UNIVERSIDADE FEDERAL FLUMINENSE
CENTRO TECNOLÓGICO
MESTRADO EM ENGENHARIA DE PRODUÇÃO

HUGO HARRY FREDERICO RIBEIRO KRAMER

ALGORITMOS PARA OTIMIZAÇÃO ENERGÉTICA EM *CLUSTERS* COMPUTACIONAIS DE GRANDE ESCALA

NITERÓI
2011

HUGO HARRY FREDERICO RIBEIRO KRAMER

ALGORITMOS PARA OTIMIZAÇÃO ENERGÉTICA EM *CLUSTERS* COMPUTACIONAIS DE GRANDE ESCALA

Dissertação apresentada ao Curso de Pós-Graduação em Engenharia de Produção da Universidade Federal Fluminense, como requisito parcial para obtenção do Grau de Mestre. Área de Concentração: Sistemas, Apoio à decisão e Logística.

Orientador: Prof. Dr. EDUARDO UCHOA BARBOZA

Niterói

2011

HUGO HARRY FREDERICO RIBEIRO KRAMER

ALGORITMOS PARA OTIMIZAÇÃO ENERGÉTICA EM *CLUSTERS* COMPUTACIONAIS DE GRANDE ESCALA

Dissertação apresentada ao Curso de Pós-Graduação em Engenharia de Produção da Universidade Federal Fluminense, como requisito parcial para obtenção do Grau de Mestre. Área de Concentração: Sistemas, Apoio à decisão e Logística.

Aprovada em fevereiro de 2011.

BANCA EXAMINADORA

Prof. Dr. EDUARDO UCHOA BARBOZA - Orientador

UFF

Prof. Dr. ARTUR ALVES PESSOA

UFF

Prof. Dr. MARCUS VINICIUS S. POGGI DE ARAGÃO

PUC-RIO

Niterói

2011

Este trabalho é dedicado aos meus pais Hugo e Mariza pelo pleno amor, esforço e entrega ao longo da minha vida.

Agradecimentos

Primeiramente aos meus pais por todo o carinho e dedicação prestados a mim e meus irmãos além de todas as dificuldades enfrentadas para que pudéssemos ter saúde, a melhor educação possível e união familiar. Agradeço também a todos os meus familiares, em especial à minha tia Minininha e ao meu tio Tuti pelo suporte incondicional.

Ao Professor Eduardo Uchoa, pela efetiva orientação e todo o apoio que possibilitaram a conclusão deste trabalho e pela disponibilidade e presteza para ajudar também em situações que excedem o contexto acadêmico. Por isso, sou profundamente agradecido e sinto-me honrado em ser orientado pelo Prof. Eduardo Uchoa.

Agradeço ao grande amigo Anand Subramanian pela imensa e fundamental ajuda na realização deste trabalho e também pela amizade e constante disponibilidade para ajudar. Agradeço também ao amigo Vinicius Petrucci pelo auxílio sem o qual este trabalho não teria se concretizado.

Ao amigo/irmão Jorge Guida pela amizade e companheirismo e também à sua família pelo carinho, aos amigos de Passa Quatro-MG e aos amigos Téo, Calanguinho, Birinho, Léo, Tico e aos que possa ter esquecido no momento.

Aos colegas do IC-UFF que me receberam muito bem e torceram por mim quando cheguei a Niterói e a todos os demais que fiz ao longo do mestrado.

Obrigado aos amigos e colegas do Logis, Marcos, Luiz, Prof. Artur, Ubiratam e Rafaelli pela ajuda e companheirismo.

À CAPES pela bolsa de mestrado que foi de fundamental importância para dar as condições necessárias para que eu pudesse cursar o mestrado.

Aos professores e funcionários do Departamento de Engenharia de Produção da Universidade Federal Fluminense que de certa forma contribuíram para a realização deste trabalho.

Resumo

De maneira crescente, *clusters* de servidores têm sido empregados no apoio ao desenvolvimento e implementação de uma grande variedade de serviços, com demandas por processamento distintas e variáveis ao longo do tempo, em ambientes computacionais escaláveis e eficientes. Plataformas emergentes conhecidas como computação em nuvem colocam à disposição diversos serviços *web* que são hospedados e compartilhados em uma estrutura de servidores físicos e consolidados através de técnicas de virtualização. A grande quantidade de energia consumida por estes *clusters* de servidores se tornou uma importante questão dos pontos de vista econômico e ecológico, criando a necessidade de se investigar técnicas de otimização capazes de melhorar a eficiência energética de tais infraestruturas computacionais. Neste trabalho são apresentados modelos e algoritmos para se obter o máximo de economia de energia em um *cluster*, não apenas ligando/desligando servidores, mas também ajustando as frequências de operação de suas CPUs. Os novos modelos diferem de outros encontrados na literatura por serem mais aderentes à realidade, considerando inclusive a heterogeneidade dos servidores. Os algoritmos de otimização propostos são baseados em técnicas de geração de colunas e os experimentos realizados indicam que eles são capazes de obter soluções de alta qualidade, de forma robusta e em baixo tempo computacional, mesmo para instâncias de grande porte.

Palavras-chave: Geração de colunas, Gerenciamento de energia, Virtualização de *clusters* de servidores.

Abstract

Increasingly, clusters of servers have been deployed in large data centers to support the development and implementation of many kinds of services, with distinct workload demands that vary over time, in a scalable and efficient computing environment. Emerging trends are utility/cloud computing platforms, where many network services, implemented and supported using server virtualization techniques, are hosted on a shared cluster infrastructure of physical servers. The energy consumed to maintain these large server clusters became a very important economic and ecological concern, which in turn, requires major investigation of optimization techniques to improve the energy efficiency of their computing infrastructure. This work deals with models and algorithms whose the goal is to obtain the maximum energy economy in a server cluster, not only by the use of on/off mechanisms, but also adjusting its CPUs operating frequencies. Such new models differs from those found in literature by their more realistic assumptions, which include the servers heterogeneity. The proposed optimization algorithms are based in column generation techniques, and the experiments suggest that they are suitable to obtain consistently high quality solutions in a short amount of processing time, even in large instances.

Keywords: Column generation, Energy management, Server cluster virtualization.

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Abreviações

DVFS	*Dynamic Voltage and Frequency Scaling*
GC	Geração de Colunas
HA	Heurística de Arredondamento
HPMRI	Heurística do Problema Mestre Restrito Inteiro
LA	Lista de Serviços
LB	*Lower Bound*
LS	Lista de Servidores
MFLOP	Milhões de Operações de Ponto Flutuante por Segundo
OC	Otimização Combinatória
PCCVH	Problema de Configuração de *Clusters* Virtualizados Heterogêneos
PCM	Problema de Corte de Materiais
PCMML	Problema de Corte de Materiais com Múltiplas Larguras
PEC	Problema de Empacotamento em Caixas
PECTV	Problema de Empacotamento em Caixas de Tamanho Variável
PL	Programação Linear
PLI	Programação Linear Inteira
PM	Problema Mestre
PMR	Problema Mestre Restrito
PMRI	Problema Mestre Restrito Inteiro

Capítulo 1

Introdução

Um *cluster* de servidores é um sistema distribuído que consiste de centenas ou milhares de máquinas interligadas por uma rede rápida (Cardellini et al, 2002). Um número crescente desses sistemas tem sido empregado em *data centers* de grande escala no apoio ao desenvolvimento e implementação de diferentes serviços e aplicações em ambientes computacionais escaláveis e eficientes, especialmente focados em serviços baseados na Internet.

O surgimento das plataformas conhecidas como computação em nuvem (Hayes, 2008; Church et al, 2008), a exemplo da Amazon EC2 e da Google App Engine, resulta no aumento da demanda por tais arquiteturas de *clusters* de servidores. Nessas plataformas, os serviços são hospedados em sua maioria em servidores físicos compartilhados e suas cargas de processamento podem ser distintas, além de estarem sujeitas a variações ao longo do tempo.

Os *clusters* de servidores que viabilizam estas plataformas têm, em geral, um grande porte de maneira que seja possível atender a grandes demandas de processamento e implicam em um elevado consumo de energia elétrica, contribuindo indiretamente com o aumento da geração e emissão de gases de efeito estufa e a consequente deterioração do meio ambiente (Kaplan et al, 2008), e diretamente com a elevação dos custos envolvidos para mantê-los em operação. Por exemplo, de acordo com Koomey (2007), a energia consumida em 2005 por servidores e sua infraestrutura associada estava em torno de 1,2% do total consumido nos Estados Unidos a um custo estimado em US$ 2,7 bilhões neste país e US$ 7,2 bilhões no mundo. O mesmo autor estima o aumento do consumo de energia elétrica por *clusters* entre 2005 e 2010 em aproximadamente 40% se o crescimento seguir a tendência do período entre 2000 e 2005. Isto mostra que, de fato, o tema deve ser visto

como uma questão de interesse quanto à investigação do uso de técnicas de otimização que contemplem o aprimoramento da eficiência energética dos *clusters* e da infraestrutura envolvida (Bianchini e Rajamony, 2004; Fan et al, 2007; Ranganathan, 2010).

A melhor utilização dos recursos de *hardware* de um *cluster* (e, com isso, o aumento da eficiência energética destes ambientes computacionais), onde diversos serviços independentes e com demandas por processamento distintas devem ser executados, se faz possível por meio da aplicação de técnicas de virtualização. Segundo Kiyanclar (2005), virtualização é a reprodução fiel em *software* de uma arquitetura completa dando a ilusão de uma máquina real às aplicações executadas sobre a máquina virtual.

O emprego de técnicas de virtualização possibilita a utilização de uma ou mais máquinas virtuais em um único servidor físico e tornam possível a consolidação de vários serviços por servidor. Além disso, é possível distribuir dinamicamente os serviços aos servidores de um ambiente virtualizado permitindo que alguns servidores sejam desligados em períodos de baixa demanda e ligados novamente quando for exigido. Isto também pode ser combinado com o que se denomina em inglês por *Dynamic Voltage and Frequency Scaling* (DVFS), uma técnica que consiste em variar a frequência e a tensão do processador em tempo de execução de acordo com as necessidades de processamento com o intuito de utilizar a energia de maneira mais eficiente. Exemplos de mecanismos DVFS encontrados nos microprocessadores atuais são a "Enhanced Speedstep Technology" da Intel e a "PowerNOW!" da AMD.

As técnicas de virtualização, bem como DVFS e outras vêm sendo exploradas na literatura com foco na racionalização do consumo energético em processadores (Rusu et al, 2006; Horvath et al, 2007; Bertini et al, 2007), tipicamente conhecidos como os principais consumidores de energia em um servidor. O uso destas técnicas é justificado pelo fato de que, durante a maior parte do tempo, os servidores em *data centers* operam com uma carga que varia entre 10% e 50% de sua capacidade de pico (Barroso e Hölzle, 2007) e podem viabilizar a consolidação das cargas de processamento dos serviços, resultando no aumento da taxa de utilização dos equipamentos e redução do consumo de energia (Wang et al, 2008; Srikantaiah et al, 2008; Kusic et al, 2009).

Na prática, os *clusters* são, em sua grande maioria, heterogêneos, ou seja, compostos por servidores com diferentes características de *hardware*. Segundo Heath et al (2005), mesmo um *cluster* homogêneo no momento de sua instalação, rapidamente se torna hete-

rogêneo pelas seguintes razões: (i) os componentes que apresentam falhas são, em geral, substituídos por outros mais novos e com melhor performance, (ii) qualquer acréscimo adicional na performance ou capacidade para atender a um eventual aumento da demanda se dá pela incorporação de novos componentes ao *cluster*, diferentes dos já existentes e (iii) os servidores tradicionais no estilo PC estão sendo substituídos por servidores no estilo *blade*, com o objetivo de diminuir o espaço ocupado e facilitar o manuseio.

