Escalonamento em grades móveis: uma abordagem ciente do consumo de energia

Luiz César Borro

Escalonamento em grades móveis: uma abordagem ciente do consumo de energia

Luiz César Borro

Orientadora: **Profa. Dra. Sarita Mazzini Bruschi**

Dissertação apresentada ao Instituto de Ciências Matemáticas e de Computação - ICMC-USP, como parte dos requisitos para obtenção do título de Mestre em Ciências - Ciências de Computação e Matemática Computacional. *VERSÃO REVISADA*

USP – São Carlos
Março de 2014

Ficha catalográfica elaborada pela Biblioteca Prof. Achille Bassi
e Seção Técnica de Informática, ICMC/USP,
com os dados fornecidos pelo(a) autor(a)

Borro, Luiz César
B737e Escalonamento em grades móveis: uma abordagem ciente do consumo de energia / Luiz César Borro; orientadora Sarita Mazzini Bruschi. -- São Carlos, 2014.
53 p.

Dissertação (Mestrado - Programa de Pós-Graduação em Ciências de Computação e Matemática Computacional) -- Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, 2014.

1. Grades Móveis. 2. Escalonamento. 3. Avaliação de Desempenho. I. Bruschi, Sarita Mazzini, orient. II. Título.

"Isn't it enough to see that a garden is beautiful without having to believe that there are fairies at the bottom of it too?".
Douglas Adams

Agradecimentos

Agradeço, em primeiro lugar, à minha família, especialmente meus pais, Oscar e Alda, e irmãos, André e Rafael, pelo amor e carinho, bem como pela confiança em mim depositada. Sem o apoio e compreensão de vocês, este trabalho não seria possível.

De maneira especial, agradeço à minha orientadora Profa. Dra. Sarita Mazzini Bruschi pela paciência e compreensão, além de toda atenção e empenho dedicados a mim nesses últimos dois anos.

Ao pessoal de Sanca, Alan Valejo, Geraldo Pereira e Zhang Yifei, pelas agradabilíssimas horas gastas em discussões sobre os mais infinitos assuntos.

Ao CNPq e à FAPESP pelo apoio financeiro.

Resumo

Considerando-se o contexto de gerenciamento energético em grades móveis, neste trabalho foram propostos dois algoritmos de escalonamento (*Maximum Regret* e *Greedy*) que, além de minimizar o consumo de energia, visam assegurar o cumprimento dos requisitos de qualidade de serviço das aplicações submetidas pelos usuários. Tais algoritmos foram projetados a partir de soluções heurísticas para o problema de escalonamento ciente de consumo de energia em grades móveis, que foi modelado como um problema de otimização envolvendo variáveis binárias. Por meio de experimentos, que consideraram tanto cenários estáticos quanto dinâmicos, foi demonstrada a viabilidade dos algoritmos de escalonamento propostos em relação à redução do consumo de energia. Em seu pior caso, o algoritmo *Maximum Regret* foi 12,18% pior que o referencial determinado pela melhor solução do *solver* Gurobi; já no pior caso do algoritmo *Greedy*, tal diferença foi de apenas 8,14%.

Abstract

Considering the context of energy management in mobile grids, this work proposes two scheduling algorithms (Maximum Regret and Greedy) that aim not only to reduce the energy consumption of the mobile devices, but also to ensure the QoS (Quality of Service) requirements of the running applications. These algorithms were designed based on heuristics for the energy aware scheduling problem in mobile grids, which was modeled as an optimization problem with integer variables. The performances of the proposed scheduling algorithms were evaluated by an extensive set of experiments, which demonstrated the feasibility of the adopted approach regarding energy consumption minimization. In its worst case, the Maximum Regret algorithm was 12.18% worse than the best solution provided by the Gurobi solver. While in the Greedy's worst case the performance difference was just 8.14%.

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Siglas

API	*Application Programming Interface*
BSS	*Basic Service Set*
DFPLTF	*Dynamic Fastest Processor to Largest Task First*
DVS	*Dynamic Voltage Scaling*
EDF	*Earliest Deadline First*
EES	*Energy Efficient Scheduling*
ESS	*Extended Service Set*
FIFS	*First Input First Service*
FP6-IST	*Information Society Technologies in the 6th Framework Programme*
GC	*Grid Controller*
GPS	*Global Positioning System*
ICMP	*Internet Control Message Protocol*
IP	*Internet Protocol*
OGSA	*Open Grid Services Architecture*
P2P	*Peer-to-Peer*
PDA	*Personal Digital Assistant*
PLI	*Programação Linear Inteira*
QoS	*Quality of Service*
SEAS	*Simple Energy-Aware Scheduler*
SSP	*Subset-Sum Problem*
TCP	*Transmission Control Protocol*
UDP	*User Datagram Protocol*
WAP	*Wireless Access Point*
WLAN	*Wireless Local Area Network*
WQ	*Workqueue*
WQR	*Workqueue with Replication*
XML	*eXtensible Markup Language*

Capítulo 1

Introdução

Na década de 1990, o aumento da complexidade das aplicações científicas motivou o desenvolvimento de uma infraestrutura de computação distribuída inspirada nas redes de energia elétrica (Buyya et al., 2005). Nessa infraestrutura, conhecida como grade computacional, recursos computacionais (por exemplo, processadores, discos e licenças de *software*) geograficamente distribuídos podem ser compartilhados de maneira coordenada (Foster et al., 2001). Dessa forma, problemas complexos que demandam grande quantidade de recursos podem ser resolvidos de maneira colaborativa.

Inicialmente concebido considerando-se apenas o compartilhamento de recursos entre nós fixos, o conceito de grade computacional tem sido estendido gradativamente para ambiente móveis (Furthmüller & Waldhorst, 2010; Rodriguez et al., 2011). A integração entre dispositivos móveis (por exemplo, *smartphones*, *tablets* e *notebooks*) e grades computacionais, denominada de grade móvel (*mobile grid*), pode ser realizada de duas maneiras distintas. Na primeira abordagem, os dispositivos móveis apenas consomem recursos computacionais, funcionando como uma interface de acesso à grade. Já na segunda abordagem, os dispositivos móveis podem atuar como provedores de recursos, tendo a possibilidade de participar diretamente da execução colaborativa de aplicações.

Inviável há alguns anos, a concepção de uma grade móvel voltada para o compartilhamento de recursos pode ser atualmente justificada por dois motivos (Furthmüller & Waldhorst, 2010; Rodriguez et al., 2011). Primeiro, a rápida evolução tecnológica dos dispositivos móveis tanto em termos de *hardware* quanto de *software*. Segundo, o grande número de dispositivos móveis, em especial *smartphones*, comercializados anualmente. De acordo com dados da consultoria Gartner[1], apenas em 2011, foram vendidos mundialmente mais de 470 milhões de *smartphones*, caracterizando um aumento de 58% em relação ao ano anterior.

[1] http://www.gartner.com/it/page.jsp?id=1924314

Existe potencialmente, portanto, uma grande quantidade de recursos computacionais que pode ser explorada por meio de uma grade móvel no desenvolvimento de aplicações para áreas diversas. Por exemplo, grades móveis têm sido utilizadas em aplicações sensíveis ao contexto (Coronato & Pietro, 2008), *e-healthcare* (Viswanathan et al., 2012a,b), *e-learning* (D'Andria et al., 2008), gerenciamento de instalações agrícolas (Wu et al., 2011), bem como na execução de aplicações científicas (Rodriguez et al., 2012a).

Deve-se ressaltar, no entanto, que uma infraestrutura para compartilhamento de recursos ofertados por dispositivos móveis constitui um ambiente altamente volátil. A disponibilidade dos recursos pode ser afetada por problemas associados à mobilidade, comunicação e capacidade de bateria, fazendo como que o desempenho das aplicações submetidas seja prejudicado.

Particularmente, a capacidade de bateria é um fator crítico, pois, além de afetar a disponibilidade, influencia diretamente os usuários de dispositivos móveis na decisão de compartilhar ou não seus recursos computacionais. Em uma pesquisa realizada por Furthmüller e Waldhorst (Furthmüller & Waldhorst, 2012), constatou-se que a capacidade de energia de um dispositivo é preponderante para essa decisão. Mais da metade dos entrevistados (53%) afirmaram que não compartilhariam seus recursos em virtude do tempo de vida de bateria disponível.

É fundamental, portanto, que uma grade móvel seja capaz de gerenciar o consumo de energia dos dispositivos participantes, sem que o desempenho do sistema seja comprometido. Nesse sentido, faz-se necessário a adoção de uma política de escalonamento que consiga estabelecer um compromisso entre o consumo de energia e os requisitos de qualidade de serviço das aplicações submetidas à grade móvel.

1.1 Objetivo

O principal objetivo deste trabalho é estudar o problema de minimização do consumo de energia no contexto de uma grade móvel voltada para o compartilhamento de recursos. Para tanto, foram propostos dois algoritmos de escalonamento que, além de buscarem minimizar o consumo de energia nos dispositivos móveis, visam assegurar o cumprimento dos requisitos de qualidade de serviço das aplicações submetidas pelos usuários.

1.2 Metodologia

O projeto dos algoritmos de escalonamento considerou soluções heurísticas para o problema de escalonamento ciente de consumo de energia em grades móveis, que foi formulado como um modelo de programação linear inteira (PLI). A avaliação de desempenho dos algoritmos propostos teve como base dois cenários. No primeiro, estático, por meio de experimentos numéricos, foi mensurada a acurácia dos algoritmos propostos, tendo-se como referência as soluções obtidas pelo *solver* Gurobi. Já no segundo cenário, dinâmico, utilizou-se simulação para avaliar o comportamento dos algoritmos em um ambiente in-

stável, onde podem ocorrer variações tanto na taxa de chegada das aplicações quanto no nível de bateria dos dispositivos móveis.

1.3 Organização da Dissertação

Esta dissertação está organizada da seguinte maneira:

Capítulo 2 Esse capítulo aborda os conceitos utilizados neste trabalho, sendo também apresentada uma discussão sobre trabalhos relacionados. Uma maior ênfase é dada à caracterização das grades móveis, bem como à discussão de estratégias de escalonamento cientes do consumo de energia.

Capítulo 3 Nesse capítulo, é feita a proposição de dois algoritmos de escalonamento cientes do consumo de energia para grades móveis. Primeiramente, é apresentada a formulação do problema de escalonamento no contexto de uma grade móvel como um modelo de programação linear inteira, para, em seguida, serem discutidos detalhes sobre o projeto dos algoritmos propostos.

Capítulo 4 Nesse capítulo, são descritos os experimentos executados para avaliar o desempenho dos algoritmos de escalonamento propostos. Também é realizada uma discussão sobre os resultados obtidos nas diferentes configurações de experimentos consideradas.

Capítulo 5 Esse capítulo apresenta conclusões sobre o trabalho realizado, sendo também discutidos possíveis trabalhos futuros envolvendo grades móveis.

Capítulo

2

Referencial Teórico

Neste capítulo, serão apresentados os conceitos básicos relacionados ao projeto de mestrado. Inicialmente serão caracterizadas as grades computacionais. Em seguida, a integração entre dispositivos móveis e grades computacionais, denominada grade móvel, será abordada. Também será discutido o escalonamento de tarefas em grade computacionais. Por fim, será apresentada uma avaliação sobre os trabalhos relacionados.

2.1 Grades Computacionais

Na década de 1990, o aumento da potência computacional necessária para a execução de aplicações científicas motivou a criação de uma infraestrutura de computação distribuída denominada grade computacional (Buyya et al., 2005). Nessa infraestrutura, computadores independentes, que podem estar geograficamente distribuídos, são interconectados de maneira a possibilitar o compartilhamento coordenado de recursos, tais como processadores, discos e canais de comunicação (Foster et al., 2001). Dessa forma, problemas que possuem alto custo computacional podem ser resolvidos colaborativamente, aproveitando-se o potencial de cada um dos participantes da grade.

Uma grade computacional deve prover acesso consistente aos recursos disponíveis, fazendo com que as descontinuidades físicas (diferenças entre tipos de *hardware*, sistemas operacionais e canais de comunicação) se tornem completamente transparentes aos usuários. Nesse sentido, define-se uma grade computacional como sendo uma infraestrutura de *hardware* e de *software* que fornece acesso confiável, consistente e transparente, possibilitando a busca, descoberta e compartilhamento de recursos computacionais, independentemente da localização geográfica (Baker et al., 2002). A utilização de uma grade computacional, portanto, faz com que um ambiente heterogêneo seja transformado em um ambiente de compartilhamento virtual e homogêneo.

De acordo com Krauter et al. (2002), as grades computacionais podem ser categorizadas em função da atividade principal a que se destinam. Nessa taxonomia, são consideradas três categorias, descritas a seguir.

Grade de Processamento (*Computacional Grid*) A capacidade de processamento dos recursos que compõem a grade é utilizada para resolver problemas com alto custo computacional. Um exemplo é o *Folding@home*[1], desenvolvido pela universidade de Stanford. Nesse projeto, computadores espalhados pelo mundo são utilizados para estudar problemas relacionados ao dobramento de proteínas (*protein folding*). Cada computador possui um programa cliente instalado que, além de executado em *background*, faz uso do processador apenas em momentos de ociosidade.

Grade de Dados (*Data Grid*) A finalidade deste tipo de grade é oferecer uma infraestrutura para armazenamento e gerenciamento de dados. A integração de bancos de dados heterogêneos distribuídos geograficamente constitui um exemplo deste tipo de grade.

Grade de Serviços (*Service Grid*) Os recursos computacionais disponíveis são utilizados para prover serviços que dificilmente poderiam ser oferecidos individualmente. Um exemplo são as plataformas de colaboração à distância, que permitem a integração de usuários e aplicações em ambientes virtuais compartilhados.

2.1.1 Organizações Virtuais

Como enfatizado anteriormente, um dos principais objetivos de uma grade computacional é o gerenciamento coordenado de recursos, tarefa que deve ser realizada de forma flexível e segura. Para tanto, são necessárias regras que definam o que pode ser compartilhado, quando e por quem. Um conjunto de usuários que sigam tais regras constitui uma organização virtual (*virtual organization*) (Foster et al., 2001).

As organizações virtuais são compostas para atender a diversas finalidades, podendo assim variar em muitos aspectos, como tamanho, escopo, duração e estrutura (Foster et al., 2001). Os usuários pertencentes a uma organização virtual definem regras para controlar o acesso a seus respectivos recursos. Tais regras, além de políticas de compartilhamento (autorização), estabelecem como deve ser realizada a identificação de usuários e recursos (autenticação).

Considerando-se os problemas relacionados à criação e gerenciamento de organizações virtuais, Foster et al. (2001) propuseram a arquitetura para grades computacionais descrita a seguir.

[1] http://folding.stanford.edu/

2.1.2 Arquitetura

Como uma grade computacional deve constituir um ambiente onde qualquer membro em potencial possa compartilhar seus recursos, a interoperabilidade se apresenta como um aspecto essencial (Foster et al., 2001). De maneira similar a uma rede de computadores, a interoperabilidade de uma grade computacional é obtida por meio de protocolos. Logo, uma grade pode ser vista essencialmente como uma arquitetura de protocolos, os quais definem as regras básicas que controlam o compartilhamento de recursos (Foster et al., 2001).

Na arquitetura proposta por Foster et al. (2001), os protocolos são organizados em uma estrutura composta por quatro camadas: **Ambiente**, **Conectividade e Segurança**, **Coletivo** e **Aplicação**. Em razão do menor número de protocolos nas camadas internas, a representação da arquitetura pelo modelo de camadas possui um formato semelhante ao de uma ampulheta (Figura 2.1).

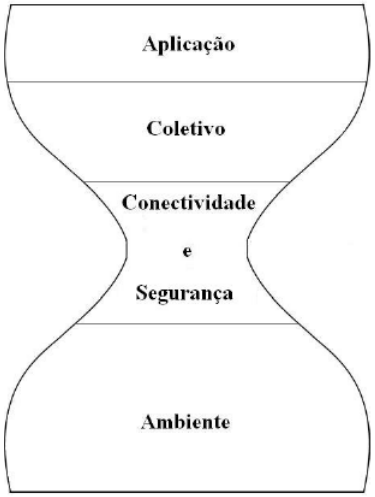


Figura 2.1: Organização em camadas de uma Grade Computacional. Adaptado de Foster (2002).

A camada **Ambiente** é responsável pela acomodação dos diversos recursos compartilhados, que podem ser tanto físicos (discos e processadores, por exemplo) quanto lógicos (sistemas de arquivos, licenças de software, entre outros). São oferecidos protocolos e mecanismos que implementam as funcionalidades fornecidas pelos recursos. Essas implementações podem ser fornecidas tanto pelo fabricante do recurso quanto pelo *middleware* da grade computacional.

