual-Ascent para o Problema de Steiner *On-Line*, o DA_Mod. Neste novo algoritmo, as operações de exclusão não geram mais reativações de terminais, pois quando um arco é utilizado por dois ou mais terminais e o responsável pelo custo reduzido é excluído, um dos outros terminais é escolhido como substituto para ser o novo responsável pelo custo reduzido. Com isso, foi possível reduzir a complexidade desta operação, o que tornou o DA_Mod competitivo em relação ao Aries. Ambos os algoritmos foram implementados e uma comparação entre seus resultados foi

realizada, mostrando que o DA_Mod mesmo fazendo mais reorganizações, teve soluções em média melhores que as obtidas pelo Aries. Além disso, os tempos globais e número de mensagens gastos pelo DA_Mod foram consideravelmente mais baixos que os gastos pelo Aries.

## Referências

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.

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. M. A., Santos, M., e Uchoa, E. (2009). A distributed dual ascent algorithm for steiner problems in multicast routing. *Networks*, 53(2):170–183.

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.

Lin, H. e Lai, S. (1998). Vtdm-a dynamic multicast routing algorithm. In *Proceedings of annual joint conference of the IEEE computer and communications societies INFOCOM'98*, volume 3, pages 1426–1432.

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. (2001). 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.

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.

Santos, M., Drummond, L. M. A., e Uchoa, E. (2009). Dual-ascent distribuído para o problema de steiner on-line. In *Anais do simpósio brasileiro de redes de computadores e sistemas distribuídos SBRC'09*, pages 231–244.

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