Universidade Federal Fluminense

JULIANA MENDES NASCENTE SILVA

Estratégias de balanceamento de carga para um algoritmo branch-and-bound paralelo para executar em Grids computacionais

NITERÓI

2006

JULIANA MENDES NASCENTE SILVA

Estratégias de balanceamento de carga para um algoritmo branch-and-bound paralelo para executar em Grids computacionais

Dissertação de Mestrado submetida ao Programa de Pós-Graduação em Computação da Universidade Federal Fluminense como requisito parcial para a obtenção do título de Mestre. Área de concentração: Sistemas Distribuídos e Processamento Paralelo.

Orientadora:

Lúcia M. A. Drummond

Universidade Federal Fluminense

NITERÓI

2006

Estratégias de balanceamento de carga para um algoritmo *branch-and-bound* paralelo para executar em *Grids* computacionais

Juliana Mendes Nascente Silva

Dissertação de Mestrado submetida ao Programa de Pós-Graduação em Computação da Universidade Federal Fluminense como requisito parcial para a obtenção do título de Mestre.

Aprovada por:

Profa. Lúcia M. A. Drummond / IC-UFF (Presidente)

Profa. Maria Clícia S. Castro / UERJ

Profa. Roseli Wedemann / UERJ

Profa. Simone Lima Martins / IC-UFF

Prof. Dorgival Olavo Guedes Neto/ DCC-UFMG

Niterói, 15 de Fevereiro de 2006.

Dedico este trabalho primeiramente à Deus, meu tudo! Meus pais, minha força. A todos os amigos, em especial ao Lê, pela paciência.

Agradecimentos

Agradeço primeiramente à Deus, minha força, meu refúgio...por ter me dado deixado desfrutar do conhecimento e da sabedoria. A meus pais e irmãs, vovó, tia Maria, pela força! Aos meus amigos que torceram de longe, Adalbert, Pop's, Andreísa, Rodrigo, Dany, Paula...meus alunos! E aqueles que torceram e sofreram comigo cada pedacinho desta etapa: Lê, pela paciência, carinho, dedicação... Jaquinho, pelas aflições no laboratório. Lu, minha irmãzinha de quarto no qual pudemos partilhar alegrias e aflições de nossas vidas! Stênio, pela ajuda, pela amizade! Ao Johnny pela partilha, amizade! Renathinha, Dany, Vivi, Rodrigo, Diego, Jonivan, Ary, Cris, Brunex.... enfim todos aqueles que fizeram dos momentos de desespero do laboratório momentos inesquecíveis de cumplicidade e amizade...

Ao Alexandre, Eduardo Uchoa pelos conhecimentos que não hesitaram em me passar. À Lúcia, orientadora e conselheira. A Roseli, pelo aprendizado que está implicitamente contido neste trabalho.

Agradeço especialmente ao Ulisses, que acreditou em mim mesmo antes quando eu mesma não acreditava. Pelos conselhos, pela amizade!!! A DOCTUM pelo apoio financeiro, sem o qual este trabalho não poderia ser desenvolvido. Aos professores e amigos da FIC!!! Enfim a todos que participaram direta ou indiretamente neste período de busca por conhecimento... *AMO MUITO VOCÊS*!!!

Resumo

Esta dissertação propõe três estratégias de balanceamento de carga para o algoritmo *branch-and-bound* paralelo aplicado ao Problema de *Steiner* em Grafos (PSG) para ser executado em *Grids* computacionais. Geralmente, *Grids* são formados por processadores heterogêneos e não dedicados. Além disto, são organizados de modo hierárquico: processadores pertencentes a um mesmo *cluster* são conectados através de *links* de velocidade mais alta do que processadores de *clusters* geograficamente distantes. A utilização de uma estratégia de balanceamento de carga dinâmica, capaz de adequar a carga ainda não resolvida aos recursos diponíveis no ambiente, é essencial para melhorar a eficiência paralela do algoritmo.

Foram propostas duas estratégias totalmente distribuídas e uma centralizada que estimam o tamanho da carga a ser enviada e avaliam os desempenhos apresentados pelo processadores mediante a uma requisição de carga. Os testes foram realizados com instâncias do PSG contidas no repositório *SteinLib*. As estratégias se mostraram eficientes quando comparadas com outras estratégias de balanceamento de carga existente para o problema.

Abstract

This work introduces three techniques of load balancing strategies for a distributed branch-and-bound algorithm, applied to the Steiner Problem in Graphs (SPG), to be executed on computational Grids. Many Grids are composed of cluster of processors not dedicateds and heterogeneous. Moreover, the processors belonging to a same cluster are connected via highspeed links and the clusters, geographically distant, are connected through low-speed links, in a hierarchical fashion. In order to improve the the efficience of parallel algorithms in these enviroments, dynamic load solved among the available resources, are crucial.

Two completely distributeds strategies and a centralized one that employs the usual master-worker paradigm were proposed. They estimate the size of the load to be transfered and evaluate the performance presented by processors at each load request. The experiments were carried out using SPG intance from *SteinLib*. The proposed strategies showed to be efficiente when compared with other existing load balance strategies for this problem.

Two strategies total distributed and one employ the usual master-worker paradigm. They estimate the size of the load and evaluate the performances presented for the processors to do transference. The tests experiments had been carried through instances of the SPG contained in *SteinLib*. The strategies demonstrated its efficiency and scalability.

Palavras-chave

1. *Branch-and-Bound*
2. Balanceamento de carga
3. *Grids* computacionais

Glossário

PSG : Problema de *Steiner* em Grafos;

B&B : *Branch-and-Bound*;

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

Uma aplicação paralela objetiva diminuir o tempo de execução de uma tarefa dividindo-a em várias subtarefas que são executadas paralelamente em vários processadores. Tarefas adequadas ao paralelismo são aquelas que exigem um alto poder computacional para alcançar um resultado. Problemas de otimização da classe NP-difícil, simulações físicas e cálculos científicos são exemplos destes tipos de tarefas.

Uma *Grid* computacional é uma plataforma de execução paralela e/ou distribuída que agrega recursos heterogêneos pertencentes à diversas organizações dispersas geograficamente [Foster e Kesselman 1998] [Foster e Kesselman 1999]. Uma das vantagens reconhecidas deste tipo de ambiente é a possibilidade de alocar grandes e variadas quantidades de processadores a uma aplicação paralela a um custo muito menor do que as alternativas tradicionalmente conhecidas, como por exemplo, supercomputadores paralelos. Uma *Grid* pode ser formada pela combinação de computadores, multicomputadores simétricos, processadores maciçamente paralelos ou ainda redes de estações de trabalho ou PCs, normalmente chamadas de *clusters*. Neste último caso, a comunicação entre os processadores que pertencem a *clusters* diferentes é realizada através da Internet, portanto, é dependente da latência da rede. Já a comunicação entre processadores pertencentes a um mesmo *cluster* é realizada por uma rede local. Assim, as taxas de transmissões nos canais de comunicação que interligam os processadores é variada, o que sugere uma hierarquia de comunicação: processadores que pertencem a um mesmo *cluster* se comunicam por canais de comunicação que possuem taxas de transmissão mais altas do que os canais que interligam processadores que pertencem a *clusters* diferentes.

O desenvolvimento de uma aplicação paralela deve considerar os vários aspectos da plataforma onde será executada. Além da hierarquia de comunicação, deve ser considerada a heterogeneidade dos recursos que compõem o ambiente: os processadores podem possuir

diferentes velocidades e arquiteturas. Outra particularidade é que *Grids*, normalmente, não são dedicadas, ou seja, o tempo de execução da aplicação é influenciado pelos processos de outros usuários que disputam o processador. Qualquer processo que não faz parte da aplicação e que cause esta disputa é chamado de carga externa. Contudo, a carga externa é dinâmica e pode variar tanto durante a aplicação como em execuções consecutivas da aplicação. Assim, o desempenho de um processador não diz respeito apenas à velocidade com que executa as instruções, mas depende, também, da quantidade de carga que resolve em um período de tempo.

Para que os recursos de uma *Grid* sejam aproveitados em sua totalidade são necessárias estratégias de balanceamento de carga bem estruturadas associadas à aplicação paralela, sendo estas estratégias, responsáveis por evitar, dinamicamente, a ociosidade dos processadores. Estratégias de balanceamento de carga incluem a transferência de subtarefas (ou cargas) de processadores sobrecarregados para processadores subcarregados. A quantidade de carga transferida deve ser proporcional ao desempenho dos processadores envolvidos no balanceamento. Entretanto, a escolha da quantidade e de quais subtarefas deverão ser transferidas não é uma tarefa trivial, pois, por muitas vezes, os tamanhos das subtarefas são diferentes e não são conhecidos *a priori*. Vale ressaltar que um processador subcarregado é considerado, neste trabalho, como aquele que está sem carga, ou seja, ocioso, enquanto processador sobrecarregado é aquele que posssui carga ainda não resolvida quando existem processadores subcarregados na aplicação.

Neste trabalho foram desenvolvidas e analisadas três novas estratégias de balanceamento de carga para o algoritmo *branch-and-bound* paralelo para executar em ambientes *Grids*, aplicado ao Problema de *Steiner* em Grafos (PSG). Este é um problema da classe NP-difícil e pode ser computacionalmente muito custoso. O algoritmo *branch-and-bound* é um método exato muito utilizado para encontrar a solução ótima de problemas de otimização combinatória como o PSG. Dado um grafo ponderado G com custo positivo nas arestas e um conjunto de vértices, chamados terminais, o Problema de *Steiner* em Grafos consiste em encontrar uma árvore em G que contenha todos os vértices terminais e cujo o custo total das arestas seja mínimo.

O *branch-and-bound* possui um grande potencial para o paralelismo porque a busca por uma solução acontece através de enumeração de árvores e a computação das subárvores pode ser realizada quase independente uma das outras. No *branch-and-bound* distribuído, a cada processador é alocada uma subárvore cujo tamanho pode variar, fazendo com que alguns processadores se tornem ociosos rapidamente. Diante deste fato, a utilização de

técnicas de balanceamento de carga no algoritmo é fundamental para garantir a eficiência paralela da aplicação.

As três estratégias de balanceamento de carga desenvolvidas neste trabalho utilizam informações sobre o desempenho dos processadores e sobre o tamanho das subtarefas (capacidade de ramificação dos nós das subárvores) para definir a quantidade de carga a ser transferida entre os processadores. Das três estratégias implementadas, duas delas, chamadas *GlobalAdapt* e *HibridaAdapt* são totalmente distribuídas. Um processador quando se torna ocioso envia um pedido de carga para um outro processador, que, por sua vez, em resposta ao pedido, divide a carga de acordo com o desempenho apresentado pelos mesmos. Na estratégia *GlobalAdapt*, um processador enviará uma requisição de carga a qualquer processador que participa da aplicação, independente de sua posição geográfica. A técnica *HibridaAdapt* considera a hierarquia de comunicação, priorizando o balanceamento de carga entre processadores de um mesmo *cluster*, mas permite também que ocorra transferência de carga entre *clusters* diferentes, quando todos os processadores dentro do *cluster* estiverem ociosos. A última estratégia, denominada *MeAdapt*, é centralizada no sentido em que existe um processador mestre responsável por manter todos os processadores escravos ocupados. Um processo ao se tornar ocioso envia um pedido de carga ao mestre, que responde ao pedido enviando uma quantidade de carga proporcional ao desempenho do processo que fez a requisição em relação aos outros processos participantes da aplicação.

As estratégias *GlobalAdapt* e *HibridaAdapt* foram baseadas, respectivamente, em duas estratégias existentes para o algoritmo *branch-and-bound* paralelo aplicado ao PSG propostas por [Drummond et al. 2005] chamadas, *Global* e *Híbrida*, sendo que, esta última, a *Híbrida*, introduziu a utilização do conceito de hierarquia de comunicação com o intuito de aumentar a eficiência paralela do algoritmo. Apesar das técnicas terem apresentado bons resultados, não consideram qualquer outra característica do ambiente ou particularidade do problema.

A versão seqüencial do algoritmo *branch-and-bound* aplicado ao PSG, no qual foi baseado o paralelo utilizado neste trabalho, foi proposto por Uchoa e Werneck em [Uchoa 2001] e [Werneck 2001], respectivamente. De um conjunto de 400 instâncias do PSG difíceis, propostas por Duin em [Duin 1993] como um *benchmark* e um desafio para algoritmos futuros, este algoritmo foi capaz de solucionar quase todas as instâncias com tempos razoavelmente pequenos. As instâncias estão disponíveis no repositório SteinLib [Koch et al. 2003].

O objetivo deste trabalho foi desenvolver, classificar, analisar e avaliar o desempenho das estratégias propostas. Para tanto, as estratégias *GlobalAdapt*, *HibridaAdapt* e as propostas por [Drummond et al. 2005] foram executadas para dez diferentes instâncias retiradas de [Koch et al. 2003]. Os resultados foram comparados e analisados considerando-se: i) a ociosidade dos processos durante a aplicação, ou seja, foi verificado se a quantidade de subtarefas enviadas durante uma transferência foi suficiente para evitar que processos se tornassem ociosos rapidamente ; ii) o número de mensagens trocadas entre os processos; iii) o tempo de execução do algoritmo; iv) a eficiência das estratégias em relação à carga externa e a heterogeneidade dos processadores que participam do ambiente. A estratégia centralizada *MeAdapt* foi comparada com as duas versões distribuídas com o intuito de avaliar o *overhead* de comunicação e a ociosidade dos processos diante do paradigma mestre-escravo.

Este trabalho está dividido da seguinte forma: o Capítulo 2 aborda conceitos de balanceamento de carga em Grids computacionais e apresenta as principais técnicas de balanceamento de carga citadas na literatura; o Capítulo 3 subsiste na explicação do algoritmo *branch-and-bound* seqüencial e do problema no qual foi aplicado; os procedimentos do *branch-and-bound* distribuído estão descritos no Capítulo 4; as estratégias de balanceamento de carga desenvolvidas constituem o Capítulo 5; o Capítulo 6 apresenta, discute e compara os resultados obtidos quando utilizadas as técnicas desenvolvidas neste trabalho e quando utilizadas as técnicas propostas por [Drummond et al. 2005]; finalmente, o Capítulo 7 apresenta as conclusões e sugestões de trabalhos futuros.

Capítulo 2

Balanceamento de Carga

Estratégias de balanceamento de carga objetivam distribuir e/ou redistribuir a carga entre os processadores de maneira a utilizar o ambiente em sua totalidade. Este capítulo é destinado a conceituar, descrever e classificar as técnicas mais utilizadas para equilibrar a carga de uma aplicação paralela e as estratégias primordiais de balanceamento implementadas para o algoritmo *branch-and-bound* distribuído disponíveis na literatura.

2.1 Estratégias de balanceamento de carga para *Grids* computacionais

Grids proporcionam um ambiente de alto desempenho utilizado para solucionar problemas que demandam alto poder computacional. Uma das vantagens desta plataforma é a economia, pois, normalmente, é formada pela união de redes locais de estações de trabalho ou PC's, denominados *clusters*, disponibilizada por instituições que almejam um supercomputador. Devido à sua natureza estrutural, as Grids, geralmente, apresentam as seguintes características:

- **Heterogeneidade**: os processadores que participam do ambiente podem possuir arquiteturas diferentes. Portanto, a velocidade da execução das instruções podem não ser a mesma para todos os processadores. Isto implica que a divisão das subtarefas em partes iguais entre os processadores não assegura que todos os recursos ficarão devidamente carregados durante toda a execução da aplicação.
- **Compartilhamento**: este fator intensifica a heterogeneidade do ambiente. Geralmente, Grids não são totalmente dedicadas. Individualmente, os processadores,

além de estarem alocados aos processos da aplicação, podem estar alocados a processos de outros usuários, que podem ser, por exemplo, aplicações paralelas, execução de programas seqüenciais que demandam muito processamento ou ainda, apenas utilização dos recursos oferecidos pelo sistema operacional. Outros processos que disputam o processador com a aplicação paralela são chamados carga externa.

- **Hierarquia de comunicação**: normalmente, Grids são formadas por um conjunto de *clusters*. Processadores de um mesmo *cluster* são interligados por uma rede local enquanto processadores que pertencem a *clusters* diferentes são conectados via Internet, fazendo com que a comunicação dentro do *cluster* seja mais rápida do que entre *clusters*, o que determina dois níveis de comunicação.

Balanceamento de carga consiste na divisão da tarefa entre os processadores proporcionalmente ao desempenho e disponibilidade dos recursos que compõem o ambiente de execução. Para utilizar o ambiente em sua totalidade, reduzindo o efeito dos fatores de desequilíbrio ([Zaki e Parthasarathy 1997] [Willebeek-LeMair e Reeves 1993]), estratégias de balanceamento devem considerar: i) a estrutura hierárquica de comunicação do ambiente com o intuito de diminuir o *overhead* de comunicação entre os processos e diminuir assim o tempo de execução da aplicação; ii) as informações sobre o índice de carga externa e poder computacional dos processadores, para definir dinamicamente o desempenho dos processadores que participam do ambiente. A exploração destas características pode diminuir o número de transferência de carga entre os processos, compatibilizando as quantidades de carga aos desempenhos dos processadores, diminuindo o *overhead* de comunicação entre eles e, conseqüentemente, o tempo de execução da aplicação.

É válido notar que qualquer decisão sobre transferência de carga envolve a comunicação entre os processos, ou seja, um processo precisa de informações contidas em outros processos, como por exemplo, a quantidade de carga ainda não resolvida (índice interno de carga), desempenho, etc. Por outro lado, a troca de informação demasiada entre os processadores tem como conseqüência a redução da eficiência da aplicação. Dependendo da aplicação, o aumento do *overhead* de comunicação pode não compensar a utilização de procedimentos para balancear a carga.

O balanceamento de carga, basicamente, consiste na transferência de carga entre processos e envolve decisões como: i) quantos e quais processadores estarão envolvidos na transferência; ii) se a transferência será provocada por um processo que está sobrecarregado (considerando aqui um processo que possui uma quantidade de carga acima de um limite definido pela aplicação) para outros menos carregados (*receiver initiated*), ou se

será ocasionada por um processador que se torna ocioso e envia requisições de carga a processadores vizinhos (*sender initiated*) [Baoukov e Soverik 1999]; iii) a quantidade de carga que será transferida; iv) se o método será centralizado, ou seja, um único processador concentrará as informações sobre a transferência e avisará os outros processadores sobre como será o o procedimento de transferência de carga, ou se transferência será distribuída, ou seja, qualquer processador pode provocar a transferência de carga sem a intervenção de um coordenador.

As técnicas de balanceamento mencionadas na literatura se distinguem na exploração das características do ambiente e em quais decisões serão tomadas para realizar a transferência. Plastino em [Plastino 1999], sugeriu um conjunto de critérios de classificação para avaliar estratégias de balanceamento. Tais critérios são utilizados neste trabalho para classificar as estratégias desenvolvidas e definir qual é a melhor estratégia para o algoritmo *branch-and-bound*. A Tabela 2.1 resume os critérios mais relevantes propostos por [Plastino 1999].

Contudo, se os tamanhos das subtarefas geradas são diferentes e podem variar durante a execução da aplicação, estratégias de balancemanto de carga baseadas apenas na característica das Grids não seriam suficientemente eficazes. Este trabalho propõe estratégias de balanceamento para o algoritmo *branch-and-bound* que, além de considerarem particularidades do ambiente, estimam o tamanho das subtarefas a fim de evitar a distribuição indevida de carga. Para exemplificar, considere que o processador P_1 apresente um desempenho 30% melhor do que o processador P_2 e que a subtarefa S_1 gaste 25% a mais de processamento para ser resolvida do que a subtarefa S_2. Se nenhuma informação a respeito do tamanho das subtarefas for considerada poderia acontecer de S_1 ser alocada a P_2 e S_2 ser alocada a P_1. Conseqüentemente, P_1 ficaria ocioso rapidamente, provocando uma nova transferência de carga.

2.2 Estratégias de balanceamento de carga para o algoritmo *branch-and-bound*

O algoritmo *Branch-and-bound*, B&B, é uma técnica bastante utilizada para resolver problemas de otimização combinatória do tipo NP-difícil. O B&B busca em um espaço de soluções a solução ótima para um determinado problema, realizando operações de ramificação, chamadas *branching*, que dividem um problema em dois subproblemas similares (idéia proposta em [Land e Doig 1960] em 1960). Cada subproblema trabalha com uma

Critério	**Classificação**	
Momento de distribuição	**Estático:** quando a distribuição de carga ocorre somente no início da aplicação	**Dinâmico:** a distribuição ocorre no início da aplicação, mas pode ser alterada durante a execução
Forma de distribuição	**Sob demanda:** a distribuição acontece ao longo da execução. Um coordenador distribui um bloco de tarefas para cada processador e este, quando termina a execução, recebe outro bloco de tarefas	**Por transferência:** o balanceamento é realizado através do envio de carga de um processador sobrecarregado para um subcarregado
Utilização do índice de carga externa	**Integrado:** utiliza algum índice de carga externa	**Isolado:** não utiliza índice de carga externa
Escopo do balanceamento de carga	**Global:** a transferência de carga pode ocorre entre qualquer processador do sistema	**Local:** a transferência de carga ocorre entre processadores de um mesmo grupo (disjuntos ou não)
Localização da execução do algoritmo	**Centralizado:** é executado em um único processador que é responsável por informar aos processadores envolvidos sobre a transferência de carga	**Distribuído:** é executado localmente em cada processador
Sincronismo	**Síncrono:** participação, ao mesmo tempo, de todos os processadores	**Assíncrono:** o algoritmo é executado independentemente em cada processador
Ativação do algoritmo	**Periódico**	**Por evento**
Política de localização	**Não cego:** utiliza índice de carga interna na transferência	**Cego:** não utiliza índice de carga interna

Tabela 2.1: Critérios de classificação de estratégias de balanceamento de carga.

sub-região diferente do espaço de solução. A resolução dos subproblemas também inclui operações de ramificação (ou seja, o procedimento é repetido recursivamente). Assim, o algoritmo induz uma enumeração de árvores. O tamanho da árvore gerada pelo problema é proporcional ao tamanho da entrada do problema. Assim, se o espaço de solução é muito grande (e verdadeiramente são para problemas da classe NP-difícil) a busca pela solução se torna inviável devido ao tempo gasto para executar o algoritmo. Este problema é tratado utilizando *bounding*, que é um modo de encontrar limites para a solução ótima dentro de uma região viável. O interessante da abordagem B&B é que, para problemas de minimização, se o limite inferior de uma sub-região R é maior que o limite superior de qualquer outra sub-região, seguramente R pode ser descartada da busca. Este ato é chamado *pruning* ou poda [Gendron 1994].

Soluções aproximadas do problema fornecem limites superiores (para problema de minimização) válidos. Wong em [Wong 1984] propôs um algoritmo denominado *dual ascent*, que rapidamente encontra um limite inferior (dual) para o problema. Poggi de Aragão et al em [Wong 2001] propuseram três heurísticas adicionais para melhorar os limites do *dual ascent*, denominadas: *dual scaling*, *dual adjustment* e *active fixing by reduced costs*. O algoritmo B&B utilizado neste trabalho é baseado nestas heurísticas, e sua versão seqüencial está completamente descrita em [Uchoa 2001].

Nem sempre instâncias grandes são resolvidas em tempo razoável quando utilizado um B&B seqüencial. Para solucionar grandes instâncias, vários pesquisadores têm implementado algoritmos B&B paralelos com o intuito de melhorar o tempo de execução [Gendron 1994] [Baoukov e Soverik 1999] [Bruin et al. 1995] [Culler et al. 1993]. O B&B possui um grande potencial de paralelização porque as computações das subárvores podem ser realizadas de modo quase independente. A única informação trocada entre os processadores é o limite (primal) utilizado para poda. Contudo, a poda das ramificações intensifica o problema de desbalanceamento de carga entre os processadores, fazendo-se necessária a utilização de técnicas capazes de transferir cargas ainda não resolvidas entre os processadores ociosos.

A transferência de carga no B&B consiste na transferência das ramificações, também chamadas subárvores, entre os processadores. O tamanho de uma ramificação é dado pela profundidade alcançada pelas mesmas. A decisão sobre a quantidade de carga a ser transferida entre os processadores deve considerar, além do desempenho do processador, o tamanho das subárvores que serão transferidas. Por exemplo, considere uma transferência de carga entre dois processadores, P_1 e P_2. O dois processadores apresentam baixo índice

de carga externa e bom desempenho. O processador P_1 envia uma requisição de carga a P_2 que responde enviando metade das ramificações ainda não resolvidas. Esta situação manteria os dois processadores ocupados igualmente se não fosse a diferença no tamanho das ramificações. Se P_2 envia a P_1 ramificações de profundidades pequenas, P_1 se tornará ocioso rapidamente, fazendo com que uma nova requisição e transferência sejam realizadas. Assim, a quantidade de carga a ser enviada deve ser inversamente proporcional ao tamanho das subárvores.

Vários trabalhos recentes abordam algoritmos B&B paralelos para Grids, dentre eles [Gendron 1994] [k. Anstreicher et al. 2002] [Aida et al. 2003] [Ralphs et al. 2004]. Muitos deles adotam a abordagem centralizada, ou seja, um único processo, chamado mestre, mantém a árvore de tarefas a serem resolvidas e envia subárvores para os demais processos quando estes se tornam subcarregados.

Roucairol e Dowaji em [Roucairol e Dowaji 1995] propuseram três estratégias de balanceamento de carga. Em tais abordagens, cada processo mantém uma fila de subproblemas classificados de acordo com a sua potencialidade, definida como a estimativa da capacidade de um subproblema gerar trabalho, e a distância entre o subproblema e a solução ótima. Esta fila é resolvida utilizando a política do *melhor primeiro*. Em tais abordagens, o balanceamento de carga é centralizado, onde existe um processo mestre responsável por determinar a quantidade de subproblemas que será transferida de um processo sobrecarregado para outro subcarregado. A diferença entre as três estratégias consiste na variação da quantidade de subproblemas que serão enviados e na posição em que são retirados da fila, ou seja, início ou fim da mesma, sendo que nenhuma característica do ambiente é explorada. Segundo os critérios de classificação apresentados na Tabela 2.1, as estratégias possuem a mesma classificação: são dinâmicas, por transferência e isoladas, ou seja, não são consideradas as cargas externas à aplicação. As estratégias são do tipo global e ativadas por evento. A execução é assíncrona e não cega.

Linderoth et al em [k. Anstreicher et al. 2002] e [Goux et al. 2001] utilizam um *framework* chamado MW para equilibrar a carga de um B&B aplicado ao Problema Quadrático de Alocação. O MW implementa o paradigma mestre-escravo. O processo mestre delega tarefas aos escravos ociosos que enviam o resultado da tarefa de volta ao mestre. Segundo os autores, a abordagem centralizada é conveniente para o B&B, pois é dinâmica e, nesse caso, também permite a inclusão de um procedimento para tratar falhas de forma simples. Quando um escravo se torna ocioso durante a aplicação outra tarefa lhe é designada. Para melhorar a eficiência paralela do algoritmo, utiliza a seguinte estratégia: a

um escravo disponível (ocioso) é dado um nó da lista de nós do mestre. O escravo avalia a subárvore em sua primeira ramificação. Se depois de t_{max} segundos o escravo não terminar de resolver a subárvore, este envia de volta ao mestre os nós ainda não resolvidos. Esta técnica diminui a quantidade de requisição de carga, já que o escravo ficará certamente ocupado durante os t_{max} segundos, e o balanceamento de carga é obtido, uma vez que os escravos enviam ao mestre a carga que não foram capazes de resolver. Além disso, para evitar o "gargalo" do mestre é necessário que o mesmo gerencie uma fila pequena de nós. Para tanto, são enviados inicialmente os nós mais profundos e o t_{max} deve ser ajustado de modo que os escravos enviem tarefas de volta ao mestre. Esta estratégia, de acordo com Tabela 2.1, é classificada como dinâmica, sob demanda e isolada. É global, centralizada, assíncrona, por evento e não cega.

Segundo Aida et al em [Aida et al. 2003], a abordagem centralizada não é escalável e conveniente para ambientes Grids, podendo ocorrer uma degradação significativa no desempenho da aplicação devido ao alto *overhead* de comunicação entre os processos escravos e o mestre (afirmação também defendida por [Ralphs et al. 2004]). O mestre pode se tornar um "gargalo", não atendendo devidamente as requisições de todos os escravos. Assim, os autores propuseram, então, um B&B distribuído, baseado em um paradigma mestre-escravo hierárquico, onde um processo supervisor controla a ativação de um conjunto de processos formado por um processo mestre e vários escravos. A distribuição de tarefas entre os processos ocorre primeiramente do processo supervisor para os mestres e posteriormente dos mestres para os processos escravos. Os resultados computacionais são colhidos na ordem inversa. Os processos mestres mantêm uma fila de subárvores ainda não exploradas e enviam estas subárvores aos processos escravos que computam a subtarefa e retornam o resultado. O processo supervisor é responsável por realizar o balanceamento de carga entre os processos mestres. O tamanho das subárvores enviadas entre os processos mestres e escravos é baseado no tamanho do problema. Apesar dos bons resultados, esta estratégia não é completamente distribuída, ficando a cargo do processo supervisor a distribuição e redistribuição de carga entre os mestres e estes entre os escravos. Esta estratégia é dinâmica, sob demanda e isolada. A estratégia é classificada como local e ativada por evento, a execução é assíncrona, e cega (Tabela 2.1).

Lúcia M. A. Drummond et al em [Drummond et al. 2005] propõem duas estratégias totalmente distribuídas, *Global* e *Híbrida*, sendo que esta última consideram a hierarquia de comunicação apresentada pela Grid. Entretanto, nenhuma informação do problema é utilizada para manter o equilíbrio de carga entre os processadores.

As estratégias desenvolvidas neste trabalho utilizam a hierarquia de comunicação introduzida por [Drummond et al. 2005], avaliam o desempenho dos processadores participantes em relação a capacidade de resolver tarefas da aplicação e estimam o tamanho das subárvores que ainda não foram resolvidas. Associar a cada tarefa um valor referente a estimativa de seu tamanho é importante para evitar o envio de tarefas rápidas e por conseqüência evitar que processadores se tornem ociosos rapidamente. As estratégias propostas neste trabalho são classificadas, de acordo com a Tabela 2.1, como segue:

- *GlabalAdapt*: dinâmica, sob demanda. É integrada, pois utiliza o índice de carga externa para avaliar o desempenho dos processadores. É global, ou seja, permite a comunicação entre qualquer processador que compõe o ambiente, é distribuída, assíncrona e não cega.

- *HibridaAdapt*: é dinâmica, sob demanda e integrada. É incialmente local, pois, prioriza a comunicação entre processadores de um mesmo grupo, e posteriormente global, pois permite a transferência de carga entre processos de grupos diferentes quando não mais houver carga no grupo. A comunicação entre processadores de *cluster* diferentes é realizada através de um processador líder. É distribuída, assíncrona e não cega.

- *MeAdapt*: é dinâmica, sob demanda e integrada. É global e centralizada, um processo mestre é responsável por enviar carga a um processo que se torna ocioso, assíncrona e não cega.

No capítulo 5 estão descritas detalhadamente como foram exploradas e implementadas as pecularidades do ambiente e os problemas discutidos neste capítulo.

Capítulo 3

Branch-and-Bound seqüencial aplicado ao Problema de Steiner em Grafos - PSG

O problema de *Steiner* em grafos (PSG) é muito conhecido e estudado pelos pesquisadores devido a sua ampla utilização e o alto poder computacional que exige para que a solução ótima seja encontrada (veja [Du et al. 2000]). Este capítulo descreve detalhes deste problema e especifica o funcionamento do algoritmo B&B e os demais procedimentos que, quando combinados, solucionam o PSG.

3.1 O Problema de *Steiner* em Grafos

O problema de *Steiner* em Grafos (PSG) é definido como: dado um grafo G(V,E) não direcionado, um conjunto de vértices terminais T e uma função atribuindo um custo $c > 0$ às arestas, encontrar um subgrafo G'(V',E') com $T \subseteq V'$ de forma a minimizar a $\sum_{e \in E'} c_e$. Para exemplificar, considere o grafo apresentado na Figura 3.1 (a) que representa uma instância do PSG, onde os vértices não terminais estão representados pelos círculos e os terminais por quadrados. O grafo dado na Figura 3.1 (b) destaca a solução ótima para esta instância. O conjunto formado pelas arestas em destaque e pelos vértices onde as mesmas incidem é chamado *Árvore de Steiner*.

São várias as aplicações práticas deste problema e destacam-se as seguintes:

- Roteamento de ligações elétricas em projetos de circuitos VLSI, onde as arestas do grafo representam as trilhas onde podem ser passadas energia elétrica e a interceção destas trilhas são os vértices. Deseja-se interligar alguns componentes para que o *chip* execute sua função. Para tanto, o custo destas ligações deve ser mínimo.

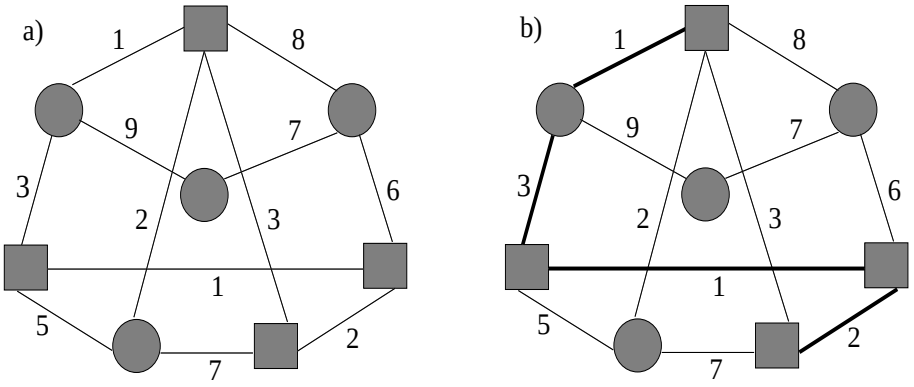

Figura 3.1: (a) Instância do problema *Steiner* em grafos; (b) árvore de *steiner* correspondente.