A camada **Conectividade e Segurança** define um conjunto de protocolos necessários para atender aos requisitos de comunicação e autenticação das transações de uma grade computacional. Os protocolos de comunicação como IP (*Internet Protocol*), ICMP (*Internet Control Message Protocol*), TCP (*Transmission Control Protocol*) e UDP (*User Datagram Protocol*) compõem essa camada. Já os aspectos de segurança são implementados por meio de mecanismos de autenticação e criptografia. Em uma organização virtual, os mecanismos

de autenticação devem possuir as seguintes características:

***Log-on* único** Após o processo de autenticação para utilizar um dado recurso da grade, um usuário deve poder utilizar outros recursos sem a necessidade de uma nova autenticação.

Delegação É necessário que um usuário possa atribuir a um determinado programa os seus direitos, de forma que o mesmo atue no sistema em seu nome. Esse programa deve, opcionalmente, ser capaz de atribuir a outros programas.

Integração com soluções locais de segurança Existem várias soluções de segurança locais que podem ser empregadas pelos provedores de recursos. A grade deve, portanto, ser capaz de interagir com esses sistemas.

Relacionamentos confiáveis baseados no usuário Caso um usuário utilize recursos de múltiplos provedores conjuntamente, não se deve exigir a cooperação/interação entre esses provedores para o estabelecimento da segurança do ambiente. Por exemplo, se um usuário possui o direito de utilizar os recursos A e B, ele deve ser capaz de utilizá-los em conjunto sem que haja a interação entre entre os provedores de A e B.

Na camada **Coletivo**, encontram-se protocolos, serviços e APIs (*Application Programming Interfaces*) associados a coleções de recursos. Nessa camada, são disponibilizados serviços de diretórios, bem como os serviços de alocação e escalonamento. Os serviços de diretório armazenam e disponibilizam informações sobre os recursos ofertados, permitindo que os usuários da grade possam consultá-los. Já os serviços de alocação e escalonamento visam alocar recursos de maneira apropriada, atendendo as especifidades das requisições submetidas à grade.

Por fim, a camada **Aplicação** é formada pelas aplicações que utilizam os recursos da grade computacional.

2.1.3 Exemplos de *Middleware*

O ambiente de uma grade computacional possui duas características principais a heterogeneidade e a dinamicidade dos recursos. Logo, para que a grade computacional possa funcionar adequadamente é necessária a existência de uma camada de *software*, denominada de *middleware*, responsável pelo gerenciamento seguro e transparente desses recursos. A seguir são apresentados alguns exemplos de *middleware* para grades computacionais.

Globus Toolkit

Atualmente em sua quinta versão, o *Globus Toolkit* (Foster & Kesselman, 1996; Foster, 2005) é uma ferramenta de código aberto que possibilita o compartilhamento seguro e

transparente de recursos entre organizações virtuais. Ele provê um conjunto de programas, serviços e bibliotecas que auxiliam e possibilitam a concepção de grades computacionais, incluindo mecanismos de segurança, gerenciamento de recursos e dados, detecção de falhas e portabilidade.

Os componentes que constituem o *Globus Toolkit* são implementados com base na especificação OGSA (*Open Grid Services Architecture*), proposta pelo comitê *The Globus Alliance*[2]. Como ilustrado na Figura 2.2, esses componentes são agrupados de acordo com sua finalidade em quatro classes: segurança, gerenciamento de dados, gerenciamento de execução e ambiente de tempo de execução.

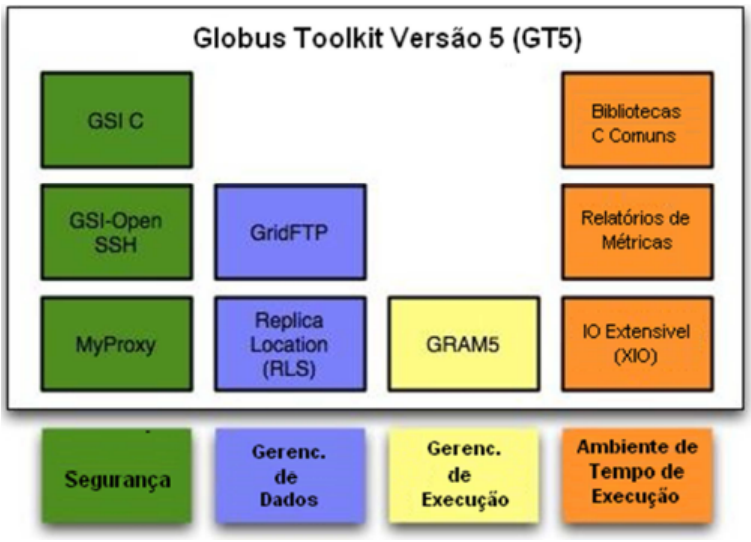


Figura 2.2: Organização dos componentes que compõem o Globus Toolkit versão 5.

OurGrid

O *OurGrid*[3] (Cirne et al., 2003; Andrade et al., 2005), desenvolvido pela Universidade Federal de Campina Grande, funciona como uma rede *peer-to-peer* (P2P) de compartilhamento de recursos. Uma grade baseada no *OurGrid* é caracterizada como uma rede de favores, na qual um usuário oferece de maneira voluntária recursos ociosos para utilização de terceiros, bem como pode utilizar recursos remotos quando sua capacidade local não é suficiente para a execução de uma dada aplicação.

Deve-se ressaltar que o *OurGrid* apenas executa aplicações do tipo *bag-of-tasks*, isto é, compostas por tarefas independentes, que não possuem comunicação entre si.

BOINC

O BOINC[4], desenvolvido e mantido pela Universidade da Califórnia em Berkeley, é um *middleware* voltado para a implementação de projetos de computação voluntária baseados

[2]http://www.globus.org/ogsa/
[3]http://www.ourgrid.org/
[4]http://boinc.berkeley.edu/

em grades computacionais. Diversos projetos científicos utilizam o BOINC para explorar a capacidade ociosa de computadores voluntários espalhados por todo o mundo. Dois exemplos desses projetos são: o *LHC@home*[5], voltado para a análise de dados referentes a experimentos de colisão de partículas executados pelos laboratórios do CERN; e o *Rosseta@home*[6], que assim como o *Folding@home*, é utilizado para a realização de estudos relativos à conformação tridimensional de estruturas proteicas.

A execução de aplicações em computadores voluntários é passível de erros tanto por mau funcionamento quanto por comportamento malicioso do usuário, que pode fazer alterações nos resultados obtidos. Para evitar esse tipo de problema, é feita a replicação da carga de trabalho, sendo os resultados obtidos comparados posteriormente. Tendo-se obtido um consenso, o resultado é considerado verdadeiro; caso contrário, é realizada uma nova requisição de processamento.

2.2 Grade móveis

Durante a década passada, os dispositivos móveis sofreram uma evolução tecnológica significativa em termos tanto de *hardware* quanto de *software*. De simples telefones e PDAs (*Personal Digital Assistants*), transformaram-se em estações de trabalho portáteis, que permitem realizar operações antes executadas apenas por computadores pessoais. Tal evolução fez com que esses dispositivos passassem a ser opções viáveis no contexto de sistemas de computação distribuída. Particularmente, a integração entre dispositivos móveis e grades computacionais, denominada de grade móvel (*mobile grid*), tem despontado como um tópico de pesquisa promissor (Coronato & Pietro, 2008; Black & Edgar, 2009; Ghosh & Das, 2010; Furthmüller & Waldhorst, 2012; Viswanathan et al., 2012a,b; Rodriguez et al., 2012b).

Há basicamente duas abordagens para a concepção de uma grade móvel. Na primeira, os dispositivos móveis apenas consomem recursos computacionais, funcionando como uma interface de acesso à grade. Por outro lado, na segunda abordagem, os dispositivos móveis podem atuar como provedores de recursos, tendo a possibilidade de participar diretamente da execução das aplicações submetidas à grade. A seguir, são discutidas cada uma dessas duas abordagens.

2.2.1 Dispositivos Móveis como Consumidores de Recursos

Nesta abordagem, parte-se do princípio que os dispositivos móveis são limitados, particularmente em termos de processamento, quando comparados a computadores pessoais, portanto, podendo apenas atuar como consumidores de recursos. No papel de consumidor, um dispositivo móvel pode utilizar recursos ofertados pela grade a fim de aumentar sua capacidade de execução de aplicações ou reduzir seu consumo de energia.

[5]http://lhcathome.web.cern.ch/
[6]http://boinc.bakerlab.org/

A interação com a grade pode ser dificultada por aspectos como desconexões e comunicação intermitente. São necessárias, portanto, medidas que garantam acesso transparente e confiável aos recursos disponibilizados. Uma possível solução para tal problema é a utilização de uma arquitetura baseada em *proxies* como a proposta por Park et al. (2003). Nessa arquitetura, os *proxies* são elementos mediadores, sendo responsáveis pela submissão, monitoramento e retorno das requisições realizadas pelos dispositivos móveis.

Outro exemplo é a arquitetura proposta por Borges et al. (2009), que possibilita a submissão e monitoramento de aplicações *workflow*[7] submetidas por usuários móveis. Nela, são fornecidos mecanismos para apoiar o fluxo de execução das aplicações, possibilitando que modificações adaptativas sejam realizadas em resposta a problemas ocorridos no ambiente móvel. Esses mecanismos são baseados em características não funcionais, como consumo de energia e confiabilidade.

2.2.2 Dispositivos Móveis como Provedores de Recursos

A adoção desta abordagem pode ser justificada pelo crescente número de dispositivos móveis comercializados, juntamente com a evolução tecnológica desses dispositivos em termos de processamento, armazenamento e comunicação (Furthmüller & Waldhorst, 2010). Especificamente em relação à capacidade de processamento, em um estudo realizado por Rodriguez et al. (2012a), demonstrou-se que a utilização de um sistema distribuído composto por *smartphones* pode ser útil na resolução de problemas que envolvam algoritmos iterativos com predomínio de operações inteiras.

De acordo com Katsaros & Polyzos (2007), provisão de recursos móveis pode ser realizada considerando-se duas arquiteturas fundamentalmente distintas: grades móveis *on-site* e grades móveis *ad hoc*.

As grades móveis *on-site* dependem necessariamente de uma infraestrutura de rede preexistente, sendo os dispositivos móveis coordenados por uma entidade central. Informações sobre a localização e disponibilidade dos recursos são monitoradas e repassadas para a entidade central, que mantém uma lista com o *status* de cada dispositivo. Um exemplo de arquitetura *on-site* foi proposta por Ghosh & Das (2010). Tal arquitetura, ilustrada na Figura 2.3, baseia-se em uma rede de telefonia celular na qual cada célula é composta por um conjunto de dispositivos móveis e um ponto de acesso sem fio (WAP, do inglês *Wireless Access Point*). Como os autores consideraram o protocolo IEEE 802.1, cada célula caracteriza um conjunto básico de serviços (BSS, do inglês *Basic Service Set*). Em cada BSS, existe um servidor WAP que é responsável pelo monitoramento dos recursos disponibilizados pelos dispositivos móveis. Esses servidores estão conectados a um roteador de borda, que serve de interface com a entidade que controla a grade móvel (GC, do inglês *Grid Controller*). Múltiplos BSSs podem ser conectados para formar um conjunto

[7] Aplicações nas quais a execução de uma tarefa deve ser finalizada antes que sua sucessora comece a ser executada.

estendido de serviços (ESS, do inglês *Extended Service Set*). Com relação à mobilidade, os dispositivos podem migrar entre diferentes BSSs desde que estes pertençam a um mesmo ESS.

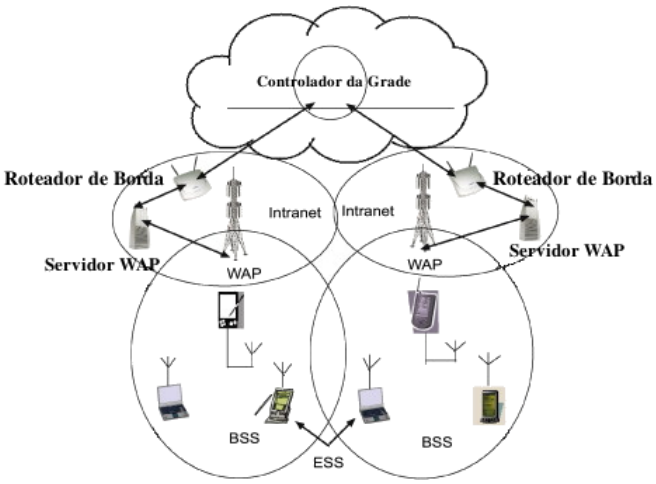


Figura 2.3: Arquitetura de grade móvel *on-site* com base em uma rede de telefonia celular. Fonte: Ghosh & Das (2010).

Já as grades móveis *ad hoc* não dependem de uma infraestrutura de rede preestabelecida. A topologia da rede é formada de maneira dinâmica e independente, com os dispositivos móveis podendo participar ou sair da grade a qualquer momento, criando-se assim cenários de compartilhamento temporários (Figura 2.4). Além disso, tanto a demanda quanto a disponibilidade de serviços e recursos podem apresentar uma alta taxa de variação, em comparação com os cenários encontrados nas grades *on-site* (Lima, 2007).

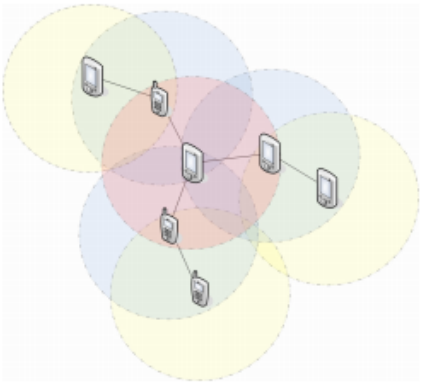


Figura 2.4: Cenários de compartilhamento temporários em uma grade *ad hoc*. Cada círculo tracejado representa o alcance de transmissão do nó posicionado no respectivo centro. Fonte: Lima (2007).

2.2.3 Desafios

Devido principalmente a aspectos como mobilidade, capacidade de bateria e segurança, a concepção de uma grade móvel voltada para o compartilhamento de recursos é uma

tarefa que apresenta diversos desafios. Além disso, outro aspecto essencial está relacionado a como incentivar usuários de dispositivos móveis a compartilhar seus recurso computacionais. A seguir, são discutidos alguns dos problemas que devem ser enfrentados durante o projeto de uma grade móvel, visto que afetam diretamente o desempenho do sistema.

Mobilidade

O padrão de mobilidade dos dispositivos que compõem uma grade móvel pode levar a falhas na execução de aplicações, bem como a um aumento no custo de comunicação (Shah & Park, 2012). Além disso, no contexto de uma grade móvel *ad hoc* de saltos múltiplos, um dispositivo pode funcionar como elemento intermediário na transmissão dos dados de uma aplicação sendo executada em dispositivos vizinhos. Logo, a mobilidade de um único dispositivo pode ter um grande impacto sobre o desempenho da aplicação.

Assumindo-se que o padrão de movimentação de um dispositivo pode ser caracterizado com um processo estocástico estacionário (média constante), Ghosh & Das (2010) propuseram um mecanismo de controle de mobilidade para a arquitetura descrita na Figura 2.3. Cada atualização relativa à localização dos dispositivos móveis pertencentes a um BSS é repassadas ao GC. A partir da análise dos padrões de mobilidades armazenados, o GC realiza predições sobre o tempo que um recurso estará sob responsabilidade de um determinado WAP.

Capacidade de Bateria

A capacidade de bateria é um fator crítico em uma grade móvel, pois influencia diretamente os proprietários de dispositivos móveis na decisão de compartilhar ou não seus recursos computacionais. Em uma pesquisa realizada por Furthmüller e Waldhorst (Furthmüller & Waldhorst, 2012), foi constatado que a capacidade de bateria de um dispositivo é preponderante para essa decisão. Mais precisamente, 53% dos entrevistados afirmaram que não compartilhariam seus recursos em virtude do tempo de vida de bateria disponível.

Além disso, o gerenciamento ineficaz dos recursos ofertados pode afetar significativamente o custo de consumo de energia e de comunicação, limitando o tempo de vida de bateria dos dispositivos. Nesse sentindo, Black & Edgar (2009) sugerem que o cenário mais favorável para o compartilhamento em uma grade móvel ocorreria durante os períodos de recarga dos dispositivos participantes, especialmente à noite. Nesses períodos, geralmente os dispositivos tem acesso à rede, além de possuírem ciclos de ociosos de CPU.