Neste contexto, o problema a ser tratado neste trabalho é o de encontrar a melhor configuração para o conjunto de servidores de um *cluster* virtualizado heterogêneo de grande escala com o objetivo de minimizar os custos relacionados ao consumo de energia elétrica de forma que a demanda por processamento e a execução dos serviços no *cluster* seja garantida. Para isto, considera-se que mais de um serviço pode ser executado por um servidor físico por meio de técnicas de virtualização, um determinado servidor físico pode ser ligado e desligado dependendo da situação e a capacidade de processamento de um servidor pode ser ajustada de acordo com a demanda dos serviços utilizando o mecanismo de DVFS. Daqui em diante, este problema será chamado de Problema de Configuração de *Clusters* Virtualizados Heterogêneos (PCCVH).

Neste trabalho são propostas abordagens baseadas em Geração de Colunas (GC) para tratar o PCCVH de forma a encontrar soluções de boa qualidade em tempo computacional razoável, tendo em vista que em situações práticas os serviços e suas cargas são dinâmicos e o *cluster* deve ser reconfigurado periodicamente para garantir o atendimento das demandas a cada mudança. É importante notar que uma abordagem de otimização aplicada ao PCCVH pode ser economicamente significante, já que uma redução do consumo de energia elétrica em torno de 10% em um *cluster* de larga escala pode representar uma considerável redução dos custos de operação de uma organização em que estes ambientes computacionais são seus principais recursos.

No entanto, não é do conhecimento do autor a existência de trabalhos que considerem em conjunto o emprego de virtualização, DVFS e a possibilidade de ligar/desligar servidores em uma abordagem de otimização capaz de obter soluções de qualidade com baixo tempo de processamento para tratar da economia de energia em *clusters* de servidores heterogêneos virtualizados.

Os objetivos deste trabalho são:

- Definir um problema de configuração de *clusters* que considera virtualização, DVFS

e heterogeneidade dos servidores;

- Propor uma formulação de Programação Linear Inteira para modelar o PCCVH;
- Propor uma relaxação de Programação Linear Inteira para encontrar limites inferiores para o PCCVH;
- Aplicar a Decomposição de Dantzig-Wolfe e obter formulações que permitam o uso de algoritmos de GC;
- Propor heurísticas baseadas em algoritmos de GC para obter soluções inteiras viáveis para o PCCVH;
- Gerar instâncias de problemas-teste para avaliação do desempenho dos algoritmos propostos;
- Testar os algoritmos propostos nas instâncias geradas;
- Avaliar o desempenho dos algoritmos com base nos resultados obtidos nas instâncias geradas;
- Apresentar perspectivas de trabalhos futuros para resolução do PCCVH.

O restante desta dissertação está organizado da maneira que segue. O Capítulo 2 descreve os fundamentos teóricos necessários para o desenvolvimento das abordagens propostas. No Capítulo 3 é apresentada uma breve revisão sobre os Problemas de Corte e Empacotamento, já que o PCCVH pode ser visto como uma variante deste grupo de problemas. O Capítulo 4 descreve o PCCVH, como é tratado na literatura e apresenta as abordagens de solução propostas nesta dissertação. O Capítulo 5 apresenta a metodologia aplicada e os resultados obtidos nos experimentos realizados para testar os algoritmos propostos. Por fim, o Capítulo 6 contém as conclusões acerca das abordagens de solução propostas para tratar o PCCVH.

Capítulo 2

Decomposição de Dantzig-Wolfe e Geração de Colunas

O método de GC foi pela primeira vez proposto no contexto de OC por Gilmore e Gomory (1961) para resolver uma relaxação linear do Problema de Corte de Materiais (PCM), conhecido em inglês como *Cutting-Stock Problem*. No entanto, esta relaxação proposta por Gilmore e Gomory tinha um número exponencial de variáveis e não podia ser tratada diretamente pelo algoritmo do Simplex.

Problemas de Programação Linear (PL) com um grande número de variáveis tornam o algoritmo do Simplex muito lento, já que, a cada iteração deste algoritmo, todas as colunas associadas às variáveis não-básicas são avaliadas para determinar aquela que entrará na base. Ao invés disso, em uma abordagem de solução via GC, o problema é reformulado através de um método de decomposição e suas colunas são geradas implicitamente por meio da resolução de um subproblema.

A ideia de trabalhar implicitamente com as variáveis de um problema de PL foi sugerida por Ford e Fulkerson (1958) para tratar do Problema de Fluxo de Vários Produtos (*Multi-Commodity Flow Problem*). Esta ideia levou Dantzig e Wolfe (1960) a propor uma técnica de decomposição de problemas de PL, onde um problema é dividido em duas ou mais partes, permitindo trabalhar implicitamente com as variáveis, de maneira que as colunas são geradas quando necessário ao longo do processo de solução. Esta técnica é conhecida como Decomposição de Dantzig-Wolfe (Dantzig e Wolfe, 1960).

O restante do capítulo está organizado da seguinte maneira: a Seção 2.1 ilustra o princípio da Decomposição de Dantzig-Wolfe para PL. A Seção 2.2 trata do método de

GC para resolver problemas de PL obtidos por Decomposição de Dantzig-Wolfe. Por fim, a Seção 2.3 mostra como o método de GC pode ser utilizado para a solução de problemas de Programação Linear Inteira (PLI).

2.1 Decomposição de Dantzig-Wolfe

A utilização de métodos de decomposição como técnica de solução de um problema consiste em dividí-lo em um problema principal (Problema Mestre) e problemas menores (subproblemas) que são resolvidos separadamente. A ideia central está baseada em utilizar métodos eficientes para resolver os subproblemas e combinar os resultados para encontrar uma solução válida do Problema Mestre (PM). A principal motivação para o uso de métodos de decomposição está em tornar possível a resolução de problemas de larga escala onde as técnicas convencionais de solução se tornam ineficientes.

Considere um problema de PL da seguinte forma

$$(PL_1)\ Min\ z = c^T x \tag{2.1}$$

Sujeito a

$$Ax \leq b \tag{2.2}$$

$$Dx \leq d \tag{2.3}$$

$$x \geq 0, \tag{2.4}$$

onde $x \in \mathbb{R}^n_+$ representa um n-vetor de variáveis de decisão não negativas e $c \in \mathbb{R}^n$ é o n-vetor de coeficientes da função objetivo (2.1). As matrizes $A \in \mathbb{R}^{p \times n}$ e $D \in \mathbb{R}^{q \times n}$ contêm os valores dos coeficientes das restrições (2.2) e (2.3), respectivamente. Os vetores $b \in \mathbb{R}^p$ e $d \in \mathbb{R}^q$ contêm os valores do lado direito das restrições.

Sendo P o poliedro definido por (2.3) e (2.4), tem-se que:

$$P = \{Dx \leq d,\ x \geq 0\}. \tag{2.5}$$

Logo, é possível afirmar que o problema PL_1 é equivalente a:

$$(PL_2)\ Min\ z = c^T x \tag{2.6}$$

Sujeito a

$$Ax \leq b \tag{2.7}$$

$$x \in P. \tag{2.8}$$

Supondo que o poliedro P seja limitado, o seguinte teorema é verdadeiro (ver Dantzig e Thapa (2003)):

Teorema 2.1. *Toda solução viável de um conjunto poliedral convexo que tem a forma* $Dx \leq d, x \geq 0$ *pode ser representada como uma combinação convexa do conjunto finito* R *de seus pontos extremos.*

Isto posto, é possível escrever uma solução $x \in P$ como combinação convexa dos seus pontos extremos da forma que segue:

$$x = \sum_{r \in R} r\lambda_r \tag{2.9}$$

$$\sum_{r \in R} \lambda_r = 1 \tag{2.10}$$

$$\lambda \geq 0. \tag{2.11}$$

Definição 2.1. *Substituindo (2.9) - (2.11) em (2.1) - (2.2) transforma-se o problema original* PL_1 *que possui* $p + q$ *linhas em um problema de PL equivalente com* $p + 1$ *linhas e possivelmente um número bem maior de colunas (*$|R|$*), onde* p *é o número de linhas da matriz* A*,* q *é o número de linhas da matriz* D*, e* R *é o conjunto de pontos extremos no poliedro* P*. Este problema é definido a seguir.*

$$(MP)\ Min\ z = \sum_{r \in R} (c^T r)\lambda_r \tag{2.12}$$

Sujeito a

$$\sum_{r \in R} (Ar)\lambda_r \leq b \tag{2.13}$$

$$\sum_{r \in R} \lambda_r = 1 \tag{2.14}$$

$$\lambda \geq 0. \tag{2.15}$$

Vale notar que as linhas (2.13) de MP têm uma relação de correspondência um-para-um com as linhas definidas por (2.2).

Teorema 2.2. *Qualquer λ que satisfaz (2.13) - (2.15) determina um vetor x através de (2.9) que é uma solução válida para o problema PL_1. Se, além disso, λ^* é ótimo para PM, então, por (2.9) é possível gerar uma solução ótima viável x^* para PL_1. Se esta solução ótima para MP não é única, então x^* não precisa ser uma solução básica viável para o problema original PL_1.*

De acordo com o que foi exposto, mostra-se que é possível reformular um problema de PL da forma dada por (2.1) - (2.4) aplicando a Decomposição de Dantzig-Wolfe para obter um problema equivalente da forma (2.12) - (2.15) com dimensão menor em termos de linhas. Porém, esta reformulação resulta em um número de colunas que, em geral, é muito maior que o encontrado no problema original. Isso se dá pelo fato de que tais colunas correspondem a todos os $|R|$ pontos extremos do poliedro P.

Em termos práticos, o uso direto da formulação obtida através da Decomposição de Dantzig-Wolfe com todas as $|R|$ colunas é proibitivo e não implica em vantagem se comparado com o uso da formulação original. Para superar esta desvantagem, ao invés de manter todo o conjunto R de colunas em MP e escolher a melhor delas para entrar na base a cada iteração do algoritmo do Simplex, considera-se apenas um subconjunto "interessante" $\bar{R} \subset R$ das colunas de MP em um PL conhecido como Problema Mestre Restrito (PMR). As demais colunas podem ser tratadas implicitamente através do método de GC, descrito na seção a seguir.

2.2 Geração de Colunas

Ao passo que a Decomposição de Dantzig-Wolfe é um procedimento para reformulação de problemas, as técnicas de GC, baseadas na ideia proposta por Ford e Fulkerson (1958) de tratar as variáveis de um problema de PL implicitamente, emergiram como um dos métodos mais eficientes para resolver problemas com um grande número de variáveis. Esta característica é observada no caso dos modelos resultantes da aplicação da Decomposição de Dantzig-Wolfe, que, em geral, dependem da enumeração completa de todas as soluções possíveis de um ou mais subproblemas.

