Escalonadores de Tarefas Dependentes para Grades Robustos às Incertezas das Informações de Entrada

Este exemplar corresponde à redação final da Tese devidamente corrigida e defendida por Daniel Macêdo Batista e aprovada pela Banca Examinadora.

Campinas, 23 de fevereiro de 2010.

Prof. Dr. Nelson Luis Saldanha da Fonseca
Instituto de Computação, Unicamp
(Orientador)

Tese apresentada ao Instituto de Computação, UNICAMP, como requisito parcial para a obtenção do título de Doutor em Ciência da Computação.

Substitua pela ficha catalográfica

(Esta página deve ser o verso da página anterior mesmo no caso em que não se imprime frente e verso, i.é., até 100 páginas.)

Substitua pela folha com as assinaturas da banca

Instituto de Computação
Universidade Estadual de Campinas

Escalonadores de Tarefas Dependentes para Grades Robustos às Incertezas das Informações de Entrada

Daniel Macêdo Batista[1]

22 de janeiro de 2010

Banca Examinadora:

- Prof. Dr. Nelson Luis Saldanha da Fonseca
  Instituto de Computação, Unicamp (Orientador)
- Prof. Dr. Flávio Keidi Miyazawa
  Instituto de Computação, Unicamp
- Profa. Dra. Islene Calciolari Garcia
  Instituto de Computação, Unicamp
- Prof. Dr. Artur Ziviani
  Laboratório Nacional de Computação Científica
- Prof. Dr. Gustavo Bittencourt Figueiredo
  NUPERC, Universidade Salvador
- Prof. Dr. Cid Carvalho de Souza
  Instituto de Computação, Unicamp (Suplente)
- Prof. Dr. Bruno Richard Schulze
  Laboratório Nacional de Computação Científica (Suplente)
- Prof. Dr. Otto Carlos Muniz Bandeira Duarte
  COPPE, UFRJ (Suplente)

[1]Auxílio Financeiro do CNPq processo 141517/2006-9

Resumo

Para que escalonadores em grades derivem escalonamentos, é necessário que se forneçam as demandas das aplicações e as disponibilidades dos recursos das grades. No entanto, a falta de controle centralizado, o desconhecimento dos usuários e a imprecisão das ferramentas de medição fazem com que as informações fornecidas aos escalonadores difiram dos valores reais que deveriam ser considerados para se obter escalonamentos quase-ótimos.

A presente Tese introduz dois escalonadores de tarefas robustos às incertezas das informações providas como entrada ao escalonador. Um dos escalonadores lida com informações imprecisas sobre as demandas das aplicações, enquanto que o outro considera tanto imprecisões das demandas quanto da disponibilidade de recursos. A eficácia e a eficiência dos escalonadores robustos às incertezas são avaliadas através de simulação. Comparam-se os escalonamentos gerados pelos escalonadores robustos com os produzidos por escalonadores sensíveis às informações incertas.

A eficácia de estimadores de largura de banda disponível são, também, avaliadas, através de medição, à luz da adoção destes em sistemas de grades, a fim de que se possa utilizar suas estimativas como informação de entrada a escalonadores robustos.

Abstract

Schedulers need information on the application demands and on the grid resource availability as input to derive efficient schedules for the tasks of a grid application. However, information provided to schedulers differ from the true values due to the lack of central control in a grid and the lack of ownership of resources as well as the precision of estimations provided by measurement tools.

This thesis introduces two robust schedulers based on fuzzy optimization. The first scheduler deals with uncertainties on the application demands while the other with uncertainties of both application demands and resource availability. The effectiveness of these schedulers are evaluated via simulation and the schedules produced by them are compared to those of their non-fuzzy counterpart.

Moreover, the efficacy of available bandwidth estimators is assessed in order to evaluate their use in grid systems for providing schedulers with useful input information.

Ao meu pai José, à minha mãe Bernadete e ao meu irmão Joberson.

Agradecimentos

Lá se foram 4 anos e mais uma etapa importante da minha vida está concluída! A citação do Roger Waters que aparecerá daqui a três páginas resume bem como foram esses 4 anos. Seria impossível ter escrito esta Tese sem o apoio da minha família e dos meus amigos. Também é impossível, ao lembrar de tudo que passamos juntos nesse tempo, conter as lágrimas.

Nelson, muito obrigado por ter sido meu orientador. Suas sugestões para a melhoria do meu trabalho foram essenciais para que chegasse onde eu cheguei. Eu não tenho dúvida de que a experiência que eu obtive trabalhando junto com você nesta Tese, nos artigos, nas disciplinas, na organização dos eventos e na COMST fará muita diferença na minha carreira. Espero que possamos continuar trabalhando juntos nos próximos anos.

Muito obrigado aos Pistolinhas pela companhia!! Augusto, Bartho (finalmente o Bartho pode ser considerado um pistolinha), Cléo, Gustavo, Triste e Carlinha.

Carlinha, você merece um parágrafo só para você. Você veio para Campinas!!! Largou o ambiente paradisíaco de Salvador para suportar o frio daqui. Ter você por perto durante a etapa final do meu doutorado me deu muito ânimo para seguir em frente. Vai ser difícil retribuir tudo de bom que você me proporcionou por conta da sua companhia. Você sempre me ajudou quando eu estive desanimado, cansado e reclamando de tudo. Você merece é um prêmio por ter aguentado o meu mau humor tantas vezes. Obrigado fofinha!!!

Muito obrigado ao pessoal do curso de GNU/Linux! Carlinha (de novo!), César, Christian, Geraldo, Jáder, Joana, Jorge, Luciano e Neila. As aulas e os encontros foram inesquecíveis. Não foram poucas as vezes em que as nossas reuniões foram as melhores coisas que aconteceram nas semanas.

Neilinha, você também merece um parágrafo seu. Nos conhecemos justamente por causa desta Tese. Com tantos trabalhos para você escolher na disciplina de Redes você terminou escolhendo um que fez você trabalhar comigo. Que cara de sorte eu sou! A gente acabou descobrindo um monte de interesses em comum e eu ganhei uma grande amiga. Jamais vou esquecer das coincidências, com destaque para aquela sua ida inesperada para Salvador. E jamais vou esquecer da guitarra do Beatles Rock Band. Ter o jogo me ajudou a colocar o *stress* de lado muitas vezes. Obrigado por compartilhar comigo um pouco das

suas opiniões sobre o mundo e sobre religião. Obrigado por me ouvir tantas vezes e muito obrigado também pelas parafernálias eletrônicas. Graças a você eu acabei entrando no mundo da Apple. Ah, e obrigado pelas dicas de músicas! Trabalhar ao som delas facilitou a escrita de muitos artigos.

Muito obrigado ao pessoal do curso de jogos! Carlos e Christian, foi muito bom aprender um pouco de programação para jogos com vocês. Na verdade o Carlos também merece um muito obrigado pela amizade e pelo incentivo para que eu adquirisse o PS3 e o PSP. Voltar a jogar videogame foi excelente.

Muitíssimo obrigado a todos que passaram pelo LRC nesses 4 anos! Aprendi muito com você. Vou repetir um monte de nome aqui mas não tem problema: André, Carlos Froldi, Carlos Senna, Cláudio, César, Christian, Flavio, Franklin, Geraldo, Gustavo Bittencourt, Jorge, Jó, Juliana Borin, Juliana de Santi, Leonardo, Luciano, Luiz, Neumar, Pedro, Rafael e Walisson. Ah André, suas aulas sobre fuzzy ajudaram a alcançar os resultados apresentados nesta Tese. Muito obrigado! Também tenho que agradecer ao Gustavo Mamede, à Juliana de Santi, ao André e ao Carlos Froldi por terem achado tempo para revisarem os meus textos. Obrigado!

Muito obrigado aos melhores pais e ao melhor irmão do mundo por existirem e por fazerem parte da minha família!

E assim como fiz nos agradecimentos da dissertação de mestrado, eu não poderia deixar de agradecer ao povo brasileiro. A bolsa e a taxa de bancada do CNPq ajudaram para que eu permanecesse com o foco em minha pesquisa durante esses 4 anos. Muito obrigado!

Together we stand, divided we fall.

(Roger Waters)

Sumário

Lista de Tabelas

Lista de Figuras

Lista de Algoritmos

Capítulo 1

Introdução

Os crescentes avanços nas tecnologias de comunicação nos anos 90, traduzidas principalmente no sucesso da Internet, motivaram usuários, desenvolvedores e administradores de *clusters* e de supercomputadores a construírem aplicações que utilizassem recursos em escala mundial. Aplicações que antes utilizavam uma quantidade limitada de recursos de uma única organização passaram a ter à disposição centenas de milhares de computadores que passaram a formar organizações virtuais. O poder de processamento desses novos computadores virtuais era resultante principalmente do aproveitamento dos ciclos não utilizados de computadores pessoais.

Os ótimos resultados alcançados com as primeiras iniciativas de computação paralela em nível mundial, com destaque para o projeto SETI@home [97], que utiliza a capacidade ociosa de computadores para analisar sinais de rádio em busca de inteligência extraterrestre, motivaram a pesquisa e o desenvolvimento de soluções que permitissem a construção de ambientes voltados para a execução das mais diversas aplicações. Instituições de pesquisa envolvidas com áreas muito dependentes da computação, como física, química e astronomia, passaram a propor projetos que compartilhavam não apenas o poder de processamento das máquinas; equipamentos científicos, softwares e até mesmo os enlaces de rede passaram a ser compartilhados.

No final dos anos 90, esses ambientes virtuais, que utilizavam os enlaces de rede de longa distância como se fossem barramentos internos de um computador, passaram a ser chamados de grades em alusão à grade de energia elétrica, em que o usuário recebe o serviço sem se preocupar onde e como ele está sendo realizado [34]. Desde então, argumentos a favor da utilização de grades são cada vez mais frequentes. Enlaces de rede com capacidade de transmissão cada vez maiores, equipamentos pessoais cada vez mais poderosos devido ao paralelismo existente nos *chips* e a carência de aplicações que tiram proveito de toda a largura de banda e de todo o processamento disponível têm transformado a Internet em um enorme supercomputador virtual através do uso de sua

capacidade ociosa.

Tantas vantagens na utilização das grades fez com que elas saíssem dos ambientes acadêmicos e de pesquisa e passassem a surgir como solução para empresas e usuários que estão interessados em obter serviços de computação sem ter que se preocupar com a localização desses. A analogia com o sistema elétrico, que ainda era ficção nos anos 90, começa a se tornar realidade e as tecnologias que foram desenvolvidas servem de alicerce para a chamada "computação em nuvem" (*cloud computing*) [41], na qual os usuários não precisam instalar aplicativos e nem armazenar os seus arquivos localmente, pois estes estão disponíveis na nuvem.

Apesar das inúmeras vantagens na utilização das grades, suas implementações exigem soluções de diversas questões oriundas da não utilização de sistemas confinados em redes locais; O escalonamento é um dos principais mecanismos que precisa ser definido. As aplicações são submetidas às grades como um conjunto de programas menores chamados de tarefas, a serem executadas nos diversos recursos disponíveis. A heterogeneidade dos recursos e a necessidade de considerar que a rede está disponível para diversas aplicações ao mesmo tempo impõe restrições na busca pelo melhor subconjunto de recursos para executar as tarefas.

A inexistência de um escalonador ideal para tarefas em grades tem motivado a comunidade científica a propor diversas soluções para o escalonamento. O interesse em pesquisar essas soluções pode ser observado nas publicações da área de grades. Em 2009, conferências internacionais como o IEEE IPDPS [92] e o IEEE CCGRID [98] tiveram, cada uma, duas sessões para acomodar todos os artigos relacionados com escalonamento; esse foi o único tópico a possuir duas sessões em ambas as conferências.

Para que os escalonadores de tarefas possam encontrar escalonamentos próximos dos ótimos é necessário que eles recebam como entrada os requisitos das tarefas e o estado atual dos recursos. Somente de posse dessas informações, é possível estimar os ganhos que a aplicação teria caso fosse escalonada em um dado conjunto de recursos. No entanto, a falta de um controle centralizado, o desconhecimento dos usuários e a imprecisão nas ferramentas que estimam as capacidades disponíveis dos recursos fazem com que as informações passadas para o escalonador possam ser diferentes das reais [67].

Algumas propostas encontradas na literatura propõem que os escalonadores de tarefas sejam executados repetidamente durante a execução das aplicações [78] [19] [39] [75] [1] [72] [71] [18]. Caso seja observada, em tempo de execução, alguma diferença em relação à execução anterior, migrações de tarefas são realizadas. O problema dessas soluções é o aumento na complexidade do sistema de gerência da grade. O monitoramento constante da disponibilidade dos recursos e a migração de tarefas aumenta a intrusão na grade e gera carga extra para a execução das aplicações. Isso pode ser evitado caso os escalonadores de tarefas sejam robustos frente às incertezas nas informações dos requisitos das aplicações

e da disponibilidade dos recursos.

Escalonadores de tarefas robustos devem fornecer escalonamentos próximos do ótimo mesmo que as informações sobre as demandas das aplicações e sobre a disponibilidade dos recursos não sejam precisas. A imprecisão nestes valores é normalmente traduzida em um intervalo de valores possivelmente válidos. Por exemplo, ao invés de receber como entrada um único número que represente a disponibilidade no caminho entre dois recursos de processamento, o escalonador deve receber um intervalo de variação de largura de banda disponível para tomar decisões de escalonamento.

Esta Tese apresenta dois escalonadores de tarefas robustos às incertezas das informações de entrada e avalia o desempenho de estimadores de largura de banda disponível para utilização em grades. Esses estimadores caracterizam-se por fornecerem intervalos que representam a disponibilidade nos caminhos de rede entre os recursos de processamento.

O primeiro escalonador de tarefas apresentado é modelado como um problema de programação inteira 0–1. Ele utiliza técnicas de otimização fuzzy para lidar com as incertezas nas descrições das aplicações e propõe escalonamentos para minimizar o tempo de execução das aplicações nas grades. A motivação para utilizar programação inteira foi consequência dos bons resultados alcançados em estudos envolvendo o escalonamento de tarefas em ambientes sem incerteza [12]. O segundo escalonador também utiliza programação inteira 0–1. Ele expande o primeiro para ser robusto tanto às incertezas nas descrições das aplicações quanto às incertezas na disponibilidade dos recursos.

Utiliza-se simulação para avaliar os dois escalonadores em termos da qualidade dos escalonamentos produzidos e dos seus tempos de execução em função das incertezas nas informações passadas como entrada. Diversas aplicações e topologias de grades são utilizadas nos experimentos. A comparação entre os resultados produzidos pelos novos escalonadores e os resultados produzidos por escalonadores que ignoram as incertezas demonstram a utilidade das novas propostas.

Além disso, os estimadores de largura de banda disponível são avaliados em diversos experimentos, utilizando medição, para que se possa indicar o estimador com melhores características para ser utilizado em conjunto com escalonadores que sejam robustos às incertezas na disponibilidade dos recursos.

As seções a seguir apresentam algumas informações sobre esta Tese: na Seção 1.1 são apresentadas as contribuições. A Seção 1.2 lista todas as publicações decorrentes dos resultados encontrados e a Seção 1.3 resume o conteúdo dos demais capítulos.

1.1 Contribuições

A lista apresentada a seguir descreve todas as contribuições desta Tese:

- Investigação do estado da arte de mecanismos propostos para lidar com incertezas e flutuações durante a execução de aplicações em grades;
- Proposta de um escalonador de tarefas para grades robusto às incertezas na descrição das aplicações. Experimentos de simulação mostram que o escalonador produz escalonamentos melhores do que aqueles propostos por escalonadores que ignoram as incertezas consideradas;
- Proposta de um escalonador de tarefas para grades robusto às incertezas na descrição das aplicações e na disponibilidade dos recursos. Experimentos de simulação comprovam a utilidade do escalonador;
- Apresentação do estado da arte de ferramentas para estimação de largura de banda disponível e avaliação de desempenho de duas ferramentas do ponto de vista de um estimador ideal para grades.

1.2 Publicações realizadas

Os resultados apresentados nesta Tese geraram 11 publicações. A investigação de mecanismos existentes para lidar com a dinâmica da grade gerou resultados que foram publicados em [7], em [11] e em [10]. Os resultados alcançados com o tratamento de incertezas na descrição das aplicações foram publicados em [13], em [8] e em [14]. Os resultados alcançados com o tratamento de incertezas na disponibilidade dos recursos das grades foram publicados em [15]. Os resultados obtidos com a análise de desempenho de estimadores de largura de banda disponível em grades foram publicados em [5], em [4] e em [6]. Um resumo do andamento da pesquisa que levou a todos os resultados apresentados nesta Tese foi publicado em [9]. Foram também elaborados 2 artigos que foram submetidos a periódicos de circulação internacional.

1.3 Organização da Tese

O objetivo desta Tese é apresentar mecanismos que garantam um bom funcionamento das grades independente das incertezas inerentes ao ambiente. A dinâmica dos recursos, as incertezas das demandas das aplicações e as imprecisões dos estimadores da disponibilidade dos recursos devem ser contornadas de modo a tornar o uso das grades transparente para o usuário.

O Capítulo 2 fornece conceitos básicos necessários para a compreensão da Tese: a definição de grade é apresentada; o funcionamento de um escalonador de tarefas é descrito; três escalonadores de tarefas que são utilizados na análise de desempenho dos novos

escalonadores são resumidos; a importância de implementar mecanismos para lidar com as incertezas é destacada, através de um exemplo numérico, e propostas existentes para lidar com as incertezas são resumidas. Os pontos negativos das ferramentas existentes são utilizados como motivação para os escalonadores propostos. A importância de considerar o estado da rede é apresentada, a fim de servir de motivação para o foco dado nos experimentos realizados nos capítulos seguintes.

O Capítulo 3 apresenta o escalonador IPDT-FUZZY, um escalonador de tarefas robusto às incertezas na descrição de aplicações formadas por tarefas dependentes e descritas por DAGs (*Directed Acyclic Graphs* – Grafos acíclicos orientados). O escalonador implementa uma busca aleatória guiada por programação inteira 0–1 que emprega técnicas de otimização fuzzy. Simulações são utilizadas para mostrar as vantagens do escalonador IPDT-FUZZY sobre o escalonador IPDT, que serve de base para a sua construção e que ignora as incertezas.

O Capítulo 4 apresenta o escalonador IP-FULL-FUZZY, um escalonador de tarefas robusto às incertezas na descrição de aplicações e na disponibilidade dos recursos. Assim como o escalonador IPDT-FUZZY, o escalonador IP-FULL-FUZZY é voltado para aplicações formadas por tarefas dependentes descritas por DAGs e implementa o mesmo tipo de busca aleatória guiada. Através de resultados derivados via simulações, observa-se as vantagens em utilizar o escalonador IP-FULL-FUZZY quando comparado a outros escalonadores, dentre eles, os escalonadores clássicos HEFT e CPOP [74], que assim como o escalonador IPDT, ignoram as incertezas.

O Capítulo 5 apresenta resultados decorrentes da comparação dos estimadores de largura de banda disponível `abget` [3] e `pathload` [43] do ponto de vista das métricas relevantes para as suas utilizações em grades. Medições em cenários emulados apontam vantagens na utilização do `pathload`, embora ele careça de algumas funcionalidades que podem melhorar o seu desempenho na estimativa da disponibilidade dos caminhos entre recursos de processamento de grades.

O Capítulo 6 finaliza a Tese com a apresentação das conclusões e com sugestões de trabalhos futuros.

Capítulo 2

Conceitos fundamentais

Este capítulo apresenta os conceitos básicos e os trabalhos relacionados necessários para a compreensão das contribuições desta Tese. A Seção 2.1 apresenta a definição de grade. A Seção 2.2 descreve o processo de escalonamento de tarefas e resume alguns escalonadores de tarefas previamente publicados. A Seção 2.3 discute a necessidade de haver monitoramento regular do estado da rede. A Seção 2.4 apresenta propostas existentes na literatura para lidar com as incertezas presentes em grades e ilustra a necessidade de haver um tratamento preventivo das incertezas. A Seção 2.5 apresenta um resumo que relaciona o conteúdo apresentado com os próximos capítulos da Tese.

2.1 Grades

No final dos anos 90 [34], Ian Foster e Carl Kesselman empregaram o termo "grade" como referência para uma nova abordagem utilizada na solução de problemas computacionais que possuem alta demanda de recursos. Até então, a principal solução empregada para lidar com esses problemas consistia na utilização de *clusters* ou supercomputadores. *Clusters* caracterizam-se por serem uma solução de baixo custo, enquanto supercomputadores são custosos e, tipicamente, concentram-se nos centros de processamento de dados de organizações com alto poder financeiro. Apesar do aparente sucesso na utilização de ambas abordagens, elas carecem de flexibilidade. Quando ocorre sobrecarga de recursos, a ampliação nas capacidades dos ambientes não é trivial de ser implementada. Novos nós precisam ser adicionados ao *cluster* e novos dispositivos instalados nos supercomputadores. Não é, também, trivial o aproveitamento desses recursos para outras atividades, caso estes sejam subutilizados, dado que os equipamentos são fisicamente isolados e não podem mudar de função, como, por exemplo, de nó de processamento a computador pessoal. A solução para a utilização eficiente de recursos consiste, portanto, no fornecimento de capacidades computacionais sob demanda e em função dos requisitos das aplicações

submetidas.

Com os avanços nas tecnologias de comunicação, os enlaces de longa distância tiveram suas taxas de transmissão ampliadas, bem como houve decréscimo significativo nas probabilidades de perdas de informação. Essas melhorias viabilizaram a construção de ambientes flexíveis de alto desempenho computacional através da interligação de recursos espalhados geograficamente por diversas organizações. Os ambientes (*clusters* e supercomputadores) antes confinados em uma única rede local se expandiram, permitindo que organizações compartilhassem seus recursos e construíssem organizações virtuais para a solução de seus problemas computacionais [21]; surgiram, assim, as grades. Dispositivos localizados em organizações passaram a ser compartilhados e, em contrapartida, essas organizações passaram a ter acesso a mais recursos de processamento e armazenamento. Tal compartilhamento possibilitou o avanço de diversas áreas do conhecimento como física, química e astronomia [84] [96] [85] [36], levando à criação do termo "*e-Science*" [88], que designa as pesquisas que se beneficiam da infraestrutura computacional de alto desempenho emergente.

A utilização do termo "grade", que já fora utilizado para descrever a infraestrutura do sistema elétrico, não foi uma mera coincidência. Os cientistas e engenheiros buscam criar infraestruturas de computação tão confiáveis, ubíquas e transparentes para o usuário quanto as infraestruturas de eletricidade disponíveis ao redor do mundo.

A definição de grade sofreu diversas mudanças devido aos avanços nas tecnologias desde o final dos anos 90. Uma definição coerente nos dias de hoje é apresentada por Ian Foster [32]: grade é um sistema que, através da utilização de *interfaces* e protocolos padronizados, abertos e de propósito geral, coordena recursos que não estão sujeitos a um controle centralizado com o objetivo de forneces qualidades de serviçio não-triviais. A Figura 2.1 ilustra vários recursos compartilhados em uma possível configuração de grade. Observa-se a presença de recursos de diversos tipos, que variam desde computadores pessoais a até supercomputadores dedicados à execução de aplicações com alta demanda por processamento. Na realidade, uma grade pode agregar recursos de vários *clusters* isolados e até mesmo de redes inteiras, com recursos de processamento, armazenamento e instrumentos para processamento de aplicações específicas. Enlaces de rede de longa distância interligam os diversos recursos, criando uma organização virtual na qual as aplicações são executadas.

A utilização de dispositivos heterogêneos é viabilizada pelo uso de uma camada de software, denominada middleware, que abstrai os detalhes dos sistemas operacionais e da rede para as aplicações [16]. Os middlewares implementam serviços para atender as necessidades comuns de todas as aplicações, como a alocação de recursos e a transferência de dados.

Nas grades, os problemas de sobrecarga e subutilização de recursos são resolvidos

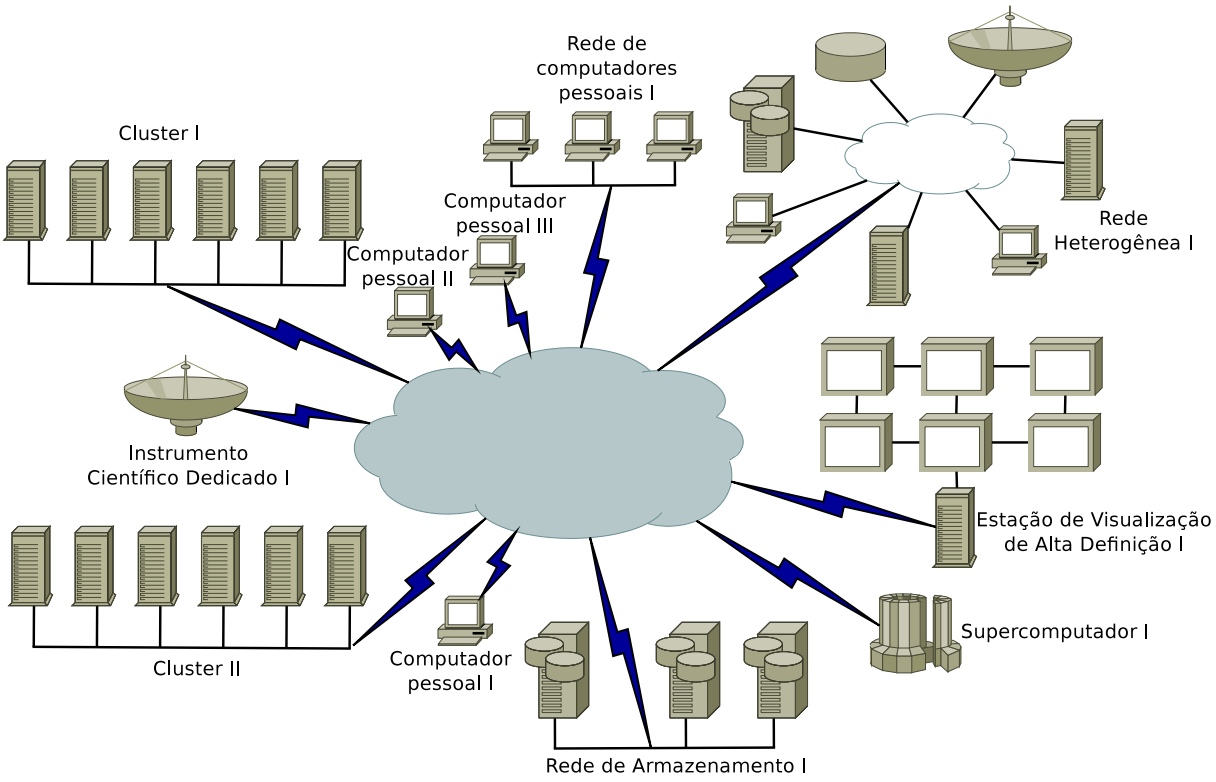


Figura 2.1: Exemplo de recursos agregados em uma grade

através da entrada e da saída de recursos, sem intervenção física, para atender os requisitos das aplicações; entretanto, pelo fato dos recursos das grades serem distribuídos e pertencerem a diversos domínios administrativos, várias questões, algumas delas inexistentes em sistemas paralelos convencionais, devem ser endereçadas, tais como:

- Alocação dos recursos para as aplicações: a busca pelo escalonamento ótimo é um problema $\mathcal{NP}$-difícil;
- Importância da rede: a rede representa um papel fundamental na grade, pois constitui o barramento do sistema virtual;
- Flutuação no estado dos recursos: como os recursos não são dedicados às aplicações, eles entram e saem da grade sem aviso prévio, o que acarreta em variações frequentes na capacidade disponibilizada para as aplicações;
- Incerteza na descrição das aplicações: aplicações executadas pela primeira vez, aplicações com erros ou usuários sem conhecimento da aplicação podem levar a descrições de requisitos diferentes daqueles realmente utilizados;

- Incerteza na descrição do estado da grade: ferramentas de monitoramento precisam ser pouco intrusivas e ter noção da dinâmica da grade, para fornecerem estimativas de disponibilidade dos recursos próximas dos valores reais, durante um intervalo de tempo aceitável para as aplicações.

Todos esses pontos têm sido alvo de estudos por vários pesquisadores, com destaque para os abordados pelo *Open Grid Forum* (OGF) [93], uma comunidade internacional voltada para a discussão de tópicos relacionados à utilização das grades na indústria e na academia. Merecem destaque, também, os pesquisadores envolvidos com o projeto do Globus *Toolkit* [33], um projeto voltado para a construção de um middleware para grades. Ambos os grupos de pesquisadores trabalham com o objetivo de definir padrões a serem adotados pela comunidade que desenvolve, utiliza e gerencia aplicações para grades. A definição de padrões para todos os pontos listados anteriormente, entretanto, ainda está longe de ser alcançada.

Apesar de vários argumentos e resultados isolados em favor das grades, evidências mais concretas das suas vantagens são necessárias para ampliar as suas utilizações. Essas provas têm surgido com os projetos que tiram proveito das soluções desenvolvidas para grades nos últimos anos. Dois exemplos de projetos são a grade de computação do LHC (*LHC Computing Grid* – LCG) [96] e o Folding@Home [99]. O LHC, sigla para *Large Hadron Collider*, é um acelerador de partículas sob responsabilidade da Organização Europeia para Investigação Nuclear (*Conseil Européen pour la Recherche Nucléaire* – CERN). O LHC é o maior equipamento científico do mundo e a infraestrutura computacional que mantém o seu funcionamento foi projetada para armazenar, analisar e transferir cerca de 15PetaBytes de dados por ano, o que equivale a 1% da taxa mundial de produção de informação. Outras aplicações de *e-Science* com necessidades de transferência de dados na ordem de PetaBytes motivam também o uso de grades [36]. O CERN dificilmente conseguiria lidar com essa quantidade de dados, se não fosse utilizada uma grade. Várias organizações espalhadas pelo mundo cedem recursos para auxiliar no processamento e no armazenamento dos dados; em troca, as organizações têm acesso às informações geradas pelo LHC, podendo utilizá-las para suas próprias pesquisas em física de partículas.

O Folding@Home é um exemplo de projeto que utiliza ciclos ociosos de equipamentos pessoais para a realização de simulações que consomem muita capacidade de processamento. O projeto tem como objetivo simular o processo químico de enrolamento de proteínas através de processos cliente, que podem ser executados tanto em computadores pessoais quanto em videogames. Cada cliente copia uma tarefa do servidor do projeto pela Internet, processa localmente e envia o resultado de volta também pela Internet. Em Janeiro de 2010, as estatísticas disponibilizadas em [99] mostraram que a união de todos os dispositivos pessoais ao redor do mundo tem levado a uma capacidade agregada de processamento de 4,7PetaFLOPS (*FLoating point Operations Per Second* – Operações de

ponto flutuante por segundo). A título de comparação, o supercomputador mais rápido do mundo, segundo a lista TOP500, em novembro de 2009 [95], tinha uma capacidade máxima de 2,3PetaFLOPS.

Assim como ocorreu com a Internet, as tecnologias desenvolvidas para resolver os problemas em grades começam a sair do meio acadêmico e a se espalhar pelos mais diversos setores da sociedade. Isso pode ser observado com a utilização, cada vez maior, do termo "computação em nuvem". Ele tem sido usado por diversos provedores de serviços na Internet para descrever soluções computacionais que são executadas remotamente ao usuário. O processamento e o armazenamento de dados dessas soluções são realizados na nuvem, de forma transparente, graças a mecanismos e protocolos propostos para resolver problemas em grades computacionais, conforme observado em [41].

2.2 Escalonamento de tarefas em grades

Um dos primeiros passos realizados no processo de execução de aplicações em grades é o escalonamento das tarefas que compõem essas aplicações. A fim de tirar proveito do paralelismo possibilitado pelos recursos que formam as grades, as aplicações são decompostas em tarefas, cabendo ao escalonador definir o escalonamento, ou seja, o recurso que será utilizado por cada tarefa e o intervalo de tempo no qual ele será utilizado.

Figura 2.2: Entradas e saída de um escalonador de tarefas genérico para grades

A Figura 2.2 apresenta um diagrama que descreve um escalonador de tarefas em alto nível. Ele recebe como entrada a descrição da aplicação submetida à grade, o estado dos recursos compartilhados e o objetivo a ser alcançado com o escalonamento. A saída do escalonador é o escalonamento das tarefas, que é utilizado para realizar a alocação dos recursos da grade. O ideal é que o escalonador devolva como saída o melhor recurso no qual cada tarefa deve ser executada e o melhor intervalo de tempo em que a execução deve ser realizada, a fim de alcançar um objetivo específico. No entanto, quando o objetivo é minimizar o tempo de execução da aplicação, mais conhecido como *makespan*, a busca pelo escalonamento ótimo reduz-se a um problema $\mathcal{NP}$-difícil [60]. O *makespan* costuma ser utilizado de forma indireta para avaliar a qualidade de um dado escalonamento. Costuma-se utilizar o *speedup*, uma métrica que compara o tempo de execução serial da aplicação

(T_{max}) com o tempo de execução que a aplicação teria caso utilizasse aquele escalonamento (T). Para encontrar o *speedup* de um dado escalonamento basta calcular $\frac{T_{max}}{T}$. Quanto menor o *makespan*, maior o *speedup* e, consequentemente, melhor é o escalonamento [1].

Como não há garantias de que as disponibilidades dos recursos permanecerão constantes com o passar do tempo, um escalonamento deve ser derivado o mais rápido possível, a fim de se evitar que as informações sobre a disponibilidade dos recursos, recebidas como entrada, tornem-se inválidas. A necessidade por uma resposta rápida aliada com o fato do problema ser $\mathcal{NP}$-difícil justifica a implementação de escalonadores baseados em heurísticas ou em buscas aleatórias guiadas por alguma técnica de otimização [74]. Como será discutido no decorrer desta Tese, além de serem rápidos e de devolverem escalonamentos próximos do ótimo, escalonadores de tarefas devem, também, lidar com as incertezas presentes nas descrições das aplicações e nas informações sobre disponibilidade dos recursos.

As tarefas que compõem uma aplicação podem ter dependências entre si. Quando as dependências existem, as tarefas formam grafos direcionados para representar as dependências. Quando as dependências não existem, as tarefas formam grafos vazios, ou seja, grafos que não possuem arestas, e são mais conhecidas como *Bag-of-Tasks* (BoT). A descrição da aplicação passada como entrada para o escalonador de tarefas depende do tipo de aplicação. A Figura 2.3 exemplifica o grafo direcionado de uma aplicação na qual há dependências entre as tarefas, enquanto que a Figura 2.4 exemplifica o grafo vazio de uma aplicação BoT.

Os vértices representam as tarefas e os seus rótulos identificam as tarefas tanto na Figura 2.3 quanto na Figura 2.4. Os pesos (valores entre colchetes) representam os pesos computacionais das tarefas em quantidade de instruções. O tempo de execução não pode ser usado como peso computacional porque ele vai depender do recurso que for alocado para executar a tarefa. Na Figura 2.3, os arcos representam as dependências entre as tarefas. Os pesos dos arcos representam os pesos de comunicação das dependências em quantidade de bits. A existência de um arco da tarefa T_i para a tarefa T_j implica que a tarefa T_j só pode iniciar a sua execução após a tarefa T_i terminar de executar e de enviar os dados de dependência para T_j.

O estado dos recursos informado como entrada para o escalonador pode, também, ser representado por um grafo, o que é exemplificado na Figura 2.5.

Os vértices do grafo na Figura 2.5 representam os dispositivos de processamento da grade e os seus rótulos identificam cada dispositivo. As arestas representam a existência de um caminho entre dois dispositivos. Os pesos dos vértices representam o inverso da capacidade de processamento disponível nos computadores, enquanto os pesos das arestas

[1] a partir deste ponto a referência ao "*speedup* do escalonamento produzido pelo escalonador" será apresentada como o "*speedup* do escalonador" ou o "*speedup* produzido pelo escalonador"

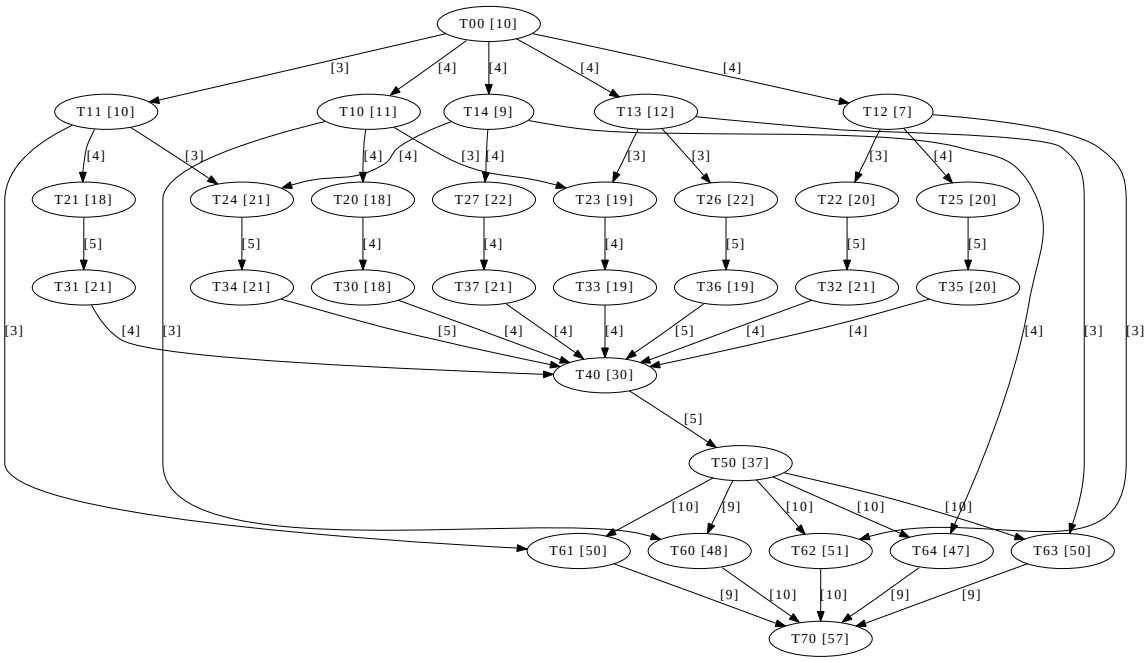


Figura 2.3: Exemplo da descrição de uma aplicação com tarefas dependentes

representam o inverso da largura de banda disponível. O tempo esperado para a execução de uma tarefa ou para a transferência de dados de dependência pode ser encontrado através da multiplicação do peso informado no grafo da aplicação pelo peso informado no grafo dos recursos. É importante observar que dispositivos de processamento equipados com múltiplos processadores, ou com processadores que possuam múltiplos núcleos, podem ter essas características representadas no grafo através da criação de um vértice para cada elemento de processamento e de arestas de peso zero conectando todos esses vértices entre si.