- Projeto de redes telefônicas, ferroviárias, entre outras. O objetivo é conectar os terminais que serão servidos de maneira eficiente, mesmo que para isto utilizem-se outras bifurcações. Os canais de comunicação e os terminais são, respectivamente, o grafo e o conjunto de vértices terminais.

3.2 O algoritmo B&B para o Problema de *Steiner* em Grafos

Como já visto na seção 2.2, o algoritmo B&B resolve problemas de otimização combinatória construindo uma enumeração de árvores, onde cada nó da árvore representa uma região viável do problema, ou seja, um espaço de busca que possui uma solução para o problema. A árvore enumera implicitamente todas as soluções possíveis da instância. A solução ótima é obtida percorrendo todos os nós da árvore.

Neste trabalho, o espaço de solução é delimitado por um limite superior, chamado primal, e um limite inferior, chamado dual. O algoritmo *Branch-and-Bound* da Figura 3.2 apresenta os procedimentos de ramificação e poda na árvore e tem como parâmetro de entrada um grafo G. Cada chamada, realizada recursivamente durante a solução dos nós, define mais um nó na árvore. A variável global *SolOtima* armazena o valor mínimo do primal encontrado. As variáveis LD e LP são, respectivamente os limites dual e primal. O procedimento inicia atribuindo novos valores para LP e LD. Se o valor de LP é menor do que a solução armazenada em *SolOtima*, o valor de *SolOtima* é atualizado com LP. Se LD é maior que o valor de LP então a melhor solução para aquele espaço foi encontrada. Caso contrário, o espaço de solução será particionado em G_1 e G_2 e o *Branch-and-Bound* será chamado recursivamente para cada um destes grafos.

```
procedure Branch-and-Bound(G)
 1: SolOtima = ∞;
 2: LP ← limite superior para G;
 3: LD ← limite inferior para G);
 4: se (LP < SolOtima) então
 5:    SolOtima = LP;
 6: fim se
 7: se (LD > LP) então
 8:    retorne;
 9: senão
10:    Particione G;
11:    Branch-and-Bound (G1);
12:    Branch-and-Bound (G2);
13: fim se
fim.
```

Figura 3.2: Algoritmo *Branch-and-Bound*

No algoritmo *Branch-and-Bound*, sendo G a instância que se deseja solucionar, a árvore de enumeração é construída como segue: o grafo G é a raiz da árvore binária. Um vértice v não terminal é escolhido em G. A primeira ramificação, filho da esquerda da raiz, é um grafo G_1, onde v se torna um vértice terminal. A segunda ramificação, filho da direita da raiz é uma grafo $G_2 = G - (v + I_v)$, onde I_v consiste no conjunto de todas as arestas incidentes no vértice v. Sobre cada ramificação é aplicado o algoritmo *Branch-and-Bound*, que produz novas ramificações até que o valor do dual seja maior que o valor do primal. A Figura 3.3 exemplifica este procedimento.

Um nó folha da árvore é aquele que possui o valor do dual maior ou igual ao primal. Assim, os valores limites são utilizados com o intuito de podar a árvore e diminuir o espaço de busca. A qualidade dos limites é o que determina a eficiência do algoritmo, ou seja, quanto menor o espaço de busca, menos nós serão gerados e conseqüentemente o tempo de execução do algoritmo será reduzido. O limite inferior é uma estimativa otimista da solução ótima do problema e em muitos casos é obtido a partir da relaxação linear do problema. Algoritmos duais fornecem bons valores para o limite inferior da solução e, no melhor caso, permitem provar a otimalidade de uma solução primal conhecida. Os algoritmos utilizados para encontrar os limites estão melhores descritos em [Uchoa 2001], [Uchoa 2002] e [Werneck 2001].

As ramificações são resolvidas independentes uma das outras o que sugere a paralelização da técnica. Facilmente pode-se perceber que a única informação importante para um nó é o valor do primal, pois evita a avaliação desnecessária de uma subárvore. Os procedimentos de distribuição inicial de carga, difusão do primal, detecção de terminação

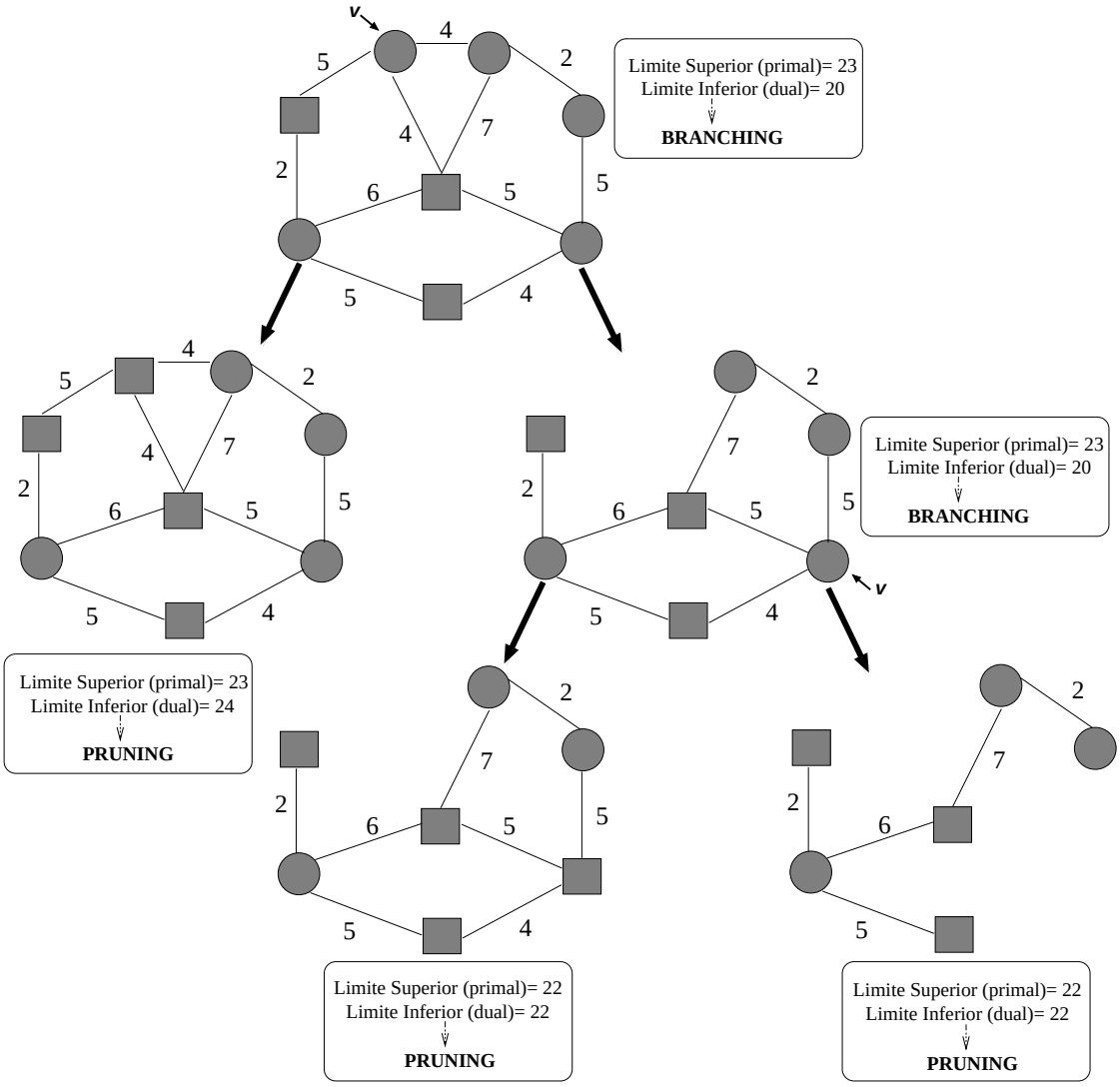

Figura 3.3: Árvore formada pela execução do algoritmo *Branch-and-Bound* apresentado na Figura 3.2

e balanceamento de carga utilizados pelo B&B distribuído estão descritos detalhadamente no próximo capítulo. Contudo, para facilitar explicações posteriores alguns considerações estão descritas nos ítens abaixo:

- Como visto nesta seção, o B&B organiza a busca por soluções através de uma enumeração de árvore binária. A solução consiste em uma busca em profundidade, sendo que, os nós da direita das ramificações são armazenados em uma fila para que sejam resolvidos posteriormente. Os nós que pertencem a esta fila são chamados carga acumulada.

- Armazenar um nó na fila significa armazenar o caminho do nó na árvore B&B e não armazenar o problema (o grafo) que aquele nó possui. Este caminho consiste de números inteiros e é duas vezes maior que a profundidade do nó na árvore. Assim, o

tamanho das mensagens trocadas entre os processos não causam qualquer aumento no *overhead* de comunicação.

Capítulo 4

Branch-and-Bound distribuído aplicado ao Problema de Steiner em Grafos

O B&B induz uma enumeração de árvore onde cada ramificação pode ser resolvida quase independente uma das outras. Esta característica faz com que o B&B seja muito apto ao paralelismo.

O algoritmo B&B distribuído utilizado neste trabalho, foi composto inicialmente pelos procedimentos de distribução inicial, difusão do primal e detecção de terminação, propostos inicialmente por Lúcia M.A. Drummond et al em [Drummond et al. 2004]. Posteriormente, foram adicionados os procedimentos de balanceamento de carga e tolerância a falhas propostos por Lúcia M. A. Drummond et al em [Drummond et al. 2005]. Apesar dos bons resultados apresentados pelo algoritmo, algumas questões a respeito do balanceamento de carga não foram considerados, como por exemplo, tamanho das subtarefas e desempenho dos processadores durante a transferência de carga. Neste trabalho são propostas e analisadas outros três procedimentos de balanceamento de carga com o objetivo de melhorar a eficiência paralela do algoritmo B&B aplicado ao PSG. Vale ressaltar que, neste trabalho, não foi utilizado o procedimento de tolerância a falhas.

Antes da descrição dos procedimentos, são necessárias algumas considerações a respeito da aplicação paralela:

- A distribuição das tarefas entre os processadores é estática. A quantidade de processos e máquinas que participam da aplicação é definida inicialmente. A cada processador é alocado um processo que permanece ativo até o final da execução.
- A cada processo é atribuída uma identificação, $i \in [0..m-1]$, onde m corresponde a quantidade de processos que participam da aplicação, sendo que, as identificações

de processos que pertencem ao mesmo *cluster* são consecutivas.

- Os *clusters*, como os processos, recebem identificações únicas do intervalo $[0..n-1]$, onde n é o número de *clusters*.

- A cada *cluster* está associado um processo líder, l_c, sendo $0 \leq c < n$. O líder l_c é o processo de menor identificação do *cluster* c. Os líderes têm por objetivo detectar a terminação do algoritmo e podem assumir funções especiais nas estratégias de balanceamento de carga. Apesar de possuírem funções extras, os líderes atuam como qualquer outro processo durante a aplicação, portanto, não podem ser comparados a processos mestres.

- A carga ou tarefa a ser realizada pelo processo consiste na solução de uma subárvore da árvore do *Branch-and-Bound*.

4.1 Distribuição inicial

O procedimento chamado *Distribuição Inicial* é responsável pela distribuíção inicial de carga entre os processos, sendo realizada da seguinte forma: o processo de identificação igual a 0 (líder da aplicação) inicia a execução da primeira subárvore. A cada operação de ramificação, envia uma mensagem contendo uma subárvore para o *cluster*, cujo identificador é calculado por $next\_cluster = c + 2^{level}$, onde *level* indica a profundidade do nó na árvore B&B que o líder do *cluster* c executa. Este mesmo procedimento é realizado por todos os outros líderes dos *clusters* após receberem a primeira ramificação. Quando *nextcluster* atinge um valor igual a $n-1$, os processos iniciam o compartilhamento local de carga, ou seja, compartilham sua carga com os processos pertencentes a seu próprio *cluster*. Isto implica que *level* será reiniciado com zero para os líderes, e cada processo i enviará um novo nó ao processo $next\_process = i + 2^{level}$.

Para exemplificar, considere um ambiente *Grid* formado por quatro *clusters*, $c_0 = \{0, 1, 2, 3\}$, $c_1 = \{4, 5\}$, $c_2 = \{6, 7\}$ e $c_3 = \{8, 9\}$. Inicialmente, o processo l_0, líder de c_0, executa o nó raiz da árvore e envia o nó da direita ao processo l_1, líder de c_1. Em sua segunda ramificação, o processo l_0 envia carga ao l_2. O processo l_1 executa o nó recebido e envia a sua primeira ramificação a l_3. Logo, inicia-se a divisão de carga dentro dos *clusters*. No c_0, o processo 0, envia os nós da direita do terceiro nível e quarto nível da subárvore para os processos 1 e 2. O processo 1 ramifica e envia ao processo 3. Semelhante ao c_0, em c_1, o processo 4 envia carga ao 5. O processo 6 envia carga para o processo 7 em c_2 e

o processo 8 envia carga ao 9 em c_3. A Figura 4.1 apresenta esta divisão. Os processos estão representados pelos círculos sendo que o número no interior indica a identificação do processo. Os círculos mais escuros representam os líderes de *cluster*. Vale lembrar que o processo $l_0 = 0$ é o processo líder da aplicação. A mesma definição será utilizada para as demais figuras deste trabalho.

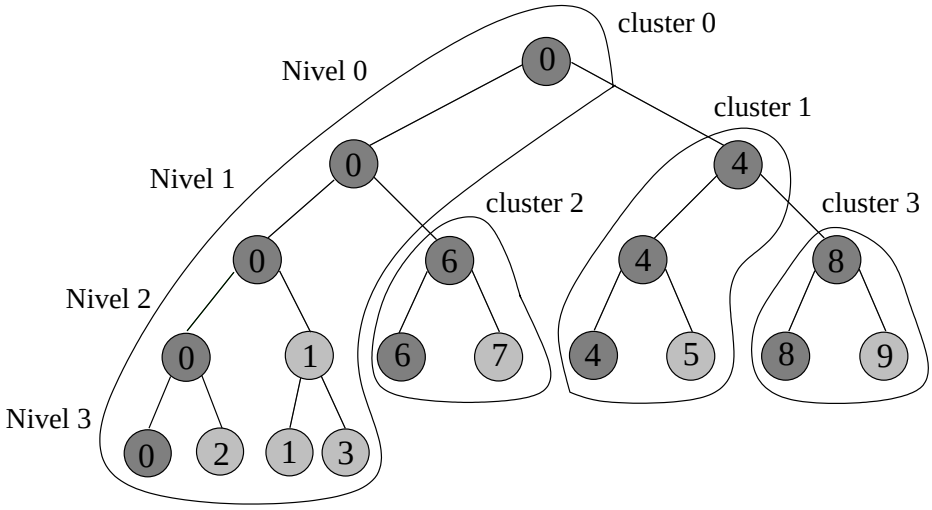

Figura 4.1: Distribuição inicial de carga

O envio de carga de um processo i para um processo k se dá através de uma mensagem denominada *BRANCH*. Esta mensagem contém o caminho do nó raiz da subárvore que será executada por k. É importante notar que, durante a distribuição inicial, se o nó executado por i não se ramifica, ou seja, se i não possuir carga a ser eviada para k, envia uma mensagem *AWAKE* com o intuito de indicar ao processo k que o procedimento de balanceamento de carga deverá ser iniciado. Este procedimento está descrito na seção 4.3.

O algoritmo da Figura 4.2 ilustra o procedimento de distribuição inicial. A notação $Entrada = NULL$ indica que este procedimento foi disparado pelo processo 0 do c_0 e $Entrada = BRANCH$ indica que o procedimento foi inciado por um processo i mediante o recebimento de uma mensagem, ou seja, de uma ramificação.

4.2 Difusão do primal

A difusão do valor do primal entre os processos participantes da aplicação é essencial para limitar o crescimento das subárvores executadas por cada processo. Este procedimento é realizado de forma hierárquica e é ativado quando um processo encontra um melhor valor para o primal. Mais especificadamente, o procedimento é iniciado quando um menor valor

procedure `Distribuição Inicial`

1: $level = 0$;
2: **se** ((Entrada = mensagem $NULL$ e $(c = 0)$ e $(i = 0)$) ou (Entrada = mensagem $BRANCH$)) **então**
3: **enquanto** $next\_cluster < n$ **faça**
4: Executa $Branch - and - Bound$;
5: $level = level + 1$;
6: $next\_cluster = c + 2^{level}$;
7: Envia $BRANCH$ para next_cluster;
8: **fim enquanto**
9: $level = 0$;
10: $next\_process = i + 2^{level}$;
11: **enquanto** $next\_process < tam_c$ **faça**
12: Executa $Branch - and - Bound$;
13: $level = level + 1$;
14: $next\_process = i + 2^{level}$;
15: Envia $BRANCH$ para next_process;
16: **fim enquanto**
17: **fim se**

fim.

Figura 4.2: Procedimento de distribuição inicial

para a variável *SolOtima* do algoritmo apresentado na Figura 3.2 é encontrado por um processo i. A difusão do *PRIMAL* pode ser descrita da seguinte forma: o processo i, após atualizar o valor de sua variável, envia uma mensagem, chamada *PRIMAL*, para todos os processos pertencentes a seu *cluster*. Quando os processos recebem a mensagem, se o valor de *SolOtima* é maior do que o valor recebido, atualizam o valor de *SolOtima*. Caso contrário a mensagem é descartada. Quando o processo líder do *cluster* recebe a mensagem, além de atualizar, se necessário, o valor de *SolOtima*, envia a mensagem recebida para todos os líderes dos outros *clusters*, que por sua vez, fazem a verificação (e atualização) e propagam a mensagem recebida para todos os processos pertencentes aos seus *clusters*. Este procedimento é descrito no algoritmo apresentado na Figura 4.3, sendo que, c denota o *cluster* ao qual o processo i pertence, l_c o líder do *cluster* c, onde, $0 \leq c \leq n-1$ e $0 \leq k \leq tam_c$ e tam_c é o número de processos do *cluster* c. Vale ressaltar que o algoritmo segue diferentes passos dependendo do evento ou tipo de mensagem recebida e do processo que o executa. Assim, $Entrada = NULL$ significa que este procedimento foi disparado por um processo que encontrou um melhor valor para o limite PRIMAL (no caso, o processo i) e não pelo recebimento de uma mensagem. Por outro lado, quando $Entrada = mensagem\ PRIMAL$ um processo iniciou a sua participação no procedimento devido ao recebimento de uma mensagem do tipo *PRIMAL* de outro processo. Os caracteres subescritos que acompanham *PRIMAL* indicam qual o processo

```
procedure Difusão do Primal
 1: se (Entrada = NULL) então
 2:    se (PRIMAL < SolOtima) então
 3:       SolOtima = PRIMAL;
 4:       para todo k ∈ c tal que 0 ≤ k ≤ tam_c e k ≠ i faça
 5:          Envia PRIMAL para k;
 6:       fim para
 7:    fim se
 8: fim se
 9: se (Entrada = mensagem PRIMAL_k) então
10:    se (PRIMAL < SolOtima) então
11:       SolOtima = PRIMAL;
12:       se (i = l_c) então
13:          para todo l_w tal que 0 ≤ w ≤ n − 1 e w ≠ c faça
14:             Envia PRIMAL para l_w;
15:          fim para
16:       fim se
17:    fim se
18: fim se
19: se (Entrada = mensagem PRIMAL_lw) então
20:    se (PRIMAL < SolOtima) então
21:       SolOtima = PRIMAL;
22:       para todo k tal que 0 ≤ k ≤ tam_c e k ≠ l_c faça
23:          Envia PRIMAL para k;
24:       fim para
25:    fim se
26: fim se
fim.
```

Figura 4.3: Procedimento de difusão do primal

enviou a mensagem. Esta notação será utilizada para todos os algoritmos contidos neste trabalho.

A Figura 4.4 exemplifica o procedimento, onde em 4.4 (a), o processo 1, após encontrar um novo valor primal, propaga-o para os processos de seu *cluster* através da mensagem *PRIMAL*; em (b), o processo l_0 envia a mensagem aos os líderes l_1 e l_2; e finalmente, l_1 e l_2, difundem a mensagem para os processos pertencentes a seus *clusters*, (c).

4.3 Balanceamento de carga

Conforme descrito no Capítulo 2, o balanceamento de carga é essencial quando combinados a plataforma *Grid* de execução paralela e o algoritmo B&B. A Tabela 4.3 comprova a necessidade de estratégias eficientes de balanceamento em uma aplicação paralela, pois possibilita a comparação entre os tempos de execução (tempo de relógio de parede) do

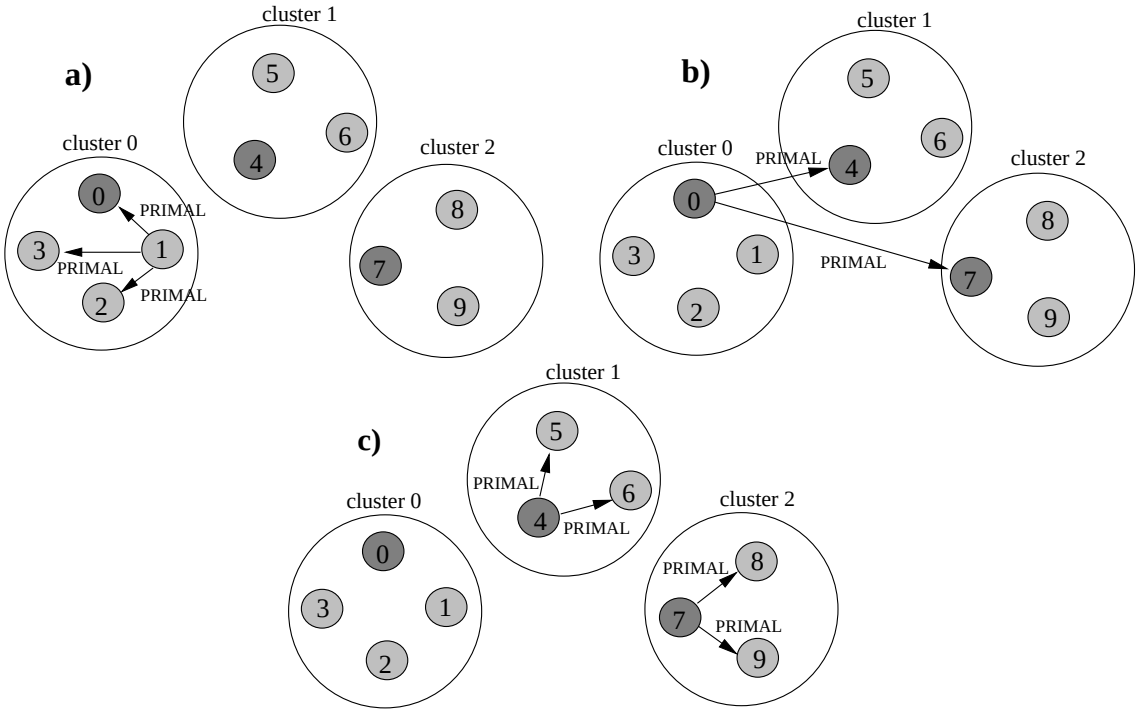

Figura 4.4: Difusão do primal

algoritmo seqüencial e de duas versões paralelas. Na primeira coluna são apresentadas as instâncias utilizadas; na segunda coluna (SEQ) os tempos (em horas) do algoritmo seqüencial proposto por [Uchoa 2001]; e na terceira (PAR1) e quarta colunas (PAR2), são apresentados os tempos de execução das versões paralelas sem a utilização de qualquer estratégia de balanceamento de carga e utilizando a estratégia de balanceamento *Global*, respectivamente, conforme apresentado em [Drummond et al. 2005]. As duas últimas colunas (SP1 e SP2), apresentam os *speedups* das versões paralelas. Estes testes foram executados em um *cluster* com 16 processadores Pentium 2.4 GHz exclusivamente dedicados para a aplicação. O algoritmo seqüencial, por tomar decisões aleatórias, foi executado cinco vezes com diferentes sementes e o resultado apresentado é a média dos tempos de relógio obtidos.

Instâncias	SEQ	PAR1	PAR2	SP1	SP2
i640-211	72,14	14,08	3,68	5,12	19,58
i640-212	5,75	0,73	0,15	7,92	39,36
i640-213	5,16	0,83	0,34	6,23	15,07
i640-214	52,02	7,90	1,51	6,58	34,47
i640-215	21,98	4,73	1,18	4,65	18,69

Tabela 4.1: Comparação entre B&B seqüencial, paralelo e paralelo com balanceamento de carga.

Lúcia M. A. Drummond em [Drummond et al. 2005] desenvolveram duas estratégias de balanceamento de carga *receiver-initiated* denominadas, *Global* e *Híbrida*. Nas duas estratégias, a transferência de carga entre processos consiste em uma mensagem do tipo *BRANCH* contendo um único nó da árvore (raiz de uma subárvore). O critério FIFO (primeiro a chegar é o promeiro sair) é utilizado na escolha do nó a ser enviado. Note que nenhuma consdideração sobre o problema ou desempenho dos processadores são considerados. A estratégia *Global*, diferente da *Híbrida*, não considera a hierarquia de comunicação entre os processos. Na primeira estratégia, quando um processo i não possui mais subárvores a serem resolvidas, envia uma mensagem do tipo *LOADREQUEST* ao processo $k_1 = (i + m - 1)\ mod\ m$, onde m é o número de processos que participam da aplicação. k_1, ao receber a requisição de carga enviará uma mensagem *BRANCH* contendo um nó da sua subárvore para i ou, se k_1 também estiver ocioso, enviará uma mensagem *IDLE* indicando que não possui carga pendente. O processo i se receber carga, continua a execução da aplicação, caso contrário, envia pedido de carga ao processo k_2 tal que $k_2 = (i + m - 2)\ mod\ m$, posteriormente a $k_3 = (i + m - 3)\ mod\ m$ e assim por diante até que obtenha carga ou envie pedido a todos os processos da aplicação. Esta última opção tem como conseqüência o inicío do procedimento de terminação da aplicação, melhor descrita na próxima seção. Assim, nesta estratégia, um processo ocioso não distingue se o processo a quem irá requisitar carga pertence ou não a seu *cluster* porque envia pedidos de carga seguindo uma ordem cíclica envolvendo todos os processos participantes da aplicação. Vale ressaltar que o envio de pedidos de carga segue a ordem anti-horária porque os processos anteriores a i receberam, inicialmente, uma subárvore maior que a dos processos posteriores. Esta afirmação não considera prováveis podas e não é verdadeira para o processo 0.

Considerando que ζ_i é o conjunto de nós ainda não tratados (carga acumulada) do proceso i, k o próximo processo a quem i irá pedir carga, sendo $0 \leq vez \leq m - 1$, e, reforçando que m e n são, respectivamente, o número de processos e o número de *clusters* participantes da aplicação, o algoritmo apresentado na Figura 4.5 descreve o procedimento *Global* de balanceamento de carga. A identificação dos processos a que i irá enviar requisição de carga é controlado pela variável vez, que quando tem seu valor igual a m, ativa o procedimento de suspeita de témino da aplicação enviando mensagem *SUSPECTEND* para todos os processos participantes da aplicação. Esta mensagem indica que i entra em estado de terminação por não conseguir carga. Note que i somente reenvia a mensagem *LOADREQUEST* após o recebimento de uma mensagem *IDLE*, ou seja, após o *feedback* do processo k. Vale ressaltar que a notação do processo k acompanhado de um

```
procedure Global
 1: vez = 1;
 2: se (Entrada = NULL) então
 3:    se (ζ_i = ∅) então
 4:       k = (i + m − vez) mod m;
 5:       Envia LOADREQUEST para k;
 6:    fim se
 7: fim se
 8: se (Entrada = mensagem IDLE_k) então
 9:    vez = vez + 1;
10:    se (vez < m ) então
11:       k = (i + m − vez) mod m;
12:       Envia LOADREQUEST para k;
13:    senão
14:       para todo k tal que 0 ≤ k ≤ m − 1 e k ≠ i faça
15:          Envia SUSPECTEND para k;
16:       fim para
17:       retorne;
18:    fim se
19: fim se
20: se (Entrada = mensagem BRANCH_k) então
21:    Executa Branch-and-Bound;
22: fim se
fim.
```

Figura 4.5: Procedimento de balanceamento de carga - *Global*

índice é utilizada somente para melhor especificar o funcionamento da técnica. Já que a atribuição da identificação de k é realizada através do cálculo mostrado acima ele se torna dispensável. Esta notação será utilizada nos demais algoritmos apresentados neste trabalho.

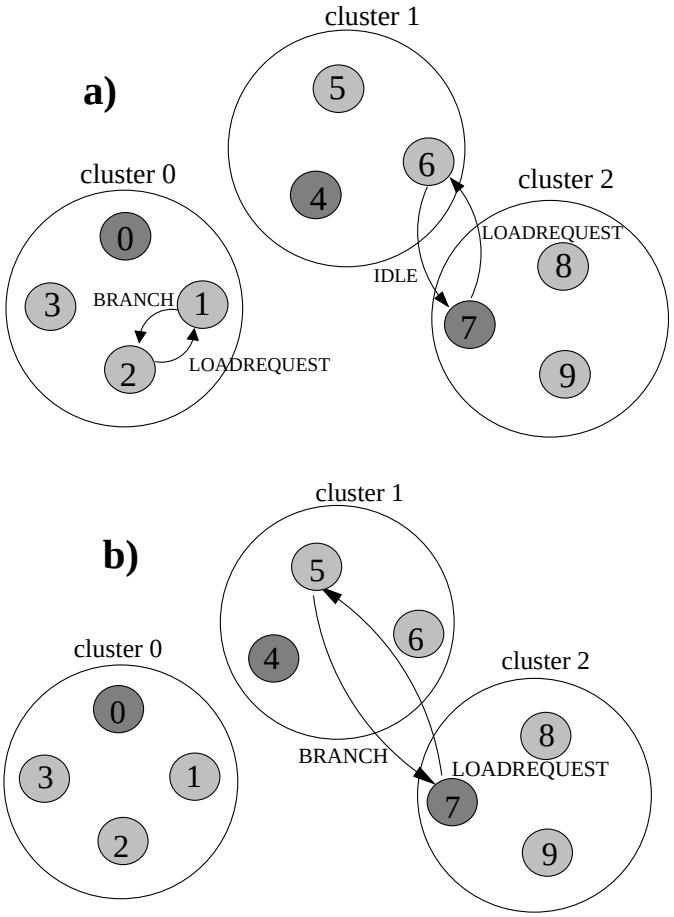

Figura 4.6: Balanceamento de carga - *Global*

A Figura 4.6 exemplifica a estratégia *Global*, mostrando a execução do algoritmo apresentado na Figura 4.6 para os processos 2 e 7, que ao se tornarem ociosos, enviam pedido de carga aos processos de identificação imediatamente menores, 1 e 6 (Figura 4.6 (a)). A execução do procedimento de balanceamento de carga termina para o processo 2 quando recebe um *BRANCH* do processo 1 logo no primeiro passo. Já para o processo 7, foi necessário enviar pedido de carga a outro processo, uma vez que o processo 6 enviou uma mensagem *IDLE*. Logo, na Figura 4.6 (b), o processo 7 envia mensagem *LOADREQUEST* ao processo 5, que lhe responde com uma mensagem *BRANCH*. Note que, ao receber um *BRANCH*, um processo termina a execução do procedimento de balanceamento de carga *Global* e inicia a execução do algoritmo *Branch-and-Bound*.

Na estratégia *Híbrida*, considerando que processos de um mesmo *cluster* definem um grupo para fins de balanceamento de carga, a transferência de carga é inicialmente local (entre processos pertencentes a um mesmo *cluster*) e torna-se global somente quando todos os processos do grupo se tornam ociosos. Um processo i pertencente a um *cluster* c quando se torna ocioso, envia $LOADREQUEST$ ao processo $k_1 = |(i - l_c - 1)| \ mod \ tam_c + l_c$,

onde tam_c é igual ao número de processos pertencentes a c. Como na estratégia *Global*, se i não receber carga, enviará pedido a $k_2 = |(i - l_c - 2)|\ mod\ tam_c + l_c$, posteriormante ao processo $k_3 = |(i - l_c - 3)|\ mod\ tam_c + l_c$ e assim por diante até que obtenha carga ou envie pedido a todos os processos pertencentes a seu *cluster*. Se i não consegue obter carga, envia uma mensagem *SUSPECTEND* a l_c, indicando que alcançou o estado de terminação (o processo assume que não existe mais tarefa a ser realizada). Quando l_c receber a mensagem *SUSPECTEND* de todos os processos pertencentes a seu *cluster* envia uma mensagem *LOADREQUEST_ CLUSTER* ao líder do *cluster* $w_1 = (c + n - 1)\ mod\ n$. Se l_{w_1} não possuir carga, tentará conseguir carga com os processos de seu *cluster*. Se conseguir, responderá a l_c com uma mensagem *BRANCH_ CLUSTER* contendo um nó de sua subárvore ou, caso contrário, com uma mensagem *IDLE_ CLUSTER* indicando que o seu *cluster* também está ocioso. Se l_c receber um *BRANCH_ CLUSTER* distribuirá a carga recebida entre os processos de seu *cluster*, como no procedimento de distribuição incial. Caso contrário, se receber a mensagem *IDLE_ CULSTER* tentará conseguir carga com o líder do *cluster* $w_2 = (c + n - 2)\ mod\ n$, posteriormente com o líder do *cluster* $w_3 = (c + n - 3)\ mod\ n$, assim por diante até que obtenha carga ou envie pedido a todos os líderes que participam da aplicação. Este último caso implica em uma onda de suspeita de témino da aplicação.

Para evitar ondas de suspeitas de término desnecessárias, as mensagem *NOTNOW* e *NOTNOW_ CLUSTER* são enviadas por um processo k em resposta a um *LOADREQUEST* e *LOADREQUEST_ CLUSTER*. Um processo k envia ao processo i uma mensagem do tipo *NOTNOW*, se k está executando um nó e não possui carga a ser enviada. Mais especificadamente, k está executando um nó que poderá gerar outros nós, logo, k, apesar de estar com a fila vazia, garante a i que não deve iniciar uma onda de suspeita, mas sim, enviar pedido de carga a outro processo. Este procedimento também é utilizado na estratégia *Global*.