Um algoritmo de GC foi posto em prática pela primeira vez por Gilmore e Gomory (1961, 1963) como parte de um algoritmo heurístico para resolver o PCM. Ao longo dos anos, algoritmos envolvendo GC foram empregados com sucesso a uma grande variedade de aplicações, tais como: Problemas de Roteamento de Veículos (Desrosiers et al, 1984; Skitt Reuven e Robin, 1985; Agarwal et al, 1989; Desrochers et al, 1992; Pessoa et al, 2008; Moreno et al, 2010; Bettinelli et al, 2010), Problemas de Corte e de Empacotamento (Valério de Carvalho, 1999; Vanderbeck, 1999; Valério de Carvalho, 2005; Belov et al, 2005; Belov e Scheithauer, 2006; Alves e Valério de Carvalho, 2007, 2008), Problema de Atribuição Generalizada (Savelsbergh, 1997; Pigatti et al, 2005), Problema de Dimensionamento de Lotes (Cattrysse et al, 1993; Vanderbeck, 1994, 1998) entre outros.

A estrutura de um algoritmo de GC é composta de duas partes que interagem entre si: um PMR, onde apenas um subconjunto das colunas do PM é considerado, e um conjunto de subproblemas independentes. O PMR atua em um nível superior, fornecendo as informações necessárias para que os subproblemas sejam resolvidos e encontrem as melhores colunas não básicas.

De acordo com Wilhelm (2001), os algoritmos de GC podem ser classificados em três tipos. No primeiro, um grande conjunto de colunas é gerado e adicionado ao PMR, que é então resolvido. Nesse caso, as colunas são adicionadas ao PMR uma única vez. No segundo grupo estão os casos em que há uma interação entre o PMR e os subproblemas, onde as colunas são adicionadas de forma iterativa. Porém, estes casos não se baseiam na aplicação da Decomposição de Dantzig-Wolfe a uma formulação compacta do problema. Por fim, no terceiro grupo estão os métodos que se baseiam na Decomposição de Dantzig-Wolfe e na interação entre o PMR e os subproblemas.

Considerando os algoritmos de GC pertencentes ao terceiro grupo, a aplicação

a problemas de PL da forma de PL_1 após a obtenção do problema MP por meio da Decomposição de Dantzig-Wolfe se dá como segue: a cada iteração toma-se um PMR com um conjunto $R' \subset R$ das colunas de MP. Além disso, sejam π um vetor de dimensão p que contém as variáveis duais associadas às restrições (2.13) e ν um vetor unitário contendo a variável dual da restrição (2.14).

Antes de começar o processo de solução de um problema de PL por GC, o PMR deve ser inicializado com uma base viável. A criação de variáveis artificiais com custo alto M que garantam viabilidade é uma maneira simples de inicialização do PMR. Outra forma de inicialização é fornecer uma solução heurística viável para o PMR. A vantagem consiste em não ser necessário estimar o valor de M, além de resultar em um limite primal mais próximo do ótimo. Vanderbeck (2005) sugere que pode valer a pena realizar um esforço maior na inicialização do PMR, já que, se realizada de forma inteligente, pode resultar no que o autor chama de *warm start*. Detalhes adicionais sobre inicialização do PMR podem ser encontrados em Vanderbeck (1994). Tendo em mãos um PMR inicializado, este é resolvido e com isso os valores das variáveis duais π e ν são conhecidos.

Agora, o próximo passo, conhecido como *pricing*, consiste em utilizar os valores das variáveis duais obtidos através da solução do PMR para encontrar uma nova coluna que entrará em sua base. Este passo pode ser executado de forma eficiente através da solução do seguinte problema de PL, chamado de subproblema de *pricing*:

$$(SP)\ Min\ z = (c - \pi A)x - \nu \tag{2.16}$$

Sujeito a

$$Dx \leq d \tag{2.17}$$

$$x \geq 0. \tag{2.18}$$

Se a solução ótima z^* de SP for negativa, então o vetor x^* corresponde a um ponto extremo do poliedro P.

A variável do PM associada a tal ponto extremo tem custo reduzido negativo e poderá entrar na base. Em seguida, o PMR é atualizado com a adição desta variável e novamente resolvido, o que resulta em novos valores das variáveis duais e resolve-se mais

uma vez o subproblema de *pricing*. Este procedimento deve ser repetido iterativamente enquanto a solução encontrada no *pricing* for negativa. Se a solução z^* de SP for zero, tem-se que não há mais colunas que possam entrar na base e a solução atual do PM é ótima. Geralmente, ao final do algoritmo de GC, o número de variáveis no PMR ainda é muito menor que o do PM.

Uma interessante interpretação econômica acerca da Decomposição de Dantzig-Wolfe e o processo de solução via GC pode ser encontrada em Lasdon (2002), considerando o método como um sistema descentralizado de tomada de decisão.

2.3 Geração de Colunas para Programação Inteira

Considere agora o caso em que se deseja tratar um problema da forma de PL_1, porém com a restrição adicional de que as variáveis devem assumir valores inteiros. Tais problemas, que aparecem com frequência em situações práticas, são chamados de problemas de PLI e assumem a forma:

$$(PI_1)\ Min\ z = c^T x \tag{2.19}$$

Sujeito a

$$Ax \leq b \tag{2.20}$$

$$Dx \leq d \tag{2.21}$$

$$x \in \mathbb{Z}_+^n. \tag{2.22}$$

O poliedro P definido pelas restrições (2.21) e (2.22) é dado por:

$$P = \{Dx \leq d,\ x \in \mathbb{Z}_+^n\} \tag{2.23}$$

e o problema PI_1 pode ser reescrito da seguinte forma:

$$(PI_2)\ Min\ z = c^T x \tag{2.24}$$

Sujeito a

$$Ax \leq b \tag{2.25}$$

$$x \in P \tag{2.26}$$

Ao reformular um problema de PLI através da Decomposição de Dantzig-Wolfe, é necessário utilizar o conceito de discretização (Johnson, 1989; Vanderbeck, 2000). A discretização é análoga ao que é afirmado pelo Teorema 2.1 com a integralidade sendo considerada e é baseada no seguinte teorema:

Teorema 2.3. *Assuma que* $P = \{Dx \leq d,\ x \in \mathbb{Z}_+^n\}$ *é finito e não vazio. Cada ponto* $x \in P$ *pode ser escrito como uma combinação convexa inteira dos pontos em* P*, dada por* $x = \sum_{r \in P} r\lambda_r, \sum_{r \in P} \lambda_r = 1, \lambda \in \mathbb{Z}_+^{|P|}\}$*. Nota-se que tal combinação é trivial, já que se* $x = r$*, tem-se que* $\lambda_r = 1$*.*

Substituindo o resultado do Teorema 2.3 em (2.19) - (2.20) resulta no seguinte PM inteiro:

$$(MPI)\ Min\ z = \sum_{r \in P} (c^T r)\lambda_r \tag{2.27}$$

Sujeito a

$$\sum_{r \in P} (Ar)\lambda_r \leq b \tag{2.28}$$

$$\sum_{r \in P} \lambda_r = 1 \tag{2.29}$$

$$\lambda \in \mathbb{Z}_+. \tag{2.30}$$

A solução do problema MPI pode ser encontrada utilizando um procedimento de *branch-and-bound* baseado na sua relaxação linear, dada por:

$$(RLMPI)\ Min\ z = \sum_{r \in P} (c^T r)\lambda_r \tag{2.31}$$

Sujeito a

$$\sum_{r \in P} (Ar)\lambda_r \leq b \tag{2.32}$$

$$\sum_{r \in P} \lambda_r = 1 \tag{2.33}$$

$$\lambda \geq 0. \tag{2.34}$$

Tipicamente, o valor do limite dual obtido pela resolução de (2.31) - (2.34) via GC é mais forte do que o encontrado por meio da relaxação linear do problema original. Isto acontece porque a integralidade não é relaxada no subproblema de *pricing*, que tem a forma:

$$(SPI)\ Min\ z = (c - \pi A)x - \nu \tag{2.35}$$

Sujeito a

$$Dx \leq d \tag{2.36}$$

$$x \in \mathbb{Z}^n_+, \tag{2.37}$$

onde π e ν são os valores das variáveis duais associadas às restrições (2.32) e (2.33), respectivamente.

Para efeito ilustrativo, a comparação entre a relaxação linear do problema original e a do problema reformulado é exemplificada pelas figuras 2.1 e 2.2 a seguir:

A Figura 2.1 mostra em linha contínua a envoltória convexa de um problema de PLI da forma de PI_1. Além disso, em 2.1a também são representadas separadamente as restrições (2.20)(linha tracejada) e (2.21)(linha pontilhada) e, em 2.1b, o poliedro definido pela relaxação linear da formulação PI_1, ou seja, a interseção de tais restrições.

Já na Figura 2.2 tem-se, além da envoltória convexa do problema, em 2.2a as restrições (2.25) e a envoltória convexa do poliedro dado por (2.23). A interseção desses dois últimos resulta no poliedro definido pela formulação $RLMPI$. Vale notar que o poliedro definido pela relaxação linear da reformulação do problema se aproxima mais da envoltória convexa deste do que o descrito pela relaxação linear da formulação original.

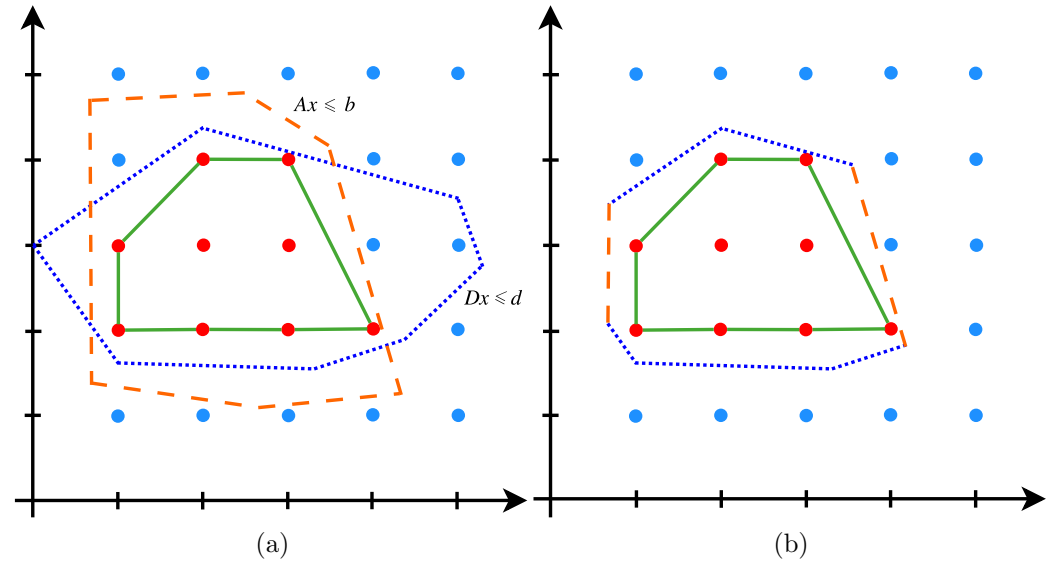

Figura 2.1: Envoltória convexa e relaxação linear da formulação para o problema original

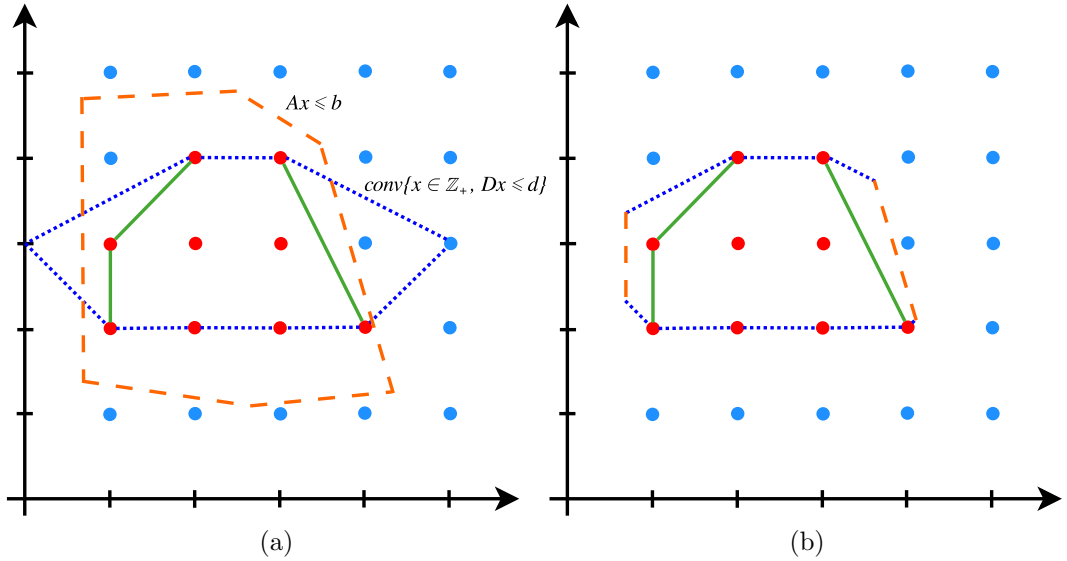

Figura 2.2: Envoltória convexa e relaxação linear da reformulação para o problema original

Isto mostra porque o limite dual de $RLMPI$ obtido por GC é, em geral, melhor do que o que se encontra por meio da relaxação linear da formulação original.