Nesta Tese, são propostos escalonadores voltados para aplicações de *e-Science* formadas por tarefas dependentes e que resolvem problemas de áreas como astronomia e química. Todo o trabalho realizado foca em aplicações formadas por tarefas dependentes descritas por DAGs. Exemplos de aplicações descritas por DAGs incluem visualização de imagens e simulações dinâmicas de moléculas. Embora os escalonadores sejam voltados para aplicações descritas por DAGs, é possível utilizá-los nos escalonamentos de aplicações BoT, adicionando-se duas tarefas artificiais com peso zero ao grafo. A primeira deve ser predecessora de todas as tarefas originais da aplicação e a segunda deve ser sucessora. As aplicações descritas por DAGs consideradas nesta Tese são formadas por tarefas que transferem muitos dados entre si e, consequentemente, os pesos dos arcos dos DAGs e os pesos das arestas do grafo dos recursos não podem ser ignorados. O objetivo a ser alcançado pelos escalonamentos é a minimização do *makespan* das aplicações.

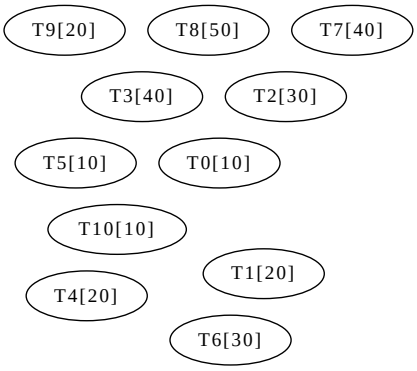


Figura 2.4: Exemplo da descrição de uma aplicação com tarefas independentes

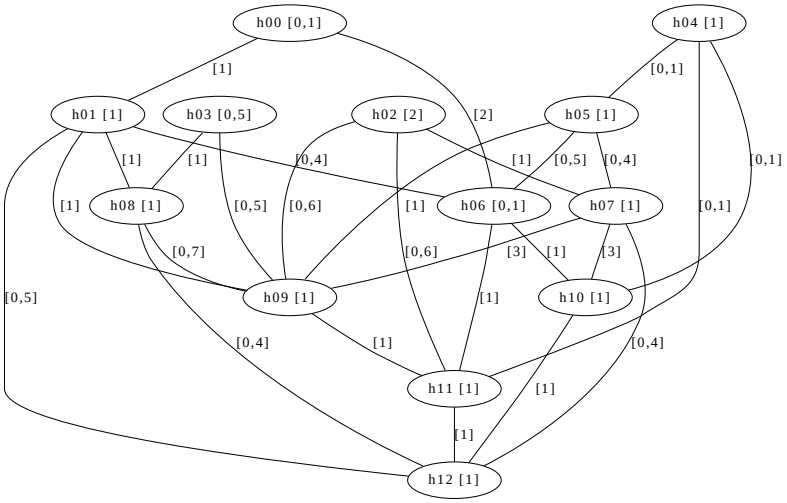


Figura 2.5: Exemplo do estado dos recursos de uma grade

Apesar do diagrama descrito na Figura 2.2 considerar que a descrição de uma única aplicação é informada como entrada para o escalonador, no caso de DAGs, é possível agregar várias descrições em uma só. Conforme provado em [81], quando o objetivo a ser alcançado com o escalonamento é a minimização do tempo de execução e caso hajam várias aplicações que tenham sido submetidas simultaneamente à grade, basta interligar todos os DAGs, como se cada um fosse uma única tarefa, da mesma forma descrita para transformar uma aplicação BoT em um DAG.

As subseções seguintes apresentam exemplos de escalonadores e de técnicas utilizadas para escalonar tarefas em grades. A Subseção 2.2.1 resume os escalonadores HEFT e CPOP, dois escalonadores baseados em heurísticas que são, usualmente, utilizados para a comparação de novas propostas de escalonamento. A Subseção 2.2.2 resume o escalonador IPDT, um escalonador de tarefas que resolve um problema de programação inteira e que

serviu de base para a construção dos novos escalonadores de tarefas propostos nesta Tese.

2.2.1 Os escalonadores de tarefas HEFT e CPOP

Em [74], foram introduzidos os escalonadores HEFT (*Heterogeneous-Earliest-Finish-Time*) e CPOP (*Critical-Path-on-a-Processor*). Ambos são baseados em heurísticas, tem como objetivo minimizar o *makespan* de aplicações submetidas a sistemas paralelos heterogêneos, e escalonam tarefas uma a uma conforme uma lista de prioridade. Os escalonadores CPOP e HEFT demandam um baixo tempo de execução, dado que são baseados em heurísticas.

No HEFT, a prioridade das tarefas é diretamente proporcional ao tamanho do caminho de cada uma até a tarefa de saída do DAG, enquanto que na ordenação das tarefas no CPOP, utiliza-se tanto o maior caminho até a tarefa de saída do DAG quanto o maior caminho da tarefa de entrada até a tarefa em questão. Além disso, o CPOP escalona as tarefas do caminho crítico do DAG no computador que minimiza o tempo de execução sequencial de todas elas (o caminho crítico é o maior caminho no DAG, somando os pesos das tarefas e os pesos dos arcos, da tarefa de entrada até a tarefa de saída). O processo de seleção do computador e do intervalo de tempo de execução para cada tarefa, tanto no CPOP quanto no HEFT, utiliza um algoritmo que busca inserir as tarefas nos intervalos de ociosidade dos computadores, de modo que os instantes da finalização das tarefas sejam minimizados.

É importante observar que um pré-requisito para a utilização dos escalonadores HEFT e CPOP é que todos os computadores tenham enlaces entre si. Para utilizá-los em outras grades, é necessário considerar que computadores sem enlaces entre si possuem enlaces virtuais que demandam uma duração infinita (∞), para transmitir 1 bit.

Apesar de ambos os escalonadores apresentados em [74] não serem orientados para ambientes com incertezas, eles são utilizados nesta Tese, tanto para mostrar o impacto negativo de não considerar incertezas (Subseção 2.4.2), quanto para servir de referência na avaliação de desempenho (Seção 4.3), dado que a utilização desses escalonadores, como referência de comparação, é bastante comum na literatura [66] [81] [63] [79] [48].

2.2.2 O escalonador de tarefas IPDT

Em [12], apresenta-se o escalonador IPDT (*Integer Programming with a Discrete Time*), que é baseado em busca aleatória guiada por um problema de programação inteira 0–1. O escalonador recebe dois grafos como entrada. O grafo $H = (V_H, A_H)$ representa a topologia formada pelos recursos da grade com m computadores e o DAG $D = (V_D, A_D)$ representa as dependências entre as n tarefas da aplicação a ser escalonada. A ordem crescente dos índices das tarefas representa uma ordem topológica dos vértices do DAG.

Cada tarefa i possui uma quantidade de instruções I_i ($I_i \in \mathbb{R}_+$) e cada dependência de dados entre uma tarefa i e uma tarefa j possui uma quantidade de bits $B_{i,j}$ ($B_{i,j} \in \mathbb{R}_+$).

Cada computador k possui a sua capacidade de processamento disponível, TI_k, representada em termos do intervalo de tempo que o computador leva para executar 1 instrução ($TI_k \in \mathbb{R}_+$). A largura de banda disponível entre os computadores k e l é representada por $TB_{k,l}$, que é o intervalo de tempo necessário para se transferir 1 bit entre o computador k e o computador l ($TB_{k,l} \in \mathbb{R}_+$). O conjunto de computadores alcançáveis pelo computador k é descrito por $\delta(k)$.

O escalonamento encontrado pelo escalonador IPDT é dado pelos valores das variáveis $x_{i,t,k}$ ($\in \{0,1\}$); $x_{i,t,k}$ é uma variável binária que vale 1 se, e somente se, a tarefa i terminar sua execução no instante de tempo t e no computador k.

O objetivo do escalonador é minimizar o tempo de execução da aplicação, que coincide com o tempo de finalização de execução da última tarefa do DAG, a n-ésima tarefa. Por conveniência, na formulação do problema resolvido pelo IPDT exibida a seguir, utiliza-se a notação $\mathcal{T} = \{1, \ldots, T_{max}\}$ para representar a linha do tempo discreta, onde T_{max} é o intervalo de tempo que a aplicação levaria para executar caso todas suas tarefas executassem serialmente no computador mais rápido da grade, i.e., $T_{max} = \min(\{\, TI_{k|k \in V_H} \,\}) \times \sum_{i=1}^{n} I_i$. O escalonador IPDT resolve o seguinte problema de programação inteira:

$$\text{Minimize} \sum_{t\in\mathcal{T}}\sum_{k\in\mathcal{H}} t\, x_{n,t,k}$$

sujeito a

$$\sum_{t\in\mathcal{T}}\sum_{k\in V_H} x_{j,t,k} = 1 \qquad \text{para } j \in V_D; \qquad \text{(D1)}$$

$$x_{j,t,k} = 0 \qquad \text{para } j \in V_D, \quad k \in V_H, \quad t \in \{1,\ldots,\lceil I_j\, TI_k\rceil\}; \qquad \text{(D2)}$$

$$\sum_{k\in\delta(l)}\sum_{s=1}^{\lceil t-I_j\, TI_l-B_{i,j}\, TB_{k,l}\rceil} x_{i,s,k} \geq \sum_{s=1}^{t} x_{j,s,l} \quad \text{para } j \in V_D, \quad ij \in A_D, \quad \text{para } l \in V_H, \quad t \in \mathcal{T}; \qquad \text{(D3)}$$

$$\sum_{j\in V_D}\sum_{s=t}^{\lceil t+I_j\, TI_k-1\rceil} x_{j,s,k} \leq 1 \qquad \text{para } k \in V_H, \quad t \in \mathcal{T}, \quad t \leq \lceil T_{max} - I_j\, TI_k\rceil; \qquad \text{(D4)}$$

$$x_{j,t,k} \in \{0,1\} \qquad \text{para } j \in V_D, \quad l \in V_H, \quad t \in \mathcal{T}. \qquad \text{(D5)}$$

A restrição (D1) garante que cada tarefa do DAG executa em um único computador e finaliza a sua execução em um único instante de tempo. A restrição (D2) garante que uma tarefa finaliza sua execução em um instante de tempo suficiente para que todas suas instruções tenham sido executadas. A restrição (D3) define que a j-ésima tarefa só inicia a sua execução após receber os dados de dependência de suas predecessoras. A restrição (D4) garante que cada computador executa no máximo 1 tarefa, em qualquer instante de tempo, e a restrição (D5) define o domínio das variáveis $x_{j,t,k}$.

O fato do escalonador IPDT considerar que a linha do tempo de execução é discreta é uma forma de obter tempos de execução aceitáveis para o escalonador, dado que a busca pelo escalonamento ótimo é um problema $\mathcal{NP}$-difícil. A experiência positiva com a utilização do escalonador IPDT na produção de altos valores de *speedup* [12] motivaram o uso do mesmo como base para os escalonadores IPDT-FUZZY e IP-FULL-FUZZY, que serão apresentados respectivamente no Capítulo 3 e no Capítulo 4.

2.3 Monitoramento de recursos em grades

Para que os escalonadores de tarefas proponham bons escalonamentos é necessário que haja conhecimento da disponibilidade dos recursos das grades, para que se possa avaliar qual o melhor computador para executar uma tarefa. Embora a capacidade de processamento disponível nos computadores tenha um papel importante na decisão de escalonamento, os parâmetros relacionados com a rede, principalmente a largura de banda disponível, não podem ser ignorados. É necessário monitorar a disponibilidade de largura de banda nos caminhos para se obter escalonamentos eficientes.

Além de crítico para a obtenção de informações precisas a serem utilizadas no escalonamento de tarefas, o monitoramento exerce papel importante em sistemas que reagem às flutuações da disponibilidade dos recursos [12] [39]. A motivação para a utilização de tais sistemas vem do fato dos recursos das grades não serem exclusivos para as aplicações em execução. Nestes sistemas, os recursos são monitorados e, se uma alocação de recursos diferente da estabelecida fornecer ganhos para as aplicações em execução, tarefas podem ser migradas. Ter estimativas precisas da disponibilidade de recursos é de fundamental importância para avaliar os ganhos que possam ser alcançados com a migração de tarefas.

Os métodos empregadas por monitores de recursos em grades não são ideais, ou seja, eles não fornecem as estimativas precisas a cerca da disponibilidade de recursos. Esta Tese propõe que escalonadores de tarefas recebam como entrada as incertezas referentes às estimativas dos monitores de recursos e proponham escalonamentos que levem em consideração o impacto dessas incertezas, em especial das incertezas relacionadas com a disponibilidade de largura de banda.

As subseções seguintes apresentam informações relacionadas com a estimativa de largura de banda disponível em grades. A Subseção 2.3.1 discute a importância de se considerar as estimativas de disponibilidade de largura de banda para o funcionamento das grades e a Subseção 2.3.2 resume as métricas relacionadas com a estimação de largura de banda.

2.3.1 Importância do monitoramento da largura de banda disponível

Exemplos que reforçam que a largura de banda disponível deve ser considerada nas decisões de escalonamento e migração para aplicações em grades são encontrados em [70] e em [42]. Em [70], são apresentadas medições em enlaces de rede de uma grade construída sobre uma rede de pesquisa nos Estados Unidos. Durante a execução de aplicações na grade, observa-se que a utilização destes enlaces aumenta de cinco a oito vezes com relação à sua utilização em um dia normal. Escalonar tarefas de aplicações como as

exemplificadas em [70] em computadores interligados por enlaces saturados ou com baixa disponibilidade podem levar a insatisfação tanto de usuários das grades quanto de usuários que estejam compartilhando os recursos de comunicação com a grade. Ter estimativas da disponibilidade da rede, portanto, é de suma importância.

Em [42], apresenta-se uma proposta de ajuste automático de parâmetros do protocolo `GridFTP` para se alcançar a utilização máxima dos enlaces de grades e, dessa forma, diminuir os tempos de execução das aplicações (O `GridFTP` é utilizado para transferência de dados em grades construídas com o middleware `Globus`). Pelos resultados obtidos, mostra-se que os ajustes ideais dos parâmetros do protocolo estão diretamente relacionados com a estimativa de largura de banda disponível dos enlaces, bem como com o produto banda $\times$ atraso entre os nós da grade.

Relatórios baseados em medições feitas em grades reais e em previsões da utilização de grades disponibilizados em [47], [57] e [31] enfatizam a relevância de se considerar a largura de banda para aumentar o desempenho das grades. Em [47], são apresentadas estatísticas referentes ao uso da rede em diversas grades ao redor do mundo. Boa parte das aplicações executadas nessas grades transfere uma quantidade de dados que não pode ser ignorada no escalonamento de tarefas (55% transferem de 1 a 100MB e 18% transferem pelo menos 10GB). Em [57], são apresentados argumentos que levam a previsões de que, dentro de 5 a 10 anos, as aplicações executadas em grades necessitarão de largura de banda disponível da ordem de Terabits por segundo, para que requisitos de qualidade de serviço sejam atendidos. A importância da disponibilidade dos recursos da rede pode também ser inferida pelo estudo em [31], que relata que ganhos podem ser alcançados quando se utiliza a vazão como parâmetro para otimizar o desempenho da grade.

2.3.2 Métricas relacionadas com largura de banda

A estimação das características de transmissão de uma rede de computadores está associada a diversas métricas, tais como a capacidade nominal do enlace, a capacidade do gargalo de um caminho, a largura de banda disponível em um caminho e a capacidade máxima de transferência (*Bulk Transfer Capacity* – BTC) entre dois computadores [61].

A Figura 2.6 ilustra as várias métricas relacionadas com a largura de banda. A figura exibe dois computadores, "Computador 1" e "Computador 2", interligados por um caminho de rede formado por 3 enlaces/saltos. Cada enlace é simbolizado por um retângulo em que a parte cinzenta representa a capacidade ocupada do enlace e a parte em branco representa a largura de banda disponível. Neste exemplo, as capacidades nominais dos enlaces são $C1$, $C2$ e $C3$, a capacidade do gargalo do caminho é $C1$ e a largura de banda disponível fim a fim é $D3$. A capacidade nominal representa o máximo que cada enlace consegue transmitir, caso ele fosse utilizado para a transferência de dados de uma única

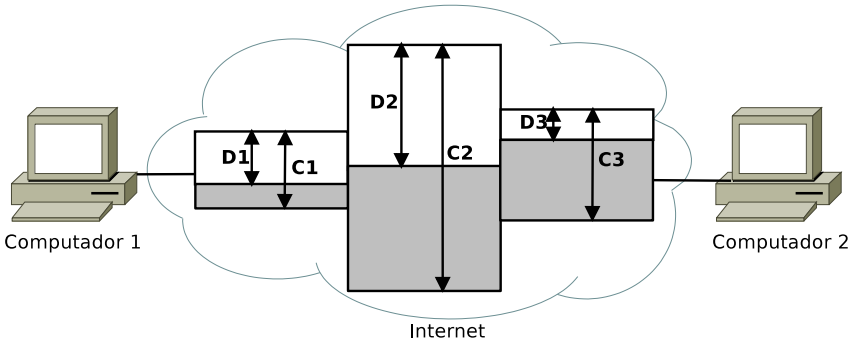


Figura 2.6: Métricas relacionadas com largura de banda (Baseada em figura de [61])

conexão sem custo extra e sem perdas. A capacidade do gargalo representa a menor capacidade nominal do caminho e a largura de banda disponível representa o máximo que um caminho pode transferir, em um certo instante de tempo.

A BTC é a vazão máxima obtida por uma conexão TCP no caminho. A BTC não pode ser representada na Figura 2.6, pois esta métrica depende do tipo de tráfego concorrente que esteja ocupando os enlaces do caminho no momento das medições. A dependência do tipo de tráfego concorrente deve-se ao algoritmo de controle de congestionamento do protocolo TCP, que leva a um compartilhamento igualitário da capacidade dos enlaces entre os fluxos TCP existentes. Por exemplo, se em um caminho a largura de banda disponível é zero mas o tráfego concorrente consiste de um único fluxo TCP, uma ferramenta ideal para medir a BTC informaria um valor igual à metade da capacidade do enlace.

As métricas relacionadas com a capacidade são úteis para estudos de planejamento de capacidade. No caso de grades, a utilização dessas métricas permite a estimação do desempenho do sistema em função das demandas das aplicações antes mesmo das grades entrarem em operação. Para os estudos relacionados com escalonamento de tarefas, a métrica mais útil é a largura de banda disponível, dado que ela representa a disponibilidade dos recursos de rede para as aplicações, informação necessária para se estimar os tempos das transferências de dados entre tarefas.

2.4 Procedimentos para lidar com informações incertas

A descrição das aplicações e as estimativas de disponibilidade dos recursos desempenham um papel fundamental para que os escalonamentos propostos sejam, na prática, próximos dos ótimos. Esses dois tipos de informação fornecidas como entrada para o escalonador de tarefas possuem, quase sempre, um certo grau de incerteza. Os usuários não tem total

conhecimento sobre as suas aplicações e as ferramentas de monitoramento, por melhor que sejam, não são 100% precisas.

Na literatura, as incertezas são expressas através da grandeza Qualidade da Informação (QoI – *Quality of Information*) [22]. No caso das descrições das aplicações, a QoI designa a porcentagem de certeza sobre as demandas das tarefas (os pesos dos vértices dos DAGs) e sobre as dependências entre elas (os pesos dos arcos dos DAGs). O valor de 100% corresponde a uma aplicação descrita corretamente.

Uma solução para diminuir o impacto causado pelas incertezas na descrição das aplicações seria a criação de perfis, utilizando-se programas como o GNU `gprof` [86]. O problema dessa solução é que as informações referentes ao desempenho de um código em uma plataforma não são válidas para o mesmo código compilado para outra plataforma, o que limita a sua utilização em ambientes heterogêneos como as grades. Uma proposta mais avançada seria armazenar as informações sobre os tempos de execução das aplicações submetidas à grade e, com base nesse histórico, utilizar a técnica *Case-Based Reasoning* (CBR) [56]. Esta técnica avalia as demandas de uma aplicação submetida à grade e busca no histórico similaridades com aplicações que já tenham sido executadas. Desta forma, é possível predizer a carga de trabalho a ser gerada pela aplicação atual, entretanto, a comparação realizada não considera que as informações da aplicação atual possam estar incorretas, o que diminui a sua utilidade em situações com alto grau de incerteza na descrição das aplicações.

Para diminuir o impacto causado pelas incertezas sobre a disponibilidade dos recursos da grade, costuma-se empregar arcabouços (*frameworks*) baseados em medição, reescalonamento e migração de tarefas. Um escalonamento inicial é proposto para a aplicação e, caso seja detectada alguma diferença entre o desempenho real da aplicação e o desempenho esperado, em tempo de execução, um novo escalonamento é proposto e tarefas são migradas. A maioria dos arcabouços considera que a diferença entre o desempenho real e o desempenho esperado é decorrente da incerteza inerente aos procedimentos de estimação da disponibilidade dos recursos, entretanto, deve-se, também, considerar a descrição incorreta da aplicação como causa da diferença.

Nota-se que tanto as soluções baseadas na análise dos históricos quanto as soluções baseadas em monitoramento e migração tem como consequência o aumento na complexidade dos sistemas de gerência da grade, dado que é necessário implementar e integrar novos mecanismos ao escalonador de tarefas. Uma solução mais simples consistiria em adicionar o suporte às incertezas diretamente nos escalonadores, solução que é adotada na implementação dos escalonadores propostos nesta Tese.

As próximas subseções fornecem argumentos a favor do tratamento das incertezas diretamente nos escalonadores de tarefas. A Subseção 2.4.1 resume alguns arcabouços que implementam técnicas para diminuir os efeitos negativos das flutuações e das incertezas

presentes durante a execução de aplicações em grades. A Subseção 2.4.2 ilustra, através de um exemplo, o impacto negativo causado pelas incertezas e a Subseção 2.4.3 discute a utilização de técnicas de otimização fuzzy como uma forma de lidar com as incertezas. Além disso, ela, também, esclarece que as técnicas de otimização fuzzy diferem de métodos que consideram as demandas das aplicações e as disponibilidades dos recursos como números aleatórios.

2.4.1 Autoajuste da alocação de recursos nos arcabouços teóricos existentes

Uma opção para lidar com a dinâmica e a incerteza nas grades é a implementação de procedimentos que autoajustem a alocação dos recursos com o objetivo de evitar a subutilização dos recursos compartilhados e garantir o cumprimento dos requisitos das aplicações. Em [12], descreve-se um procedimento com esse objetivo. O procedimento é composto de passos que estão presentes na maioria dos sistemas de alocação de recursos em grades. A Figura 2.7 ilustra como seria a execução ideal de uma aplicação com o procedimento em funcionamento. O usuário submete a aplicação à grade, uma "imagem" atual da grade é gerada e as tarefas são enviadas para os melhores recursos, de modo a atender o objetivo do usuário. Depois do início da execução, entra-se em um processo cíclico de monitoramento e modificações na alocação dos recursos, a fim de manter o desempenho da aplicação estável e próximo do ótimo, independente das variações no estado da grade ou das informações incorretas que tenham sido passadas na descrição da aplicação. Uma boa parte dos mecanismos de autoajuste modifica as alocações através de migração de tarefas. É possível, ainda, implementar métodos que modifiquem a forma como os recursos estão interligados ou agreguem novos recursos. Ao término da execução, o resultado é enviado para o usuário.

Mecanismos de autoajuste não são as únicas abordagens encontradas na literatura para lidar com a dinâmica das grades. Outras opções são os escalonadores dinâmicos e os escalonadores adaptativos [2]. Escalonadores dinâmicos escalonam as tarefas de uma aplicação por etapas, à medida que as demandas vão sendo descobertas [50]. A tarefa inicial é escalonada e tem sua execução iniciada. Somente após chegar a um ponto onde sejam conhecidas as demandas das suas sucessoras, elas são escalonadas, e assim por diante, até que todas as tarefas da aplicação tenham sido executadas. Esses escalonadores são empregados para escalonar aplicações que possuam demandas que só são conhecidas em tempo de execução. No caso de aplicações descritas por DAGs, tal situação é repre-

[2]Por conta de divergências na nomenclatura encontrada na literatura é importante esclarecer que a definição de escalonamento dinâmico adotada nesta Tese é aquela apresentada em [50] e a definição de escalonamento adaptativo adotada é aquela apresentada em [40]

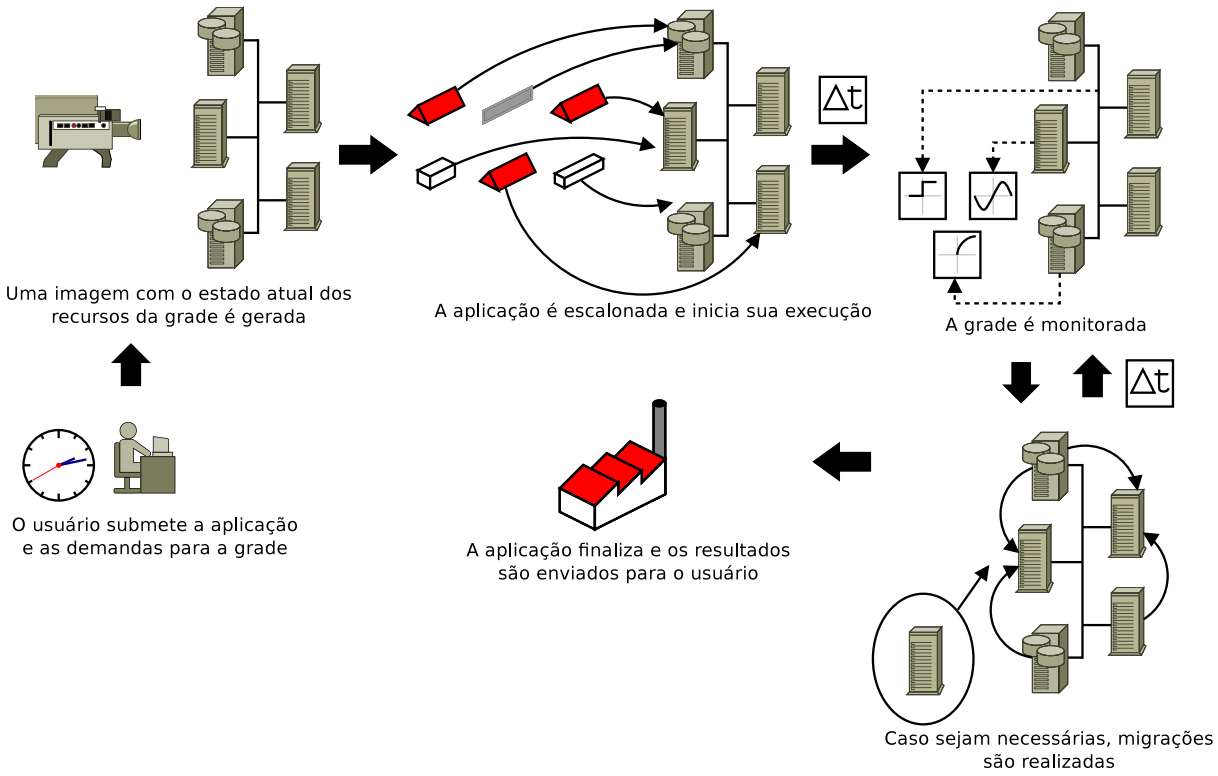


Figura 2.7: Execução ideal de aplicação em grade

sentada por pesos desconhecidos nos vértices e arcos, o que impede que um escalonamento seja proposto para todas as tarefas de uma única vez. Os pesos inicialmente desconhecidos são descobertos à medida que as tarefas iniciais são executadas, o que exige que as decisões de escalonamento sejam adiadas. Dessa forma, em cada etapa do escalonamento, o estado da grade pode ser atualizado, permitindo que o escalonamento seja realizado de acordo com a disponibilidade dos recursos da grade. Escalonadores adaptativos, também, fazem o escalonamento por etapas, porém, diferente dos escalonadores dinâmicos, eles têm conhecimento de todas as demandas das aplicações [40]. O propósito de realizar o escalonamento por etapas é adaptar o escalonamento à disponibilidade da grade, dado que, após o término de cada tarefa, o estado dos recursos pode ter sido modificado.

Embora tanto o escalonamento dinâmico quanto o escalonamento adaptativo levem em consideração a dinâmica na disponibilidade dos recursos, tal disponibilidade é verificada somente em instantes de tempo específicos. No caso do escalonamento dinâmico, esses instantes são aqueles quando as demandas desconhecidas são resolvidas. No caso do escalonamento adaptativo, esses instantes são aqueles quando novas tarefas são submetidas para serem escalonadas. Dessa forma, mudanças nos estados dos recursos que ocorram durante as execuções das tarefas são ignoradas, o que impede que a alocação dos recursos

seja modificada a fim de se adequar o escalonamento às flutuações inesperadas no estado da grade.

Vários mecanismos foram propostos desde o surgimento das grades na tentativa de autoajustar a alocação dos recursos. Alguns exemplos de mecanismos são encontrados em [78], [19], [39], [75], [1], [72], [71] e [18].

O *Network Weather Service* (NWS), apresentado em [78], é um serviço distribuído para previsão do desempenho de recursos de processamento e de rede. O objetivo do NWS é fornecer previsões precisas de características do desempenho dinâmico de um conjunto de recursos. Apesar do NWS não ser explicitamente voltado para grades, ele tem sido adotado como serviço de monitoramento e previsão para diversas propostas de escalonamento e de autoajuste de alocação de recursos. O NWS foca nas fases de monitoramento e previsão do estado dos recursos, portanto, ele não especifica técnicas a serem utilizadas nas fases de escalonamento ou reescalonamento de aplicações em grades. O NWS mede e prevê a taxa de processamento disponível, bem como o tempo necessário para estabelecer uma conexão TCP, a latência de rede TCP fim-a-fim e a BTC entre os computadores da grade. Vários modelos de previsão baseados em séries temporais são aplicados sobre os pares ("instante de tempo","valor medido")e os valores previstos são comparados com os valores medidos. A série que produzir o menor erro é utilizada para fazer previsões futuras. As técnicas empregadas para previsão são voltadas para aplicações que precisem de informações sobre o estado dos recursos em um intervalo de tempo com duração de dezenas de segundos a alguns minutos, denominadas em [78] como sendo aplicações de curta duração (*short-term*). Esse pequeno intervalo de tempo faz com que o NWS não tenha a visão do estado da grade em um futuro muito distante, o que pode gerar um escalonamento inicial ineficiente e que precisará de várias migrações para atender os requisitos de QoS de aplicações formadas por tarefas que levem horas para executar.

A *Grid Architecture for Computational Economy* (GRACE) [19] corresponde a uma arquitetura para regular a oferta e a procura de recursos em grades. Cada recurso tem um custo associado. O objetivo da GRACE é fornecer serviços que auxiliem os proprietários e os consumidores de recursos a maximizarem suas funções objetivo que, de forma geral, tendem a ser conflitantes. Enquanto os proprietários fornecem recursos e visam obter lucro com esse fornecimento, os consumidores buscam obter a QoS requisitada pelo menor custo possível. A GRACE define blocos de funções em alto nível, de modo que eles possam ser implementados independente dos tipos de recursos e aplicações executadas nas grades. Por conta dessa abordagem, a GRACE precisa que a grade esteja equipada com algum middleware que possa ser expandido. A expansão do middleware equivale à implementação dos blocos definidos na GRACE. Um dos blocos mais importantes da GRACE é o *resource broker*, responsável por escalonar as aplicações, detectar as mudanças

no estado da grade e tomar ações que diminuam o efeito negativo de mudanças críticas.

O objetivo do arcabouço para execução de aplicações em ambientes dinâmicos apresentado em [39] é dotar as grades de "inteligência" e autonomia que permitam que usuários executem aplicações no modo "submeter e esquecer". O arcabouço é apresentado como solução para adaptar a execução de aplicações através de migrações em situações que vão desde a diminuição na taxa de processamento disponível dos computadores até a decisão explícita do usuário. Os componentes do arcabouço são apresentados em alto nível, da mesma forma que é feito na GRACE, com o objetivo de permitir que o mesmo seja implementado independente das aplicações, da grade e do middleware utilizado.

Em [75], apresenta-se um arcabouço de autoajuste cujo objetivo é evitar perdas de desempenho em aplicações submetidas por usuários que desejem minimizar os seus tempos de execução. O arcabouço engloba técnicas de reescalonamento e migração de tarefas. O tempo de execução esperado para cada tarefa das aplicações é monitorado, de modo a evitar que uma tarefa próxima de finalizar sua execução seja migrada, já que quanto mais perto de finalizar sua execução, menor é a probabilidade dela obter ganhos migrando para outro recurso. O processo de submissão de aplicações no arcabouço envolve a criação de contratos com requisitos mínimos de qualidade, que devem ser fornecidos pela grade. Exemplos de requisitos de um contrato são quantidade mínima de computadores com um sistema operacional específico e enlaces de rede com um valor mínimo de largura de banda disponível.

O G-QoSM [1] é um arcabouço que provê classes de serviço semelhantes às classes da arquitetura de QoS da Internet DiffServ (QoS garantido, QoS com carga controlada e melhor esforço). O objetivo do arcabouço é prover os requisitos destas classes de serviço, independente da dinâmica existente no estado dos recursos das grades. As classes de serviço no G-QoSM possuem um custo associado e as alocações são realizadas de modo a maximizar o lucro dos proprietários, sem violar os contratos estabelecidos com os usuários.

A união dos mecanismos VNET e VTTIF (*Virtual Topology and Traffic Inference*) dão origem a um mecanismo maior, referenciado aqui como VNET+VTTIF, e utilizado para adaptar alocações de aplicações executadas sobre o middleware Virtuoso [72]. O mecanismo realiza o autoajuste através de modificações na rede sobreposta (*overlay*) formada pelos recursos das organizações virtuais existentes. O VNET é um mecanismo que lida com a migração de máquinas virtuais, de modo que, durante a execução de uma aplicação, todas as máquinas envolvidas estejam em uma única organização virtual, independente das suas localizações físicas reais. O VTTIF é o mecanismo responsável por gerenciar a topologia da rede sobreposta formada pelas máquinas. Ele monitora as comunicações realizadas entre as tarefas das aplicações e modifica a topologia de modo a adequá-la ao padrão de comunicação detectado. O principal diferencial do VNET+VTTIF para outros mecanismos está no fato dele reagir às mudanças no tráfego gerado entre as

tarefas. O objetivo do VNET+VTTIF é otimizar a execução das aplicações através de mudanças na topologia virtual da rede sobreposta sobre a qual as aplicações executam.

O sistema de monitoramento GHS (*Grid Harvest Service*) proposto em [71] foca nas fases de monitoramento e previsão do estado da grade. Apesar de ser definido como um sistema de monitoramento de grades, o GHS implementa ferramentas que o tornam um mecanismo de autoajuste. A principal motivação do GHS é o fato dos mecanismos similares existentes serem voltados para aplicações que precisam de previsões para um curto intervalo de tempo, da ordem de dezenas de segundos. O GHS, ao contrário desses mecanismos, é voltado para fornecer previsões que descrevam o estado da grade durante um longo intervalo de tempo, da ordem de dezenas de minutos.

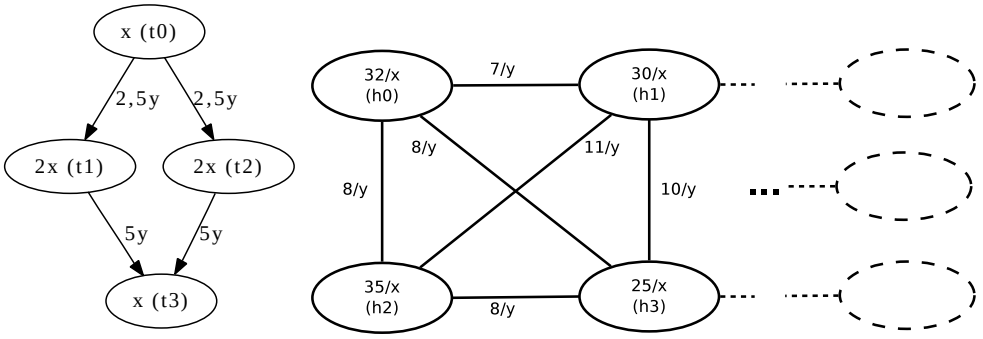
Uma abordagem definida como um escalonador mas que age como um sistema de autoajuste é o WBA (*Workflow Based Approach*) [18]. Ele age como um mecanismo para ajuste de alocações pelo fato de ser executado de forma cíclica, durante todo o tempo de vida das aplicações submetidas à grade. Em cada iteração do algoritmo, o estado dos recursos é atualizado, o que pode levar a mudanças na alocação dos recursos. O objetivo do WBA é minimizar o tempo de execução das aplicações.

Dentre os mecanismos implementados em [78], [19], [39], [75], [1], [72], [71] e [18], apenas aqueles em [39], [75] e [72] apresentam soluções para lidar com as incertezas das informações, sendo que todas agem de forma reativa. Os dois primeiros monitoram o tempo de execução das tarefas e propõem migrações quando os tempos de execução mostram-se diferentes do esperado, enquanto que o terceiro monitora o tráfego na grade e propõe a criação de topologias virtuais que mais se adequem ao padrão de tráfego.