O algoritmo apresentado na Figura 4.7 descreve este procedimento. Note que as condições das linhas 21 e 30 só serão executadas pelos líderes.

Para exemplificar a estratégia considere a figura 4.8 que está dividida em quatro partes. Na primeira delas, 4.8 (a), o processo 2 enviou pedido de carga, inicialmente, ao processo 1 que respondeu com uma mensagem do tipo *IDLE* indicando que não possui carga. O mesmo fato acontece quando 2 tenta conseguir carga com processo 0 e com o processo 3. É importante notar que o processo 2 não faz *broadcasting* da mensagem *LOADREQUEST*, pelo contrário, somente envia novo pedido de carga mediante a che-

procedure `Híbrida`

1: $vez = 1$;
2: **se** (Entrada = $NULL$) **então**
3: **se** ($\zeta_i = \emptyset$) **então**
4: $k = |(i - l_c - vez)| \ mod \ tam_c + l_c$;
5: Envia LOADREQUEST para k;
6: **fim se**
7: **fim se**
8: **se** (Entrada = mensagem $IDLE_k$) **então**
9: $vez = vez + 1$;
10: **se** ($vez < m$) **então**
11: $k = |(i - l_c - vez)| \ mod \ tam_c + l_c$;
12: Envia LOADREQUEST para k;
13: **senão**
14: Envia SUSPECTEND para l_c;
15: retorne;
16: **fim se**
17: **fim se**
18: **se** (Entrada = mensagem $BRANCH_k$) **então**
19: Executa *Branch-and-Bound*;
20: **fim se**
21: **se** (Entrada = mensagem $SUSPECTEND_k$) **então**
22: **se** (numero de suspeitas recebidas = tam_c) **então**
23: $vez = 1$;
24: **se** ($vez < n$) **então**
25: $w = (c + n - vez) \ mod \ n$;
26: Envia LOADREQUEST_CLUSTER para l_w;
27: **fim se**
28: **fim se**
29: **fim se**
30: **se** (Entrada = mensagem $BRANCH\_CLUSTER_{l_w}$) **então**
31: Executa *Distribuição Inicial*;
32: **fim se**
33: **se** (Entrada = mensagem $IDLE\_CLUSTER_{l_w}$) **então**
34: $vez = vez + 1$;
35: **se** ($vez < n$) **então**
36: $w = (c + n - vez) \ mod \ n$;
37: Envia LOADREQUEST_CLUSTER para l_w;
38: **senão**
39: **para todo** l_w tal que $0 \leq w \leq n - 1$ e $w \neq c$ **faça**
40: Envia SUSPECTEND_CLUSTER para l_w;
41: **fim para**
42: **fim se**
43: **fim se**

fim.

Figura 4.7: Procedimento de balanceamento de carga - *Híbrida*

gada da resposta do pedido anterior. Em 4.8 (b), o processo 2, depois de ter tentado conseguir carga com todos os processos de seu *cluster*, envia uma mensagem *SUSPECTEND* para 0, líder de seu *cluster*. Paralelamente, no *cluster* 2 o processo 8 envia pedido de carga ao processo 7 e recebe carga a ser tratada. Depois de receber a mensagem *SUSPECTEND* de todos os processos de seu *cluster*, 0 envia uma mensagem de *LOADREQUEST_CLUSTER* ao processo 7 (4.8 *c*). Finalmente em 4.8 *d*, 0 distribui, como no procedimento de *Distribuição Inicial*, a carga recebida entre os processos de seu *cluster*.

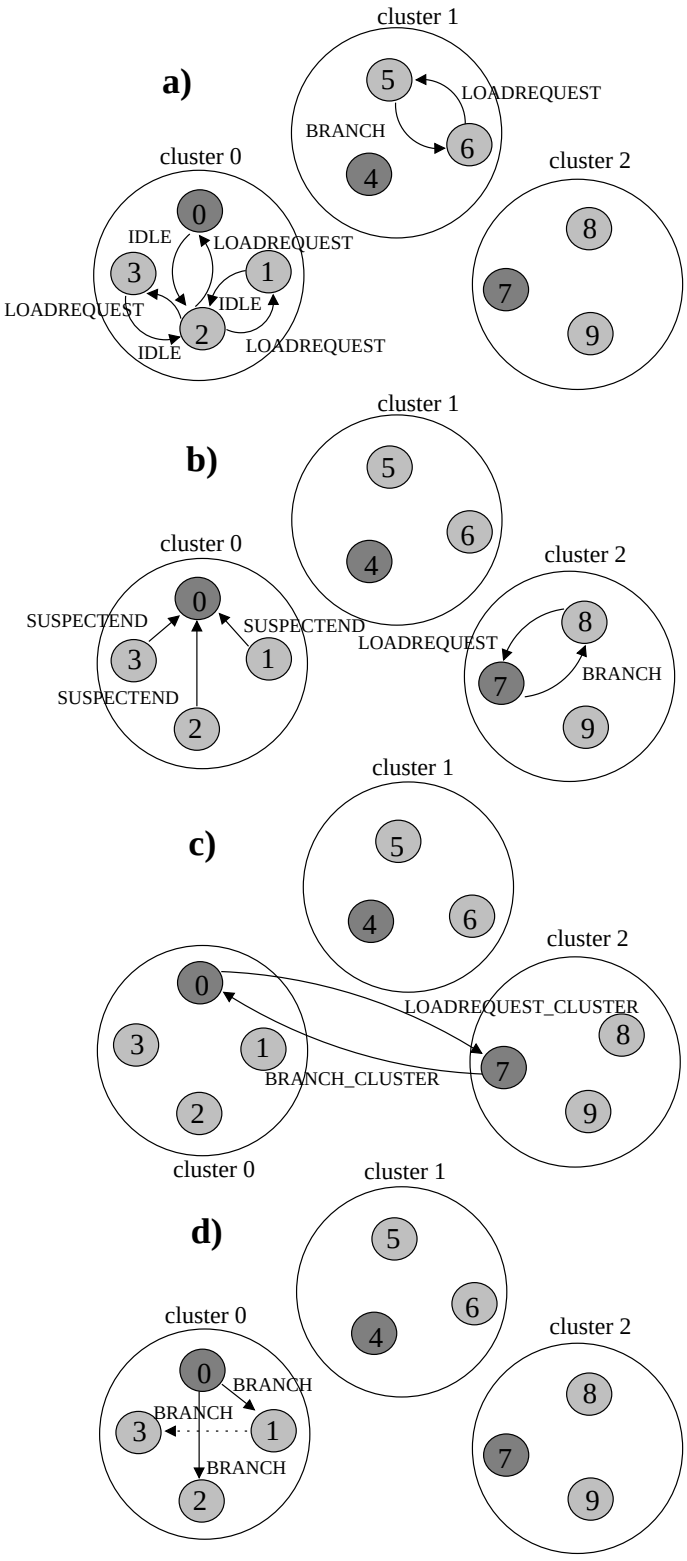

Figura 4.8: Balanceamento de carga - *Híbrida*

4.4 Detecção de terminação

Este procedimento, como o de difusão do primal, também acontece de forma hierárquica e já foi inicialmente apresentado nas seções anteriores. Na estratégia *Hibrida*, quando um processo atinge sua condição de terminação local, isto é, não é capaz de obter subárvores com os processos pertencentes a seu *cluster*, informa este fato ao seu líder enviando uma mensagem *SUSPECTEND*. Quando todos os processos deste *cluster* atingirem a mesma condição e o líder não for capaz de obter subárvores de outros líderes, ou seja, recebe *IDLE_ CLUSTER* de todos os líderes, considera-se que todas as subárvores já foram tratadas e a aplicação atingiu a condição de terminação. Assim, o líder do *cluster* em questão envia uma mensagem *SUSPECTEND_ CLUSTER* aos outros líderes indicando que alcançou a terminação. O processo 0, líder da aplicação, depois de receber *SUSPECTEND_ CLUSTER* de todos os líderes, envia-lhes uma mensagem *TERMINATE*, informando-os que a aplicação deve ser terminada e estes, propagam a mensagem aos processos de seu *cluster*. Na estratégia *Global*, um processo quando não consegue carga após $m-1$ pedidos, informa aos demais processos que atingiu sua condição de terminação através do envio da mensagem *SUSPECTEND* e termina de fato quando todos os demais processos atingem a mesma condição, ou seja, depois do processo 0 receber *SUSPECTEND* de todos os processos e, conseqüentemente, enviam um *broadcast* da mensagem *TERMINATE*.

Para melhor exmplificar, a Figura 4.9 apresenta a terminação do algoritmo quando utilizada a estratégia *Hibrida*. Em 4.9 (a), apresenta-se a terminação entre os *clusters*, onde os processos líderes 4 e 7 enviam supeita de término ao processo 0, que é líder da aplicação. A Figura 4.9 (b) apresenta a teminação da aplicação.

Um processo, na estratégia *Global*, inicia a suspeita de terminação após um ciclo de $m-1$ pedidos de carga, ou seja, depois que envia pedido a todos os processos pertencentes a aplicação. Poderia-se assumir diferentes tamanhos para o cliclo, por exemplo, poderia-se considerar que depois de $(m-1/2)+1$ envios de pedidos a aplicação estaria próxima do fim. Assim, os processos adiantariam o procedimento de detecção de terminação, diminuindo, assim, o tempo de execução do B&B. Entretanto, depois de vários testes, foi comprovado que à medida que se diminui o tamanho do ciclo, ondas de suspeita de término são iniciadas sem necessidade, aumentando consideravelmente o *overhead* de comunicação entre os processos e diminuindo a eficiência do algoritmo [Baoukov e Soverik 1999].

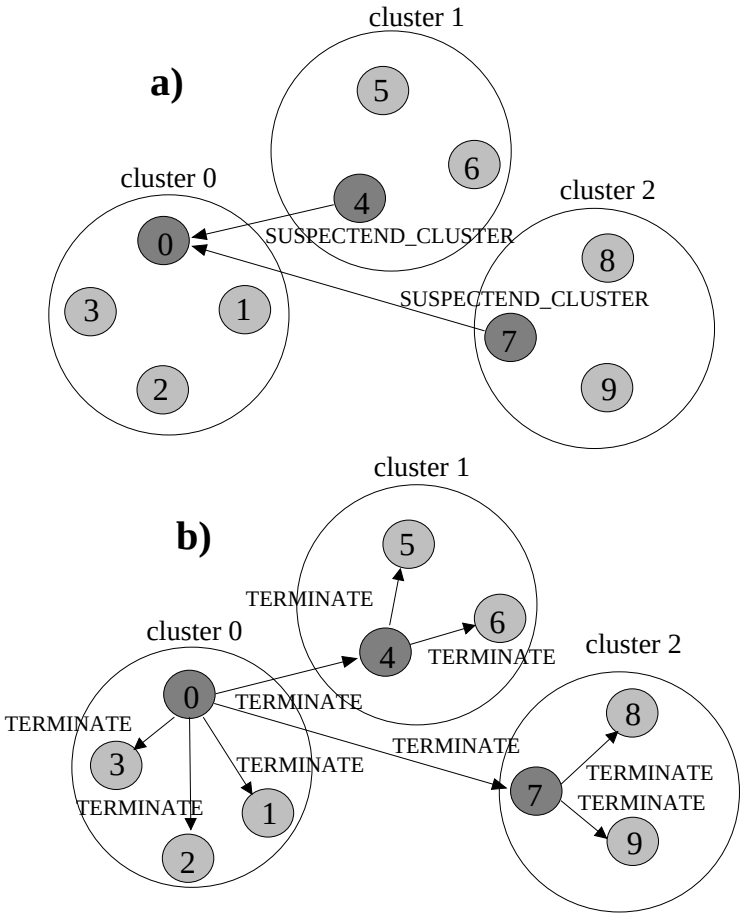

Figura 4.9: Detecção de terminação

Capítulo 5

Estratégias de balanceamento de carga adaptativas

As estratégias desenvolvidas por [Drummond et al. 2005], *Global* e *Híbrida*, apresentadas no Capítulo 4, apesar de apresentarem bons resultados, não consideram informações do problema ou informações do ambiente durante a transferência de carga entre os processos. Este fato pode prejudicar consideravelmente a eficiência das estratégias porque a carga transferida entre os processos não são, normalmente, adequadas aos desempenhos dos mesmos.

Este capítulo descreve as estratégias de balanceamento desenvolvidas neste trabalho. As novas estratégias, chamadas *GlobalAdapt*, *HibridaAdapt* e *MeAdapt*, são consideradas adaptativas porque, além de considerarem o conceito de hierarquia de comunicação propostas por [Drummond et al. 2005], utilizam, em tempo de execução, informações do problema e do desempenho dos processadores para adaptar carga ainda pendente aos recursos disponíveis do ambiente.

5.1 Estratégias adaptativas

No Capítulo 2 foram descritas algumas características do ambiente e do algoritmo B&B que podem prejudicar o desempenho da aplicação. As estratégias desenvolvidas neste trabalho exploram estas peculiaridades com o intuito de diminuir a ociosidade dos processos e o número de mensagens trocadas pelos mesmos. As estratégias estão minuciosamente descritas nas subseções abaixo segundo os critérios de classificação, localização da execução do algoritmo e escopo do balanceamento de carga.

5.1.1 *GlobalAdapt* e *HibridaAdapt*

As estratégias *GlobalAdapt* e *HibridaAdapt* são baseadas, respectivamente, nas estratégias propostas por [Drummond et al. 2005], *Global* e *Híbrida*. Logo, em relação ao escopo de execução do balanceamento descrito na seção 2.1, as novas estratégias possuem a mesma classificação.

Na estratégia *HibridaAdapt*, igualmente à *Híbrida*, quando um processo i pertencente a um *cluster* c, termina a execução de sua subárvore, ou seja, se torna subcarregado, envia pedido de carga a um processo pertencente a seu próprio *cluster*. Se este não possuir carga, i envia novamente o pedido a outro vizinho e assim sucessivamente até obter carga. Caso todos os processos de seu *cluster* estejam ociosos envia ao líder de c uma mensagem indicando o ocorrido. O líder tendo conhecimento que não existe mais carga a ser resolvida em seu *cluster*, envia pedido de carga ao líder do *cluster* vizinho. Se o líder de c receber carga em resposta ao pedido, o mesmo redistribui a carga entre os processos de seu *cluster*, caso contrário, pede carga a outro líder e assim sucessivamente até que consiga carga ou, se não existir mais carga a ser resolvida, inicia a terminação do algoritmo. Na estratégia *GlobalAdapt*, como na *Global*, a transferência de carga ocorre independente da distância geográfica dos processos envolvidos no balanceamento. Um processo quando se torna subcarregado envia pedido de carga a qualquer processo participante da aplicação, independente de sua posição geográfica. Vale relatar que o pedido é sempre enviado de forma circular, iniciando a partir do processo que possui identificação imediatamente menor que a do processo em questão e continua até o recebimento de carga ou não exista mais carga a ser resolvida. Neste caso, o inicia a terminação do algoritmo.

A quantidade de carga a ser transferida entre os processos nas estratégias adaptativas dependem da informação do desempenho apresentado pelos processadores durante a execução da aplicação. Assim, tanto na estratégia *HibridaAdapt* como *GlobalAdapt*, quando um processo se torna ocioso, envia, juntamente com a mensagem de requisição de carga, a informação relativa ao desempenho do processador no qual está alocado. O processo que receber esta requisição irá dividir com o processo que enviou o pedido sua carga pendente, proporcionalmente aos desempenhos de tal forma que as subárvores são enviadas de acordo com os seus tamanhos. Entretanto, este valor não é conhecido *a priori*. Para tanto, os tamanhos das subárvores enraizadas pelos nós presentes na fila são obtidos a partir de estimativa.

A carga pendente ou carga acumulada representa o conjunto de nós ainda não explorados. Cada processo na distribuição inicial recebe o caminho do nó a ser explorado.

Este nó, depois de explorado, gera dois nós filhos. Os nós filhos também são explorados e produzirão, cada um, outros dois nós, e assim por diante, até que seja alcançado um nó folha. O B&B resolve a árvore utilizando a política de *busca em profundidade*, o processo sempre explora o nó da esquerda e armazena o caminho do nó da direita em uma fila. A carga pendente é simplesmente um conjunto de caminhos para nós que ainda não foram explorados, ou seja, são as raízes das subárvores da direita que futuramente serão resolvidas pelo processo. Portanto, por longo deste texto, a expressão "envio de subárvore" corresponde ao "envio do nó raiz da subárvore".

Os procedimentos utilizados por um processo para analisar o desempenho dos processadores e estimar o tamanho das subárvores durante o balanceamento estão minunciosamente descritos nas próximas seções.

5.1.1.1 Avaliação do desempenho dos processadores

A eficiência de um balanceamento de carga é inversamente proporcional ao tempo total em que os processos ficam ociosos durante a aplicação. Se durante a transferência não é enviada a quantidade ideal de carga, ou seja, uma quantidade de carga compatível com o poder computacional do processador onde o processo está alocado, existirão processos sobrecarregados e subcarregados ao mesmo tempo durante a execução, o que resultará em um baixo desempenho da aplicação.

Para avaliar o desempenho dos processadores durante a execução da aplicação, a seguinte estratégia foi adotada. Quando um processo i se torna ocioso, este envia uma mensagem *LOADREQUEST* contendo o valor M_i a um processo k, onde M_i correspondente à razão entre o número r de nós resolvidos no período de tempo t e o próprio tempo t. Sendo que t é o intervalo de tempo de relógio de parede iniciado a partir do recebimento de uma carga até o final da execução da mesma, lembrando que, k é escolhido utilizando o mesmo critério proposto por [Drummond et al. 2005].

O processo k, ao receber a mensagem, calcula M_k de forma análoga ao processo i. A análise destas informações, M_i e M_k, permite ao processo k apontar qual dos processadores tem apresentado melhor desempenho e dividir, proporcionalmente, a sua carga pendente com o processo i. Assim, considerando que a carga pendente de k é ζ_k, k responderá ao processo i com uma mensagem *BRANCH* contendo N nós, onde N é dado por:

$$N = M_i * \frac{\zeta_k}{M_i + M_k} \tag{5.1}$$

O valor de ζ_k é dado pela soma estimada dos nós das subárvores que serão geradas pelos nós raizes presentes na fila de carga pendente do processo k. Os N nós são escolhidos da seguinte forma: para cada nó da fila, calcula-se a estimativa do tamanho da subárvore que será gerada pelo mesmo. Os nós são ordenados em ordem decrescente em relação à estimativa e são retirados do primeiro para o último de modo que o valor da soma das estimativas seja o mais próximo possível de N. O cáculo da estimativa das subárvores está melhor descrito na próxima seção.

5.1.1.2 Cálculo da estimativa das subárvores

No algoritmo B&B, o tamanho da subárvore gerada por um nó não é conhecido a *priori*. Por este motivo, em uma transferência de carga entre processos é necessária uma maneira de estimar o tamanho das subárvores enviadas. Sem este cuidado, um processo que recebeu, por exemplo, uma única subárvore que gera uma grande quantidade de ramificações poderá ficar sobrecarregado enquanto outros processos que receberam uma maior quantidade de subárvores, que produzem um menor número de ramificações, poderão ficar ociosos muito mais rapidamente.

O cálculo da estimativa de um nó n é baseado nas soluções encontradas pelos nós ancestrais de n. A solução de um nó é o valor encontrado para a função objetivo do PSG e acontece quando o limite inferior, o dual, ultrapassa o valor do limite superior, o primal, no espaço de busca da solução. Sabe-se que quanto mais estreito o espaço de busca, ou seja, quanto mais próximos estiverem os valores dos limites dual e primal, é mais provável que a solução esteja próxima. Assim, a estimativa do tamanho da subárvore gerada por um nó n é baseada nos valores primal e dual obtidos e é dada por

$$E = \frac{primal - dual_n}{ganho} \tag{5.2}$$

onde, *primal* é a solução mínima encontrada até o momento para o problema; $dual_n$ é o valor do limite inferior calculado para o nó n; e *ganho* é um valor correspondente a quanto o espaço de busca diminuiu desde o nó raiz da árvore até n e é dado por:

$$ganho = ganho + \frac{dual_n - dual_{a_n}}{c_{nos}} \tag{5.3}$$

onde $dual_{a_n}$ é o valor dual gerado pelo nó pai de n, e c_{nos} é o número de nós explorados na subárvore onde n é um nó folha.

Nós	$dual_n$	$dual_{a_n}$	$primal$	$ganho$	E
1	7849	0	8115	0,00	-
2	7867	7849	8115	9,00	27,56
3	7867	7867	8115	9,00	27,56
4	7883	7867	8115	13,00	17,85
5	7883	7883	8115	13,00	17,85
6	7900	7883	8095	15,83	12,32
7	7904	7900	8095	16,40	11,64
8	7904	7904	8095	16,40	11,64
9	7904	7904	8095	16,40	11,64
10	7904	7904	8095	16,40	11,64
11	7904	7904	8095	16,40	11,64
12	7904	7904	8081	16,40	10,79
13	7904	7904	8077	16,40	10,55
14	7904	7904	8077	16,40	10,55
15	7904	7904	8077	16,40	10,55
16	7906	7904	8077	16,53	10,34
17	7943	7906	8077	18,71	7,16
18	7943	7943	8039	18,71	5,13
19	8048	7943	8039	24,23	-0,37

Tabela 5.1: Cálculo da estimativa para a instância de incidência *i320-211*

Para exemplificar o cálculo, considere a execução do algoritmo B&B paralelo para a instância *i320-211* [Koch et al. 2003], mostrada na Tabela 5.1.1.2. A subárvore apresentada é a primeira subárvore gerada pelo processo líder da aplicação, de tamanho real igual a 19 (coluna Nós). É fácil perceber que, se o valor do dual de n não melhora em relação ao de a_n e se o valor do primal permanece, o valor da estimativa não cresce, indicando que o espaço de busca não diminui após a exploração de n, portanto a subárvore prevista por n provavelmente será a mesma prevista por a_n. Este fato pode ser compravado observando os nós 7, 8, 9, 10 e 11 da tabela. O contrário pode ser percebido nos valores dos nós 16 e 17, onde o nó 17 consegue uma boa melhoria no valor do dual e a estimativa deste nó é bem menor do que a de seu nó ancestral. A estimativa negativa dada pelo nó 19 indica que o nó foi resolvido, ou seja, o espaço de busca foi todo explorado. Observe que o valor do dual é maior que o do primal para este nó.

Vale ressaltar que, os nós de alta profundidade e pertencentes a uma subárvore consideravelmente grande, para instâncias maiores como as utilizadas nos testes, o valor da estimativa tende a ser igual ao número de nós realmente presentes na subárvore. Isto acontece devido ao acúmulo de informação.

A fila de carga pendente dos processos nas estratégias propostas, não possui apenas o

caminho dos nós ainda não explorados, mas possui a estimativa do tamanho da subárvore que os mesmos poderão gerar. Assim, diante do recebimento de requisição de carga, são enviados, de acordo com o desempenho dos processos em questao, aqueles nós que, estimativamente, gerarão maiores subárvores. Este mecanismo está melhor descrito na seção abaixo.

5.1.1.3 Algoritmos adaptativos

Os algoritmos apresentados nas Figuras 5.1 e 5.2 correspondem, repectivamente, às técnicas *GlobalAdapt* e *HibridaAdapt*. Diferente dos algoritmos apresentados na Seção 4.3, estes incluem não só os passos a serem seguidos quando o processo se torna ocioso, mas também os passos a serem executados quando o processo i recebe uma mensagem do tipo $LOADREQUEST$ de um processo k, a fim de mostrar como é calculado o número de nós que devem ser enviados e como eles são retirados da fila de carga acumulada.

Nas duas estratégias, ao verificar que sua fila de subárvores está vazia, o processo i calcula o desempenho, M_i, com que resolveu os nós mais recentemente recebidos, mais precisamente da última subárvore resolvida (linha 4 do *GlobalAdapt* e *HibridaAdapt*). Em seguida, o processo i envia o valor M_i contido em uma mensagem do tipo $LOADREQUEST$, inicialmente, para o processo k_1, depois para k_2 e assim por diante até que consiga carga ou entre em estado de terminação. Note que os valores para os $k's$ são calculados, respectivamente, para as duas estratégias como discutido na Seção 4.3.

Quando o processo i recebe uma mensagem $LOADREQUEST$ de um processo k, verifica o estado de ζ_i (linha 25 do *GlobalAdapt* e 22 do *HibridaAdapt*). Se ζ_i é vazio, i responde a requisição enviando a k uma mensagem do tipo *IDLE*. Caso contrário i calcula seu desempenho, M_i e o número de nós N que devem ser enviados a k, processo de quem recebeu a requisição de carga. Assim, o processo i envia uma mensagem do tipo $BRANCH$ contendo N subárvores para k (linha 29 do *GlobalAdapt* e linhas 26 do *HibridaAdapt*). Vale lembrar que, no cálculo do desempenho apresentado nos algoritmos, r refere-se ao número de nós resolvidos em um período de tempo t.

O critério utilizado para retirar os N nós de ζ_i é o seguinte: i) ordena-se em ordem decrescente, segundo as estimativas, as subárvores presentes em ζ; ii) retira-se de ζ_i as S primeiras subárvores cuja soma das estimativas dê o valor mais próximo de N. Este último passo corresponde a função $RetiraFila(N)$ (linha 28 do *GlobalAdapt* e as linhas 25 e 45 do *HibridaAdapt*).

Posição	Estimativa
1	21,95
2	26,34
3	27,38
4	22,22
5	21,14
6	21,13
7	23,24
8	17,63
9	3,62
Total	184,65

Tabela 5.2: Cálculo da estimativa

Para exemplificar a função $RetiraFila(N)$, considere a Tabela 5.2 obtida na execução da instância *i640-212*, utilizando a estratégia *GlobalAdapt*, em um Grid composto de 2 *clusters* com 8 processadores cada, simulado através do *cluster* da Universidade Federal Fluminense - UFF. Esta Tabela apresenta a estimativa dos nós presentes na fila de carga pendente ζ do processo 0 em um determinado momento da execução. A primeira coluna indica a posição da fila de carga pendente onde o nó está armazenado e a segunda indica a estimativa do nó. Note que, ζ_0 totaliza 9 subárvores, que, estimativamente, equivalem a 184,65 nós. O processo 0 ao receber uma mensagem do tipo $LOADREQUEST$ do processo 15, baseado no desempenho recebido de 15 e no seu próprio desempenho, calculou, segundo a equação 5.1, que devia enviar a 15 uma mensagem $BRANCH$ contendo 171,04 nós. Para tanto, o processo 0 retirou de ζ_0 as subárvores 3, 2, 7, 4, 1, 5, 6 e 9 nesta ordem, que totalizam 167,02 nós. Note que a subárvore 8 não foi enviada, pois a sua estimativa somada às estimativas das subárvores já retiradas de ζ_0 ultrapassaria a quantidade de nós que deveriam ser enviados. Portanto, foram enviadas as 7 primeiras subárvores mais a última subárvore, que aproximadamente totalizam os N nós que deveriam ser enviados.

É importante notar que o envio de um $LOADREQUEST\_CLUSTER$ é suficiente para afirmar que todos os processos daquele *cluster* estão ociosos. Contudo, o desempenho enviado junto com a mensagem não deve ser relativo apenas ao desempenho do processo que está enviando a mensagem, ou seja, do líder do *cluster*, mas deve ser um valor relativo ao desempenho de todos os processos pertencentes aquele *cluster*. Caso contrário, a quantidade de carga pode não ser suficiente, fazendo com que alguns, ou todos os processadores fiquem ociosos mais rapidamente. O desempenho do *cluster*, chamado M_c, enviado pelo líder é baseado nos desempenhos dos processos e no tamanho do *cluster* e é dado por:

$$M_c = \sum_{i=0}^{m-1} M_i \tag{5.4}$$

É importante ressaltar que o líder, l_w, que receber $LOADREQUEST\_CLUSTER$ de l_c compara o desempenho do *cluster* c com o seu próprio desempenho e não com o desempenho do seu *cluster*, já que as subárvores serão retiradas apenas de sua fila.

procedure `GlobalAdapt`

1: $vez = 1$;
2: **se** (Entrada = $NULL$) **então**
3: **se** ($\zeta_i = \emptyset$) **então**
4: $M_i = \frac{r}{t}$;
5: $k = (i + m - vez)\ mod\ m$;
6: Envia LOADREQUEST(M_i) para k;
7: **fim se**
8: **fim se**
9: **se** (Entrada = mensagem $IDLE_k$) **então**
10: $vez = vez + 1$;
11: **se** ($vez < m$) **então**
12: $k = (i + m - vez)\ mod\ m$;
13: Envia LOADREQUEST(M_i) para k;
14: **senão**
15: **para todo** k tal que $0 \leq k \leq m - 1$ **faça**
16: Envia SUSPECTEND para k;
17: **fim para**
18: retorne;
19: **fim se**
20: **fim se**
21: **se** (Entrada = mensagem $BRANCH(S)_k$) **então**
22: Executa *Branch-and-Bound(S)*;
23: **fim se**
24: **se** (Entrada = mensagem $LOADREQUEST(M_k)_k$) **então**
25: **se** ($\zeta_i \neq \emptyset$) **então**
26: $M_i = \frac{r}{t}$;
27: $N = M_k * \frac{\zeta_i}{M_i + M_k}$;
28: $S = RetiraFila(N)$;
29: Envia BRANCH(S) para k;
30: **senão**
31: Envia IDLE para k;
32: **fim se**
33: **fim se**

fim.

Figura 5.1: Procedimento de balanceamento de carga - *GlobalAdapt*

procedure `HibridaAdapt`

1: $vez = 1$;
2: **se** (Entrada = $NULL$) **então**
3: **se** ($\zeta_i = \emptyset$) **então**
4: $M_i = \frac{r}{t}$;
5: $k = |(i - l_c - vez)| \ mod \ tam_c + l_c$;
6: Envia LOADREQUEST(M_i) para k;
7: **fim se**
8: **fim se**
9: **se** (Entrada = mensagem $IDLE_k$) **então**
10: $vez = vez + 1$;
11: **se** ($vez < tam_c$) **então**
12: $k = |(i - l_c - vez)| \ mod \ tam_c + l_c$;
13: Envia LOADREQUEST(M_i) para k;
14: **senão**
15: Envia SUSPECTEND(M_i) para l_c;
16: **fim se**
17: **fim se**
18: **se** (Entrada = mensagem $BRANCH(S)_k$) **então**
19: Executa *Branch-and-Bound(S)*;
20: **fim se**
21: **se** (Entrada = mensagem $LOADREQUEST(M_k)_k$) **então**
22: **se** ($\zeta_i \neq \emptyset$) **então**
23: $M_i = \frac{r}{t}$;
24: $N = M_k * \frac{\zeta_i}{M_i + M_k}$;
25: $S = RetiraFila(N)$;
26: Envia BRANCH(S) para k;
27: **senão**
28: Envia IDLE para k;
29: **fim se**
30: **fim se**
31: **se** (Entrada = mensagem $SUSPECTEND(M_k)_k$) **então**
32: **se** ($i = l_c$ e numero de suspeitas recebidas = tam_c) **então**
33: $M_c = \sum_{i=0}^{m-1} M_i$;
34: $vez = 1$;
35: **se** ($vez < n$) **então**
36: $w = (c + n - vez) \ mod \ n$;
37: Envia $LOADREQUEST\_CLUSTER(M_c)$ para l_w;
38: **fim se**
39: **fim se**
40: **fim se**
41: **se** (Entrada = mensagem $LOADREQUEST\_CLUSTER(M_k)_{l_w}$) **então**
42: **se** ($\zeta_i \neq \emptyset$) **então**
43: $M_i = \frac{r}{t}$;
44: $N = M_k * \frac{\zeta_i}{M_i + M_k}$;
45: $S = RetiraFila(N)$;
46: Envia $BRANCH\_CLUSTER(S)$ para l_w;
47: **senão**
48: Envia $IDLE\_CLUSTER$ para l_w;
49: **fim se**
50: **fim se**
51: **se** (Entrada = mensagem $IDLE\_CLUSTER_{l_w}$) **então**
52: $vez = vez + 1$;
53: **se** ($vez < n$) **então**
54: $w = (c + n - vez) \ mod \ n$;
55: Envia $LOADREQUEST\_CLUSTER(M_c)$ para l_w;
56: **senão**
57: **para todo** l_w tal que $0 \leq w \leq n - 1$ e $w \neq c$ **faça**
58: Envia $SUSPECTEND\_CLUSTER$ para l_w;
59: **fim para**
60: **fim se**
61: **fim se**

fim.

Figura 5.2: Procedimento de balanceamento de carga - *HibridaAdapt*

5.1.2 *MeAdapt*

Pelo paradigma mestre-escravo, um processo denominado mestre é responsável por gerenciar a aplicação paralela. Este paradigma é utilizado por vários autores na literatura [k. Anstreicher et al. 2002] [Shao et al. 2000] [Heymann et al. 2000] [Oliveira et al. 2005]. Algumas vantagens que tornam este paradigma apropriado para ser utilizado em várias aplicações paralelas são:

- **Centralização de informações**. Como o processo mestre controla todas as ações dos processos escravos, informações tais como o desempenho dos processadores e quantidade de carga ainda não processadas são facilmente obtidas. Assim, é alcançado um balanceamento global de carga rapidamente, ou seja, a carga ainda não resolvida é equilibrada proporcionalmente entre todos os processos participantes da aplicação, diminuindo a ociosidade dos processos.
- **Centralização de dados**. Atualização de dados durante a execução da aplicação é realizada apenas em um processo.
- **Controle de execução da aplicação**. O processo mestre tem controle sobre a execução de todos os processos da aplicação, logo, procedimentos como a terminação do algoritmo e a falha de algum processador são rapidamente detectadas pelo mestre. Além disso, identificar processos ociosos na aplicação é uma tarefa trivial.

Diante destes benefícios, uma estratégia de balanceamento de carga baseada neste paradigma foi desenvolvida. Já que foi definido na aplicação um processo mestre, os procedimentos de difusão do primal e detecção de terminação foram modificados com o intuito de tirar proveito das informações contidas no mestre, objetivando a melhoria da eficiência da aplicação. A segui está descrito o novo funcionamento dos procedimentos do algoritmo.