Da Figura 2.2 pode-se concluir também que a reformulação do problema equivale à convexificação das restrições (2.21) de PI_1. Portanto, $RLMPI$ é equivalente a:

$$(RLMPI')\ Min\ z = c^T x \tag{2.38}$$

Sujeito a

$$Ax \leq b \tag{2.39}$$

$$x \in conv\{Dx \leq d, x \in \mathbb{Z}_+^n\} \tag{2.40}$$

$$x \geq 0. \tag{2.41}$$

Ou seja, as soluções da relaxação $RLMPI$ são necessariamente combinações convexas dos pontos do conjunto finito P e as colunas são implicitamente geradas considerando todos esses pontos. Como o subproblema de *pricing* permanece sendo um problema de PLI (possivelmente NP-difícil), a abordagem de GC para tratar problemas inteiros só se justifica quando é possível resolver os subproblemas de forma eficiente a cada iteração.

Para se encontrar a solução inteira ótima do PLI e não apenas um limite dual, pode-se usar um algoritmo de *branch-and-bound* em que cada nó obtém um limite dual resolvendo um problema do tipo $RLMPI$ via GC. Uma ramificação (*branching*) deve ser efetuada se este limite dual não for inteiro nem igual ao limite primal corrente. As restrições que definem cada um dos ramos devem ser adicionadas ao sistema $Ax \leq b$.

Capítulo 3

Problemas de Corte de Materiais e de Empacotamento em Caixas

Os problemas de corte de materiais (*cutting-stock problems*) e de empacotamento em caixas (*bin-packing problems*) estão entre os mais importantes na área de OC tanto pela relevância teórica (ver Sweeney e Paternoster, 1992; Bischoff e Wäscher, 1995; Wang e Wäscher, 2002; Oliveira e Wäscher, 2007) quanto pelo fato de serem problemas com diversas aplicações práticas. Podem ser citadas aplicações destes problemas na indústria (Arenales et al, 2007), em logística (Ertek e Kilic, 2006), em ciência da computação (Chung et al, 2006; Petrucci et al, 2010) e outras (Malkevitch, 2004).

Devido à grande quantidade de situações práticas que podem ser tratadas como problemas de corte ou de empacotamento e também ao grande interesse em pesquisá-los, muitas variantes surgiram na literatura. Com o objetivo de unificar a nomenclatura dos problemas de corte e empacotamento, uma tipologia foi proposta por Dyckhoff (1990). Posteriormente, uma nova tipologia foi apresentada por Wäscher et al (2007) baseada na anterior, com a introdução novos critérios de classificação além da correção de algumas deficiências.

De acordo com Wäscher et al (2007), os problemas de corte e empacotamento têm em comum uma estrutura básica, onde são dados um conjunto de objetos grandes e um conjuntos de objetos pequenos. Nos problemas de corte, os objetos grandes são as matérias-primas, as quais possuem tamanhos padronizados e devem ser cortadas para obter os objetos pequenos, ou seja, o corte das matérias-primas deve resultar em itens com dimensões variadas e não padronizadas que dependem das necessidades dos clien-

tes. Cada matéria-prima tem um custo associado e o objetivo é atender as demandas dos clientes minimizando a quantidade de matéria-prima utilizada. Nos problemas de empacotamento, todos os objetos pequenos, usualmente chamados de itens, devem ser empacotados nos objetos grandes, geralmente chamados de contêineres ou caixas, com o objetivo de minimizar o número de caixas utilizadas.

Matematicamente, o PCM consiste em determinar o menor número de matérias-primas de largura W que devem ser cortadas de modo a obter itens de n tamanhos diferentes com demandas por b_i quantidades inteiras positivas de itens com tamanhos w_i inteiros positivos, $i = 1, 2, ..., n$. A formulação a seguir para o PCM foi proposta por Kantorovich (1960), onde K é um limite superior que deve ser estimado para o número de matérias-primas que deverão ser cortadas.

$$Min \sum_{k=1}^{K} y_k \tag{3.1}$$

Sujeito a

$$\sum_{k=1}^{K} x_{ik} = b_i, \qquad i = 1, ..., n \tag{3.2}$$

$$\sum_{i=1}^{n} w_i x_{ik} \leq W y_k, \qquad k = 1, ..., K \tag{3.3}$$

$$x_{ik} \in \mathbb{Z}_+, \qquad i = 1, ..., n; k = 1, ..., K \tag{3.4}$$

$$y_k \in \{0, 1\}, \qquad k = 1, ..., K. \tag{3.5}$$

Nesta formulação, uma variável $y_k = 1$ se a matéria-prima k for utilizada e 0 caso contrário. As variáveis x_{ik} indicam o número de vezes que um item de tamanho w_i é obtido pelo corte de uma matéria-prima k. A função objetivo (3.1) indica que o número de matérias-primas cortadas deve ser minimizado. As restrições (3.2) garantem que as demandas pelos itens i sejam atendidas. As restrições (3.3) evitam que a largura da matéria-prima k seja respeitada. Por fim, as restrições (3.4) e (3.5) estão relacionadas aos domínios das variáveis x_{ik} e y_k, respectivamente. Para o Problema de Empacotamento em Caixas (PEC), esta formulação também é válida. No entanto, tem-se que $b_i = 1$ para $i = 1, ..., n$ e $x_{ik} \in \{0, 1\}$ para $i = 1, ..., n$ e $k = 1, ..., K$.

Similarmente ao que ocorre em diversos problemas de OC, a formulação de PLI para o PCM e o PEC proposta por Kantorovich (1960) tem seu uso direto inviabilizado para tratar situações em que as quantidades de matérias-primas e itens demandados assumem valores razoáveis. Isto ocorre devido ao fato de que esta formulação não é capaz de obter limites duais fortes e ainda por conter bastante simetria em suas variáveis. Entretanto esta formulação pode ser vista como o ponto de partida para tratar o problema através da aplicação da Decomposição de Dantzig-Wolfe, obtendo uma formulação que pode ser resolvida por algoritmos de GC. Esta abordagem foi proposta por Gilmore e Gomory (1961, 1963) e a formulação tem a forma a seguir.

$$Min \sum_{k=1}^{q} \lambda_k \tag{3.6}$$

Sujeito a

$$\sum_{k=1}^{q} a_{ik}\lambda_k = b_i, \qquad i = 1, ..., n \tag{3.7}$$

$$\lambda_k \in \mathbb{Z}_+, \qquad k = 1, ..., q, \tag{3.8}$$

onde uma variável λ_k é definida para cada padrão de corte viável possível de uma matéria-prima k. Estes padrões de corte são indexados de 1 a q e as constantes a_{ik} representam o número de vezes que um item i aparece em um padrão k. Outras formulações para estes problemas podem ser encontradas na literatura e algumas delas podem ser vistas em Valério de Carvalho (2002).

Nos problemas clássicos de corte e de empacotamento, a qualidade de uma solução é medida pela quantidade de matérias-primas/caixas utilizadas, devido ao tamanho W ser fixo. Relaxando esta condição de tamanho fixo das caixas, Friesen e Langston (1986) propuseram o Problema de Empacotamento em Caixas de Tamanho Variável (PECTV), ou *Variable Sized Bin-Packing Problem* em inglês, que é uma generalização do PEC. No PECTV tem-se m tipos diferentes de caixas com capacidades inteiras W_k com uma quantidade ilimitada de caixas de cada tamanho. O objetivo é empacotar todos os itens de forma que a soma dos tamanhos das caixas utilizadas seja minimizada. Uma formulação de PLI para o problema é mostrada a seguir.

$$Min \sum_{j=1}^{U_k} \sum_{k=1}^{m} W_k y_{jk} \tag{3.9}$$

Sujeito a

$$\sum_{j=1}^{U_k} \sum_{k=1}^{m} x_{ijk} = 1, \qquad i = 1, ..., n \tag{3.10}$$

$$\sum_{i=1}^{n} w_i x_{ijk} \leq W_k y_{jk}, \qquad j = 1, ..., U_k; k = 1, ..., m \tag{3.11}$$

$$x_{ijk} \in \{0, 1\}, \qquad i = 1, ..., n; j = 1, ..., U_k; k = 1, ..., m \tag{3.12}$$

$$y_{jk} \in \{0, 1\}, \qquad j = 1, ..., U_k; k = 1, ..., m, \tag{3.13}$$

onde U_k é um limite superior para o número de caixas de cada tipo k necessárias para empacotar todos os itens. Uma variável $x_{ijk} = 1$ se um item i for empacotado em uma caixa j do tipo k e 0 caso contrário. As variáveis binárias y_{jk} indicam se uma caixa j do tipo k é utilizada no empacotamento ou não. A função objetivo (3.9) busca minimizar a soma das capacidades das caixas utilizadas. As restrições (3.10) garantem que todos os itens sejam empacotados. As restrições (3.11) impedem que as capacidades das caixas sejam ultrapassadas. As restrições (3.12) e (3.13) definem os domínios das variáveis do problema.

Analogamente ao que ocorre com o PEC, é possível relaxar a restrição de que a largura da matéria-prima deve ter um valor fixo no PCM. Desta forma, define-se o Problema de Corte de Materiais com Múltiplas Larguras (PCMML), conhecido em inglês por *Multiple Length Cutting-Stock Problem*, com uma formulação que pode ser escrita substituindo a restrição (3.10) por:

$$\sum_{j=1}^{U_k} \sum_{k=1}^{m} x_{ijk} = b_i, \qquad i = 1, ..., n, \tag{3.14}$$

onde b_i é a quantidade demandada de um item i. Estas formulações para o PECTV e PCMML, assim como para os problemas clássicos de corte e empacotamento, têm seu uso direto proibitivo em instâncias de tamanho razoável de modo que uma mesma solução

pode ser representada de diversas formas diferentes.