2.4.2 Exemplo ilustrativo da necessidade dos escalonadores tratarem incertezas das informações de entrada

Esta subseção ilustra a necessidade de mecanismos de escalonamento que lidem adequadamente com as incertezas das informações de entrada. A Figura 2.8(a) apresenta o DAG de uma aplicação a ser escalonada na grade representada pelo grafo da Figura 2.8(b).

Os escalonadores HEFT e CPOP, que não implementam nenhum mecanismo para lidar com incertezas, foram utilizados para propor os escalonamentos. A fim de avaliar o impacto que os escalonadores sofrem com estimativas incorretas, o tempo necessário para transferir 1 bit entre dois computadores na grade foi aumentado em 25%, 50%, 100% e 200%, a fim de simular casos em que os estimadores superestimaram a largura de banda disponível ou casos em que os usuários descreveram demandas de comunicação menores do que as reais. Para cada um desses valores, os dois escalonadores forneceram escalonamentos considerando que haviam ferramentas de monitoramento precisas ou que haviam ferramentas de monitoramento imprecisas. No cenário com ferramentas precisas,



(a) DAG. (b) Alguns computadores da grade.

Figura 2.8: Exemplo para ilustrar a necessidade de tratar incertezas no escalonamento

os escalonadores receberam como entrada os novos tempos de transferência de dados (com os devidos incrementos), enquanto que no cenário com ferramentas imprecisas os escalonadores receberam como entrada os tempos originais, decorrentes das multiplicações dos pesos informados nos grafos das figuras 2.8(a) e 2.8(b), sem tomar conhecimento das incertezas. Neste último caso, pode-se dizer que os escalonadores foram executados com estimativas que possuíam os valores de QoI iguais a 80%, 66.67%, 50% e 33.33%, valores estes que estão associados, respectivamente, aos incrementos de 25%, 50%, 100% e 200%. A Equação 2.1 descreve a relação entre o valor da QoI e os valores reais (dr) e informados (di) das demandas das aplicações. No cenário com ferramentas precisas, as estimativas possuíam QoI de 100%, dado que os valores corretos foram informados para os escalonadores.

$$\text{QoI } = \min\left(\forall_{i\in \text{ demandas}}\ 1 - \frac{|dr_i - di_i|}{dr_i}\right) \times 100\% \tag{2.1}$$

A Figura 2.9 apresenta o *makespan* dos escalonadores nos cenários em que o valor da QoI foi igual a 100% (curvas "CPOP (estimativas precisas)" e "HEFT (estimativas precisas)") e nos cenários em que o valor da QoI foi menor que 100% (curvas "CPOP (estimativas imprecisas)" e "HEFT (estimativas imprecisas)").

A Figura 2.9 mostra que, quando os escalonadores foram executados em cenários com estimativas precisas, os seus *makespans* foram menores ou iguais do que os *makespans* quando foram simuladas estimativas imprecisas. Observando as curvas dos cenários com QoI igual a 100% e as curvas dos cenários com QoI menor que 100%, nota-se que para incrementos de até 50% no tempo de transferência dos dados, as estimativas precisas não fizeram diferença, ou seja, incrementos até 50% não foram suficientes para afetar o tempo de execução final da aplicação. Isso se dá pelo fato de uma aplicação em grades ser formada por tarefas que executam em paralelo e, portanto, o atraso na transferência

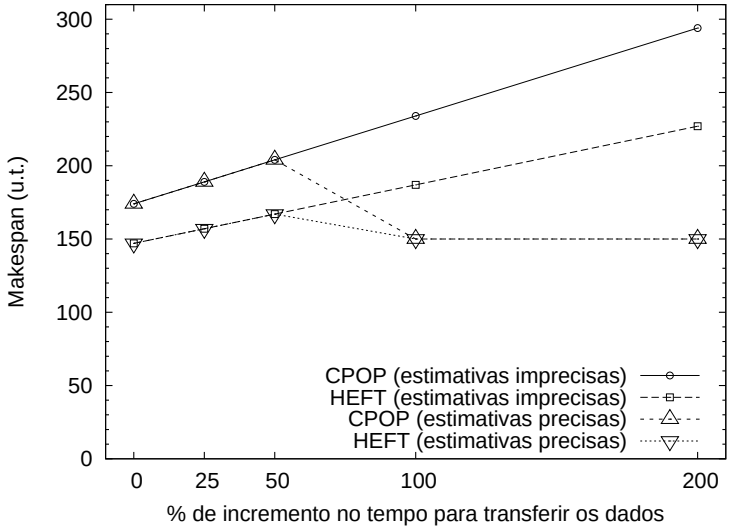


Figura 2.9: *Makespan* em cenários com QoI=100% e com QoI<100%

de dados entre um par de tarefas nem sempre afeta a aplicação como um todo. Por outro lado, para incrementos a partir de 100%, os escalonamentos propostos em cenários com estimativas precisas mostraram-se melhores do que aqueles propostos quando a QoI foi menor que 100%. Isso ocorreu pelo fato dos escalonadores fornecerem escalonamentos que evitaram a utilização da rede. Nos casos em que o valor da QoI foi menor que 100%, os escalonadores mantiveram a utilização da rede, já que os escalonamentos foram produzidos levando em consideração os pesos originais dos grafos.

Nos casos do CPOP e do HEFT executados em cenários nos quais o valor da QoI foi menor que 100%, os *makespans* aumentaram à medida que as incertezas aumentaram. Para um incremento no tempo de 200%, o *makespan* dos escalonadores CPOP e HEFT foram, respectivamente, 70% e 54% maiores do que aqueles quando não houve incertezas. Esses números comprovam que as incertezas das informações passadas como entrada para os escalonadores causam impactos negativos na execução das aplicações em grades e justificam a implementação de mecanismos que tornem os escalonadores robustos independente do valor da QoI.

2.4.3 Fuzzy × Probabilidade

A Subseção 2.4.2 apresentou um exemplo no qual as informações passadas como entrada para os escalonadores de tarefas, no ato de submissão de uma aplicação, diferiram dos dados reais, observados durante a execução da aplicação. Situações como a do exemplo são comuns em grades devido ao desconhecimento que os usuários têm a cerca dos detalhes

das aplicações, à dinâmica do ambiente e à imprecisão das ferramentas de monitoramento. Considerar que as informações disponibilizadas para os escalonadores são precisas é irreal para ambientes de larga escala e sem controle centralizado como as grades [67].

Como os pesos nos grafos das aplicações e das grades, passados como entrada para os escalonadores, não refletem os pesos reais observados durante a execução das aplicações, eles não podem ser representados por valores determinísticos em modelos matemáticos como aquele utilizado no escalonador IPDT (Subseção 2.2.2). As incertezas relacionadas com esses pesos justificam a representação deles como uma faixa de valores ao invés de um único valor. Dessa forma, os intervalos servem para representar as imprecisões dos pesos.

Conforme definido em [82], problemas que lidam com grandezas que não tenham critérios precisos para defini-las podem ser resolvidos com métodos existentes na teoria de conjuntos fuzzy, como números fuzzy e otimização fuzzy. É importante observar que apesar da teoria de conjuntos fuzzy lidar com grandezas incertas há todo um arcabouço matemático que permite os estudos precisos e rigorosos dos fenômenos relacionados com essas grandezas.

A teoria de conjuntos fuzzy provê uma forma de representar valores que não são determinísticos. Por exemplo, um usuário que não tenha certeza sobre a quantidade de bytes que serão transferidos entre duas tarefas da sua aplicação pode representar essa grandeza como um número fuzzy. Ao invés de um único valor B ($B \in \mathbb{R}$), o usuário pode descrever o número fuzzy $\widetilde{B}$ como sendo o conjunto de pares ordenados $\{(b, \mu_{\widetilde{B}}(b)) | b \in \mathbb{R}\}$ com $\mu_{\widetilde{B}} \in [0, 1]$. Valores de $\mu_{\widetilde{B}}(b)$ mais próximos de 1 estão associados a valores de b com uma "certeza maior" por parte do usuário.

O arcabouço matemático dos conjuntos fuzzy define diversos tipos de cálculo que podem ser realizados com números fuzzy. É possível realizar operações básicas como adição e subtração e até mesmo descrever problemas de otimização similares àquele apresentado na Subseção 2.2.2. A adoção de números fuzzy na descrição do problema de escalonamento combina com a existência de incertezas nas informações de entrada fornecidas aos escalonadores. Optou-se, consequentemente, por modelar os dois escalonadores propostos nesta Tese como problemas de otimização fuzzy. Nestes escalonadores, as incertezas referentes às informações passadas como entrada são representadas através da utilização de números fuzzy.

É importante observar que as incertezas presentes nas informações não podem ser confundidas com valores aleatórios. A teoria da probabilidade é útil para modelar situações nas quais há uma regularidade na ocorrência dos eventos [49]. É a regularidade que permite que eventos sejam descritos por distribuições de probabilidade, o que não ocorre nos cenários descritos nesta Tese. A descrição das demandas de uma aplicação através de distribuições de probabilidade, por exemplo, exigiria execuções prévias da aplicação,

o que não necessariamente ocorre na prática. Conforme esclarecido em [46]: *"Fuzziness is found in our decisions, in our thinking... Fuzziness is often confused with probability. A statement is probabilistic if it expresses a likelihood or degree of certainty if it is the outcome of clearly defined but random occurring events."*.

2.5 Resumo conclusivo

Este capítulo apresentou conceitos básicos necessários para a compreensão do conteúdo desta Tese. A definição de grade foi apresentada, o funcionamento de um escalonador de tarefas foi descrito e três escalonadores de tarefas que são utilizados na análise de desempenho dos escalonadores propostos nos próximos capítulos foram resumidos. A importância de considerar o estado da rede foi também apresentada, a fim de servir de motivação para os experimentos apresentados no Capítulo 5, que compara dois estimadores de largura de banda disponível.

Foi utilizado um exemplo numérico para demonstrar o impacto negativo de se ignorar as incertezas das grades. Observou-se aumentos de até 70% nos *makespans* de aplicações que foram escalonadas por escalonadores que receberam informações incorretas a cerca da disponibilidade da grade.

Algumas propostas existentes para lidar com as incertezas foram também resumidas. A maioria dessas propostas lida com incertezas utilizando mecanismos que realizam um ciclo contínuo de monitoramento, reescalonamento e migração de tarefas, ou seja, eles agem de forma reativa na tentativa de diminuir o impacto negativo das incertezas. A carga extra e a intrusão geradas na grade por esses ambientes justificam a implementação de mecanismos que lidem com as incertezas diretamente nos escalonadores de tarefas.

O capítulo justificou a utilização de mecanismos derivados da teoria de conjuntos fuzzy para a modelagem de escalonadores de tarefas robustos às incertezas. Esses escalonadores serão apresentados nos capítulos 3 e 4.

Capítulo 3

Escalonamento sob incertezas na descrição das aplicações

Conforme foi apresentado na Seção 2.2, o escalonamento de tarefas é uma etapa essencial para garantir que a execução das aplicações beneficiem-se da disponibilidade de recursos, bem como para uso eficiente dos mesmos. Escalonamentos que impliquem em tempos de execução equivalentes àqueles das execuções seriais e escalonamentos que não levem em consideração os custos de comunicação devem ser evitados para que se possa utilizar os recursos eficientemente.

Como a busca pelo escalonamento ótimo é um problema $\mathcal{NP}$-difícil, o algoritmo implementado pelo escalonador de tarefas desempenha um papel importante na qualidade do escalonamento. Ele deve ser rápido o suficiente para que possa ser utilizado em tempo hábil, bem como deve fornecer escalonamentos próximos do ótimo [12]. Tão importante quanto o algoritmo implementado no escalonador de tarefas, é a QoI referente à descrição da aplicação. É de pouca valia um algoritmo ótimo e eficiente se os pesos do DAG para o qual o escalonamento é produzido não corresponderem às demandas reais das aplicações, conforme ilustrado no exemplo das Figuras 2.8 e 2.9. A questão de informações imprecisas é crítica tanto para aquelas aplicações que geram dados diretamente em tempo de execução, visto que os pesos referentes às dependências entre as tarefas dificilmente são informados com uma QoI de 100%, quanto para aplicações que transferem dados da ordem de Petabytes como a grade computacional do LHC do CERN [96], dado que as incertezas, por menores que sejam, afetam a disponibilidade de recursos da grade.

É, portanto, necessária a concepção de escalonadores para grades que sejam robustos às incertezas das descrições das demandas das aplicações. Tais incertezas existem por diversas causas, como desconhecimento do usuário, falhas na aplicação e até mesmo por ações de usuários maliciosos, que descrevem as aplicações com requisitos diferentes a fim de obter maior prioridade no escalonamento. Incertezas nas informações não são exclusivas

de ambientes de grades e já foram estudadas em sistemas paralelos convencionais [24] [23] [51], porém as soluções propostas não podem ser reutilizadas no contexto de grades, pois sistemas paralelos convencionais são ambientes confinados e, portanto, as transferências de dados entre computadores não afeta, significativamente, o tempo de execução das aplicações. Por outro lado, a disponibilidade dos recursos da rede não pode ser ignorada no contexto de grades.

Com a consideração das incertezas nos escalonadores de tarefas, o diagrama que representa as entradas e saídas do escalonador, ilustrado na Figura 2.2, deve ser modificado conforme a Figura 3.1. Nesta figura, observa-se que a QoI referente à descrição das aplicações faz parte das entradas do escalonador.

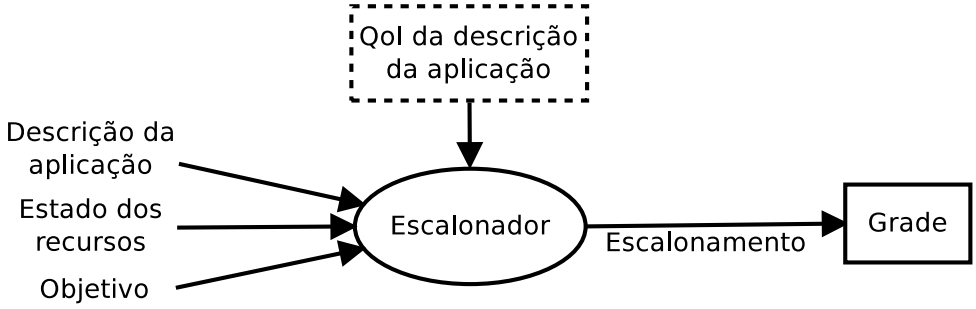


Figura 3.1: Entradas e saída de um escalonador de tarefas para grades com suporte às incertezas na descrição das aplicações

Existem, também, outras opções para tentar diminuir o impacto causado pelas incertezas na descrição das aplicações. Uma delas é aumentar a QoI através da implementação de um sistema que registre o histórico das execuções anteriores e construa modelos específicos para as aplicações, como proposto em [63] e em [75]. Essa solução, entretanto, é ineficiente para aplicações executadas poucas vezes e para aquelas que, mesmo sendo executadas diversas vezes, não apresentam um comportamento fácil de inferir a partir dos dados de entrada. Uma outra opção consiste em monitorar a execução das aplicações e tomar medidas reativas, como proposto em [39] e em [72]. O problema dessa opção é o fato dos limiares para se tomar reações serem desconhecidos e dependentes das aplicações.

Soluções reativas [63] [75] [27] [66] e soluções preventivas [26] [58] [30] [72] [39] para tratar as incertezas em grades têm sido propostas na literatura. Elas levam em consideração a QoI ou tentam aumentar o conhecimento sobre as aplicações através da análise do histórico das execuções passadas. Estas soluções, entretanto, falham por não identificarem todas as incertezas presentes em aplicações formadas por tarefas dependentes, e, em especial, aquelas com incertezas sobre a quantidade de dados transferidos entre as tarefas. Uma outra falha dessas propostas consiste no fato delas não considerarem ambientes heterogêneos, característica marcante das grades reais.

A partir dos trabalhos existentes na literatura, propôs-se um novo escalonador de tarefas para grades que fornece escalonamentos robustos frente às incertezas na descrição de aplicações formadas por tarefas dependentes e descritas por DAGs. Este capítulo apresenta esse escalonador. Diferente de outras propostas apresentadas na literatura, o escalonador trata incertezas nas demandas de comunicação das aplicações e não considera nenhuma topologia específica formada pelos recursos da grade. O escalonador, denominado IPDT-FUZZY, baseia-se em técnicas de otimização fuzzy. Números fuzzy triangulares representam as demandas das aplicações e a busca pelo melhor escalonamento é formulada como um problema de programação inteira 0–1. O escalonador IPDT-FUZZY é comparado com o escalonador IPDT, que foi resumido na Subseção 2.2.2 e que serviu de base para a sua construção. Resultados obtidos através de simulações de três aplicações que são comumente executadas em grades confirmam as vantagens em se utilizar o escalonador IPDT-FUZZY.

As próximas seções estão assim organizadas: a Seção 3.1 apresenta trabalhos relacionados ao escalonador IPDT-FUZZY. A Seção 3.2 descreve o problema de programação inteira 0–1 que constitui o escalonador IPDT-FUZZY. A Seção 3.3 apresenta os experimentos de simulação realizados para avaliar o desempenho do escalonador proposto e a Seção 3.4 resume os resultados obtidos e fornece conclusões parciais da Tese.

3.1 Trabalhos relacionados

Os argumentos apresentados nesta Tese para justificar a implementação de escalonadores que lidem com incertezas nas descrições das aplicações são reforçados pelos resultados publicados no trabalho em [18], cujo objetivo é propor e comparar duas abordagens para o escalonamento de aplicações formadas por tarefas dependentes. A primeira abordagem escalona as tarefas sem ter noção do DAG como um todo, ou seja, considera apenas os pesos computacionais. A segunda abordagem leva, adicionalmente, em consideração os pesos de comunicação entre tarefas dependentes. Diferente de outros trabalhos que propõem escalonadores de tarefas para grades, em [18] faz-se uma análise do impacto que incertezas na descrição das tarefas causam no *makespan* das aplicações. Mostra-se que o *makespan* das aplicações cresce com o aumento da incerteza (em certos casos chega a aumentar 400%). Apesar de reconhecer o impacto negativo causado pelas incertezas, não são propostas modificações nos escalonadores em [18], a fim de diminuí-lo. A análise de desempenho realizada em [18] é tomada como referência para a avaliação do escalonador IPDT-FUZZY. Tanto a faixa de valores de incerteza quanto uma das três aplicações simuladas são baseadas naquelas utilizadas em [18].

Em [27], propõe-se um escalonador que tem por objetivo utilizar valores reais dos tempos de execução das tarefas para aumentar o desempenho de aplicações limitadas por E/S

(*I/O bound*) e interativas. Assim como no escalonador IPDT-FUZZY, as demandas de comunicação entre as tarefas de uma aplicação não são ignoradas no escalonamento, apesar da proposta ser orientada a *clusters*. Esta proposta é baseada no monitoramento das tarefas, durante as suas execuções, a fim de agrupá-las em diferentes classes que são representadas como conjuntos fuzzy: limitada por E/S, intensiva em comunicação ou intensiva em processamento. O grau de pertinência das tarefas é computado utilizando medições realizadas nos computadores que depois são passadas para um estimador bayesiano. Caso seja detectada mudança de classe de uma tarefa, o escalonador busca preencher os intervalos de tempo vazios que surgem, devido à espera por dado de E/S ou à comunicação, com novas tarefas que possam ser executadas durante o tempo de espera. A análise de desempenho da proposta é realizada utilizando-se um simulador que modela uma topologia de rede local com 8 e com 16 máquinas. Os resultados apresentados mostram uma melhor utilização das máquinas e aumento na vazão do sistema. A principal diferença desta proposta para a empregada no escalonador IPDT-FUZZY é o fato de serem tomadas ações reativas às mudanças nos pesos esperados das tarefas. As simulações diferem por serem considerados ambientes homogêneos, com poucos computadores e com uma topologia de rede que é irreal para grades. O escalonador IPDT-FUZZY toma ações preventivas e os experimentos de simulação realizados consideram ambientes heterogêneos, uma das principais características das grades.

O estudo apresentado em [26] objetiva a criação de um modelo que descreva a carga de trabalho imposta por aplicações paralelas a supercomputadores. Dentre as várias métricas coletadas em relatórios de 4 supercomputadores, é avaliada a correlação entre o tempo de execução real e o tempo de execução requisitado pelo usuário para cada tarefa. Várias distribuições de probabilidade são derivadas, a fim de criar cargas de trabalho sintéticas que permitam a simulação de instantes de chegada de tarefas, cancelamento de tarefas, tempo de execução real das tarefas e tempo de execução previsto pelo usuário. Nesta proposta, diferentemente do considerado na formulação do escalonador IPDT-FUZZY, as tarefas nunca excedem o tempo de execução previsto pelos usuários. Caso as tarefas não finalizem até o tempo previsto, o ambiente de execução força as suas finalizações. Adotar esse comportamento em grades traria diversos problemas, dado que os recursos não são dedicados e nem homogêneos, tornando impraticável prever o tempo de execução preciso para todas as possibilidades de escalonamento. Assim como a proposta apresentada em [27], a proposta em [26] diferencia-se da empregada no escalonador IPDT-FUZZY pela análise do impacto das incertezas em ambientes paralelos formados por recursos homogêneos. Outra diferença é a consideração de tarefas independentes. Por outro lado, o trabalho é complementar ao apresentado nesta Tese, pois avalia os requisitos de aplicações executadas em ambientes paralelos reais.

Em [20], apresenta-se uma proposta para tratar as incertezas através da identificação

de quais tarefas das aplicações são mais críticas. Uma tarefa mais crítica é aquela cujo atraso na execução influencia diretamente o tempo de execução total da aplicação. Uma vez detectadas as tarefas mais críticas, elas são alocadas para um conjunto exclusivo de computadores. A abordagem adotada em [20] difere da empregada no escalonador IPDT-FUZZY por não considerar o impacto causado pelas incertezas nos requisitos de comunicação das aplicações. Uma similaridade encontrada entre os resultados publicados em [20] e os resultados gerados pelo escalonador IPDT-FUZZY é o fato dos ganhos alcançados pelo escalonador não serem expressivos quando há incertezas menores ou iguais a 40%. Diferente do realizado na análise de desempenho do escalonador IPDT-FUZZY, os autores de [20] não conduziram experimentos com incertezas maiores que 40%.

Em [65], apresenta-se o módulo responsável pela descrição da capacidade dos recursos disponíveis em grades gerenciadas pelo middleware ISAM (Infraestrutura de Suporte às Aplicações Móveis). O módulo tem como objetivo devolver uma descrição da grade que seja o mais próxima possível da real. Valores capturados por sensores na grade são passados como entrada para um algoritmo baseado em uma rede bayesiana que (i) autoajusta os seus valores com relação a mudanças dinâmicas no ambiente; (ii) usa algoritmos de aprendizagem para predizer o estado dos recursos; e (iii) considera o estado passado para prever o estado atual. Com a descrição mais próxima do real, o módulo resolve o problema de incerteza sobre a disponibilidade dos recursos porém não lida com incertezas na descrição das aplicações, o que é abordado pelo escalonador IPDT-FUZZY. Assim como na avaliação de desempenho realizada nesta Tese, em [65] há comentários a respeito do tempo de execução dos escalonadores. Os experimentos realizados mostram ganhos de 21,3% em relação a um escalonamento que utiliza a descrição fornecida pelo NWS, porém, avalia-se, apenas, o impacto das incertezas na capacidade de processamento disponível de uma grade homogênea formada por 20 computadores.

Em [66], apresenta-se uma proposta para lidar dinamicamente com as incertezas. Se uma tarefa extrapola o tempo previsto de execução, ela entra para um conjunto de tarefas passíveis de serem migradas. O que define se a tarefa entrará nesse conjunto são dois valores (*slack* e *spare time*) que definem até quando uma tarefa pode aumentar seu tempo de execução sem que o *makespan* da aplicação seja modificado. Os experimentos realizados mostram que os algoritmos utilizados no reescalonamento exercem papel fundamental para o tratamento das incertezas, quando a QoI varia de 10 a 50% em uma grade de apenas 3 a 8 computadores. Apesar da formulação considerar o impacto na transmissão dos dados, ela não considera o efeito da QoI no caso de imprecisões no tempo de transferência dos dados da aplicação, o que a difere da formulação do escalonador IPDT-FUZZY. Além disso, os experimentos realizados em [66] não consideram incertezas diferentes daquelas esperadas, o que não ocorre nos experimentos realizados para avaliar o desempenho do escalonador IPDT-FUZZY.

Uma análise de carga de trabalho semelhante à apresentada nos experimentos em [26] e voltada para grades é apresentada em [58]. Um *cluster* que faz parte da grade do projeto EGEE (*Enabling Grids for E-sciencE*) [90] teve seus relatórios de utilização do sistema analisados, a fim de se buscar modelos estatísticos que descrevam o tempo entre chegadas de tarefas e o tempo de execução de cada tarefa. Chegou-se à conclusão que o primeiro exibe dependência de longa duração enquanto o segundo é melhor modelado por um processo hiperexponencial com 3 estados. No *cluster* analisado, os usuários fornecem uma previsão do tempo de execução de suas tarefas, entretanto, na análise realizada, tais valores não são utilizados para a criação dos modelos. Nas aplicações analisadas, os recursos mais utilizados são os recursos de processamento. Poucos bits são transferidos entre as tarefas (a grande maioria das dependências de dados é da ordem de KB) e não há proposta de modelos estatísticos que descrevam a utilização da rede por parte das aplicações. Um problema desta análise é o fato dela se concentrar em um único *cluster* de uma grade, o que limita a generalidade das conclusões apresentadas.

Em [30], propõe-se que as incertezas sejam utilizadas como critério para que aplicações tenham permissão de utilizar os recursos das grades. A proposta não diferencia se a origem das incertezas foi um erro do usuário na hora de descrever os requisitos ou se foi alguma variação no estado dos recursos alocados. Uma diferença do trabalho em [30] para o realizado nesta Tese é a consideração da ocorrência de falhas nas tarefas, o que pode levar as mesmas a não serem executadas. Outra diferença é o fato da proposta em [30] ser voltada para grades que negociam SLAs (*Service Level Agreements*) com as aplicações. Após avaliar as incertezas de uma aplicação, a proposta toma a decisão de aceitar ou não a aplicação na grade. Uma semelhança entre o trabalho publicado em [30] e o realizado no desenvolvimento do escalonador IPDT-FUZZY é o fato dos requisitos das aplicações serem modelados como números fuzzy triangulares. Números fuzzy triangulares costumam ser usados para representar matematicamente grandezas incertas que não possuam uma função de pertinência mais específica conhecida.

A proposta de autoajuste da alocação de recursos em grades apresentada em [12] depende de escalonadores de tarefas eficientes tanto em termos do *makespan* produzido quanto em termos do tempo de execução necessário para derivar o escalonamento. Apresenta-se um conjunto de escalonadores e dentre eles está o escalonador IPDT. O escalonador IPDT-FUZZY engloba o escalonador IPDT por incluir técnicas de otimização fuzzy na formulação do problema de programação inteira e por descrever as demandas das aplicações como números fuzzy triangulares. Como a derivação do escalonador IPDT-FUZZY baseou-se na formulação do escalonador IPDT, a análise de desempenho apresentada na Seção 3.3 compara os escalonamentos propostos por ambos. Dessa forma é possível avaliar a efetividade do emprego de técnicas de otimização fuzzy.

Em resumo, um número considerável de trabalhos na literatura lida com as incerte-

zas de forma reativa e muitos deles ignoram alguma das duas características marcantes das grades: a dependência dos recursos da rede e a heterogeneidade dos recursos e das aplicações. O escalonador IPDT-FUZZY é uma solução preventiva para lidar com as incertezas na descrição das aplicações em grades. Por não agir de forma reativa, o escalonador evita a carga extra do monitoramento constante e das migrações. Além disso, o escalonador considera a heterogeneidade nas aplicações e nos recursos das grades, concentra-se nas incertezas relacionadas com as demandas de comunicação das aplicações e é voltado para aplicações de *e-Science* que são formadas por tarefas dependentes, cuja principal característica é a transferência maciça de dados via rede.

3.2 O escalonador IPDT-FUZZY

O escalonador IPDT-FUZZY foi projetado para lidar tanto com incertezas nas demandas de comunicação quanto com incertezas nas demandas computacionais. O escalonador amplia a formulação do escalonador IPDT, apresentado na Subseção 2.2.2, através do suporte às incertezas nas demandas das aplicações. Como mencionado na Seção 3.1, trabalhos anteriores negligenciaram as questões referentes às demandas de comunicação. As avaliações do escalonador IPDT-FUZZY dão, consequentemente, ênfase a situações que envolvam incertezas nas demandas de comunicação das aplicações.

Assim como o escalonador IPDT, o escalonador IPDT-FUZZY resolve um problema de programação inteira 0–1 que recebe dois grafos como entrada. O grafo $H = (V_H, A_H)$ representa a topologia formada pelos recursos da grade e o DAG $D = (V_D, A_D)$ representa as dependências entre as tarefas da aplicação a ser escalonada. V_H é o conjunto de m computadores da grade conectados pelo conjunto de enlaces A_H. Os computadores são identificados pelos rótulos $\{1, \ldots, m\}$. V_D é o conjunto de n tarefas da aplicação, com suas dependências representadas pelo conjunto de arcos A_D. As tarefas são identificadas pelos rótulos $\{1, \ldots, n\}$, de modo que a ordem crescente dos rótulos representa uma ordem topológica do DAG. Os DAGs aceitos pelo escalonador devem ter uma única tarefa de entrada e uma única tarefa de saída. DAGs que não atendam a essas condições devem ser modificados através da adição de tarefas artificiais com pesos zero. O escalonador devolve como saída os instantes em que cada tarefa deve ser executada nos computadores da grade, bem como os instantes em que cada transferência de dados deve ocorrer. De posse dessas informações, é possível gerar um Diagrama de Gantt que descreve todo o escalonamento proposto para a aplicação.

As características do DAG D que precisam ser conhecidas são: I_i: quantidade de instruções da tarefa i ($I_i \in \mathbb{R}_+$); $B_{i,j}$: quantidade de bits dos dados de dependência entre as tarefas i e j ($B_{i,j} \in \mathbb{R}_+$); e A_D, que é o conjunto de arcos ij tal que $i < j$, e existe um arco do vértice i para o vértice j no DAG D. Os conjuntos de pesos I e B são as

informações que dependem do conhecimento do usuário e que dificilmente são descritas corretamente na prática.

As características do grafo H que precisam ser conhecidas são: TI_k: intervalo de tempo que o computador k leva para executar 1 instrução ($TI_k \in \mathbb{R}_+$); $TB_{k,l}$: intervalo de tempo necessário para transferir 1 bit pelo enlace que conecta o computador k ao computador l ($TB_{k,l} \in \mathbb{R}_+$); e $\delta(k)$: conjunto de computadores com enlace para o computador k, incluindo o próprio computador k. Considera-se que $\forall k \in \{1, \ldots, m\}$, $TB_{k,k} = 0$, ou seja, se uma tarefa for executada no mesmo computador que sua única tarefa predecessora ela não sofrerá atraso decorrente da transferência dos dados. Tal hipótese assume que o computador possui espaço de armazenamento suficiente e que os dados estão disponíveis imediatamente após o término de execução das tarefas. Os valores de TI e de TB dependem, respectivamente, da carga de processamento no computador e do congestionamento nos enlaces de rede que interligam os computadores das grades. O escalonador IPDT-FUZZY não leva em consideração incertezas nos valores de TI e de TB. Tal consideração é feita na formulação do escalonador IP-FULL-FUZZY, que será apresentado no Capítulo 4.

O método escolhido para repassar as incertezas da descrição das aplicações ao escalonador foi a consideração de que os pesos do DAG (elementos dos conjuntos I e B) são números fuzzy. Assim como em [30], considera-se que os números fuzzy são triangulares. A utilização de números fuzzy triangulares é reforçada pelo fato de que não existem referências disponíveis que indiquem com precisão um tipo de número fuzzy para representar as demandas das aplicações. O valor de um número fuzzy varia em um intervalo $[min, max]$ e é associado com uma função de pertinência $\mu(x), x \in \mathbb{R}, \mu(x) \in [0, 1]$; para valores de x menores que min e para valores de x maiores que max a função de pertinência assume o valor zero. Para valores de x entre min e max, a função de pertinência pode ser representada por diversas formas e é essa forma que vai classificar o tipo de número fuzzy. O gráfico da Figura 3.2 ilustra um número fuzzy triangular.

Uma tarefa i é descrita como tendo $\widetilde{I_i}$ instruções com uma incerteza de $\sigma\%$ sobre esse valor. A quantidade de instruções da tarefa i é representada, utilizando a notação de números fuzzy, por $[\underline{I_i}, I_i, \overline{I_i}]$ onde $\underline{I_i} = I_i(1 - \frac{\sigma}{100})$ e $\overline{I_i} = I_i(1 + \frac{\sigma}{100})$. De modo similar, as demandas de comunicação $\widetilde{B_{i,j}}$ do DAG são representadas por $[\underline{B_{i,j}}, B_{i,j}, \overline{B_{i,j}}]$ com uma incerteza de $\rho\%$. $\underline{B_{i,j}} = B_{i,j}(1 - \frac{\rho}{100})$ e $\overline{B_{i,j}} = B_{i,j}(1 + \frac{\rho}{100})$.

É importante observar que os valores de σ e ρ não são aleatórios. Eles devem ser definidos pelo administrador da grade ou pelo usuário que submete a aplicação de modo a refletir o conhecimento que se tem a cerca das aplicações. A Subseção 3.3.3 apresenta resultados alcançados com a utilização de valores aleatórios ao invés de números fuzzy. Nota-se que o primeiro leva a um pior desempenho.

Para contornar os efeitos da complexidade $\mathcal{NP}$-difícil do problema de encontrar o

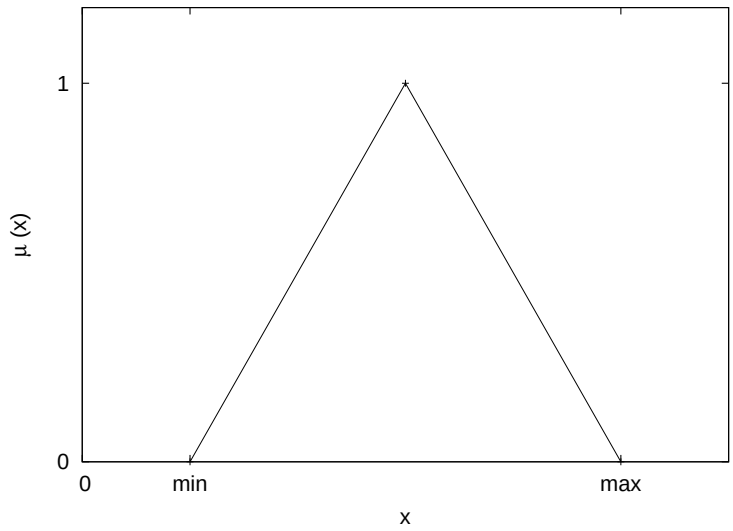


Figura 3.2: Um número fuzzy triangular [min,max]

escalonamento ótimo para tarefas em grades, no escalonador IPDT-FUZZY a linha do tempo é discreta. Embora a discretização do tempo introduza aproximações e uma perda de precisão, sob certas circunstâncias, esta perda pode não ser significante. A diminuição no tempo de execução do escalonador considerando a linha do tempo discreta mostra-se vantajosa, ao se comparar os resultados produzidos com os de um escalonador que considera o tempo contínuo. Assume-se, por conveniência, que a linha do tempo de execução do DAG a ser escalonado é representada por $\mathcal{T}' = \{1, \ldots, T'_{max}\}$, onde $T'_{max} = T_{max}(1+\frac{\sigma}{100})$ e T_{max} é o maior *makespan* possível para o DAG. Outro valor que precisa ser conhecido é $T'_{min} = T_{min}(1-\frac{\sigma}{100})$, onde T_{min} é o menor *makespan* possível da aplicação, i.e., o tempo que levaria para executar sequencialmente todas as tarefas do caminho mais longo do DAG, em termos dos pesos I, no computador mais rápido da grade ($T_{min} = \min(\{TI_{k|k\in V_H}\}) \times \sum_{i\in \text{ maior caminho}} I_i$). Os valores de T_{max}, T_{min}, T'_{max} e T'_{min} podem ser calculados diretamente pelos pesos do DAG e do grafo da grade; não há necessidade de executar a aplicação no computador mais rápido da grade, a fim de se encontrar esses valores.

O escalonamento encontrado pelo escalonador IPDT-FUZZY é dado pelos valores das variáveis $x_{i,t,k}$ ($\in \{0,1\}$) e f_i ($\in \mathbb{N}^*$). $x_{i,t,k}$ é uma variável binária que vale 1 se, e somente se, a tarefa i terminar sua execução no instante de tempo t e no computador k; f_i é uma variável inteira que armazena o instante de tempo no qual a tarefa i termina sua execução ($f_i = \sum_{t\in\mathcal{T}'} \sum_{k\in V_H} t \times x_{i,t,k}$). Como a variável f_i pode ser representada em termos das variáveis $x_{i,t,k}$, a formulação do problema só precisa encontrar os valores destas últimas variáveis, que assumem valores $\in \{0,1\}$.