- **Distribuição inicial de carga**: neste procedimento, o processo mestre inicia a execução do algoritmo B&B resolvendo o nó raiz da árvore. A cada ramificação, envia carga ao processo $i = level + 1$. Assumindo que o processo 0 é o mestre, o procedimento acontece enquanto o valor de i for menor que m.
- **Difusão do primal**: quando um processo encontra um novo valor para o limiar primal, este processo envia para o mestre uma mensagem *PRIMAL* contendo este novo valor. O mestre, ao receber a mensagem, compara se o valor recebido é menor

que o valor armazenado. Caso afirmativo, o mestre envia a mensagem recebida a todos os escravos. Caso contrário, a mensagem é descartada.

- **Balanceamento de carga**: o processo mestre, como qualquer outro processo, computa a primeira subárvore que lhe é designada. Ao terminar a computação, o mestre inicia um procedimento chamado *busca por tarefas*. Este procedimento consiste em envios sucessivos de pedido de carga aos processos escravos com o objetivo de acumulá-la. Assim, quando um processo i terminar a execução de sua subárvore, o mestre poderá lhe enviar tarefa sem que i fique ocioso por um longo período de tempo. Este período de tempo seria relativo ao tempo em que o mestre gastaria para conseguir tarefa com os outros escravos e reenviá-las a i. A *busca por tarefas* realizada pelo mestre é restringida por um limite de carga acumulada, denominada lim, baseado no desempenho apresentado pelos processos e na quantidade de processos escravos que participam da aplicação e é dado por:

$$lim = \frac{(\sum_{i=1}^{i} M_i) + m - 1}{\alpha} \tag{5.5}$$

onde α é um parâmetro de redução utilizado para calibrar o tamanho da fila de carga do mestre cujo valor é utilizado neste trabalho é igual a 2, obtido empiricamente. Enquanto aguarda a resposta da requisição de carga, o mestre responde a possíveis requisições dos escravos. Logo a quantidade de processos ociosos pode diminuir entre o pedido e o recebimento de carga. Este controle é essencial, uma vez que, se o limite de carga do mestre for grande, muitas requisições de carga são necessárias para que o mestre alcance seu limite. Contudo, os escravos podem se tornar ociosos depois de múltiplas requisições, aumentando assim, o tempo de ociosidade dos escravos inutilmente.

O algoritmo 5.3 descreve a estratégia de balanceamento de carga deste paradigma. Igualmente como discutido nas seções anteriores, quando um processo escravo i se torna ocioso, envia ao mestre uma mensagem *LOADREQUEST* contendo o seu desempenho. O processo mestre, se possuir carga, envia uma mensagem *BRANCH* contendo a quantidade de nós referente ao desempenho do escravo i em relação aos demais (linha 19 e 37). Note que o desempenho do mestre não é adicionado. O mestre obtém o desempenho dos escravos da seguinte forma: a cada r nós resolvidos, o escravo envia ao mestre seu desempenho M_i, que neste caso, é a razão entre r e o tempo t gasto para resolver os r nós. Assim, o mestre mantém a informação do desempenho relativo a todos os escravos.

procedure `MeAdapt`

1: $k = 1$;
2: **se** (Entrada = mensagem $NULL$) **então**
3: **se** ($i = mestre$ e $total_carga < lim$) **então**
4: $M_{mestre} = (\sum_{j=1}^{m-1} M_j)/m - 1$;
5: Envia LOADREQUEST(M_{mestre}) para k;
6: $k = k + 1$;
7: **fim se**
8: **senão**
9: **se** ($\zeta_i = \emptyset$) **então**
10: $M_i = \frac{r}{t}$;
11: Envia LOADREQUEST(M_i) para $mestre$;
12: **fim se**
13: **fim se**
14: **se** (Entrada = mensagem $LOADREQUEST_k$) **então**
15: **se** ($\zeta_{mestre} \neq \emptyset$) **então**
16: $M_{mestre} = (\sum_{k=1}^{m-1} M_k)/m - 1$;
17: $N = M_k * \frac{\zeta_{mestre}}{M_{mestre}+M_k}$;
18: $S = RetiraFila(N)$;
19: Envia BRANCH(S) para k;
20: **senão**
21: Coloque k no conjunto de processos ociosos;
22: Envia IDLE para k;
23: **fim se**
24: **fim se**
25: **se** (Entrada = mensagem $LOADREQUEST_{mestre}$) **então**
26: **se** ($\zeta_i \neq \emptyset$) **então**
27: $M_i = \frac{r}{t}$;
28: $N = M_{mestre} * \frac{\zeta_i}{M_{mestre}+M_i}$;
29: $S = RetiraFila(N)$;
30: Envia BRANCH(S) para $mestre$;
31: **fim se**
32: **fim se**
33: **se** (Entrada = mensagem $BRANCH_k$) **então**
34: **para todo** k tal que $ocioso = TRUE$ **faça**
35: $N = M_j * \frac{\zeta_{mestre}}{M_{mestre}+M_j}$;
36: $S = RetiraFila(N)$;
37: Envia $BRANCH(S)$ para todos os processos ociosos;
38: **fim para**
39: **fim se**

fim.

Figura 5.3: Procedimento de balanceamento de carga - *MeAdapt*

Na estratégia *MeAdapt*, para evitar possíveis gargalos no processo mestre como discutido nos capítulos anteriores, quando o mesmo termina a execução da primeira subárvore, permanece somente com a função de balancear a carga entre seus escravos. Esta escolha é justificada pelo fato de que as primeiras subárvores recebidas são grandes, mantendo os processos com carga durante este período, ou seja, poucos escravos requisitam carga. Assim, logo após a execução de sua tarefa, o mestre inicia o processo de *busca por tarefas* como descrito anteriormente.

O mestre envia pedido de carga de forma cíclica no sentido horário, ou seja, inicia enviando pedido de carga ao escravo de identificação imediatamente maior que a sua identificação, depois ao segundo maior e assim por diante até envie pedidos a todos os escravos. Se necessário, o ciclo de pedidos é novamente realizado. Neste trabalho foi definido que o processo mestre é o processo 0 do *cluster* 0, por este motivo o valor de k no algoritmo apresentado em 5.3 é 1.

- **Detecção de terminação**: quando o processo mestre não consegue mais obter carga com os seus escravos e recebeu *LOADREQUEST* de todos os mesmos, o mestre envia uma mensagem chamada *TERMINATE* a todos os escravos indicando que não existe mais carga a ser resolvida e que o processo pode terminar a sua execução da aplicação. Após o envio da mensagem, o mestre também termina.

Capítulo 6

Resultados Experimentais

O algoritmo paralelo proposto foi implementado em C++ e MPICH-G2 v1.2.6, uma versão do MPI habilitada para o *Globus* ([Foster e Kesselman 1998] [Foster e Kesselman 1999] [Karonis et al. 2003] [Snir et al. 1996]). Os testes foram executados em uma *Grid* simulada em um *cluster* da Universidade Federal Fluminense - UFF, composta por 16 processadores *Pentiun 4 2.6 GHz com 512 Mb de RAM* divididos logicamente em dois outros *clusters* contendo 8 processadores cada um. Para simular o *overhead* de comunicação entre processos de *clusters* diferentes, a troca de mensagens entre os mesmos teve um atraso no envio de ts milisegundos, onde ts é uma média de tempo gasto para enviar uma mensagem de um processo do *cluster* da UFF para a um processo do *cluster* de uma outra universidade do Estado do Rio de Janeiro, a PUC-Rio.

As instâncias do PSG utilizadas para testes, chamadas instâncias de incidência, foram propostas por [Duin 1993] e estão disponíveis em [Koch et al. 2003]. Instâncias de incidência são construídas a partir de um grafo aleatório, sendo o custo das arestas um valor inteiro escolhido em uma distribuição normal cuja média depende do número de nós terminais em que a aresta é incidente. A média é 300 quando a aresta é incidente em dois vértices terminais, 200 quando é incidente em apenas um e 100 se não é incidente a nenhum vértice terminal.

As instâncias incidentes presentes em [Koch et al. 2003] estão divididas em quatro séries e se diferenciam no número de arestas, vértices e terminais. Estas são nomeadas pelo perfixo i seguido pelo número de vértices que possuem: i080, i160, i320 e i640. Em cada série, há 20 diferentes combinações de números de nós terminais e números de arestas. Em cada combinação existem 5 instâncias [Drummond et al. 2005] [Uchoa 2001]. O algoritmo seqüencial no qual é baseado o distribuído utilizado neste trabalho, resolveu 24 instâncias

Instâncias	Vértices	Arestas	Terminais	Solução Ótima
i320-311	320	1845	80	17945
i320-313	320	1845	80	17991
i320-314	320	1845	80	18088
i320-341	320	10208	80	16296
i320-343	320	10208	80	16281
i320-344	320	10208	80	16295
i320-345	320	10208	80	16289
i640-212	640	4135	50	11795
i640-213	640	4135	50	11879
i640-214	640	4135	50	11898
i640-215	640	4135	50	12081

Tabela 6.1: Instâncias incidentes.

da classe i320 que estavam em aberto e 12 das 42 da classe i640. As instâncias utilizadas nos testes foram escolhidas dentre estas duas classes e são apresentadas na Tabela 6.1. Vale lembrar que o nível de dificuldade para solucionar o problema não é proporcional a quantidade de arestas, vértices ou terminais, mas é dependente da forma com que as arestas se conectam no grafo.

6.1 Métricas

O objetivo deste trabalho, como já discutido anteriormente, não consiste apenas em avaliar as estratégias desenvolvidas, mas consiste também em compará-las com as técnicas propostas por [Drummond et al. 2005]. Com o intuito de facilitar a descrição dos testes, o algoritmo B&B será denominado pela estratégia de balanceamento de carga que utiliza. Por exemplo, se o B&B utiliza a estratégia de balanceamento *Global* será chamado *Global*.

A Grid (simulada) utilizada nos testes não é dedicada. A presença de carga externa proporciona um ambiente paralelo diferente a cada execução da aplicação. Para que uma comparação justa das estratégias pudesse ser realizada, os algoritmos foram executados simultaneamente, ou seja, foram executados sujeitos ao mesmo ambiente, como proposto por [Oliveira et al. 2005]. Assim, um algoritmo é carga externa para o outro e os demais processos são carga externa para os dois, desta forma, o algoritmo que apresenta maior tempo de execução é beneficiado pelo término do outro algoritmo. Por exemplo, considere que estão sendo executados os algoritmos *Global* e *GlobalAdapt*. Se o *GlobalAdapt* termina antes que o *Global*, este não terá mais que dividir o processador com o *GlobalAdapt* e, portanto, diminuirá seu tempo de relógio de execução.

Outro fator que pode tornar a execução dos algoritmos imprevisíveis é a aleatoriedade apresentada pelo B&B na busca de uma solução (veja [Lai e Sahni 1984]). Por exemplo, se o algoritmo *Global* encontrar antes do *GlobalAdapt* bons valores para os limites primal e dual, suas subárvores serão podadas nos níveis mais baixos e conseqüentemente apresentará um menor tempo de execução, uma vez que executará menos nós. Para evitar a aleatoriedade do algoritmo, a solução ótima (o limite primal) da instância em questão foi passada como parâmetro de entrada. Entretanto, este fato diminui mas não elimina o fator de aleatoriedade devido aos critérios de decisão utilizados pela heurística que define os valores dos limites. Assim, os algoritmos foram executados 5 vezes para cada instância e os resultados apresentados são os valores médios encontrados nas execuções.

As estratégias foram avaliadas segundo os critérios descritos abaixo:

- **Tempo da aplicação**: O tempo, tanto de CPU quanto de relógio, gasto para resolver uma aplicação é a maneira mais usual de avaliar o desempenho de um algoritmo. Entretanto, em um ambiente paralelo esta medida deve considerar a heterogeneidade dos processadores e, por isso, o tempo de CPU na execução da aplicação foi medido e normalizado de acordo com a equação proposta por [Drummond et al. 2005]:

$$TCPU = \sum_{i=0}^{m-1} \alpha_i * Tcpu_i \tag{6.1}$$

  onde $Tcpu_i$ é o tempo de CPU gasto pelo processo i e α_i é o fator de normalização associado a máquina em que i foi executado. Este fator é obtido executando um *benchmark* nas máquinas que participam do ambiente e comparando com o tempo de execução do mesmo *benchmark* quando executado em uma máquina padrão. A máquina padrão utilizada neste trabalho foi um *Pentium 4 2.6 Ghz com 512 Mb RAM*.

- **Ociosidade dos processos**: a eficiência de uma estratégia de balanceamento de carga é inversamente proporcional ao tempo de ociosidade dos processos. Esta métrica consiste em calcular o tempo de CPU em que os processos ficaram ociosos durante a executação do algoritmo. Esta medida foi possível porque o MPICH-G2 implementa o recebimento de mensagens via *poll*, ou seja, espera ocupada. Desta forma, o tempo de CPU gasto em uma aplicação também inclui o tempo de CPU utilizado enquanto os processos esperam por uma mensagem. O tempo de CPU em que os processos ficaram ociosos foi totalizado e chamado neste trabalho de Toc.

Para facilitar a análise, o tempo de ociosidade dos processos durante a aplicação é dado por uma taxa de ociosidade denominada Tax, dada por

$$Tax = \frac{Toc}{TCPU} \tag{6.2}$$

Vale ressaltar que o valor de Toc é normalizado, assim como $TCPU$.

- **Ambientes de execução**: a justificativa de utilizar estratégias de balanceamento de carga mais robustas é tratar a influência de possível carga externa e a heterogeneidade dos processadores durante a execução da aplicação. Para avaliar a robustez das estratégias propostas, foram propostos três ambientes de execução de modo que é possível avaliar a capacidade das estratégias de balanceamento de carga se adequarem a possíveis diversificações da plataforma paralela. O primeiro ambiente, denominado A_1, é composto por 16 máquinas *Pentium 4 2.6 Ghz com 512 Mb RAM*. Este ambiente é homogêneo e a presença de carga externa é quase inexistente (lembrando que as duas estratégias são executadas simultaneamente, logo uma é carga externa para a outra). O segundo ambiente, A_2, tem a mesma formação de A_1, entretanto, foi incrementado com carga externa. Esta carga foi gerada executando, em alguns processadores, a versão seqüencial do algoritmo B&B para diferentes instâncias do problema e um algoritmo paralelo capaz de simular carga através do uso de *threads* proposto por [Thomé et al. 2006]. Note que, para estes dois ambientes, não é necessário normalizar o tempo de CPU, pois as máquinas são idênticas. Finalmente, o terceiro ambiente A_3 também foi contituído por 16 máquinas, sendo 6 máquinas de diferentes configurações, que variam entre processadores *Pentium II 233 Mhz com 128 Mb de RAM* e *Pentium 4 2.8 Ghz com 512 Mb RAM*, e 10 máquinas *Pentium 4 2.6 Ghz com 512 Mb RAM*, proporcionando, um ambiente heterogêneo. Além disso, o ambiente A_3 também foi incrementado com carga externa.

- **Mensagens trocadas entre os processos**: como já discutido, a execução do algoritmo B&B paralelo exige poucas trocas de mensagens, somente aquelas relativas a distribuição inicial, a difusão do primal e detecção de terminação. Portanto, todas as outras mensagens trocadas entre os processos são relativas ao balanceamento de carga, logo, quanto menor o *overhead* de comunicação entre os processos mais robusta é a técnica de balanceamento de carga.

- **Eficiência paralela da aplicação**: duas medidas foram utilizadas para avaliar a eficiência da aplicação paralela. A primeira delas, chamada $Effo$, investiga o

desempenho da aplicação em relação ao tempo de ociosidade dos processos (Toc). Esta foi proposta por [k. Anstreicher et al. 2002] e é dada por:

$$Effo = \frac{TCPU}{Toc} \tag{6.3}$$

A outra medida, Eff_s, apresenta a relação entre o tempo de CPU do algoritmo seqüencial e o tempo total de CPU gasto pelo algoritmo paralelo e é dada por:

$$Effs = \frac{Tseq}{TCPU} \tag{6.4}$$

Para melhor avaliar as estratégias os testes foram divididos em cinco seções:

- *Global* x *GlobalAdapt*;
- *Híbrida* x *HibridaAdapt*;
- *GlobalAdapt* x *HibridaAdapt* x *MeAdapt*;
- *GlobalAdapt* x *HibridaAdapt* x *MeAdapt* - 4 *clusters*;

6.2 *Global* x *GlobalAdapt*

O primeiro teste consistiu na execução simultânea dos algoritmos *Global* e *GlobalAdapt*, para os ambientes A_1, A_2 e A_3, cada um formado por 2 *clusters*, contendo 8 processos cada.

Como já discutido, estratégias de balanceamento robustas evitam a ociosidade dos processos. Nos algoritmos em questão, um processo, quando se torna ocioso, envia sucessivas requisições de carga, tendo como conseqüência o aumento do *overhead* de comunicação. A Tabela 6.2 apresenta a média do número de mensagens enviadas pelos processos para todos os ambientes testados. As colunas *Int* e *Ext* apresentam, respectivamente, o número médio de mensagens trocadas entre processos do mesmo *cluster* e entre processos de *clusters* diferentes. Estas mensagens incluem não só as mensagens referentes ao balanceamento de carga, mas também as mensagens trocadas pelos processos nos procedimentos *Distribuição Inicial*, *Detecção de Terminação* e *Difusão do Primal*. É fácil perceber que, para todos os ambientes, a estratégia *GlobalAdapt*, reduziu quantitativamente o número médio de mensagens enviadas, o que indica que esta estratégia foi capaz de manter os

processos balanceados por um período de tempo maior que a estratégia *Global*. É interessante notar que, para a menor instância testada (*i320-343*), a estratégia *GlobalAdapt*, não obteve um resultado melhor que a *Global* no ambiente A_1 em termos de número de mensagens, porque em ambientes homogêneos e dependendo do tamanho da instância, técnicas robustas para manter os processos balanceados não são geralmente adequadas. Uma única exeção, para os outros dois ambientes, ocorre para a mesma instância (*i320-343*) no ambiente A_2 no número de mensagens externas com uma diferença menor que 10 mensagens.

Instâncias	***Global***						***GlobalAdapt***					
	A_1		A_2		A_3		A_1		A_2		A_3	
	Int	***Ext***	***Int***	***Ext***	***Int***	***Ext***	***Int***	***Ext***	***Int***	***Ext***	***Int***	***Ext***
320-311	6051,2	2417,6	5612,2	2262,0	7031,8	3007,6	**4535,8**	**1836,6**	**4112,8**	**1759,4**	**5126,8**	**2176,2**
320-313	11065,8	4234,8	12137,8	4247,6	7409,0	3005,6	**6476,2**	**2526,0**	**7719,4**	**2924,8**	**5230,8**	**2182,6**
320-314	19532,8	8085,2	16171,4	6527,8	15443,0	6811,0	**9193,8**	**3712,8**	**8330,2**	**3303,8**	**7036,0**	3073,6
320-341	10603,0	4042,8	11648,4	4263,6	10420,2	4011,6	**8543,0**	**3466,4**	**8769,0**	**3460,6**	**7923,0**	**3244,6**
320-343	**4423,2**	**1937,6**	5063,4	**2201,8**	5160,8	2317,0	4677,8	2073,4	**5025,4**	2211,6	**4687,2**	**2164,4**
320-344	6145,2	2601,2	6707,4	2761,4	6007,2	2485,4	**4663,0**	**1971,0**	**5447,2**	**2307,2**	**5145,2**	**2187,6**
320-345	9471,4	3735,6	8063,6	3181,0	8645,6	3505,6	**6721,6**	**2862,2**	**6868,0**	**2677,6**	**6223,8**	**2715,8**
640-212	8430,4	3456,2	8025,2	3176,0	7078,8	2806,0	**5731,6**	**2359,0**	**6498,4**	**2720,6**	**5132,6**	**2255,6**
640-213	6710,6	2928,0	8471,6	3644,2	6604,2	2991,4	**6122,2**	**2526,2**	**5701,4**	**2478,2**	**6203,6**	**2635,8**
640-215	17910,8	6215,2	19568,8	7301,0	15721,8	5893,4	**6806,6**	**2802,0**	**8270,4**	**3267,2**	**7186,2**	**2930,4**
640-214	34498,2	11816,4	31081,6	11038,2	35324,6	12742,8	**11147,4**	**4205,0**	**11977,6**	**4329,8**	**11699,4**	**4192,4**

Tabela 6.2: Quantidade de mensagens enviadas trocadas pelos processos quando utilizadas as estratégias *Global* e *GlobalAdapt*.

As estratégias desenvolvidas provocam um acréscimo no tempo de CPU, particularmente, na seleção das subárvores que serão enviadas durante a transferência. Estas decisões envolvem a ordenação da fila de carga pendente e a escolha de quais nós serão enviados, sendo que este último exige comparações entre os nós escolhidos e o número de nós a serem enviados. As Tabelas 6.3, 6.4 e 6.5 apresentam, para os ambientes A_1, A_2 e A_3, respectivamente, o tempo de CPU médio das cinco execuções obtido para cada instância (coluna $TCPU$) quando utilizados os algoritmos *Global* e *GlobalAdapt*, a taxa média Tax de ociosidade dos processos, a média da eficiência paralela do algoritmo, tanto para $Effo$ quanto para $Effs$. Para as instâncias testadas estas decisões foram importantes, pois, em relação a estratégia *Global*, diminuiram a ociosidade dos processos, aumentando a eficiência paralela, $Effo$, em um comportamento médio de 80,15% para o ambiente A_1, 74,77% para o ambiente A_2 e 67,84% para o ambiente A_3. Este comportamento também pode ser verificado pela taxa Tax de ociosidade na Tabela 6.6 que apresenta o ganho percentual médio da estratégia *GlobalAdapt* em relação a *Global* em relação às diversas métricas previamente discutidas. Entretanto, pode-se perceber que o tempo de execução da aplicação não reduziu como o tempo de ociosidade, devido ao acrécimo do tempo de CPU discutido anteriormente e por isto, a $Effs$ do algoritmo *GlobalAdapt* apresentou quase os mesmos resultados que a *Global*. Vale lembrar que os valores apresentados na Tabela 6.5 foram normalizados de acordo com a expressão sugerida por [Drummond et al. 2005] e descrita no início deste capítulo.

Em relação a eficiência, $Effs$, no primeiro e segundo ambiente, a estratégia *Global* apresentou-se em média melhor que a *GlobalAdapt*. Este ganho é devido, principalmente ao melhor desempenho obtido pela instância *i320-343*. Já para o ambiente A_3 a estratégia *GlobalAdapt* obteve, em média, uma melhora de 28,31% em relação a *Global*. Diante deste fato pode-se comprovar que a estratégia *GlobalAdapt*, apesar do acréscimo de cálculos, se mostrou eficiente, principalmente diante de um ambiente heterogêneo e compartilhado. Os valores discutidos acima estão descritos na Tabela 6.6.

A Tabela 6.7 apresenta os tempos médios de relógio de parede. A linha "*Melhora*" indica, para cada ambiente, o valor referente à porcentagem média de melhoria obtida pela estratégia *GlobalAdapt* sobre a *Global*. Pode-se perceber que quando o ambiente é homogêneo e sem carga externa (coluna A_1), a utilização da estratégia adaptativa, apesar de diminuir a ociosidade dos processos, contribui com uma melhora de menos de 1% no tempo de execução da aplicação. Este fato ocorre devido aos cálculos realizados para se adequar a carga pendente aos desempenhos dos recursos. Contudo, uma vez que os processadores são homogêneos e quase dedicados (o algoritmo *Global* é carga externa para

Instâncias	A_1							
	Global				*GlobalAdapt*			
	TCPU	*Tax*	*Effo*	*Effs*	*TCPU*	*Tax*	*Effo*	*Effs*
320-311	8765,96	0,014	71,40	**1,34**	**8737,65**	**0,009**	**113,20**	**1,34**
320-313	**15840,88**	0,015	67,80	**1,47**	16027,79	**0,008**	**130,80**	1,46
320-314	**34923,72**	0,014	74,00	**0,98**	35064,50	**0,006**	**172,80**	**0,98**
320-341	**8012,21**	0,092	10,60	**0,65**	8108,20	**0,081**	**12,40**	**0,65**
320-343	**1840,02**	0,183	5,00	**0,71**	2031,45	**0,177**	**5,40**	0,65
320-344	3273,24	0,127	7,60	**0,87**	**3272,59**	**0,099**	**10,00**	**0,87**
320-345	5195,56	0,130	7,80	0,76	**5151,78**	**0,097**	**10,40**	**0,77**
640-212	6083,33	0,068	14,80	0,98	**5964,71**	**0,044**	**22,80**	**1,00**
640-213	**8117,87**	0,030	33,60	**1,13**	8157,75	**0,027**	**39,60**	1,12
640-214	72308,12	0,024	41,40	**1,34**	**71473,72**	**0,007**	**146,60**	**1,34**
640-215	52556,07	0,014	72,60	**1,01**	**52538,10**	**0,005**	**203,80**	**1,01**

Tabela 6.3: Comparação entre as estratégias *Global* e *GlobalAdapt* quanto aos parâmetros de eficiência e tempo - ambiente A_1.

Instâncias	A_2							
	Global				*GlobalAdapt*			
	TCPU	*Tax*	*Effo*	*Effs*	*TCPU*	*Tax*	*Effo*	*Effs*
320-311	**8673,44**	0,016	63,60	**1,35**	8758,24	**0,009**	**115,60**	1,34
320-313	15947,52	0,020	49,80	1,46	**15921,48**	**0,012**	**82,00**	**1,47**
320-314	**34197,67**	0,013	79,80	**1,00**	34374,61	**0,006**	**179,40**	**1,00**
320-341	8350,67	0,123	7,80	**0,63**	**8308,96**	**0,098**	**9,80**	**0,63**
320-343	**1975,94**	**0,221**	**4,00**	**0,67**	2120,76	0,225	**4,00**	0,62
320-344	3506,01	0,166	5,60	0,81	**3471,74**	**0,132**	**7,20**	**0,82**
320-345	**5220,96**	0,145	6,60	**0,76**	5275,48	**0,116**	**8,20**	0,75
640-212	6076,72	0,071	13,60	**0,98**	**6040,53**	**0,056**	**17,80**	**0,98**
640-213	8333,71	0,049	20,00	1,10	**8106,06**	**0,027**	**37,80**	**1,13**
640-214	72743,69	0,025	44,40	0,97	**70974,62**	**0,009**	**123,20**	**1,00**
640-215	52687,29	0,020	49,60	1,01	**51948,17**	**0,007**	**136,60**	**1,03**

Tabela 6.4: Comparação entre as estratégias *Global* e *GlobalAdapt* quanto aos parâmetros de eficiência e tempo - ambiente A_2.

Instâncias	A_3							
	Global				*GlobalAdapt*			
	TCPU	*Tax*	*Effo*	*Effs*	*TCPU*	*Tax*	*Effo*	*Effs*
320-311	8750,11	0,044	22,69	1,34	**8716,00**	**0,030**	**33,71**	**1,35**
320-313	15843,86	0,016	61,50	1,47	**15706,65**	**0,013**	**76,09**	**1,49**
320-314	34582,53	0,030	32,81	0,99	**34195,08**	**0,011**	**93,34**	**1,00**
320-341	8605,44	0,132	7,58	**0,61**	**8626,50**	**0,123**	**8,12**	**0,61**
320-343	2151,64	0,266	3,77	0,61	**2101,47**	**0,219**	**4,56**	**0,63**
320-344	**3424,83**	0,130	7,71	**0,83**	3475,28	**0,103**	**9,71**	0,82
320-345	5646,15	0,179	5,58	0,70	**5478,64**	**0,156**	**6,42**	**0,72**
640-212	6204,31	0,081	12,30	0,96	**6069,17**	**0,062**	**16,08**	**0,98**
640-213	8151,94	0,050	20,14	1,12	**6881,17**	**0,031**	**32,48**	**1,33**
640-214	71382,82	0,029	34,85	0,99	**70556,41**	**0,010**	**104,13**	**1,00**
640-215	51256,88	0,020	50,19	**1,04**	**51102,71**	**0,009**	**115,21**	**1,04**

Tabela 6.5: Comparação entre as estratégias *Global* e *GlobalAdapt* quanto aos parâmetros de eficiência e tempo - ambiente A_3.

Instâncias	***INT***	***EXT***	***TCPU***	***Tax***	***Effo***	***Effs***
A_1	32,43	30,74	-0,85	70,38	80,15	-9,41
A_2	31,06	28,71	-0,10	35,12	74,77	-0,37
A_3	31,01	29,00	2,29	32,63	67,84	28,31

Tabela 6.6: Ganho médio percentual.

Instâncias	A_1		A_2		A_3	
	Global	***Global Adapt***	***Global***	***Global Adapt***	***Global***	***Global Adapt***
320-311	**1141,09**	1155,59	**1646,46**	1678,36	2016,79	**2005,27**
320-313	**2079,62**	2104,43	3016,84	**3007,66**	9848,84	**8249,33**
320-314	**4556,44**	4606,35	**5078,56**	5139,62	7898,91	**7635,87**
320-341	1273,44	**1264,86**	1992,55	**1926,42**	2461,65	**2441,10**
320-343	**363,06**	380,05	**599,30**	612,47	779,03	**723,82**
320-344	556,94	**548,04**	898,55	**875,61**	1137,62	**1099,95**
320-345	879,81	**840,16**	1279,12	**1261,04**	1757,67	**1698,82**
640-212	893,89	**874,38**	1007,28	**992,13**	1578,84	**1521,88**
640-213	1130,20	**1076,40**	1633,81	**1596,13**	1761,71	**1248,64**
640-214	10215,51	**9904,32**	13312,73	**12999,83**	17360,26	**16185,17**
640-215	6874,25	**6842,94**	9619,9	**9479,08**	11477,67	**11295,20**
Melhora	**0,82%**		**0,90%**		**6,89%**	

Tabela 6.7: Comparação entre as estratégias *Global* e *GlobalAdapt* - tempo relógio (em segundos).

GlobalAdapt e vive-versa), identificar qual processador possui melhor desempenho é desnecessário durante o balanceamento. Além disso, dividir proporcionalmente a carga no final da aplicação causa um *overhead* no tempo de execução da mesma. A redução do tempo, que acontece para quase todas as instâncias, é obtida pelo envio de uma maior quantidade de carga, impedindo assim, a ociosidade dos processos. Uma sugestão para evitar o cálculo do desempenho e do tamanho da carga seria o envio da metade da fila de carga pendente durante qualquer transferência de carga entre os processos.

Utilizando um ambiente sujeito à carga externa, a estratégia *GlobalAdapt* diminui a diferença para as instância que no ambiente anterior obtiveram tempo de execução maior que a estratégia *Global*. No ambiente A_3, onde há uma grande diferença entre o desempenho dos processadores, a estratégia *GlobalAdapt* reduziu o tempo de relógio em todas as instâncias. Vale lembrar que a melhoria é obtida somente pela redução dos processos ociosos, nenhuma alteração no algoritmo *Branch-and-Bound* foi realizada.

Para melhor analisar a influência da heterogeneidade e o compartilhamento dos processadores, as Tabelas 6.8, 6.10 e 6.11 apresentam detalhes de uma execução da estratégia *GlobalAdapt* para a instância *i640-212*, nos ambientes A_1, A_2 e A_3, repectivamente. Nestas tabelas, a linha $TCPU$ e $TCPUN$ compreendem, repectivamente, o tempo de CPU

gasto e tempo de CPU normalizado utilizados pelos processos para resolver as subárvores que lhes foram designadas. A linha $QNOS$ indica a quantidade de nós resolvidos pelo processo; $QNOSP$ apresenta a quantidade de nós da primeira subárvore recebida pelo nó; e, finalmente a linha, TOC e TOC indicam o tempo de ociosidade e o tempo de ociosidade normalizado dos processos.

Pode-se observar que nas primeiras Tabelas,6.8 e 6.9, todos os processos resolveram quantidades aproximadas de nós, indicando um ambiente quase dedicado. Nas 6.10 e 6.11 pode-se perceber que o ambiente, influenciado por carga externa, faz com que os processos apresentem desempenhos diferentes, diversificando a quantidade de nós resolvidos e o tempo de execução entre os processadores. Na tabela referente ao ambiente A_3, esta diversificação é ainda maior, já que além da carga externa, existem processadores com capacidades diferentes.

Poderia-se imaginar que a quantidade de nós resolvida por cada nó em um ambiente homogêneo seria igual. Entretanto, variações são ocasionadas pelas podas das subárvores devido a aleatoriedade do algoritmo. Os valores apresentados em 6.8 podem ser comparados com os valores apresentados na Tabela 6.9, que apresenta a quantidade de nós resolvidos por cada processo no mesmo ambiente quando utilizada a estratégia *Global*. Diante da quantidade de nós total resolvidos, a média de nós que cada processo deveria resolver é 839,19 para a estratégia *GlobalAdapt* e 827,56 para *Global*.

Outro ponto importante que pode ser observado através da Tabelas 6.8, 6.10, 6.11 e princialmente na 6.9 é a quantidade de nós presentes na primeira subárvore recebida pelos processos (subárvore recebida durante o procedimento de distribuição inicial). Processos de menores identificações, normalmente recebem subárvores maiores que aqueles de maior identificação, pois as subárvores pertencentes aos níveis mais baixo da árvore possuem maior tamanho. Este fato comprova que um processo ociososo tem mais possibilidade de obter carga ao enviar pedidos de carga aos processos de identificação imediatamente menor.