Como a taxa de evolução da capacidade das baterias de íons de lítio foi insignificante quando comparada à evolução das funcionalidades dos dispositivos móveis (Paradiso & Starner, 2005; Rodriguez et al., 2012a), pode-se afirmar que, no médio prazo, a utilização adequada da energia desses dispositivos ainda será um ponto chave para o desempenho

das grades móveis.

Segurança

O compartilhamento de recursos em uma grade móvel exige um alto nível de segurança, visto que aplicações de terceiros são executadas no sistema local. Nesse sentindo, um usuário de dispositivo móvel deve ser capaz de estabelecer regras definindo quais recursos podem ser compartilhados. Além disso, fazem-se necessários mecanismos que garantam a segurança de dados pessoais armazenados nos dispositivos, tais como fotos, vídeos, contatos e históricos de chamadas (Rosado et al., 2011).

Incentivo à participação

Proprietários de dispositivos móveis podem possuir motivações diferentes para compartilhar seus recursos computacionais: alguns podem participar voluntariamente, de maneira semelhante a projetos como *Folding@home*, enquanto outros podem estar interessados em alguma forma de compensação. Como incentivar os proprietários de dispositivos a compartilhar seus recursos computacionais é um fator chave para o sucesso de uma grade móvel, há alguns trabalhos na literatura que discutem mecanismos de incentivo à participação (Teske et al., 2011; Duan et al., 2012; Li & Shen, 2012).

Particularmente, no trabalho de Duan et al. (2012), são apresentados e avaliados mecanismos de colaboração baseados em recompensas. Os autores consideram que, ao submeter uma aplicação, o usuário define uma recompensa que pode ser dividida entre os proprietários dos diversos dispositivos móveis participantes. Os autores discutem como um usuário pode definir um contrato ótimo, especificando recompensas baseadas nas características particulares de cada dispositivo. Por exemplo, dispositivos com maior potência de processamento podem receber recompensas maiores, para incentivá-los a receber mais expressivas quantidades de carga de trabalho.

2.2.4 Exemplos de Middleware

Para que uma grade móvel tenha um desempenho satisfatório, o *middleware* deve gerenciar adequadamente aspectos como mobilidade, comunicação e capacidade de bateria. A seguir, são apresentados alguns exemplos de *middleware* voltados para ambientes móveis.

Akogrimo

Akogrimo foi um projeto financiado pelo programa FP6-IST (*Information Society Technologies in the 6th Framework Programme*) da Comissão Europeia. O projeto durou aproximadamente três anos (Julho de 2004 a Setembro de 2007), tendo sido pioneiro no desenvolvimento de soluções voltadas para a integração de dispositivos móveis ao ambiente de grades computacionais.

O *middleware* do projeto Akogrimo possui duas camadas lógicas, sendo uma de serviços de rede e outra de serviços de infraestrutura da grade. A camada de rede é baseada no

protocolo IPV6 móvel, fornecendo serviços como autenticação, autorização, tarifação e cobrança. A partir dos serviços da camada de rede, a camada de infraestrutura da grade provê funcionalidades voltadas para execução e monitoramento de aplicações em ambientes móveis (Stiller & Waldburger, 2005). Dentre os serviços fornecidos por essa camada, destaca-se o gerenciamento de organizações virtuais móveis, responsável por controlar o compartilhamento dos recursos disponibilizados.

MiPeG

MiPeG (Coronato & Pietro, 2008) é um *middleware* para grades computacionais voltado para execução de aplicações ubíquas. Ele consiste de um conjunto de serviços que seguem a especificação OGSA. Além da provisão de contexto, esses serviços oferecem mecanismos para a integração e gerenciamento de dispositivos móveis.

A integração dos dispositivos móveis à grade é realizada de forma a permitir que esses dispositivos atuem ativamente como provedores de recursos. Cada dispositivo integrado é automaticamente registrado como um recurso ativo e passa a ser monitorado pela grade. Assim, por exemplo, é possível que um *smartphone*, além de executar tarefas submetidas à grade, possa oferecer em forma de serviços recursos como GPS (*Global Positioning System*) e acelerômetro.

Grid Anywhere

Apesar de ter sido desenvolvido inicialmente para o compartilhamento de recursos entre computadores e *set-top boxes* (receptores de TV digital), o *middleware Grid Anywhere* (Teixeira et al., 2010; Teixeira, 2012) é extensível para ambientes móveis. Em uma grade baseada no *Grid Anywhere*, cada dispositivo participante representa um nó em uma rede P2P. O compartilhamento de recursos é realizado de maneira bidirecional, possibilitando que os dispositivos participantes atuem tanto como provedores quanto consumidores de recursos.

O compartilhamento de recursos é feito por meio da migração de objetos Java. Uma vez que um objeto é transportado para um dado participante, seus métodos podem ser invocados remotamente e o objeto pode, posteriormente, regressar à origem. A execução de um objeto é feita em um ambiente seguro (*sandbox*), assim garantindo que operações que possam prejudicar o recurso hospedeiro não sejam executadas.

Abordagem de Viswanathan et al.

Viswanathan et al. (2012b) propuseram um *middleware* voltado para o desenvolvimento de aplicações de *e-healthcare* sensíveis ao contexto. Ele permite que dispositivos heterogêneos em uma determinada vizinhança formem um conjunto elástico de recursos computacionais, que possa ser aproveitado para processar coletivamente grandes quantidades de dados obtidos por meio de sensores médicos.

Como ilustrado na Figura 2.5, os dispositivos participantes atuam como provedores

de serviços ou árbitros. Os provedores de serviços podem prover dados (por exemplo, monitor cardíaco), recursos (por exemplo, *tablets* e *smartphones*) ou ambos. Já um dispositivo árbitro funciona como um *broker*, sendo responsável pelo atendimento das requisições submetidas pelos usuários (neste caso, profissionais de saúde), bem como pelo gerenciamento dos servidores de dados e recursos.

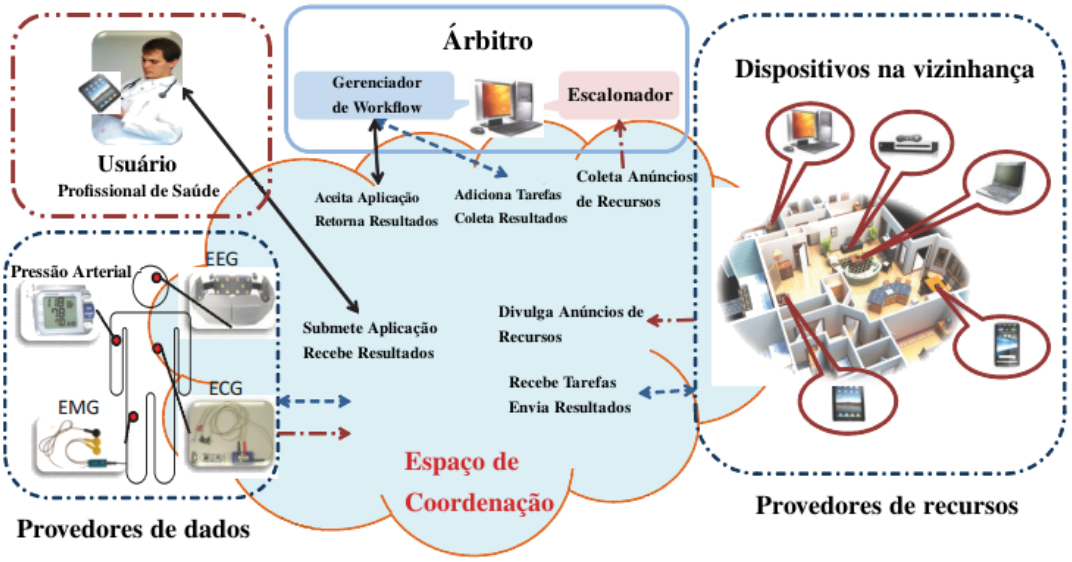


Figura 2.5: Grade móvel voltada para aplicações de *e-Healthcare* proposta por Viswanathan et al. (2012b). Fonte: Viswanathan et al. (2012b).

O escalonamento das tarefas é baseado em uma política ciente de consumo de energia, que visa maximizar o tempo de funcionamento de cada provedor de serviço, possibilitando que a heterogeneidade do conjunto de recursos seja mantida por períodos mais longos. Além disso, foram implementados mecanismos que permitem que a grade móvel se auto-organize em resposta a eventos de desconexão dos provedores de serviços.

2.3 Escalonamento de Tarefas em Grades Computacionais

Em um cenário típico de grade computacional, as aplicações submetidas para execução podem gerar um grande número de tarefas[8], tornando-se necessário, portanto, ter mecanismos que possibilitem o escalonamento automático e eficiente dessas tarefas nos recursos adequados (Xhafa & Abraham, 2010). Considerando-se a arquitetura apresentada na Seção 2.1, o escalonamento de tarefas é caracterizado como um serviço da camada Coletivo, que é provido por um componente de *software* denominado escalonador.

De forma geral, o processo de escalonamento de tarefas (Figura 2.6) pode ser dividido em três etapas sequenciais, executadas pelo escalonador de maneira transparente para

[8]Neste trabalho, considerada-se uma tarefa como sendo a unidade básica de trabalho a ser executada como parte de uma aplicação.

o usuário: descoberta dos recursos, seleção dos recursos baseada em uma política de escalonamento, e execução das tarefas (Dong & Akl, 2006).

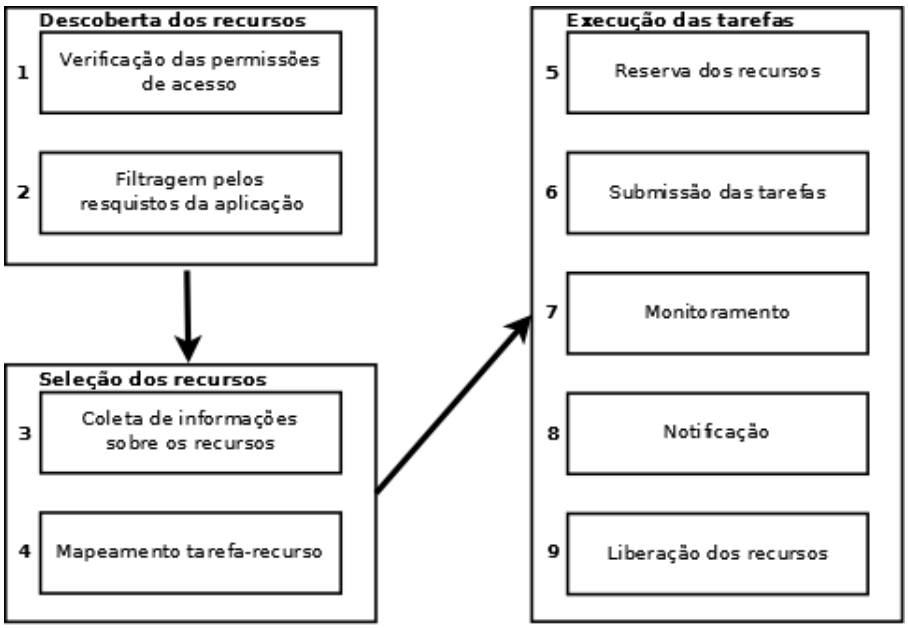


Figura 2.6: Etapas do escalonamento de tarefas em grades computacionais. Adaptado de Teodoro (2013).

O processo se inicia com a obtenção de um conjunto de recursos para o qual o usuário tem permissão de acesso (Passo 1). Em seguida, o conjunto passa por um processo de filtragem que leva em consideração os requisitos impostos pela aplicação submetida (Passo 2). A partir do conjunto filtrado, com base em uma política de escalonamento, são selecionados os recursos que irão executar as tarefas que compõem a aplicação. Esta etapa é realizada em dois passos: coleta de informações sobre o *status* de cada recurso (Passo 3), e determinação do plano de escalonamento (Passo 4), isto é, em qual recurso cada tarefa será mapeada. Antes do envio das tarefas para execução (Passo 6), um processo de reserva antecipada (Passo 5) pode ser realizado para garantir que os recursos selecionados estejam disponíveis no momento adequado. Durante a execução da aplicação, cada tarefa pode ter seu progresso monitorado (Passo 7). Dependendo da política de escalonamento adotada, o monitoramento pode ser utilizado para determinar quando se faz necessário o re-escalonamento das tarefas (migração). Ao final da execução da aplicação, o usuário recebe uma notificação (Passo 8), e os recursos alocados são liberados (Passo 9).

O escalonamento pode ser realizado por um ou mais escalonadores, caracterizando três possíveis modelos de organização: centralizado, hierárquico e descentralizado (Krauter et al., 2002). A adoção de um modelo apropriado é importante para a escalabilidade, autonomia e desempenho do sistema.

No modelo centralizado, um único escalonador controla todos os recursos ofertados na grade. Já no modelo hierárquico, o mapeamento das tarefas é realizado de forma coordenada por escalonadores organizados em níveis. Por exemplo, em uma organização

que considera dois níveis de hierarquia, cada escalonador localizado no nível inferior controla um conjunto de recursos. Por outro lado, os escalonadores localizados no nível superior, conhecidos como meta-escalonadores, coordenam um ou mais escalonadores de nível inferior. Por fim, no modelo descentralizado, vários escalonadores sem controle central decidem sobre o mapeamento das tarefas.

2.3.1 Taxonomia dos Algoritmos de Escalonamento

Casavant & Kuhl (1988) propuseram uma classificação hierárquica dos algoritmos de escalonamento em sistemas computacionais distribuídos, ilustrada na Figura 2.7. Nessa taxonomia, os algoritmos são primeiramente divididos como locais e globais. O escalonamento local envolve o mapeamento das fatias de tempo de uso de um único processador às tarefas residentes localmente. Por outro lado, o escalonamento global se refere ao problema de decisão de quais dos diversos recursos disponíveis as tarefas serão executadas. O problema de escalonamento em grades computacionais, obviamente, pertence ao ramo global, que pode ser subdividido em mais duas classes de escalonamento: estático e dinâmico.

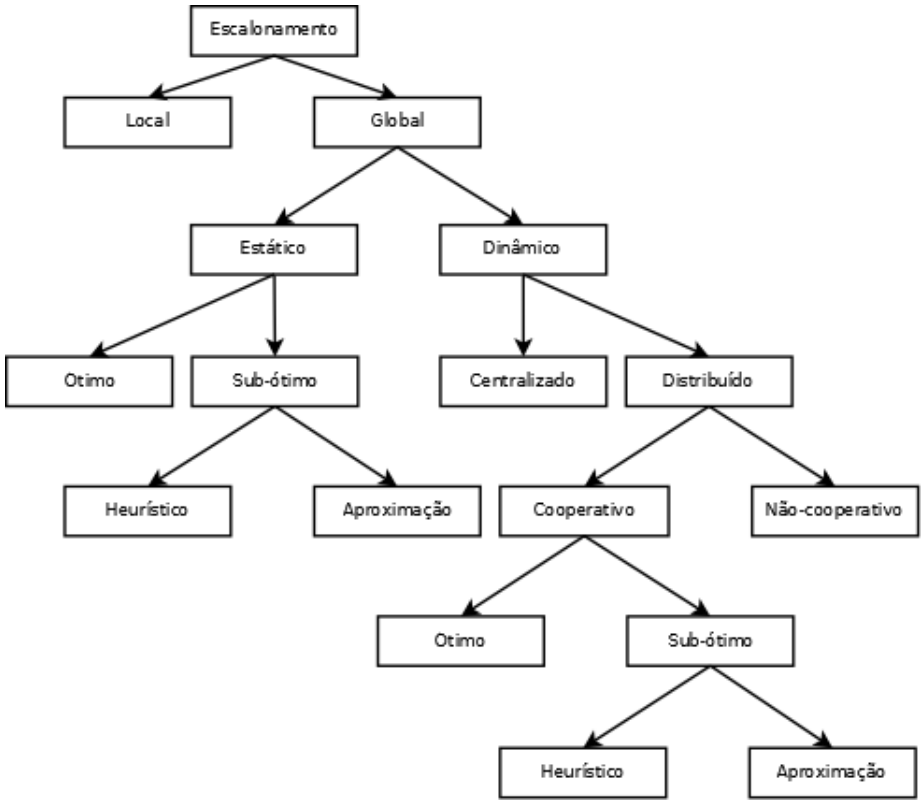


Figura 2.7: Taxonomia para o problema de escalonamento proposta por Casavant & Kuhl (1988). Adaptado de Dantas (2005).