O escalonador IPDT-FUZZY é expresso pela seguinte formulação matemática:

Maximize λ

sujeito a

$$1 - \frac{f_n - T'_{min}}{T'_{max} - T'_{min}} \geq \lambda \qquad \text{(F1)}$$

$$\sum_{t \in \mathcal{T}'} \sum_{k \in V_H} x_{j,t,k} = 1 \qquad \text{para } j \in V_D; \qquad \text{(F2)}$$

$$x_{j,t,k} = 0 \qquad \text{para } j \in V_D, \quad k \in V_H, \quad t \in \{1, \ldots, \lceil \underline{I_j} TI_k \rceil\}; \qquad \text{(F3)}$$

$$\sum_{k \in \delta(l)} \sum_{s=1}^{\lceil t - \overline{I_j} TI_l - \overline{B_{i,j}} TB_{k,l} \rceil} x_{i,s,k} \geq \sum_{s=1}^{t} x_{j,s,l} \qquad \text{para } j \in V_D, \quad ij \in A_D, \quad \text{para } l \in V_H, \quad t \in \mathcal{T}'; \qquad \text{(F4)}$$

$$\sum_{j \in V_D} \sum_{s=t}^{\lceil t + \underline{I_j} TI_k - 1 \rceil} x_{j,s,k} \leq 1 \qquad \text{para } k \in V_H, \quad t \in \mathcal{T}', \quad t \leq \lceil T'_{max} - \underline{I_j} TI_k \rceil; \qquad \text{(F5)}$$

$$x_{j,t,k} \in \{0, 1\} \qquad \text{para } j \in V_D, \quad l \in V_H, \quad t \in \mathcal{T}'. \qquad \text{(F6)}$$

A função objetivo do escalonador IPDT-FUZZY maximiza o grau de satisfação λ ($\in [0, 1]$), que é inversamente proporcional ao *makespan* da aplicação (f_n) dado pelo escalonamento. O tempo de execução da aplicação é limitado pelos valores de T'_{min} e T'_{max}. A Figura 3.3 exibe o gráfico com a relação entre λ e o tempo de execução da aplicação f_n.

A fim de incluir a faixa de valores de f_n ($\in [T'_{min}, T'_{max}]$) e a relação com o grau de satisfação λ, a função do gráfico da Figura 3.3 está representada no escalonador IPDT-FUZZY pela restrição (F1). A restrição (F2) garante que cada tarefa do DAG só pode executar em um único computador e finalizar a sua execução em um único instante de tempo, enquanto que a restrição (F6) define o domínio para as variáveis $x_{j,t,k}$ utilizadas na formulação.

Como I_i e $B_{i,j}$ são números fuzzy, as restrições que utilizam ambos os valores ((F3), (F4) e (F5)) devem empregar as incertezas (σ e ρ). A restrição (F3) determina que

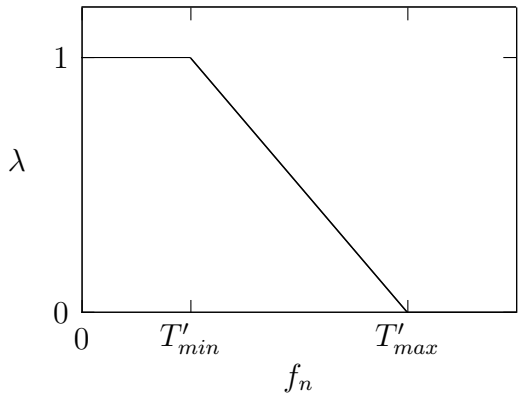


Figura 3.3: Grau de satisfação

uma tarefa (j) não pode terminar sua execução até que todas as suas instruções tenham sido completadas. Como não é possível saber o número exato de instruções, o valor I_j é substituído por $\underline{I_j}$, já que a quantidade mínima de instruções é o único valor corretamente conhecido. Dessa forma, evita-se a subutilização de recursos. A restrição (F4) define que a tarefa j só pode começar sua execução depois que todas as tarefas predecessoras terminarem suas execuções e transferirem os dados requeridos pela tarefa j. Para evitar que a tarefa j comece sua execução antes dos dados de dependência terem sido transferidos, I_j e $B_{i,j}$ são substituídos, respectivamente, pelos valores máximos $\overline{I_j}$ e $\overline{B_{i,j}}$. A restrição (F5) estabelece que cada computador só pode executar no máximo uma (1) tarefa por vez. Como o tempo mínimo de execução de uma tarefa é conhecido, só é possível garantir que um computador vai executar a tarefa durante esse tempo, motivo pelo qual o valor I_j é substituído por $\underline{I_j}$. T'_{max} é utilizado em substituição a T_{max}, a fim de representar todo o possível intervalo de execução das tarefas.

A Seção 3.3 apresentará os detalhes sobre a implementação do escalonador IPDT-FUZZY, bem como os resultados decorrentes dos experimentos de simulação que foram realizados com o objetivo de avaliar o desempenho do escalonador.

3.3 Resultados numéricos

Nesta seção, são apresentados os resultados das simulações realizadas com o objetivo de avaliar o desempenho do escalonador IPDT-FUZZY. A avaliação de desempenho é realizada através de comparações com a contraparte não fuzzy do escalonador IPDT-FUZZY, o escalonador IPDT. Ambos os escalonadores resolvem problemas de programação inteira, que têm como objetivo gerar escalonamentos que minimizem o *makespan* do DAG passado como entrada. As variáveis utilizadas nas formulações dos dois problemas de programação

inteira são similares, com exceção dos parâmetros de incerteza ρ e σ que estão presentes apenas na formulação do escalonador IPDT-FUZZY. Além disso, há uma correspondência entre a maioria das restrições presentes em cada um dos escalonadores. As restrições (D_i) do escalonador IPDT correspondem às restrições (F_{i+1}) do escalonador IPDT-FUZZY. Dessa forma, a comparação entre os dois escalonadores permite que se avalie os ganhos decorrentes da inclusão das técnicas de otimização fuzzy na formulação do problema.

A fim de garantir que os resultados encontrados fossem próximos daqueles que seriam encontrados caso fossem realizadas medições em uma grade real, simulou-se a execução de DAGs, baseados em três aplicações reais, sobre grades com topologias de rede baseadas em topologias reais.

A primeira aplicação que foi simulada é a aplicação de astronomia Montage [83], cujo modelo encontra-se presente em [18]. Os pesos do DAG baseado nessa aplicação foram definidos, de forma aleatória, seguindo uma distribuição uniforme com médias de 72×10^6 bits para os pesos dos arcos e $31,5 \times 10^{12}$ instruções para os pesos das tarefas. A Figura 3.4 ilustra o DAG da aplicação Montage sem os pesos. Nota-se que o DAG possui 26 tarefas com 39 dependências entre elas.

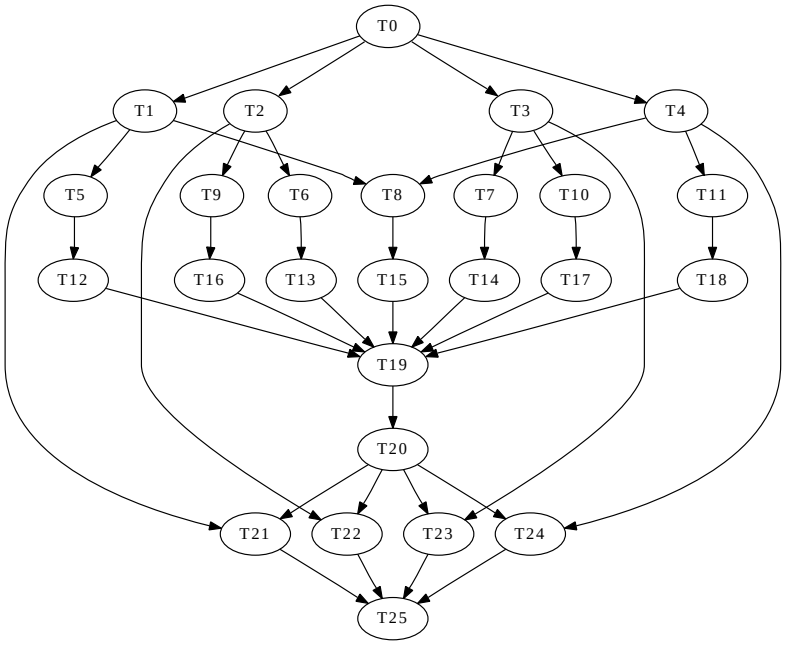


Figura 3.4: DAG da aplicação Montage

A segunda aplicação que foi simulada é a aplicação de química quântica WIEN2k [17], cujo modelo foi apresentado em [77]. Os pesos do DAG baseado nessa aplicação foram definidos da mesma forma que os pesos do DAG baseado na aplicação Montage. A Figura 3.5 ilustra o DAG da aplicação WIEN2k sem os pesos. Nota-se que o DAG possui

26 tarefas com 43 dependências entre elas.

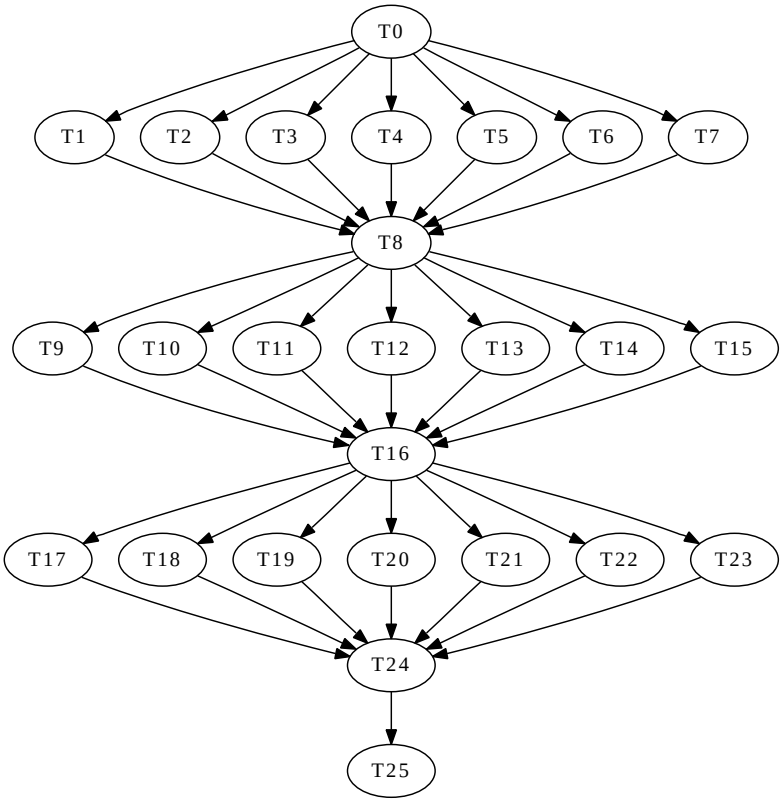


Figura 3.5: DAG da aplicação WIEN2k

A terceira aplicação que foi simulada representa uma versão modificada do modelo original da aplicação WIEN2k. As tarefas são dispostas em apenas 3 níveis, conforme exibido no DAG da Figura 3.6. Os pesos do DAG não são exibidos na figura. Eles são gerados da mesma forma que nos DAGs anteriores. Na Figura 3.6, nota-se que o DAG possui 26 tarefas e 48 dependências entre as tarefas. Esse tipo de DAG com apenas 3 níveis é muito comum em aplicações que são executadas em grades, principalmente aquelas voltadas para processamento de imagem. Nessas aplicações, a imagem original é dividida pela tarefa de entrada, cada uma das tarefas do nível intermediário executa um algoritmo, nessas divisões, e gera saídas parciais que são unidas pela tarefa de saída do DAG [68].

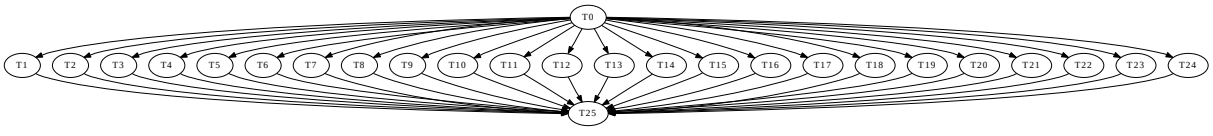


Figura 3.6: DAG modificado da aplicação WIEN2k

Os grafos das grades que foram passados como entrada para os escalonadores foram gerados através de um gerador de topologias. Geradores de topologias utilizam expressões matemáticas e probabilidades para definir se dois vértices de um grafo terão arestas entre eles e qual será o peso dessa aresta. Os vértices representam os computadores e/ou os comutadores da topologia e as arestas representam os enlaces, com os pesos equivalentes à capacidade ou à largura de banda disponível dos enlaces. Como as grades são ambientes computacionais que expandem-se por todo o planeta, muitas delas utilizam a Internet para a interligação de seus recursos. Dessa forma, é aceitável simular uma grade sobre uma topologia que seja similar às topologias formadas por recursos reais conectados via Internet. No entanto, por causa das constantes mudanças na configuração da Internet, obter uma formulação matemática que gere topologias coerentes com a evolução temporal da mesma é uma tarefa difícil. As topologias geradas nos experimentos que serão relatados nesta seção utilizaram o método Doar-Leslie publicado em 1993 [28]. Como pode ser observado em [29], [54] e [2], existem geradores de topologia mais recentes do que o método Doar-Leslie, entretanto, o método empregado nesses geradores recentes tem sido alvo de críticas que afirmam que as topologias geradas por eles não são tão próximas de topologias reais como afirmado pelos autores [37]. Dado que nem os geradores de topologia mais recentes conseguem gerar topologias que sejam comprovadamente próximas daquelas encontradas na Internet, optou-se por utilizar o método Doar-Leslie, apesar de ser mais antigo, pela sua simplicidade de implementação.

Vinte grades foram utilizadas nos experimentos. Cada uma delas possui 50 computadores e pesos médios de $\frac{9}{5{,}25\times10^{12}}$ minutos/instrução (9726MIPS, o que é equivalente ao desempenho nominal de um processador Intel Pentium IV) para os computadores e $\frac{1}{6\times10^{9}}$ minutos/bit (100Mbps) para os enlaces. Os demais parâmetros utilizados no gerador de topologias foram $\beta = 0,98$ e $\alpha = 0,98$. O primeiro representa o nível de conectividade dos nós (quanto mais próximo de 1, maior é a probabilidade do grafo gerado ser um grafo completo) e o segundo representa a relação entre a quantidade de enlaces com maior disponibilidade e a quantidade de enlaces com menor disponibilidade (quanto mais próximo de 1, maior é a probabilidade de que a média e a mediana das disponibilidades dos enlaces sejam iguais).

Os valores das incertezas passados como entrada para o escalonador IPDT-FUZZY podem ser os mais diversos possíveis, dado que eles dependem do conhecimento que os usuários e os administradores das grades têm das aplicações. Foram utilizados valores que já haviam sido empregados em estudos similares encontrados na literatura [66] [18]. Os valores utilizados para representar a incerteza nas demandas de comunicação das aplicações (ρ) variaram no conjunto {40%,100%,200%,400%}, valores que correspondem, respectivamente, a valores de QoI iguais a $\{71,43\%, 50\%, 33,33\%, 20\%\}$ (derivado pela Equação 2.1 e pela definição de ρ). Conforme explicado na Seção 3.2, o foco dos experimentos é ava-

liar o escalonador IPDT-FUZZY em situações onde hajam incertezas nas demandas de comunicação das aplicações. Desta forma, o valor de σ em todos os experimentos será igual a zero.

A fim de se avaliar a robustez dos escalonamentos, deve-se utilizar DAGs cujos pesos sejam diferentes daqueles passados como entrada para o escalonador. São nesses novos DAGs que as métricas de qualidade do escalonamento são avaliadas. Nos experimentos de simulação os novos DAGs foram gerados através do aumento dos pesos dos arcos dos DAGs originais. Os aumentos foram definidos de forma aleatória segundo uma distribuição uniforme no intervalo de 0 a $x\%$, com $x \in \{40, 100, 200, 400\}$. Para cada valor de x, vinte novos DAGs foram gerados. A fim de deixar clara a diferença entre os valores de x e de ρ, o primeiro será referenciado como "incerteza ocorrida" e o segundo como "incerteza esperada" no decorrer desta Tese.

Para avaliar o desempenho dos escalonadores, foi implementado um simulador que recebe como entrada um escalonamento de tarefas, um DAG, que pode ter pesos diferentes do DAG original, e um grafo correspondendo à topologia da grade, que deve ser igual ao grafo de topologia original. A Figura 3.7 ilustra o processo de simulação realizado e que corresponde à sequência de eventos descrita no Algoritmo 1.

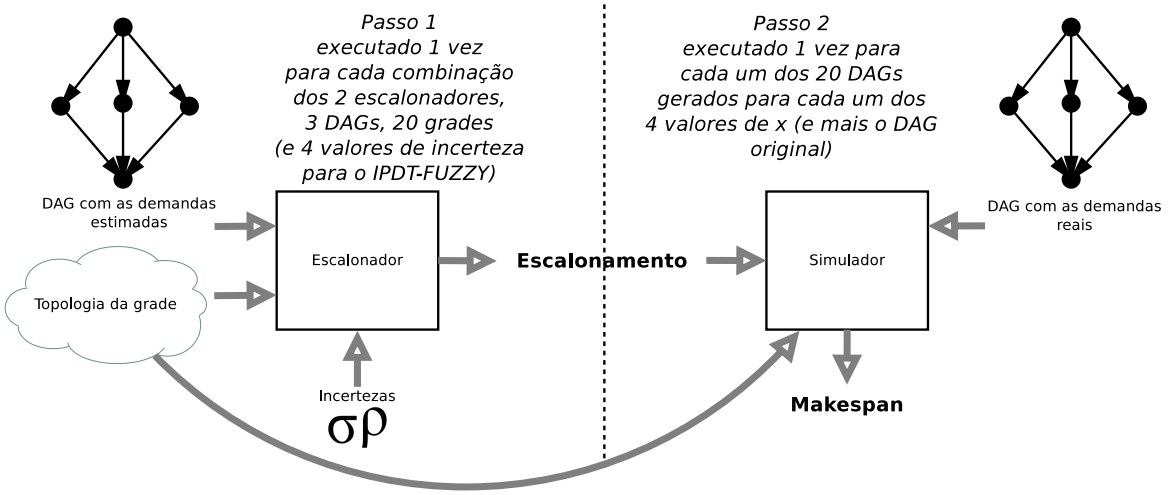


Figura 3.7: Diagrama de fluxo do processo de simulação

Todos os escalonadores e o simulador foram implementados em C. Os escalonadores utilizaram a biblioteca de otimização `Xpress` [91] versão 2006A.1. Todas as execuções foram realizadas em um computador equipado com um processador Pentium IV, 3,2GHz, 2,5GB de memória RAM e sistema operacional Debian GNU/Linux versão Lenny.

As Subseções 3.3.1, 3.3.2 e 3.3.3 apresentam, respectivamente, os resultados obtidos

Algoritmo 1 Simulação dos cenários para avaliação de desempenho do escalonador IPDT-FUZZY

```
1: para cada um dos 3 DAGs faça
2:   para cada uma das 20 topologias de grade faça
3:     para cada um dos 2 escalonadores faça
4:       se o escalonador é o IPDT-FUZZY então
5:         para cada um dos 4 valores de ρ faça
6:           gere um escalonamento
7:           para cada um dos 4 valores de x faça
8:             para cada um dos 20 DAGs referentes ao valor de x faça
9:               calcule o makespan com o escalonamento do passo 6
10:            fim para
11:          fim para
12:          calcule o makespan considerando o DAG original com o escalonamento do
             passo 6
13:        fim para
14:      senão
15:        gere um escalonamento
16:        para cada um dos 4 valores de x faça
17:          para cada um dos 20 DAGs referentes ao valor de x faça
18:            calcule o makespan com o escalonamento do passo 15
19:          fim para
20:        fim para
21:        calcule o makespan considerando o DAG original com o escalonamento do
           passo 15
22:      fim se
23:    fim para
24:  fim para
25: fim para
```

com os experimentos que avaliaram a qualidade do escalonador em termos de *speedup*, utilização da rede na grade e tempo de execução em várias situações nas quais os pesos reais dos arcos do DAG foram, na prática, diferentes daqueles passados como entrada para os escalonadores. Os intervalos de confiança exibidos nos gráficos apresentados a seguir foram calculados com um nível de confiança de 95%.

3.3.1 *Speedup*

As Figuras 3.8, 3.9 e 3.10 exibem os gráficos que plotam o *speedup* médio dos escalonamentos derivados pelos escalonadores para, respectivamente, o DAG da aplicação

Montage, o DAG da aplicação WIEN2k e o DAG modificado da aplicação WIEN2k (a partir deste ponto os DAGs serão referenciados como DAG Montage, DAG WIEN2k e DAG WIEN2k-modificado). Os *speedups* são plotados em função de diferentes valores de incerteza ocorrida nos pesos dos arcos dos DAGs. Em cada gráfico, há cinco curvas; uma curva descreve o comportamento do escalonador IPDT e as outras quatro descrevem o comportamento do escalonador IPDT-FUZZY para cada um dos valores de ρ.

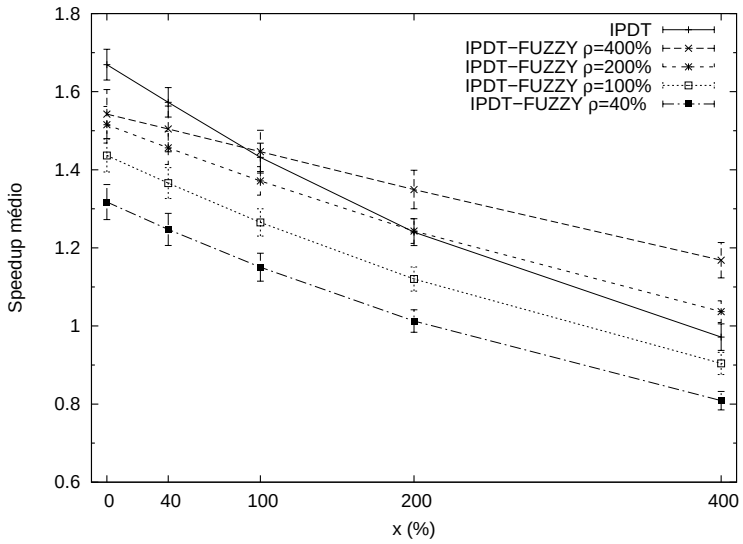


Figura 3.8: *Speedup* médio para o DAG Montage

Observa-se nos gráficos da Figura 3.8 que o escalonamento do DAG Montage pelo escalonador IPDT-FUZZY com altos valores de ρ levou a *speedups* maiores ou iguais do que aqueles dos escalonamentos produzidos pelo escalonador IPDT para praticamente todos os valores de x. Nota-se que quando o escalonador IPDT-FUZZY é projetado com $\rho = 400\%$, ele produz *speedups* melhores que o escalonador IPDT nos casos onde a incerteza ocorrida é maior que 100% ($x > 100\%$). Por exemplo, quando $x = 400\%$ o *speedup* do escalonador IPDT-FUZZY é, em média, 20,27% maior do que aquele do escalonador IPDT. Ainda para $\rho = 400\%$, quando a incerteza ocorrida é de 40% ou 100% ($0\% < x \leq 100\%$), o *speedup* produzido pelo escalonador IPDT-FUZZY é equivalente àquele do escalonador IPDT. O escalonador IPDT-FUZZY também produz *speedups* maiores ou equivalentes aos do escalonador IPDT, quando é projetado com $\rho = 200\%$ e quando a incerteza ocorrida é maior que 40% ($x > 40\%$). Pode-se notar que o *speedup* do escalonador IPDT decresce duas vezes mais rápido do que o *speedup* do escalonador IPDT-FUZZY com $\rho = 400\%$, em função do aumento de x. A partir desses dados, pode-se concluir que para o DAG Montage o escalonador IPDT-FUZZY, projetado com a expectativa de altos valores de incerteza, é mais robusto do que o escalonador IPDT, mesmo quando as incertezas ocorridas estão

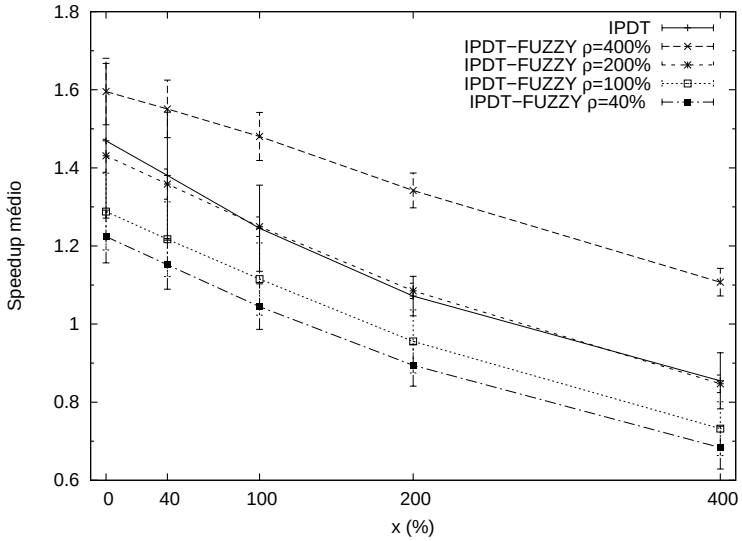


Figura 3.9: *Speedup* médio para o DAG WIEN2k

além das incertezas esperadas.

É importante observar que quando o escalonador IPDT-FUZZY foi projetado para valores baixos de incerteza, os seus escalonamentos não apresentaram ganhos significativos, quando comparados com os do escalonador IPDT. Isso ocorre pela falta de precisão introduzida pelas técnicas de otimização fuzzy, dada a sua flexibilidade limitada quando ρ assume valores pequenos. Por outro lado, quando a flexibilidade decorrente da incerteza aumenta (valores maiores de ρ), a habilidade que as técnicas fuzzy têm de lidar com as incertezas destaca-se, levando a resultados melhores.

Na Figura 3.9, observa-se que os escalonamentos para o DAG WIEN2k produzidos pelos escalonadores IPDT-FUZZY e IPDT tiveram características similares às observadas nos escalonamentos para o DAG Montage. O escalonador IPDT-FUZZY projetado com baixos valores de ρ apresenta resultados piores do que o escalonador IPDT. No entanto, para altos valores de incerteza, o escalonador IPDT-FUZZY consegue superar o escalonador IPDT. Para $\rho = 400\%$, o escalonador IPDT-FUZZY produz *speedups* equivalentes ou melhores do que aqueles produzidos pelo escalonador IPDT. Por exemplo, para $x = 400\%$, o *speedup* do escalonador IPDT-FUZZY com $\rho = 400\%$ mostra-se 29,55% maior do que o do escalonador IPDT. O escalonamento produzido pelo escalonador IPDT-FUZZY é melhor do que aquele produzido pelo escalonador IPDT, a partir de um valor de x igual a 100%, para $\rho = 400\%$. O *speedup* do escalonador IPDT decresce mais rápido do que o do escalonador IPDT-FUZZY em função de x.

O gráfico da Figura 3.10 mostra que os escalonamentos produzidos pelo escalonador IPDT-FUZZY para o DAG WIEN2k-modificado apresentaram *speedups* que não seguiram

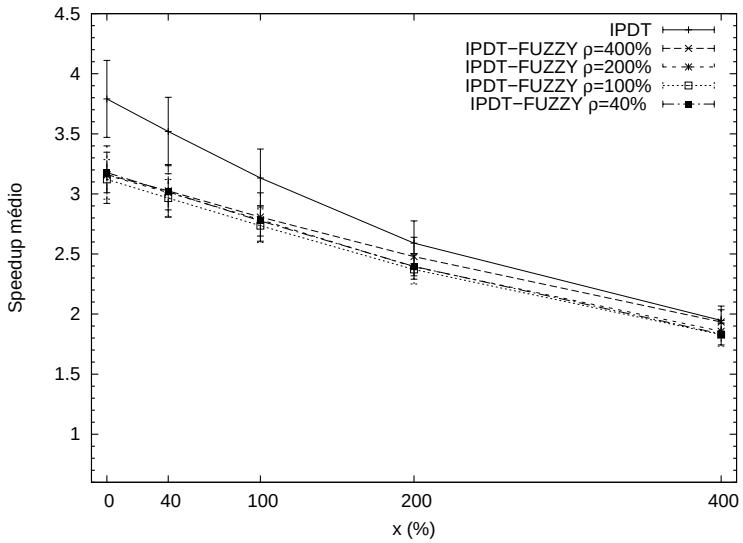


Figura 3.10: *Speedup* médio para o DAG WIEN2k-modificado

o mesmo comportamento observado nos casos anteriores. Isso ocorre por conta da estrutura do DAG. Mesmo para valores altos de ρ, a execução das tarefas em computadores distintos permanece como uma boa opção e o escalonador propõe que as tarefas sejam paralelizadas, entretanto, os aumentos nos tempos necessários para transferir os dados de dependência, por conta das incertezas, aumentam o tempo de execução da aplicação, o que leva a uma diminuição do *speedup*. Pelo fato de haverem poucos níveis no DAG WIEN2k-modificado (Figura 3.5), o atraso da execução de uma única tarefa tem um impacto maior no tempo de execução do DAG como um todo do que se houvessem mais níveis, o que faz com que os escalonadores gerem escalonamentos que usem mais a rede, a fim de tirar proveito do paralelismo. Apesar desse cenário representar aparentemente um caso desfavorável para o escalonador IPDT-FUZZY, é importante observar que, para $\rho = 400\%$, o escalonador produz escalonamentos equivalentes aos do escalonador IPDT, nos casos onde a incerteza ocorrida é maior ou igual a 200%. Além disso, observa-se a robustez dos escalonamentos produzidos pelo escalonador IPDT-FUZZY, dado que o *speedup* do escalonador IPDT decresce mais rápido do que o do escalonador IPDT-FUZZY em função do aumento de x.

Observando os gráficos das figuras 3.8 a 3.10, é possível observar que há pontos de intersecção entre as curvas referentes ao escalonador IPDT-FUZZY e a curva referente ao escalonador IPDT. Esses pontos de intersecção representam os pontos a partir dos quais o escalonador IPDT-FUZZY apresenta vantagem sobre o escalonador IPDT. Como é de se esperar, os ganhos do escalonador IPDT-FUZZY em relação ao escalonador IPDT são menores para valores de x próximos a 0%, pois o escalonador IPDT-FUZZY diminui

a quantidade de tarefas executadas em paralelo na grade, a fim de evitar que as transferências de dados aumentem o tempo de execução das aplicações, dada as incertezas esperadas (ρ). Se as incertezas não se concretizam, não há ganhos em se diminuir a quantidade de tarefas executadas em paralelo, logo o *speedup* do escalonador IPDT-FUZZY mostra-se pior do que aquele do escalonador IPDT, que propõe a execução de tarefas paralelas independente da incerteza esperada. No entanto, quando o valor de x aumenta, as decisões tomadas pelo escalonador IPDT-FUZZY mostram-se vantajosas e os *speedups* deste escalonador aumentam em relação ao escalonador IPDT, gerando, assim, os pontos de intersecção observados nos gráficos.

3.3.2 Utilização da rede

Como nos experimentos realizados, as incertezas esperadas são aplicadas apenas nos pesos dos arcos do DAG, é de se esperar que os ganhos obtidos pelo escalonador IPDT-FUZZY com relação ao escalonador IPDT sejam decorrentes da melhor utilização dos recursos de rede. O escalonador IPDT, por não considerar incertezas, propõe escalonamentos que utilizam os enlaces de rede, na expectativa de que os pesos passados como entrada correspondam às demandas exatas geradas pela aplicação no momento da execução. Caso os pesos mostrem-se menores do que os valores reais, o intervalo de tempo necessário para transferir os bits de dependência torna-se maior do que aquele previsto, o que aumenta as chances de que a aplicação execute em um intervalo de tempo maior, o que, por sua vez, diminui o *speedup*.

Um escalonador projetado para lidar com diferentes valores de incerteza, como o escalonador IPDT-FUZZY, propõe escalonamentos mais robustos através da diminuição do paralelismo das tarefas. Tarefas dependentes e interligadas por arcos com pesos com alta incerteza possuem uma probabilidade alta de serem executadas em um único computador, ao invés de serem executadas em computadores diferentes. Dessa forma, a transferência dos dados de dependência pelos enlaces da grade deixa de existir, diminuindo as chances de aumento no tempo de execução da aplicação.

Avaliar a utilização da rede decorrente das propostas de escalonamento em grades tem importância pelo fato das grades serem ambientes compartilhados. Além de aplicações paralelas, outros usuários que compartilhem recursos das grades podem executar aplicações convencionais. Desse modo, escalonadores que proponham o uso intenso da rede, ou que proponham escalonamentos que utilizem os recursos de comunicação mais do que o previsto, podem causar insatisfação tanto para usuários das grades quanto para usuários localizados em organizações que forneçam recursos para grades.

A Figura 3.11 exibe um gráfico que permite a análise da utilização dos recursos de comunicação da grade em função dos escalonadores executados para o DAG Montage.

Comprovam-se as expectativas em torno da maior utilização da rede por parte do escalonador IPDT.

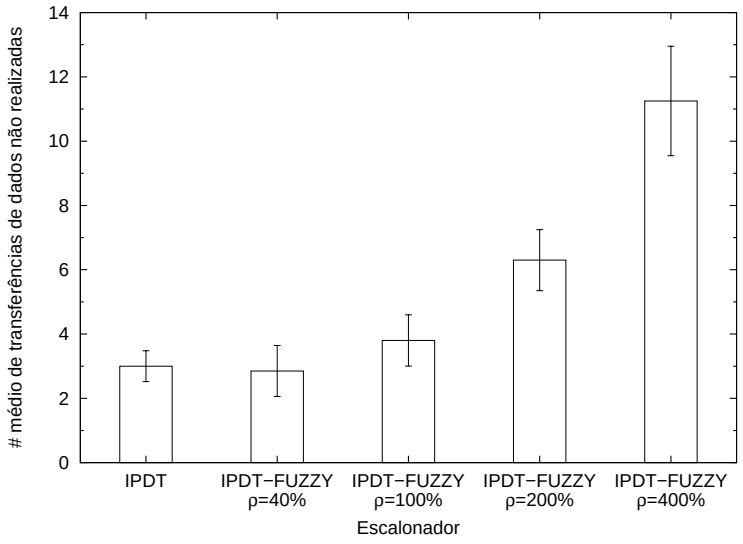


Figura 3.11: Número médio de dependências de dados do DAG Montage que não utilizaram a rede

Os 39 arcos do DAG Montage significam 39 dependências de dados que poderão utilizar os enlaces entre recursos de processamento das grades simuladas. O gráfico da Figura 3.11 plota a quantidade média dessas dependências de dados que **não** foram transferidas via rede para cada um dos escalonadores utilizados nos experimentos. É possível notar que dentre todos os escalonadores, o escalonador IPDT é o escalonador que mais utiliza os recursos de comunicação, visto que, em média, somente três dependências de dados do DAG não são transferidas via rede. Quando o escalonador IPDT-FUZZY foi utilizado, a quantidade de dependências de dados não transferidas via rede mostrou-se diretamente proporcional à incerteza esperada, ρ. O escalonador que apresentou os melhores valores de *speedup* sob altas incertezas, o IPDT-FUZZY com $\rho = 400\%$, foi o escalonador que fez menos uso da rede. Em média, cerca de 11 dependências de dados não foram transferidas via rede, um aumento de cerca de 266,67% em relação aos números apresentados pelo escalonador IPDT.

Os resultados comprovam a expectativa de que escalonadores que lidam com incertezas propõe uma menor utilização da rede do que aqueles escalonadores que não o fazem. Esse comportamento é apresentado a fim de evitar aumentos no tempo de execução, decorrentes de aumentos nos tempos de transferência dos dados de dependência.

3.3.3 Tempo de execução

As figuras 3.12, 3.13 e 3.14 exibem os gráficos que plotam, respectivamente, o tempo de execução médio dos escalonadores para os DAGs Montage, WIEN2k e WIEN2k-modificado.

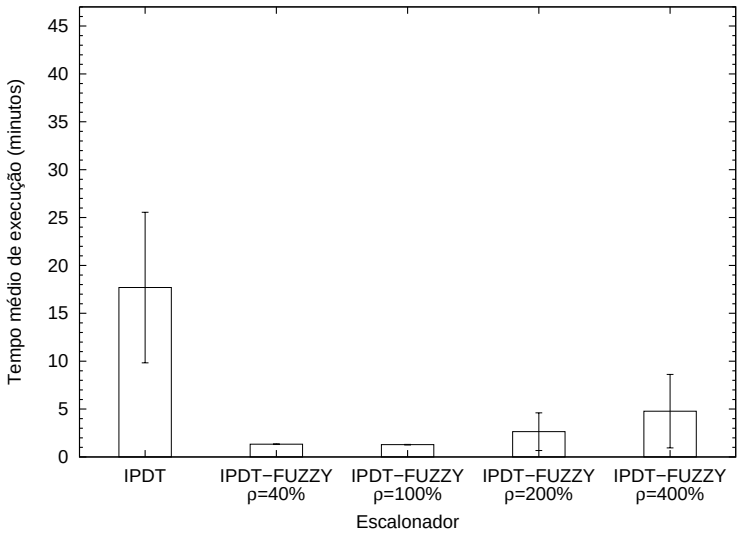


Figura 3.12: Tempo médio de execução para o DAG Montage

Observa-se que o tempo de execução do escalonador IPDT-FUZZY é significativamente menor do que o tempo de execução do escalonador IPDT (aproximadamente 85,08%, 95,67% e 94% menor para os DAGs Montage, WIEN2K e WIEN2k-modificado, respectivamente). Nota-se que o escalonador IPDT-FUZZY, além de fornecer bons resultados quando a expectativa de incerteza é alta, e além de fazer menos uso dos recursos de comunicação, requer menos tempo de processamento do que o escalonador IPDT.

É importante observar que o maior número de restrições, quando comparado com o escalonador IPDT, não afeta o desempenho do escalonador IPDT-FUZZY, o que é relevante quando se modela problemas utilizando programação linear. Mesmo problemas de grande porte com milhares de variáveis e restrições podem ser resolvidos em unidades de segundos ou minutos, como pode ser verificado em [55].