GlobalAdapt - ambiente A_1																
	0	***1***	***2***	***3***	***4***	***5***	***6***	***7***	***8***	***9***	***10***	***11***	***12***	***13***	***14***	***15***
TCPU	394,41	403,61	423,67	415,16	411,93	403,83	390,14	386,15	370,94	354,79	369,69	357,47	349,28	333,03	327,29	344,21
QNOS	762	1028	959	936	926	829	853	881	775	861	842	858	775	706	698	738
QNOSP	396	495	348	389	287	216	142	150	271	396	63	244	184	65	57	17
TOC	22,83	13,25	3,37	2,64	4,38	6,32	8,87	11,36	16,35	19,59	19,26	24,00	31,79	38,02	37,32	40,20
MSGIN	301	173	229	275	304	220	279	285	270	417	368	488	511	607	597	572
MSGEXT	232	84	103	123	123	77	110	149	136	180	132	182	185	234	240	278

Tabela 6.8: Dados por processo quando utilizada a estratégia *GlobalAdapt* - ambiente A_1.

Global - ambiente A_1																
	0	***1***	***2***	***3***	***4***	***5***	***6***	***7***	***8***	***9***	***10***	***11***	***12***	***13***	***14***	***15***
TCPU	415,81	446,51	442,06	436,2	404,58	388,67	395,46	386,9	368,61	359,07	348,42	349,74	345,31	358,93	333,27	369,91
QNOS	812	1048	1075	990	945	804	829	810	669	778	714	830	750	774	710	100
QNOSP	810	620	408	396	356	274	142	150	284	430	63	263	222	95	71	17
TOC	0,88	3,01	7,02	12,14	25,21	23,01	30,42	31,48	37,42	42,55	43,8	42,49	41,17	37	35,92	4,92
MSGIN	95	208	333	483	507	640	697	796	591	607	657	665	649	582	563	96
MSGEXT	52	87	132	182	168	239	271	368	291	247	254	249	239	186	234	62

Tabela 6.9: Dados por processo quando utilizada a estratégia *Global* - ambiente A_1.

GlobalAdapt - ambiente A_2																
	0	***1***	***2***	***3***	***4***	***5***	***6***	***7***	***8***	***9***	***10***	***11***	***12***	***13***	***14***	***15***
TCPU	447,21	306,48	440,01	435,40	425,70	284,64	417,55	292,26	390,64	267,27	392,56	410,88	284,23	401,57	402,07	417,26
QNOS	855	732	929	1018	935	603	882	631	727	575	815	967	681	985	1023	1021
QNOSP	615	378	425	378	360	231	142	150	315	288	63	292	179	83	55	17
TOC	10,13	3,77	9,83	12,43	11,32	13,61	17,11	19,61	30,44	20,72	25,68	19,93	11,52	18,47	20,84	22,42
MSGIN	175	231	274	240	319	395	464	353	514	474	487	326	344	339	412	359
MSGEXT	106	105	121	88	118	142	180	151	296	217	228	146	145	136	181	185

Tabela 6.10: Dados por processo quando utilizada a estratégia *GlobalAdapt* - ambiente A_2.

GlobalAdapt - ambiente A_3																
	0	***1***	***2***	***3***	***4***	***5***	***6***	***7***	***8***	***9***	***10***	***11***	***12***	***13***	***14***	***15***
TCPU	585,76	396,24	298,24	326,06	437,09	445,25	447,92	455,73	446,99	432,16	620,37	430,53	253,27	364,42	401,25	369,91
TCPUN	585,76	396,24	298,24	228,24	463,32	445,25	479,27	455,73	446,99	216,08	620,37	430,53	124,10	364,42	401,25	48,09
TREAL	1474,35	1474,47	1473,72	1473,86	1474,10	1474,69	1474,39	1474,66	1474,54	1474,28	1474,45	1474,64	1474,37	1474,37	1474,53	1475,08
QNOS	1088	814	600	470	1033	1019	1109	1040	884	530	1427	1012	325	779	1017	100
QNOSP	901	584	245	197	277	232	142	150	257	187	63	372	85	138	99	17
TOC	47,53	32,44	24,25	8,81	10,85	9,82	11,51	18,75	21,56	1,14	35,93	20,00	1,80	45,15	33,42	4,92
TOCN	47,53	32,44	24,25	6,17	11,50	9,82	12,32	18,75	21,56	0,57	35,93	20,00	0,88	45,15	33,42	0,64
MSGIN	157	132	155	300	243	353	409	464	358	341	315	413	350	475	442	96
MSGEXT	82	55	60	136	75	125	157	224	207	138	118	158	123	203	208	62

Tabela 6.11: Dados por processo quando utilizada a estratégia *GlobalAdapt* - ambiente A_3.

As Tabelas 6.12, 6.14 e 6.16 apresentam os tipos e as quantidades de mensagens enviadas pelos processos quando utilizada a estratégia *GlobalAdapt* e as Tabelas 6.13, 6.15 e 6.17 mostram estes resultados quando utilizada a estratégia *Global* na solução da instância *i640-212* para os ambientes A_1, A_2 e A_3, respectivamente. A primeira coluna apresenta os tipos de mensagens enviadas pelos processos, que são: *SUSPECTEND*, *PRIMAL*, *BRANCH*, *NOTNOW*, *IDLE*, *LOADREQUEST*, *SUSPECTEND_ CLUSTER*, *PRIMAL_ CLUSTER*, *BRANCH_ CLUSTER*, *NOTNOW_ CLUSTER*, *IDLE_ CLUSTER*, *LOADREQUEST_ CLUSTER*.

Comparando as tabelas que possuem valores da estratégia *GlobalAdapt* e *Global* para os respectivos ambientes, é fácil perceber que os processos na *GlobalAdapt* enviaram menos mensagens do tipo *LOADREQUEST*, e conseqüentemente, evitaram mensagens do tipo *IDLE* tanto para comunicação entre *clusters* quanto dentro do *cluster*. Nenhum processo, nas duas estratégias, enviaram a mensagem do tipo *PRIMAL* ou *PRIMAL_ CLUSTER*, uma vez que a solução ótima foi passada como parâmetro.

Outro fator interessante mais facilmente percebido nas Tabelas 6.13, 6.15 e 6.17, é a respeito do número de mensagens do tipo *LOADREQUEST* enviadas. Os processos de maiores identificações enviam mais pedidos de carga para os três ambientes. Isto significa que as maiores subárvores estão concentradas nos processos de menores identificações, ou seja, apesar dos processos de 0 a 7 e 8 a 15 receberem, respectivamente, subárvores do mesmo nível da árvore B&B, as subárvores tratadas pelos processos do *cluster 0* são maiores que as tratadas pelo *cluster 1*. Este fato também pode ser notado nas Tabelas 6.12, 6.14 e 6.16, entretanto, vale lembrar que estes valores são influenciados pela quantidade de carga transferida mediante a requisição de carga, ou seja, é dependente do desempenho do processador no qual o processo está alocado.

Quanto às mensagens *LOADREQUEST_ CLUSTER*, é interessante notar que, para as duas estratégias, os processos de menores identificações dentro dos *clusters* são os que mais as enviam. Isto ocorre devido ao ciclo de envio de pedidos de carga. Os processos 0 e 8, por exemplo, sempre que enviam pedidos de carga, enviam o pedido para os processos 15 e 7, respectivamente, que não pertencem a seus *clusters*, o que contribui, também, para o aumento do tempo de ociosidade destes processos, já que envio de uma mensagem para um processo de *cluster* diferente há um atraso de ts milisegundos.

GlobalAdapt - ambiente A_1																	
Tipo da Mensagem	***0***	***1***	***2***	***3***	***4***	***5***	***6***	***7***	***8***	***9***	***10***	***11***	***12***	***13***	***14***	***15***	***TOTAL***
Suspectend	30	15	150	180	165	45	75	90	60	180	60	105	105	135	120	150	1665
Primal	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Branch	14	16	16	34	39	52	39	33	39	48	77	78	72	71	49	37	714
Notnow	6	1	0	1	3	6	4	4	6	6	14	15	15	25	18	10	134
Idle	23	18	21	24	32	43	62	71	73	81	85	95	108	90	75	52	953
Loadrequest	228	102	31	25	42	48	63	57	63	70	95	129	145	204	241	243	1786
Suspeita_Cluster	16	8	80	96	88	24	40	48	32	96	32	56	56	72	64	80	888
Primal_Cluster	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Branch_Cluster	1	4	2	7	9	19	17	32	3	1	6	4	7	5	13	37	167
Notnow_Cluster	0	0	0	0	1	2	4	4	0	1	1	0	1	4	4	10	32
Idle_Cluster	8	8	11	12	17	24	41	57	16	18	19	26	29	33	38	45	402
Lodrequest_Cluster	207	64	10	8	8	8	8	8	56	32	37	30	26	38	27	26	593
TOTAL	533	236	321	387	404	271	353	404	348	533	426	538	564	677	649	690	7334

Tabela 6.12: Tipo e quantidade de mensagens enviadas por processo - *GlobalAdapt* - ambiente A_1.

Global - ambiente A_1																	
Tipo da Mensagem	***0***	***1***	***2***	***3***	***4***	***5***	***6***	***7***	***8***	***9***	***10***	***11***	***12***	***13***	***14***	***15***	***TOTAL***
Suspectend	30	105	120	135	60	165	150	180	45	45	45	60	75	15	90	128	1448
Primal	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Branch	15	52	106	127	86	69	97	82	86	60	66	79	45	68	48	20	1106
Notnow	0	2	4	9	19	20	26	31	17	30	25	20	25	23	16	1	268
Idle	19	23	54	115	165	183	171	157	178	171	143	128	119	78	50	48	1802
Loadrequest	31	20	32	58	112	136	179	240	190	219	284	291	300	308	374	391	3165
Suspeita_Cluster	16	56	64	72	32	88	80	96	24	24	24	32	40	8	48	56	760
Primal_Cluster	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Branch_Cluster	4	12	38	43	23	17	43	82	1	0	0	0	0	1	0	2	266
Notnow_Cluster	0	1	2	2	7	10	12	31	0	1	0	0	1	0	0	1	68
Idle_Cluster	8	10	20	57	94	113	127	150	8	9	10	10	10	11	12	12	661
Lodrequest_Cluster	24	8	8	8	12	11	9	9	183	131	126	120	103	76	92	67	987
TOTAL	147	289	448	626	610	812	894	1058	732	690	723	740	718	588	730	726	10531

Tabela 6.13: Tipo e quantidade de mensagens enviadas por processo - *Global* - ambiente A_1.

GlobalAdapt - ambiente A_2																	
Tipo da Mensagem	***0***	***1***	***2***	***3***	***4***	***5***	***6***	***7***	***8***	***9***	***10***	***11***	***12***	***13***	***14***	***15***	***TOTAL***
Suspectend	30	105	150	45	120	165	180	15	210	195	225	60	135	75	90	45	1845
Primal	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Branch	15	22	27	36	46	38	66	58	44	43	25	26	48	68	64	18	644
Notnow	3	4	0	7	5	9	6	11	11	13	10	5	4	9	10	6	113
Idle	37	38	52	51	63	71	65	71	71	61	56	47	44	48	52	47	874
Loadrequest	90	52	35	82	58	77	101	142	119	109	120	148	82	99	132	170	1616
Suspeita_Cluster	16	56	80	24	64	88	96	8	112	104	120	32	72	40	48	24	984
Primal_Cluster	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Branch_Cluster	5	5	6	7	8	3	31	58	0	0	1	1	7	8	13	18	171
Notnow_Cluster	0	0	0	0	0	1	3	10	0	0	0	1	0	1	3	5	24
Idle_Cluster	11	15	19	24	30	36	36	60	13	13	13	13	15	21	27	38	384
Lodrequest_Cluster	74	29	16	33	16	14	14	15	112	47	43	59	20	26	26	27	571
TOTAL	281	326	385	309	410	502	598	448	692	585	613	392	427	395	465	398	7226

Tabela 6.14: Tipo e quantidade de mensagens enviadas por processo - *GlobalAdapt* - ambiente A_2.

Global - ambiente A_2																	
Tipo da Mensagem	***0***	***1***	***2***	***3***	***4***	***5***	***6***	***7***	***8***	***9***	***10***	***11***	***12***	***13***	***14***	***15***	***TOTAL***
Suspectend	45	75	105	120	150	135	90	15	60	30	75	90	90	75	60	111	1326
Primal	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Branch	21	63	70	75	60	45	61	68	70	87	85	86	96	56	73	25	1041
Notnow	4	3	5	14	15	18	14	16	14	20	24	23	40	25	17	3	255
Idle	42	47	73	85	107	110	102	119	130	154	174	174	143	110	81	76	1727
Loadrequest	59	58	93	113	117	133	166	111	144	124	175	230	298	398	357	436	3012
Suspeita_Cluster	24	40	56	64	80	72	48	8	32	16	40	48	48	40	32	48	696
Primal_Cluster	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Branch_Cluster	2	15	18	19	18	15	39	66	5	3	4	1	2	7	9	8	231
Notnow_Cluster	0	0	1	3	1	6	10	15	1	0	1	1	1	1	2	3	46
Idle_Cluster	8	9	22	37	57	69	79	112	10	16	19	23	24	26	33	41	585
Lodrequest_Cluster	51	37	43	34	24	21	23	8	137	77	77	70	69	74	62	47	854
TOTAL	256	347	486	564	629	624	632	538	603	527	674	746	811	812	726	798	9773

Tabela 6.15: Tipo e quantidade de mensagens enviadas por processo - *Global* - ambiente A_2.

GlobalAdapt - ambiente A_3																	
Tipo da Mensagem	***0***	***1***	***2***	***3***	***4***	***5***	***6***	***7***	***8***	***9***	***10***	***11***	***12***	***13***	***14***	***15***	***TOTAL***
Suspectend	75	30	45	210	90	165	195	240	120	135	60	150	90	180	225	15	2025
Primal	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Branch	15	16	17	23	45	51	54	44	50	45	38	53	52	26	17	11	557
Notnow	2	0	2	4	3	7	6	5	7	14	4	7	21	5	3	5	95
Idle	26	27	25	29	28	31	39	46	39	53	59	46	40	38	21	26	573
Loadrequest	39	46	50	25	52	62	71	81	93	64	105	108	100	156	130	28	1210
Suspeita_Cluster	40	16	24	112	48	88	104	128	64	72	32	80	48	96	120	8	1080
Primal_Cluster	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Branch_Cluster	2	3	2	3	4	11	17	44	0	0	0	1	4	3	8	11	113
Notnow_Cluster	0	0	1	0	0	0	2	5	0	0	0	0	0	0	2	5	15
Idle_Cluster	8	9	11	13	15	18	26	39	8	8	8	8	9	13	14	19	226
Lodrequest_Cluster	32	27	22	8	8	8	8	8	86	28	29	20	15	21	18	8	346
TOTAL	239	174	199	427	293	441	522	640	467	419	335	473	379	538	558	136	6240

Tabela 6.16: Tipo e quantidade de mensagens enviadas por processo - *GlobalAdapt* - ambiente A_3.

Global - ambiente A_3																	
Tipo da Mensagem	***0***	***1***	***2***	***3***	***4***	***5***	***6***	***7***	***8***	***9***	***10***	***11***	***12***	***13***	***14***	***15***	***TOTAL***
Suspectend	135	120	30	45	75	90	150	165	150	165	195	180	15	75	105	89	1784
Primal	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Branch	24	33	48	57	48	56	66	40	32	50	39	17	48	41	23	31	653
Notnow	3	1	4	12	8	15	16	12	4	15	13	7	8	6	1	4	129
Idle	59	62	67	85	104	86	81	62	55	49	43	28	32	42	32	41	928
Loadrequest	33	49	54	62	105	147	144	211	146	96	151	177	52	101	133	62	1723
Suspeita_Cluster	72	64	16	24	40	48	80	88	80	88	104	96	8	40	56	32	936
Primal_Cluster	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Branch_Cluster	5	4	10	10	4	7	16	40	0	0	0	0	4	4	11	31	146
Notnow_Cluster	0	0	0	1	1	1	4	12	0	0	0	0	1	2	0	4	26
Idle_Cluster	16	20	23	31	40	43	47	55	8	8	8	8	8	12	18	26	371
Lodrequest_Cluster	26	23	18	19	21	19	14	13	131	56	59	59	22	25	22	8	535
TOTAL	373	376	270	346	446	512	618	698	606	527	612	572	198	348	401	328	7231

Tabela 6.17: Tipo e quantidade de mensagens enviadas por processo - *Global* - ambiente A_3.

6.3 *Híbrida* x *HibridaAdapt*

Nesta seção são apresentados os resultados do segundo teste, que consistiu na execução simultânea das estratégias *Híbrida* e *HibridaAdapt* para os ambientes A_1, A_2 e A_3. O ambiente, como no primeiro teste, é composto por 2 *clusters* contendo 8 processadores cada um.

Como na estratégia *GlobalAdapt*, a estratégia *HibridaAdapt* diminuiu o número de mensagens trocadas entre os processos, com exceção das mensagens externas para o ambiente A_3. Estes valores podem ser vistos na Tabela 6.18, onde é apresentada a média da quantidade de mensagens enviadas pelos processos para cada ambiente utilizado. Como na seção anterior, as colunas *Int* e *Ext* indicam o número de mensagens internas e externas aos *clusters*. Se for considerado o resultado médio, apresentados na Tabela 6.22, a estratégia adaptativa, considerando número de mensagens internas, obteve o ganho de 49,04% pra o ambiente A_1, 37,24% para A_2 e 34,58% para A_3. Para mensagens externas a melhora foi de 12,36%, 26,24% e -38,70%, repectivamente. Este último valor é conseqüência dos valores obtidos para as istâncias *i320-341*, *i320-345* e *i640-212* em um ambiente heterogêneo e com presença de carga externa. Isto mostra que a estratégia *HibridaAdapt* não diminui o *overhead* de comunicação para este ambiente. Quando analisada a média do número de mensagens enviadas a estratégia *HibridaAdapt* se mostra melhor que a *Híbrida*. A justificativa para este comportamento está melhor explicado na análise das tabelas seguintes. Apesar de não terem sido executadas no mesmo ambiente, é importante notar que a quantidade de mensagens externas diminui quantitativamente em relação as estratégias apresentadas na seção anterior.

Instâncias	***Híbrida***						***HibridaAdapt***					
	A_1		A_2		A_3		A_1		A_2		A_3	
	Int	***Ext***	***Int***	***Ext***	***Int***	***Ext***	***Int***	***Ext***	***Int***	***Ext***	***Int***	***Ext***
320-311	7459,2	164,6	8728,8	262,6	6596,0	355,4	**5297,6**	**71,0**	**6148,0**	**122,0**	**4359,2**	**322,2**
320-313	9557,0	196,6	9244,0	191,8	13194,2	488,6	**4697,0**	**39,0**	**6038,8**	**88,2**	**7889,8**	**294,2**
320-314	35528,8	1083,4	33872,8	1115,0	24268,8	753,0	**13399,8**	**157,8**	**12876,4**	**209,8**	**12098,8**	**183,8**
320-341	7978,4	**44,2**	7155,4	71,0	6580,2	**40,6**	**3631,2**	48,6	**3309,6**	**43,0**	**3308,0**	87,8
320-343	2524,8	**22,2**	2225,2	**20,6**	2202,0	**27,4**	**1957,0**	49,0	**1892,0**	44,6	**1751,2**	43,4
320-344	3506,4	**31,8**	3509,4	71,0	3150,6	82,2	**2489,2**	48,2	**2414,8**	**54,2**	**2170,0**	**58,6**
320-345	5848,8	**34,6**	5195,0	**67,4**	4845,0	**41,8**	**3658,2**	69,8	**3885,8**	90,2	**4144,2**	193,4
640-212	2706,0	**30,60**	2914,6	45,4	3291,0	51,4	**2072,6**	31,0	**2370,4**	**42,2**	**2283,6**	**31,8**
640-213	5496,6	127,0	6186,6	153,0	8000,4	261,8	**4045,6**	**58,2**	**4447,2**	**93,8**	**6044,8**	**172,6**
640-214	15120,2	230,6	16728,4	334,6	16621,0	**149,8**	**9071,0**	**88,2**	**9054,0**	**128,2**	**12838,6**	431,4
640-215	19961,4	430,2	22481,4	717,0	17915,2	431,4	**6964,0**	**73,8**	**7721,2**	**116,6**	**6720,6**	**112,2**

Tabela 6.18: Comparação entre as estratégias *Híbrida* e *HibridaAdapt* quanto à quantidade de mensagens enviadas pelos processos.

Instâncias	A_1							
	Híbrida				***HibridaAdapt***			
	TCPU	***Tax***	***Effo***	***Effs***	***TCPU***	***Tax***	***Effo***	***Effs***
320-311	8993,29	0,044	22,60	1,30	**8861,60**	**0,031**	**32,20**	**1,32**
320-313	16197,25	0,028	37,20	1,44	**15946,43**	**0,013**	**83,80**	**1,46**
320-314	36637,47	0,055	18,00	0,94	**35470,25**	**0,030**	**36,40**	**0,97**
320-341	**8184,14**	**0,115**	**8,20**	**0,64**	8550,99	0,168	5,80	0,61
320-343	**1934,72**	0,216	4,00	**0,68**	1989,55	**0,197**	**4,40**	0,66
320-344	3324,86	0,146	6,40	0,85	**3245,18**	**0,123**	**7,80**	**0,87**
320-345	5317,30	0,145	6,60	0,75	**5070,48**	**0,118**	**8,00**	**0,78**
640-212	5930,13	0,043	23,00	1,00	**5867,59**	**0,034**	**29,20**	**1,01**
640-213	8416,51	0,051	19,20	1,09	**8338,62**	**0,043**	**23,60**	**1,10**
640-214	**71716,22**	0,018	64,20	**0,99**	72075,35	**0,014**	**116,40**	0,98
640-215	53872,19	0,030	32,80	0,99	**53118,12**	**0,017**	**74,80**	**1,00**

Tabela 6.19: Comparação entre as estratégias *Híbrida* e *HibridaAdapt* quanto aos parâmetros de eficiência e tempo - ambiente A_1.

A eficiência das estratégias *Híbrida* e *HibridaAdapt* são mostradas nas Tabelas 6.19, 6.20 e 6.21, sendo 6.19 os valores obtidos quando executadas no ambiente A_1, 6.20 em A_2 e 6.21 em A_3. É interessante verificar que a ociosidade dos processos quando utilizada a estratégia *HibridaAdapt* é reduzida visto que a eficiência $Effo$ aumentou em média em relação à *Híbrida* em 50,27% no ambiente A_1, 69,84% em A_2 e 40,36% em A_3. É notável uma degradação da eficiência da técnica no último ambiente. Possivelmente isso ococorre porque a quantidade de carga enviada em resposta a um $LOADREQUEST\_CLUSTER$ não foi devidamente calibrada. O cálculo da quantidade de carga transferida entre os *clusters* é baseado na média do desempenho apresentado pelos processadores. Entretanto, este valor pode não ser justo, uma vez que nada impede dos processadores do *cluster* receberem ou terminarem a execução de carga externa no espaço de tempo relativo ao envio do desempenho até o recebimento de carga, provocando desbalanceamento entre os mesmos.

Esta observação pode ser constatada quando analisadas e comparadas as Tabelas 6.23 e 6.24. A primeira delas apresenta a média da diferença de ociosidade apresentada pelos *clusters*. Isto quer dizer que, para a instância *i320-311* apresentada na Tabela 6.23, na estratégia *HibridaAdapt*, um dos *clusters* ficou em média 170,31 segundos mais ocioso que o outro. A linha "Melhora" apresenta a média de melhoria da estratégia *HibridaAdapt* em relação a *Híbrida*. Observando as instâncias individualmente, pode-se notar que, quando o valor da diferença de ociosidade aumenta, a diferença obtida em tempo de relógio de uma estratégia para a outra aumenta proporcionalmente. Para exemplificar, considere a instância *i640-214*. Esta é a maior instância do conjunto testado e teoricamente, quanto maior a instância e mais diversificado o ambiente, melhores resultados são obtidos pelas

Instâncias	A_2							
	Híbrida				***HibridaAdapt***			
	TCPU	***Tax***	***Effo***	***Effs***	***TCPU***	***Tax***	***Effo***	***Effs***
320-311	9116,83	0,49	17,80	1,29	**8886,04**	**0,036**	**28,40**	**1,32**
320-313	16237,42	0,024	41,40	1,44	**15866,59**	**0,017**	**57,20**	**1,47**
320-314	36336,61	0,049	20,20	0,95	**35349,87**	**0,021**	**53,60**	**0,97**
320-341	8073,38	0,108	9,00	0,65	**7499,92**	**0,056**	**17,60**	**0,70**
320-343	1**827,90**	0,177	**5,20**	**0,72**	1868,44	**0,176**	**5,20**	0,70
320-344	3329,69	0,144	6,60	0,85	**3198,47**	**0,108**	**8,80**	**0,89**
320-345	5156,86	0,129	7,20	0,77	**5035,56**	**0,114**	**8,40**	**0,79**
640-212	5971,99	0,054	18,40	**1,00**	**5918,25**	**0,042**	**24,60**	**1,00**
640-213	8374,54	0,055	17,60	1,09	**8341,05**	**0,045**	**22,00**	**1,10**
640-214	71849,27	0,021	55,40	**0,99**	**71529,97**	**0,013**	**93,00**	**0,99**
640-215	53795,16	0,035	28,40	0,99	**52235,21**	**0,011**	**94,60**	**1,02**

Tabela 6.20: Comparação entre as estratégias *Híbrida* e *HibridaAdapt* quanto aos parâmetros de eficiência e tempo - ambiente A_2.

Instâncias	A_3							
	Híbrida				***HibridaAdapt***			
	TCPU	***Tax***	***Effo***	***Effs***	***TCPU***	***Tax***	***Effo***	***Effs***
320-311	8985,79	0,073	13,20	1,31	**8892,16**	**0,063**	**15,20**	**1,32**
320-313	16146,1	0,053	18,40	1,45	**15860,78**	**0,038**	**26,40**	**1,47**
320-314	34310,2	0,030	32,80	1,00	**33876,48**	**0,016**	**66,00**	**1,01**
320-341	8328,83	0,114	8,20	0,63	**7844,84**	**0,074**	**13,20**	**0,67**
320-343	**1908,62**	0,198	4,60	**0,69**	1951,87	**0,180**	**5,00**	0,67
320-344	3445,71	0,150	6,20	0,82	**3373,26**	**0,127**	**7,60**	**0,84**
320-345	**5332,45**	**0,126**	**7,60**	**0,74**	5497,84	0,144	7,40	0,72
640-212	6186,65	0,056	17,80	**0,96**	**6067,99**	**0,040**	**24,60**	**0,96**
640-213	8618,88	0,082	11,60	1,06	**8409,44**	**0,066**	**15,00**	**1,09**
640-214	**72102,05**	**0,019**	**58,40**	**0,98**	72359,66	0,022	47,40	**0,98**
640-215	52734,58	0,026	38,40	1,01	**52170,06**	**0,011**	**94,40**	**1,02**

Tabela 6.21: Comparação entre as estratégias *Híbrida* e *HibridaAdapt* quanto aos parâmetros de eficiência e tempo - ambiente A_3 .

Instâncias	***INT***	***EXT***	***TCPU***	***Tax***	***Effo***	***Effs***
A_1	40,04	12,36	0,80	20,68	50,27	0,87
A_2	37,24	26,74	2,12	32,10	69,84	2,22
A_3	34,58	-38,70	1,06	20,43	40,36	0,95

Tabela 6.22: Ganho médio percentual.

técnicas adaptativas. Entretanto, no primeiro ambiente, a estratégia *HibridaAdapt* melhorou cerca de 11,63% a diferença de ociosidade entre os *clusters* e cerca de 0,09% o tempo de execução. Para A_2 houve um ganho de 14,9% em diferença de ociosidade e 1,6% em tempo de relógio. No último ambiente, devido a maior diversidade de carga, a *HibridaAdapt* não foi apta a enviar uma quantidade de carga suficiente de modo a evitar a ociosidade do *cluster*, já que a diferença de ociosidade entre os *clusters* foi 23,48% maior do que quando utilizada a estratégia proposta por [Drummond et al. 2005]. Conseqüentemente, o tempo de execução da *HibridaAdapt* foi 1,3% maior do que a *Híbrida*. Note que esta afirmação não é verdadeira para a instância *i320-343*. Como já discutido, esta instância é pequena, a menor do conjunto testado, sendo desnecessário o uso de estratégias de balanceamento robustas.

Instâncias	A_1		A_2		A_3	
	Híbrida	***Híbrida Adapt***	***Híbrida***	***Híbrida Adapt***	***Híbrida***	***Híbrida Adapt***
320-311	299,97	**170,31**	369,77	**226,98**	487,33	**373,15**
320-313	347,03	**108,96**	320,24	**210,78**	684,91	**444,59**
320-314	1727,60	**631,57**	1420,93	**514,95**	681,85	**314,78**
320-341	**411,95**	888,02	367,42	**50,63**	332,48	**107,09**
320-343	152,97	**126,16**	**110,52**	117,35	98,12	**94,33**
320-344	207,01	**97,09**	192,32	**68,38**	163,52	**142,38**
320-345	290,99	**188,23**	277,13	**176,41**	**127,61**	260,85
640-212	57,25	**35,13**	91,15	**46,86**	124,25	**97,25**
640-213	277,30	**170,89**	308,24	**228,82**	481,63	**358,34**
640-214	607,98	**537,29**	684,84	**582,79**	**563,01**	695,20
640-215	1401,61	**705,95**	1461,72	**404,16**	1043,83	**395,50**
Melhora	**27,63%**		**43,54%**		**16,23%**	

Tabela 6.23: Comparação entre as estratégias *Híbrida* e *HibridaAdapt* - Diferença de ociosidade.

Apesar da diferença de ociosidade entre os *clusters*, a estratégia adaptativa mostrou-se melhor que a *Híbrida* uma vez que diminui quantitativamente a ociosidade dos processos dentro dos *clusters*. Ainda sobre as Tabelas 6.19, 6.20 e 6.21, é fácil perceber através da coluna Tax, que os processos ficaram menos ociosos. A eficiência $Effs$ obtida é quase a mesma nas duas estratégias. O valor de $Effs$ é proporcional ao tempo de CPU obtido pelo algoritmo e, como já discutido na seção anterior, apesar da estratégia adaptativa diminuir o tempo de CPU relativo a ociosidade dos processos, utiliza de CPU para realizar os cálculos relativos ao desempenho dos processadores, a quantidade de carga a ser transferida e a retirada da mesma da fila.

A quantidade de nós resolvidos pelas estratégias são mostrados na Tabela 6.25. A diferença entre o número de nós resolvidos por cada estratégia é, em média, menos de

Instâncias	A_1		A_2		A_3	
	Híbrida	***Híbrida Adapt***	***Híbrida***	***Híbrida Adapt***	***Híbrida***	***Híbrida Adapt***
320-311	1277,10	**1250,72**	1902,85	**1813,88**	2399,44	**2395,26**
320-313	2215,75	**2156,23**	3186,02	**3104,19**	4342,50	**4190,27**
320-314	5113,08	**4883,42**	7415,89	**7004,22**	8268,18	**8001,49**
320-341	**1348,09**	1475,05	1869,90	**1654,58**	2173,73	**1987,80**
320-343	**380,63**	390,48	**510,77**	526,86	**550,88**	551,88
320-344	571,23	**551,78**	840,41	**782,96**	952,99	**920,45**
320-345	889,87	**845,89**	1288,22	**1248,26**	**1503,94**	1605,77
640-212	1104,32	**1083,34**	1117,01	**1094,65**	1511,59	**1465,48**
640-213	1199,69	**1184,25**	1797,91	**1765,34**	2097,16	**2015,64**
640-214	9489,91	**9480,92**	14533,64	**14301,26**	**15554,65**	15756,57
640-215	7453,67	**7272,12**	11059,67	**10455,89**	11577,80	**10663,21**
Melhora	**1,03%**		**3,82%**		**2,31%**	

Tabela 6.24: Comparação entre as estratégias *Híbrida* e *HibridaAdapt* - tempo relógio (em segundos).

0,5% para todos os ambientes, comprovando que a comparação entre os algoritmos é justa.

Instâncias	A_1		A_2		A_3	
	Híbrida	***Híbrida Adapt***	***Híbrida***	***Híbrida Adapt***	***Híbrida***	***Híbrida Adapt***
320-311	46547,80	47049,80	46621,00	46698,20	46461,80	46811,00
320-313	99686,20	100563,60	100113,00	99061,80	99860,20	100040,00
320-314	189803,00	190495,60	189846,20	191107,80	187701,00	189265,60
320-341	16737,80	17047,00	16833,40	16893,00	16690,60	16911,00
320-343	2849,00	2905,80	2932,60	2879,40	2847,80	2848,60
320-344	6399,40	6299,00	6395,00	6399,40	6494,20	6371,40
320-345	10519,00	10268,40	10539,00	10464,20	10451,40	10149,00
640-212	13389,00	13478,20	13451	13371	13417,4	13453,4
640-213	26178,20	25908,20	25864,20	25930,20	26026,60	25723,80
640-214	192417,40	194811,00	191598,20	193072,80	193693,80	194381,20
640-215	166754,60	167307,60	166341,00	166231,00	166003,00	166681,60

Tabela 6.25: Comparação entre as estratégias *Híbrida* e *HibridaAdapt* quanto ao número de nós resolvidos.

Para melhor entender o funcionamento da estratégia híbrida, considere as Tabelas 6.26, 6.27 e 6.28 que apresentam detalhes da execução da instância *i640-212* nos ambientes A_1, A_2 e A_3, respectivamente, onde $TCPU$ são os tempos de CPU e tempo de CPU normalizado gasto para resolver a instância, $QNOS$ e $QNOSP$, são o número de nós resolvidos pelo processo e o número de nós da primeira subárvore resolvida pelo processo, TOC e $TOCN$ são os tempos de CPU e tempo de CPU normalizado em que o processo ficou ocioso, finalmente, $MSGIN$ e $MSGEXT$, são o número de mensagens internas e externas enviadas pelo processo. Analisando estas tabelas, pode-se comprovar que somente os líderes enviam mensagens para processo de *cluster* diferente ($MSGEXT$), ou seja, para

o processo líder do outro *cluster*. Pode-se comprovar também, que o *cluster* que totaliza uma maior quantidade de nós resolvidos apresentam um tempo de ociosidade menor. Além disso, para todos os ambientes, na maioria das vezes, os processos de maiores identificaões possuem um tempo de ociosidade maior, que é conseqüência da ordem de envio de pedidos de carga. Comparando a estratégia *HibridaAdapt*, Tabela 6.28, com a *Híbrida*, Tabela 6.29, a a ociosidade dos processos e o número de mensagens internas enviadas por processo é menor. A respeito da mensagem externa, neste caso em especial, a estratégia *HibridaAdapt* precisou enviar mais mensagens que a *Híbrida* para alcançar um melhor balanceamento entre seus *clusters*. Entretanto, de acordo com a Tabela 6.18, em média, a estratégia *HibridaAdapt* envia menos mensagens do que a *Híbrida*.