Também vale mencionar que estes quatro problemas apresentados pertencem à classe de problemas que são fortemente NP-difíceis e despertam grande interesse na investigação e desenvolvimento de algoritmos eficientes para tratá-los, bem como suas variantes. Entre as abordagens, podem ser mencionadas as que envolvem algoritmos aproximativos para obtenção de limites duais (Fekete e Schepers, 2001; Khanafer et al, 2010), os algoritmos heurísticos, onde uma extensa revisão pode ser encontrada em Coffman Jr et al (1996), e também as abordagens exatas baseadas em algoritmos de PLI (Vanderbeck, 1999; Valério de Carvalho, 1999; Belov e Scheithauer, 2002; Monaci, 2003; Belov et al, 2005; Belov e Scheithauer, 2006; Alves e Valério de Carvalho, 2007, 2008).

Como será mostrado no capítulo seguinte, o PCCVH pode ser visto como uma generalização do PECTV observando a equivalência de conceitos entre os itens e os serviços, os tamanhos dos itens e as demandas dos serviços, os tipos de caixas e os tipos de servidores, bem como suas capacidades.

Capítulo 4

O Problema de Configuração de *Clusters* Virtualizados Heterogêneos

Considere uma situação em que um conjunto de serviços deve rodar em um *cluster* computacional virtualizado. O nome "virtualizado" se refere a uma forte tendência da computação moderna, onde máquinas virtuais são executadas com o propósito de emular uma máquina física rodando um determinado sistema operacional. Isto torna possível a execução de uma grande variedade de aplicativos implementados e compatíveis com sistemas operacionais distintos.

Tais aplicativos são desenvolvidos para realizar tarefas ou prestar serviços baseados na Internet e, por isso, são hospedados e executados em grandes *clusters* mantidos por organizações públicas ou privadas, nestes últimos casos cobrando-se taxas. Os usuários que requisitam os serviços não precisam conhecer, e tipicamente não conhecem, a localização física dos servidores que formam o *cluster*. Estes podem até estar divididos em mais de um local (o que de certa forma contradiz o nome *cluster*), uma vez que toda a comunicação entre o *cluster*, o usuário e os aplicativos é feita através da Internet. O único interesse dos usuários é que o *cluster* garanta a qualidade de serviço, atendendo a demanda por processamento acordada ou contratada. O emprego de *clusters* virtualizados para prestar serviços baseados em aplicações *web* faz parte do que hoje é também conhecido por "computação em nuvem".

Um *cluster* computacional, virtualizado ou não, geralmente é formado por servidores (unidades de processamento) heterogêneos, ou seja, com características potencialmente diferentes. A principal característica de um servidor é o seu tipo de CPU. Cada tipo de

CPU possui uma capacidade de processamento e um consumo energético padrões, assumindo a operação na sua frequência nominal. Entretanto, as modernas CPUs podem operar em algumas frequências que não a frequência nominal. Essas frequências alternativas, determinadas pelo fabricante, podem ser menores (*underclocking*) ou até maiores (*overclocking*) do que a nominal. A capacidade de processamento de uma CPU é linearmente proporcional à frequência. O consumo energético da CPU também cresce com a frequência, porém esse crescimento segue uma curva não linear e não convexa. Tipicamente essa curva começa com um crescimento sublinear, a partir de certo ponto existe uma inflexão e nas frequências mais altas o crescimento passa a ser aproximadamente quadrático.

Para decidir em qual frequência de CPU um servidor do *cluster* deve operar, as abordagens propostas se baseiam na técnica de DVFS disponível nos microprocessadores atuais. Esta técnica permite o ajuste dinâmico da performance de CPU por meio da combinação de um conjunto de frequências e tensões pré-definidas, resultando em estados de performance, ou *P-states*. Estes estados são mostrados na Tabela 4.1 junto com suas respectivas frequências, tensões e consumos de energia para um processador Intel Pentium M 1,6 GHz, cujos dados são disponibilizados pela própria fabricante[1].

Tabela 4.1: Exemplos de estados de operação e consumos de energia para diferentes frequências em um processador Intel Pentium M 1,6 GHz

P-state	Frequência (GHz)	Tensão (V)	Consumo (Watts)
P0	1,6	1,484	25
P1	1,4	1,420	17
P2	1,2	1,276	13
P3	1,0	1,164	10
P4	0,8	1,036	8
P5	0,6	0,956	6

A Figura 4.1 ilustra para este mesmo processador a curva que representa o consumo energético com relação a cada frequência disponível de acordo com os dados da Tabela 4.1.

Cada serviço que deve rodar no *cluster* se caracteriza pela sua demanda por processamento. Essas demandas costumam ser razoavelmente menores do que a capacidade de um servidor, de forma que o mesmo servidor pode rodar alguns serviços ao mesmo

[1]Enhanced Intel SpeedStep Technology for the Intel Pentium M Processor: `ftp://download.intel.com/design/network/papers/30117401.pdf`

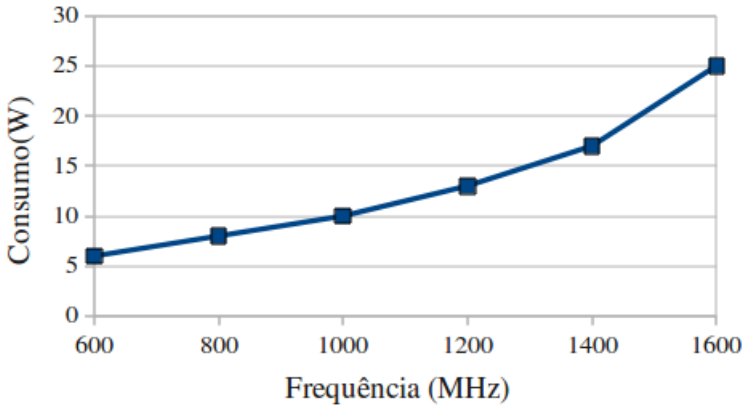

Figura 4.1: Frequências e consumo de energia para um processador Intel Pentium M 1,6 GHz

tempo. Naturalmente, a soma das demandas dos serviços atribuídas a um servidor não deve exceder a sua capacidade de processamento na frequência escolhida, para não prejudicar a qualidade de serviço. Quando nenhum serviço for atribuído a um determinado servidor, considera-se que o servidor estará desligado e seu consumo energético será nulo.

Uma *configuração* de um *cluster* consiste em determinar: (i) quais servidores estarão ligados; (ii) em quais frequências suas CPUs devem operar e (iii) quais serviços devem ser atribuídos a cada servidor ligado. No PCCVH busca-se encontrar uma configuração capaz de atender a demanda por processamento de um dado conjunto de serviços com um consumo energético total mínimo.

As diferenças entre um *cluster* virtualizado e um *cluster* tradicional, um conjunto de máquinas operando num único local conhecido pelos proprietários dos serviços, incluem:

- Um *cluster* virtualizado costuma ter maior porte. Isso significa que um algoritmo adequado para o PCCVH deve ser capaz de ser aplicado em instâncias com vários milhares de servidores e dezenas de milhares de serviços.

- O conjunto de serviços que rodam em um *cluster* virtualizado costuma variar de forma mais dinâmica, com aplicações começando, terminando ou mudando sua demanda por processamento. É razoável que se façam reconfigurações de um *cluster* virtualizado com maior frequência, por exemplo, a cada dez minutos. Nesse exemplo, um algoritmo adequado para o PCCVH deve ser capaz de fornecer soluções em no máximo alguns segundos.

Isto quer dizer que os algoritmos que serão propostos também se aplicam a *clusters*

tradicionais, supostamente menores e menos dinâmicos.

Algumas abordagens de otimização, baseadas no PEC, são encontradas na literatura para tratar da configuração de *clusters* de servidores virtualizados. Bichler et al (2006) trata da atribuição de máquinas virtuais com foco no planejamento da capacidade, enquanto Khanna et al (2006) buscam minimizar os custos de migração de serviços entre os servidores de forma que a performance seja garantida. No entanto, estes trabalhos não abordam a questão do consumo energético.

Wang et al (2008) apresentam uma arquitetura de controle para garantia de eficiência energética em tempo real para ambientes virtualizados, mas não consideram mecanismos de liga/desliga nem a alocação das máquinas virtuais. Um procedimento heurístico de solução é apresentado por Srikantaiah et al (2008) para consolidação de serviços em servidores virtualizados buscando economia de energia. Entretanto, a técnica de DVFS não é considerada e o algoritmo não apresenta garantias de que as soluções obtidas sejam ao menos próximas do ótimo.

Abordagens que consideram tanto DVFS quanto mecanismos de liga/desliga podem ser encontradas em Elnozahy et al (2003); Sharma et al (2003); Kandasamy et al (2004); Rusu et al (2006); Bertini et al (2007); Khargharia et al (2008) com o objetivo de minimizar o consumo energético, mas não são aplicáveis a *clusters* de servidores virtualizados e, portanto, não tratam da consolidação de serviços nos servidores. Uma revisão de trabalhos que abordam a racionalização do consumo de energia em ambientes de computação em nuvem é apresentada em Berl et al (2010).

Neste trabalho, são apresentadas abordagens baseadas em técnicas de otimização que se diferenciam das demais por levar em conta a configuração de *clusters* considerando de forma simultânea técnicas de virtualização, DVFS, heterogeneidade dos servidores e mecanismos de liga/desliga de forma a obter a consolidação dos serviços com o menor consumo de energia elétrica.

Matematicamente, o PCCVH pode ser definido da seguinte forma: sejam o conjunto $N = \{1, 2, ..., i, ..., n\}$ dos tipos de servidores no *cluster*, $S_i = \{1, 2, ..., s, ..., q_i\}$ o conjunto dos servidores do tipo $i \in N$, o conjunto $F_i = \{1, 2, ..., j, ..., r_i\}$ das frequências de CPU em que cada servidor do tipo $i \in N$ pode operar, e $K = \{1, 2, ..., k, ..., m\}$ o conjunto dos serviços que devem ser executados no *cluster*. Cada servidor do tipo $i \in N$ operando a uma frequência de CPU $j \in F_i$ tem capacidade D_{ij} (por exemplo, em

requisições/s). O consumo energético (por exemplo, em Watts) para manter um servidor do tipo $i \in N$ operando na frequência $j \in F_i$ é dado por C_{ij}. A demanda (na mesma unidade da capacidade D_{ij}) de um serviço $k \in K$ é representada por d_k.

Desta forma, o objetivo do PCCVH é encontrar uma configuração que garanta a execução de todos os serviços de forma que o consumo de energia elétrica para operação do *cluster* seja minimizado.

4.1 Formulação Matemática

O PCCVH é um problema NP-difícil, desde que pode ser visto como uma generalização do PECTV, onde os serviços seriam os itens que devem ser empacotados, as demandas seriam os tamanhos dos itens e as caixas seriam os servidores com sua capacidade e seu custo determinados pela configuração. A diferença é que aqui um tipo de caixa (servidor) pode assumir um entre r_i, $i \in N$ possíveis tamanhos (capacidades, determinadas pela frequência), o que também vai determinar seu custo. Outra diferença se dá pelo fato de o número de caixas de cada tipo ser limitado.