No escalonamento estático, o mapeamento das tarefas é realizado antes da execução da aplicação, sendo, portanto, necessário o conhecimento antecipado de informações acerca dos recursos. Mesmo em caso de falha de algum recurso da grade, o mapeamento

obtido não pode ser alterado em tempo de execução. Por outro lado, no escalonamento dinâmico, assume-se que muito pouco se sabe sobre os recursos que compõem a grade, sendo o mapeamento de tarefas realizado durante o tempo de vida da aplicação. Devido à natureza dinâmica das grades, em geral, faz-se também necessária a redistribuição das tarefas entre os recursos computacionais.

Os algoritmos de escalonamento também podem ser classificados como ótimos e sub-ótimos. Um escalonamento ótimo pode apenas ser realizado quando se há conhecimento prévio de todas as informações relativas aos recursos e à aplicação. No entanto, a complexidade do problema de escalonamento em sistemas distribuídos juntamente com a dinamicidade inerente às grades computacionais tornam praticamente impossível um mapeamento ótimo das tarefas (Dong & Akl, 2006). Por esse motivo, os algoritmos de escalonamento geralmente visam, por meio de aproximações ou heurísticas, obter mapeamentos sub-ótimos.

Os algoritmos sub-ótimos são divididos em duas classes: aproximados e heurísticos. Em um algoritmo aproximado, reduz-se o espaço de busca do problema de escalonamento. Busca-se, portanto, um mapeamento de tarefas suficientemente bom em relação a uma estimativa previamente definida. Já um algoritmo heurístico utiliza regras empíricas para orientar a busca por um mapeamento de boa qualidade.

Quando implementados em uma abordagem dinâmica, os algoritmos de escalonamento podem ser empregados em um cenário centralizado, como também podem ser executados por múltiplos escalonadores distribuídos. A vantagem do cenário centralizado está na facilidade de implementação. No entanto, em função de problemas de escalabilidade, o escalonamento dinâmico centralizado não é indicado para grades de grande escala (Xhafa & Abraham, 2010). Além disso, em caso de uma falha do escalonador, todo o sistema é prejudicado (Dong & Akl, 2006; Xhafa & Abraham, 2010).

Para algoritmos de escalonamento dinâmicos e distribuídos, deve-se considerar mais um importante aspecto: a interação entre as diferentes escalonadores, que pode ser cooperativa ou independente (não-cooperativa). No modo não-cooperativo, o escalonamento é feito localmente, não se preocupando com o desempenho da grade como um todo. Por outro lado, no modo cooperativo, as decisões locais de escalonamento são tomadas de forma coordenada, objetivando uma meta global.

2.3.2 Algoritmos de Escalonamento para Tarefas Independentes

A relação de dependência entre as tarefas que compõem uma aplicação influencia diretamente na definição do algoritmo de escalonamento (Dong & Akl, 2006). A existência de dependência significa que as tarefas possuem uma ordem determinada de execução, consequentemente uma dada tarefa não pode ser executada enquanto suas tarefas predecessoras ainda não tiverem sido finalizadas. Tarefas independentes, por outro lado, podem ser executadas em qualquer ordem.

Aplicações compostas por tarefas independentes, denominadas de *bag-of-tasks*, facilitam o escalonamento, pois permitem a utilização de algoritmos que fazem pouco (ou nenhum) uso de informações sobre a infraestrutura da grade. Apesar de sua simplicidade intrínseca, problemas de várias áreas, tais como visão computacional (Fernández et al., 2008), mineração de dados (da Silva et al., 2004) e biologia computacional (Stiles et al., 1998), podem ser modelados como aplicações *bag-of-tasks*.

No contexto das grades computacionais, destacam-se os seguintes algoritmos de escalonamento para tarefas independentes.

Max-Min Algoritmo estático. Baseia-se na ideia de atribuir as maiores tarefas para os recursos mais rápidos (com maior potência computacional). Para cada um dos recursos, define-se a tarefa com o maior tempo de conclusão, sendo selecionada a tupla (recurso, tarefa) associada. O algoritmo Max-Min visa estabelecer uma sobreposição da execução das tarefas de longa duração com as curta duração (Fujimoto & Hagihara, 2004). Por exemplo, se há apenas uma tarefa de longa duração, ela será executada paralelamente a um conjunto de tarefas de curta duração (Fujimoto & Hagihara, 2004)

Min-Min Algoritmo estático. As menores tarefas são atribuídas aos recursos como maior potência de processamento. O comportamento comum desse algoritmo é executar paralelamente tarefas de pequena duração deixando por último as de longa duração. Comparativamente, apresenta um resultado inferior ao do Max-Min (Fujimoto & Hagihara, 2004).

Sufferage Algoritmo estático. Utiliza informações sobre o desempenho dos recursos para realizar o escalonamento. A ideia do *Sufferage* é determinar o quanto uma tarefa seria prejudicada caso não fosse atribuída ao recurso que a executaria de maneira mais eficiente. O valor de prejuízo é definido como a diferença entre os dois melhores tempos de execução previstos para a tarefa, considerando-se todos os recursos disponíveis.

XSufferage Baseado no *Sufferage*, é um algoritmo de escalonamento que utiliza informações sobre o desempenho dos recursos, bem como da rede que os interliga. Nesse sentindo, o desempenho do algoritmo está diretamente relacionado ao custo das transferências de dados necessárias para a execução das aplicações.

***Workqueue* (WQ)** Algoritmo dinâmico. Não necessita de informações sobre as tarefas ou recursos. O escalonamento é realizado aleatoriamente, alocando-se uma tarefa qualquer a um recurso disponível. Após a conclusão de uma tarefa, o recurso envia os resultados e o escalonador atribui uma nova tarefa para o recurso. A ideia por trás do WQ é que mais tarefas serão atribuídas aos recursos mais rápidos, enquanto os mais lentos irão processar uma carga de trabalho menor (Da Silva et al., 2003).

***Workqueue with Replication* (WQR)** Algoritmo dinâmico. O comportamento inicial é semelhante ao do *Workqueue*. A diferença começa quando todas as tarefas já foram alocadas. Nesse momento, tarefas em execução são escolhidas para serem replicadas em recursos ociosos. Ao fim da tarefa original, suas réplicas também são finalizadas. Se uma das réplicas terminar antes da tarefa original, são finalizadas as outras réplicas juntamente com a tarefa original. O algoritmo WQR apresenta um bom desempenho quando há recursos suficientes para a replicação das tarefas (Da Silva et al., 2003).

***Dynamic Fastest Processor to Largest Task First* (DFPLTF)** Algoritmo dinâmico. A partir de um conjunto de tarefas que ainda não foram escalonadas, a maior tarefa recebe prioridade, sendo atribuída ao recurso que puder executá-la mais rapidamente. O processo é repetido até que todos os recursos da grade estejam alocados ou não haja mais tarefas. Em seguida, a execução da aplicação é iniciada. Quando uma tarefa termina, todas as tarefas que não estão em execução são re-escalonadas até que todas os recursos estejam alocados novamente ou não haja mais tarefas. O processo continua até que todas as tarefas sejam concluídas. O algoritmo DFPLTF visa minimizar os efeitos associados à dinamicidade da grade (Da Silva et al., 2003).

2.4 Trabalhos Relacionados

Recentemente, a redução do consumo de energia em sistemas computacionais se tornou um importante tópico de pesquisa. Em particular, na área de sistemas distribuídos, diversos trabalhos que propõem estratégias de escalonamento cientes do consumo de energia foram publicados nos últimos anos (Huang et al., 2006; Kim et al., 2007; Li & Li, 2010; Rodriguez et al., 2010; Ma et al., 2012; Li & Li, 2012; Rodriguez et al., 2012b). Essas estratégias visam minimizar o consumo de energia sem que o desempenho das aplicações seja comprometido.

Em Kim et al. (2007), são apresentados algoritmos de escalonamento cientes do consumo de energia para aplicações *bag-of-tasks* em *clusters* homogêneos com suporte a DVS (*Dynamic Voltage Scaling*). DVS é uma técnica que permite ajustar dinamicamente as voltagens de processadores, com o objetivo de reduzir o consumo de energia. O modelo de escalonamento adotado para os *clusters* considera restrições de *deadline* das aplicações, sendo aplicado um regime de admissão. Baseados na heurística EDF (*Earliest Deadline First*), os algoritmos propostos ajustam as voltagens dos processadores de forma a minimizar o consumo total de energia e garantir o cumprimento dos *deadlines* das aplicações. Nos experimentos executados, observou-se que os algoritmos propostos proporcionaram uma redução no consumo de energia às custas de uma degradação na taxa de aceitação das aplicações.

Ma et al. (2012) propuseram o algoritmo EES (*Energy Efficient Scheduling*) para o escalon-

amento de tarefas independentes em sistemas distribuídos heterogêneos de alto desempenho. O algoritmo foi projetado a partir de uma solução heurística para o problema de escalonamento ciente de consumo de energia com restrições de *deadline*, que foi formulado como um modelo de programação linear inteira. O escalonamento das tarefas é realizado de forma a garantir que a maior quantidade possível de carga de trabalho seja atribuída aos recursos mais eficientes do ponto de vista energético. Em comparação a um referencial ótimo, determinado pela solução da relaxação linear do problema de escalonamento considerado, o algoritmo EES apresentou bom desempenho em termos de minimização de energia e satisfação de *deadline*.

No contexto de uma grade móvel, Huang et al. (2006) propuseram um escalonamento hierárquico de dois níveis, cujo objetivo é utilizar de maneira eficiente a energia dos dispositivos móveis. O primeiro nível emprega o algoritmo FIFS (*First Input First Service*) para escalonar as tarefas em nós fixos, que podem atuar como *proxies* entre a grade e um domínio sem fio. No segundo nível, um algoritmo baseado em uma heurística Min-Min é utilizado para realizar o escalonamento em cada um dos domínios sem fio. Na heurística, a seleção dos elementos móveis é conduzida de forma a minimizar a energia consumida durante a execução das tarefas.

Em Li & Li (2012), visando equilibrar o consumo de energia em uma grade móvel, os autores propuseram um algoritmo de escalonamento colaborativo baseado em uma estratégia orientada ao mercado (*market-oriented*). Um modelo para determinação de preços foi adotado, sendo a energia tratada como um recurso que pode ser negociado entre os dispositivos móveis. O escalonamento das tarefas é realizado de maneira a maximizar a utilidade da grade móvel (definida em função das negociações realizadas), não excedendo o *deadline* das aplicações e a capacidade de bateria de cada dispositivo. Uma estratégia semelhante também é empregada em Li & Li (2010). Deve-se ressaltar, no entanto, que estratégias orientadas ao mercado são baseadas na lei da oferta e procura, sendo normalmente um processo iterativo e lento (Ma et al., 2012), que pode ser inviável em alguns cenários de grade móvel.

Rodriguez et al. (2010) desenvolveram o SEAS (*Simple Energy-Aware Scheduler*), um escalonador para grades móveis com ênfase para tarefas que exigem uso intensivo do processador. A política de escalonamento adotada pelo SEAS se baseia em três aspectos dos dispositivos móveis: potência de processamento, carga de bateria e taxa de consumo de bateria. A partir de uma estimativa do tempo de vida de bateria, o escalonador determina periodicamente o número máximo de tarefas que pode ser alocado em cada dispositivo, métrica utilizada para determinar o mapeamento das tarefas (preferência é dada ao dispositivo que suporta o maior número de tarefas). Como a taxa de consumo pode variar em função de diferentes fatores (por exemplo, intensidade de uso da rede), os autores expandiram a proposta original, adicionando mecanismos de balanceamento de cargas baseados em técnicas de *job stealing* (Rodriguez et al., 2012b).

2.5 Considerações Finais

Neste capítulo, foram abordados conceitos importantes para o projeto descrito nesta dissertação. Maior ênfase foi dada para caracterização das grades móveis, bem como para a discussão de estratégias de escalonamento que visam minimizar o consumo de energia em sistemas distribuídos. No próximo capítulo, será apresentada a proposição de dois algoritmos heurísticos para o problema de escalonamento ciente de consumo de energia em grades móveis.

Capítulo

3

Escalonamento Ciente do Consumo de Energia em Grades Móveis

Como discutido no capítulo anterior, o gerenciamento do consumo de energia dos dispositivos participantes é um ponto crítico para o sucesso de uma grade móvel. Mais precisamente, uma grade móvel deve ser capaz de gerenciar o consumo de energia dos recursos ofertados, sem que o desempenho do sistema seja comprometido. É essencial, portanto, que a política de escalonamento adotada consiga estabelecer um compromisso entre o consumo de energia e os requisitos das aplicações submetidas.

Nesse sentindo, este trabalho de mestrado propõem dois algoritmos de escalonamento que enfatizam a questão energética no contexto de uma grade móvel. O objetivo é, dentro de um intervalo de tempo razoável, otimizar o consumo de energia, assegurando que as aplicações submetidas cumpram os *deadlines* definidos pelos usuários. De maneira similar ao trabalho de Ma et al. (2012), os algoritmos propostos são baseados em soluções heurísticas para o problema de escalonamento ciente de consumo de energia. No entanto, deve ser ressaltado que, no contexto de uma grade móvel, tal problema, além das restrições de *deadline* das aplicações, também envolve restrições relativas à capacidade de bateria dos dispositivos.

Para o projeto dos algoritmos, foram consideradas três etapas. Primeiro, levando-se em consideração aspectos como arquitetura, tipo de aplicação e caracterização dos recursos, criou-se um modelo de grade móvel. Em seguida, a partir desse modelo, o escalonamento ciente de consumo de energia foi formulado como um problema de otimização com variáveis binárias. Por fim, tendo-se em vista particularidades do problema, os algoritmos foram projetados com base em soluções heurísticas. Nas próximas seções, cada uma dessas etapas é discutida em detalhes.

3.1 Modelo de Grade Móvel

O modelo de grade móvel adotado neste trabalho considera a arquitetura *on-site* ilustrada na Figura 3.1. Nesta arquitetura, os recursos, caracterizados por dispositivos móveis localizados em uma área bem definida (um *hotspot* de uma WLAN, por exemplo), são coordenados por um servidor *proxy* associado a um ponto de acesso (PA). Esse servidor *proxy* atua como um *broker*, sendo responsável por coletar informações sobre os recursos (potência de processamento, a estimativa de consumo de energia e a capacidade de bateria disponível), bem como por escalonar as tarefas que compõem as aplicações submetidas pelos usuários da grade móvel.

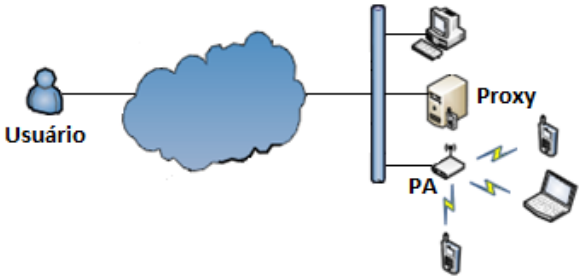


Figura 3.1: Arquitetura de grade móvel considerada na modelagem do problema de escalonamento.

As aplicações submetidas para execução na grade móvel são do tipo *bag-of-tasks*, sendo, portanto, compostas por tarefas independentes que podem ser escalonadas de maneira arbitrária. Além disso, cada aplicação está associada a um prazo máximo de execução (*deadline*), que funciona como um requisito de qualidade de serviço. O escalonamento é não-preemptivo, sendo as tarefas executadas sem interrupção, respeitando a ordem com que foram mapeadas aos recursos.

Como a capacidade de bateria dos dispositivos móveis influencia diretamente na decisão dos proprietários de compartilhar ou não seus recursos computacionais (Furthmüller & Waldhorst, 2012), a participação na grade móvel é incentivada por meio de compensações diretamente proporcionais a energia consumida pela execução das aplicações submetidas. Nesse sentindo, o custo para manutenção da grade móvel é caracterizado em função do consumo energético dos dispositivos móveis participantes. É importante, logo, que a grade móvel seja capaz de gerenciar o consumo de energia de maneira adequada, reduzindo os custos associados. Também é fundamental assegurar que o desempenho do sistema não seja comprometido.

3.2 Formulação do Problema

Considerando-se o modelo descrito, o problema de escalonamento ciente do consumo de energia em grades móveis é definido da seguinte forma: dado um conjunto de m

recursos e uma aplicação *bag-of-tasks* composta por n tarefas, atribua cada uma das tarefas a exatamente um recurso, de maneira a minimizar o consumo de energia, bem como satisfazer as restrições relativas ao *deadline* e à capacidade de bateria dos recursos. Mais especificamente, esta definição é uma variação do problema generalizado de atribuição (GAP, do inglês *Generalized Assignment Problem*).

Logo, de acordo com a notação apresentada na Tabela 3.1, o problema de escalonamento pode ser formulado pelo modelo de programação linear inteira definido por (3.1)–(3.4), no qual x_{ij} é uma variável binária que recebe 1 apenas quando j-ésima tarefa da aplicação é atribuída ao i-ésmio recurso. O tempo de execução da tarefa j no recurso i é dado por $t_{ij} = w_j/q_i$. Já consumo de energia em função da execução da tarefa j no recurso i é dado por $e_{ij} = t_{ij} \times p_i$.