HibridaAdapt - ambiente A_1																
	0	***1***	***2***	***3***	***4***	***5***	***6***	***7***	***8***	***9***	***10***	***11***	***12***	***13***	***14***	***15***
TCPU	481,09	496,68	324,92	333,05	464,70	474,41	322,94	321,48	351,22	356,07	350,13	346,75	326,84	335,35	330,32	340,29
QNOS	866	1072	679	740	972	879	664	757	874	900	890	821	832	1015	899	843
QNOSP	621	357	189	142	183	63	184	99	799	387	246	150	381	282	63	17
TOC	11,66	19,85	16,14	12,80	31,51	27,62	18,25	16,67	6,88	3,48	5,98	6,82	15,12	13,71	12,25	10,73
MSGIN	140	114	125	135	183	180	152	102	91	62	99	134	160	184	139	139
MSGEXT	23	0	0	0	0	0	0	0	22	0	0	0	0	0	0	0

Tabela 6.26: Dados por processo quando utilizada a estratégia *HibridaAdapt* - ambiente A_1.

HibridaAdapt - ambiente A_2																
	0	***1***	***2***	***3***	***4***	***5***	***6***	***7***	***8***	***9***	***10***	***11***	***12***	***13***	***14***	***15***
TCPU	510,33	264,35	351,89	530,41	522,51	258,46	241,68	474,84	318,1	487,44	323,05	317,39	472,65	185,11	297,19	430,53
QNOS	978	583	762	1197	1055	512	537	1050	715	1101	717	682	1251	447	692	1114
QNOSP	656	198	264	142	296	63	180	99	425	417	298	150	541	139	124	17
TOC	21,61	4,87	7,61	9,8	10,63	7,91	16	30,56	16,56	27,43	17,29	19,9	33,51	14,53	28,31	50,6
MSGIN	176	83	72	85	139	177	215	216	361	198	171	160	190	182	233	261
MSGEXT	38	0	0	0	0	0	0	0	37	0	0	0	0	0	0	0

Tabela 6.27: Dados por processo quando utilizada a estratégia *HibridaAdapt* - ambiente A_2.

HibridaAdapt - ambiente A_3																
	0	***1***	***2***	***3***	***4***	***5***	***6***	***7***	***8***	***9***	***10***	***11***	***12***	***13***	***14***	***15***
TCPU	559,56	360,39	359,66	653,81	352,45	358,79	518,49	536,71	400,82	462,46	391,93	391,96	415,82	212,02	351,34	612,85
TCPUN	559,56	360,39	359,66	457,67	373,6	358,79	554,78	536,71	400,82	231,23	391,93	391,96	203,75	212,02	351,34	435,12
QNOS	1088	688	720	1071	729	679	1106	1161	972	593	877	939	517	539	921	1103
QNOSP	1120	364	247	135	166	63	185	99	885	229	217	150	182	249	259	17
TOC	18,14	17,32	19,12	29,13	14,29	18,96	30,14	30,98	3,42	0,83	6,02	6,61	2,83	11,89	23,79	13,23
TOCN	18,14	17,32	19,12	20,39	15,15	18,96	32,25	30,98	3,42	0,42	6,02	6,61	1,39	11,89	23,79	9,39
MSGIN	143	66	99	122	155	199	204	150	81	64	68	82	106	137	146	38
MSGEXT	30	0	0	0	0	0	0	0	29	0	0	0	0	0	0	0

Tabela 6.28: Dados por processo quando utilizada a estratégia *HibridaAdapt* - ambiente A_3.

Híbrida - ambiente A_3																
	0	***1***	***2***	***3***	***4***	***5***	***6***	***7***	***8***	***9***	***10***	***11***	***12***	***13***	***14***	***15***
TCPU	569,98	384,96	383,38	684,15	356,69	363,48	529,88	517,55	426,5	510,49	409,43	409	423,92	224,11	360,83	591,39
TCPUN	569,98	384,96	383,38	478,91	378,09	363,48	566,97	517,55	426,5	255,25	409,43	409	207,72	224,11	360,83	419,89
QNOS	1142	724	900	918	645	721	1113	997	981	617	944	912	553	534	804	1040
QNOSP	968	348	353	142	242	63	311	99	983	241	346	150	320	268	259	17
TOC	20,51	24,45	23,99	57,83	29,21	31,61	63,01	70,08	11,9	0,41	15,6	15,78	5,2	15,8	37,42	24,39
TOCN	20,51	24,45	23,99	40,48	30,96	31,61	67,42	70,08	11,9	0,21	15,6	15,78	2,55	15,8	37,42	17,32
MSGIN	294	242	213	236	250	288	342	347	116	110	149	198	173	254	166	110
MSGEXT	5	0	0	0	0	0	0	0	44	0	0	0	0	0	0	0

Tabela 6.29: Dados por processo quando utilizada a estratégia *Híbrida* - ambiente A_3.

As Tabelas 6.30 e 6.32 apresentam os tipos e a quantidade de mensagens enviadas por processo na execução do algoritmo B&B para a instância *i640-212* quando utilizada a estratégia de balanceamento *HibridaAdapt* e as Tabelas 6.31 e 6.33 quando utilizada a *Híbrida* nos ambientes A_1 e A_3. É interessante ressaltar, para as duas estratégias, que as mensagens enviadas entre processos de *clusters* diferentes são enviadas somente pelos processos líderes, 0 e 8. No ambiente A_1 a estratégia *HibridaAdapt* precisou enviar um maior número de *LOADREQUEST* para manter os processos balanceados. Esse fato é conseqüência dos processadores apresentarem quase os mesmos desempenhos e do tamanho da instância. Neste caso, evitar a ociosidade entre os processos dividindo, proporcionalmente, a carga, principalmente perto do final da execução, faz com que os processadores se tornem ociosos rapidamente e iniciem uma nova busca por carga, ou seja, a carga pendende perto do final da execução é muito pequena para ser divida. Entretanto, para instâncias maiores e ambientes mais heterogêneos e compartilhados, isto é válido, pois, apesar da carga pendente ser pequena, pode existir algum processador que irá solucioná-la mais rapidamente do que aquele que a está resolvendo, o que pode ser verificado nas Tabelas 6.32 e 6.33. Nestas Tabelas também podemos observar que o número de mensagens do tipo *LOADREQUEST* enviadas é quantitativamente menor na estratégia *HibridaAdpat*.

A estratégia *HibridaAdapt* no ambiente A_3, diminuiu também as mensagens do tipo *LOADREQUEST- _CLUSTER*, o que pode ser explicado pelo envio de uma quantidade de carga proporcional ao desempenho do *cluster*, evitando a ociosidade do mesmo. É válido perceber que a nova estratégia minimiza o problema da estratégia *Híbrida* discutido em [Drummond et al. 2005], que diz que a estratégia *Híbrida* não é melhor do que a *Global* pela pequena quantidade de carga transferida entre *clusters* mediante requisição de carga, fazendo com que os *clusters* se tornem ociosos mais rapidamente.

HibridaAdapt - ambiente A_1																	
Tipo da Mensagem	***0***	***1***	***2***	***3***	***4***	***5***	***6***	***7***	***8***	***9***	***10***	***11***	***12***	***13***	***14***	***15***	***TOTAL***
Suspectend	7	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	22
Primal	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Branch	12	37	25	58	34	29	29	22	6	21	29	53	36	34	20	19	464
Notnow	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1
Idle	20	16	20	24	39	30	24	26	16	12	15	20	36	27	29	16	370
Loadrequest	70	43	47	31	66	76	62	43	38	20	37	36	50	76	56	69	820
Suspeita_Cluster	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	2
Primal_Cluster	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Branch_Cluster	1	0	0	0	0	0	0	0	3	0	0	0	0	0	0	0	4
Notnow_Cluster	0	0	0	0	0	0	0	0	16	0	0	0	0	0	0	0	16
Idle_Cluster	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	2
Lodrequest_Cluster	20	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	21
TOTAL	132	97	93	114	140	137	116	92	83	54	82	110	123	138	106	105	1722

Tabela 6.30: Tipo e quantidade de mensagens enviadas por processo - *HibridaAdapt* - ambiente A_1.

Híbrida - ambiente A_1																	
Tipo da Mensagem	***0***	***1***	***2***	***3***	***4***	***5***	***6***	***7***	***8***	***9***	***10***	***11***	***12***	***13***	***14***	***15***	***TOTAL***
Suspectend	4	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	19
Primal	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Branch	13	23	44	28	34	32	28	3	13	40	41	17	29	64	42	2	453
Notnow	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Idle	10	15	26	38	38	30	22	11	9	11	33	38	33	32	42	18	406
Loadrequest	26	16	18	44	51	77	80	76	40	21	24	51	46	59	92	6	727
Suspeita_Cluster	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	2
Primal_Cluster	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Branch_Cluster	0	0	0	0	0	0	0	0	11	0	0	0	0	0	0	0	11
Notnow_Cluster	1	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	3
Idle_Cluster	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	2
Lodrequest_Cluster	14	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	15
TOTAL	70	55	89	111	124	140	131	91	79	73	99	107	109	156	177	27	1638

Tabela 6.31: Tipo e quantidade de mensagens enviadas por processo - *Híbrida* - ambiente A_1.

HibridaAdapt - ambiente A_3																	
Tipo da Mensagem	***0***	***1***	***2***	***3***	***4***	***5***	***6***	***7***	***8***	***9***	***10***	***11***	***12***	***13***	***14***	***15***	***TOTAL***
Suspectend	8	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	30
Primal	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Branch	17	20	25	31	44	52	23	10	14	12	14	27	38	32	5	8	372
Notnow	0	0	0	0	0	0	0	0	22	0	0	0	0	0	0	0	22
Idle	11	21	24	29	35	38	37	25	14	14	13	11	19	26	11	12	340
Loadrequest	69	17	31	38	48	64	87	77	15	25	28	27	29	46	81	12	694
Suspeita_Cluster	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	2
Primal_Cluster	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Branch_Cluster	1	0	0	0	0	0	0	0	4	0	0	0	0	0	0	0	5
Notnow_Cluster	0	0	0	0	0	0	0	0	22	0	0	0	0	0	0	0	22
Idle_Cluster	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	2
Lodrequest_Cluster	27	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	28
TOTAL	135	60	82	100	129	156	149	114	95	52	56	66	87	105	98	33	1517

Tabela 6.32: Tipo e quantidade de mensagens enviadas por processo - *HibridaAdapt* - ambiente A_3.

Híbrida - ambiente A_3																	
Tipo da Mensagem	***0***	***1***	***2***	***3***	***4***	***5***	***6***	***7***	***8***	***9***	***10***	***11***	***12***	***13***	***14***	***15***	***TOTAL***
Suspectend	11	5	5	5	5	5	5	5	1	1	1	1	1	1	1	1	54
Primal	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Branch	53	20	35	63	55	68	54	29	18	42	27	31	48	29	5	27	604
Notnow	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Idle	52	52	44	47	75	85	91	69	30	37	52	37	43	29	8	10	761
Loadrequest	124	113	98	82	78	81	129	165	15	21	43	87	56	132	101	11	1336
Suspeita_Cluster	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	2
Primal_Cluster	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Branch_Cluster	1	0	0	0	0	0	0	0	7	0	0	0	0	0	0	0	8
Notnow_Cluster	0	0	0	0	0	0	0	0	34	0	0	0	0	0	0	0	34
Idle_Cluster	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	2
Lodrequest_Cluster	42	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	43
TOTAL	285	190	182	197	213	239	279	268	108	101	123	156	148	191	115	49	2844

Tabela 6.33: Tipo e quantidade de mensagens enviadas por processo - *Híbrida* - ambiente A_3.

6.4 *MeAdapt* x *GlobalAdapt* x *HibridaAdapt*

Os resultados apresentados nas seções anteriores mostram que a utilização das estratégias adaptativas evitam a ociosidade dos processos e melhoraram a eficiência do algoritmo *Branch-and-Bound* paralelo. Esta seção apresenta os resultados da comparação entre as estratégias totalmente distribuídas previamente testadas, *GlobalAdapt* e *HibridaAdapt*, e a estratégia centralizada, *MeAdapt*. As três estratégias foram executadas simultaneamente para os ambientes A_1 e A_3. O ambiente é o mesmo dos testes anteriores: 2 *clusters* contendo 8 processadores cada um.

A Tabela 6.34 apresenta a média do número de mensagens trocadas entre processos. É fácil perceber que a estratégia *GlobalAdapt* enviou mais mensagens que as outras estratégias nos dois ambientes. A *HibridaAdapt* diminuiu quantitativamante o número de mensagens externas enviadas quando comparada com a *GlobalAdapt*, comprovando-se que a mesma reduz a comunicação entre processos de *clusters* diferentes. A estratégia *MeAdapt* foi a que apresentou a menor quantidade de mensagens enviadas entre os processos. Este fato é conseqüência de uma das vantagens apresentadas pelo paradigma mestre-escravo. O processo mestre, por manter a informação global, e neste caso atualizada, sobre desempenho de todos os processos, envia uma quantidade de carga para um escravo ocioso proporcional ao desempenho do mesmo em relação a todos os outros escravos, ou seja, o balanceamento é global, enquanto nas outras estratégias o balanceamento acontece somente entre dois processos ociosos, o processo que enviou e o que recebeu a requisição de carga. Além disso, o custo do procedimento de *Detecção de Terminação* é muito reduzido, uma vez que não são necessárias as mensagens referente à terminação como nas outras ténicas. O mestre, ao receber *IDLE* de todos os escravos na *busca por carga* considera que a aplicação chegou ao fim, pois não existe mais carga a ser resolvida. Logo, as mensagens *SUSPECTEND* e *SUSPECTEND_CLUSTER* são desnecessárias. Já as mensagens relativas ao balanceamento de carga como *LOADREQUEST*, *IDLE* ou *NOTNOW* são reduzidas ao número de requisições de carga enviadas pelo mestre.

Instâncias	***GlobalAdapt***				***HibridaAdapt***				***MeAdapt***			
	A_1		A_3		A_1		A_3		A_1		A_3	
	Int	***Ext***	***Int***	***Ext***	***Int***	***Ext***	***Int***	***Ext***	***Int***	***Ext***	***Int***	***Ext***
320-311	4224,60	**109,40**	4238,60	1864,60	5484,60	113,40	4769,20	**115,00**	**1758,00**	1137,00	**1698,20**	1063,40
320-313	6713,60	2697,80	5566,60	2223,40	10270,60	**153,40**	9309,00	**153,00**	**2175,60**	1389,60	**1792,20**	1159,20
320-314	8441,20	3356,40	8678,60	3543,20	16950,60	**285,00**	8238,80	**101,40**	**2349,40**	1509,00	**1882,40**	1216,20
320-341	7172,40	2843,80	7476,20	3000,60	3958,00	**57,00**	4803,40	**181,00**	**890,00**	578,20	**919,80**	622,20
320-343	5101,40	2264,80	4390,80	1944,60	1917,00	**66,60**	1653,80	**49,80**	**697,00**	458,60	**631,60**	404,80
320-344	5161,20	2278,20	4718,40	2023,40	2329,60	**53,80**	2853,40	**115,40**	**707,60**	436,60	**705,40**	425,80
320-345	7042,40	2828,80	6009,80	2480,80	4247,20	**98,60**	4121,00	**119,00**	**710,00**	441,00	**818,20**	477,60
640-212	5238,60	2190,00	5470,20	2338,00	2152,00	**33,40**	2244,40	**32,60**	**1044,20**	671,60	**1212,40**	747,80
640-213	5908,25	2473,00	6474,25	2766,75	3912,20	**72,20**	5524,20	**137,40**	**1227,80**	792,80	**1414,80**	913,60
640-214	10006,60	3628,60	10245,60	3796,40	12460,00	**285,00**	9025,80	**155,40**	**1896,00**	1201,80	**2100,20**	1304,60
640-215	7017,40	2809,20	5615,00	2238,00	6390,60	**69,00**	5166,40	**45,80**	**1675,60**	1052,40	**1499,00**	940,20
Média	9046,15		8827,62		6487,25		5355,93		**2254,53**		**2177,24**	

Tabela 6.34: Comparação das estratégias *GlobalAdapt*, *HibridaAdapt* e *MeAdapt* quanto à quantidade de mensagens enviadas pelos processos.

É interessante notar que, apesar da estratégia *MeAdapt* reduzir o número de mensagens trocadas entre os processos em relação às outras estratégias, possui uma desvantagem que tem como conseqüência o aumento no tempo de execução (relógio) do algoritmo. O processo mestre somente executa uma única subárvore e durante o restante da aplicação é responsável por manter os escravos balanceados. Logo, a execução do algoritmo B&B é realizada paralelamente, na maior parte do tempo, por $m-1$ processadores e não por m como nas outras estratégias. Para instâncias grandes, a perda de um processador implica diretamente no aumento do tempo de execução. Entretanto, para instâncias pequenas, considerando que as primeiras subárvores são maiores, o mestre, resolvendo a primeira subárvore, executa uma quantidade de carga proporcional a dos outros processos. Portanto, o paradigma mestre-escravo utilizado pela estratégia *MeAdapt* é mais eficiente quando executado para instâncias menores. A estratégia *GlobalAdapt* se mostra mais eficiente para as instâncias maiores e a estratégia *HibridaAdapt* para ambiente A_3 com instâncias maiores, se mostrou mais eficiente do que a estratégia *MeAdapt*. Os valores relativos aos tempos de execução são apresentados na Tabela 6.35.

Instâncias	A_1			A_3		
	Global Adapt	***Híbrida Adapt***	***MeAdapt***	***Global Adapt***	***Híbrida Adapt***	***MeAdapt***
320-311	**1740,31**	1826,46	1744,39	**2511,27**	2659,23	2586,98
320-313	**3179,18**	3367,29	3226,15	**4188,54**	4517,51	4428,86
320-314	**6805,84**	7183,52	7009,21	**9215,52**	9253,03	9381,67
320-341	1705,5	1943,85	**1443,64**	2585,93	2780,31	**2095,98**
320-343	728,16	768,19	**535,71**	736,4	732,26	**533,75**
320-344	850,94	893,25	**713,89**	1155,67	1291,69	**916,25**
320-345	1141,43	1206,42	**912,86**	1657,72	1842,31	**1385,91**
640-212	1263,51	1284,36	**1224,11**	1887,37	1904,01	**1840,58**
640-213	1680,8	1779,12	**1673,15**	2443,44	2641,08	**2419,77**
640-214	**14574,26**	15087,08	14932,19	**19133,45**	19629,31	19834,7
640-215	**10226,56**	10452,88	10530,88	**14902,31**	15158,2	15530,69

Tabela 6.35: Comparação entre as estratégias *GlobalAdapt*, *HibridaAdapt* e *MeAdapt* - tempo relógio (em segundos).

O tempo de CPU, a taxa de ociosidade e as eficiências, $Effo$ e $Effs$, são apresentadas na Tabela 6.36 para o ambiente A_1 e na Tabela 6.37 para o ambiente A_3. Os valores obtidos da eficiência em relação a quantidade de processos ociosos durante a execução da aplicação ($Effo$) confirmam o fato discutido acima, pois Tax e $Effo$ apresentam melhores valores para a estratégia *GlobalAdapt* nas instâncias maiores e para *MeAdapt* nas menores. É importante notar que, para instâncias maiores, o mestre atinge seu limite de carga lim mais rapidamente, ficando ocioso, ou seja, à espera de requerimentos de carga enquanto os outros processadores estão sobrecarregados.

Instâncias	Ambiente A_1											
	GlobalAdapt				***HibridaAdapt***				***MeAdapt***			
	TCPU	***Tax***	***Effo***	***Effs***	***TCPU***	***Tax***	***Effo***	***Effs***	***TCPU***	***Tax***	***Effo***	***Effs***
320-311	**8629,50**	**0,010**	**109,40**	**1,36**	8923,15	0,040	25,46	1,31	8934,49	0,025	40,53	1,31
320-313	**15876,69**	**0,009**	**113,08**	**1,47**	16583,31	0,041	36,64	1,41	16312,29	0,023	43,84	1,43
320-314	**34797,15**	**0,005**	**186,57**	**0,99**	36049,68	0,028	36,22	0,95	35339,60	0,019	52,74	0,97
320-341	7957,58	0,068	15,00	0,66	8618,68	0,172	6,74	0,61	**6924,86**	**0,037**	**27,24**	**0,76**
320-343	2112,32	0,198	5,23	0,62	2035,57	0,226	4,46	0,65	**1688,91**	**0,174**	**5,76**	**0,78**
320-344	3326,62	0,106	9,62	0,85	3310,05	0,135	7,55	0,86	**2923,91**	**0,075**	**13,34**	**0,97**
320-345	5215,04	0,102	10,02	0,76	5239,68	0,136	7,62	0,76	**4369,94**	**0,049**	**20,43**	**0,91**
640-212	6342,33	**0,036**	**29,62**	0,94	5952,28	0,041	24,54	0,94	**5947,86**	0,039	26,22	**1,00**
640-213	**8086,15**	**0,030**	**34,42**	**1,13**	8403,92	0,054	18,83	1,09	8213,77	**0,030**	33,63	1,11
640-214	**71288,02**	**0,006**	**177,40**	**0,99**	72626,22	0,023	48,49	0,98	72151,39	0,018	56,40	0,98
640-215	**52333,84**	**0,005**	**182,42**	**1,02**	52857,14	0,010	112,49	1,01	53403,70	0,019	52,96	1,00

Tabela 6.36: Comparação entre as estratégias *GlobalAdapt*, *HibridaAdapt* e *MeAdapt* - ambiente A_1.

Instâncias	Ambiente A_3											
	GlobalAdapt				***HibridaAdapt***				***MeAdapt***			
	TCPU	***Tax***	***Effo***	***Effs***	***TCPU***	***Tax***	***Effo***	***Effs***	***TCPU***	***Tax***	***Effo***	***Effs***
320-311	**8469,54**	**0,012**	**79,60**	**1,38**	8694,13	0,029	31,98	1,35	8713,33	0,031	29,78	1,35
320-313	**15193,27**	**0,008**	**117,03**	**1,54**	15888,69	0,030	43,09	1,47	15939,68	0,027	34,10	1,46
320-314	**34199,92**	**0,007**	**154,17**	**1,00**	34428,13	0,009	120,09	**1,00**	34737,32	0,025	36,90	0,99
320-341	8327,70	0,081	12,09	0,63	8558,47	0,110	8,35	0,61	**7254,00**	**0,036**	**26,20**	**0,72**
320-343	2023,20	0,167	5,97	0,65	1909,14	0,162	6,02	0,69	**1647,74**	**0,117**	**8,30**	**0,80**
320-344	3434,15	0,109	8,71	0,83	3671,96	0,160	6,17	0,77	**3006,05**	**0,069**	**13,90**	**0,94**
320-345	5202,81	0,087	10,90	0,76	5556,18	0,135	7,55	0,71	**4597,99**	**0,060**	**16,00**	**0,86**
640-212	6035,13	0,045	21,25	**0,99**	**6019,49**	**0,036**	**26,64**	**0,99**	6089,51	0,041	23,16	0,98
640-213	8208,32	**0,028**	**34,35**	1,11	8523,08	0,059	16,72	1,07	**8161,70**	0,035	26,39	**1,12**
640-214	**71374,08**	**0,007**	**140,49**	**0,99**	71572,08	0,014	74,12	**0,99**	72526,04	0,021	43,08	0,98
640-215	**52279,92**	**0,005**	**215,51**	**1,02**	52510,16	0,008	141,94	1,01	53758,58	0,024	37,62	0,99

Tabela 6.37: Comparação entre as estratégias *GlobalAdapt*, *HibridaAdapt* e *MeAdapt* - ambiente A_3.

A Tabela 6.38 apresenta a média da quantidade de nós resolvidos por cada processador em cada um dos ambientes. Pode-se comprovar que a comparação entre as estratégias é válida, uma vez que a diferença média de nós resolvidos pelas estratégias para cada instância é mínima, sendo que o algoritmo *GlobalAdapt* resolveu 1,44% de nós a menos que a *MeAdapt* e 0,87% a menos que o *HibridaAdapt*.

Instâncias	A_1			A_3		
	Global Adapt	***Híbrida Adapt***	***MeAdapt***	***Global Adapt***	***Híbrida Adapt***	***MeAdapt***
320-311	46694,6	46660,6	48544,8	46814,6	47086,2	48062,2
320-313	99622,6	100543,2	102634,4	98759	101181,6	102906,2
320-314	190932,8	192705,6	191031,2	191994,2	192772,2	191107
320-341	16921,4	16885	16779,8	16643	16783,8	16861,4
320-343	2887,4	2834,2	2766,6	2833	2834,2	2833
320-344	6365,8	6359,4	6503	6323,8	6454,6	6460,2
320-345	10168,6	10425	10458,2	10069,4	10252,6	10301,8
640-212	15889	13606,6	13808,6	13353,8	13644,2	13888,8
640-213	25624,2	25776	25961,2	25975,4	26048,6	25913,4
640-214	193682,6	193627,2	194933,6	193926,2	192410,2	194425,6
640-215	166626,6	167147,4	167982,2	166302,6	166055	166589,8

Tabela 6.38: Comparação entre as estratégias *GlobalAdapt*, *HibridaAdapt* e *MeAdapt* - número de nós resolvidos.

As Tabelas 6.39, 6.40 e 6.41 apresentam dados individuais dos processos que participam da solução da instância *i640-212*, respectivamente, para as estratégias *GlobalAdapt*, *HibridaAdap* e *MeAdap* no ambiente A_3. Estes resultados são interessantes para enfatizar as diferenças entre as estratégias durante a execução da aplicação. Na Tabela 6.39, pode-se perceber que todos os processos enviam tanto mensagens internas quanto externas, ou seja, não existe restrição de comunicação entre os mesmos. Já na Tabela 6.40, somente os processos líderes, no caso 0 e 8, enviam mensagens externas. Por fim, na Tabela 6.41, onde existe somente a troca de mensagens entre o mestre e os escravos, verifica-se que todos os processos que não pertencem ao *cluster* do mestre enviam somente mensagens externas.

Quanto à ociosidade dos processos, as Tabelas 6.39, 6.40 e 6.41 mostram que na estratégia *MeAdapt*, como o processo mestre acumula informações de todos os processos participantes e alcança um balanceamento de carga global mais rapidamente que as outras estratégias, a diferença entre o tempo de ociosidade dos processos é pequena. Esta afirmação é verdadeira quando não é considerado o processo mestre. Através da Tabela 6.41 pode-se observar que o processo mestre totaliza 204,06 segundos de CPU na solução da aplicação, sendo que 136,81 segundos deste tempo permaneceu ocioso (não resolvendo nós da árvore), ou seja, permanece ocioso 67,4% do total do seu tempo de CPU, enquanto

o escravo mais ocioso da aplicação permaneceu apenas 2,39%. É interessante ressaltar que para instâncias maiores a ociosidade dos escravos diminui enquanto a ociosidade do mestre aumenta. Por exemplo, em uma execução da maior instância, a *i640-214*, o mestre permaneceu 71,49% do tempo ocioso enquanto o escravo que ficou mais ocioso totalizou 0,11%. Na estratégia *HibridaAdapt*, a tendência é que o tempo de ociosidade aumente nos processos de maior identificação dentro dos *clusters*. Na estratégia *GlobalAdapt*, os processos obtiveram maior tempo de ociosidade em relação as outras estratégias e a ociosidade aumentou com a identificação dos processos. Este fato, tanto para a estratégia *GlobalAdapt* como para *HibridaAdapt*, se dá pela distribuição inicial de carga. Processos com menores identificações, na maioria das vezes recebem subárvores maiores do que os processos com maiores identificações.

Outro fato interessante a ser discutido é a diferença de ociosidade apresentada pelos *clusters*, ou seja, o quanto os processos de um *cluster* ficaram mais ociosos do que os processos do outro *cluster*. A estratégia *HibridaAdapt* intensifica essa diferença, pois, a transferência de carga entre *clusters* só acontece quando todos os processos do *cluster* se tornam ociosos. Na Tabela 6.40, verifica-se que o *cluster 0* permaneceu quase três vezes mais ocioso do que o *cluster 1*. Na estratégia *GlobalAdapt*, a diferença é muito pequena, sendo que o *cluster 1* se apresentou mais ocioso. A estratégia *MeAdapt*, devido ao processo mestre, a diferença é maior do que a apresentada pela estratégia *HibridaAdapt*, e, desconsiderando o processo mestre, a diferença de ociosidade apresentada pelos *clusters* é muito pequena.

GlobalAdapt - ambiente A_3																
	0	***1***	***2***	***3***	***4***	***5***	***6***	***7***	***8***	***9***	***10***	***11***	***12***	***13***	***14***	***15***
TCPU	440,46	417,26	413,08	522,84	402,77	417,56	534,43	554,86	413,67	361,39	345,48	364,72	391,06	246,79	376,60	509,09
TCPUN	440,46	417,26	413,08	365,99	426,94	417,56	571,84	554,86	413,67	180,70	345,48	364,72	191,62	246,79	376,60	361,45
QNOS	865,00	1010,00	775,00	780,00	967,00	1003,00	1315,00	1244,00	807,00	382,00	672,00	833,00	443,00	594,00	888,00	875,00
QNOSP	742,00	364,00	274,00	284,00	285,00	135,00	142,00	150,00	283,00	209,00	63,00	355,00	140,00	92,00	61,00	17,00
TOC	11,27	8,92	12,38	16,26	14,11	16,63	27,15	28,13	31,63	3,65	38,95	34,45	1,84	18,26	26,51	6,80
TOCN	11,27	8,92	12,38	11,38	14,96	16,63	29,05	28,13	31,63	1,83	38,95	34,45	0,90	18,26	26,51	4,83
MSGS IN	207,00	239,00	270,00	250,00	262,00	347,00	403,00	498,00	406,00	301,00	334,00	396,00	242,00	299,00	272,00	247,00
MSGS EXT	111,00	108,00	117,00	109,00	102,00	135,00	152,00	237,00	224,00	111,00	131,00	165,00	73,00	104,00	99,00	122,00

Tabela 6.39: Dados por processo quando utilizada a estratégia *GlobalAdapt* - ambiente A_3.

HibridaAdapt - ambiente A_3																
	0	***1***	***2***	***3***	***4***	***5***	***6***	***7***	***8***	***9***	***10***	***11***	***12***	***13***	***14***	***15***
TCPU	430,79	417,21	408,04	527,46	359,95	354,93	486,36	493,15	398,86	413,19	394,69	392,14	378,87	253,83	385,50	516,38
TCPUN	430,79	417,21	408,04	369,22	381,55	354,93	520,41	493,15	398,86	206,60	394,69	392,14	185,65	253,83	385,50	366,63
QNOS	930,00	928,00	909,00	678,00	698,00	627,00	1069,00	1069,00	1073,00	561,00	928,00	891,00	509,00	652,00	1009,00	988,00
QNOSP	438,00	334,00	329,00	142,00	171,00	63,00	174,00	99,00	1068,00	145,00	217,00	150,00	181,00	178,00	127,00	17,00
TOC	9,06	9,94	12,18	24,47	22,93	27,49	25,24	20,86	2,98	1,59	8,12	11,80	3,63	10,79	14,19	7,41
TOCN	9,06	9,94	12,18	17,13	24,31	27,49	27,01	20,86	2,98	0,80	8,12	11,80	1,78	10,79	14,19	5,26
MSGS IN	150,00	131,00	164,00	225,00	286,00	289,00	197,00	194,00	73,00	48,00	72,00	105,00	104,00	100,00	93,00	90,00
MSGS EXT	13,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	12,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00

Tabela 6.40: Dados por processo quando utilizada a estratégia *HibridaAdapt* - ambiente A_3.

MeAdapt - ambiente A_3																
	0	***1***	***2***	***3***	***4***	***5***	***6***	***7***	***8***	***9***	***10***	***11***	***12***	***13***	***14***	***15***
TCPU	204,06	398,70	398,60	513,70	394,35	395,86	553,17	528,17	399,41	372,44	426,40	413,42	388,66	271,45	427,90	507,18
TCPUN	204,06	398,70	398,60	359,59	418,01	395,86	591,89	528,17	399,41	186,22	426,40	413,42	190,44	271,45	427,90	360,10
QNOS	119,00	891,00	913,00	742,00	912,00	953,00	1288,00	1256,00	943,00	508,00	1009,00	1045,00	477,00	637,00	1078,00	824,00
QNOSP	119,00	467,00	459,00	282,00	338,00	512,00	279,00	766,00	212,00	150,00	129,00	47,00	139,00	53,00	69,00	111,00
TOC	136,81	2,77	3,19	5,21	4,62	4,15	10,64	12,68	10,70	1,89	10,46	10,12	0,66	5,78	10,39	5,72
TOCN	136,81	2,77	3,19	3,65	4,90	4,15	11,38	12,68	10,70	0,95	10,46	10,12	0,32	5,78	10,39	4,06
MSGS IN	380,00	33,00	46,00	26,00	36,00	39,00	44,00	47,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
MSGS EXT	248,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	40,00	42,00	48,00	36,00	22,00	41,00	45,00	30,00

Tabela 6.41: Dados por processo quando utilizada a estratégia *MeAdapt* - ambiente A_3.