O PCCVH pode ser modelado como um problema de PLI como descrito na formulação $\mathcal{F}_1$ a seguir que é baseada no modelo proposto por Petrucci et al (2009). As variáveis de decisão são definidas como segue: $y_{isj} = \{0, 1\}$ é uma variável binária que indica se o servidor $s \in S_i$ do tipo $i \in N$ está ligado e operando na frequência $j \in F_i$, e $x_{isjk} = \{0, 1\}$ é uma variável binária que indica se o serviço $k \in K$ é executado no servidor $s \in S_i$ do tipo $i \in N$ que está ligado e operando na frequência de CPU $j \in F_i$.

$$(\mathcal{F}_1)\ z = Min \sum_{i \in N} \sum_{s \in S_i} \sum_{j \in F_i} C_{ij} y_{isj} \tag{4.1}$$

Sujeito a

$$\sum_{k \in K} d_k x_{isjk} \leq D_{ij} y_{isj} \qquad \forall i \in N, s \in S_i, j \in F_i \tag{4.2}$$

$$\sum_{i \in N} \sum_{s \in S_i} \sum_{j \in F_i} x_{isjk} = 1 \qquad \forall k \in K \tag{4.3}$$

$$\sum_{j \in F_i} y_{isj} \leq 1 \qquad \forall i \in N, s \in S_i \tag{4.4}$$

$$x_{isjk} \in \{0, 1\} \qquad \forall i \in N, s \in S_i, j \in F_i, k \in K \tag{4.5}$$

$$y_{isj} \in \{0, 1\} \qquad \forall i \in N, s \in S_i, j \in F_i. \tag{4.6}$$

A função objetivo dada por (4.1) busca encontrar uma configuração do *cluster* com o menor custo de operação. As restrições (4.2) garantem que a capacidade D_{ij} de um servidor $s \in S_i$ do tipo $i \in N$ operando na frequência $j \in F_i$ não seja ultrapassada. As restrições (4.3) impõem que cada serviço $k \in K$ seja executado no *cluster*, enquanto as restrições (4.4) garantem que cada servidor $s \in S_i$ do tipo $i \in N$ pode operar em, no máximo, uma frequência $j \in F_i$. As expressões (4.5) e (4.6) definem a natureza das variáveis x_{isjk} e y_{isj}.

A formulação $\mathcal{F}_1$ possui $\sum_{i \in N} q_i \cdot r_i$ variáveis y, $(\sum_{i \in N} q_i \cdot r_i) \cdot m$ variáveis x, $\sum_{i \in N} q_i \cdot r_i$ restrições do tipo (4.2), m restrições do tipo (4.3) e $\sum_{i \in N} q_i$ restrições do tipo (4.4). Isto faz com que o uso direto de $\mathcal{F}_1$ para determinar a configuração de um *cluster* de grande escala seja impraticável, já que o número de variáveis e restrições se tornará grande demais para ser tratado por um resolvedor de problemas de PLI.

No entanto, com o objetivo de obter apenas um *Lower Bound* (LB) para o consumo energético ótimo, é proposta uma relaxação de PLI para o PCCVH, permitindo que as demandas dos serviços sejam fracionadas em diferentes servidores. Vale notar que o caso da fragmentação dos serviços não é uma suposição válida para o PCCVH.

Seja v_{ij} uma variável inteira que indica quantos servidores do tipo $i \in N$ operam na frequência $j \in F_i$. A relaxação $\mathcal{F}_2$ é baseada na relaxação para o PECTV que pode ser encontrada em Monaci (2003) e Haouari e Serairi (2009) e é escrita da forma que segue.

$$(\mathcal{F}_2) \quad z = Min \sum_{i \in N} \sum_{j \in F_i} C_{ij} v_{ij} \tag{4.7}$$

Sujeito a

$$\sum_{j \in F_i} v_{ij} \leq q_i \qquad \forall i \in N \tag{4.8}$$

$$\sum_{i \in N} \sum_{j \in F_i} D_{ij} v_{ij} \geq \sum_{k \in K} d_k \tag{4.9}$$

$$v_{ij} \in \mathbb{Z}_+ \qquad \forall i \in N, j \in F_i. \tag{4.10}$$

A função objetivo (4.7) minimiza a soma dos custos operacionais. As restrições (4.8) garantem que o número de servidores do tipo $i \in N$ ligados não ultrapasse q_i. As restrições (4.9) garantem que haja capacidade de processamento suficiente para atender a soma das demandas dos serviços. As restrições (4.10) definem o domínio das variáveis de decisão. É possível provar que $z_{\mathcal{F}_2} \leq z_{\mathcal{F}_1}$. Entretanto, ainda que esta formulação não possa ser usada para encontrar uma solução factível para o PCCVH, o custo de sua solução pode ser usado para medir a qualidade de uma dada solução inteira viável para o problema.

4.2 Abordagem de Geração de Colunas

Esta seção descreve a abordagem de solução proposta com o objetivo de encontrar soluções inteiras viáveis através de algoritmos baseados em GC. Como o uso direto da formulação $\mathcal{F}_1$ para obter soluções inteiras viáveis se torna proibitivo quando o número de servidores e serviços cresce, e consequentemente o número de variáveis, uma estratégia interessante é reformular o problema através da Decomposição de Dantzig-Wolfe. Isto permite resolver a relaxação linear da formulação resultante por meio de um algoritmo de GC. Uma abordagem de GC é indicada por tratar as variáveis de forma implícita, reduzindo consideravelmente o número de variáveis que são realmente consideradas na resolução do problema. Para tanto, devem ser consideradas as definições as seguir:

Definição 4.1. *Um padrão é definido como o vetor de incidência em* $\{0,1\}^m$ *associado*

a um subconjunto de K. Para um padrão p, p_k denota o k-ésimo componente de p. Seja $P(i)$ o conjunto de todos os padrões viáveis para um servidor do tipo $i \in N$, ou seja, aqueles tal que

$$\sum_{k \in K} p_k d_k \leq D_{ir_i} \qquad \forall i \in N, \tag{4.11}$$

onde D_{ir_i} é a capacidade máxima de um servidor do tipo $i \in N$ e está associada à sua frequência mais alta.

Definição 4.2. *$f(p, i)$ é a menor frequência tal que:*

$$\sum_{k \in K} p_k d_k \leq D_{if(p,i)}. \tag{4.12}$$

Assim sendo, a reformulação do problema resultante da aplicação da Decomposição de Dantzig-Wolfe é dada a seguir.

$$Min \sum_{i \in N} \sum_{p \in P(i)} C_{if(p,i)} \lambda_{ip} \tag{4.13}$$

Sujeito a

$$\sum_{i \in N} \sum_{p \in P(i)} p_k \lambda_{ip} = 1 \qquad \forall k \in K \tag{4.14}$$

$$\sum_{p \in P(i)} \lambda_{ip} \leq q_i \qquad \forall i \in N \tag{4.15}$$

$$\lambda_{ip} \in \{0, 1\} \qquad \forall i \in N, p \in P(i). \tag{4.16}$$

Nesta formulação, uma variável λ_{ip} é igual a 1 se os serviços no padrão $p \in P(i)$ forem atribuídos a um servidor do tipo $i \in N$ operando na frequência $f(p, i)$.

4.2.1 Agregação dos Serviços por Demanda

A formulação (4.13 - 4.16) obtida pela Decomposição de Dantzig-Wolfe tem menos linhas que a formulação $\mathcal{F}_1$. No entanto, em ambas as formulações, tem-se que as restrições

(4.3) e (4.14) são definidas para cada serviço $k \in K$. Segundo Alves e Valério de Carvalho (2007), em modelos com tais características existe uma solução ótima com variáveis duais iguais para, a exemplo do PEC, itens com o mesmo tamanho. Gilmore e Gomory (1963) também relataram este fato em várias iterações do algoritmo de GC para o PCM.

Para o PCCVH, quando o número de serviços for muito grande, pode ocorrer que muitos destes tenham demandas de processamento iguais ou próximas, fazendo com que as variáveis duais associadas às restrições (4.14) assumam valores iguais ao longo do processo de solução de (4.13 - 4.16) por GC.

Por isso, pode-se usar um esquema para a agregação das linhas de (4.13 - 4.16), onde as restrições (4.14) serão agora definidas para cada demanda diferente e não para cada serviço. Isto equivale a forçar que as variáveis duais dos serviços que possuam a mesma demanda assumam valores iguais. A implicação imediata é a diminuição do número de linhas da formulação para o número de demandas diferentes. Também é possível haver melhoria na convergência do algoritmo de GC quando este esquema de agregação é utilizado.

Isto posto, é definido o conjunto $K' = \{1, 2, ..., k, ..., m'\}$ como o conjunto dos índices dos grupos de serviços agregados e n_k como o número de serviços com demanda d_k. Além disso, assume-se que o conjunto K' está ordenado em ordem decrescente de acordo com os valores das demandas d_k, $k \in K'$. Desta forma, a Definição 4.1 é reescrita da forma que segue.

Definição 4.3. *Um padrão é definido como o vetor de incidência em $\mathbb{Z}^{m'}$ associado a um subconjunto de K'. Para um padrão p, $p_k \leq n_k$ denota o k-ésimo componente de p. Seja $P(i)$ o conjunto de todos os padrões viáveis para um servidor do tipo $i \in N$, ou seja, aqueles tal que*

$$\sum_{k \in K'} p_k d_k \leq D_{ir_i} \qquad \forall i \in N, \tag{4.17}$$

onde D_{ir_i} é a capacidade máxima de um servidor do tipo $i \in N$ e está associada à sua frequência mais alta.

Assim, usando este esquema de agregação de linhas é possível substituir as restrições (4.14) por (4.19). Por outro lado, ao relaxar a integralidade das variáveis λ, obtém-se

o problema mestre $\mathcal{MP}$, definido a seguir.

$$(\mathcal{MP})\ Min \sum_{i \in N} \sum_{p \in P(i)} C_{if(p,i)} \lambda_{ip} \tag{4.18}$$

Sujeito a

$$\sum_{i \in N} \sum_{p \in P(i)} p_k \lambda_{ip} \geq n_k \qquad \forall k \in K' \tag{4.19}$$

$$\sum_{p \in P(i)} \lambda_{ip} \leq q_i \qquad \forall i \in N \tag{4.20}$$

$$\lambda_{ip} \geq 0 \qquad \forall i \in N, p \in P(i). \tag{4.21}$$

Vale notar que as restrições (4.19) também são relaxadas para "maior ou igual", já que isto não altera o LB obtido por $\mathcal{MP}$ e ainda ajuda na aceleração da convergência do algoritmo de GC. Esta formulação também pode ser vista como uma generalização do modelo para GC para o PCMML (Belov e Scheithauer, 2002; Alves e Valério de Carvalho, 2008).

4.2.2 Variáveis de Troca

Além do esquema de agregação de linhas, a convergência do algoritmo de GC pode ser mais acelerada se um pequeno número de *variáveis de troca* for adicionado a $\mathcal{MP}$ resultando em uma formulação estendida para o problema.

As variáveis adicionais $t_k \in \mathbb{Z}_+$ representam a substituição de um certo número de serviços agregados k pela mesma quantidade de serviços agregados $k-1$, $k \in K'$. Estas variáveis sempre representam a troca de um serviço agregado por outro com demanda imediatamente menor que, por sua vez, pode ser trocado da mesma forma por outro e assim por diante. Como resultado, cada variável λ adicionada a $\mathcal{MP}$ que normalmente representaria um padrão único, agora pode ser usada em outras soluções que utilizam padrões diferentes com demandas menores. A inclusão das variáveis de troca é equivalente a adicionar desigualdades ao problema dual, restringindo seu espaço de solução (Valério de Carvalho, 2005). A adição de desigualdades ao problema dual é aplicada por Alves e Valério de Carvalho (2007) ao PECTV, como uma das diferentes estratégias de estabilização

e aceleração de um algoritmo de GC proposto para o problema.