Os baixos tempos de execução do escalonador IPDT-FUZZY podem ser comprovados pela comparação com uma versão do escalonador IPDT que considera os pesos dos arcos dos DAGs aumentados pelos valores de x. Mesmo em uma comparação desfavorável como esta para o escalonador IPDT-FUZZY, pode-se notar, na Figura 3.15, que enquanto o menor tempo de execução do escalonador IPDT é 21,68min, quando ele é executado com um aumento de 50% nos pesos das dependências de dados, o tempo de execução médio do escalonador IPDT-FUZZY com $\rho = 400\%$ é de 2,64min, um valor que é 87,82%

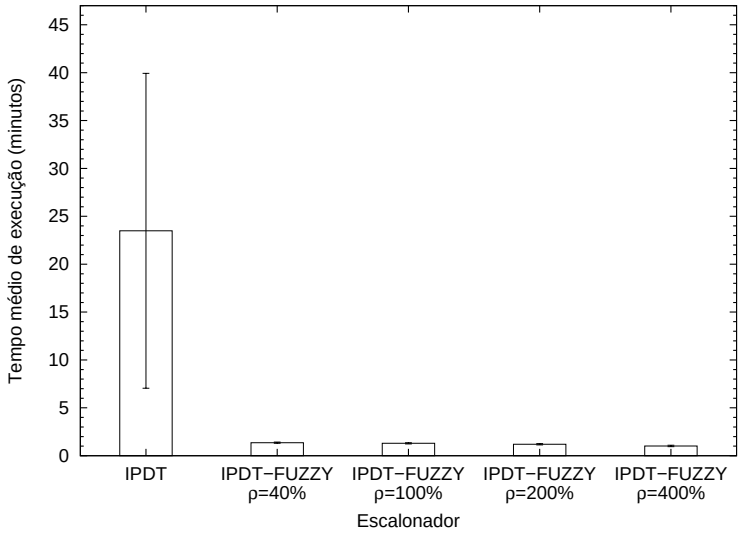


Figura 3.13: Tempo médio de execução para o DAG WIEN2k

menor (O gráfico da Figura 3.15 apresenta valores adicionais de x pois havia o objetivo de encontrar um ponto em que o tempo de execução do escalonador IPDT alcançasse seu mínimo). Comparando o tempo de execução do escalonador IPDT-FUZZY com o do escalonador IPDT executado com um aumento de 400%, observa-se que a diminuição no tempo de execução chega a 97,77%. Além disso, os *speedups* alcançados pelo escalonador IPDT-FUZZY são equivalentes àqueles fornecidos pelo escalonador IPDT, como pode ser observado na Figura 3.16.

3.4 Resumo conclusivo

Aplicações descritas de forma incorreta por usuários de grades podem levar a perdas de desempenho imprevisíveis caso não seja implementado nenhum mecanismo que lide com a incerteza da informação. Neste capítulo, apresentou-se o escalonador IPDT-FUZZY, um escalonador que utiliza técnicas de otimização fuzzy com o objetivo de fornecer escalonamentos robustos frente às incertezas nos pesos de comunicação de DAGs.

Experimentos de simulação realizados com três DAGs executados em várias grades comprovaram a eficácia do escalonador em situações onde há uma expectativa alta de incerteza na descrição dos requisitos de comunicação. O escalonador IPDT-FUZZY mostrou-se robusto em termos de *speedup* produzido, tempo de execução para fornecer o escalonamento e utilização dos recursos de comunicação das grades. Os resultados alcançados pelo escalonador IPDT-FUZZY foram comparados com os resultados do escalonador IPDT que não implementa técnicas de otimização fuzzy. Pelas comparações,

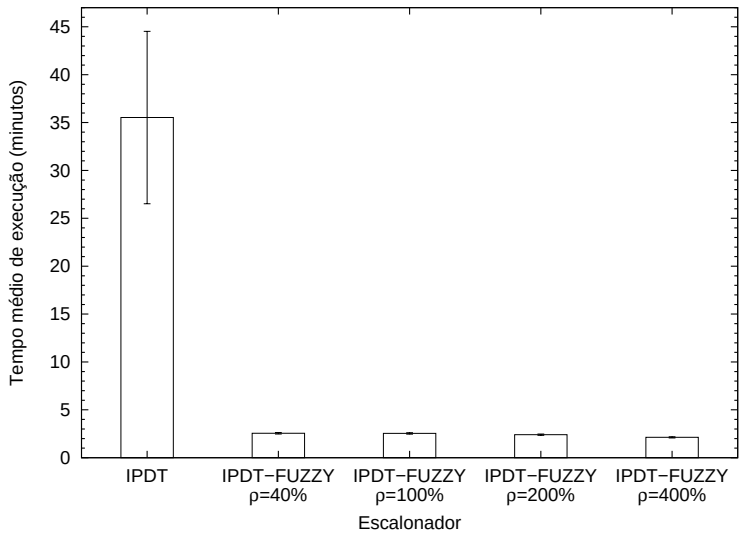


Figura 3.14: Tempo médio de execução para o DAG WIEN2k-modificado

o escalonador IPDT-FUZZY produziu escalonamentos com *speedups* até 29,55% maiores, enquanto o tempo de execução foi até 95,67% menor do que os equivalentes produzidos pelo escalonador IPDT.

O escalonador IPDT-FUZZY foi também comparado com o escalonador IPDT em cenários nos quais o escalonador IPDT sempre tinha acesso às informações corretas sobre as demandas de comunicação das aplicações e em cenários nos quais o escalonador IPDT utilizou uma distribuição uniforme para variar os pesos dos DAGs passados como entrada. Mesmo nos cenários desvantajosos para o escalonador IPDT-FUZZY, observou-se que os *speedups* dos escalonamentos propostos por ele foram tão bons quanto aqueles propostos pelo escalonador determinístico, com a vantagem do tempo de execução ter sido até 97,77% inferior.

Os resultados alcançados com a implementação do escalonador IPDT-FUZZY levaram à generalização do escalonador, a fim de que ele possa lidar com incertezas nas informações sobre a disponibilidade dos recursos da grade. Os resultados decorrentes desta generalização do escalonador IPDT-FUZZY são apresentados no Capítulo 4.

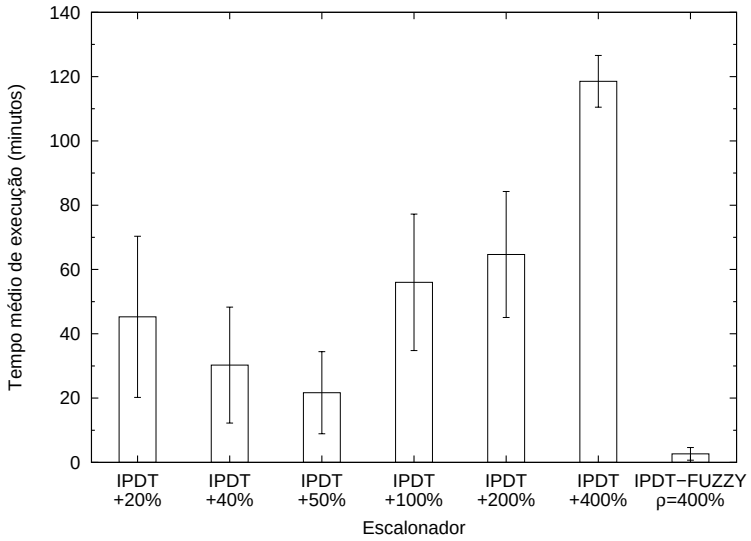


Figura 3.15: Tempo médio de execução do escalonador IPDT (aumentado pelo valor de x) e do escalonador IPDT-FUZZY ($\rho = 400\%$) para o DAG Montage

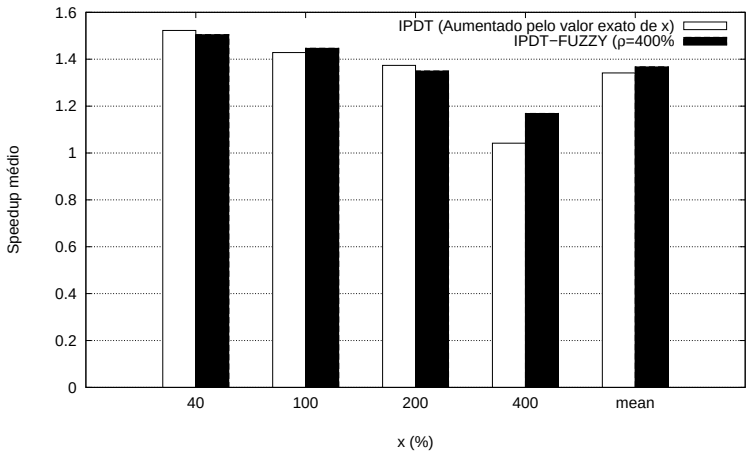


Figura 3.16: *Speedup* médio do escalonador IPDT (aumentado pelo valor de x) e do escalonador IPDT-FUZZY ($\rho = 400\%$) para o DAG Montage

Capítulo 4

Escalonamento sob incertezas na disponibilidade dos recursos

A falta de controle centralizado e o fato das aplicações não se apropriarem de recursos das grades tornam necessária a construção de mecanismos que informem a disponibilidade dos recursos de tempos em tempos para os escalonadores de tarefas. Somente com uma visão atualizada do estado da grade é possível estimar o tempo de execução das aplicações e garantir que os escalonamentos encontrados sejam de fato próximos do ótimo. Porém, não basta monitorar a grade em intervalos de tempo curtos. A qualidade das ferramentas é tão importante quanto o período de monitoramento. O ideal seria que houvessem ferramentas de monitoramento que não injetassem nenhuma carga na grade e que em um intervalo de tempo igual a zero devolvessem informações exatas a respeito da disponibilidade de todos os recursos da grade. Entretanto, tal ferramenta não existe. As ferramentas atuais fornecem estimativas da disponibilidade do recurso e essas estimativas são utilizadas pelos escalonadores para a derivação dos escalonamentos. Os experimentos apresentados na Subseção 2.4.2 ilustraram os problemas que podem ocorrer quando essas estimativas estão incorretas.

Na realidade, mesmo que houvessem ferramentas ideais para o monitoramento dos recursos da grade ainda haveria um problema. Como os recursos são compartilhados por diversos usuários e podem entrar e sair da grade de forma assíncrona, as suas disponibilidades variam com o passar do tempo. Não há nenhuma garantia de que uma estimativa feita no instante de tempo t será válida no instante $t + \Delta t$.

A Seção 2.4 destacou que as incertezas na disponibilidade dos recursos das grades costumam ser tratadas por arcabouços baseados em medições, reescalonamentos e migrações de tarefas. Caso sejam detectadas mudanças significativas no estado dos recursos da grade, a aplicação é reescalonada, o que pode levar à migração de tarefas que já estejam executando. Outra forma de tratar as incertezas no estado dos recursos é através do

monitoramento do tempo de execução das tarefas que compõem as aplicações [66] [30]. Caso as tarefas executem em intervalos de tempo diferentes dos esperados, elas são reescalonadas. Estas soluções consideram que mudanças nos tempos de execução são causadas pela variação na capacidade disponível dos recursos ou por informações incorretas a cerca das demandas das aplicações. Assim como os arcabouços de migração, estas soluções são dependentes de ferramentas de monitoramento precisas.

A utilização de mecanismos baseados em medições, reescalonamentos e migrações causa dois problemas. O primeiro é consequência do monitoramento frequente, dado que a injeção de tráfego na grade aumenta o nível de incerteza (O Capítulo 5 apresenta resultados de medições em ferramentas de monitoramento que comprovam o impacto negativo causado pelo tráfego injetado na rede). O segundo é consequência dos reescalonamentos de tarefas dado que migrações desnecessárias podem aumentar o *makespan* das aplicações.

Como não é suficiente considerar a existência de mecanismos ideais para o monitoramento das grades e como a carga extra da migração pode acarretar no aumento do *makespan* das aplicações, a implementação de escalonadores de tarefas que sejam robustos às incertezas presentes nas informações de disponibilidade deve ser considerada. É importante lembrar, entretanto, que como explicado no Capítulo 3, as incertezas na descrição das aplicações são críticas para o correto funcionamento das grades. Um escalonador de tarefas que tenha mecanismos para lidar tanto com as incertezas na descrição das aplicações quanto com as incertezas nas informações sobre disponibilidade dos recursos representa um grande avanço nos mecanismos de alocação de recursos em grades. Comparando esse novo escalonador com o escalonador IPDT-FUZZY, observa-se que ele teria uma entrada adicional que corresponderia à QoI referente ao estado dos recursos. Dessa forma, o diagrama da Figura 3.1 evolui para aquele apresentado na Figura 4.1.

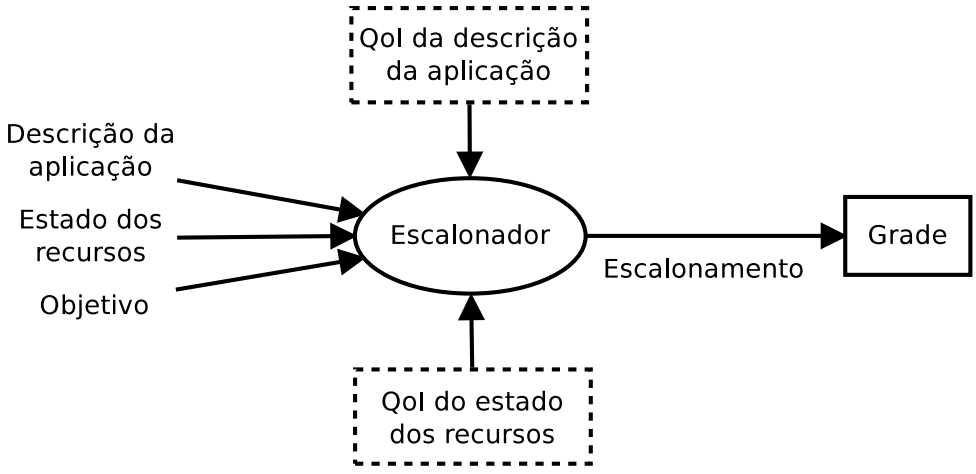


Figura 4.1: Entradas e saída de um escalonador de tarefas para grades com suporte às incertezas na descrição das aplicações e nas informações de disponibilidade dos recursos

O funcionamento de um escalonador de tarefas que implementasse o diagrama ilustrado na Figura 4.1 dependeria de ferramentas de monitoramento que devolvessem as estimativas acompanhadas da QoI, e essas ferramentas existem [43] [3]. Elas devolvem a estimativa de disponibilidade como um intervalo devido às incertezas decorrentes da medição.

A existência de ferramentas que descrevem a disponibilidade dos recursos como intervalos e os ganhos alcançados pelo escalonador IPDT-FUZZY motivaram a implementação de um escalonador de tarefas que considere a descrição da capacidade disponível dos recursos como números fuzzy. Este capítulo apresenta o escalonador IP-FULL-FUZZY que, além de fazer essa consideração, considera as incertezas presentes nas demandas das aplicações. Assim como o escalonador IPDT-FUZZY, a busca pelo melhor escalonamento é formulada através de programação inteira 0–1 e são utilizadas técnicas de otimização fuzzy. O escalonador IP-FULL-FUZZY é comparado, principalmente, com o escalonador IPDT-FUZZY, que serviu de base para a sua construção, e com o escalonador RANDOM, uma versão do escalonador IPDT que relaxa as restrições do problema, a fim de diminuir o tempo de execução. Resultados obtidos através de simulações de três aplicações, as quais foram utilizadas na análise de desempenho do escalonador IPDT-FUZZY, comprovam que o escalonador IP-FULL-FUZZY é robusto frente às incertezas na quantidade de dados transferidos entre tarefas de aplicações e frente às incertezas na largura de banda disponível entre os computadores das grades.

As próximas seções deste capítulo estão organizadas da seguinte forma: a Seção 4.1 resume os trabalhos relacionados com o escalonador IP-FULL-FUZZY. A Seção 4.2 apresenta o escalonador IP-FULL-FUZZY através da descrição do problema de programação inteira 0–1 que o constitui. A Seção 4.3 relata os experimentos de simulação realizados para avaliar o desempenho do escalonador e o capítulo é finalizado com um resumo dos resultados obtidos e com uma conclusão parcial da Tese na Seção 4.4.

4.1 Trabalhos relacionados

O escalonador IP-FULL-FUZZY assemelha-se com o escalonador IPDT-FUZZY quando comparado com outros trabalhos existentes na literatura. A análise de desempenho realizada em [18] é tomada como referência para a avaliação do escalonador e a modelagem apresentada em [30] justifica a utilização de números fuzzy triangulares. O escalonador IP-FULL-FUZZY também está relacionado com os trabalhos resumidos na Subseção 2.4.1. Naquela Subseção observou-se que nenhum trabalho trata as incertezas na descrição das aplicações e na disponibilidade dos recursos de forma preventiva como o escalonador IP-FULL-FUZZY.

O trabalho apresentado em [35] reforça a importância em se tratar incertezas em problemas de escalonamento. Apesar do trabalho não ser específico para o escalonamento

de tarefas em grades ele estuda as implicações em utilizar diferentes números fuzzy em casos nos quais o tempo de execução de tarefas é incerto. A utilização de números fuzzy triangulares é justificada quando não há informações suficientes para reforçar a utilização de outros tipos de números. Assim como em [30], considera-se que existem incertezas nas informações sobre o tempo de execução das tarefas, independente se as suas causas são informações incorretas passadas pelos usuários ou estimativas incorretas fornecidas pelas ferramentas de monitoramento. Essa é uma diferença para a modelagem do escalonador IP-FULL-FUZZY. Como essas incertezas provêem de fontes diferentes (usuários × ferramentas de monitoramento), elas são tratadas separadamente.

A importância em considerar as incertezas na largura de banda disponível é reconhecida no documento publicado em [52] pelo *Open Grid Forum* (OGF), uma organização formada por entidades da indústria e da academia em busca de uma maior difusão e evolução das grades. A dificuldade de predizer a largura de banda disponível é reconhecida em [38] e justifica a implementação de ferramentas que descrevam a largura de banda disponível através de um intervalo de valores. A existência dessas ferramentas garante a utilidade do escalonador IP-FULL-FUZZY.

Dois exemplos de ferramentas de monitoramento que fornecem as estimativas como faixas de valores são o `pathload` [43] e o `abget` [3]. Ambas as ferramentas estimam a largura de banda disponível fim-a-fim entre dois computadores na Internet. As ferramentas enviam fluxos consecutivos de dados a diferentes taxas, durante um curto intervalo de tempo, e medem a variação do atraso em uma única via (OWD – *One Way Delay*) entre os pacotes. Ao invés de fornecer a estimativa como um único número, como feito por ferramentas tradicionais, ambas as ferramentas aplicam algoritmos sobre os dados de variação de atraso, a fim de estimar um intervalo que represente a largura de banda disponível. As saídas do `pathload` e do `abget` podem ser aplicadas diretamente como entrada para o escalonador IP-FULL-FUZZY já que o mesmo espera números fuzzy que podem ser representados como intervalos.

Em resumo, boa parte dos trabalhos existentes na literatura confiam nos mecanismos de monitoramento das grades, agem de forma reativa e não separam as incertezas decorrentes do desconhecimento dos usuários das incertezas decorrentes das imprecisões do monitoramento. O escalonador IP-FULL-FUZZY lida com os dois tipos de incerteza presentes no escalonamento de tarefas em grades. Ele age de forma preventiva, possibilita que o monitoramento da grade seja feito em períodos maiores e diminui a necessidade de migração, diminuindo, assim, a intrusão da gerência na grade e aumentando a disponibilidade dos recursos para as aplicações. Além disso, ele pode ser utilizado na prática com ferramentas de monitoramento existentes, dado que aceita as entradas na forma de intervalos.

4.2 O escalonador IP-FULL-FUZZY

Apesar do escalonador IP-FULL-FUZZY apresentado nesta seção ser projetado para lidar tanto com incertezas referentes ao tempo de processamento nos computadores quanto com incertezas referentes ao tempo de transferência de dados via rede, este capítulo enfatiza os ganhos que o escalonador alcança em situações nas quais hajam incertezas relacionadas com a disponibilidade de largura de banda. Tal foco deve-se ao fato dos trabalhos anteriores negligenciarem as questões referentes à comunicação via rede e pelo fato de existirem ferramentas de estimativa de largura de banda disponível que fornecem as estimativas através de intervalos.

O escalonador IP-FULL-FUZZY baseia-se no escalonador IPDT-FUZZY. A diferença entre eles está no fato do IP-FULL-FUZZY considerar que a disponibilidade de processamento dos computadores, TI, e a disponibilidade dos enlaces da grade, TB, são dadas por números fuzzy, assim como são I e B. O escalonador IP-FULL-FUZZY devolve como saída uma lista dos computadores alocados para cada tarefa, os instantes de tempo em que as tarefas devem ser executadas e os instantes de tempo em que os dados de dependência entre as tarefas devem ser transferidos via rede.

Considera-se que os números fuzzy referentes aos valores de $\widetilde{TI}$ e de $\widetilde{TB}$ são números fuzzy triangulares, para reproduzir situações em que o erro na estimativa do peso pode ser tanto positiva como negativa. Assim, a capacidade de processamento disponível em um computador k na grade é representada, utilizando a notação de números fuzzy, por $[\underline{TI_k}, TI_k, \overline{TI_k}]$ onde $\underline{TI_k} = TI_k(1 - \frac{\chi}{100})$ e $\overline{TI_k} = TI_k(1 + \frac{\chi}{100})$. $\chi\%$ representa a incerteza em torno da capacidade de processamento disponível. De modo similar, a largura de banda disponível entre os computadores k e l é representada por $[\underline{TB_{k,l}}, TB_{k,l}, \overline{TB_{k,l}}]$ com uma incerteza de $\omega\%$. $\underline{TB_{k,l}} = TB_{k,l}(1 - \frac{\omega}{100})$ e $\overline{TB_{k,l}} = TB_{k,l}(1 + \frac{\omega}{100})$.

Os valores de χ e de ω traduzem as incertezas a cerca da disponibilidade dos recursos da grade. Quanto menor a variação na disponibilidade de processamento dos computadores, menor o valor de χ. De forma similar, quanto menor a variação na largura de banda disponível entre os computadores da grade, menor o valor de ω.

Como o escalonador IP-FULL-FUZZY lida com incertezas nas descrições das aplicações, todas as considerações feitas sobre os valores I, B, ρ e σ na Subseção 3.2 continuam válidas. Continua válida, também, a consideração de que a linha do tempo de execução das aplicações é discreta.

Por conveniência, na formulação do problema resolvido pelo escalonador, utiliza-se a notação $\mathcal{T}'' = \{1, \ldots, T''_{max}\}$, onde $T''_{max} = T_{max}(1 + \frac{\sigma}{100})(1 + \frac{\chi}{100})$ e T_{max} é o maior *makespan* possível para o DAG. Na descrição, utiliza-se também o valor T''_{min}, onde $T''_{min} = T_{min}(1 - \frac{\sigma}{100})(1 - \frac{\chi}{100})$ e T_{min} é o menor *makespan* possível para o DAG. Observa-se que T''_{max} e T''_{min} podem ser calculados diretamente pelos pesos do DAG e do grafo da grade.

O escalonador IP-FULL-FUZZY é expresso pela formulação matemática a seguir:

Maximize λ

sujeito a

$$1 - \frac{f_n - T''_{min}}{T''_{max} - T''_{min}} \geq \lambda \qquad \text{(FF1)}$$

$$\sum_{t \in \mathcal{T}''} \sum_{k \in V_H} x_{j,t,k} = 1 \qquad \text{para } j \in V_D; \qquad \text{(FF2)}$$

$$x_{j,t,k} = 0 \qquad \text{para } j \in V_D, k \in V_H, \quad \text{(FF3)}$$
$$t \in \{1, \ldots, \lceil \underline{I_j}\, \underline{TI_k} \rceil\};$$

$$\sum_{k \in \delta(l)} \sum_{s=1}^{\lceil t - \overline{I_j}\ \overline{TI_l} - \overline{B_{i,j}}\ \overline{TB_{k,l}} \rceil} x_{i,s,k}$$
$$\geq \sum_{s=1}^{t} x_{j,s,l} \qquad \text{para } j \in V_D, ij \in A_D, \quad \text{(FF4)}$$
$$\text{para } l \in V_H, t \in \mathcal{T}'';$$

$$\sum_{j \in V_D} \sum_{s=t}^{\lceil t + \underline{I_j}\, \underline{TI_k} - 1 \rceil} x_{j,s,k} \leq 1 \qquad \text{para } k \in V_H, t \in \mathcal{T}'', \quad \text{(FF5)}$$
$$t \leq \lceil T''_{max} - \underline{I_j}\, \underline{TI_k} \rceil;$$

$$x_{j,t,k} \in \{0, 1\} \qquad \text{para } j \in V_D, l \in V_H, \quad \text{(FF6)}$$
$$t \in \mathcal{T}''.$$

A função objetivo e as restrições do problema resolvido pelo escalonador IP-FULL-FUZZY são similares àquelas do escalonador IPDT-FUZZY. As restrições (FF_i) do escalonador IP-FULL-FUZZY equivalem às restrições (F_i) do escalonador IPDT-FUZZY. A diferença entre as formulações está na utilização dos novos limites para o tempo de execução da aplicação na restrição (FF1) (T''_{max} e T''_{min} ao invés de T'_{max} e T'_{min}) e na utilização dos novos limites dos valores de TI e TB nas restrições (FF3), (FF4) e (FF5).

A Seção 4.3 apresentará os detalhes a cerca da implementação do escalonador IP-FULL-FUZZY, bem como os resultados decorrentes dos experimentos de simulação realizados para avaliar o seu desempenho.

4.3 Resultados numéricos

Nesta seção, são apresentados os resultados das simulações realizadas com o objetivo de avaliar o desempenho do escalonador IP-FULL-FUZZY. A avaliação de desempenho é realizada através de comparações com o escalonador IPDT-FUZZY, com o escalonador RANDOM e com os escalonadores clássicos HEFT e CPOP. São comparados o *speedup*, o tempo de execução e a utilização da rede dos escalonadores.

O objetivo da comparação com o escalonador IPDT-FUZZY é mensurar os ganhos decorrentes da consideração das incertezas na disponibilidade dos recursos, dado que essa é a única diferença entre os dois escalonadores. A comparação com o escalonador RANDOM tem por objetivo comparar o escalonador IP-FULL-FUZZY com um escalonador baseado em programação linear que executa rápido, já que o escalonador RANDOM corresponde simplesmente a uma versão do escalonador IPDT que relaxa as restrições de integralidade. Com o relaxamento, as variáveis $x_{i,t,k}$ podem assumir valores reais $\in [0, 1]$ e que são tratadas como as probabilidades de que cada tarefa i finalize sua execução em um dado computador k no instante de tempo t. De posse das probabilidades são realizados 1000 sorteios de valores aleatórios para encontrar o melhor escalonamento. Uma consequência do relaxamento das restrições do escalonador é uma piora na qualidade dos escalonamentos devolvidos. O objetivo da comparação com os escalonadores HEFT e CPOP é devido a eles serem escalonadores clássicos que não empregam nenhuma técnica para lidar com as incertezas. Dessa forma, é possível mensurar os ganhos do escalonador IP-FULL-FUZZY em relação aos escalonadores que são, largamente, utilizados na literatura e que ignoram as incertezas nas informações passadas como entrada.

Como o foco do estudo é nas incertezas referentes à utilização da rede, os valores de σ e χ foram mantidos iguais a zero em todos os experimentos. Como o escalonador IPDT-FUZZY ignora as incertezas presentes na descrição dos recursos, o valor de ω foi ignorado por ele. Como os escalonadores RANDOM, HEFT e CPOP não tratam incertezas, eles ignoraram tanto os valores de ω quanto os valores de ρ. Nos experimentos nos quais foram utilizadas, os valores das variáveis ω e ρ variaram no conjunto {25%,50%,100%,200%} [66] [18].

Os mesmos cenários e o mesmo ambiente computacional descritos na avaliação do escalonador IPDT-FUZZY (Seção 3.3) foram utilizados na avaliação do escalonador IP-FULL-FUZZY. Os processos de simulação foram similares, com a diferença de que nesta avaliação foram utilizados quatro escalonadores ao invés de dois e de que as descrições dos recursos das grades simuladas também foram variadas.

A Subseção 4.3.1 apresenta os resultados obtidos com os experimentos que avaliaram a qualidade do escalonador em termos do *speedup* frente a incertezas nas demandas das aplicações. As Subseções 4.3.2, 4.3.3 e 4.3.4 apresentam, respectivamente, os resultados

obtidos com os experimentos que avaliaram a qualidade do escalonador em termos de *speedup*, melhor utilização da rede na grade e tempo de execução frente às incertezas tanto na disponibilidade dos recursos quanto nas demandas das aplicações. Os intervalos de confiança exibidos nos gráficos apresentados a seguir foram calculados com nível de confiança de 95%.

4.3.1 *Speedup* em função das incertezas nas demandas das aplicações

As Figuras 4.2, 4.3 e 4.4 exibem, respectivamente, os gráficos com o *speedup* médio para os DAGs Montage, WIEN2k e WIEN2k-modificado. Os *speedups* são plotados em função da incerteza ocorrida nos pesos dos arcos dos DAGs. Nos experimentos relatados nesta subseção, não foi considerada incerteza na disponibilidade dos recursos da grade ($\omega = 0$). Como o escalonador IP-FULL-FUZZY teve a sua modelagem baseada no escalonador IPDT-FUZZY e a única diferença entre eles é o tratamento das incertezas na disponibilidade dos recursos, era de se esperar que ambos produzissem os mesmos escalonamentos nestes experimentos, o que ocorreu em todos os gráficos, como pode-se observar. Esse resultado demonstra a precisão do escalonador IP-FULL-FUZZY mesmo quando a disponibilidade dos recursos é conhecida.

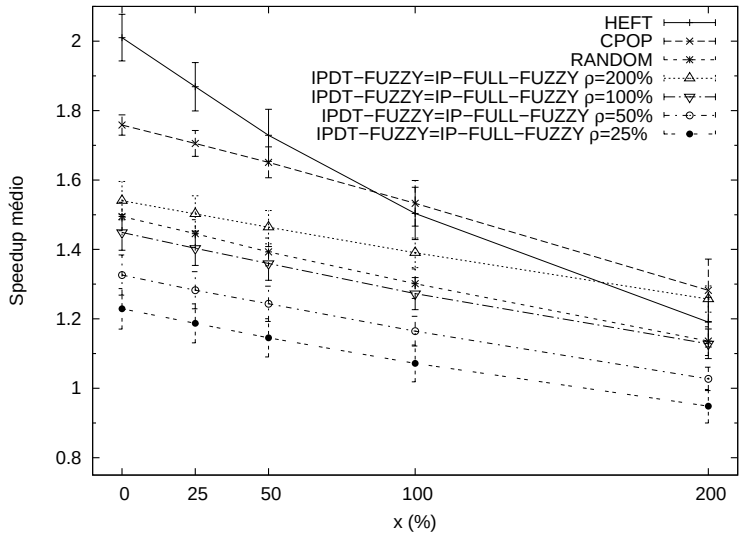


Figura 4.2: *Speedup* médio para o DAG Montage

O gráfico da Figura 4.2 mostra que os escalonadores com suporte a incertezas propõem escalonamentos para o DAG Montage piores do que aqueles propostos pelo RANDOM para valores de $\rho < 100\%$. Quando $\rho = 100\%$, todos esses escalonadores comportam-se de forma similar, entretanto, quando $\rho = 200\%$, os escalonadores IPDT-FUZZY e IP-

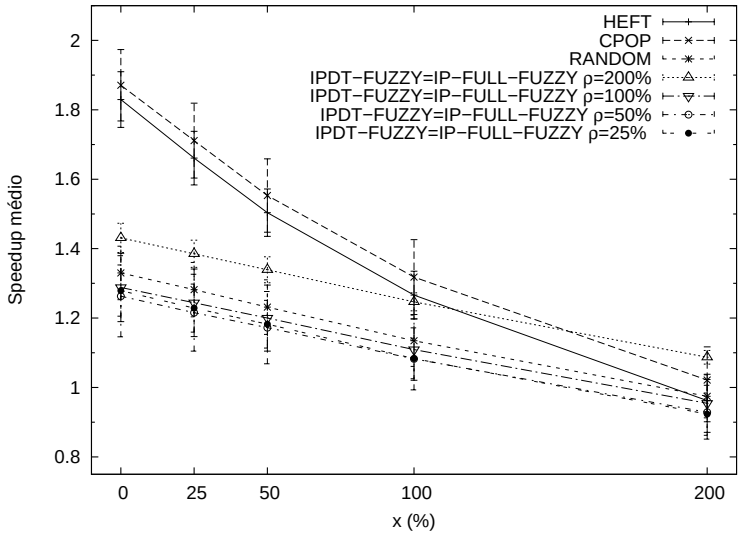


Figura 4.3: *Speedup* médio para o DAG WIEN2k

FULL-FUZZY sugerem escalonamentos com *speedups* maiores do que aqueles sugeridos pelo escalonador RANDOM. Por exemplo, quando $\rho = 200\%$ e $x = 200\%$ o *speedup* dado pelo escalonador IPDT-FUZZY é, em média, 11% maior do que o dado pelo escalonador RANDOM. Nota-se, também, que o *speedup* produzido pelo escalonador RANDOM decai mais rápido, em função do valor de x, do que o *speedup* do escalonador IP-FULL-FUZZY com $\rho = 200\%$, o que mostra que o desempenho do escalonador sem suporte a incertezas degrada mais quando informações incorretas são passadas como entrada.

Comparando os resultados do escalonador IP-FULL-FUZZY com aqueles dos escalonadores CPOP e HEFT, na Figura 4.2, é possível observar que o CPOP e o HEFT fornecem escalonamentos melhores do que o IP-FULL-FUZZY projetado com qualquer valor de ρ, para os valores de x menores ou iguais a 50%. No entanto, para valores de x maiores que 50%, as técnicas de otimização fuzzy do escalonador IP-FULL-FUZZY fazem a diferença e os escalonamentos mostram-se, no mínimo, semelhantes àqueles dos escalonadores HEFT e CPOP. Um ponto importante a ser observado é a taxa com que os *speedups* dos escalonadores HEFT e CPOP diminuem em função de x. Como era de se esperar, e reforçando os resultados preliminares apresentados na Subseção 2.4.2, ambos decaem muito mais rápido do que o *speedup* do escalonador IP-FULL-FUZZY.

Avaliando os escalonamentos do DAG WIEN2k pelo gráfico da Figura 4.3, notam-se resultados semelhantes àqueles observadas no escalonamento do DAG Montage. Os escalonadores IP-FULL-FUZZY e IPDT-FUZZY apresentam resultados melhores do que o escalonador RANDOM quando são projetados com $\rho = 200\%$ e obtêm resultados semelhantes quando projetados com $\rho = 100\%$. Quando $\rho = 200\%$ e $x = 200\%$, o *speedup*

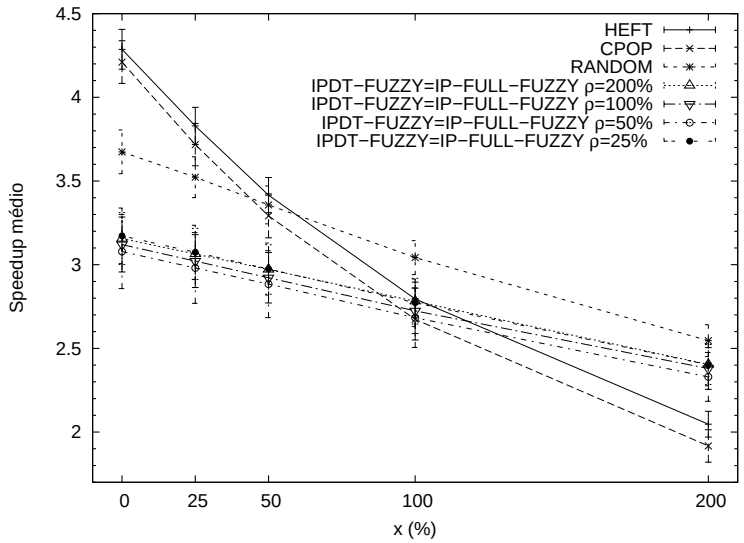


Figura 4.4: *Speedup* médio para o DAG WIEN2k-modificado

produzido pelo escalonador IP-FULL-FUZZY chega a ser 11% maior do que aquele produzido pelo escalonador RANDOM. A comparação com os escalonadores HEFT e CPOP apresenta também resultados similares, ou seja, os escalonamentos ficaram similares a partir de $x = 100\%$ e os *speedups* desses escalonadores decaíram mais rápido do que aqueles do escalonador IP-FULL-FUZZY.

Pelo gráfico da Figura 4.4, que exibe os resultados para o DAG WIEN2k-modificado, observa-se que a diferença entre o decaimento do *speedup* pelo escalonador IP-FULL-FUZZY e o decaimento do *speedup* pelo escalonador RANDOM é bem maior do que o observado nos outros DAGs. Neste caso, o *speedup* do escalonador RANDOM chega a cair 50% mais rápido do que o *speedup* do escalonador IP-FULL-FUZZY. Por conta disso, apesar dos *speedups* do escalonador IP-FULL-FUZZY serem piores do que aqueles do escalonador RANDOM para valores baixos de x, quando este valor aumenta ($x > 100\%$), os *speedups* tornam-se semelhantes. Este comportamento deve-se ao fato de que o DAG escalonado (Figura 3.6) possui apenas três níveis. Dessa forma, imprecisões nos pesos dos arcos tem um efeito imediato no *makespan* da aplicação, dado que a tarefa de saída só pode ser executada depois que todos os dados de dependência tiverem sido enviados.

É possível observar no gráfico da Figura 4.4 que os *speedups* dos escalonadores HEFT e CPOP decaem muito mais rápido do que nos cenários anteriores, o que se deve ao fato do DAG WIEN2K-modificado apresentar um paralelismo maior do que os DAGs anteriores. Tanto o escalonador HEFT quanto o escalonador CPOP propõem escalonamentos que fazem mais uso da rede, entretanto, dadas as incertezas, esses escalonamentos apresentam resultados insatisfatórios; por exemplo, quando $x = 50\%$, todos os escalonadores

propõem escalonamentos similares mas, à medida que x aumenta, observa-se ganhos para o escalonador IP-FULL-FUZZY que chegam a até 25% em relação ao CPOP e a 17% em relação ao HEFT.