A ociosidade apresentada pelos processos é melhor datalhada nas Tabelas 6.42 e 6.43, 6.44 e 6.45 e 6.46 e 6.47, onde são apresentados o número de nós recebidos (coluna *Nos*) e o tempo gasto para resolvê-los em segundos (coluna *T(s)*) por cada processo em resposta aos trinta primeiros *LOADREQUEST* enviados durante a execução da instância *i640-212* no ambiente A_3. Por exemplo, a primeira linha da segunda coluna corresponde a quantidade de nós recebidos pelo processo 0 em resposta a primeira mensagem *LOADREQUEST* enviada pelo mesmo. Na estratégia *GlobalAdapt*, as Tabela 6.42 e 6.43, os processos com menor identificação enviam menos requisições, pois a quantidade de carga transferida entre estes processos é maior e, portanto, tornam-se menos ociosos. É interessante notar que este fato não é conseqüência somente das maiores subárvores estarem concentradas entre os processos de menor identificação, mas também da seqüência dos envios de pedidos de carga dos processos. Um processo ao se tornar ocioso, envia pedido de carga ao processo de identificação imediatamente menor, e somente se não receber carga deste processo, é que envia pedido de carga ao próximo. Ao se tornar ocioso novamente o ciclo é reiniciado. Portanto, a chance do processo, por exemplo o 15, enviar pedido de carga ao processo 0 é muito pequena. Uma maneira de evitar esta concentração de carga é não reiniciar o ciclo de requisições de carga, ou seja, um processo ao se tornar ocioso novamente, enviaria *LOADREQUEST* ao processo de identificação imediatamente menor que a do último processo para o qual enviou pedido de carga. Na estratégia *HibridaAdapt*, Tabelas 6.44 e 6.45, percebe-se o mesmo fato, entretanto, a carga pendente concentra-se nos processos de menores identificações pertencentes a cada *cluster*. Finalmente nas Tabelas 6.46 e 6.47, a estratégia *MeAdapt* se mostrou eficiente, já que o maior número de requisições de carga enviados são dos processos 7 e 10 que totaliza apenas 15 requisições. Comprovadamente, para esta instância, o mestre alcança um melhor balancemanto global de carga, uma vez que a quantidade de *LOADREQUEST* enviada é menor que a das outras estratégias. Note que o processo 0 não faz nenhuma requisição de carga, pois só executa a primeira subárvore da árvore B&B que é referente à que lhe é designada no procedimento de *Distribuição Inicial*.

É importante notar que a eficiência da estratégia *MeAdapt*, entretanto, pode ser comprometida com o aumento do número de processadores. Além disso, para instâncias maiores, a utilização de um processo, o mestre, dedicado ao procedimento de balanceamento de carga, pode acarretar perda na eficiência do algoritmo.

GlobalAdapt - ambiente A_3 - processos de 0 a 7																
	0		***1***		***2***		***3***		***4***		***5***		***6***		***7***	
	Nos	***T(s)***	***Nos***	***T(s)***	***Nos***	***T(s)***	***Nos***	***T(s)***	***Nos***	***T(s)***	***Nos***	***T(s)***	***Nos***	***T(s)***	***Nos***	***T(s)***
1	742	377,49	364	147,23	274	142,1	284	181,12	285	137,96	135	61,1	142	63,21	150	68,02
2	15	3,16	451	184	1	1,15	1	0,93	5	2,24	163	68,2	25	6,75	71	24,53
3	25	13,47	47	24,37	189	109,57	117	66,95	1	1,48	10	4,33	218	90,84	147	64,66
4	28	12,62	65	18,49	108	52,18	132	118,07	383	128,65	9	6,24	9	1,8	15	6,33
5	13	5,54	5	3,1	138	67,41	26	20,3	52	24,47	1	0,38	7	4,28	3	1,31
6	25	9,68	1	1,36	1	1,18	5	4,18	12	7,07	1	0,42	25	10,86	1	0,5
7	13	6,46	40	16,43	1	1,02	67	41,67	49	19,65	1	3,28	1	0,36	9	2,24
8	1	1,09	30	10,85	4	2,35	111	55,75	36	12,66	35	18,74	108	51,6	1	0,96
9	1	1,54	7	3,93	23	9	5	6,79	27	12,07	281	104,13	244	76,91	2	0,73
10	1	5,38			17	7,59	9	6,58	63	28,9	63	18,43	105	41,35	13	6,01
11	1	1,05			8	4,36	15	5,75	1	0,59	21	10,14	72	25,93	11	6,53
12					2	1,18	8	3,25	1	0,5	8	5,91	53	15,95	129	47,65
13					1	1,14			1	0,35	35	12,95	8	2,76	9	6,14
14					1	1,01			25	5,65	14	6,42	3	1,73	108	45,16
15					5	4,08			5	3,24	18	8,1	3	1,5	60	28,11
16					2	1,03			10	2,1	25	10,33	1	0,58	57	25,82
17									2	2,27	111	35,56	6	3,12	49	25,06
18									1	1,59	23	8,32	44	14,08	9	3,14
19									8	7,58	20	8,38	7	2,63	17	8,36
20											15	5,18	27	8,3	20	9,47
21											1	0,44	12	5,26	1	2,06
22											1	0,99	27	10,79	40	14,52
23											1	0,73	4	1,6	16	7,01
24											2	2,56	24	10,43	2	1,29
25											2	2,32	74	32,85	2	1,32
26											7	8,1	13	4,35	1	0,58
27													6	4,2	2	1,37
28													3	1,48	1	0,75
29													1	0,32	1	0,85
30													12	4,46	1	0,89

Tabela 6.42: Primeiros 30 *BRANCHS* recebidos por processo - *GlobalAdapt* - processos de 0 a 7.

GlobalAdapt - ambiente A_3 - processos de 8 a 15																
	8		***9***		***10***		***11***		***12***		***13***		***14***		***15***	
	Nos	***T(s)***	***Nos***	***T(s)***	***Nos***	***T(s)***	***Nos***	***T(s)***	***Nos***	***T(s)***	***Nos***	***T(s)***	***Nos***	***T(s)***	***Nos***	***T(s)***
1	283	166,03	209	160,61	63	44,29	355	135,79	140	113,08	92	38,43	61	24,18	17	17,7
2	5	1,72	2	2,94	284	113,16	77	27,37	24	17,08	75	31,85	35	12,81	24	13,33
3	13	5,64	35	46,68	3	1,11	115	45,14	5	3,86	16	9,45	30	10,51	1	0,67
4	1	0,92	5	5,85	75	43,58	13	7,66	19	12,82	27	5,54	23	12,26	15	11,16
5	109	51,1	5	3,67	13	7,74	2	1,59	18	11,96	1	0,44	56	25,82	24	11,36
6	2	1,94	2	3,58	1	0,74	1	0,74	3	2,03	1	0,53	42	18,39	1	1
7	1	0,72	8	5,34	1	0,59	3	2,55	7	6,7	1	0,33	38	13,77	7	5,42
8	43	9,75	3	2,18	1	0,3	5	1,45	9	7,35	1	0,79	2	1,11	55	36,61
9	14	5,4	3	3,23	1	0,48	22	8,26	35	23,47	56	14,48	30	9,03	53	35,73
10	14	9,57	2	1,88	3	1,6	1	0,37	3	4,09	1	0,44	1	0,32	14	9,73
11	21	7,55	1	2,2	7	4,33	1	1,79	5	4,52	9	3,09	2	1,66	7	4,83
12	4	3,05	2	4,34	4	2,16	11	4,87	8	9,3	1	0,69	4	1,83	11	6,54
13	3	1,69	6	8,29	1	1,04	1	1,24	1	2,25	1	0,38	46	13,93	3	2,29
14	1	0,79	1	0,56	19	3,7	1	1,08	1	2,62	2	1,35	36	11,3	1	0,71
15	2	1,81	1	0,99	1	0,32	6	2,81	2	3,45	4	2,97	4	1,57	1	0,6
16	5	3,03	3	1,78	3	2,16	1	1,23	3	2,56	8	3,11	5	2,28	1	1,05
17	5	1,61	1	0,78	3	1,71	9	3,68	1	1,39	23	8,28	5	2,04	13	9,37
18	15	6,91	5	3,44	5	1,6	2	2,01	1	1,68	9	3,66	11	2,42	1	0,82
19	30	10,58	1	0,96	2	2,76	3	0,92	1	2,39	2	0,6	1	1,1	1	0,53
20	3	1,38	1	0,94	1	2,2	2	1,23	8	6,07	2	2,27	1	0,77	1	0,46
21	9	3,18	3	2,65	1	0,73	1	0,72	3	1,89	5	1,35	10	2,78	1	0,61
22	12	3,63	1	1,7	1	0,31	1	0,92	1	2,22	7	3,24	11	4,52	1	0,61
23	1	1,17	10	9,19	1	1,2	1	1,3	1	2,09	1	0,82	19	9,05	29	14,92
24	1	0,54	3	4,52	3	2,16	5	1,91	8	7,57	1	1,17	1	1,99	6	1,95
25	7	3,99	1	0,5	16	11,22	11	2,63	1	1,48	1	1,21	3	1,44	11	7,08
26	17	7,17	12	13,77	1	1,55	2	0,72	1	1,08	1	0,9	6	2,64	23	11,32
27	5	2,41	1	3,27	15	3,64	1	1,23	7	3,05	5	1,85	1	0,64	1	0,44
28	16	9,3	1	1,07	4	1,47	10	4,25	1	0,68	4	0,98	1	0,56	77	45,94
29	5	1,64	3	2,68	1	0,65	3	2,75	5	3,35	1	0,6	1	2,23	1	0,54
30	15	7,05	5	5,55	1	0,89	1	1,53	1	1,15	9	2,43	6	3,22	1	1,01

Tabela 6.43: Primeiros 30 *BRANCHS* recebidos por processo - *GlobalAdapt* - processos de 8 a 15.

HibridaAdapt - ambiente A_3 - processos de 0 a 7																
	0		***1***		***2***		***3***		***4***		***5***		***6***		***7***	
	Nos	***T(s)***	***Nos***	***T(s)***	***Nos***	***T(s)***	***Nos***	***T(s)***	***Nos***	***T(s)***	***Nos***	***T(s)***	***Nos***	***T(s)***	***Nos***	***T(s)***
1	438	227,36	334	168,75	329	150,84	142	104,13	171	103,57	63	44,19	174	68,66	99	40,60
2	29	13,20	393	156,97	25	13,00	52	40,81	8	5,21	71	40,74	59	43,31	373	148,56
3	26	9,04	91	35,64	6	1,75	6	5,55	4	3,51	34	15,24	43	25,79	2	2,38
4	55	25,37	3	1,15	95	36,54	6	3,51	1	0,97	1	0,42	13	4,47	1	0,77
5	5	2,24	4	3,55	213	90,56	17	17,48	3	1,46	2	2,27	1	0,81	1	1,37
6	32	18,87	10	5,89	68	24,00	10	7,34	1	0,87	2	1,28	1	1,00	7	4,71
7	3	2,00	4	1,88	13	6,49	6	9,79	1	1,15	1	1,64	1	0,85	3	1,03
8	5	2,84	1	0,79	53	26,78	17	15,55	17	6,81	1	1,04	54	22,34	1	1,35
9	34	14,16	1	0,82	20	9,41	1	2,42	3	2,55	9	6,18	1	0,91	1	1,26
10	19	3,96	33	13,08	3	1,55	1	0,84	1	0,61	2	2,02	7	3,78	1	0,99
11	1	0,96	1	1,07	5	3,64	14	13,01	3	1,47	1	0,72	1	0,92	5	2,76
12	1	0,54	1	0,43	1	1,31	47	38,05	17	8,33	1	0,99	1	1,49	1	0,71
13	17	6,54	1	0,51	1	1,34	79	49,77	1	1,21	1	0,52	1	0,80	1	1,22
14	2	2,29	1	0,41	1	1,24	9	7,60	9	3,23	1	0,61	2	0,71	1	0,65
15	1	1,08	2	2,39	11	5,92	55	46,85	9	7,05	1	3,13	8	6,38	82	43,03
16	141	48,48	23	10,79	7	2,91	14	11,98	1	1,95	1	1,20	5	3,17	43	29,19
17	32	9,94	1	1,03	3	2,55	45	37,35	12	8,15	1	1,02	3	2,68	4	2,62
18	40	16,34	17	6,49	1	2,28	33	18,77	1	0,81	1	0,62	11	3,35	1	1,11
19	1	0,63	4	1,90	1	0,22	4	4,44	36	19,40	2	0,51	6	2,14	1	0,71
20	1	0,21	1	0,90	6	3,07	8	6,89	65	42,29	41	17,59	2	2,05	3	2,11
21	6	2,29	2	0,87	30	15,14	18	8,54	23	9,17	123	65,52	1	0,68	1	0,59
22	1	1,13			10	1,77	1	1,24	6	5,54	9	7,83	69	33,96	10	7,19
23	5	3,73			3	1,39	3	3,22	92	32,44	1	1,32	65	36,58	5	2,24
24	10	3,48			1	0,95	1	1,74	6	3,10	1	0,74	18	7,35	1	0,46
25	4	1,17			1	0,66	2	3,07	2	1,75	8	5,55	21	10,99	2	2,93
26	2	1,02			2	0,92	12	9,69	2	1,28	3	2,44	1	1,03	1	1,09
27	1	0,38					1	1,06	1	0,44	1	0,98	1	0,68	1	0,67
28	8	4,40					2	1,53	86	24,35	2	1,47	3	3,41	1	1,82
29	2	1,47					1	1,61	1	0,56	1	1,05	7	3,14	11	2,44
30	4	1,51					1	1,93	1	0,56	6	5,22	2	1,79	24	11,32

Tabela 6.44: *BRANCHS* recebidos por processo - *HibridaAdapt - processos de 0 a 7.*

HibridaAdapt - ambiente A_3 - processos de 8 a 15																
	8		***9***		***10***		***11***		***12***		***13***		***14***		***15***	
	Nos	***T(s)***	***Nos***	***T(s)***	***Nos***	***T(s)***	***Nos***	***T(s)***	***Nos***	***T(s)***	***Nos***	***T(s)***	***Nos***	***T(s)***	***Nos***	***T(s)***
1	1068	393,90	145	112,60	217	104,68	150	69,00	181	126,43	178	60,45	127	57,78	17	18,42
2	1	0,75	156	90,23	218	93,41	56	26,70	18	18,27	33	17,35	22	12,50	29	19,90
3	2	1,99	91	56,93	132	55,98	195	90,16	30	21,82	222	77,82	22	13,80	119	56,11
4	1	1,23	167	150,38	252	84,67	28	13,61	20	15,46	57	24,38	67	19,12	22	15,53
5	1	0,78	1	0,74	21	7,66	269	103,20	15	9,11	112	42,15	181	69,80	11	5,24
6			1	1,42	50	24,40	61	23,13	1	0,82	9	3,96	251	91,22	1	1,40
7					8	5,24	28	8,76	17	11,54	2	1,43	116	26,16	21	9,97
8					29	14,39	1	1,70	47	41,20	2	1,11	106	36,09	7	3,70
9					1	0,37	5	3,00	131	79,12	1	1,17	1	1,04	80	39,75
10							9	4,21	1	1,83	3	1,75	23	9,36	455	227,90
11							16	8,68	7	6,54	1	0,81	16	11,35	57	25,76
12							22	7,99	1	1,15	1	0,47	1	0,67	39	15,17
13							2	2,17	1	0,93	1	0,43	2	1,39	15	7,65
14							5	3,39	1	3,35	1	1,03	59	25,82	31	11,88
15							1	1,71	2	3,79	1	0,66	4	1,83	1	0,95
16							8	4,93	6	2,39	7	5,52	11	3,61	1	0,66
17							1	1,09	3	3,18	1	1,25			4	5,03
18							5	1,87	1	2,92	1	0,79			1	0,72
19							5	2,50	1	1,31	5	1,90			2	1,67
20							2	1,42	1	2,90	3	1,69			7	4,98
21							16	5,99	4	3,73	1	1,57			4	3,05
22							6	2,60	5	6,68	7	2,32			9	4,35
23									14	11,68	3	1,90			43	22,30
24									1	1,48					7	5,96
25															1	0,53
26															2	3,11
27															1	1,13
28															1	1,80
29																
30																

Tabela 6.45: Primeiros 30 *BRANCHS* recebidos por processo - *HibridaAdapt* - processos de 8 a 15.

MeAdapt - ambiente A_3 - processos de 0 a 7																
	0		***1***		***2***		***3***		***4***		***5***		***6***		***7***	
	Nos	***T(s)***	***Nos***	***T(s)***	***Nos***	***T(s)***	***Nos***	***T(s)***	***Nos***	***T(s)***	***Nos***	***T(s)***	***Nos***	***T(s)***	***Nos***	***T(s)***
1			467	192,52	459	210,44	282	237,90	338	178,25	512	216,48	279	126,62	766	308,78
2			324	164,98	106	48,72	65	23,97	141	58,57	84	33,68	349	170,95	148	71,57
3			63	23,79	78	37,56	387	243,01	60	17,06	39	18,94	222	94,84	119	43,99
4			9	4,20	41	12,32	5	2,13	270	93,60	88	34,47	78	25,64	12	8,93
5			19	6,46	112	38,21	2	2,27	63	27,50	10	5,62	183	62,84	96	34,79
6			1	0,66	35	13,41	1	2,67	9	3,56	73	30,56	105	34,85	43	18,73
7			4	3,09	42	15,15			5	1,70	74	23,29	3	1,42	27	13,82
8			4	2,43	27	12,42			21	8,82	59	22,81	19	5,39	20	4,80
9					10	5,52			4	1,75	1	0,60	1	1,63	7	3,15
10					1	2,30			1	1,11	7	3,56	3	1,98	1	0,60
11					1	1,10					6	4,04	6	2,83	1	0,61
12					1	0,95							24	9,48	5	2,97
13													16	7,95	3	2,14
14															1	0,74
15															7	3,92
16																
17																
18																
19																
20																
21																
22																
23																
24																
25																
26																
27																
28																
29																
30																

Tabela 6.46: *BRANCHS* recebidos por processo - *MeAdapt - processos de 0 a 7.*

MeAdapt - ambiente A_3 - processos de 8 a 15																
	8		***9***		***10***		***11***		***12***		***13***		***14***		***15***	
	Nos	***T(s)***	***Nos***	***T(s)***	***Nos***	***T(s)***	***Nos***	***T(s)***	***Nos***	***T(s)***	***Nos***	***T(s)***	***Nos***	***T(s)***	***Nos***	***T(s)***
1	212	108,16	150	104,33	129	73,32	47	28,19	139	145,68	53	28,74	69	33,89	111	65,9
2	307	133,95	10	14,66	255	100,26	61	35,23	72	38,64	47	22,80	12	4,08	161	109,18
3	197	58,22	74	57,98	144	50,27	537	186,77	266	204,01	198	73,48	22	5,42	114	79,22
4	167	60,91	52	28,99	97	49,53	137	55,04			91	42,04	384	150,33	339	193,88
5	25	12,70	5	4,17	71	23,59	95	30,28			91	39,06	47	22,82	58	31,51
6	5	3,65	32	40,23	91	33,41	46	13,69			44	24,97	200	70,01	8	3,79
7	24	10,60	28	16,87	8	5,35	56	26,73			1	0,55	250	93,66	32	19,56
8	1	0,47	80	48,94	65	29,74	3	2,46			8	3,92	24	9,35	1	0,58
9	1	0,62	60	41,62	83	27,23	9	4,60			78	21,23	1	0,39		
10	1	1,84	9	4,94	8	4,89	54	23,93			1	2,17	5	2,80		
11	1	1,14	4	2,70	13	5,25					6	4,36	32	14,51		
12	1	0,84	3	4,47	9	4,18					19	5,28	7	2,99		
13	1	0,71	1	1,38	3	1,64							25	11,41		
14					24	7,79										
15					9	3,29										
16																
17																
18																
19																
20																
21																
22																
23																
24																
25																
26																
27																
28																
29																
30																

Tabela 6.47: Primeiros 30 *BRANCHS* recebidos por processo - *MeAdapt* - processos de 8 a 15.

6.4.1 Ociosidade provacada pelos procedimentos de *Distribuição Inicial* e *Detecção de Terminação*

Para ilustrar o *overhead* de comunicação causado pelos procedimentos de *Distribuição Inicial* e *Detecção de Terminação*, considere as Tabelas 6.48, 6.49 e 6.50. Essas tabelas apresentam o tempo de CPU da aplicação ($TCPU$) o tempo de ociosidade (TOC) e os tempos TOI e TOF, que correspondem, respectivamente, ao tempo decorrido desde que o processo foi iniciado até receber carga pela primeira vez e o tempo decorrido pelo processo entre o fim da execução de sua última subárvore e o fim da execução da aplicação. As tabelas contêm as médias obtidas de cinco execuções da instância *i640-212* no ambiente A_1, quando utilizada a estratégia *GlobalAdapt*, *HibriaaAdapt* e *MeAdapt*, respectivamente.

As estratégias *GlobalAdapt* e *HibridaAdapt*, que utilizam o procedimento de distribuição inicial descrito na Figura 4.1, evitam a ociosidade inicial dos processos. Além disso, a distribuição se mostra mais uniforme, pois é realizada simultaneamente pelos processos à medida que estes recebem carga. Desconsiderando o processo 0, que inicia a distribuição de carga, o processo 1 foi o que menos esperou inicialmente por carga, 0,33 segundos, enquanto, o processo 14, totalizaou 1,19 segundos de espera na estratégia *GlobalAdapt*. Na estratégia *HibridaAdapt*, pode-se perceber o mesmo, entretanto, os valores são um pouco mais elevados e a diferença de tempo para receber a carga inicial entre os processos 15 e 14 foi de apenas 0,03 segundos. Já na estratégia *MeAdapt*, onde a distribuição inicial é realizada somente pelo líder, o tempo de espera é quase crescente em relação a identificação dos processos. Vale notar que estes tempos são influenciados pelo tempo gasto pelo processo para gerar nova carga e pela latência de rede. Foi possível mostrar que na *MeAdapt* grande parte do tempo total de ociosidade dos processos se deve à espera da primeira carga a ser resolvida, ou seja, no procedimento de *Distribuição Inicial*. É válido lembrar que o valor elevado de TOC para esta estratégia deve-se a ociosidade do processo mestre, já discutida anteriormente. Por outro lado, as estratégias *GlobalAdapt* e *HibridaAdapt*, utilizam, em média, cerca de, respectivamente, 4,4% e 6,4% do tempo total de ociosidade no procedimento de *Distribuição Inicial*.

O procedimento *Detecção de Terminação* é mais custoso para a estratégia *HibridaAdapt*, quando desconsiderado o processo mestre da estratégia *MeAdapt*, já que há a necessidade de que todos os processos de um *cluster* se tornem ociosos para que uma suspeita de término da aplicação seja iniciada. Na estratégia *GlobalAdapt*, diferente da *HibridaAdapt*, qualquer processo pode iniciar a suspeita de término, fazendo com que este procedimento seja realizado mais rapidamente. Já para a estratégia *MeAdapt* é ainda

menos custosa levando em consideração somente os processos escravos, uma vez que o mestre possui informações sobre o estado de todos os processos.

GlobalAdapt - ambiente A_1																	
	0	***1***	***2***	***3***	***4***	***5***	***6***	***7***	***8***	***9***	***10***	***11***	***12***	***13***	***14***	***15***	***Total***
TCPU	404,90	470,11	441,98	447,69	437,61	456,59	454,60	476,18	412,14	424,15	414,94	422,45	418,68	601,84	597,69	634,77	7516,31
TOC	8,44	11,00	9,81	11,41	12,68	10,51	13,23	15,06	21,24	18,08	19,44	18,80	19,71	25,37	22,42	20,40	257,59
TOI	0,00	0,33	0,63	0,47	0,94	0,52	0,81	0,56	1,15	0,59	0,87	0,63	1,06	0,82	1,19	0,82	11,4
TOF	1,53	0,82	1,34	4,77	2,08	1,46	0,91	1,42	3,04	1,19	1,32	1,60	1,51	0,92	1,72	1,44	27,09

Tabela 6.48: Dados por processo quando utilizada a estratégia *GlobalAdapt* - ambiente A_1.

HibridaAdapt - ambiente A_1																	
	0	***1***	***2***	***3***	***4***	***5***	***6***	***7***	***8***	***9***	***10***	***11***	***12***	***13***	***14***	***15***	***Total***
TCPU	348,88	348,88	395,60	393,90	377,12	375,98	366,75	365,83	337,27	330,70	322,54	315,92	314,41	462,03	465,54	462,94	5984,31
TOC	10,83	10,83	7,92	7,99	14,02	14,31	17,31	12,26	9,77	24,25	27,69	31,98	33,98	38,30	35,80	34,31	331,56
TOI	0,00	0,72	1,23	0,85	1,51	1,08	1,36	1,02	0,85	0,80	1,06	0,94	1,33	1,36	1,53	1,56	17,02
TOF	0,83	0,83	0,97	1,32	1,07	1,43	2,61	1,77	0,18	5,04	8,74	5,35	12,26	9,20	11,31	6,22	69,16

Tabela 6.49: Dados por processo quando utilizada a estratégia *HibridaAdapt* - ambiente A_1.

MeAdapt - ambiente A_1																	
	0	***1***	***2***	***3***	***4***	***5***	***6***	***7***	***8***	***9***	***10***	***11***	***12***	***13***	***14***	***15***	***Total***
TCPU	115,87	270,76	271,05	269,68	268,02	268,87	264,98	255,79	287,75	294,75	297,53	295,90	813,80	807,50	806,58	293,14	5881,96
TOC	2720,04	1,30	1,88	2,17	2,17	2,52	2,53	3,24	3,61	4,63	4,84	4,61	14,56	19,14	18,61	5,28	2811,15
TOI	0,00	0,77	1,19	1,32	1,54	1,75	1,92	2,08	2,27	2,47	2,48	2,42	8,82	10,20	10,06	2,96	52,25
TOF	600,33	0,02	0,10	0,14	0,16	0,01	0,07	0,32	0,44	0,44	0,58	0,57	1,70	1,98	1,67	1,08	609,63

Tabela 6.50: Dados por processo quando utilizada a estratégia *MeAdapt* - ambiente A_1.

O tempo de ociosidade dos processos entre os trinta primeiros envios de uma requisição de carga e os seus repectivos recebimentos são apresentados nas Tabelas 6.51 e 6.52 para a estratégia *GlobalAdapt*, nas Tabelas 6.53 e 6.54 para a estratégia *HibridaAdapt* e nas Tabelas 6.55 e 6.56 para a *MeAdapt*. A coluna NS representa o número de nós recebidos pelos processos mediante a um envio de requisição de carga. A coluna $T(s)$ representa o tempo de ociosidade entre um envio de pedido e o seu recebimento e a coluna % indica o percentual do tempo de ociosidade do processo em relação ao tempo total que o mesmo ficou ocioso durante a aplicação. Observe que para o processo 0, que é mestre, nenhum valor foi mostrado já que o mesmo executa o procedimento *busca por tarefas* e esta, apesar de incluir requisição e recebimento de carga, não incluem solução de nós da árvore B&B. Pode-se perceber que na estratégia *MeAdapt* os processos enviaram menos pedidos de carga e o tempo de ociosidade entre o envio e o recebimento é normalmente menor do que para as outras estratégias. Mas vale lembrar que a quantidade de processadores utilizados nos testes foi relativamente baixa o que não resultou em um gargalo no mestre. Na estratégia *GlobalAdapt* o aumento no tempo de ociosidade entre o envio do pedido e a resposta se dá quando o processo só consegue obter carga depois de vários pedidos, ou seja, só existe carga disponível com um processo de identificação mais distante. Na estratégia *HibridaAdapt* este fato é conseqüência do envio de pedido de carga ao outro *cluster*.

GlobalAdapt - processos de 0 a 7 -ambiente A_1																								
	0			*1*			*2*			*3*			*4*			*5*			*6*			*7*		
	NS	***T(s)***	***%***	***NS***	***T(s)***	***%***	***NS***	***T(s)***	***%***	***NS***	***T(s)***	***%***	***NS***	***T(s)***	***%***	***NS***	***T(s)***	***%***	***NS***	***T(s)***	***%***	***NS***	***T(s)***	***%***
1	312	0,20	2,83	413	0,31	4,17	329	0,60	5,63	322	0,41	4,57	241	0,77	8,16	162	0,45	7,61	142	0,65	7,74	150	0,54	4,82
2	3	0,30	4,25	1	0,50	6,72	1	0,41	3,85	35	0,69	7,69	28	0,44	4,66	71	0,60	10,15	22	0,34	4,05	11	0,65	5,80
3	116	0,10	1,42	1	0,44	5,91	1	0,27	2,54	9	0,68	7,58	47	0,02	0,21	13	0,05	0,85	100	0,13	1,55	6	0,14	1,25
4	60	0,82	11,61	1	0,07	0,94	20	1,37	12,86	1	1,11	12,37	15	0,24	2,54	90	0,01	0,17	5	0,09	1,07	84	0,12	1,07
5	93	0,19	2,69	1	0,02	0,27	25	0,20	1,88	70	0,55	6,13	1	0,30	3,18	5	0,04	0,68	1	0,11	1,31	11	0,50	4,46
6	3	0,70	9,92	52	0,08	1,08	6	0,11	1,03	42	0,12	1,34	1	1,01	10,70	17	0,03	0,51	35	0,32	3,81	49	0,06	0,54
7	37	0,40	5,67	82	0,01	0,13	69	0,06	0,56	41	0,13	1,45	5	0,25	2,65	10	0,65	11,00	47	0,01	0,12	5	0,28	2,50
8	10	0,98	13,88	29	0,03	0,40	61	0,02	0,19	202	0,00	0,00	51	0,91	9,64	1	0,52	8,80	2	0,23	2,74	2	0,07	0,63
9	1	0,19	2,69	63	0,08	1,08	1	0,19	1,78	3	0,07	0,78	15	0,15	1,59	19	0,53	8,97	5	0,32	3,81	25	0,29	2,59
10	1	0,08	1,13	25	0,06	0,81	1	0,00	0,00	1	0,09	1,00	6	0,05	0,53	74	0,04	0,68	28	0,14	1,67	1	0,66	5,89
11	1	0,74	10,48	109	1,13	15,19	1	0,06	0,56	1	0,12	1,34	5	0,01	0,11	9	0,11	1,86	1	1,29	15,36	1	0,56	5,00
12	3	2,36	33,43	2	0,08	1,08	1	0,00	0,00	16	0,05	0,56	13	0,09	0,95	4	0,13	2,20	5	0,22	2,62	40	0,30	2,68
13				21	1,37	18,41	70	0,85	7,98	1	0,11	1,23	13	0,00	0,00	19	0,31	5,25	12	0,01	0,12	1	0,24	2,14
14				1	0,01	0,13	84	0,07	0,66	8	0,10	1,11	26	0,20	2,12	6	0,03	0,51	15	0,02	0,24	13	0,13	1,16
15				2	0,03	0,40	1	0,19	1,78	3	0,04	0,45	7	0,05	0,53	7	0,50	8,46	15	0,01	0,12	4	0,01	0,09
16				2	0,45	6,05	5	0,18	1,69	1	0,26	2,90	9	0,09	0,95	29	0,05	0,85	2	0,05	0,60	1	0,02	0,18
17				4	0,00	0,00	16	0,67	6,29	1	0,02	0,22	216	0,09	0,95	1	0,01	0,17	1	0,03	0,36	1	0,14	1,25
18				6	0,04	0,54	8	0,07	0,66	3	0,14	1,56	23	0,17	1,80	1	0,00	0,00	7	0,78	9,29	9	0,06	0,54
19				1	1,05	14,11	1	0,11	1,03	2	0,13	1,45	12	0,03	0,32	163	0,36	6,09	7	0,08	0,95	1	0,00	0,00
20				1	0,10	1,34	1	0,09	0,85	1	1,81	20,18	2	0,01	0,11	141	0,62	10,49	11	0,05	0,60	1	0,20	1,79
21				7	1,10	14,78	1	0,39	3,66				5	0,18	1,91	1	0,06	1,02	3	0,00	0,00	19	0,49	4,38
22							7	0,17	1,60				7	0,01	0,11				6	0,08	0,95	1	0,06	0,54
23							8	0,27	2,54				1	0,08	0,85				1	0,08	0,95	2	0,09	0,80
24							3	2,02	18,97				11	0,02	0,21				1	0,17	2,02	1	0,55	4,91
25													1	0,07	0,74				4	0,45	5,36	1	0,04	0,36
26																			18	0,68	8,10	1	0,24	2,14
27																			34	0,01	0,12	1	0,05	0,45
28																			122	0,03	0,36	3	0,90	8,04
29																			152	0,06	0,71	1	0,09	0,80
30																			1	1,22	14,52	11	0,52	4,64

Tabela 6.51: Ociosidade dos processos para os primeiros 30 envios de carga - processos de 0 a 7.