O Problema Mestre Estendido ($\mathcal{MP}_{\text{ex}}$), resultante da adição das variáveis de troca a $\mathcal{MP}$ é escrito da seguinte forma.

$$(\mathcal{MP}_{\text{ex}})\ Min \sum_{i \in N} \sum_{p \in P(i)} C_{if(p,i)} \lambda_{ip} \tag{4.22}$$

Sujeito a

$$\sum_{i \in N} \sum_{p \in P(i)} p_k \lambda_{ip} - t_k + t_{k-1} \geq n_k \qquad \forall k \in K' \tag{4.23}$$

$$\sum_{p \in P(i)} \lambda_{ip} \leq q_i \qquad \forall i \in N \tag{4.24}$$

$$\lambda_{ip} \geq 0 \qquad \forall i \in N, p \in P(i) \tag{4.25}$$

$$t_k \geq 0 \qquad \forall k \in K'. \tag{4.26}$$

4.2.3 Problema Mestre Restrito

Devido ao grande número de colunas em um problema mestre resultante da Decomposição de Dantzig-Wolfe, deve-se trabalhar de forma implícita com tais colunas, sendo esta a ideia fundamental do algoritmo de GC. Desta forma, apenas uma pequena versão do problema mestre, o PMR, com número de colunas limitado e suficiente para garantir viabilidade é tratada. As colunas são geradas por um subproblema e apenas as que forem "interessantes" são adicionadas ao PMR.

4.2.4 Subproblema de *Pricing*

Numa abordagem de GC, as colunas são geradas por um subproblema chamado de Subproblema de *Pricing* e são adicionadas ao PMR apenas quando isto for vantajoso. No caso do PCCVH, uma coluna gerada pelo subproblema só será adicionada ao PMR se o valor da função objetivo (4.27), que representa o custo reduzido da coluna, for menor que zero.

Sejam α_i, $i \in N$, as variáveis duais associadas às restrições (4.20) para $\mathcal{MP}$ e (4.24) para $\mathcal{MP}_{\text{ex}}$. Sejam também π_k, $k \in K'$, as variáveis duais associadas às restrições

(4.19) no caso de $\mathcal{MP}$ e (4.23) no caso de $\mathcal{MP}_{\text{ex}}$. Com base nas Definições 4.2 e 4.3, para cada par (i, j), $i \in N$ e $j \in F_i$, o Subproblema de *Pricing* SP_{ij} pode ser formulado como segue.

$$(\text{SP}_{ij})\ Min\ C_{ij} - \alpha_i - \sum_{k \in K'} \pi_k p_k \tag{4.27}$$

Sujeito a

$$\sum_{k \in K'} d_k p_k \leq D_{ij} \tag{4.28}$$

$$p_k \in \mathbb{Z}_+ \qquad \forall k \in K' \tag{4.29}$$

Como C_{ij} e α_i são constantes para cada SP_{ij}, $i \in N$, $j \in F_i$, o subproblema de *pricing* pode ser visto como um Problema da Mochila. Mesmo sendo NP-difícil, este problema é bem tratável na prática podendo ser resolvido através de algoritmos pseudo-polinomiais de complexidade $O(m'D_{ij})$

A solução encontrada no *pricing* representa uma configuração válida para um dado tipo de servidor $i \in N$ operando numa frequência $j \in F_i$, onde as variáveis p_k indicam quantos serviços com demanda d_k, $k \in K'$, serão atribuídos à configuração.

4.2.5 Base Incial para Geração de Colunas

Para gerar uma base inicial viável para os algoritmos de GC, uma heurística gulosa simples foi desenvolvida (Algoritmo 1). Seja LS uma lista que contém todos os servidores do *cluster* e LA uma lista composta por todos os serviços (linha 3). Primeiramente, LS é ordenada de acordo com as capacidades máximas dos servidores e LA de acordo com as demandas, ambas em ordem decrescente (linha 4). Em seguida, enquanto LA não estiver vazia, deve-se tentar atribuir os serviços restantes à máxima frequência do primeiro elemento corrente de LS, sempre dando prioridade àqueles serviços com maior demanda (linhas 7 - 14). Por fim, as variáveis λ (padrões) associadas à solução completa do problema gerada pela heurística são adicionadas ao PMR (linha 15).

Algoritmo 1 GeraBaseInicial()

```
1: PMR ← ∅
2: sol ← ∅
3: Inicializa a Lista de Servidores (LS) e a Lista de Serviços (LA)
4: Ordena LS e LA em ordem decrescente
5: enquanto LA não estiver vazia faça
6:     Remove o primeiro elemento de LS (ServidorAtual)
7:     enquanto pelo menos um serviço k ∈ LA puder ser atribuído à maior frequência do ServidorAtual
       faça
8:         para k = 1 . . . |LA| faça
9:             Tenta atribuir o elemento k ∈ LA ao ServidorAtual
10:        fim para
11:        Atualiza a solução parcial sol incluindo os serviços atribuídos
12:        Atualiza LA removendo os serviços atribuídos
13:    fim enquanto
14: fim enquanto
15: Atualiza o PMR adicionando as variáveis λ (padrões) associados a sol
16: retorna PMR
```

4.2.6 Algoritmo de Geração de Colunas

O pseudocódigo do algoritmo de GC é apresentado em Algoritmo 2. Assume-se que os resolvedores de PL e de Problemas da Mochila são caixas-pretas e que uma base inicial é fornecida ao PMR pela heurística descrita no Algoritmo 1. Vale ressaltar que, no caso de $\mathcal{MP}_{\text{ex}}$, as variáveis de troca t também devem ser adicionadas ao PMR na inicialização. A variável booleana *colsAdicionadas* é definida como verdadeiro se alguma coluna tiver sido adicionada ao PMR em uma iteração. Caso contrário, *colsAdicionadas* é definida como falso. A primeira iteração do algoritmo se dá pela solução do PMR (linha 3). Em seguida (linhas 4 - 19), enquanto *colsAdicionadas* for verdadeiro, o subproblema SP_{ij} correspondente a cada par (i, j), $i \in N$, $j \in F_i$, é resolvido e a coluna é adicionada ao PMR se o valor da função objetivo do subproblema for menor que zero. Após todas as colunas correspondentes ao par (i, j) terem sido adicionadas, o PMR é resolvido novamente.

4.2.7 Heurísticas para Solução Primal

Esta seção apresenta as duas abordagens propostas neste trabalho para obter soluções inteiras viáveis para o PCCVH. A primeira heurística se baseia na solução de um problema de PLI contendo as colunas ao final da resolução do PMR via GC, enquanto a segunda se baseia na solução de uma sequência de problemas de PL, também por GC.

Algoritmo 2 GeracaoDeColunas(PMR)

```
1: sol_SP ← ∅
2: colsAdicionadas ← verdadeiro
3: sol ← ResolvedorPL(PMR)
4: enquanto colsAdicionadas faça
5:    colsAdicionadas ← falso
6:    para cada tipo de servidor i ∈ N faça
7:       para cada frequência j ∈ F_i faça
8:          Atualiza SP_ij com os valores de α_i e π_k, ∀k ∈ K'
9:          sol_SP ← ResolvedorSP(SP_ij)
10:         se f(sol_SP) < 0 então
11:            Adiciona a coluna associada à variável λ ao PMR
12:            colsAdicionadas ← verdadeiro
13:         fim se
14:      fim para
15:      se colsAdicionadas então
16:         sol ← ResolvedorPL(PMR)
17:      fim se
18:   fim para
19: fim enquanto
20: retorna sol
```

Heurística do Problema Mestre Restrito Inteiro

A Heurística do PMR Inteiro (HPMRI) consiste em redefinir as variáveis do PMR obtido ao final do algoritmo de GC. As variáveis deixam de ser contínuas e agora serão definidas no domínio dos números inteiros não negativos e, em seguida, o PMR com as variáveis inteiras é resolvido. É importante mencionar que, o PMR Inteiro (PMRI) resultante da redefinição das variáveis pode não possuir uma solução inteira viável e, além disso, esta abordagem de solução depende de um resolvedor de problemas de PLI que seja capaz de encontrar soluções ótimas ou próximas da ótima em um tempo computacional aceitável. Também é válido ressaltar que uma solução ótima obtida para o PMRI não necessariamente será ótima para o PCCVH.

Heurística de Arredondamento

Ao contrário da HPMRI, a Heurística de Arredondamento (HA) não se baseia em PLI e, consequentemente, não depende de um resolvedor para este tipo de problema. O objetivo desta heurística é encontrar soluções inteiras viáveis por meio do arredondamento dos valores das variáveis fracionárias para cima em uma solução obtida pelo algoritmo de GC. O pseudocódigo mostrado no Algoritmo 3 contém um esboço de HA com suas três etapas: (i) geração de uma base inicial; (ii) construção de uma solução inteira viável; e (iii) aplicação de um procedimento de refinamento da solução construída.

Algoritmo 3 HeuristicaDeArredondamento()

1: $sol \leftarrow \emptyset$
2: PMR $\leftarrow$ GeraBaseInicial()
3: $sol \leftarrow$ Construcao(PMR)
4: $sol \leftarrow$ Refinamento(PMR, sol)
5: **retorna** sol

O pseudocódigo do procedimento que realiza a construção de uma solução inteira viável é apresentado no Algoritmo 4, onde $i \in \{1, ..., |N|\}$ é o índice associado a um tipo de servidor e, além disso, assume-se que quanto maior for o valor de i, menor será a capacidade de um servidor deste tipo em sua frequência mais alta.

Algoritmo 4 Construcao(PMR)

1: $i \leftarrow |N|$
2: $maxCols \leftarrow 1000$
3: $sol \leftarrow$ GeracaoDeColunas(PMR)
4: **enquanto** sol for fracionária **faça**
5: **se** sol for inviável **ou** não houver mais variáveis λ que possam ser arredondadas para cima **então**
6: Atualiza PMR restaurando os *bounds* originais destas variáveis, i.e., $\lambda \geq 0$, associadas ao tipo de servidor i atualmente fixado
7: $i \leftarrow i - 1$
8: **se** $i = 0$ **então**
9: $i = |N|$
10: **fim se**
11: **senão**
12: Atualiza PMR arredondando para cima a variável fracionária λ_p associada a $g^{min} = \min\{\lceil \lambda_p \rceil - \lambda_p \mid \forall p \in \mathcal{P}\}$
13: **fim se**
14: **se** número de colunas de PMR $> maxCols$ **então**
15: Remove as colunas associadas às variáveis λ que estão fora da base
16: **fim se**
17: $sol \leftarrow$ GeracaoDeColunas(PMR)
18: **fim enquanto**
19: **retorna** sol

Inicialmente, é definido $i = |N|$ (linha 1). O número máximo de colunas permitido durante a fase de construção é denotado por $maxCols$ e, após experimentos preliminares, adotou-se que $maxCols = 1000$ (linha 2). Em seguida, uma solução inicial, provavelmente fracionária, é obtida através do algoritmo de GC (linha 3). Enquanto a solução for fracionária, busca-se gerar uma solução inteira viável modificando a solução encontrada pela GC (linhas 4 - 18). Caso a solução obtida pela GC seja inviável ou não for mais possível arredondar o valor de pelo menos uma variável λ de forma que o atendimento às restrições do problema seja mantido, deve-se desfazer a fixação das variáveis associadas ao servidor i (linhas 5 - 6), ou seja, tais variáveis λ terão seus *bounds* restaurados para $\lambda \geq 0$. Feito isso, o valor de i deve ser decrementado ou, caso $i = 0$, definido como $i = |N|$ (linhas

7 - 10). Se for encontrada uma solução viável fracionária, seleciona-se uma variável para arredondar de acordo com o critério seguinte. Seja $\mathcal{P}$ o conjunto das variáveis fracionárias que possuem valores fracionários em uma solução obtida por GC. Uma variável λ_p, $p \in \mathcal{P}$ é escolhida de acordo com $g^{min} = min\{g(p)|\forall p \in \mathcal{P}\}$, onde $g(p) = \lceil \lambda_p \rceil - \lambda_p$ (linha 12). Se o número de colunas do PMR for maior que $maxCols$, aquelas colunas associadas às variáveis λ que estiverem fora da base são removidas (linha 15). Por fim, o algoritmo de GC é resolvido sobre o PMR atual (linha 17).