Nos experimentos relatados nas próximas subseções, os escalonadores HEFT e CPOP foram executados e os *speedups* dos escalonamentos sugeridos por eles foram piores do que os relatados até aqui. Para facilitar a visualização dos gráficos esses resultados serão omitidos e a análise concentrar-se-á na comparação do escalonador IP-FULL-FUZZY com os escalonadores IPDT-FUZZY e RANDOM.

4.3.2 *Speedup* em função das incertezas na disponibilidade de largura de banda

Em todos os cenários simulados relatados nesta subseção, os escalonadores IPDT-FUZZY e IP-FULL-FUZZY foram executados com a mesma expectativa de incertezas na descrição das demandas de comunicação ($\rho = 50\%$). Optou-se pelo valor de $\rho = 50\%$ com o objetivo de avaliar os escalonadores em situação desfavorável a eles, levando-se em consideração os resultados relatados na Subseção 4.3.1. O escalonador IP-FULL-FUZZY teve as expectativas em torno da incerteza na largura de banda disponível (ω) variadas no conjunto {25%,50%,100%,200%}.

A Figura 4.5 exibe o gráfico que plota o *speedup* médio dos escalonadores em função de diferentes valores de incerteza ocorrida na largura de banda disponível durante o escalonamento do DAG Montage.

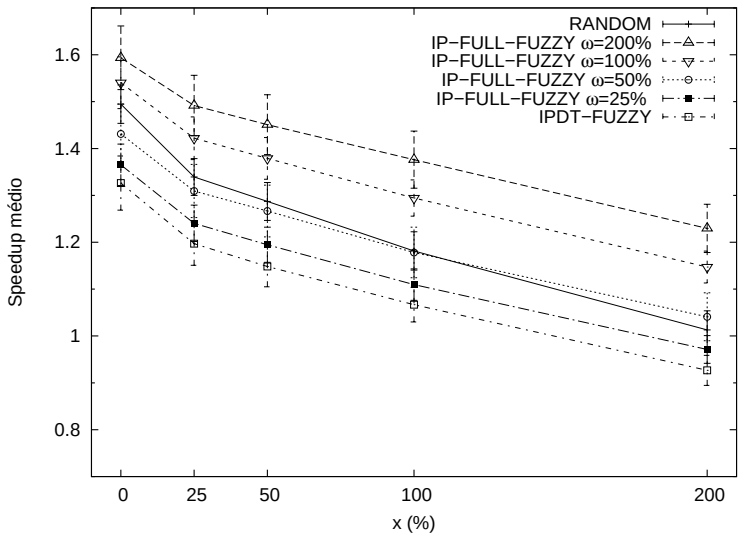


Figura 4.5: *Speedup* médio para o DAG Montage ($\rho = 50\%$)

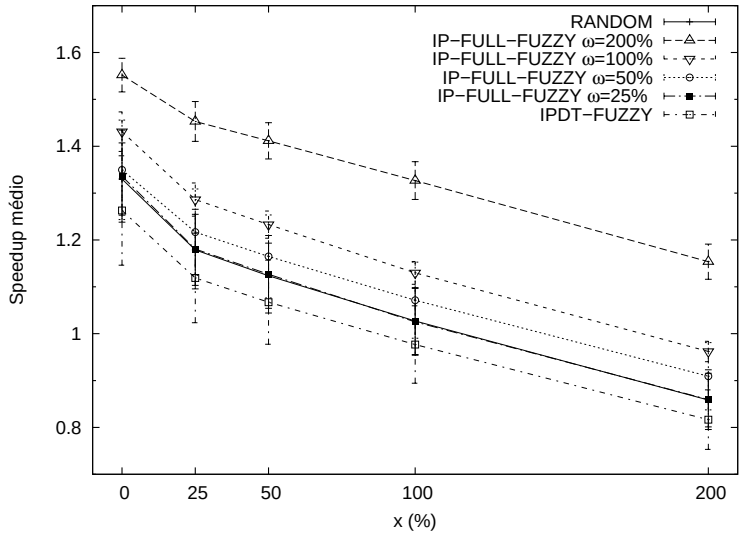


Figura 4.6: *Speedup* médio para o DAG WIEN2k ($\rho = 50\%$)

No gráfico da Figura 4.5, nota-se que os *speedups* produzidos pelo escalonador IPDT-FUZZY são menores do que os *speedups* produzidos por outros escalonadores para qualquer valor de x. Esses resultados comprovam a importância de se considerar as incertezas das capacidades disponíveis. Nota-se, também, que quando o escalonador IP-FULL-FUZZY é executado com $\omega = 25\%$, ele apresenta resultados piores do que o escalonador RANDOM. O resultado é consequência da falta de flexibilidade do escalonador IP-FULL-FUZZY pelo baixo valor de ω. Quando o escalonador IP-FULL-FUZZY é executado com $\omega = 50\%$, os seus escalonamentos assemelham-se àqueles do escalonador RANDOM e quando ele é executado com $\omega > 50\%$, seus escalonamentos mostram-se melhores do que os do RANDOM. Por exemplo, o *speedup* do escalonador IP-FULL-FUZZY com $\omega = 200\%$ chega a ser em média 21% maior do que o do escalonador RANDOM para $x = 200\%$ e 33% maior do que o do escalonador IPDT-FUZZY, nas mesmas condições. Assim como observado nos experimentos em que houveram incertezas na descrição da aplicação, o escalonador RANDOM apresenta uma queda de *speedup* mais acentuada do que o escalonador IP-FULL-FUZZY.

Os resultados encontrados com o escalonamento do DAG WIEN2k estão resumidos no gráfico da Figura 4.6. Pode-se observar que as relações entre os *speedups* dos escalonadores são semelhantes àquelas observadas no escalonamento do DAG Montage. O escalonador IPDT-FUZZY não apresenta bons resultados pelo fato de não ter sido projetado para lidar com incertezas na estimativa de disponibilidade dos recursos. Os *speedups* do escalonador IP-FULL-FUZZY são melhores do que os do escalonador RANDOM quando $\omega > 50\%$, chegando a ser 34% maior do que aqueles do escalonador RANDOM e 41% maior do que

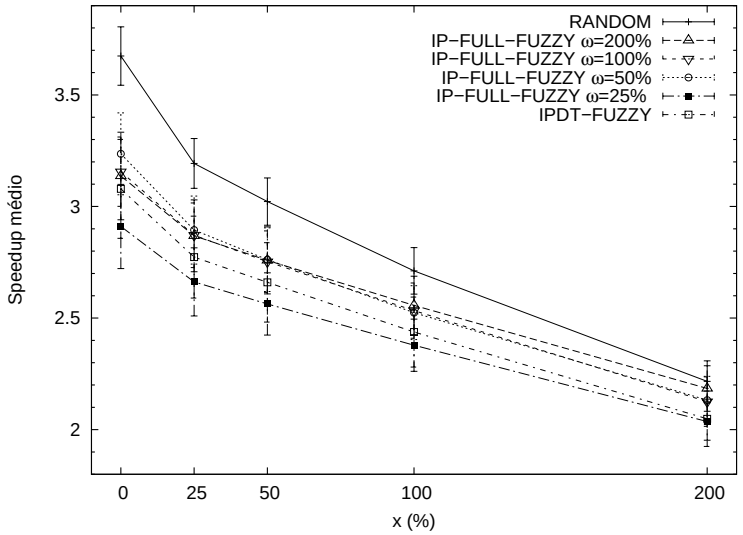


Figura 4.7: *Speedup* médio para o DAG WIEN2k-modificado ($\rho = 50\%$)

aqueles do escalonador IPDT-FUZZY para $x = 200\%$.

A Figura 4.7 exibe os resultados para o DAG WIEN2k-modificado. Nota-se que o decaimento do *speedup* do escalonador RANDOM, um escalonador que não foi projetado para lidar com as incertezas, ocorre 53% mais rápido do que o do *speedup* do escalonador IP-FULL-FUZZY. Desta forma, apesar do *speedup* do escalonador IP-FULL-FUZZY ser pior do que o do escalonador RANDOM quando não houveram incertezas, ambos os valores tornam-se semelhantes quando a incerteza ocorrida torna-se maior. Diferente dos experimentos em que houve incerteza na descrição das aplicações, nos quais os valores tornaram-se semelhantes quando $x = 200\%$, nestes experimentos, os valores tornaram-se semelhantes quando $x = 100\%$.

Os resultados obtidos comprovam a robustez do escalonador IP-FULL-FUZZY em ambientes onde há incertezas na largura de banda disponível entre os computadores das grades. Os resultados comprovam, também, o impacto negativo de não se considerar as incertezas na disponibilidade dos recursos, dado os resultados inferiores produzidos pelo escalonador IPDT-FUZZY.

4.3.3 Utilização da rede em função das incertezas na disponibilidade de largura de banda

Assim como o escalonador IPDT-FUZZY, o escalonador IP-FULL-FUZZY tende a alocar tarefas dependentes para o mesmo computador, quando a estimativa do intervalo de tempo necessário para transferir os dados de dependência é incerta. No caso do escalonador

IPDT-FUZZY, essa incerteza advém do fato das aplicações serem descritas de forma incorreta pelos usuários. No caso do escalonador IP-FULL-FUZZY, a incerteza pode também advir das imprecisões das ferramentas de monitoramento.

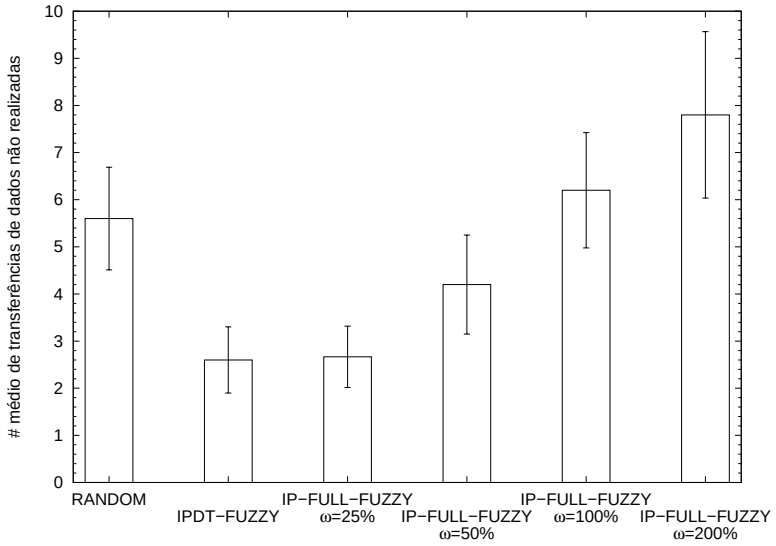


Figura 4.8: Número médio de dependências de dados do DAG Montage que não utilizaram a rede ($\rho = 50\%$)

A Figura 4.8 plota a quantidade de dependências de dados do DAG Montage que não foram transferidas via rede devido à alocação de tarefas dependentes para o mesmo computador. Pode ser visto na figura que a quantidade de dados não transferidos pelos escalonamentos propostos pelo escalonador IP-FULL-FUZZY aumenta com o aumento da incerteza esperada (ω), tornando o escalonador mais robusto à medida que as incertezas aumentam. Por exemplo, quando o escalonador foi executado com $\omega = 200\%$, os dados de $\cong 8$ das 39 dependências do DAG não foram transferidas via rede. Neste caso, o escalonador IP-FULL-FUZZY transferiu duas vezes menos a quantidade de transferências realizadas pelo escalonador IPDT-FUZZY e 0,40 vezes menos a quantidade realizada pelo escalonador RANDOM. A vantagem de ter diminuído a transferência de dados via rede pode ser comprovada através da Figura 4.5, que mostra que o *speedup* do escalonador IP-FULL-FUZZY foi, em média, 33% maior do que o do escalonador IPDT-FUZZY e 21% maior do que o do escalonador RANDOM.

A diminuição da utilização da rede por parte do escalonador IP-FULL-FUZZY em função do aumento de ω também pode ser observado nos gráficos das Figuras 4.9 e 4.10 que exibem, respectivamente, a utilização da rede para os DAGs WIEN2k e WIEN2k-modificado.

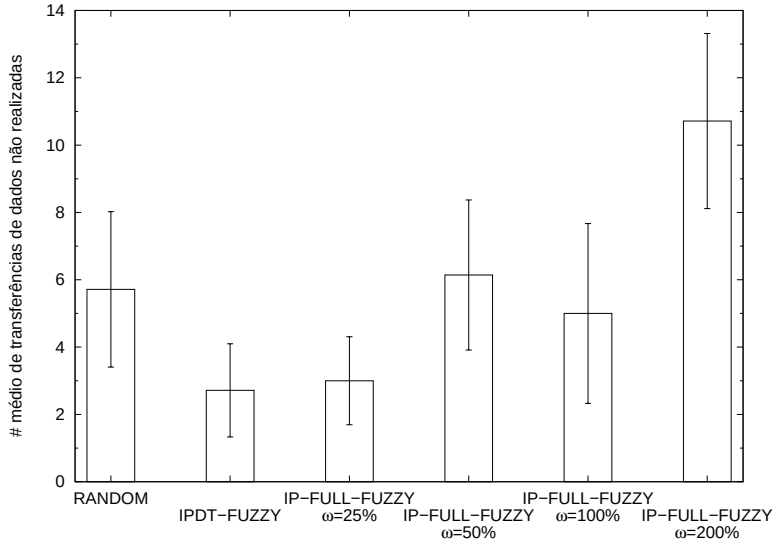


Figura 4.9: Número médio de dependências de dados do DAG WIEN2k que não utilizaram a rede ($\rho = 50\%$)

4.3.4 Tempo de execução em função das incertezas na disponibilidade de largura de banda

As figuras 4.11, 4.12 e 4.13 exibem os gráficos que plotam o tempo de execução médio de todos os escalonadores utilizados nos experimentos para, respectivamente, os DAGs Montage, WIEN2k e WIEN2k-modificado.

Na Figura 4.11, observa-se que o escalonador RANDOM apresenta tempos de execução médios menores do que os outros escalonadores. Esse resultado era esperado, pois a formulação do escalonador RANDOM engloba relaxamento das restrições do problema de programação inteira. Comparando os tempos de execução do escalonador RANDOM com aqueles do escalonador IP-FULL-FUZZY para $\omega \in \{25\%, 100\%\}$, nota-se que o segundo é cerca de 30% maior. Apesar dessa diferença não estar, a princípio, compatível com os ganhos em termos de *speedup*, é preciso ter consciência de que a escala de tempo envolvida no tempo de execução dos escalonadores é bem menor do que aquela das aplicações para grades. Um atraso de pouco mais de 30 segundos no fornecimento do escalonamento é aceitável dado que o ganho final no tempo de execução da aplicação pode chegar a dezenas ou até mesmo a centenas de minutos.

Apesar do escalonador RANDOM fornecer escalonamentos mais rápido do que o escalonador IP-FULL-FUZZY para a maioria dos casos, quando o escalonador IP-FULL-FUZZY é executado com $\omega = 200\%$ para o DAG Montage, o seu tempo de execução mostra-se, em média, 13,25% menor do que o do escalonador RANDOM (Figura 4.11) e 11% menor do que o do escalonador RANDOM para o DAG WIEN2k (Figura 4.12).

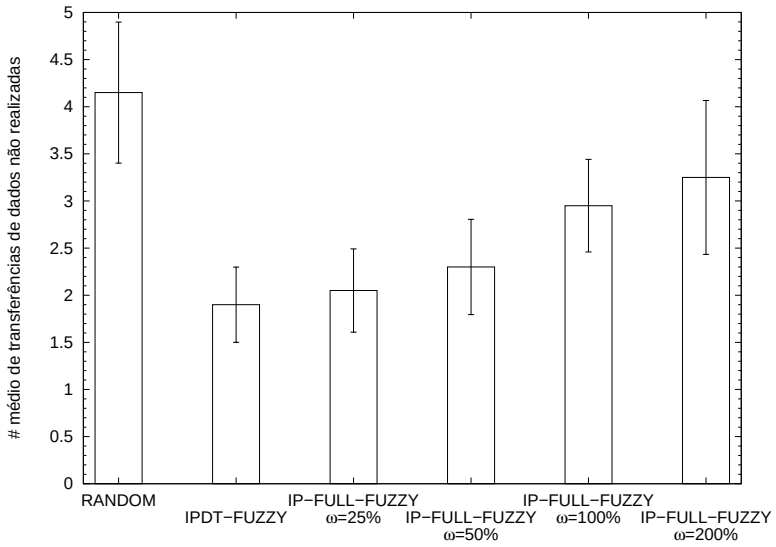


Figura 4.10: Número médio de dependências de dados do DAG WIEN2k-modificado que não utilizaram a rede ($\rho = 50\%$)

No entanto, os resultados obtidos com $\omega = 50\%$ mostram que em certos casos o escalonador IP-FULL-FUZZY pode apresentar complexidade muito maior do que o esperado. De todas as grades simuladas nos experimentos com $\omega = 50\%$, o tempo de execução do escalonador para uma única grade foi uma ordem de grandeza maior do que para as outras grades. Por esse motivo, é importante limitar o tempo de execução do escalonador IP-FULL-FUZZY para evitar perdas de desempenho.

Pelo gráfico da Figura 4.13, nota-se que para todos os valores de ω considerados nos experimentos onde o DAG WIEN2k-modificado foi escalonado, o tempo de execução requerido pelo escalonador RANDOM foi menor do que o do escalonador IP-FULL-FUZZY, entretanto, é importante observar o decaimento do tempo de execução do escalonador IP-FULL-FUZZY com o aumento da incerteza esperada. Esse comportamento favorece a utilização do escalonador em ambientes com altos valores de incerteza mesmo se o requisito considerado for baixo tempo de execução.

Um ponto importante a ser destacado é o fato do tempo de execução médio do escalonador IP-FULL-FUZZY projetado com altos valores de incerteza ($\omega > 50\%$) sempre ser menor do que aquele exigido pelo escalonador IPDT-FUZZY para todos os DAGs utilizados nas simulações.

4.4 Resumo conclusivo

Estimativas incorretas a cerca das largura de banda disponível entre os computadores das grades podem levar a perdas de desempenho, da mesma forma que aplicações descritas

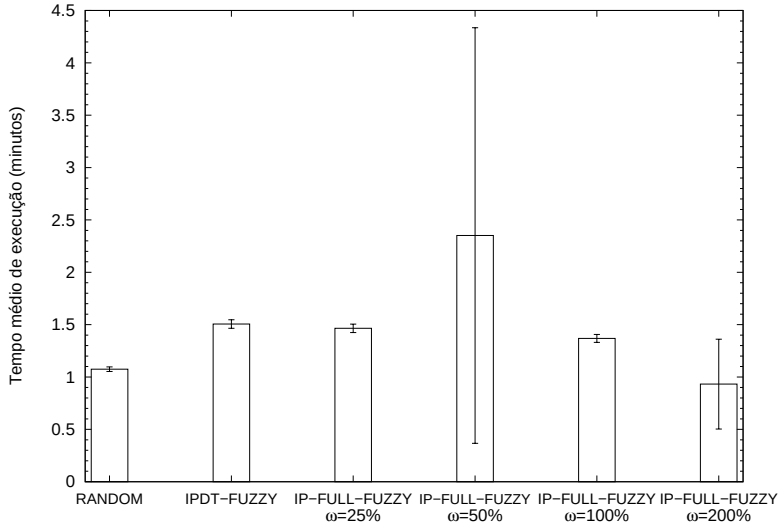


Figura 4.11: Tempo médio de execução para o DAG Montage ($\rho = 50\%$)

de forma incorreta. Neste capítulo, apresentou-se o escalonador IP-FULL-FUZZY, um escalonador que toma como base o escalonador IPDT-FUZZY e consegue lidar tanto com as incertezas nos pesos dos DAGs das aplicações quanto com as incertezas nos pesos dos grafos de topologia das grades.

Experimentos de simulação realizados com vários DAGs e várias grades comprovaram a eficácia do escalonador em situações onde há uma expectativa alta de incerteza na largura de banda disponível entre os computadores. O escalonador IP-FULL-FUZZY mostrou-se robusto em termos de *speedup* produzido e apresentou, em média, um tempo de processamento equivalente àquele do escalonador IPDT-FUZZY, escalonador que lida apenas com as incertezas nas descrições das aplicações.

Em situações onde há incertezas na disponibilidade dos recursos, o *speedup* do escalonador IP-FULL-FUZZY chegou a ser, em média, até 41% maior do que o do escalonador IPDT-FUZZY e até 34% maior do que o do escalonador RANDOM, que aloca os recursos de forma pseudo-aleatória e ignora todas as fontes de incerteza. Além disso, observou-se que o tempo de execução do escalonador IP-FULL-FUZZY pode ser, em média, até 13,25% menor do que aquele consumido pelo escalonador RANDOM, entretanto, é importante definir limites de tempo para a execução do escalonador IP-FULL-FUZZY, para evitar que o tempo necessário para derivação do escalonamento invalide a solução. Os escalonamentos produzidos pelo escalonador IP-FULL-FUZZY foram comparados com aqueles devolvidos pelos escalonadores HEFT e CPOP e observou-se, respectivamente, ganhos de até 25% e 17% no *speedup*.

Os resultados encontrados comprovam o valor de se utilizar técnicas de otimização fuzzy para o escalonamento de tarefas em grades. Conforme explicado anteriormente,

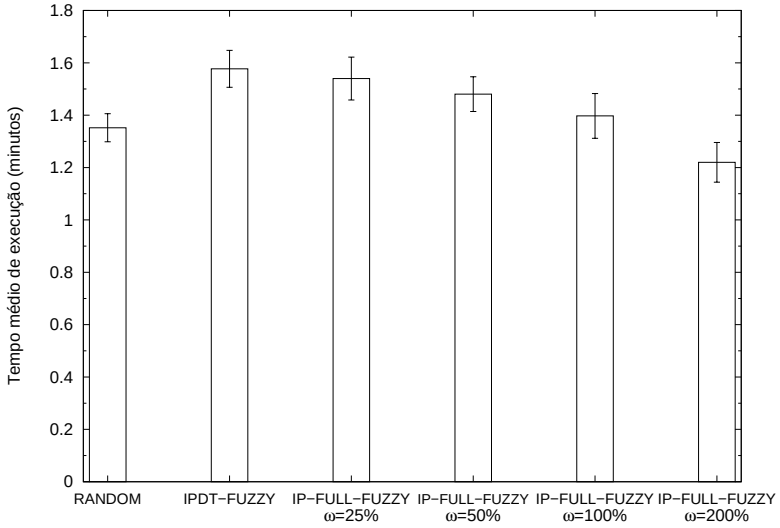


Figura 4.12: Tempo médio de execução para o DAG WIEN2k ($\rho = 50\%$)

para que o escalonador IP-FULL-FUZZY possa ser utilizado na prática é necessário que as ferramentas de monitoramento forneçam estimativas sobre a disponibilidade dos recursos da grade como intervalos e, além disso, apresentem bom desempenho. O Capítulo 5 avalia duas ferramentas que podem ser utilizadas com este fim.

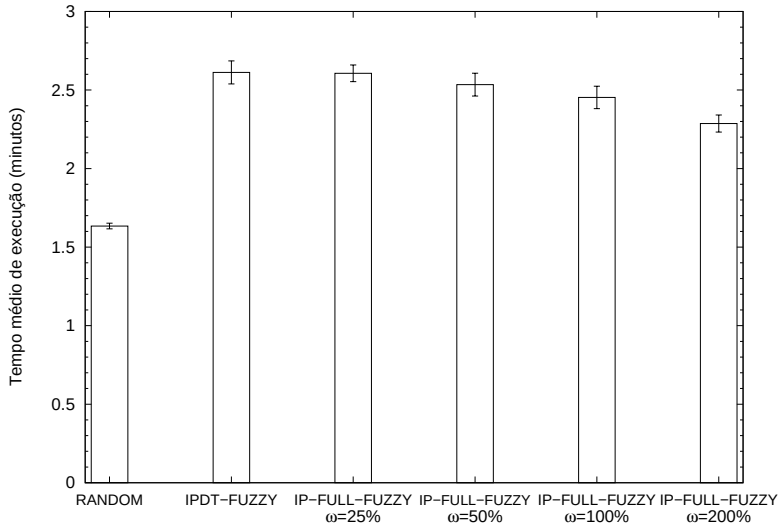


Figura 4.13: Tempo médio de execução para o DAG WIEN2k-modificado ($\rho = 50\%$)

Capítulo 5

Avaliação de estimadores de largura de banda disponível

A estimativa da largura de banda disponível nos enlaces das grades exerce um papel fundamental na qualidade dos escalonamentos de tarefas. Diferente de informações estáticas como a capacidade máxima dos enlaces, a largura de banda disponível é uma métrica dinâmica que precisa ser frequentemente medida. Dentro de uma certa janela de tempo, o valor que representa a largura de banda disponível é incerto dada a dinâmica do ambiente e as imprecisões das ferramentas utilizadas [44] [45]. Ignorar tais incertezas pode aumentar o aumento no *makespan* das aplicações, como demonstrado nos experimentos da Seção 4.3.2, nos quais os escalonamentos que desconsideraram as incertezas, foram responsáveis pelo aumento do *makespan* em até 41%. O ideal, portanto, é encontrar uma faixa de valores que represente a largura de banda disponível nos enlaces em um certo instante de tempo. Essas faixas de valores podem ser diretamente aplicadas no escalonador IP-FULL-FUZZY, que foi apresentado no Capítulo 4.

Existem diversas formas de se obter a largura de banda disponível em enlaces de rede: (i) fazer previsões baseadas no histórico de medições passadas; (ii) medir diretamente nos equipamentos de interconexão; e (iii) utilizar estimadores nos dispositivos finais. Ferramentas baseadas no primeiro método têm sido propostas e vêm sendo utilizadas em grades para armazenar o histórico de medições, como no Ganglia [53], bem como para fazer previsões, como no NWS e no GHS, entretanto, esta abordagem de monitoramento exige um conhecimento prévio de todos os recursos que compõem a grade. Grades formadas por recursos que entram e saem frequentemente não podem utilizá-la, pois a entrada e a saída de recursos afetam o método de medição direta nos dispositivos de interconexão. O problema decorre da necessidade de intervenção de administradores nos dispositivos com o objetivo de disponibilizar as informações para os usuários e para os escalonadores de tarefas via SNMP ou serviços web [80]. O último método consiste em sondar o caminho

entre os nós e, a partir de métricas coletadas, estimar a capacidade disponível no enlace. A vantagem deste método advêm de se poder utilizá-lo sem intervenção em recursos que não sejam os dispositivos finais da grade. Desse modo, torna-se possível descobrir a largura de banda disponível até mesmo para um nó que tenha sido recentemente agregado à grade pela primeira vez.

Dadas as aparentes vantagens em utilizar estimadores de largura de banda disponível em grades, este capítulo tem como objetivo comparar, através de medições, os estimadores `pathload` e `abget`, considerando métricas relevantes para as grades. As duas ferramentas foram escolhidas pelo fato delas devolverem as estimativas como intervalos, que podem ser utilizados diretamente no escalonador IP-FULL-FUZZY. Além da precisão das estimativas, são avaliados o tempo de convergência das ferramentas, a intrusão nos enlaces da grade e o comportamento quando mais de uma instância é executada simultaneamente. O tempo de convergência é uma métrica importante, pois afeta o tempo de execução total da aplicação na grade. Assim como o tempo de execução dos escalonadores de tarefas [12], o tempo de convergência dos estimadores deve ser relativamente pequeno quando comparado com o tempo de execução total da aplicação. A intrusão nos enlaces impacta a disponibilidade dos recursos para outros usuários, pois as grades utilizam recursos que são compartilhados por diversas aplicações. O comportamento quando ocorrem execuções simultâneas da ferramenta é importante dado que esse tipo de situação ocorre com frequência em grades, devido ao comportamento assíncrono dos usuários.

As próximas seções deste capítulo estão organizadas da seguinte forma. A Seção 5.1 apresenta trabalhos relacionados. A Seção 5.2 resume as ferramentas `pathload` e `abget`. A Seção 5.3 descreve os experimentos realizados e discute os resultados obtidos, enquanto a Seção 5.4 finaliza o capítulo com um resumo dos resultados e com conclusões parciais da Tese.

5.1 Trabalhos relacionados

Trabalhos que comparam estimadores de largura de banda passante disponível e avaliam o `pathload` e o `abget` são encontrados em [64], [69], [43] e [3], entretanto, em nenhum desses trabalhos são avaliadas todas as métricas relatadas neste capítulo. A maioria dos trabalhos restringe-se em comparar as ferramentas com relação à precisão das medições.

Devido a heterogeneidade das aplicações e dos usuários, é difícil estimar a largura de banda disponível entre dois nós como um único valor determinístico. O ideal é que a estimativa seja representada como uma faixa de valores. Em [89], apresenta-se uma lista de diversas ferramentas para estimação da largura de banda disponível, entretanto, com exceção do `pathload`, nenhuma das ferramentas fornece estimativas por meio de um intervalo.

Uma ferramenta bastante utilizada por operadores e administradores de redes para estimar a largura de banda disponível é o iperf [87]. O iperf mede a disponibilidade do caminho entre dois computadores de uma forma bastante simples. Cada computador executa um processo local e os processos comunicam-se entre si através do envio de dados (UDP ou TCP), a fim de saturar o caminho entre eles. Ao término do envio mede-se quantos bytes B foram recebidos durante o intervalo de tempo Δt, sendo a largura de banda disponível igual a $\frac{B}{\Delta t}$. A intrusão na rede gerada pelo iperf e o fato da estimativa ser fornecida como uma média fazem desta ferramenta uma opção ineficiente para ser utilizada em grades.

Em [73], apresenta-se uma forma de diminuir a intrusão do iperf quando executado com a opção de enviar dados via protocolo TCP. Utiliza-se o protocolo TCP com modificações que permitem a identificação do momento em que a fase de partida lenta termina. A estimativa do iperf passa a ser realizada a partir desse instante. Como após a fase de partida lenta a vazão alcançada pelo TCP está mais estável, a estimação pode ser realizada muito mais rápida e com menos intrusão. Apesar desta modificação resolver o problema da intrusão, a estimativa continua sendo fornecida através de uma média.

O DIChirp [59] é um estimador de largura de banda disponível que utiliza acesso direto ao hardware das placas de rede para evitar estimativas incorretas que são causadas pelas trocas de contexto dos processos no sistema operacional. O DIChirp estima a largura de banda disponível através do envio de conjuntos de pacotes chamados *chirps* entre os dois computadores envolvidos na estimativa. O processo tem início com o envio dos *chirps* à menor taxa possível. Caso o destinatário detecte aumento no atraso dos pacotes dentro de um *chirp*, considera-se que a taxa de envio daquele *chirp* é maior do que a largura de banda disponível entre os dois computadores. A última taxa utilizada sem variações no atraso é o valor da largura de banda disponível devolvido pelo DIChirp. Assim como o iperf, o DIChirp fornece estimativas através de um único valor, o que o torna ineficiente para ser utilizado em grades.

Em resumo, os trabalhos publicados na literatura com o objetivo de avaliar o desempenho de estimadores de largura de banda disponível não apresentam resultados obtidos para todas as métricas relevantes para grades. Além disso, das várias ferramentas que tem seus códigos disponibilizados na Internet, somente o pathload e o abget reconhecem as incertezas do processo de medição e fornecem as estimativas como intervalos.

5.2 Os estimadores pathload e abget

O pathload e o abget fazem as suas estimativas utilizando uma técnica chamada *Self-Loading Periodic Streams* (SLoPS). São transmitidos diversos fluxos de pacotes a diferentes taxas entre os nós finais. Mede-se o OWD de cada pacote para cada taxa. Se

for detectado aumento do OWD, infere-se que a taxa de transmissão é maior do que a largura de banda disponível; caso contrário, a taxa é menor. As informações a respeito das variações no OWD são trocadas entre os nós finais, a fim de que, através de um algoritmo iterativo, seja encontrado o intervalo que representa a largura de banda disponível no caminho.

Embora as ferramentas utilizem a técnica SLoPS, elas diferem na forma como a mesma é implementada. Quando se deseja estimar a largura de banda do computador A para o computador B utilizando o `pathload`, deve-se executar um processo "remetente" no computador A e um processo "destinatário" no computador B. As informações a cerca das variações no OWD são trocadas entre os computadores através de uma conexão TCP de controle. O "remetente" inicia o processo de estimação enviando 1 fluxo de pacotes UDP a uma taxa inicial $T^{inicial}$. À medida que os pacotes chegam no "destinatário", o OWD é calculado. Caso não seja observado aumento nos OWD dos pacotes, o "destinatário" solicita ao "remetente", pela conexão TCP de controle, que ele aumente a taxa de envio dos pacotes. Vários algoritmos de ajuste são implementados no `pathload` para evitar que variações no OWD que não sejam causadas pela taxa maior do que a largura de banda disponível sejam interpretadas de forma incorreta. Ao final da estimação, o `pathload` fornece como saída o intervalo $[T^{min}, T^{max}]$ que corresponde à largura de banda disponível no caminho entre o computador A e o computador B.

Para estimar a largura de banda do computador A para o computador B utilizando o `abget` basta que seja executado um processo no computador B direcionado para o endereço do computador A, desde que o computador A possua um servidor TCP em funcionamento. Caso não haja tal servidor, o processo do computador B pode ser direcionado para o endereço de algum servidor TCP (por exemplo, um servidor HTTP) que esteja localizado na rede local do computador A. O algoritmo implementado no `abget` simula o funcionamento do protocolo TCP, de modo a controlar a taxa com a qual o computador A envia pacotes para o computador B. O `abget` ignora a implementação padrão do TCP do sistema operacional e manipula os ACKs enviados para A. Falsos ACKs são enviados para A informando o reconhecimento de somente 1 MSS. Ao receber cada ACK, A envia para B um pacote com o tamanho de 1 MSS; portanto, se B enviar os ACKs com um período P, B induz A a enviar dados a uma taxa $T = MSS/P$. À medida que os pacotes de A chegam em B, o OWD é medido e a taxa com que os ACKs são gerados é ajustada, a fim de se encontrar o intervalo $[T^{min}, T^{max}]$ que corresponde à largura de banda disponível.

O `abget` fornece duas opções para a convergência à estimativa da largura de banda disponível. A primeira opção utiliza uma busca binária, enquanto a segunda realiza uma busca linear. Na busca binária, o `abget` multiplica ou divide por 2 a taxa de envio dos pacotes de estimação, caso seja inferido, respectivamente, que a taxa é menor ou maior do que a largura de banda disponível. No caso da busca linear, a alteração do valor da

taxa é realizada através da adição ou subtração de 1 unidade dessa taxa (o `abget` exige que o usuário tenha conhecimento da ordem de grandeza da largura de banda disponível para permitir que a adição e a subtração de 1 unidade sejam realizadas na mesma ordem de grandeza).

As diferenças entre o `abget` e o `pathload` estão resumidas na Tabela 5.1. Comparando as duas ferramentas, é possível observar que a principal vantagem do `abget` está na dispensa da execução de processos em dois computadores. Por outro lado, uma desvantagem advêm da necessidade de ser executado com privilégio de superusuário para que se possa sobrepor a implementação padrão do TCP do sistema operacional. Outra desvantagem é a necessidade do usuário informar muitos valores de parâmetros, a fim de que a convergência para o intervalo correspondente à largura de banda disponível seja realizada de forma rápida. O `pathload` não necessita essas informações dado o uso da conexão TCP de controle, que permite o ajuste fino do SLoPS em tempo de execução. Uma desvantagem do `pathload` é a utilização de pacotes UDP, pois estes podem ser bloqueados nos *firewalls* das organizações. O `abget` não possui esse problema, pois a maioria das organizações já mantém liberada a passagem de pacotes para servidores HTTP, entretanto, a exigência de haver um servidor TCP na rede de cada organização que compõem a grade é uma desvantagem adicional do `abget`.

Tabela 5.1: Comparação entre o `abget` e o `pathload`

Ferramenta	**Acesso aos dois computadores**	**Privilégio de superusuário**	**Configuração de *firewall***
`pathload`	Necessário	Desnecessário	Necessária
`abget`	Desnecessário	Necessário	Desnecessária

5.3 Resultados numéricos

O `pathload` e o `abget` foram avaliados em quatro cenários distintos. No primeiro cenário, relatado na Subseção 5.3.1, os experimentos têm como objetivo avaliar as ferramentas em um ambiente com enlaces de baixa capacidade nominal (10Mbps), enquanto os experimentos no segundo cenário, relatados na Subseção 5.3.2, avaliam as ferramentas em um ambiente com enlaces de alta capacidade nominal (1Gbps). O terceiro e o quarto cenários, respectivamente nas subseções 5.3.3 e 5.3.4, estendem os dois primeiros através da adição de mais enlaces e de mais computadores. Avaliar as ferramentas em todos esses cenários é importante dada a heterogeneidade da capacidade dos enlaces e da quantidade de organizações que compõem as grades.

5.3.1 Cenário com enlaces com baixa capacidade nominal

O primeiro cenário, resumido na Figura 5.1, foi emulado no programa `NCTUns`, um emulador/simulador de redes amplamente utilizado na literatura [76]. O `NCTUns` foi usado pela sua facilidade em criar topologias de redes virtuais e por permitir a interação desta topologia com computadores reais. Os computadores reais podem executar aplicações que utilizem a rede sem necessidade de modificações. Nos experimentos do primeiro cenário, o `NCTUns` foi necessário para que as aplicações acessassem enlaces com 10Mbps de capacidade nominal, ao invés dos 1Gbps da rede sobre a qual os computadores estavam conectados. As estimativas de largura de banda disponível, neste primeiro cenário, foram realizadas do computador cronos e do computador eolo para o computador mnemosyne. Todos os 3 computadores são computadores reais localizados em uma mesma rede local e interligados por uma rede Gigabit Ethernet. O `NCTUns` foi executado na máquina urano, também localizada na mesma rede local. A fim de observar o comportamento dos estimadores, sob diversas condições da rede, utilizou-se dois computadores virtuais no `NCTUns` para gerar tráfego de interferência que afeta a largura de banda disponível entre os computadores reais. A Tabela 5.2 resume as configurações dos computadores envolvidos nos experimentos.