A função objetivo (3.1) visa minimizar o consumo total de energia necessário para a execução das tarefas que compõem a aplicação. As restrições definidas em (3.2) asseguram que o *deadline* da aplicação e a capacidade de bateria dos recursos não sejam excedidos. As restrições de atribuição (3.3) garantem que cada tarefa seja atribuída a exatamente um recurso. O domínio das variáveis é definido em (3.4). Deve-se ressaltar que, devido às restrições de atribuição (3.3), uma instância do GAP não necessariamente possui uma solução factível (Martello & Toth, 1990).

Tabela 3.1: Notação utilizada na formulação do problema de escalonamento ciente do consumo de energia.

Parâmetro	Descrição
q_i	Potência de processamento do recurso i (MIPS, Milhões de Instruções por Segundo)
p_i	Estimativa do consumo de energia do recurso i (Watt)
c_i	Capacidade de bateria do recurso i (Joule)
w_j	Carga de trabalho da j-ésima tarefa da aplicação (MI, Milhões de Instruções)
d	*Deadline* da aplicação (segundos)
t_{ij}	Tempo de execução da tarefa j no recurso i (segundos)
e_{ij}	Consumo de energia relativo à execução da tarefa j no recursos i (Joule)

$$\text{Minimizar} \quad E = \sum_{i=1}^{m} \sum_{j=1}^{n} e_{ij} x_{ij} \tag{3.1}$$

$$\text{Sujeito a} \quad \sum_{j=1}^{n} t_{ij} x_{ij} \leq \min\{d, c_i/p_i\}, \ \forall i \tag{3.2}$$

$$\sum_{i=1}^{m} x_{ij} = 1, \ \forall j \tag{3.3}$$

$$x_{ij} \in \{0,1\}, \ \forall i,j \tag{3.4}$$

3.3 Algoritmos Heurísticos

Como o GAP é $\mathcal{NP}$-difícil no sentido forte (*strong sense*) (Martello & Toth, 1990), solucionar de maneira ótima o problema de programação linear inteira apresentado é um processo computacionalmente caro, inviável no contexto de uma grade móvel de grande dimensão, uma vez que o tempo necessário para encontrar uma solução ótima pode ser sensivelmente superior à escala de tempo de execução das aplicações submetidas pelos usuários. Logo, foram propostos dois algoritmos heurísticos para solucionar tal problema. O primeiro algoritmo propõe uma adaptação da estratégia de *maximum regret*, proposta por Martello & Toth (1981). Já o segundo consiste em um algoritmo guloso (*greedy*), que privilegia os recursos mais energeticamente eficientes.

3.3.1 *Maximum regret*

A ideia do algoritmo *Maximum Regret* (Algoritmo 1) é determinar o quanto uma tarefa seria prejudicada caso não fosse atribuída ao recurso mais "desejável". Definida com base no consumo de energia, seja $f_{ij} = -e_{ij}$ uma função que indica o "desejo" de se atribuir a tarefa j ao recurso i, ou seja, quanto maior o consumo de energia, menor o desejo de atribuição da tarefa. Para cada tarefa j ainda não atribuída e para os recursos ainda disponíveis, calcule o valor de *regret* (r_j), que é dado pela diferença entre o maior e o segundo maior valor de f_{ij} (Linha 15). A tarefa escolhida para atribuição ($j^\star$) é dada pelo maior valor de r_j.

Em relação ao número de iterações necessárias para a obtenção de uma solução factível, o laço interior (Linha 10) é executado $O(n^2)$ vezes, cada umas delas exigindo $O(m)$ operações para definir F_j e calcular r_j (linhas 11–17). Pode-se afirmar, portanto, que a complexidade do algoritmo *maximum regret* é $O(mn^2)$.

3.3.2 *Greedy*

O algoritmo *Greedy* (Algoritmo 2) privilegia os recursos mais energeticamente eficientes, sendo caracterizado por duas etapas principais: seleção dos recursos (Linha 3); e determinação da carga de trabalho que pode ser alocada em tais recursos sem que as restrições de *deadline* e capacidade de bateria sejam violadas (Linha 4).

Considerando-se os recursos disponíveis, seleciona-se o recurso $i^\star$ com menor valor para a razão p_i/q_i (custo em Joule por MI). Em seguida, deve-se determinar a maior carga de trabalho possível que pode ser alocada no recurso, processo que corresponde a solucionar o problema da soma do subconjunto (SSP, do inglês *Subset-Sum Problem*), definido por (3.5)–(3.7), onde uma x_j é variável binária que recebe 1 apenas se a tarefa j é atribuída ao recurso $i^\star$. O SSP é um caso particular do problema da mochila (*knapsack problem*) no qual, em relação a um determinado valor alvo, espera-se a obter uma solução que minimize o desvio negativo (Martello & Toth, 1990).

Algoritmo 1: *Maximum Regret*

Entrada: Um conjunto de recursos $R = \{R_1 \dots R_m\}$ sendo $R_i = (q_i, p_i, c_i)$; Uma aplicação $A = (d, W = \{w_1 \dots w_n\})$

para cada *recurso i* **faça**
  $d_i \leftarrow d$
fim para cada
para cada *tarefa j e recurso i* **faça**
  $t_{ij} \leftarrow w_j / q_i$
  $e_{ij} \leftarrow t_{ij} p_i$
  $f_{ij} \leftarrow -e_{ij}$
fim para cada
repita
  para cada *tarefa j não atribuída* **faça**
    $F_j \leftarrow \{i : t_{ij} \leq \min\{d_i, c_i / p_i\}\}$
    se $F_j = \emptyset$ **então** $r_j \leftarrow -\infty$
    senão
      se $|F_j| = 1$ **então** $r_j \leftarrow \infty$ **senão**
        $r_j = \max\{f_{ij} : i \in F_j\} - \max_2\{f_{ij} : i \in F_j\}$
      fim se
    fim se
  fim para cada
  $j^\star \leftarrow \arg\max\{r_j\}$
  $i^\star \leftarrow \arg\max_i\{f_{ij^\star} : i \in F_{j^\star}\}$
  se $r_{j^\star} \neq -\infty$ **então**
    atribua a tarefa $j^\star$ ao recurso $i^\star$
    $d_{i^\star} \leftarrow d_{i^\star} - t_{i^\star j^\star}$
    $c_{i^\star} \leftarrow c_{i^\star} - e_{i^\star j^\star}$
  fim se
até *que todas as tarefas tenham sido atribuídas* ***Ou*** *mais nenhuma tarefa possa ser alocada nos recursos disponíveis*;

Algoritmo 2: *Greedy*

Entrada: Um conjunto de recursos $R = \{R_1 \dots R_m\}$ sendo $R_i = (q_i, p_i, c_i)$; Uma aplicação $A = (d, W = \{w_1 \dots w_n\})$

1 ordenar W em ordem decrescente
2 **repita**
3 $\quad i^\star \leftarrow \underset{i:R_i \in R}{\arg\min}\{p_i/q_i\}$
4 $\quad g \leftarrow \text{MTGS}(W, \min\{d, c_{i^\star}/p_{i^\star}\}, q_{i^\star})$
5 $\quad$ **para cada** $j \in g$ **faça**
6 $\quad\quad$ atribua a tarefa j ao recurso $i^\star$
7 $\quad$ **fim para cada**
8 $\quad R \leftarrow R \backslash \{R_{i^\star}\}$
9 $\quad W \leftarrow W \backslash \{w_j : j \in g\}$
10 **até** *que todas as tarefas tenham sido atribuídas* ***Ou*** *mais nenhuma tarefa possa ser alocada nos recursos disponíveis*;

$$\text{Maximizar} \quad E = \sum_{j=1}^{n} w_j x_j \tag{3.5}$$

$$\text{Sujeito a} \quad \sum_{j=1}^{n} w_j x_j \leq \min\{d, c_{i^\star}/p_{i^\star}\} \times q_{i^\star} \tag{3.6}$$

$$x_j \in \{0, 1\}, \ \forall j \tag{3.7}$$

Assim como o GAP, o problema da soma do subconjunto pertence à classe $\mathcal{NP}$-difícil (Martello & Toth, 1990). Dessa forma, para obter soluções factíveis suficientemente próximas à solução ótima em um intervalo de tempo razoável, foi escolhido o algoritmo aproximativo MTGS (Algoritmo 3), proposto por Martello & Toth (1984), que possui complexidade $O(n^2)$ e razão de desempenho no pior caso $r = \frac{3}{4}$.

A ideia por trás do algoritmo MTGS é avaliar soluções obtidas para diferentes subconjuntos de tarefas. Mais precisamente, a partir de um conjunto inicial de tarefas $\{1, \dots, n\}$, são definidos n subconjuntos: $\{1, \dots, n\}$, $\{2, \dots, n\}$, $\{3, \dots, n\}$ e assim por diante. Para cada um desses subconjuntos, uma abordagem gulosa de atribuição de tarefas é aplicada (linhas 6–11) , sendo escolhida a solução que proporciona a maior alocação de carga de trabalho no recurso. A saída do algoritmo é um subconjunto g contendo as tarefas que devem ser alocadas no recurso considerado.

3.4 Considerações Finais

Neste capítulo, foram apresentados dois algoritmos de escalonamento ciente do consumo de energia para grades móveis. Para o desenvolvimento desses algoritmos, o problema de escalonamento em grades móveis foi formulado como um problema de otimização,

Algoritmo 3: MTGS

Entrada: $W, d, q_{i^\star}$
Saída: g

```
1  z ← 0;
2  g ← ∅
3  para cada j tal que w_j ∈ W faça
4  |  d̄ ← d
5  |  ḡ ← ∅
6  |  para cada k tal que w_k ∈ W E k ≥ j faça
7  |  |  se w_k/q_{i*} ≤ d̄ então
8  |  |  |  d̄ ← d̄ − (w_k/q_{i*})
9  |  |  |  ḡ ← ḡ ∪ {k}
10 |  |  fim se
11 |  fim para cada
12 |  se d − d̄ > z então
13 |  |  z ← d − d̄
14 |  |  g ← ḡ
15 |  fim se
16 fim para cada
```

cujo objetivo é minimizar o consumo total de energia necessário para a execução de aplicações *bag-of-tasks*, de maneira satisfazer as restrições relativas ao *deadline* e à capacidade de bateria dos recursos.

Como tal problema pertence à classe $\mathcal{NP}$-difícil, obter soluções ótimas é um processo computacionalmente caro, inviável no contexto de uma grade móvel de grande escala. Logo, adotou-se uma abordagem heurística para o projeto dos algoritmos de escalonamento, visando a obtenção de soluções de boa qualidade em um intervalo compatível com a escala de tempo de execução das aplicações. No próximo capítulo, serão apresentados e discutidos os experimentos realizados para avaliar o desempenho dos algoritmos heurísticos propostos.

Capítulo

4

Avaliação de Desempenho

Neste capítulo, são detalhados os experimentos que foram executados para avaliar os algoritmos propostos. A avaliação de desempenho foi composta por dois conjuntos distintos de experimentos. No primeiro conjunto, considerando-se um cenário estático, buscou-se principalmente determinar a acurácia dos algoritmos, isto é, a capacidade de encontrar soluções factíveis próximas o suficiente de um referencial determinado. No segundo conjunto, o objetivo foi avaliar o comportamento dos algoritmos em um ambiente dinâmico, onde podem ocorrer variações tanto na taxa de submissão das aplicações quanto no nível de bateria dos dispositivos móveis participantes.

4.1 Conjunto de Experimentos I

A acurácia dos algoritmos foi avaliada por meio de experimentos executados no ambiente de programação numérica **R**[1]. Foram utilizadas como referenciais comparativos, as soluções para o problema de escalonamento obtidas por meio do *solver* **Gurobi Optimizer**[2] (versão 5.5, licença acadêmica). O Gurobi utiliza uma variação do algoritmo *branch and bound* para tentar encontrar soluções ótimas para problemas de otimização. Além da acurácia, o tempo de resposta para a obtenção de soluções factíveis também foi avaliado.

4.1.1 Planejamento de Experimentos

O planejamento de experimentos é uma etapa fundamental na avaliação de desempenho de qualquer sistema computacional. Nela, definem-se as características que serão avaliadas, examinando-se quais delas podem influenciar no desempenho do sistema (Jain, 1991). O objetivo do planejamento de experimentos é manipular de forma planejada cer-

[1] http://www.r-project.org/
[2] http://www.gurobi.com/

tas variáveis independentes (fatores), definindo-se os valores mais prováveis que essas variáveis podem assumir (níveis), com a finalidade de verificar o efeito que esta manipulação provoca na variável de resposta (variável dependente).

Neste primeiro conjunto de experimentos, utilizou-se o planejamento fatorial completo, que contabiliza todas as combinações considerando todos os fatores e níveis. Particularmente, os experimentos tiveram como base quatro fatores, sendo um deles com três níveis, e os demais com dois (Tabela 4.1): solução para o problema de escalonamento, quantidade de recursos, quantidade de tarefas e o *deadline* da aplicação. Como o objetivo dos experimentos é determinar a capacidade dos algoritmos propostos em obter boas soluções dentro de um intervalo de tempo razoável, foram consideradas como variáveis de resposta duas métricas: o consumo de total energia e o tempo de resposta.

Tabela 4.1: Fatores e níveis considerados no planejamento fatorial completo.

Fatores	Níveis
Solução	*Maximum Regret*, *Greedy* e Gurobi
Recursos	30 e 45
Tarefas	50 e 70
Deadline	curto e estendido

As instâncias do problema de escalonamento ciente de consumo de energia foram derivadas a partir de cinco configurações de dispositivos móveis, distribuídas de maneira uniforme de acordo com o número de recursos disponíveis. A potência de processamento (q_i) e o consumo de energia (p_i) para cada uma dessas configurações (Tabela 4.2) foram estimados com base no trabalho de Carroll & Heiser (2010). Com relação à carga de trabalho (w_j), os valores adotados dependem da granularidade da aplicação, determinada pela quantidade de tarefas (Tabela 4.3). Nesse sentindo, as respectivas distribuições foram estabelecidas de maneira a garantir que o tamanho esperado de uma aplicação seja de $2,5 \times 10^5$ MI.

O tamanho do *deadline* de uma aplicação é definido a partir da carga de trabalho das tarefas e da potência de processamento dos recursos disponíveis: $3 \times mean(w_j)/mean(q_i)$ para o *deadline* curto, e $4 \times mean(w_j)/mean(q_i)$ para o estendido. O consumo de energia referente à execução das tarefas possui um erro de até 1%, determinado com base em uma distribuição uniforme. Por fim, assumiu-se também que a energia necessária para executar uma aplicação é insignificante em relação a capacidade de bateria dos dispositivos.

Em relação ao *solver* Gurobi, os seguintes parâmetros foram modificados em relação à configuração padrão: TimeLimit = 60 segundos e MIPGap = 1×10^{-4}. O parâmetro TimeLimit limita o tempo total gasto na busca por soluções factíveis. Já o parâmetro MIPGap determina que o processo de busca será encerrado quando a diferença relativa entre os limitantes inferior e superior da função objetivo for menor que MIPGap vezes o limitante superior.

Tabela 4.2: Configuração dos dispositivos móveis considerados nos experimentos. Valores foram derivados das estimativas apresentadas por Carroll & Heiser (2010).

Dispositivo	q_i (MIPS)	p_i (mW)
1	110	60
2	295	435
3	440	180
4	460	500
5	539	234

Tabela 4.3: Carga de trabalho das tarefas que compõem uma aplicação. A distribuição uniforme considerada depende da granularidade da aplicação.

Número de Tarefas	w_i (MI)
50	[4000;6000]
70	[2900;4241]

Para cada experimento, foram realizadas 30 replicações[3], utilizadas para determinar a média, desvio padrão e intervalo de confiança de 95% de acordo com a distribuição *T-student*. A quantidade de replicações foi adotada para que fosse possível obter intervalos de confiança suficientemente pequenos.