GlobalAdapt - processo de 8 a 15 - ambiente A_1																								
Num	***8***			***9***			***10***			***11***			***12***			***13***			***14***			***15***		
	Nos	***T(s)***	***%***	***Nos***	***T(s)***	***%***	***Nos***	***T(s)***	***%***	***Nos***	***T(s)***	***%***	***Nos***	***T(s)***	***%***	***Nos***	***T(s)***	***%***	***Nos***	***T(s)***	***%***	***Nos***	***T(s)***	***%***
1	177	0,88	8,18	234	0,54	7,47	63	0,71	6,90	183	0,59	6,36	97	0,82	7,72	103	0,71	4,10	59	0,86	4,57	17	0,74	4,46
2	7	0,31	2,88	100	1,19	16,46	81	0,09	0,87	15	0,19	2,05	71	0,11	1,04	134	0,91	5,25	10	0,22	1,17	34	0,02	0,12
3	108	0,01	0,09	1	0,66	9,13	139	0,31	3,01	70	0,18	1,94	27	0,01	0,09	32	0,43	2,48	44	0,12	0,64	1	0,01	0,06
4	1	0,94	8,74	2	0,11	1,52	13	0,94	9,14	50	0,00	0,00	7	0,03	0,28	2	0,29	1,67	4	0,17	0,90	5	0,05	0,30
5	20	1,70	15,80	2	0,62	8,58	14	0,22	2,14	38	0,04	0,43	27	0,26	2,45	5	0,00	0,00	66	0,60	3,19	9	0,17	1,02
6	31	0,28	2,60	9	0,31	4,29	4	0,29	2,82	12	0,14	1,51	8	0,02	0,19	7	0,05	0,29	79	0,02	0,11	11	0,27	1,63
7	1	0,47	4,37	397	0,17	2,35	1	1,16	11,27	32	0,14	1,51	46	0,26	2,45	23	0,15	0,87	8	0,00	0,00	30	0,05	0,30
8	11	0,59	5,48	35	0,04	0,55	3	0,01	0,10	1	0,73	7,87	21	0,04	0,38	45	0,23	1,33	18	0,08	0,43	133	1,46	8,80
9	4	0,17	1,58	3	0,12	1,66	10	0,27	2,62	5	1,95	21,04	27	0,00	0,00	20	0,25	1,44	7	0,48	2,55	13	0,06	0,36
10	1	0,05	0,46	1	0,01	0,14	1	0,07	0,68	3	0,19	2,05	4	0,03	0,28	4	0,06	0,35	22	0,01	0,05	1	0,09	0,54
11	4	0,05	0,46	45	0,23	3,18	1	0,02	0,19	338	0,58	6,26	21	0,04	0,38	1	0,02	0,12	32	0,02	0,11	1	0,04	0,24
12	7	0,86	7,99	1	0,13	1,80	1	0,01	0,10	96	0,02	0,22	3	0,02	0,19	13	0,26	1,50	1	0,16	0,85	3	0,03	0,18
13	1	0,96	8,92	10	0,06	0,83	5	0,90	8,75	21	0,24	2,59	2	0,06	0,56	29	0,15	0,87	2	0,02	0,11	5	0,12	0,72
14	105	0,19	1,77	1	0,70	9,68	316	0,14	1,36	25	0,03	0,32	5	2,17	20,43	8	0,07	0,40	1	0,15	0,80	104	0,04	0,24
15	224	0,18	1,67				74	0,03	0,29	5	0,16	1,73	1	0,36	3,39	3	0,08	0,46	19	0,79	4,20	11	0,98	5,91
16	57	0,12	1,12				1	0,06	0,58				1	0,00	0,00	5	0,43	2,48	9	0,01	0,05	14	0,01	0,06
17	16	0,10	0,93				1	0,28	2,72				3	0,14	1,32	3	0,02	0,12	12	0,03	0,16	1	0,04	0,24
18							1	0,20	1,94				1	0,06	0,56	11	0,12	0,69	17	0,15	0,80	7	0,10	0,60
19							1	0,08	0,78				22	0,75	7,06	2	0,05	0,29	1	0,08	0,43	44	0,01	0,06
20							2	0,02	0,19				76	0,08	0,75	1	0,52	3,00	5	0,30	1,60	7	0,20	1,21
21							44	0,21	2,04				195	0,04	0,38	5	0,07	0,40	7	0,05	0,27	1	0,36	2,17
22							1	0,07	0,68				35	0,34	3,20	33	0,02	0,12	1	0,07	0,37	18	0,02	0,12
23							3	0,37	3,60				45	0,31	2,92	8	0,08	0,46	11	0,42	2,23	51	0,90	5,42
24							1	0,23	2,24				33	0,05	0,47	1	0,20	1,15	1	0,27	1,44	1	0,02	0,12
25							2	0,58	5,64				47	0,08	0,75	9	0,14	0,81	1	0,01	0,05	15	0,07	0,42
26													1	0,19	1,79	11	0,60	3,46	47	0,08	0,43	4	0,18	1,08
27													3	0,07	0,66	6	0,02	0,12	1	0,05	0,27	2	0,57	3,44
28													1	0,24	2,26	9	0,23	1,33	10	0,43	2,29	1	0,16	0,96
29													2	0,46	4,33	1	0,34	1,96	1	0,37	1,97	10	0,14	0,84
30																1	0,64	3,69	1	0,03	0,16	13	0,79	4,76

Tabela 6.52: Ociosidade dos processos para os primeiros 30 envios de carga - processos de 8 a 15.

HibridaAdapt - processos de 0 a 7 -ambiente A_1																								
	0			*1*			*2*			*3*			*4*			*5*			*6*			*7*		
	NS	***T(s)***	***%***	***NS***	***T(s)***	***%***	***NS***	***T(s)***	***%***	***NS***	***T(s)***	***%***	***NS***	***T(s)***	***%***	***NS***	***T(s)***	***%***	***NS***	***T(s)***	***%***	***NS***	***T(s)***	***%***
1	627	0,00	0,00	352	0,69	14,56	279	1,13	18,14	142	0,90	11,32	143	1,50	8,79	63	0,76	3,76	184	1,14	5,28	99	0,87	10,70
2	5	0,29	6,42	345	0,04	0,84	41	0,28	4,49	90	0,24	3,02	1	0,65	3,81	50	1,42	7,02	24	0,46	2,13	367	0,46	5,66
3	1	0,46	10,18	13	0,26	5,49	39	0,49	7,87	5	0,08	1,01	6	0,24	1,41	29	0,27	1,34	11	0,27	1,25	8	0,88	10,82
4	1	0,49	10,84	1	0,61	12,87	5	0,20	3,21	20	0,09	1,13	4	0,25	1,46	61	0,62	3,07	26	0,26	1,20	1	0,14	1,72
5	32	0,16	3,54	3	0,26	5,49	36	0,19	3,05	53	0,55	6,92	1	0,22	1,29	1	0,23	1,14	16	0,45	2,08	3	0,19	2,34
6	5	0,39	8,63	2	0,33	6,96	289	0,31	4,98	41	0,04	0,50	35	0,36	2,11	3	0,15	0,74	1	0,15	0,69	124	0,06	0,74
7	10	0,05	1,11	1	0,35	7,38	29	0,12	1,93	6	0,12	1,51	5	0,40	2,34	24	1,03	5,09	1	0,33	1,53	13	0,64	7,87
8	1	0,25	5,53	79	0,84	17,72	7	0,05	0,80	5	0,40	5,03	1	0,23	1,35	1	0,64	3,17	1	0,45	2,08	21	0,03	0,37
9	1	0,51	11,28	1	0,08	1,69	7	0,29	4,65	7	0,33	4,15	19	0,84	4,92	4	0,44	2,18	1	0,44	2,04	9	0,27	3,32
10	1	0,20	4,42	1	0,09	1,90	1	0,31	4,98	98	0,56	7,04	1	0,42	2,46	7	0,78	3,86	43	0,10	0,46	2	0,14	1,72
11	12	0,08	1,77	5	0,15	3,16	2	0,85	13,64	182	0,36	4,53	3	0,04	0,23	5	0,55	2,72	7	0,35	1,62	3	0,44	5,41
12	1	0,13	2,88	3	0,14	2,95	33	0,77	12,36	26	0,15	1,89	1	0,63	3,69	1	0,25	1,24	6	0,91	4,21	87	0,54	6,64
13	8	0,31	6,86	1	0,23	4,85	81	0,02	0,32	20	0,32	4,03	4	0,15	0,88	1	0,17	0,84	1	0,14	0,65	3	0,68	8,36
14	14	0,38	8,41	3	0,19	4,01	1	0,13	2,09	1	0,31	3,90	23	0,14	0,82	1	0,94	4,65	2	0,44	2,04	11	0,79	9,72
15	3	0,11	2,43				1	0,22	3,53	1	0,64	8,05	1	0,14	0,82	1	0,16	0,79	1	0,18	0,83	3	0,07	0,86
16	1	0,26	5,75							3	0,25	3,14	2	0,08	0,47	1	0,66	3,26	5	0,61	2,82	3	0,10	1,23
17										1	0,70	8,81	1	0,12	0,70	1	0,48	2,37	1	0,46	2,13	44	0,06	0,74
18										1	0,44	5,53	3	1,32	7,73	1	0,34	1,68	1	0,94	4,35	7	0,11	1,35
19										94	0,73	9,18	1	0,76	4,45	6	0,40	1,98	1	0,31	1,43	1	0,16	1,97
20										3	0,31	3,90	3	0,81	4,75	165	0,25	1,24	1	0,80	3,70	1	0,25	3,08
21													3	0,02	0,12	59	0,30	1,48	1	0,53	2,45	1	0,19	2,34
22													48	0,35	2,05	4	0,24	1,19	1	0,27	1,25	24	0,48	5,90
23													31	0,00	0,00	2	0,69	3,41	5	0,39	1,80	1	0,12	1,48
24													11	0,12	0,70	3	0,37	1,83	1	0,63	2,92			
25													46	0,64	3,75	5	0,61	3,02	8	0,31	1,43			
26													86	0,88	5,16	6	0,08	0,40	164	0,55	2,55			
27													1	0,15	0,88	3	0,26	1,29	33	0,94	4,35			
28													15	0,53	3,10	2	0,14	0,69	1	0,06	0,28			
29													5	0,36	2,11	3	0,33	1,63	2	0,70	3,24			
30													1	0,14	0,82	1	0,43	2,13	17	0,54	2,50			

Tabela 6.53: Ociosidade dos processos para os primeiros 30 envios de carga - processos de 0 a 7.

HibridaAdapt - processo de 8 a 15 - ambiente A_1																								
Num	***8***			***9***			***10***			***11***			***12***			***13***			***14***			***15***		
	Nos	***T(s)***	***%***	***Nos***	***T(s)***	***%***	***Nos***	***T(s)***	***%***	***Nos***	***T(s)***	***%***	***Nos***	***T(s)***	***%***	***Nos***	***T(s)***	***%***	***Nos***	***T(s)***	***%***	***Nos***	***T(s)***	***%***
1	733	0,89	13,38	321	0,82	6,94	300	0,90	5,78	150	0,90	4,10	213	1,24	5,09	201	1,33	4,41	76	1,35	5,45	17	1,28	4,32
2	1	0,09	1,35	105	0,19	1,61	15	0,53	3,41	42	1,12	5,10	8	0,46	1,89	88	0,60	1,99	78	0,51	2,06	35	0,03	0,10
3	2	0,05	0,75	211	0,16	1,35	24	0,15	0,96	35	0,09	0,41	2	0,19	0,78	293	0,32	1,06	156	0,06	0,24	176	0,28	0,95
4	11	0,66	9,92	47	0,10	0,85	58	0,11	0,71	33	0,27	1,23	1	0,05	0,21	12	0,62	2,06	132	0,20	0,81	15	0,42	1,42
5	1	0,13	1,95	35	0,23	1,95	61	0,06	0,39	11	0,31	1,41	21	0,66	2,71	1	0,16	0,53	161	0,12	0,48	29	0,10	0,34
6	1	0,09	1,35	8	0,29	2,45	50	0,24	1,54	5	0,25	1,14	4	0,21	0,86	1	0,00	0,00	13	0,43	1,73	11	0,05	0,17
7	1	0,48	7,22	76	0,95	8,04	45	0,23	1,48	27	0,29	1,32	1	0,56	2,30	120	0,97	3,22	1	0,02	0,08	1	0,21	0,71
8	108	0,83	12,48	1	0,19	1,61	28	0,39	2,51	3	0,23	1,05	30	0,70	2,87	98	0,35	1,16	41	0,53	2,14	19	0,39	1,32
9	5	0,24	3,61	1	0,23	1,95	20	0,21	1,35	1	0,12	0,55	13	0,23	0,94	18	0,16	0,53	6	0,20	0,81	399	0,18	0,61
10	6	0,63	9,47	1	0,30	2,54	3	0,16	1,03	1	0,09	0,41	1	0,12	0,49	23	0,38	1,26	67	0,40	1,61	119	0,08	0,27
11	1	0,10	1,50	19	0,56	4,74	2	0,26	1,67	136	0,39	1,78	1	0,08	0,33	18	0,35	1,16	56	0,40	1,61	23	0,32	1,08
12	1	0,08	1,20	2	0,75	6,35	6	0,27	1,74	11	0,21	0,96	9	0,31	1,27	2	0,24	0,80	54	0,14	0,56	66	0,05	0,17
13	1	0,05	0,75				15	0,14	0,90	9	1,04	4,74	7	0,11	0,45	1	0,23	0,76	12	0,18	0,73	1	0,05	0,17
14	1	0,07	1,05				10	0,12	0,77	8	0,18	0,82	3	0,12	0,49	11	0,42	1,39	27	0,18	0,73	1	0,18	0,61
15	1	0,39	5,86				30	0,13	0,84	1	0,09	0,41	8	0,28	1,15	1	0,44	1,46	2	0,13	0,52	1	0,34	1,15
16	1	0,47	7,07				4	0,21	1,35	7	0,36	1,64	1	0,15	0,62	1	0,23	0,76	32	0,18	0,73	34	0,25	0,84
17	1	0,62	9,32				3	0,30	1,93	8	0,08	0,36	9	0,41	1,68	56	1,17	3,88	30	0,22	0,89	28	0,11	0,37
18	3	0,07	1,05				2	0,33	2,12	7	0,05	0,23	114	1,10	4,51	6	0,53	1,76	24	0,52	2,10	11	0,25	0,84
19	4	0,28	4,21				1	0,09	0,58	1	0,04	0,18	49	0,34	1,39	1	1,20	3,98	52	0,32	1,29	47	0,45	1,52
20	1	0,43	6,47				1	0,42	2,70	1	0,34	1,55	1	0,20	0,82	1	0,96	3,18	15	0,09	0,36	16	0,23	0,78
21							69	0,34	2,19	1	0,13	0,59	1	0,10	0,41	1	0,66	2,19	5	0,04	0,16	5	0,07	0,24
22							1	0,11	0,71	19	0,46	2,10	2	0,10	0,41	1	0,25	0,83	1	0,42	1,69	1	0,23	0,78
23							1	0,38	2,44	10	0,04	0,18	5	0,98	4,02	7	1,70	5,63	1	0,12	0,48	1	0,39	1,32
24							1	0,41	2,63	6	0,31	1,41	1	0,06	0,25	1	0,86	2,85	2	0,94	3,79	1	0,05	0,17
25							2	0,10	0,64	1	0,02	0,09	30	0,13	0,53	3	1,10	3,65	1	1,44	5,81	1	1,49	5,03
26							1	0,20	1,29	4	0,02	0,09	1	0,10	0,41	1	0,11	0,36	3	0,49	1,98	1	0,38	1,28
27							2	0,25	1,61	3	0,74	3,37	2	0,48	1,97	2	0,13	0,43	1	0,43	1,73	1	0,07	0,24
28							1	0,17	1,09	19	0,60	2,73	1	0,48	1,97	1	0,09	0,30	8	0,60	2,42	1	1,22	4,12
29							1	0,41	2,63	4	0,08	0,36	1	0,24	0,98	1	0,17	0,56	25	1,03	4,15	1	0,52	1,76
30							14	0,73	4,69	4	0,13	0,59	1	0,68	2,79	3	0,21	0,70	1	0,15	0,61	1	0,94	3,17

Tabela 6.54: Ociosidade dos processos para os primeiros 30 envios de carga - processos de 8 a 15.

MeAdapt - processos de 0 a 7 -ambiente A_1																								
	0			*1*			*2*			*3*			*4*			*5*			*6*			*7*		
	NS	*T(s)*	*%*	*NS*	*T(s)*	*%*	*NS*	*T(s)*	*%*	*NS*	*T(s)*	*%*	*NS*	*T(s)*	*%*	*NS*	*T(s)*	*%*	*NS*	*T(s)*	*%*	*NS*	*T(s)*	*%*
1				597	0,80	65,57	393	1,02	43,40	206	1,40	59,57	218	1,44	68,90	396	1,78	67,68	165	1,83	63,10	195	1,83	50,27
2				15	0,01	0,82	81	0,00	0,00	192	0,02	0,85	191	0,02	0,96	69	0,00	0,00	327	0,01	0,34	71	0,04	1,10
3				2	0,03	2,46	20	0,00	0,00	62	0,01	0,43	70	0,02	0,96	95	0,03	1,14	179	0,02	0,69	249	0,02	0,55
4				1	0,03	2,46	27	0,01	0,43	24	0,01	0,43	3	0,01	0,48	35	0,00	0,00	17	0,00	0,00	73	0,02	0,55
5				1	0,01	0,82	16	0,01	0,43	27	0,02	0,85	64	0,00	0,00	22	0,00	0,00	1	0,10	3,45	38	0,01	0,27
6				2	0,00	0,00	18	0,00	0,00	0	0,01	0,43	36	0,01	0,48	6	0,00	0,00	12	0,02	0,69	5	0,00	0,00
7				1	0,02	1,64	3	0,03	1,28	28	0,16	6,81	1	0,01	0,48	1	0,07	2,66	1	0,04	1,38	4	0,40	10,99
8				1	0,07	5,74	12	0,41	17,45	10	0,09	3,83	8	0,06	2,87	1	0,01	0,38	1	0,32	11,03	7	0,06	1,65
9				1	0,07	5,74	4	0,01	0,43	1	0,04	1,70	1	0,01	0,48	6	0,04	1,52				1	0,23	6,32
10				1	0,03	2,46	1	0,06	2,55	2	0,02	0,85	1	0,05	2,39	4	0,20	7,60				1	0,10	2,75
11				1	0,13	10,66	1	0,12	5,11				1	0,00	0,00	1	0,01	0,38				1	0,01	0,27
12				1	0,02	1,64	1	0,04	1,70							1	0,07	2,66				2	0,00	0,00
13							1	0,10	4,26							1	0,10	3,80						
14							1	0,03	1,28							1	0,05	1,90						
15							3	0,03	1,28							2	0,23	8,75						
16							2	0,08	3,40															
17																								
18																								
19																								
20																								
21																								
22																								
23																								
24																								
25																								
26																								
27																								
28																								
29																								
30																								

Tabela 6.55: Ociosidade dos processos para os primeiros 30 envios de carga - processos de 0 a 7.

MeAdapt - processo de 8 a 15 - ambiente A_1																								
Num	***8***			***9***			***10***			***11***			***12***			***13***			***14***			***15***		
	Nos	***T(s)***	***%***	***Nos***	***T(s)***	***%***	***Nos***	***T(s)***	***%***	***Nos***	***T(s)***	***%***	***Nos***	***T(s)***	***%***	***Nos***	***T(s)***	***%***	***Nos***	***T(s)***	***%***	***Nos***	***T(s)***	***%***
1	152	2,20	50,34	195	2,24	43,08	129	2,50	44,48	47	2,19	41,40	139	8,62	60,79	53	10,07	48,23	69	10,83	49,68	111	2,77	63,97
2	69	0,00	0,00	195	0,05	0,96	45	0,08	1,42	134	0,00	0,00	254	0,16	1,13	122	1,57	7,52	32	0,06	0,28	83	0,13	3,00
3	162	0,04	0,92	310	0,01	0,19	294	0,00	0,00	308	0,01	0,19	697	0,25	1,76	520	0,08	0,38	1082	0,07	0,32	296	0,02	0,46
4	100	0,00	0,00	78	0,02	0,38	31	0,02	0,36	20	0,03	0,57	84	0,13	0,92	518	0,12	0,57	121	0,00	0,00	2	0,00	0,00
5	90	0,04	0,92	41	0,02	0,38	105	0,05	0,89	68	0,00	0,00	149	0,10	0,71	47	0,14	0,67	241	0,02	0,09	95	0,00	0,00
6	28	0,04	0,92	26	0,00	0,00	31	0,01	0,18	43	0,02	0,38	174	0,08	0,56	424	0,11	0,53	179	0,10	0,46	97	0,05	1,15
7	17	0,01	0,23	6	0,18	3,46	3	0,68	12,10	67	0,02	0,38	15	0,00	0,00	19	0,05	0,24	140	0,04	0,18	77	0,05	1,15
8	19	0,99	22,65	8	0,90	17,31	3	0,97	17,26	35	0,01	0,19	148	0,08	0,56	71	0,11	0,53	2	0,00	0,00			
9	5	0,04	0,92	9	0,54	10,38	23	0,28	4,98	1	0,11	2,08	95	0,10	0,71	8	3,44	16,48	17	0,00	0,00			
10				1	0,06	1,15				10	1,01	19,09	2	0,00	0,00	1	0,31	1,48	1	1,26	5,78			
11				1	0,04	0,77				3	0,21	3,97	71	0,00	0,00	1	0,28	1,34	5	3,32	15,23			
12				1	0,01	0,19				3	0,07	1,32	16	0,01	0,07	3	0,19	0,91	1	1,07	4,91			
13				1	1,13	21,73				1	0,10	1,89	3	0,00	0,00	1	0,87	4,17	2	0,42	1,93			
14										1	0,01	0,19	46	0,07	0,49				1	1,24	5,69			
15													1	0,09	0,63									
16													8	0,82	5,78									
17													9	0,05	0,35									
18													1	0,13	0,92									
19													1	0,01	0,07									
20													1	0,46	3,24									
21													1	0,04	0,28									
22																								
23																								
24																								
25																								
26																								
27																								
28																								
29																								
30																								

Tabela 6.56: Ociosidade dos processos para os primeiros 30 envios de carga - processos de 8 a 15.

Instâncias	A_3					
	GlobalAdapt		***HibridaAdapt***		***MeAdapt***	
	Effs 16p	***Effs 24p***	***Effs 16p***	***Effs 24p***	***Effs 16p***	***Effs 24p***
320-311	1,38	**1,46**	1,35	**1,41**	1,35	**1,52**
320-313	1,54	**1,63**	1,47	**1,54**	1,46	**1,61**
320-314	1,00	**1,10**	1,00	**1,15**	0,99	**1,10**
320-341	**0,63**	**0,63**	0,61	**0,78**	0,72	**0,78**
320-343	**0,65**	0,23	**0,69**	0,22	**0,80**	0,27
320-344	**0,83**	0,44	**0,77**	0,45	**0,94**	0,52
320-345	**0,76**	0,50	**0,71**	0,53	**0,86**	0,58
640-212	0,99	**1,05**	0,99	**1,05**	0,98	**1,10**
640-213	1,11	**1,18**	1,07	**1,20**	1,12	**1,21**
640-214	0,99	**1,09**	0,99	**1,08**	0,98	**1,09**
640-215	1,02	**1,11**	1,01	**1,21**	0,99	**1,13**

Tabela 6.57: Comparação quanto à eficiência entre as estratégias *GlobalAdapt*, *HibridaAdapt* e *MeAdapt* - 2*clusters* X 4 *clusters* .

6.5 *MeAdapt* x *GlobalAdapt* x *HibridaAdapt* - 4 *clusters*

Nesta seção são apresentados os testes da execução simultânea das estratégias desenvolvidas, *GlobalAdapt*, *HibridaAdapt* e *MeAdapt* no ambiente A_3 quando utilizados 4 *clusters* contendo 6 processadores cada um.

A Tabela 6.57 apresenta a eficiência das estratégias *GlobalAdapt*, *HibridaAdapt* e *MeAdapt*. A coluna $Eff16p$ apresenta a eficiência obtida quando as intâncias são executadas utilizando 2 *cluster* contendo 8 processadores cada um totalizando 16 processadores e a coluna $Eff24p$ 4 *clusters* contendo 6 processadores cada um totalizando 24 processadores. Os valores obtidos para $Effs24p$ mostra que o algoritmo B&B associado a qualquer uma das estratégias é escalável para as instâncias que possuem um alto grau de paralelismo. Por outro lado, as instâncias *i320-343*, *i320-344* e *i320-345*, são pequenas e o aumento de processadores implica em degradação da eficiência.

Como o ambiente adotado ainda é muito pequeno e distante da realidade das *Grids* atuais miores análises em relação à escalabilidade não foram possíveis.

Capítulo 7

Conclusão

O principal objetivo deste trabalho foi propor três estratégias de balancemanto de carga e comparará-las com duas outras existentes na literatura para o algoritmo *branch-and-bound* paralelo aplicado ao Problema de *Steiner* em Grafos executado em um ambiente *Grid*. Os testes foram realizados de forma a avaliar e classificar qual das estratégias mais contribuiu para o aumento da eficiência paralela do algoritmo diante de três diferentes ambientes, sendo o primeiro deles uma *Grid* homogêneo, o segundo, uma *Grid* homogêneo mas com a presença efetiva de carga externa, e o terceiro, um *Grid* heterogêneo com presença de carga externa. Para cada ambiente, foram testadas onze instâncias de incidência do PSG contidas no repositório *SteinLib* [Koch et al. 2003].

Os resultados obtidos comprovam que as estratégias desenvolvidas, *GlobalAdapt* e *HibridaAdapt* diminuem a ocosidade dos processos, o *overhead* de comunicação entre os mesmos e, em média, reduz o tempo de execução da aplicação quando comparadas com as estratégias propostas por [Drummond et al. 2005], *Global* e *Híbrida*. Este ganho é mais efetivo em um ambiente heterogêneo e com carga externa. A utilização de informações do problema, como a estimativa do tamanho da carga a ser enviada durante a transferência, bem como informações do ambiente, como o desempenho dos processadores adotadas pelas estratégias desenvolvidas permitiu manter a carga dos processos balanceadas por maiores períodos de tempo.

Dentre as estratégias desenvolvidas, a *MeAdapt* se mostrou melhor que as outras para instâncias menores e *GlobalAdapt* para instâncias maiores. Já a estratégia *HibridaAdapt*, apesar de superar a *Hibrida*, não se mostrou mais eficiente que as outras, pois ao evitar a troca de mensagens entre *clusters* com o intuito de diminuir o *overhead* de comunicação mantem os processos ociosos por mais tempo.

Trabalhos futuros podem consistir em adequar melhor a quantidade de carga enviada durante a transferência entre *clusters* na estratégia *HibridaAdapt*. Além disso, seria interessante aumentar e variar o número de processos por *cluster* e o próprio número de *clusters* para que seja observado o desempenho das estratégias e as escalabilidades. Existem atualmente cinco instâncias do repositório *SteinLib* [Koch et al. 2003] ainda não resolvidas. Acredita-se que, aumentando-se o número de processadores, o algoritmo *branch-and-bound* associado à estratégia *GlobalAdapt* será capaz de resolvê-las.

Capítulo 8

Apresentações e Publicações

- Drummond, L.M.A.; Silva,J.M.N.; Gonçalves, A.D.; **Balanceamento de Carga em um Algoritmo Branch-and-Bound para Execução em Grades Computacionais**, *III Workshop de Grade Computacional e Aplicações, WCGA 2005 (apresentado como pôster).*

- Silva,J.M.N.; Drummond, L.M.A.; Uchoa, E. **Balanceamento de Carga em um Algoritmo Branch-and-Bound para Execução em Grades Computacionais**, *VI Workshop de Sistemas Computacionais de Alto Desempenho.*

- Drummond, L.M.A.; Uchoa, E; Gonçalves, A.D.; Silva, J.M.N.; Santos, M.C.P.; Castro, M. C. S. **A Grid-Enabled Distributed Branch-and-Bound Algorithm with Application on the Steiner Problem in Graphs**, *Parallel Computing (aceito para publicação).*

Referências

[Aida et al. 2003] Aida, K.; Natsume, W. e Futakata, Y. **Ditributed Computing with Hierarchical Master-Worker Paradigm for Parallel Branch-and-Bound Algorithms**. In: *Third International Symposium on Cluster Computing and Grid*, Tokyo, Japan, p. 156–162, 2003.

[k. Anstreicher et al. 2002]k. Anstreicher; Brixius, N.; Goux, J. e Linderoth, J. **Solving Large Quadratic Assignment Problems on Computational Grids**. *Mathematical Programming*, Series B 91, p. 563–588, 2002.

[Baoukov e Soverik 1999] Baoukov, R. e Soverik, T. **A Generic Parallel Branch-and-Bound Environment on a Network of Workstations**. In: *High Performance Dinghy Regatta*, p. 474–483, 1999.

[Bruin et al. 1995] Bruin, A.; Kindervater, G. A. P. e Trienekens, H. W. J. M. **Asynchronous Parallel Branch-and-Bound and Anomalies**. Department of Computer Science Erasmus University, Roterdam, 1995.

[Culler et al. 1993] Culler, D.; Karp, R.; Patterson, D.; Sahay, A.; Schauser, K.; Santos, E.; Subramonian, R. evon Eicken, T. **LogP: towards a realistic model of parallel computation**. In: *4th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*, San Diego, United States, p. 1–12, 1993.

[Drummond et al. 2004] **Drummond, L. M. A.; Gonçalves, A. D.; Filho, J. V.; Uchoa, E. e Castro, M. C. S.** In: *6th International Meeting VECPAR 04 High Performance Computing for Computaional Science*, Valencia, Spain, v. 3, p. 927–932, 2004.

[Drummond et al. 2005] Drummond, L. M. A.; Uchoa, E.; Gonçalves, A. D.; Silva, J. M. N.; Santos, M. C. P. e Castro, M. C. S. **A Grid-Enabled Distributed Branch-and-Bound Algorithm with Application on the Steiner Problem in Graphs**. *Parallel Computing (aceito para publicação)*, 2005.

[Du et al. 2000] Du, D.-Z.; Smith, J. M. e Rubistein, J. H. **Advances in Steiner Tree**. *Combinatorial Optimization Series*, v. 6, 2000.

[Duin 1993] Duin, C. **Steiner's Problems in Graphs**. Tese (Thèse de Doctorat) — University of Amsterdam, 1993.

[Foster e Kesselman 1998] Foster, I. e Kesselman, C. **The Globus Project: a Status Report**. In: *Seventh Heterogeneous Computing Workshop*, Orlando, Florida, p. 4–18, 1998.

[Foster e Kesselman 1999] Foster, I. e Kesselman, C. **The Globus project: a status report, Future Generation**. *Computer Systems*, v. 15, p. 607–621, 1999.

[Gendron 1994] Gendron, B. **Parallel Branch-and-Bound Algorithms: Survey and Synthesis**. *Operations Research*, v. 42, p. 1042–1066, 1994.

[Goux et al. 2001] Goux, J. P.; Kulkarni, S.; Linderoth, J. T. e Yoder, M. E. **Master-Worker: an enabling framework for applications on the computational Grid**. *Cluster Computing*, v. 4, p. 63–70, 2001.

[Heymann et al. 2000] Heymann, E.; Senar, M. A.; Luque, E. e Livny, M. **Evaluation of an Adaptive Scheduling Strategy for Master-Worker Applications on Clusters of Workstations**. *Lecture Notes in Computer Science*, v. 1970, p. 310–319, 2000.

[Karonis et al. 2003] Karonis, N.; Toonen, B. e Foster, I. **MPICH-G2: A Grid-enabled implementation of the message passing interface**. *Parallel and Distributed Computing*, v. 63, p. 551–563, 2003.

[Koch et al. 2003] Koch, T.; Martin, A. e Voss, S. **SteinLib: An Update Library on Steiner Problems in Graphs**. *Konrad-Zuse-Zentrum für Informationstechnik*, Elsevier Science Publishers B. V., Amsterdam, The Netherlands, p. 00–37, 2003.

[Lai e Sahni 1984] Lai, T. H. e Sahni, S. **Anomalies in parallel branch-and-bound algorithms**. *Communications of the ACM*, v. 27, p. 594–602, 1984.

[Land e Doig 1960] Land, A. H. e Doig, A. G. **An automatic method for solving discrete programming problems**. *Econometrica*, v. 28, p. 497–520, 1960.

[Oliveira et al. 2005] Oliveira, A.; Argolo, G.; Iglesias, P.; Martins, S. L. e Plastino, A. **Evaluating a Scientific SPMD Application on a Computational Grid with Different Load Balancing Techniques**. In: *5th IEEE International Symposium and School on Advance Distributed Systems*, Guadalajara, México, v. 3563, p. 301, 2005.

[Plastino 1999] Plastino, A. **Uma Ferramenta para Desenvolvimento de Aplicações SPMD com Suporte para Balanceamento de Carga**. Tese (Doutorado) — Universidade Católica do Rio de Janeiro, 1999.

[Ralphs et al. 2004] Ralphs, T. K.; Danyi, L. L. e Saltzman, M. J. **A Library Hierarchy for Implementing Scalable Parallel Search Algorithms**. *The Journal of Supercomputing*, v. 28, p. 594–602, 2004.

[Roucairol e Dowaji 1995] Roucairol, C. e Dowaji, S. **Load Balancing Strategy and Priority of Tasks in Distributed Environments**. In: *Computers and Communications*, p. 156–162, 1995.

[Shao et al. 2000] Shao, G.; Berman, F. e Wolski, R. **Master/Slave Computing on the Grid**. In: *9th Heterogeneous Computing Workshop*, Cancun, Mexico, p. 3–16, 2000.

[Snir et al. 1996] Snir, M.; Otto, S. M.; Huss-Lederman, S. e Dongarra, J. **MPI: The Complete Reference**. : The MIT Press, 1996.

[Thomé et al. 2006] Thomé, V.; Vianna, D.; Costa, R.; Plastino, A. e Teixeira, O. **Exploring Load Balancing in a Scientific SPMD Parallel Application**. *Special Issue of the International Journal of Computational Science and Engineering (aceito para publicação)*, 2006.

[Uchoa 2001] Uchoa, E. **Algoritmos para Problemas de Steiner com Aplicações em Projeto de Circuitos VLSI**. Tese (Doutorado) — Universidade Católica do Rio de Janeiro, 2001.

[Uchoa 2002] Uchoa, E. **Preprocessing Steiner problems from VLSI layout**. *Networks*, v. 40, p. 38–50, 2002.

[Werneck 2001] Werneck, R. **Problema de Steiner em Grafos: Algoritmos Primais, Duais e Exatos**. Tese (Doutorado) — Universidade Católica do Rio de Janeiro, 2001.

[Willebeek-LeMair e Reeves 1993] Willebeek-LeMair, M. H. e Reeves, A. P. **Strategies for Dynamic Load Balancing on Highly Parallel Computers**. *IEEE Transactions on Parallel and Distributed System*, v. 4, n. 9, p. 979–993, 1993.

[Wong 1984] Wong, R. **A dual ascent Approach for Steiner tree problems on a directed graph**. *Mathematical Programming*, v. 28, p. 271–287, 1984.

[Wong 2001] Wong, R. **M. P. Aragão and E. Uchoa and R. F. Werneck**. *Electronic Notes in Discrete Mathematics*, v. 7, p. 46–51, 2001.

[Zaki e Parthasarathy 1997] Zaki, M. e Parthasarathy, W. **Customized Dynamic Load Balancing for a Network of Workstations**. *Journal of Parallel and Distributed Computing*, v. 43, n. 2, p. 156–162, 1997.