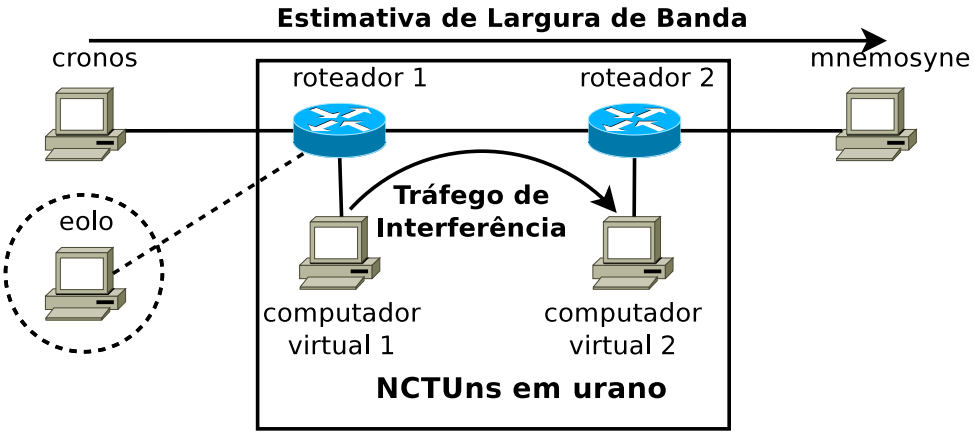


Figura 5.1: Ambiente utilizado para os experimentos com o `NCTUns`

A topologia apresentada na Figura 5.1 é uma topologia *H-hop* com $H = 1$. Nesta topologia a quantidade de enlaces intermediários entre os computadores finais é igual a H. Considerando-se uma numeração sequencial para os enlaces intermediários entre o computador de origem e o computador de destino, o enlace que define a largura de banda disponível no caminho, é sempre o $(H + 1)/2$. Esta topologia é usada para avaliar estimadores de largura de banda disponível em [43] e em [64].

A topologia da Figura 5.1 emula uma grade formada por até 5 *clusters*, ou domínios administrativos, quantidade encontrada em *testbeds* de grades [25] e que está dentro da

Tabela 5.2: Características dos computadores utilizados nos experimentos

Computador	**Processador/Memória**	**Sistema operacional (Linux)**
eolo	Intel Core 2 Quad 2.66GHz / 4GB	Debian kernel 2.6.23.1
cronos	Intel Core 2 Quad 2.40GHz / 4GB	Debian kernel 2.6.23.1
mnemosyne	Dual Xeon 2GHz / 4GB	Debian kernel 2.6.23.1
urano	Intel Core 2 Duo 2.13GHz / 4GB	Fedora kernel 2.6.25.9

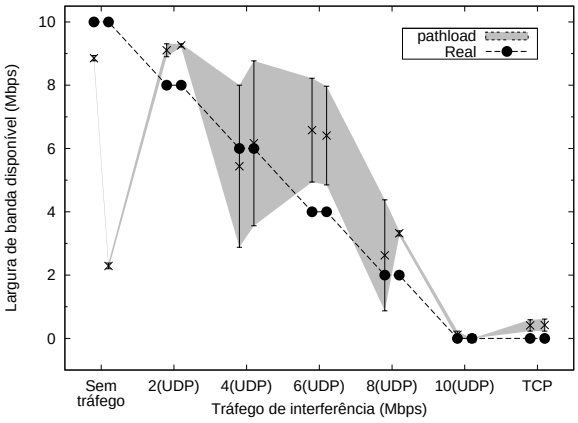
faixa que corresponde à maior quantidade de grades avaliadas em [47] (Foram obtidas informações de diversas grades espalhadas pelo mundo e concluiu-se que 71% das grades agregam de 1 a 10 domínios). Considera-se que os *clusters* representados por cronos, eolo e mnemosyne são os *clusters* disponibilizados para a execução de uma aplicação recém submetida à grade. Deve-se estimar a largura de banda disponível entre esses computadores, a fim de que se possa ter uma estimativa do tempo de execução da aplicação. Os *clusters* representados por "computador virtual 1" e "computador virtual 2" geram tráfego de interferência, a fim de simular aplicações que já estão em execução na grade.

Três métricas foram avaliadas nos experimentos: a precisão, o tempo de execução e a intrusão das ferramentas. Neste primeiro cenário, avaliou-se as ferramentas sem a ocorrência de tráfego de interferência, com tráfego de interferência UDP CBR igual a 2Mbps, 4Mbps, 6Mbps, 8Mbps e 10Mbps e com tráfego de interferência TCP. Como a capacidade nominal dos enlaces foi mantida fixa em 10Mbps, durante todas as medições, uma simples subtração permite o cálculo da largura de banda disponível entre os computadores reais. Numa primeira etapa, as medições foram realizadas somente entre cronos e mnemosyne. Na segunda etapa, as medições foram realizadas, simultaneamente, entre cronos e mnemosyne e entre eolo e mnemosyne. Apenas os resultados mais relevantes são apresentados a seguir. Em todos os experimentos, o `abget` foi executado com os parâmetros referentes à utilização de uma busca binária. Experimentos utilizando a busca linear demandam maior tempo de execução do que a busca binária, podendo o tempo ser até 300% maior (essa informação foi verificada em execuções preliminares da ferramenta, realizadas para definir os parâmetros dos experimentos).

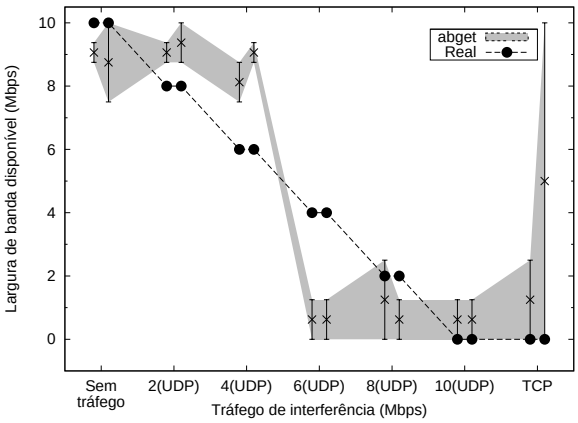
Um detalhe importante referente à configuração dos experimentos diz respeito à desabilitação da coalescência de interrupções (*Interrupt Coalescence* – IC). A IC é uma característica que algumas placas de rede possuem com o objetivo de diminuir a quantidade de interrupções que são geradas para o sistema operacional. Sem a IC habilitada, cada pacote gera 1 interrupção. Com a IC habilitada, interrupções são geradas somente após uma certa quantidade de pacotes ou após um certo intervalo de tempo. Como demonstrado em [62], estimadores de largura de banda disponível baseados em medições do OWD fornecem estimativas incorretas quando executados em máquinas com a IC habili-

tada. O `pathload` possui códigos que garantem seu correto funcionamento em ambientes com a IC habilitada, mas somente se as placas de rede forem de fabricantes específicos. O `abget` não possui nenhum código implementado para lidar com a IC. Assim sendo, a IC foi desabilitada em todas as máquinas utilizadas nos experimentos. Uma consequência da desabilitação da IC é a diminuição da vazão máxima alcançável em redes de alta velocidade ($\geq$ 1Gbps) mas isso não influencia nos resultados que foram obtidos, dado que o interesse é na precisão das ferramentas.

As Figuras 5.2 e 5.3 exibem os resultados obtidos para a primeira etapa do primeiro cenário, na qual os estimadores foram executados somente entre cronos e mnemosyne.



(a) `pathload`



(b) `abget`

Figura 5.2: Estimativas fornecidas (Primeiro cenário)

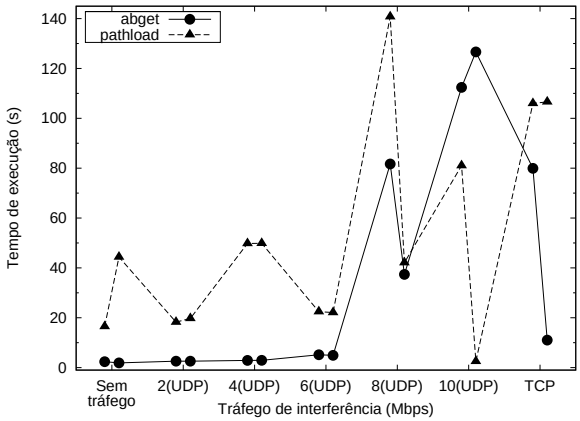
Os gráficos da Figura 5.2(a) e da Figura 5.2(b) apresentam, respectivamente, as estimativas de largura de banda disponível fornecidas pelo `pathload` e pelo `abget` em função do tráfego de interferência. Os valores no eixo horizontal estão ordenados de forma di-

retamente proporcional ao impacto na disponibilidade do caminho. A curva denominada "Real" apresenta a largura de banda disponível real entre os computadores. As áreas cinzentas denominadas "pathload" e "abget" apresentam os intervalos que foram devolvidos pelas ferramentas. Para cada configuração de tráfego de interferência as ferramentas foram executadas duas vezes seguidas. Os resultados mostram que as estimativas do `pathload` ficaram mais próximas dos valores reais quando havia tráfego de interferência na rede. Ao contrário das estimativas do *pathload*, as estimativas do `abget` não seguiram o comportamento da curva de disponibilidade de banda real do caminho entre cronos e mnemosyne. Com tráfego de interferência $\leq$ 4Mbps, o `abget` estimou que o caminho estava praticamente 100% disponível, enquanto que com tráfego de interferência $\geq$ 6Mbps, as estimativas informaram que o enlace estava praticamente 100% ocupado. É importante observar que com tráfego de interferência TCP, os intervalos das estimativas do `abget` foram, em média, maiores do que as outras configurações de tráfego de interferência.

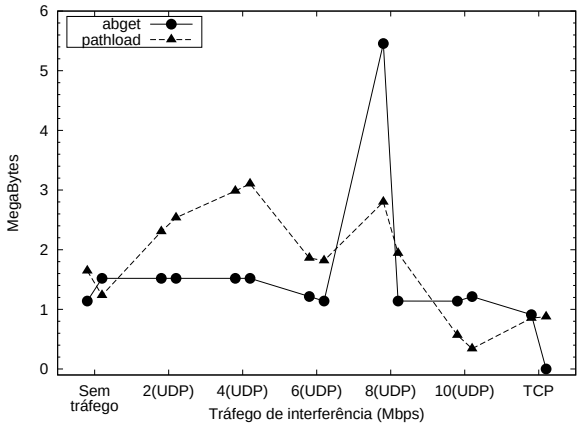
O tempo de execução dos estimadores está exibido no gráfico da Figura 5.3(a). Pelo fato do `abget` não possuir um ajuste inicial da taxa de transmissão como o `pathload` possui, ele precisa transferir uma mesma quantidade de dados no início da estimação, independente da disponibilidade dos enlaces. À medida que os enlaces mostram-se mais ocupados, o tempo de execução tende a aumentar, como pode-se observar na curva "abget". No entanto, como o `abget` foi executado com a opção de uma busca binária, ele conseguiu convergir para um resultado mais rápido do que o `pathload` quando a interferência nos enlaces foi baixa ($\leq$ 8Mbps). O baixo tempo de execução do `abget` em uma das execuções com tráfego de interferência TCP deve-se ao fato de que a ferramenta não conseguiu estabelecer conexão com o servidor web de mnemosyne devido o intenso tráfego de interferência.

As informações referentes à intrusão das ferramentas estão exibidas no gráfico da Figura 5.3(b). Neste gráfico, mostra-se a quantidade de bytes que foram injetados por cada ferramenta durante sua execução. Observa-se que o comportamento do `abget` é mais estável. O `pathload` tende a diminuir o tráfego gerado à medida que a utilização dos enlaces aumenta, pois as taxas de transmissão do `pathload`, a cada iteração do seu algoritmo, não são definidas de acordo com uma busca binária como no `abget`. Após uma iteração, o `pathload` pode diminuir bastante a sua taxa e injetar menos tráfego na rede do que o `abget`.

Alguns dos resultados obtidos para a segunda etapa do primeiro cenário, quando os estimadores foram executados entre cronos e mnemosyne e entre eolo e mnemosyne, estão exibidos nos gráficos das Figuras 5.4 e 5.5. Os resultados referentes à intrusão das ferramentas foram similares àqueles obtidos com apenas uma instância dos estimadores em execução e, consequentemente, não são exibidos. É importante observar que para que se possa realizar as execuções simultâneas do `pathload` foi necessário modificar o seu



(a) Tempo de execução



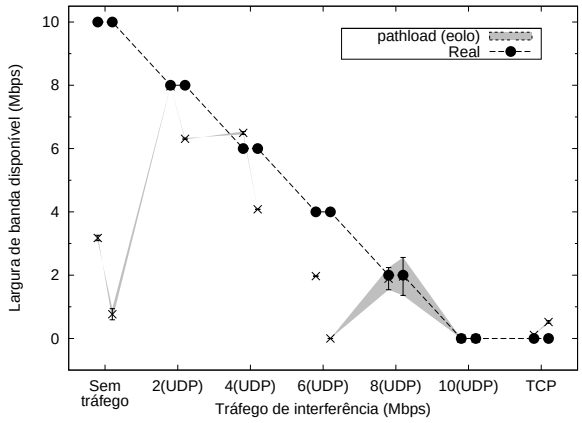
(b) Intrusão

Figura 5.3: Desempenho (Primeiro cenário)

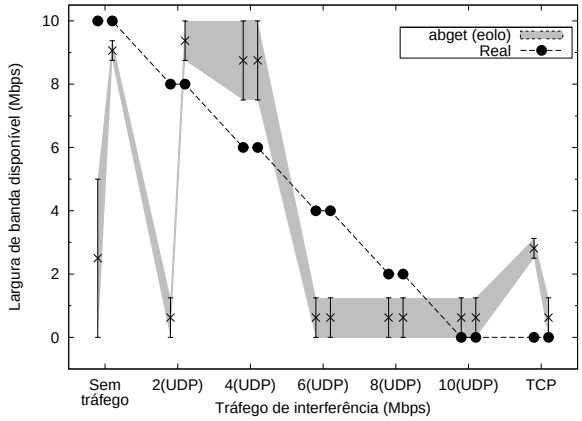
código-fonte, já que, no código original, as portas utilizadas pelo programa são definidas de forma estática e sem suporte à concorrência.

Os gráficos das Figuras 5.4(a) e 5.4(b) exibem as estimativas do `pathload` e do `abget` quando são executados entre os computadores eolo e mnemosyne. Os resultados obtidos entre os computadores cronos e mnemosyne foram semelhantes e por isso não são apresentados. O `pathload`, mais uma vez, fornece melhores resultados do que o `abget`, principalmente quando há tráfego de interferência. A principal diferença dos resultados do `pathload`, com relação aos resultados da primeira etapa, vem do fato dos intervalos serem menores nesta segunda etapa. A combinação dos tráfegos UDP das duas instâncias do estimador em execução torna a disponibilidade do enlace menor, reduzindo, assim, o intervalo conforme corretamente detectado pelo `pathload`.

O principal resultado referente aos tempos de execução (Figura 5.5) diz respeito ao



(a) `pathload`



(b) `abget`

Figura 5.4: Estimativas fornecidas com execuções simultâneas em eolo (Primeiro cenário)

crescimento mais acentuado do tempo de execução do `abget`. Assim como nos gráficos que apresentam as estimativas geradas pelas ferramentas, exibe-se, somente, o tempo de execução entre os computadores eolo e mnemosyne, pois os resultados obtidos entre cronos e mnemosyne foram similares. A combinação dos tráfegos das duas instâncias em execução amplia o tempo de execução do estimador a partir de 4Mbps (UDP), ao contrário do crescimento mais intenso a partir de 8Mbps (UDP), quando apenas uma instância estava em execução.

De um modo geral, neste primeiro cenário, observou-se estimativas mais precisas por parte do `pathload` do que do `abget`, embora o `abget`, na maioria das vezes, tenha executado mais rápido e tenha apresentado um comportamento mais estável em termos de intrusão. Ambos os estimadores executaram relativamente rápidos ($<$ 2min20seg), considerando-se o tempo de vida típico de aplicações em grades que levam horas para

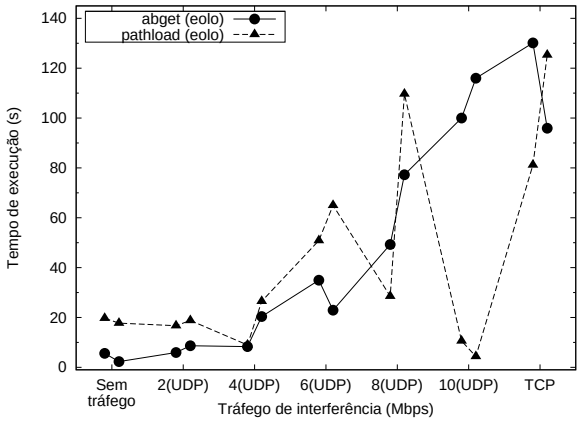


Figura 5.5: Tempo de execução com execuções simultâneas em eolo (Primeiro cenário)

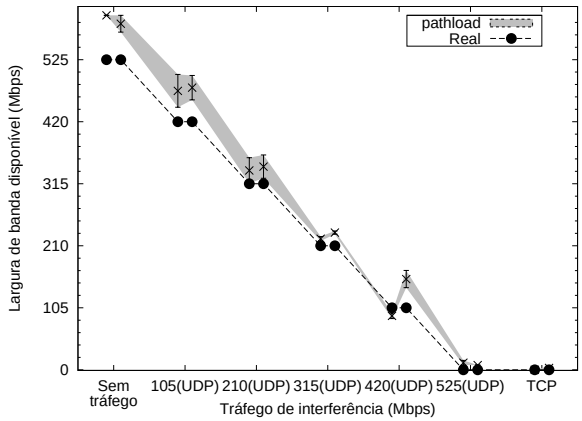
executar, e injetaram poucos bytes na rede (< 6MB).

5.3.2 Cenário com enlaces com alta capacidade nominal

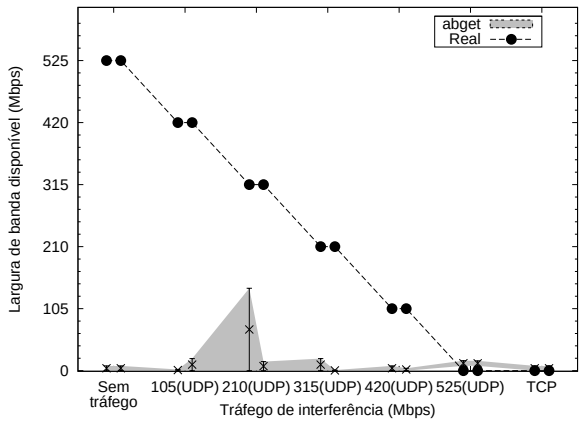
O segundo cenário construído para avaliar o `pathload` e o `abget` difere do primeiro cenário pelo fato do `NCTUns` não ser utilizado. As ferramentas foram avaliadas diretamente nos enlaces reais de 1Gbps da rede local que interliga os computadores. Sem o `NCTUns` para gerar o tráfego de interferência entre os computadores, utilizou-se o programa `iperf` com esta finalidade. O `iperf`, além de ser um programa utilizado para devolver estimativas de disponibilidade, também pode ser utilizada para gerar tráfego a taxas específicas. Os computadores e os roteadores virtuais foram criados em urano através de interfaces de rede virtuais. As estimações foram realizadas entre os mesmos computadores descritos no primeiro cenário. Apesar dos enlaces terem capacidade nominal de 1Gbps, a capacidade real observada foi de 525Mbps. Considera-se, portanto, que esta é a capacidade nominal dos enlaces. As mesmas métricas e as mesmas etapas do primeiro cenário valem para este segundo cenário. A diferença entre os cenários refere-se às taxas dos tráfegos de interferência UDP utilizados, que foram 105Mbps, 210Mbps, 315Mbps, 420Mbps e 525Mbps.

A precisão das ferramentas, quando as estimativas foram realizadas somente entre cronos e mnemosyne, está exibida nos gráficos da Figura 5.6. As estimativas do `pathload` (Figura 5.6(a)) são melhores do que as estimativas do `abget` (Figura 5.6(b)). Diferentemente do primeiro cenário, as estimativas do `pathload` seguem claramente o padrão de disponibilidade dos enlaces. De forma semelhante ao primeiro cenário, as melhores estimativas do `pathload` foram fornecidas quando havia mais tráfego de interferência na rede.

O `abget` apresentou um comportamento diferente em relação ao primeiro cenário. Apesar dos enlaces fornecerem mais capacidade de transmissão, o estimador informou que o enlace estava praticamente 100% ocupado independente do nível de tráfego de interferência. É importante esclarecer que o parâmetro que define a maior taxa que o `abget` deve estimar foi modificado para 525Mbps, ao invés dos 10Mbps utilizados no primeiro cenário, entretanto, não houve melhorias nas estimativas fornecidas. Diversos ajustes foram realizados nos demais parâmetros da ferramenta, em busca de melhores resultados, porém não se obteve sucesso. Destaca-se a quantidade de parâmetros do `abget` como um ponto negativo da ferramenta em grades nas quais novos recursos estejam constantemente sendo incorporados.



(a) `pathload`

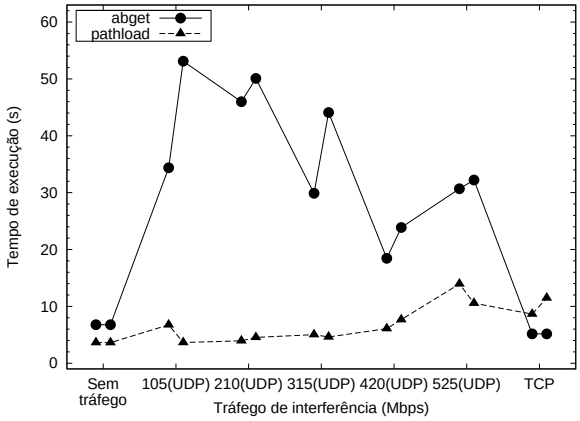


(b) `abget`

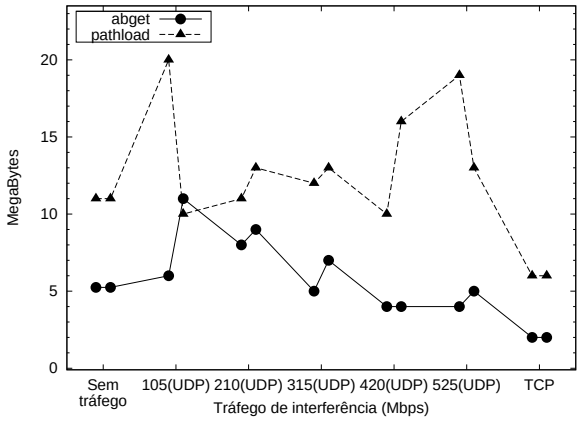
Figura 5.6: Estimativas fornecidas (Segundo cenário)

Observa-se na Figura 5.7(a) uma inversão do padrão do tempo de execução, em relação ao primeiro cenário. Os tempos de execução do `abget` foram, em média, bem maiores do

que os tempos de execução do `pathload`. O `pathload` só executou por mais tempo que o `abget` quando havia tráfego de interferência TCP. Observando as saídas das execuções do `abget`, nota-se que o intervalo de tempo despendido em cada uma das taxas durante a busca binária é muito maior do que aquele despendido pelo `pathload`. Enquanto o `pathload` interrompe o envio de pacotes em uma certa taxa e passa para a taxa seguinte quando se detecta estabilidade no atraso entre os pacotes iniciais, o `abget` mantém sempre uma quantidade constante de pacotes enviados para cada taxa. O aumento da quantidade de valores de taxas a serem avaliadas pelo `abget` neste segundo cenário é responsável pelo maior tempo de execução da ferramenta quando comparada ao `pathload`.



(a) Tempo de execução



(b) Intrusão

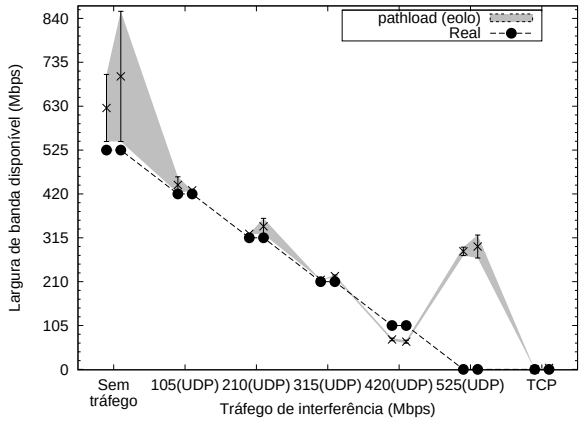
Figura 5.7: Desempenho (Segundo cenário)

Na Figura 5.7(b), observa-se que o `pathload` injetou mais tráfego na rede do que o `abget` e que o `abget` apresentou uma menor variação da quantidade de bytes enviados do que o `pathload`. Comparando os resultados da Figura 5.7(b) com os da Figura 5.3(b),

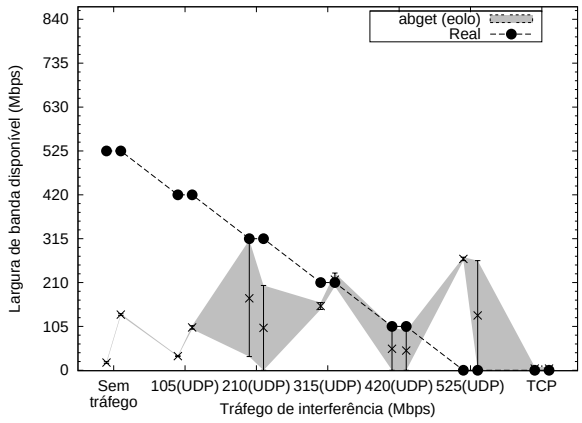
nota-se que o nível de intrusão aumentou proporcionalmente ao aumento da capacidade dos enlaces (10Mbps → 522Mbps).

As principais diferenças entre os resultados observados com a execução de duas instâncias simultâneas das ferramentas e os resultados observados com a execução de uma única instância podem ser observadas nos gráficos das Figuras 5.8 a 5.10. Assim como nos experimentos realizados no `NCTuns`, os resultados obtidos entre os computadores eolo e mnemosyne e entre cronos e mnemosyne foram semelhantes e por isso somente os primeiros são exibidos.

Observa-se que os intervalos fornecidos pelo `abget` (Figuras 5.8(a) e 5.8(b)) são maiores e que o `pathload` fornece estimativas bem diferentes da disponibilidade real, quando os enlaces estão mais congestionados. Além disso, observou-se, também, o aumento do intervalo fornecido pelo `pathload`, quando não houve tráfego de interferência na rede.



(a) `pathload`



(b) `abget`

Figura 5.8: Estimativas fornecidas com execuções simultâneas em eolo (Segundo cenário)

Nota-se, também, o crescimento do tempo de execução do `abget` quando a rede está mais congestionada (Figura 5.9). O tempo de execução chega a aumentar cerca de 1 ordem de grandeza quando o tráfego de interferência variou entre 420Mbps e 525Mbps UDP.

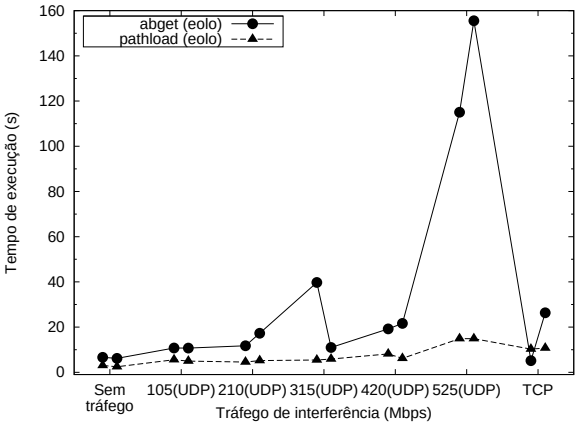


Figura 5.9: Tempo de execução com execuções simultâneas em eolo (Segundo cenário)

Destaca-se o comportamento do `pathload` quando executado em cronos e com tráfego de interferência de 105Mbps (Figura 5.10). Em uma das execuções, o estimador injetou cerca de 300MB na rede, uma quantidade que não pode ser ignorada dadas as estatísticas sobre transferência de dados por aplicações em grades apresentadas em [47] (55% das aplicações estudadas transferem de 1 a 100MB). Esse resultado indica a possibilidade das ferramentas serem muito intrusivas, justificando, portanto, a implementação de procedimentos que evitem altos níveis de intrusão. Uma solução simples para este problema consiste em solicitar do usuário um valor máximo para o tempo de execução da ferramenta ou para a quantidade de bytes que podem ser enviados. Apesar do critério de parada sugerido poder gerar estimativas piores, o resultado final pode ser atraente ao se considerar todas as métricas envolvidas.

Em resumo, pelo observado nos resultados do segundo cenário, conclui-se que o `pathload` fornece bons resultados mesmo em um ambiente formado por enlaces com alta capacidade de transmissão. Nestes ambientes, o tempo de execução do `pathload` é inferior ao do `abget`, apesar do `abget`, em média, ser menos intrusivo. Escalonadores que utilizassem o `abget` para estimar a largura de banda disponível, neste cenário, produziriam escalonamentos que subutilizariam a rede, já que as estimativas incorretas de que os enlaces estão congestionados fariam os escalonadores proporem escalonamentos que não transfeririam dados via rede. Observou-se, também, que o `pathload` tem o potencial de inundar a rede, afetando, consequentemente, o funcionamento das demais aplicações em execução.

De um modo geral, considerando todas as configurações do segundo cenário e o con-

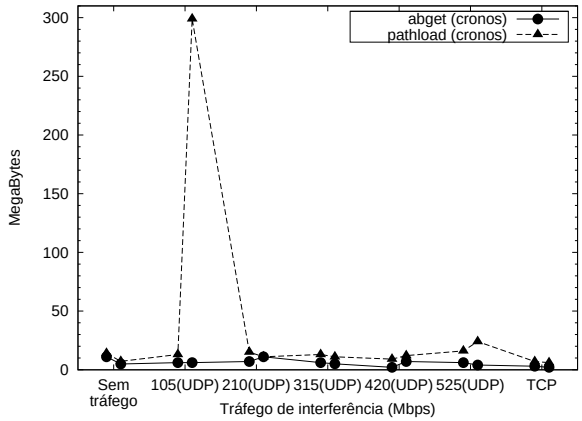


Figura 5.10: Intrusão com execuções simultâneas em cronos (Segundo cenário)

junto de métricas avaliadas, o `pathload` mostrou-se a melhor ferramenta. No entanto, usuários e administradores de grades devem ter noção da possibilidade de interferência nos demais fluxos da rede durante os procedimentos de estimação de banda. Além disso, o fato do `pathload` ter que ser executado em ambos os computadores considerados na estimação é um ponto negativo que motiva modificações no `abget` para que suas estimativas sejam tão precisas quanto as do `pathload`. Há, também, a necessidade de modificação do código do `pathload` para que ele lide com conexões concorrentes, permitindo, assim, a sua utilização simultânea entre diversos computadores da grade.

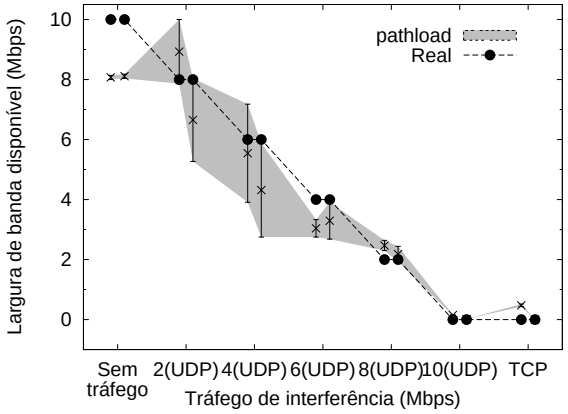
5.3.3 Cenário com enlaces com baixa capacidade nominal e H=3

A fim de se avaliar as ferramentas em redes maiores, foram realizados experimentos utilizando a topologia *H-hop* com $H = 3$ e com $H = 5$. Somente os resultados obtidos com $H = 3$ serão relatados, já que eles foram equivalentes aos resultados com $H = 5$. Assim como foi realizado nos experimentos com $H = 1$, nos experimentos com $H = 3$ as ferramentas foram avaliadas tanto em cenários com enlaces de 10Mbps quanto com enlaces de 1Gbps.

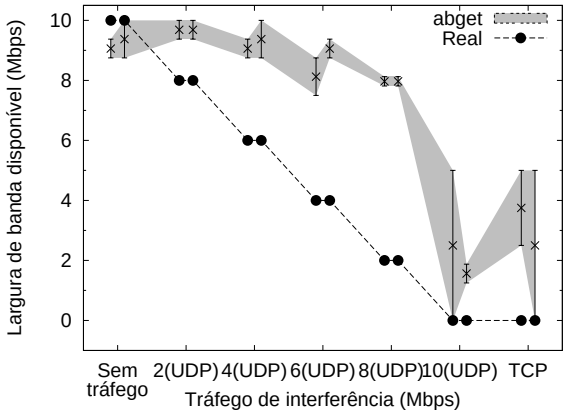
Esta subseção resume os resultados encontrados nos experimentos com $H = 3$, em uma rede emulada pelo `NCTUns` com capacidade de 10Mbps por enlace. As medições realizadas, as identificações dos computadores e as configurações de tráfego de interferência são as mesmas utilizadas nos experimentos apresentados na Subseção 5.3.1. Assim como nas subseções anteriores, somente os resultados mais relevantes são apresentados e discutidos a seguir.

A Figura 5.11(a) e a Figura 5.11(b) apresentam, respectivamente, a precisão do `pathload` e do `abget`. Assim como nos experimentos da Subseção 5.3.1, neste cenário

o `pathload` forneceu os melhores resultados nos casos em que havia tráfego de interferência. De forma semelhante aos resultados anteriores, as estimativas dadas pelo `abget` não seguem o padrão de disponibilidade de largura de banda entre os computadores. A ferramenta continua fornecendo estimativas como se o cenário fosse de situações extremas. Quando há tráfego de interferência UDP menor do que 10Mbps, a estimativa é de quase 100% de disponibilidade, enquanto que quando há tráfego de interferência UDP com taxa de 10Mbps ou tráfego de interferência TCP, a estimativa é de quase 100% de ocupação.



(a) `pathload`



(b) `abget`

Figura 5.11: Estimativas fornecidas (Terceiro cenário)

O tempo de execução das ferramentas é apresentado na Figura 5.12. Comparando os resultados com os da Figura 5.3(a), observa-se que a característica do `abget` de transferir a mesma quantidade de dados no início do processo de estimação, independente da disponibilidade do caminho, continua presente. Quando os enlaces estão mais utilizados (UDP $>$ 6Mbps), o tempo de execução tende a aumentar. De forma semelhante ao observado

nos gráficos da Figura 5.3(a), quando os enlaces estão mais disponíveis (UDP $\leq$ 6Mbps), o `abget` converge para os resultados mais rápido do que o `pathload`.

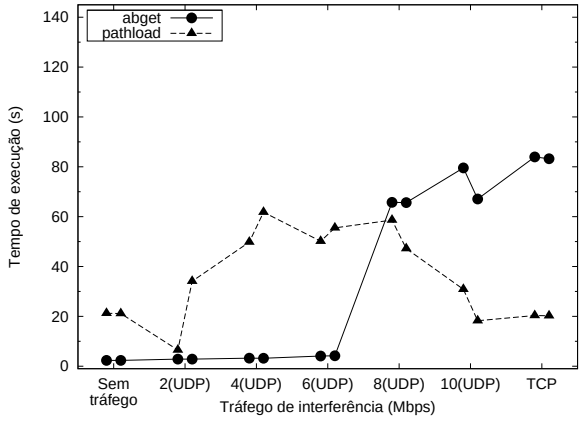


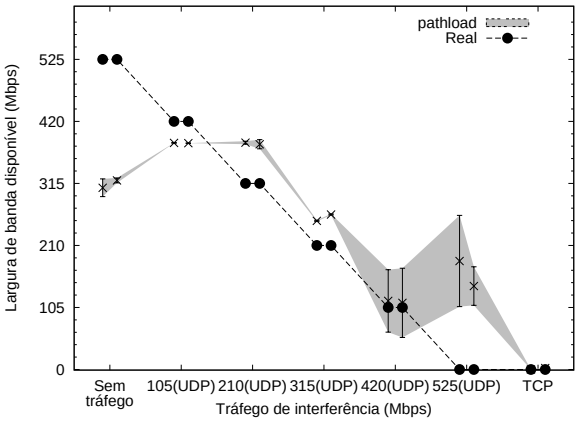
Figura 5.12: Tempo de execução (Terceiro cenário)

Os resultados obtidos apresentam o `pathload` como a ferramenta com melhor precisão em cenários nos quais os enlaces possuem uma baixa capacidade de transmissão (da ordem de 10Mbps), entretanto, a carga extra da execução do `abget` é menor do que a do `pathload`, apesar do `abget` estimar, de forma incorreta, 100% de disponibilidade no caminho para a maioria das configurações de tráfego de interferência.

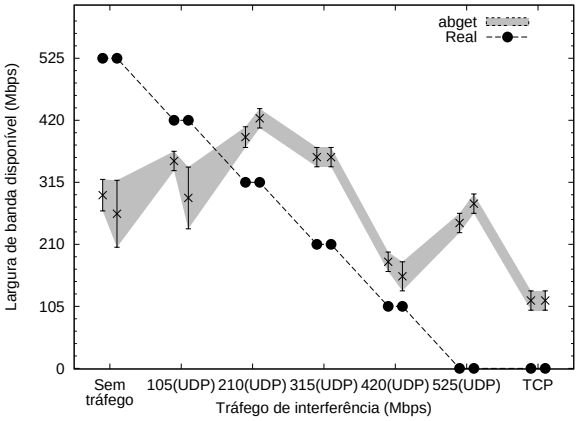
5.3.4 Cenário com enlaces com alta capacidade nominal e H=3

O quarto cenário avalia as ferramentas em uma topologia *H-hop* com $H = 3$. As demais configurações deste cenário são similares àquelas do segundo cenário, ou seja, a capacidade nominal dos enlaces emulados é de 1Gbps e o `iperf` foi utilizado para gerar tráfego de interferência.