4.1.2 Análise dos Resultados

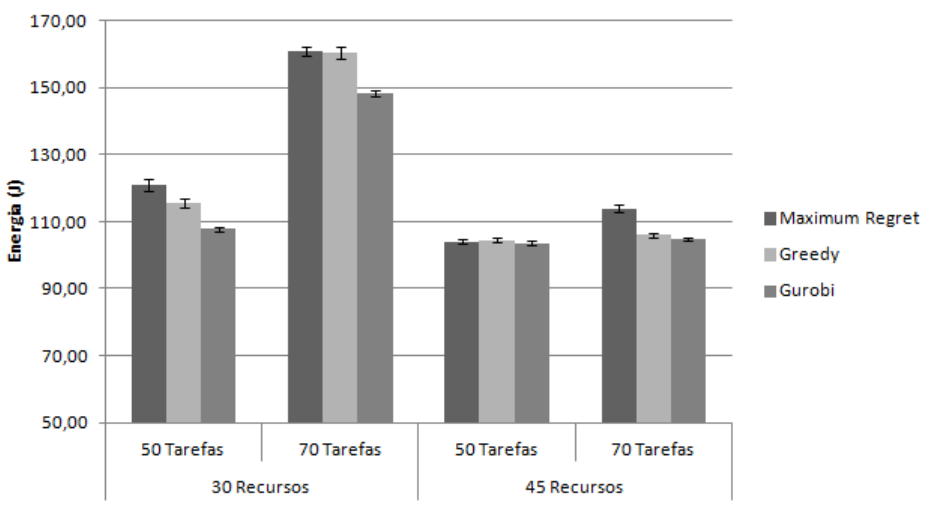
Na Figura 4.1 e na Tabela 4.4, são apresentados os resultados relativos à métrica consumo de energia. Considerando-se o cenário com *deadline* curto (Figura 4.1(a)), observa-se que as soluções obtidas pelos algoritmos propostos foram numericamente superiores que as do *solver* Gurobi, ou seja, foram inferiores em termos de otimizar o consumo de energia. Particularmente, para o algoritmo *Maximum Regret*, a maior diferença no consumo de energia em relação aos resultados do Gurobi (30 recursos, 50 tarefas) foi de 12,18%. Já para algoritmo *Greedy* (30 recursos, 70 tarefas), tal diferença foi de 8,14%. Deve-se ressaltar, no entanto, que o aumento do número de recursos disponíveis de 30 para 45 fez com que essa diferença de desempenho fosse praticamente marginal em alguns casos (*Maximum Regret* com 50 tarefas; *Greedy* com 50 e 70 tarefas). Isso ocorre, em grande parte, pois é possível escalonar um número maior de tarefas nos dispositivos energeticamente mais eficientes, sem que haja sobrecarga dos mesmos.

Outro aspecto importante está relacionado ao *deadline* da aplicação. Nesse sentido, verifica-se que, independentemente dos demais fatores, a adoção de um *deadline* mais mais relaxado (Figura 4.1(b)) implica em uma redução significativa do consumo de energia. Os algoritmos propostos, com exceção de um único experimento (30 recursos, 70 tarefas, *Maximum Regret*), tiveram um desempenho próximo ao do Gurobi. Especifica-

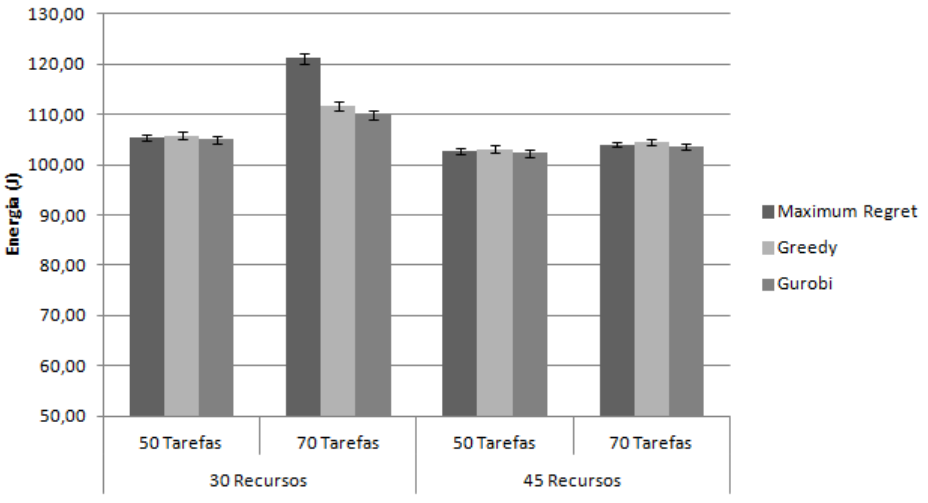
[3]Neste caso, cada replicação representa uma instância do problema de escalonamento ciente do consumo de energia

mente, para o algoritmo *Greedy*, a maior diferença no consumo de energia foi de apenas 1,60% (30 recursos, 70 tarefas).

Comparando-se os dois algoritmos propostos, apesar de o desempenho em termos de consumo de energia ser muito próximo, nota-se que o *Maximum Regret* apresenta uma maior sensibilidade em relação ao aumento no número de tarefas. Nesse sentindo, podem ser destacados os experimentos com 45 recursos, 70 tarefas e *deadline* curto, e com 30 recursos, 70 tarefas e *deadline* estendido, nos quais os resultados do *Maximum Regret* em relação ao *Greedy* foram, respectivamente, 7,42% e 8,56% piores.



(a) *deadline* curto



(b) *deadline* estendido

Figura 4.1: Consumo de energia para os experimentos planejados. Para cada experimento, é apresentado o valor médio das replicações e o respectivo intervalo de confiança. Os resultados foram organizados de acordo com o *deadline*.

Em relação à métrica tempo de resposta, os resultados obtidos nos experimentos planejados são apresentados na Tabela 4.5. Para os algoritmos propostos, observa-se que o tempo necessário para encontrar soluções factíveis é diretamente proporcional ao tamanho

Tabela 4.4: Consumo de energia para os experimentos planejados (joules). Para cada experimento, é apresentado o valor médio das replicações juntamente com o respectivo intervalo de confiança. Os dados foram sumarizados de acordo com o *deadline*.

Solução	30 Recursos		45 Recursos	
	50 Tarefas	70 Tarefas	50 Tarefas	70 Tarefas
deadline curto				
Maximum Regret	$120,96 \pm 1,64$	$160,99 \pm 1,28$	$104,03 \pm 0,69$	$114,00 \pm 1,16$
Greedy	$115,54 \pm 1,23$	$160,54 \pm 1,97$	$104,46 \pm 0,71$	$106,12 \pm 0,68$
Gurobi	$107,82 \pm 0,70$	$148,46 \pm 0,84$	$103,62 \pm 0,70$	$104,67 \pm 0,48$
deadline estendido				
Maximum Regret	$105,35 \pm 0,62$	$121,12 \pm 1,04$	$102,68 \pm 0,69$	$103,96 \pm 0,46$
Greedy	$105,81 \pm 0,64$	$111,57 \pm 0,87$	$103,14 \pm 0,70$	$104,49 \pm 0,47$
Gurobi	$104,96 \pm 0,62$	$109,81 \pm 0,83$	$102,26 \pm 0,69$	$103,59 \pm 0,46$

da instância do problema, refletindo, portanto, apenas a quantidade de variáveis envolvidas. O desempenho inferior do algoritmo *Maximum Regret* em relação ao *Greedy* pode ser explicado pela necessidade do cálculo do *regret* das tarefas a cada iteração, procedimento que exige $O(m)$ operações.

Tabela 4.5: Tempo de processamento (segundos) necessário para a obtenção de soluções factíveis. Para cada experimento, é apresentado o valor médio das replicações juntamente com o respectivo intervalo de confiança. Os dados foram sumarizados de acordo com o *deadline*.

Solução	30 Recursos		45 Recursos	
	50 Tarefas	70 Tarefas	50 Tarefas	70 Tarefas
deadline curto				
Maximum Regret	$0,05 \pm 0,00$	$0,10 \pm 0,00$	$0,06 \pm 0,00$	$0,11 \pm 0,00$
Greedy	$0,01 \pm 0,00$	$0,02 \pm 0,00$	$0,01 \pm 0,00$	$0,02 \pm 0,00$
Gurobi	$57,16 \pm 4,08$	$60,02 \pm 0,01$	$38,48 \pm 8,30$	$49,67 \pm 5,73$
deadline estendido				
Maximum Regret	$0,05 \pm 0,00$	$0,10 \pm 0,00$	$0,06 \pm 0,00$	$0,11 \pm 0,00$
Greedy	$0,01 \pm 0,00$	$0,02 \pm 0,00$	$0,01 \pm 0,00$	$0,02 \pm 0,00$
Gurobi	$24,63 \pm 9,15$	$60,04 \pm 0,01$	$52,70 \pm 5,79$	$53,87 \pm 5,60$

Por outro lado, para o *solver* Gurobi, a busca por uma solução ótima não dependeu apenas do número de variáveis, tendo sido também influenciada por características relativas à instância do problema, determinadas basicamente pela carga de trabalho das tarefas (elemento aleatório). Na maior parte das instâncias, as soluções providas pelo Gurobi atingiram o tempo limite estipulado, não satisfazendo, portanto, a condição de parada determinada pelo parâmetro MIPGap. No entanto, em alguns casos, a busca convergiu rapidamente para uma solução. Tal comportamento é ilustrado no gráfico de *boxplot* para instâncias do problema com *deadline* estendido (Figura 4.2).

Em gráfico de *boxplot*, a caixa é definida a partir do primeiro e terceiros quartis, sendo a barra interna definida pela mediana. A representação da amplitude interquartil (difer-

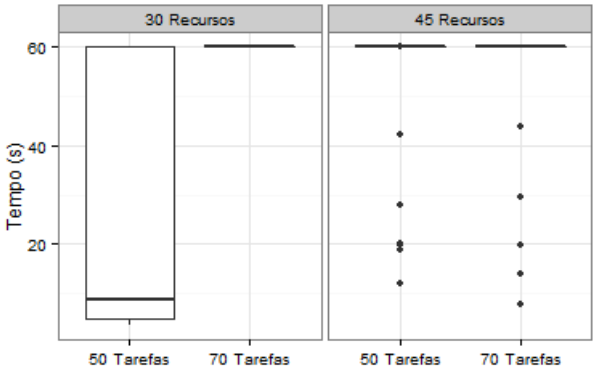


Figura 4.2: Tempo de resposta do *solver* Gurobi para instâncias do problema de escalonamento com *deadline* estendido.

ença entre o terceiro e primeiro quartis) indica o grau de dispersão dos dados, bem como auxilia na identificação de *outliers*. Especificamente paras as instâncias do problema com 45 recursos, nota-se uma concentração de valores em 60 segundos (tempo limite estipulado) e alguns poucos *outliers*, indicados por pontos.

Por fim, deve-se ressaltar que os algoritmos propostos, particularmente o *Greedy*, apresentaram tempos de processamento significativamente menores em relação aos obtidos pelo Gurobi, justificando, portanto, a utilização de heurísticas para a solução do problema de escalonamento ciente de consumo de energia.

4.1.3 Análise de Influência dos Fatores

A análise de influência dos fatores visa determinar quais fatores tiveram um impacto maior nos resultados dos experimentos, bem como também verificar se a interação entres esses fatores possui alguma significância. Mais especificamente, a análise apresentada nesta seção teve como objetivo principal avaliar se a diferença de desempenho entre os dois algoritmos propostos pode ser considerada significativa em termos de consumo de energia.

Tomando-se apenas os experimentos relativos aos algoritmos *Greedy* e *Maximum Regret*, observa-se um planejamento fatorial completo 2^k (k fatores com dois níveis cada). Particularmente, esse tipo de planejamento apresenta uma característica importante: a ortogonalidade, que garante que a influência de cada fator pode ser estimada de maneira independente, sem que sejam considerados outros fatores ou interações (Croarkin et al., 2010).

Dado que os níveis podem ser classificados arbitrariamente como baixos (-1) e altos (+1), considerando-se a ortogonalidade do planejamento fatorial completo 2^k, tem-se que a estimativa da influência de um fator ou interação possui a seguinte forma: $\bar{y}(+1) - \bar{y}(-1)$, onde $\bar{y}(+1)$ e $\bar{y}(-1)$ são as médias da variável de resposta para o fator ou interação quando são assumidos os níveis alto e baixo, respectivamente. Pelo teo-

rema central do limite, é possível assumir que a estimativa da influência de um fator ou interação segue uma distribuição normal (Croarkin et al., 2010). Além disso, como os estimadores possuem a mesma forma (diferença de médias), os desvios padrão, apesar de desconhecidos, têm o mesmo valor sob a hipótese de σ constante (Croarkin et al., 2010).

Quando a influência tende a uma distribuição normal com centro em zero, ela pode ser classificada como insignificante. Por outro lado, quando é observada uma distribuição normal centrada em um valor distante de zero, a respectiva influência pode ser considerada significativa. Assim, partindo-se da hipótese que todas as influências são próximas de zero, pode-se utilizar um gráfico de probabilidade normal (*normal probability plot*) para identificar possíveis fatores ou interações importantes (*outliers*).

No gráfico de probabilidade normal ilustrado na Figura 4.3, são apresentadas as influências dos fatores no consumo de energia para os experimentos relativos aos algoritmos *Maximum Regret* e *Greedy*. A reta centrada em zero é definida a partir da hipótese de que a influência dos fatores segue uma distribuição. Deve-se ressaltar que valores negativos de influência apenas refletem a classificação arbitrária atribuída aos níveis. Observa-se que os fatores que mais influenciam o consumo de energia são a quantidade de recursos (C), o *deadline* (D) da aplicação e a quantidade de tarefas (B). Já o fator algoritmo (A) possui uma influência pouco significativa, indicando que o desempenho dos algoritmos propostos é semelhante nas condições adotadas no planejamento de experimentos. Este comportamento também pode ser observado no gráfico da Figura 4.4, onde são apresentas as variações no consumo de energia de acordo com a mudança de nível para cada fator.

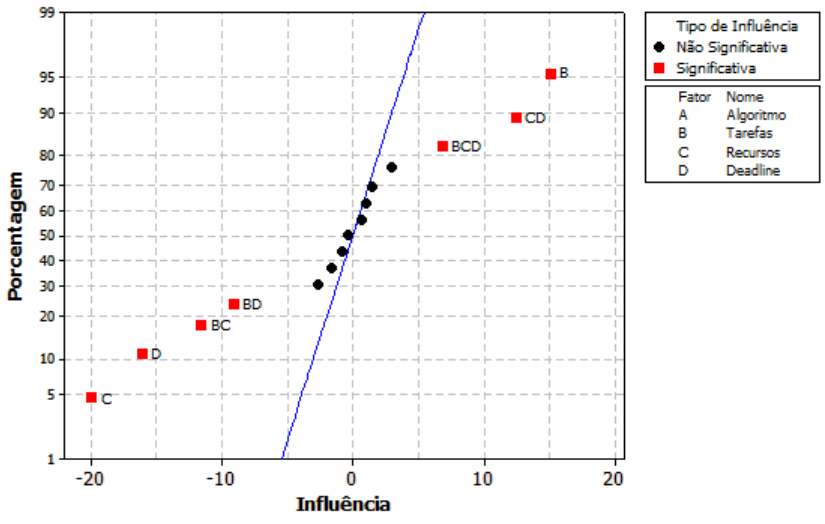


Figura 4.3: Gráfico de probabilidade normal para a influência dos fatores no consumo de energia. Apenas foram considerados os experimentos relativos aos algoritmos *Maximum regret* e *Greedy*

Com relação às interações entre os fatores, verifica-se que as interações CD, BC, BD e BCD são as que influenciam de maneira mais significativa o consumo de energia. A

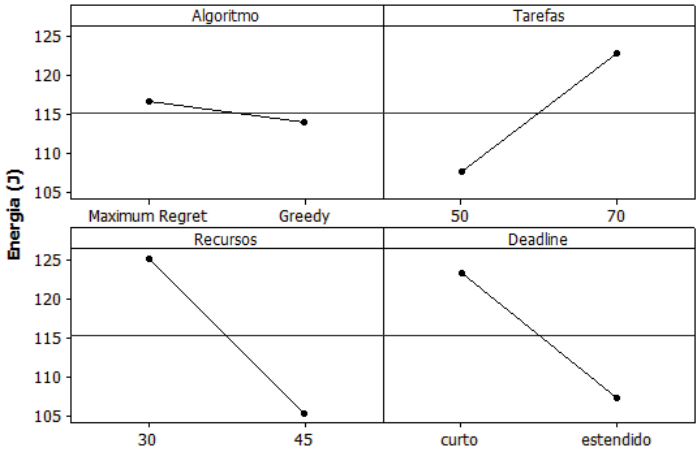


Figura 4.4: Variação no consumo de energia em função da mudança de nível em cada fator.

grande influência da interação CD evidencia que o problema de escalonamento ciente do consumo de energia é muito suscetível a variações na quantidade de recursos e no *deadline*. Por exemplo, em um cenário onde a quantidade de recursos é grande em relação ao número de tarefas e o *deadline* suficientemente longo, os algoritmos propostos tendem a encontrar soluções próximas das melhores soluções econtradas pelo Gurobi. Por outro lado, se o número de recursos for pequeno e o *deadline* apertado, as soluções podem divergir consideravelmente do referencial ótimo.

No gráfico da Figura 4.5, são apresentadas as interações dois a dois entre os fatores tarefas (B), recursos (C) e *deadline* (B). A conformação de curvas observadas (retas não paralelas) confirma a significância das interações CD, BC e BD. É possível notar que, quando são considerados apenas 30 recursos disponíveis, a variação do *deadline* é preponderante para a degradação do desempenho dos algoritmos propostos.

4.2 Conjunto de Experimentos II

Para avaliar o comportamento dos algoritmos *Maximum Regret* e *Greedy* no contexto de um ambiente dinâmico, foram realizadas simulações que consideraram aspectos como taxa de submissão das aplicações e tempo de vida de bateria dos dispositivos. Os experimentos foram desenvolvidos com base na ferramenta de simulação SimGrid[4] (Casanova et al., 2008), que possibilita a avaliação de sistemas distribuídos heterogêneos.

4.2.1 Planejamento de Experimentos

Neste conjunto de experimentos, o cenário considerado consiste de diferentes usuários submetendo aplicações para uma grade móvel, sendo o intervalo entre submissões uma variável aleatória que segue a distribuição de Poisson. Além disso, uma aplicação só é aceita para execução na grade móvel caso seja possível escaloná-la dentro do *deadline*

[4]http://simgrid.gforge.inria.fr/

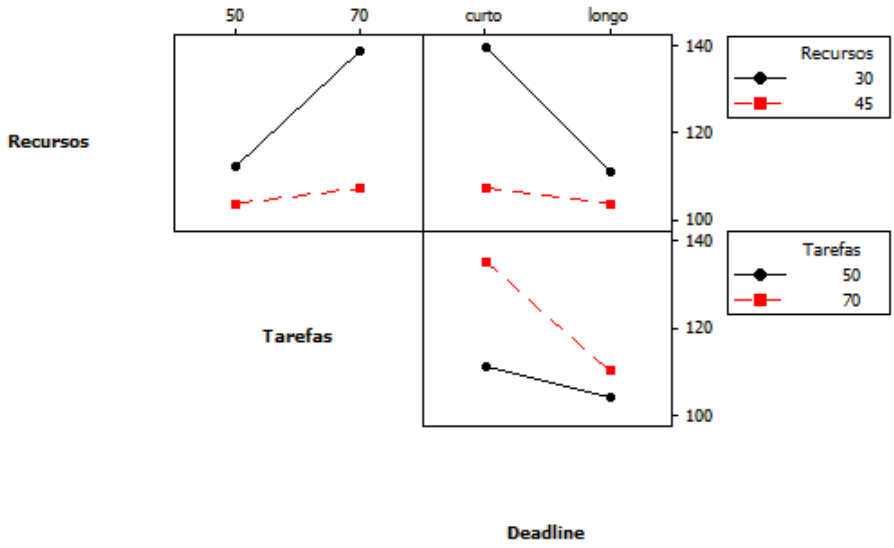


Figura 4.5: Interações dois a dois entres os fatores tarefas, recursos e *deadline*. Os valores do eixo vertical correspondem ao consumo de energia (J).

estipulado. Como variáveis de resposta, foram adotadas duas métricas: taxa de aceitação das aplicações e o consumo médio de energia por aplicação aceita.

Com base nos resultados apresentados na seção anterior, a configuração dos experimentos foi estabelecida de acordo com os piores casos observados para os algoritmos propostos: instâncias do problema de escalonamento ciente do consumo de energia com 30 recursos e 70 tarefas. A caracterização e distribuição dos tipos de recursos, assim como a carga de trabalho esperada para uma aplicação, possuem os mesmos valores estabelecidos no primeiro conjunto de experimentos. Já a capacidade de bateria de cada recurso segue uma distribuição uniforme com mínimo de 1000 J e máximo de 5000 J. Deve-se ressaltar que um dispositivo deixa de fazer parte da grade móvel quando a sua capacidade de bateria é excedida.

Adotou-se o planejamento fatorial completo abrangendo três fatores com dois níveis cada (Tabela 4.6): algoritmo de escalonamento, intervalo médio entre submissões e o *deadline* das aplicações submetidas. Cada experimento corresponde à simulação da submissão de 100 aplicações para uma grade móvel. Para estimar os valores das variáveis de resposta e seus respectivos intervalos de confiança, os experimentos foram replicados 30 vezes.

O ambiente para a execução dos experimentos foi desenvolvido com base na API MSG do SimGrid, que fornece as funções básicas para a simulação de sistemas distribuídos heterogêneos. Em uma simulação MSG, cada elemento que compõem o sistema é caracterizado como um *host*, isto é, uma entidade independente capaz de executar processos. Considerando-se o modelo de grade móvel adotado (Figura 3.1), os *hosts* foram organizados de acordo com uma relação mestre-escravo. Mais precisamente, o mestre (*proxy*)

Tabela 4.6: Fatores e níveis considerados no planejamento fatorial completo para a avaliação por simulação. Os valores adotados para o *deadline* são os mesmos do primeiro conjunto de experimentos.

Fatores	Níveis
Algoritmo	*Maximum Regret* e *Greedy*
Intervalo Médio	16 s e 32 s
Deadline	curto e estendido

executa um processo que, de acordo com uma distribuição de Poisson, gera um conjunto de aplicações ao longo do tempo, sendo também responsável pela execução dos algoritmos de escalonamento e gerenciamento dos recursos computacionais. Por outro lado, os escravos (recursos providos pelos dispositivo móveis) executam as tarefas atribuídas pelo mestre e retornam os resultados.

A configuração do sistema foi feita a partir de arquivos XML (*eXtensible Markup Language*), contendo informações sobre a largura de banda, latência e topologia da rede, além da capacidade de processamento e função (mestre ou escravo) de cada elemento do sistema. Deve-se também ressaltar que, nos cenários simulados, adotou-se que os *hosts* estão dentro de uma área bem definida, portanto não havendo indisponibilidade de recursos devido a problemas relativos à mobilidade ou comunicação. Por fim, em relação ao tempo de vida de bateria, o mestre é responsável por verificar se um determinado escravo excedeu a capacidade máxima estipulada.

4.2.2 Análise dos Resultados

Nos gráficos da Figura 4.6, são apresentados os valores obtidos para o consumo médio de energia por aplicação aceita para execução. Observa-se que o desempenho do algoritmo *Greedy* foi ligeiramente melhor em relação ao *Maximum Regret*, podendo-se destacar o experimento com intervalo médio de submissão de 32 segundos e *deadline* estendido, no qual a diferença de consumo de energia foi de 6,18%.

Em relação ao intervalo médio entre submissões, nota-se que a adoção de um intervalo pequeno em relação ao *deadline* estipulado provoca uma degradação no consumo de energia. Nessas situações, para que os *deadlines* sejam cumpridos, faz-se necessária a utilização de recursos energeticamente menos favoráveis. Considerando-se os experimentos com *deadline* curto, a redução no intervalo médio de submissão levou a um aumento no consumo de energia de 2,89% e 2,19% para os algoritmos *Maximum Regret* e *Greedy*, respectivamente. Já para o *deadline* estendido, o aumento verificado foi de 24,32% para o algoritmo *Maximum Regret*, e de 27,85% para o *Greedy*.

Os resultados obtidos para a variável de resposta taxa de aceitação são apresentados nos gráficos da Figura 4.7. Tem-se que o desempenho dos algoritmos propostos é muito próximo, não sendo observada uma diferença maior que três pontos percentuais.

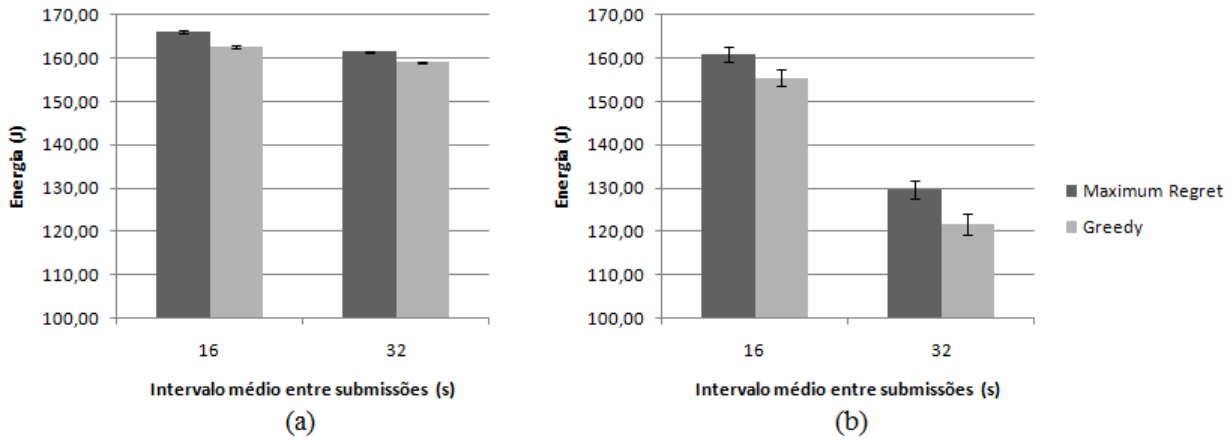


Figura 4.6: Consumo médio de energia por aplicação aceita para execução na grade móvel. Os resultados foram organizados de acordo com o *deadline*: (a) curto e (b) estendido.

Nota-se também que o tempo intervalo médio entre submissões possui um impacto significativo na taxa de aceitação independentemente do algoritmo e do *deadline* adotados. Considerando-se o *deadline* curto, a redução no intervalo médio entre submissões fez com que a taxa de aceitação caísse em 31,53% e 32,63% para os algoritmos *Maximum Regret* e *Greedy*, respectivamente. Já para o *deadline* estendido, a queda observada foi de 34,30% para o *Maximum Regret*, e de 35,67% para o *Greedy*.

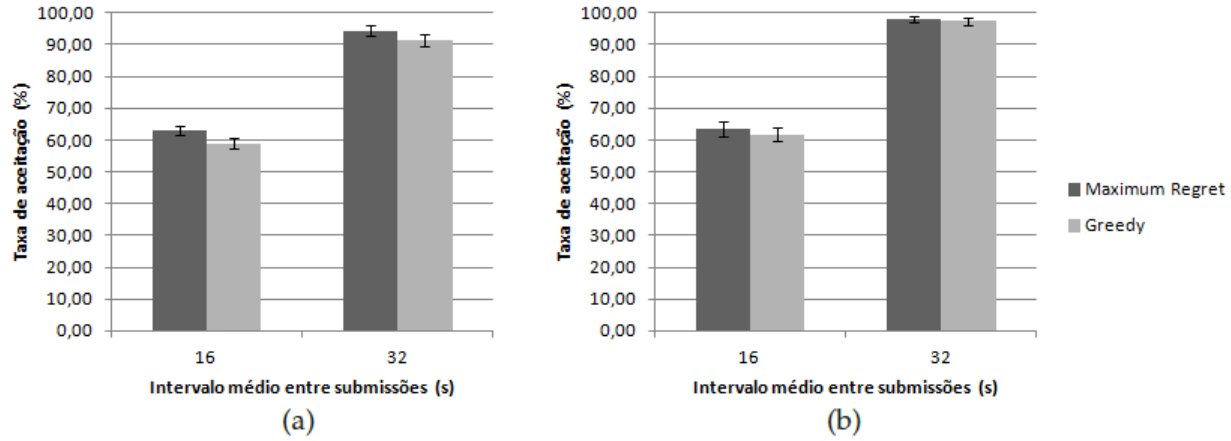


Figura 4.7: Taxa de aceitação das aplicações submetidas para execução na grade móvel. Os resultados foram organizados de acordo com o *deadline*: (a) curto e (b) estendido.

4.2.3 Análise de Influência dos Fatores

Analisando-se o gráfico de probabilidade normal ilustrado na Figura 4.8, pode-se observar que os fatores que mais influenciam o consumo médio de energia por aplicação são o *deadline* (A) e o intervalo entre submissões (B), sendo que interação AB também mostrou-se significativa. Assim como no primeiro conjunto de experimentos, o fator algoritmo (C) apresentou uma influência pouco significativa no consumo de energia, confirmando que o desempenho dos algoritmos propostos é semelhante nos cenários considerados durante

as simulações.

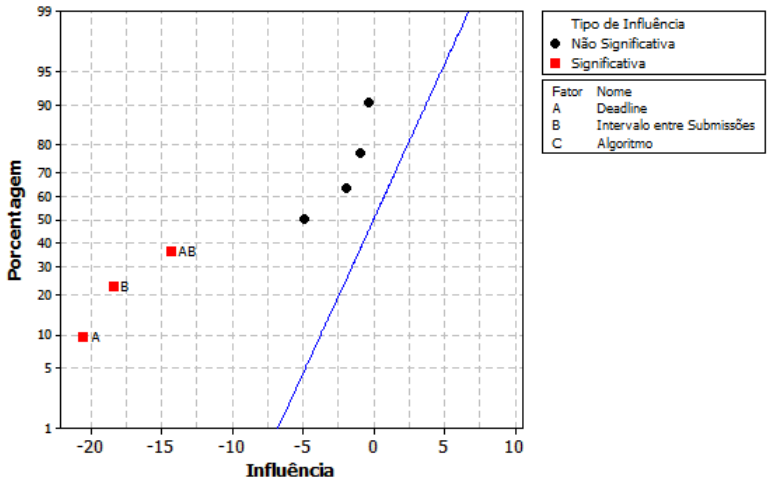


Figura 4.8: Gráfico de probabilidade normal para a influência dos fatores no consumo médio de energia por aplicação aceita para execução na grade móvel.

Por outro lado, em relação à taxa de aceitação (Figura 4.9), apenas o fator intervalo entre submissões (B) apresentou uma influência significativa. Isso evidencia que a sobrecarga do sistema (alto número de tarefas em relação aos recursos disponíveis na grade móvel) pode levar à cenários nos quais os algoritmos propostos podem não encontrar soluções factíveis para o problema de escalonamento ciente do consumo de energia.

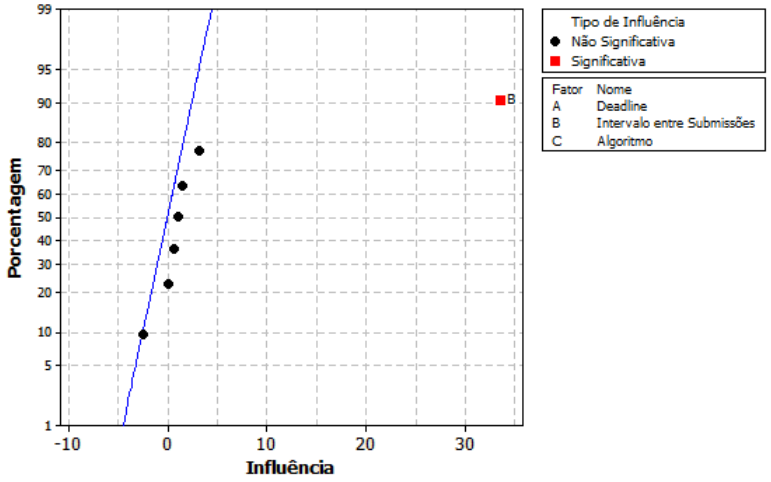


Figura 4.9: Gráfico de probabilidade normal para a influência dos fatores na taxa de aceitação de aplicações para execução na grade móvel.

4.3 Considerações Finais

Neste capítulo, foram apresentados os experimentos realizados para avaliar o desempenho dos algoritmos *Maximum Regret* e *Greedy*. Para tanto, dois conjuntos de experi-

mentos foram planejados: no primeiro, buscou-se mensurar a proximidade das soluções obtidas pelos algoritmos propostos em relação a um referencial determinado pelo *solver* Gurobi com tempo limite de um minuto; no segundo, por meio de simulação, o objetivo foi avaliar o comportamento dos algoritmos em um ambiente dinâmico.

Em relação ao consumo total de energia, pode-se demonstrar que os algoritmos propostos foram capazes de encontrar soluções próximas ao referencial adotado dentro de um intervalo de tempo razoável. Em seu pior caso, o algoritmo *Maximum Regret* foi 12,18% pior que a solução provida pelo *solver* Gurobi; já no pior caso do algoritmo *Greedy*, tal diferença foi de apenas 8,14%. Deve-se ressaltar que o Gurobi nem sempre foi capaz de encontrar soluções ótimas (isto é, que satisfizessem o parâmetro MIPGap) no tempo estipulado. Tal característica não é desejável em um sistema dinâmico como uma grade móvel, sendo justificada, portanto, a utilização de heurísticas para a solução do problema de escalonamento ciente de consumo de energia.