Nas figuras 5.13(a) e 5.13(b), fornecem-se os resultados sobre precisão das ferramentas. De forma similar aos resultados dos experimentos com $H = 1$ (Figura 5.6(a) e Figura 5.6(b)), as estimativas do `pathload` foram mais precisas do que as estimativas do `abget`. O `pathload` apresentou, mais uma vez, os melhores resultados quando havia tráfego de interferência na rede, entretanto, nota-se que a adição de enlaces no cenário afetou a precisão do `pathload`, quando comparada com a precisão obtida no segundo cenário. Os resultados do `abget` seguem o padrão de disponibilidade de banda passante, entretanto, observa-se que as estimativas fornecidas indicam que os enlaces estavam mais disponíveis do que a disponibilidade real para a maioria das configurações de tráfego de interferência.



(a) `pathload`



(b) `abget`

Figura 5.13: Estimativas fornecidas (Quarto cenário)

A principal diferença entre os resultados do segundo cenário e os resultados deste quarto cenário diz respeito ao tempo de execução do `abget`. Ao contrário do observado no gráfico da Figura 5.7(a), observa-se, na Figura 5.14, que o `abget` executou mais rápido do que o `pathload` em todos os experimentos, sendo que não houve mudanças significantes no padrão do `pathload`.

Apesar do `abget` ter fornecido estimativas mais precisas do que no cenário onde haviam um menor número de enlaces, o `pathload` apresentou melhores resultados, em termos gerais. Observa-se uma melhoria no comportamento do `abget`, entretanto, essa mudança não é suficiente para justificar a utilização do `abget` em situações nas quais haja acesso a ambos os computadores envolvidos nas estimações e em casos onde os enlaces possuam capacidades da ordem de 1Gbps, dados os melhores resultados obtidos pelo `pathload` nesses casos.

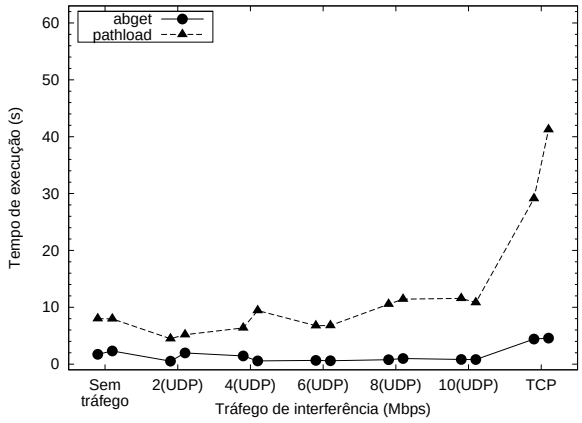


Figura 5.14: Tempo de execução (Quarto cenário)

5.4 Resumo conclusivo

Este capítulo comparou o `pathload` e o `abget`, duas ferramentas de estimativa de largura de banda disponível considerando os requisitos que tais ferramentas devem atender para a sua utilização em grades: precisão nas estimativas com informações referentes à incertezas, baixa intrusão, rapidez nas estimativas e corretude em caso de execuções concorrentes. Pelos experimentos realizados, o `pathload` apresentou melhores resultados do que o `abget`, de uma maneira geral. No entanto, modificações devem ser feitas no `pathload`, para que ele possa ser executado em vários computadores da grade simultaneamente e para evitar que ele gere tráfego que interfira nos demais fluxos da grade durante a estimação de largura de banda disponível. Devido à vantagem de ser executado em apenas um dos computadores durante o processo de estimação, é importante considerar modificações no `abget`, a fim de melhorar a sua precisão e a fim de diminuir a quantidade de parâmetros que devem ser fornecidos pelo usuário.

A Tabela 5.3 resume todos os resultados obtidos com os experimentos detalhados nas seções anteriores. Observa-se que o `pathload` só não apresentou bons resultados, em termos de precisão, quando os enlaces estiveram muito congestionados nos experimentos com enlaces com capacidade de 1Gbps e com execuções simultâneas. À medida que a complexidade dos cenários aumenta, nota-se que o tempo de execução do `abget` é menor do que o do `pathload`. Por último, nota-se que o `pathload` injetou mais bytes na rede do que o `abget`.

Tabela 5.3: Resumo dos resultados

Métrica	`pathload`	`abget`
Estimativas (execução única, 10Mbps)	mais precisas	menos precisas
Estimativas (execuções simultâneas, 10Mbps)	mais precisas	menos precisas
Tempo de execução (execução única, 10Mbps)	Alto	Baixo
Tempo de execução (execuções simultâneas, 10Mbps)	Similar	Similar
Intrusão (execução única, 10Mbps)	Alta	Baixa
Intrusão (execuções simultâneas, 10Mbps)	Alta	Baixa
Estimativas (execução única, 1Gbps)	mais precisas	menos precisas
Estimativas (execuções simultâneas, 1Gbps)	mais precisas sob alta disponibilidade	menos precisas
Tempo de execução (execução única, 1Gbps)	Baixo	Alto
Tempo de execução (execuções simultâneas, 1Gbps)	Baixo	Alto
Intrusão (execução única, 1Gbps)	Alta	Baixa
Intrusão (execuções simultâneas, 1Gbps)	Alta	Baixa

Capítulo 6

Conclusões e trabalhos futuros

A dinâmica dos recursos, as incertezas nas demandas das aplicações e as imprecisões das ferramentas utilizadas para estimar a disponibilidade dos recursos em grades dificultam o oferecimento de informações precisas para os escalonadores de tarefas. Esta Tese apresentou mecanismos que garantem um bom funcionamento das grades independente das incertezas inerentes ao ambiente.

O Capítulo 2 utilizou um exemplo numérico para demonstrar o impacto negativo de se ignorar as incertezas das grades. Observou-se aumentos de até 75% nos *makespans* de aplicações que foram escalonadas por escalonadores que receberam informações incorretas a cerca da disponibilidade da grade. O Capítulo 2 também resumiu propostas existentes para lidar com as incertezas. A maioria dessas propostas lida com as incertezas através de mecanismos que realizam um ciclo contínuo de monitoramento, reescalonamento e migração de tarefas, ou seja, eles agem de forma reativa na tentativa de diminuir o impacto negativo das incertezas. A carga extra e a intrusão geradas na grade por esses ambientes justificam a implementação de mecanismos que lidem com as incertezas diretamente nos escalonadores de tarefas.

A motivação para se implementar escalonadores robustos para grades levou à proposta de dois escalonadores de tarefas nesta Tese. Foram também investigados dois estimadores de largura de banda disponível, a fim de se avaliar a possibilidade de serem utilizados para fornecer estimativas de disponibilidade de largura de banda para um dos escalonadores propostos.

O Capítulo 3 apresentou o escalonador IPDT-FUZZY, um escalonador robusto às incertezas nas descrições das aplicações. O escalonador utiliza técnicas de otimização fuzzy e resolve um problema de programação inteira 0–1 que propõe escalonamentos de aplicações formadas por tarefas dependentes, a fim de se minimizar o *makespan*. Experimentos de simulação realizados com três DAGs executados em várias grades comprovaram a eficácia do escalonador em situações onde há uma expectativa alta de incerteza na des-

crição dos requisitos de comunicação. O escalonador mostrou-se robusto em termos de *speedup* produzido, tempo de execução para fornecer o escalonamento e utilização dos recursos de comunicação. Os resultados alcançados pelo escalonador IPDT-FUZZY foram comparados com os resultados do escalonador IPDT, um escalonador que não implementa técnicas de otimização fuzzy e que serviu de base para sua construção. Pelas comparações, o escalonador IPDT-FUZZY produziu escalonamentos com *speedups* até 29,55% maiores, enquanto o tempo de execução foi até 95,67% menor. Mesmo quando comparado com o escalonador IPDT, quando este tinha acesso a informações corretas sobre as demandas de comunicação das aplicações, o escalonador IPDT-FUZZY mostrou-se vantajoso. Os *speedups* dos escalonamentos propostos por ele foram tão bons quanto aqueles propostos pelo escalonador IPDT, com a vantagem do tempo de execução ter sido até 97,77% inferior.

Os resultados alcançados com a implementação do escalonador IPDT-FUZZY levaram à proposta do escalonador IP-FULL-FUZZY, um escalonador de tarefas que lida tanto com as incertezas nas demandas das aplicações quanto com as incertezas na disponibilidade dos recursos. O escalonador IP-FULL-FUZZY foi apresentado no Capítulo 4. Assim como o escalonador IPDT-FUZZY, o escalonador IP-FULL-FUZZY resolve um problema de programação inteira 0–1. Experimentos de simulação realizados com vários DAGs e várias grades comprovaram a eficácia do escalonador em situações onde há uma expectativa alta de incerteza na largura de banda disponível entre os computadores. O escalonador IP-FULL-FUZZY mostrou-se robusto em termos de *speedup* produzido e apresentou, em média, um tempo de processamento equivalente àquele do escalonador IPDT-FUZZY. Em situações onde foram simuladas incertezas na disponibilidade dos recursos, o *speedup* do escalonador IP-FULL-FUZZY chegou a ser, em média, até 41% maior do que o do escalonador IPDT-FUZZY e até 34% maior do que o do escalonador RANDOM, um escalonador que aloca os recursos de forma pseudo-aleatória e ignora todas as fontes de incerteza. Além disso, observou-se que o tempo de execução do escalonador IP-FULL-FUZZY pode ser, em média, até 13,25% menor do que aquele do escalonador RANDOM. Conclui-se, entretanto, que é importante definir limites de tempo para a execução do escalonador IP-FULL-FUZZY para evitar que o tempo necessário para derivar o escalonamento invalide a solução. Os escalonamentos produzidos pelo escalonador IP-FULL-FUZZY foram, também, comparados com aqueles devolvidos pelos escalonadores clássicos HEFT e CPOP e observou-se, respectivamente, ganhos de até 25% e 17% no *speedup*. Os resultados encontrados comprovam a utilidade de se lidar com as incertezas diretamente nos escalonadores de tarefas em grades.

Para que o escalonador IP-FULL-FUZZY possa ser utilizado na prática é necessário que as ferramentas de monitoramento forneçam estimativas sobre a disponibilidade dos recursos da grade como intervalos e, além disso, apresentem bom desempenho. O Capítulo 5 avaliou as ferramentas `pathload` e `abget`, que podem ser utilizadas com este fim. As fer-

ramentas foram avaliadas no contexto dos requisitos que estimadores de largura de banda disponível devam atender para a sua utilização em grades: precisão nas estimativas com informações referentes à incertezas, baixa intrusão, rapidez nas estimativas e corretude em caso de execuções concorrentes. Pelos experimentos realizados, observou-se melhores resultados do `pathload` do que do `abget`, quando todos os requisitos foram avaliados em conjunto. No entanto, modificações devem ser feitas no `pathload` para que ele possa ser executado em vários computadores da grade simultaneamente e para evitar que ele gere tráfego que interfira nos demais fluxos da grade durante suas estimativas. Devido à sua vantagem de ser executado em apenas 1 dos computadores durante as estimativas, é importante considerar modificações no `abget`, a fim de melhorar a sua precisão e a fim de diminuir a quantidade de parâmetros que devem ser passados pelo usuário.

A lista a seguir apresenta alguns pontos relacionados com o conteúdo desta Tese que merecem investigação futura:

- Comparação do escalonador IP-FULL-FUZZY com mecanismos baseados em monitoramento, reescalonamento e migração de tarefas: os cenários adequados para cada uma das ferramentas e valores numéricos que comprovem o quanto uma é melhor do que a outra devem ser investigados;

- Avaliação dos escalonadores IPDT-FUZZY e IP-FULL-FUZZY em cenários onde há incertezas tanto nas estimativas de computação quanto nas estimativas de comunicação: o foco desta Tese foi na avaliação dos escalonadores em cenários onde haviam incertezas na quantidade de dados transferidos entre as tarefas e na largura de banda disponível. Entretanto, também é importante avaliar os escalonadores quando há incertezas relacionadas com o tempo de processamento das tarefas;

- Melhorias nos estimadores de largura de banda disponível: a fim de permitir estimativas simultâneas para um mesmo recurso da grade o `pathload` deve ser modificado para aceitar conexões concorrentes e para utilizar portas dinâmicas para as conexões UDP. Ele também precisa ser modificado através da adição de um contador de bytes que interrompa a sua execução caso ele alcance um limite informado pelo usuário. Melhorias devem ser feitas no `abget` para aumentar as precisões das suas estimativas;

- Avaliação do desempenho dos mecanismos através de medições em ambientes heterogêneos como o PlanetLab [94]: ambientes como o PlanetLab permitem que cientistas testem soluções voltadas para ambientes distribuídos e heterogêneos antes deles serem utilizados em ambientes reais. O escalonador IP-FULL-FUZZY poderia ser utilizado para propor escalonamentos de tarefas em recursos do PlanetLab levando em consideração estimativas de largura de banda disponível feitas pelo `pathload`.

Referências Bibliográficas

[1] Rashid Al-Ali, Abdelhakim Hafid, Omer F. Rana, and David W. Walker. QoS Adaptation in Service-Oriented Grids. In *Proceedings of the 1st International Workshop on Middleware for Grid Computing (MGC2003)*, Junho 2003. `http://mgc2003.lncc.br/cam_ready/MGC289_final.pdf`. Último acesso em 4 de Janeiro de 2010.

[2] Réka Albert and Albert-László Barabási. Topology of Evolving Networks: Local Events and Universality. *Physical Review Letters*, 85(24):5234–5237, Dezembro 2000.

[3] Demetres Antoniades, Manos Athanatos, Antonis Papadogiannakis, Evangelos P. Markatos, and Constantine Dovrolis. Available Bandwidth Measurements as Simple as Running wget. In *Proceedings of Passive and Active Measurement Conference (PAM 2006)*, páginas 61–70, Março 2006.

[4] Daniel M. Batista, Luciano J. Chaves, Nelson L. S. da Fonseca, and Artur Ziviani. Análise de Desempenho de Estimadores de Largura de Banda Disponível para Utilização em Grades. In *Anais do XXIX Congresso da Sociedade Brasileira de Computação – VIII WPerformance*, páginas 2225–2240. SBC, Julho 2009.

[5] Daniel M. Batista, Luciano J. Chaves, Nelson L. S. da Fonseca, and Artur Ziviani. Performance Analysis of Available Bandwidth Estimation Tools for Grid Networks. In *Proceedings of the 14th IEEE International Workshop on Computer-Aided Modeling and Design of Communication Links and Networks*, páginas 1–5. IEEE, Junho 2009.

[6] Daniel M. Batista, Luciano J. Chaves, Nelson L. S. da Fonseca, and Artur Ziviani. Performance Analysis of Available Bandwidth Estimation Tools for Grid Networks. *The Journal of Supercomputing*, páginas 1–19, Outubro 2009. `http://dx.doi.org/10.1007/s11227-009-0344-z`. Último acesso em 4 de Janeiro de 2010.

[7] Daniel M. Batista and Nelson L. S. da Fonseca. A Brief Survey on Resource Allocation in Service Oriented Grids. In *Proceedings of the IEEE Globecom Workshops 2007 –*

1st IEEE Workshop on Enabling the Future Service-Oriented Internet, páginas 1–5. IEEE, Novembro 2007.

[8] Daniel M. Batista and Nelson L. S. da Fonseca. Empowering Grids with Flexibility to Cope with Uncertainties. In *ICC Workshops'08: Proceedings of the IEEE International Conference on Communications Workshops 2008 – 13th IEEE International Workshop on Computer-Aided Modeling, Analysis and Design of Communication Links and Networks*, páginas 227–231. IEEE, Maio 2008.

[9] Daniel M. Batista and Nelson L. S. da Fonseca. Um Framework para Tratamento de Incertezas e Flutuações em Grades 2008. Relatório Técnico IC-08-025, Instituto de Computação – Unicamp, Setembro 2008. `http://www.ic.unicamp.br/~reltech/2008/08-25.pdf`. Último acesso em 4 de Janeiro de 2010.

[10] Daniel M. Batista and Nelson L. S. da Fonseca. A Survey of Self-Adaptive Grids. *IEEE Communications Magazine*, 2010. Aceito para publicação.

[11] Daniel M. Batista and Nelson L. S. da Fonseca. Self-Adjustment for Service Provisioning in Grids. In Dr. Nick Antonopoulos; Mr. Georgios Exarchakos; Dr. Maozhen Li; Dr. Antonio Liotta., editor, *Handbook of Research on P2P and Grid Systems for Service-Oriented Computing: Models, Methodologies and Applications*. IGI Global, Hershey, EUA, 2010.

[12] Daniel M. Batista, Nelson L. S. da Fonseca, Flavio K. Miyazawa, and Fabrizio Granelli. Self-Adjustment of Resource Allocation for Grid Applications. *Computer Networks*, 52(9):1762–1781, Junho 2008.

[13] Daniel M. Batista, André C. Drummond, and Nelson L. S. da Fonseca. Scheduling Grid Tasks under Uncertain Demands. In *SAC '08: Proceedings of the 2008 ACM Symposium on Applied Computing*, páginas 2041–2045. ACM, Março 2008.

[14] Daniel M. Batista, André C. Drummond, and Nelson L. S. da Fonseca. Um Escalonador de Tarefas Dependentes Robusto às Incertezas nas Descrições das Aplicações em Grades. In *Anais do XXVIII Congresso da Sociedade Brasileira de Computação – VII WPerformance*, páginas 161–179. SBC, Julho 2008.

[15] Daniel M. Batista, André C. Drummond, and Nelson L. S. da Fonseca. Robust Scheduler for Grid Networks. In *SAC '09: Proceedings of the 2009 ACM Symposium on Applied Computing*, páginas 35–39. ACM, Março 2009.

[16] Philip A. Bernstein. Middleware: A Model for Distributed System Services. *Communications of the ACM*, 39(2):86–98, Fevereiro 1996.

[17] Peter Blaha, Karlheinz Schwarz, Georg Madsen, Dieter Kvasnicka, and Joachim Luitz. WIEN 2k, Setembro 2009. `http://www.wien2k.at/`. Último acesso em 4 de Janeiro de 2010.

[18] Jim Blythe, Sonal Jain, Ewa Deelman, Yolanda Gil, Karan Vahi, Anirban Mandal, and Ken Kennedy. Task Scheduling Strategies for Workflow-based Applications in Grids. In *Proceedings of the Fifth IEEE International Symposium on Cluster Computing and the Grid (CCGRID'05)*, páginas 759–767. IEEE, Maio 2005.

[19] Rajkumar Buyya, David Abramson, and Jonathan Giddy. A Case for Economy Grid Architecture for Service Oriented Grid Computing. In *Proceedings of the 15th International Parallel and Distributed Processing Symposium*, páginas 776–790. IEEE, Abril 2001.

[20] Ladislau Bölöni and Dan C. Marinescu. Robust Scheduling of Metaprograms. *Journal of Scheduling*, 5:395–412, Setembro 2002.

[21] Henri Casanova. Distributed Computing Research Issues in Grid Computing. *SIGACT News*, 33(3):50–70, Setembro 2002.

[22] Henri Casanova, Dmitrii Zagorodnov, Francine Berman, and Arnaud Legrand. Heuristics for Scheduling Parameter Sweep Applications in Grid Environments. In *HCW '00: Proceedings of the 9th Heterogeneous Computing Workshop*, páginas 349–363. IEEE, Maio 2000.

[23] Su-Hui Chiang, Andrea Arpaci-Dusseau, and Mary K. Vernon. The Impact of More Accurate Requested Runtimes on Production Job Scheduling Performance. In *8th International Workshop on Job Scheduling Strategies for Parallel Processing (JSSPP 2002)*, páginas 103–127. Springer Berlin / Heidelberg, Julho 2002.

[24] Su-Hui Chiang, Rajesh K. Mansharamani, and Mary K. Vernon. Use of Application Characteristics and Limited Preemption for Run-to-Completion Parallel Processor Scheduling Policies. In *SIGMETRICS '94: Proceedings of the 1994 ACM SIGMETRICS Conference on Measurement and Modeling of Computer Systems*, páginas 33–44. ACM, Maio 1994.

[25] Greg Chun, Holly Dail, Henri Casanova, and Allan Snavely. Benchmark Probes for Grid Assessment. In *Proceedings of the 18th International Parallel and Distributed Processing Symposium*, páginas 276–283. IEEE, Abril 2004.

[26] Walfredo Cirne and Francine Berman. A Comprehensive Model of the Supercomputer Workload. In *IEEE International Workshop on Workload Characterization (WWC-4)*, páginas 140–148. IEEE, Dezembro 2001.

[27] Fabricio A. Barbosa da Silva and Isaac D. Scherson. Improving Parallel Job Scheduling Using Runtime Measurements. In *IPDPS '00/JSSPP '00: Proceedings of the Workshop on Job Scheduling Strategies for Parallel Processing*, páginas 18–38. Springer Berlin / Heidelberg, Maio 2000.

[28] Matthew Doar and Ian M. Leslie. How Bad is Naive Multicast Routing? In *IEEE INFOCOM '93*, páginas 82–89. IEEE, Março 1993.

[29] Michalis Faloutsos, Petros Faloutsos, and Christos Faloutsos. On Power-Law Relationships of the Internet Topology. In *SIGCOMM '99: Proceedings of the Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*, páginas 251–262. ACM, Setembro 1999.

[30] Carole Fayad, Jonathan M. Garibaldi, and Djamila Ouelhadj. Fuzzy Grid Scheduling Using Tabu Search. In *Proceedings of IEEE International Fuzzy Systems Conference*, páginas 1–6. IEEE, Julho 2007.

[31] Tiziana Ferrari and Francesco Giacomini. Network Monitoring for GRID Performance Optimization. *Computer Communications*, 27(14):1357–1363, Setembro 2004. Special Issue on Network Support for Grid Computing.

[32] Ian Foster. What is the Grid? A Three Point Checklist. *GRIDToday*, 1(6), Julho 2002. `http://www-fp.mcs.anl.gov/~foster/Articles/WhatIsTheGrid.pdf`. Último acesso em 4 de Janeiro de 2010.

[33] Ian Foster. Globus Toolkit Version 4: Software for Service-Oriented Systems. *Journal of Computer Science and Technology*, 21(4):513–520, Julho 2006.

[34] Ian Foster and Carl Kesselman, editors. *The Grid: Blueprint for a New Computing Infrastructure*. Morgan Kaufmann, Agosto 1998.

[35] Inés González-Rodrígues, Camino R. Vela, and Jorge Puente. A Memetic Approach to Fuzzy Job Shop Based on Expectation Model. In *IEEE International Fuzzy Systems Conference (FUZZY-IEEE)*, páginas 7–12. IEEE, Julho 2007.

[36] Ibrahim W. Habib, Qiang Song, Zhaoming Li, and Nageswara S. V. Rao. Deployment of the GMPLS Control Plane for Grid Applications in Experimental High-Performance Networks. *IEEE Communications Magazine*, 44(3):65–73, Março 2006.

[37] Hamed Haddadi, Miguel Rio, Gianluca Iannaccone, Andrew Moore, and Richard Mortier. Network Topologies: Inference, Modeling, and Generation. *IEEE Communications Surveys & Tutorials*, 10:48–69, Abril 2008.

[38] Qi He, Constantinos Dovrolis, and Mostafa Ammar. On the Predictability of Large Transfer TCP Throughput. *Computer Networks*, 51(14):3959–3977, Outubro 2007.

[39] Eduardo Huedo, Ruben S. Montero, and Ignacio M. Llrorent. An Experimental Framework for Executing Applications in Dynamic Grid Environments. Relatório Técnico ICASE-2002-43, NASA Langley Research Center, Novembro 2002. `http://hdl.handle.net/2060/20030002654`. Último acesso em 4 de Janeiro de 2010.

[40] Eduardo Huedo, Rubén S. Montero, and Ignacio M. Llorente. Experiences on Adaptive Grid Scheduling of Parameter Sweep Applications. In *Proceedings of the 12th Euromicro Conference on Parallel, Distributed and Network-Based Processing*, páginas 28–33. IEEE, Fevereiro 2004.

[41] Ian Foster. There's Grid in them thar Clouds, Janeiro 2008. `http://ianfoster.typepad.com/blog/2008/01/theres-grid-in.html`. Último acesso em 4 de Janeiro de 2010.

[42] Takeshi Ito, Hiroyuki Ohsaki, and Makoto Imase. On Parameter Tuning of Data Transfer Protocol GridFTP for Wide-Area Grid Computing. In *Proceedings of the BroadNets 2005, Vol. 2*, páginas 1338–1344. IEEE, Outubro 2005.

[43] Manish Jain and Constantinos Dovrolis. End-to-end Available Bandwidth: Measurement Methodology, Dynamics, and Relation with TCP Throughput. *SIGCOMM Computer Communication Review*, 32(4):295–308, Outubro 2002.

[44] Manish Jain and Constantinos Dovrolis. Ten Fallacies and Pitfalls on End-to-end Available Bandwidth Estimation. In *IMC '04: Proceedings of the 4th ACM SIGCOMM Conference on Internet Measurement*, páginas 272–277. ACM, Outubro 2004.

[45] Manish Jain and Constantinos Dovrolis. End-to-end Estimation of the Available Bandwidth Variation Range. In *SIGMETRICS '05: Proceedings of the 2005 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems*, páginas 265–276. ACM, Junho 2005.

[46] Stamatios V. Kartalppoulos. *Understanding Neural Networks and Fuzzy Logic*. IEEE, 1996.

[47] Yehia El khatib and Christopher Edwards. A Survey-Based Study of Grid Traffic. In *GridNets '07: Proceedings of the First International Conference on Networks for Grid Applications*, páginas 1–8. ICST, Outubro 2007.

[48] Junghwan Kim, Jungkyu Rho, Jeong-Ook Lee, and Myeong-Cheol Ko. CPOC: Effective Static Task Scheduling for Grid Computing. In *Proceedings of the First International Conference on High Performance Computing and Communications (HPCC 2005)*, páginas 477–486. Springer Berlin / Heidelberg, Setembro 2005.

[49] Leonard Kleinrock. *Queueing Systems, Volume I: Theory.* John Wiley & Son, Janeiro 1975.

[50] Yu-Kwong Kwok and Ishfaq Ahmad. Static scheduling algorithms for allocating directed task graphs to multiprocessors. *ACM Computing Surveys*, 31(4):406–471, Dezembro 1999.

[51] Cynthia Bailey Lee, Yael Schwartzman, Jennifer Hardy, and Allan Snavely. Are User Runtime Estimates Inherently Inaccurate? In *10th International Workshop on Job Scheduling Strategies for Parallel Processing (JSSPP 2004)* , páginas 253–263. Springer Berlin / Heidelberg, Junho 2004.

[52] Bruce Lowekamp, Brian Tierney, Les Cottrell, Richard Hughes-Jones, Thilo Kielmann, and Martin Swany. A Hierarchy of Network Performance Characteristics for Grid Applications and Services, Maio 2004. Draft GFD-R-P.023 `http://www.ggf.org/documents/GFD.23.pdf`. Último acesso em 4 de Janeiro de 2010.

[53] Matthew L. Massie, Brent N.Chun, and David E. Culler. The Ganglia Distributed Monitoring System: Design, Implementation, and Experience. *Parallel Computing*, 30(7):817–840, Julho 2004.

[54] Alberto Medina, Ibrahim Matta, and John Byers. On the Origin of Power Laws in Internet Topologies. *SIGCOMM Computer Communication Review*, 30(2):18–28, Abril 2000.

[55] Hans Mittelmann. Decision Tree for Optimization Software, Agosto 2009. `http://plato.asu.edu/bench.html`. Último acesso em 4 de Janeiro de 2010.

[56] Lilian Noronha Nassif, José Marcos Nogueira, Mohamed Ahmed, Ahmed Karmouch, Roger Impey, and Flávio Vinícius de Andrade. Job Completion Prediction in Grid Using Distributed Case-based Reasoning. In *14th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprise (WETICE'05)*, páginas 249–254. IEEE, Junho 2005.

[57] Harvey B. Newman. Networking for High Energy and Nuclear Physics as Global E-Science. In *Proceedings of Computing in High Energy and Nuclear Physics*, Setembro 2004. `http://ultralight.caltech.edu/web-site/common/publications/`

`article/2004/09chep04/Network_for_HEP.pdf`. Último acesso em 4 de Janeiro de 2010.

[58] Michael Oikonomakos, Kostas Christodoulopoulos, and Emmanouel (Manos) Varvarigos. Profiling Computation Jobs in Grid Systems. In *Proceedings of the Seventh IEEE International Symposium on Cluster Computing and the Grid (CCGRID 2007)*, páginas 197–204. IEEE, Maio 2007.

[59] Yusuf Ozturk and Manish Kulkarni. DIChirp: Direct Injection Bandwidth Estimation. *International Journal of Network Management*, 18(5):377–394, Setembro 2008.

[60] Christos H. Papadimitriou and Kennet Steiglitz. *Combinatorial Optimization – Algorithms and Complexity*, páginas 363–366. Dover Publications, 1998.

[61] Ravi Prasad, Constantinos Dovrolis, Margaret Murray, and kc claffy. Bandwidth Estimation: Metrics, Measurement Techniques, and Tools. *IEEE Network*, 17(6):27–35, Novembro 2003.

[62] Ravi Prasad, Manish Jain, and Constantinos Dovrolis. Effects of Interrupt Coalescence on Network Measurements. In *Proceedings of the Passive and Active Network Measurement Workshop (PAM 2004)*, páginas 247–256. Springer Berlin / Heidelberg, Abril 2004.

[63] Radu Prodan and Thomas Fahringer. Dynamic Scheduling of Scientific Workflow Application on the Grid: a Case Study. In *SAC'05: Proceedings of the 2005 ACM Symposium on Applied Computing*, páginas 687–694. ACM, Março 2005.

[64] Manish Jain Ravi Prasad and Constantinos Dovrolis. Evaluating Pathrate and Pathload with Realistic Cross-Traffic, Dezembro 2003. Palestra no Bandwidth Estimation Workshop 2003. `http://www.caida.org/workshops/isma/0312/slides/rprasad-best.pdf`. Último acesso em 4 de Janeiro de 2010.

[65] Rodrigo Real, Adenauer Yamin, Luciano da Silva, Gustavo Frainer, Iara Augustin, Jorge Barbosa, and Cláudio Geyer. Resource Scheduling on Grid: Handling Uncertainty. In *Proceedings of the Fourth International Workshop on Grid Computing*, páginas 205–207. IEEE, Novembro 2003.

[66] Rizos Sakellariou and Henan Zhao. A Low-cost Rescheduling Policy for Efficient Mapping of Workflows on Grid Systems. *Scientific Programming*, 12:253 – 262, Dezembro 2004.

[67] Jennifer M. Schopf. Ten Actions when Grid Scheduling. In Jarek Nabrzyski and Jennifer M. Schopf and Jan Weglarz, editor, *Grid Resource Management: State of the Art and Future Trends*, páginas 15–23. Springer, 2003.

[68] Carlos R. Senna, Luiz F. Bittencourt, and Edmundo Madeira. Execution of Service Workflows in Grid Environments. In *5th International Conference on Testbeds and Research Infrastructures for the Development of Networks and Communities (TridentCom 2009)*, páginas 1–10. IEEE, Abril 2009.

[69] Alok Shriram, Margaret Murray, Young Hyun, Nevil Brownlee, Andre Broido, Marina Fomenkov, and kc claffy. Comparison of Public End-to-End Bandwidth Estimation Tools on High-Speed Links. In *Proceedings of Passive and Active Network Measurement Conference (PAM 2005)*, páginas 306–320. Springer Berlin / Heidelberg, Março 2005.

[70] John A. Silvester. CalREN: Advanced Network(s) for Education in California, Oct 2005. `http://isd.usc.edu/~jsilvest/talks-dir/20051021-cudi-merida.pdf`. Último acesso em 4 de Janeiro de 2010.

[71] Xian-He Sun and Ming Wu. GHS: A Performance System of Grid Computing. In *19th IEEE International Parallel and Distributed Processing Symposium*, page 228a. IEEE, Abril 2005.

[72] Ananth I. Sundararaj, Ashish Gupta, and Peter A. Dinda. Dynamic Topology Adaptation of Virtual Networks of Virtual Machines. In *LCR '04: Proceedings of the 7th Workshop on Languages, Compilers, and Runtime Support for Scalable Systems*, páginas 1–8. ACM, Outubro 2004.

[73] Ajay Tirumala, Les Cottrell, and Tom Dunigan. Measuring End-to-end Bandwidth with Iperf Using Web100. Relatório Técnico SLAC-PUB-9733, Stanford Linear Accelerator Center, Abril 2003. `http://www.slac.stanford.edu/pubs/slacpubs/9000/slac-pub-9733.html`. Último acesso em 4 de Janeiro de 2010.

[74] Haluk Topcuouglu, Salim Hariri, and Min you Wu. Performance-Effective and Low-Complexity Task Scheduling for Heterogeneous Computing. *IEEE Transactions on Parallel and Distributed Systems*, 13(3):260–274, Março 2002.

[75] Sathish S. Vadhiyar and Jack J. Dongarra. A Performance Oriented Migration Framework for the Grid. In *Proceedings of the 3rd IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGRID'03)*, páginas 130–137. IEEE, Maio 2003.

[76] S. Y. Wang, C. L. Choua, and C. C. Lin. The Design and Implementation of the NCTUns Network Simulation Engine. *Simulation Modelling Practice and Theory*, 15(1):57–81, Janeiro 2007.

[77] Marek Wieczorek, Radu Prodan, and Thomas Fahringer. Scheduling of Scientific Workflows in the ASKALON Grid Environment. *SIGMOD Record*, 34(3):56–62, Setembro 2005.

[78] Rich Wolski, Neil T. Spring, and Jim Hayes. The Network Weather Service: a Distributed Resource Performance Forecasting Service for Metacomputing. *Future Generation Computer Systems*, 15(5–6):757–768, Outubro 1999.

[79] Zhifeng Yu and Weisong Shi. An Adaptive Rescheduling Strategy for Grid Workflow Applications. In *IPDPS 2007: Proceedings of the IEEE International Parallel and Distributed Processing Symposium*, páginas 1–8. IEEE, Março 2007.

[80] Serafeim Zanikolas and Rizos Sakellariou. A Taxonomy of Grid Monitoring Systems. *Future Generation Computer Systems*, 21(1):163–188, Janeiro 2005.

[81] Henan Zhao and Rizos Sakellariou. Scheduling Multiple DAGs onto Heterogeneous Systems. In *Proceedings of the 20h International Parallel and Distributed Processing Symposium (IPDPS 2006)*, páginas 130–143. IEEE, Abril 2006.

[82] Hans-Jürgen Zimmermann. *Fuzzy Set Theory and its Applications*. Kluwer Academic Publishers, Abril 1996.

[83] Applications of Montage, 2006. `http://montage.ipac.caltech.edu/applications.html`. Último acesso em 4 de Janeiro de 2010.

[84] GriPhyN - Grid Physics Network, 2006. `http://www.griphyn.org/`. Último acesso em 8 de Maio de 2008.

[85] Computational Chemistry Grid, 2007. `https://www.gridchem.org/`. Último acesso em 4 de Janeiro de 2010.

[86] GNU Binutils, Agosto 2007. `http://www.gnu.org/software/binutils/`. Último acesso em 4 de Janeiro de 2010.

[87] Iperf, Abril 2008. `http://sourceforge.net/projects/iperf/?abmode=1`. Último acesso em 4 de Janeiro de 2010.

[88] 5th IEEE International Conference on e-Science, 2009. `http://www.oerc.ox.ac.uk/ieee/`. Último acesso em 4 de Janeiro de 2010.

[89] CAIDA : tools : taxonomy, Janeiro 2009. http://www.caida.org/tools/taxonomy/performance.xml#bw. Último acesso em 4 de Janeiro de 2010.

[90] EGEE Project: Objectives, Setembro 2009. http://project.eu-egee.org/. Último acesso em 4 de Janeiro de 2010.

[91] FICO Xpress-Optimizer, 2009. http://www.fico.com/en/Products/DMTools/xpress-overview/Pages/Xpress-Optimizer.aspx. Último acesso em 4 de janeiro de 2010.

[92] IPDPS – IEEE International Parallel & Distributed Processing Symposium, Julho 2009. http://www.ipdps.org/ipdps2009/2009_advance_program.html. Último acesso em 4 de Janeiro de 2010.

[93] Open Grid Forum, 2009. http://www.gridforum.org/. Último acesso em 4 de Janeiro de 2010.

[94] PlanetLab — An Open Platform for Developing, Deploying, and Accessing Planetary-Scale Services, Maio 2009. http://www.planet-lab.org/. Último acesso em 4 de Janeiro de 2010.

[95] TOP500 List - November 2009 (1-100), Novembro 2009. http://www.top500.org/list/2009/11/100. Último acesso em 4 de janeiro de 2010.

[96] Worldwide LHC Computing Grid, Dezembro 2009. http://lcg.web.cern.ch/LCG/. Último acesso em 4 de Janeiro de 2010.

[97] About SETI@home, 2010. http://setiathome.berkeley.edu/sah_about.php. Último acesso em 4 de Janeiro de 2010.

[98] CCGrid: IEEE International Symposium on Cluster, Cloud, and Grid Computing, Setembro 2010. http://grid.sjtu.edu.cn/ccgrid2009/. Último acesso em 4 de janeiro de 2010.

[99] Folding@home, 2010. http://folding.stanford.edu/. Último acesso em 4 de Janeiro de 2010.