No contexto dos experimentos por simulação, observou-se que o consumo médio de energia foi afetado diretamente pelos fatores *deadline* e intervalo médio entre submissões, refletindo, portanto, a necessidade de alocação de recursos menos favoráveis energeticamente em situações de sobrecarga do sistema. Ainda considerando situações de sobrecarga, deve-se ressaltar que algoritmos propostos nem sempre foram capazes de encontrar soluções factíveis para o problema de escalonamento.

Capítulo

5

Conclusão

Em um cenário de compartilhamento de recursos, o tempo de vida de bateria é um fator crítico para o desempenho de uma grade móvel. Logo, o gerenciamento ineficaz dos recursos compartilhados pode afetar de maneira significativa o consumo de energia e de comunicação, limitando o tempo de bateria e, consequentemente, a disponibilidade dos dispositivos participantes. Dessa forma, uma grade móvel deve ser capaz de gerenciar o consumo de energia dos recursos ofertados, sem que o desempenho do sistema seja comprometido. É essencial, portanto, que a política de escalonamento adotada consiga estabelecer uma relação equilibrada entre o consumo de energia e os requisitos das aplicações submetidas.

Considerando-se o contexto de gerenciamento energético em grades móveis, este trabalho propôs dois algoritmos de escalonamento (*Maximum Regret* e *Greedy*) que, além de buscar minimizar o consumo de energia, visam assegurar o cumprimento dos requisitos de qualidade de serviço das aplicações submetidas pelos usuários. Tais algoritmos foram projetados a partir de soluções heurísticas para o problema de escalonamento ciente de consumo de energia em grades móveis. Mais precisamente, o problema de escalonamento foi formulado como um modelo de programação linear inteira, cujo objetivo é minimizar o consumo total de energia necessário para a execução de aplicações *bag-of-tasks* em uma grade móvel *on-site*.

A avaliação de desempenho realizada teve dois objetivos principais: primeiro, mensurar a proximidade das soluções heurísticas obtidas em relação a um referencial ótimo, determinado pelo *solver* Gurobi; segundo, avaliar, por meio de simulação, o comportamento dos algoritmos em um ambiente dinâmico. A partir dos experimentos executados, pode-se demonstrar a viabilidade das políticas de escalonamento adotadas em termos de minimização do consumo de energia. No entanto, deve-se também ressaltar que os algoritmos *Maximum Regret* e *Greedy* apresentaram limitações quando a grade móvel estava

sobrecarregada (alto número de tarefas em relação aos recursos disponíveis). Nessas situações, os algoritmos propostos nem sempre foram capazes de encontrar soluções factíveis para o problema de escalonamento ciente do consumo de energia.

5.1 Contribuições

Dentre as contribuições deste trabalho, destacam-se: a formulação do problema de escalonamento ciente de consumo de energia em grades móveis como um problema de programação linear inteira; a proposição de dois algoritmos heurísticos para solucionar tal problema; avaliação estática e dinâmica dos algoritmos propostos por meio de experimentos que englobaram diferentes fatores envolvidos em uma grade móvel.

Com relação à produção científica, um resumo expandido com uma descrição do algoritmo *Maximum Regret* foi publicado nos anais da IV Escola Regional de Alto Desempenho de São Paulo (ERAD-SP 2013). O trabalho também foi apresentado oralmente no fórum de pós-graduação. Dentre os 43 trabalhos aceitos para o evento, apenas os 9 melhores foram escolhidos para a apresentação oral.

Além do resumo expandido, em Dezembro de 2013, um artigo baseado nos resultados descritos nesta dissertação foi submetido para o 32° Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC 2014).

5.2 Trabalhos Futuros

Em uma grade móvel voltada para o compartilhamento de recursos, o projeto de políticas de escalonamento envolve diversos aspectos, muitos dos quais não puderam ser considerados neste trabalho em função do tempo ou escopo. A seguir, são apresentadas possíveis propostas para a continuação deste trabalho:

- Aprimoramento da caracterização dos dispositivos móveis em termos de consumo de energia. A estimativa do consumo total de energia de um dispositivo deve levar em consideração custos associados à comunicação (WiFi, Bluetooth), iluminação de tela e execução de aplicações locais.

- Desenvolvimento de um mecanismo para análise de padrões de mobilidade dos dispositivos móveis, que permita que sejam inferidas informações sobre a disponibilidade dos recursos ofertados.

- Como contraponto à minimização global do consumo de energia, projetar e avaliar políticas de escalonamento que consigam estender de maneira uniforme o tempo de vida de bateria dos dispositivos móveis.

Referências Bibliográficas

Andrade, N., Costa, L., Germóglio, G., e Cirne, W. (2005). Peer-to-peer grid computing with the ourgrid community. In *23rd Brazilian Symposium on Computer Networks (SBRC 2005)-4th Special Tools Session*.

Baker, M., Buyya, R., e Laforenza, D. (2002). Grids and grid technologies for wide-area distributed computing. *Software Practice and Experience*, 32:1437–1466.

Black, M. e Edgar, W. (2009). Exploring mobile devices as grid resources: Using an x86 virtual machine to run boinc on an iphone. In *Grid Computing, 2009 10th IEEE/ACM International Conference on*, pp. 9–16. IEEE.

Borges, V., Rossetto, A., e Dantas, M. (2009). An architecture for handling disconnections in workflow applications in mobile grid environments. *Latin America Transactions, IEEE (Revista IEEE America Latina)*, 7(6):681–687.

Buyya, R., Abramson, D., e Venugopal, S. (2005). The grid economy. *Special Issue on Grid Computing, Proceedings of the IEEE*, 93:698–714.

Carroll, A. e Heiser, G. (2010). An analysis of power consumption in a smartphone. In *Proceedings of the 2010 USENIX Annual Technical Conference*, pp. 1–12, Boston, MA, USA.

Casanova, H., Legrand, A., e Quinson, M. (2008). Simgrid: a generic framework for large-scale distributed experiments. In *Proceedings of the Tenth International Conference on Computer Modeling and Simulation*, UKSIM '08, pp. 126–131, Washington, DC, USA. IEEE Computer Society.

Casavant, T. L. e Kuhl, J. G. (1988). A taxonomy of scheduling in general-purpose distributed computing systems. *IEEE Transactions on Software Engineering*, 14:141–154.

Cirne, W., Brasileiro, F., Sauvé, J., Andrade, N., Paranhos, D., Santos-neto, E., Medeiros, R., e Gr, F. C. (2003). Grid computing for bag of tasks applications. In *In Proc. of the 3rd IFIP Conference on E-Commerce, E-Business and EGovernment*.

Coronato, A. e Pietro, G. D. (2008). Mipeg: A middleware infrastructure for pervasive grids. *Future Generation Computer Systems*, 24(1):17–29.

Croarkin, C., Tobias, P., Filliben, J. J., Hembree, B., Guthrie, W., Trutna, L., e Prins, J., editores (2010). *NIST/SEMATECH e-Handbook of Statistical Methods*. NIST/SEMATECH.

Da Silva, D. P., Cirne, W., e Brasileiro, F. V. (2003). Trading cycles for information: Using replication to schedule bag-of-tasks applications on computational grids. In *Euro-Par 2003 Parallel Processing*, pp. 169–180. Springer.

da Silva, F. A. B., Carvalho, S., Senger, H., Hruschka, E. R., e de Farias, C. R. G. (2004). Running data mining applications on the grid: A bag-of-tasks approach. In *ICCSA (2)*, pp. 168–177.

D'Andria, F., Martrat, J., Jiménez, S., Wesner, S., Ritrovato, P., e Laria, G. (2008). The akogrimo mobile grid framework as enabling technology for next generation elearning paradigms. In *The Learning Grid Handbook*, v. 2 of *Future of Learning*, pp. 125–142. Ios Press.

Dantas, M. (2005). *Computação Distribuída de Alto Desempenho. Redes, Clusters e Grids Computacionais*. AXCEL BOOKS, Rio de Janeiro.

Dong, F. e Akl, S. G. (2006). Scheduling algorithms for grid computing: State of the art and open problems. Technical Report 2006-504, School of Computing — Queen's University, Kingston, Ontario, Canada.

Duan, L., Kubo, T., Sugiyama, K., Huang, J., Hasegawa, T., e Walrand, J. (2012). Incentive mechanisms for smartphone collaboration in data acquisition and distributed computing. In *INFOCOM, 2012 Proceedings IEEE*, pp. 1701–1709. IEEE.

Fernández, J.-J., Gordon, D., e Gordon, R. (2008). Efficient parallel implementation of iterative reconstruction algorithms for electron tomography. *J. Parallel Distrib. Comput.*, 68(5):626–640.

Foster, I. (2002). The grid: A new infrastructure for 21st century science. *Physics Today*, 55:42–47.

Foster, I. (2005). Globus toolkit version 4: Software for service-oriented systems. In *IFIP International Conference on Network and Parallel Computing*, pp. 2–13. Springer-Verlag.

Foster, I. e Kesselman, C. (1996). Globus: A metacomputing infrastructure toolkit. *International Journal of Supercomputer Applications*, 11:115–128.

Foster, I., Kesselman, C., e Tuecke, S. (2001). The anatomy of the grid: Enabling scalable virtual organizations. *The International Journal of High Performance Computing Applications*, 15:200–222.

Fujimoto, N. e Hagihara, K. (2004). A comparison among grid scheduling algorithms for independent coarse-grained tasks. In *Applications and the Internet Workshops, 2004. SAINT 2004 Workshops. 2004 International Symposium on*, pp. 674–680. IEEE.

Furthmüller, J. e Waldhorst, O. (2010). A survey on grid computing on mobile consumer devices. In *Handbook of Research on P2P and Grid Systems for Service-Oriented Computing*, Cap. 13, pp. 313–337. IGI-Global, Hershey, PA.

Furthmüller, J. e Waldhorst, O. (2012). Energy-aware resource sharing with mobile devices. *Computer Networks*, 56(7):1920–1934.

Ghosh, P. e Das, S. K. (2010). Mobility-aware cost-efficient job scheduling for single-class grid jobs in a generic mobile grid architecture. *Future Generation Computer Systems*, 26(8):1356–1367.

Huang, C.-Q., Zhu, Z.-T., Wu, Y.-H., e Xiao, Z.-H. (2006). Power-aware hierarchical scheduling with respect to resource intermittence in wireless grids. In *Proc. of the Fifth Int. Conf. on Machine Learning and Cybernetics*.

Jain, R. (1991). *The art of computer systems performance analysis - techniques for experimental design, measurement, simulation, and modeling*. Wiley professional computing. Wiley.

Katsaros, K. e Polyzos, G. C. (2007). Optimizing operation of a hierarchical campus-wide mobile grid for intermittent wireless connectivity. In *Proceedings of the 15 IEEE Workshop on Local and Metropolitan Area Networks*, pp. 111–116, New York, NY, USA. IEEE Press.

Kim, K. H., Buyya, R., e Kim, J. (2007). Power aware scheduling of bag-of-tasks applications with deadline constraints on dvs-enabled clusters. In *Proceedings of the Seventh IEEE International Symposium on Cluster Computing and the Grid*, CCGRID '07, pp. 541–548, Washington, DC, USA. IEEE Computer Society.

Krauter, K., Buyya, R., e Maheswaran, M. (2002). A taxonomy and survey of grid resource management systems for distributed computing. *Software Practice and Experience*, 32:135–164.

Li, C. e Li, L. (2010). Energy constrained resource allocation optimization for mobile grids. *Journal of Parallel and Distributed Computing*, 70(3):245–258.

Li, C. e Li, L. (2012). Collaboration among mobile agents for efficient energy allocation in mobile grid. *Information Systems Frontiers*, 14(3):711–723.

Li, Z. e Shen, H. (2012). Game-theoretic analysis of cooperation incentive strategies in mobile ad hoc networks. *Mobile Computing, IEEE Transactions on*, 11(8):1287–1303.

Lima, L. d. S. (2007). *Um Protocolo para Descoberta e Seleção de Recursos em Grades Móveis Ad hoc*. PhD thesis, Pontifícia Universidade Católicca do Rio de Janeiro, Rio de Janeiro.

Ma, Y., Gong, B., Sugihara, R., e Gupta, R. (2012). Energy-efficient deadline scheduling for heterogeneous systems. *Journal of Parallel and Distributed Computing*, 72(12):1725–1740.

Martello, S. e Toth, P. (1981). An algorithm for the generalized assignment problem. *Operational research*, 81:589–603.

Martello, S. e Toth, P. (1984). Worst-case analysis of greedy algorithms for the subset-sum problem. *Mathematical programming*, 28(2):198–205.

Martello, S. e Toth, P. (1990). *Knapsack problems: algorithms and computer implementations*. John Wiley & Sons, Inc., New York, NY, USA.

Paradiso, J. A. e Starner, T. (2005). Energy scavenging for mobile and wireless electronics. *Pervasive Computing, IEEE*, 4(1):18–27.

Park, S., bae Ko, Y., e hoon Kim, J. (2003). Disconnected operation service in mobile grid computing. In *First International Conference on Service Oriented Computing (ICSOC'2003)*, pp. 499–513.

Rodriguez, J. M., Mateos, C., e Zunino, A. (2012). Are smartphones really useful for scientific computing? In *Advances in New Technologies, Interactive Interfaces and Communicability*, pp. 38–47. Springer.

Rodriguez, J. M., Mateos, C., e Zunino, A. (2012). Energy-efficient job stealing for cpu-intensive processing in mobile devices. *Computing*, pp. 1–31.

Rodriguez, J. M., Zunino, A., e Campo, M. (2010). Mobile grid seas: simple energy-aware scheduler. In *Proc. 3rd High-Performance Computing Symposium-39th JAIIO*.

Rodriguez, J. M., Zunino, A., e Campo, M. R. (2011). Introducing mobile devices into grid systems: a survey. *International Journal of Web and Grid Services*, 7(1):1–40.

Rosado, D. G., Fernandez-Medina, E., López, J., e Piattini, M. (2011). Systematic design of secure mobile grid systems. *Journal of Network and Computer Applications*, 34(4):1168–1183.

Shah, S. C. e Park, M.-S. (2012). An effective and robust two-phase resource allocation scheme for interdependent tasks in mobile ad hoc computational grids. *Journal of Parallel and Distributed Computing*.

Stiles, J. R., Bartol, Jr., T. M., Salpeter, E. E., e Salpeter, M. M. (1998). Monte carlo simulation of neuro-transmitter release using mcell, a general simulator of cellular physiological

processes. In *Proceedings of the sixth annual conference on Computational neuroscience*, CNS '97, pp. 279–284, New York, NY, USA. Plenum Press.

Stiller, B. e Waldburger, M. (2005). Toward the mobile grid: Service provisioning in a mobile dynamic virtual organization. Technical Report 2005.07, IFI, University of Zurich.

Teixeira, F. C. (2012). *Grid Anywhere : Um middleware extensível para grades computacionais desktop*. PhD thesis, Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Paulo.

Teixeira, F. C., Santana, M. J., Santana, R. H. C., e Estrella, J. C. (2010). Grid anywhere: An architecture for grid computing able to explore the computational resources of the set-top boxes. In *Networks for Grid Applications*, pp. 79–88. Springer.

Teodoro, S. (2013). Algoritmos de escalonamento para grades computacionais voltados à eficiência energética. Master's thesis, Pontifícia Universidade Católica do Rio Grande do Sul.

Teske, H., Furthmüller, J. W., e Waldhorst, O. P. (2011). A resilient and energy-saving incentive system for resource sharing in manets. In *KiVS*, pp. 109–120.

Viswanathan, H., Chen, B., e Pompili, D. (2012). Research challenges in computation, communication, and context awareness for ubiquitous healthcare. *IEEE Communications Magazine*, 50(5):92–99.

Viswanathan, H., Lee, E. K., e Pompili, D. (2012). Mobile grid computing for data-and patient-centric ubiquitous healthcare. In *Enabling Technologies for Smartphone and Internet of Things (ETSIoT), 2012 First IEEE Workshop on*, pp. 36–41. IEEE.

Wu, T., Yeh, K., Chen, R., Chen, C. C., e Chen, Y. C. (2011). Radio frequency identification (rfid) with mobile-grid based framework for farmland irrigation facility management. *African Journal of Agricultural Research*, 6(31):6499–6512.

Xhafa, F. e Abraham, A. (2010). Computational models and heuristic methods for grid scheduling problems. *Future Generation Computer Systems*, 26:608–621.