Algoritmo 5 Refinamento(PMR, *sol*)

```
 1: sol* ← sol
 2: Atualiza PMR removendo as colunas associadas às variáveis λ que estão fora da base
 3: PMR′ ← PMR
 4: para cada tipo de servidor i ∈ N faça
 5:     Atualiza PMR restaurando os bounds originais destas variáveis, i.e., λ ≥ 0, associadas ao tipo de
        servidor i atualmente fixado
 6:     sol ← Construcao(PMR)
 7:     se f(sol) < f(sol*) então
 8:         sol* ← sol
 9:     fim se
10:     PMR ← PMR′
11: fim para
12: retorna sol*
```

Uma rotina de refinamento, cujo pseudocódigo está descrito em Algoritmo 5, é aplicada à solução obtida pelo procedimento construtivo de forma a melhorar tal solução, e funciona como segue. Seja PMR′ uma cópia da base de PMR (linhas 2 - 3). Para cada tipo de servidor $i \in N$, a fixação das variáveis λ associadas a i é desfeita e o procedimento construtivo é chamado para tentar encontrar uma solução viável inteira melhorada (linhas 4 - 11). Caso a solução seja melhorada, a melhor solução é atualizada (linhas 7 - 9). Ao final de cada iteração, PMR é atualizado com a cópia guardada em PMR′ (linha 10).

Capítulo 5

Resultados Computacionais

Este capítulo apresenta uma análise experimental das abordagens propostas no Capítulo 4 para tratar o PCCVH. Tais resultados derivam da aplicação destas abordagens para resolver um conjunto de 36 instâncias de problemas-teste. A Seção 5.1 descreve como as capacidades e os consumos energéticos de cada servidor foram medidos. Estes dados são usados para gerar as instâncias de problemas-teste da forma descrita na Seção 5.2. Na Seção 5.3 são apresentados e discutidos os resultados obtidos pelas abordagens de solução propostas no Capítulo 4 quando aplicadas às instâncias geradas.

5.1 Capacidade e Consumo Energético dos Servidores

As instâncias utilizadas para avaliar o desempenho das abordagens propostas neste trabalho são derivadas de medições de consumo energético e capacidade de processamento para quatro tipos de servidores diferentes. Estes tipos de servidores considerados possuem tecnologias distintas, o que implica em intervalos diferentes para as possíveis frequências de operação e também no número de frequências de operação possíveis para cada tipo. Foram medidos, para cada tipo de servidor operando em cada frequência de CPU disponível, os consumos energéticos e capacidades de processamento médios. Mais detalhes sobre os tipos de servidores considerados e seus respectivos consumos energéticos e capacidades de processamento medidas podem ser encontrados em Petrucci et al (2011).

A capacidade de processamento de um servidor apresenta uma relação linear crescente com a frequência de operação. O consumo energético também cresce com a frequên-

cia, porém este crescimento se dá de acordo com uma curva de comportamento não linear e não convexo, onde, tipicamente há um crescimento sublinear no início da curva e, a partir de certo ponto há uma inflexão e o crescimento passa a ser aproximadamente quadrático.

Para a medição do consumo energético, foi utilizado o medidor *WattsUP Pro* diretamente na alimentação de corrente AC com precisão definida de 1% (Electronic Educational Devices, 2010). Tais medidas de consumo são do servidor completo e não apenas de sua CPU. Na Tabela 5.1 estão sumarizados os valores dos consumos energéticos em cada frequência de CPU de cada um dos tipos de servidores considerados neste trabalho.

Tabela 5.1: Consumos energéticos (em Watts) para cada frequência de cada tipo de servidor

Processador	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9
Intel Core i7	90	94	99	104	110	117	125	134	147
AMD Phenom II X4	76	109	133	176	-	-	-	-	-
Intel Core2Duo	73	94	-	-	-	-	-	-	-
AMD Athlon 64 X2	16	34	39	49	60	74	90	108	-

As capacidades de processamento de cada tipo de servidor foram medidas para cada frequência de operação possível, tendo como unidade o número de requisições de aplicações *web* por segundo que um dado tipo de servidor numa dada frequência pode executar. Para gerar os dados de cargas de processamento necessárias para medir as capacidades dos servidores foi utilizada a ferramenta *httperf* (Mosberger e Jin, 1998). Os resultados dos valores medidos das capacidades de processamento são encontrados na Tabela 5.2 a seguir.

Tabela 5.2: Capacidades de processamento (em requisições/s) para cada frequência de cada tipo de servidor

Processador	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9
Intel Core i7	820	889	957	1025	1094	1162	1230	1299	1367
AMD Phenom II X4	300	786	935	1197	-	-	-	-	-
Intel Core2Duo	474	710	-	-	-	-	-	-	-
AMD Athlon 64 X2	128	229	255	280	306	331	356	382	-

Neste trabalho, os valores médios de consumo energético e capacidade de processamento foram medidos durante períodos de dois minutos, o suficiente para obter tais valores com desvios padrões baixos. Vale notar que o caso dos servidores *web*, como os encontrados em serviços que se baseiam na ideia de computação em nuvem, é semelhante ao dos servidores de uso intensivo de CPU e diferentes unidades de medida da capacidade

dos servidores, como por exemplo Milhões de Operações de Ponto Flutuante por Segundo (MFLOP), poderiam ter sido usadas.

5.2 Instâncias de Problemas-teste

Para testar as abordagens propostas neste trabalho para resolver o PCCVH foi gerado um conjunto de instâncias utilizando os dados obtidos da forma descrita na Seção 5.1. Tendo estes dados disponíveis, uma instância pode ser gerada ao se fornecer os seguintes dados de entrada adicionais:

- Número de tipos de servidores;
- número de servidores de cada tipo;
- número de serviços que serão executados no *cluster* e
- o intervalo dentro do qual os valores das demandas dos serviços serão geradas.

Cada instância tem um nome com a notação ct-G-X-Y-Zw, onde o parâmetro G representa um dado grupo de instâncias, o parâmetro X define o número de tipos diferentes de servidores no *cluster* e o número de serviços que serão executados é dado por $Y \in \{1000, 2000, 4000\}$. O parâmetro $Z \in \{a, b, c\}$ define o intervalo no qual as demandas dos serviços são geradas, onde $a = \{1, ..., 100\}$, $b = \{20, ..., 100\}$ e $c = \{50, ..., 100\}$. Considerando que há limitação de precisão na estimativa dos valores de demanda dos serviços, é razoável supor que, em muitos casos onde existirem milhares de serviços, muitos destes valores serão próximos ou iguais. Logo, o número de demandas com valores significativamente diferentes deverá ser bem menor do que o número de serviços. Por fim, w é usado apenas por questão de distinção entre instâncias que possuam os outros quatro parâmetros iguais.

Dois grupos de instâncias foram gerados, de forma que cada um dos grupos contém 18 problemas-teste. No primeiro grupo ($G = 1$), tem-se que o número de frequências de CPU disponíveis está associado ao tipo de servidor e estas frequências determinam os consumos energéticos e as capacidades de processamento da forma apresentada nas Tabelas 5.1 e 5.2. Logo, os valores das demandas dos serviços são gerados randomicamente dentro do intervalo especificado. Para o segundo grupo de instâncias ($G = 2$), os valores das

demandas dos serviços são mantidos, no entanto as capacidades de cada tipo de servidor em cada frequência são divididas por uma constante, neste caso constante escolhida é igual a 5, e o número de servidores disponíveis de cada tipo é multiplicado pela mesma constante. Assim sendo, a diferença entre os dois grupos está na relação entre as demandas dos serviços e as capacidades dos servidores, de forma que esta relação é maior nas instâncias do segundo grupo.

5.3 Resultados e Discussões

Os algoritmos das abordagens propostas foram implementados em C++ e os testes foram executados em um PC Intel® Core™ i5 com 3,2 GHz e 4 GB RAM e sistema operacional Linux 64 bits sendo utilizada uma única *thread*. Para a implementação da formulação $\mathcal{F}_2$ foram utilizadas as bibliotecas ILOG Concert Technology, bem como para a implementação do PMR inicial nas abordagens que se baseiam em $\mathcal{MP}$ e $\mathcal{MP}_{\text{ex}}$. Para as manipulações do PMR necessárias nos algoritmos de GC baseados em $\mathcal{MP}$ e $\mathcal{MP}_{\text{ex}}$ foi utilizada a CPLEX Callable Library. O resolvedor de PL e PLI usado foi o ILOG CPLEX 11.2. Para resolver os Problemas da Mochila necessários para a etapa de *pricing* utilizou-se o algoritmo Bouknap[1] (Pisinger, 2000) para encontrar os limites inferiores das formulações $\mathcal{MP}$ e $\mathcal{MP}_{\text{ex}}$ e também para obter os limites superiores através de HPMRI. De acordo com observações de experimentos preliminares, para HA decidiu-se utilizar o algoritmo Bouknap apenas até encontrar o limite inferior (linha 3 do Algoritmo 4) e em seguida é utilizado algoritmo Minknap[2] (Pisinger, 1997) que resolve Problemas da Mochila Binária para ser o resolvedor dos subproblemas de *pricing*.

Nas tabelas apresentadas adiante, **Instância** denota o problema-teste, **LB** é o *lower bound* obtido pela respectiva abordagem, **Tempo** indica o tempo de CPU em segundos, **Iter** é o número de iterações do algoritmo de GC, **Cols** corresponde ao número de colunas geradas, **Melhor LB** é o melhor *lower bound* obtido, **UB** é o *upper bound* obtido por uma dada abordagem e **Gap (%)** representa a diferença percentual entre o *upper bound* e o melhor *lower bound* obtido.

Na Tabela 5.3 estão dispostos os resultados das relaxações lineares obtidos pela formulação $\mathcal{F}_2$ e pelas abordagens de GC no primeiro grupo de instâncias. Dados estes

[1]Código fonte disponível em `http://www.diku.dk/hjemmesider/ansatte/pisinger/bouknap.c`
[2]Código fonte disponível em `http://www.diku.dk/hjemmesider/ansatte/pisinger/minknap.c`

resultados, percebe-se que $\mathcal{F}_2$ foi capaz de encontrar melhores limites inferiores, e em menor tempo computacional, do que os encontrados pelos algoritmos de GC baseados em $\mathcal{MP}$ e $\mathcal{MP}_{\text{ex}}$, com exceção da instância ct-1-4-2000-c2 onde tanto $\mathcal{F}_2$ quanto os algoritmos de GC obtiveram o mesmo limite inferior. Comparando estas duas últimas abordagens, observa-se que o número de iterações do algoritmo de GC necessárias para encontrar os limites inferiores foi menor ou igual na abordagem baseada em $\mathcal{MP}_{\text{ex}}$ do que na que se baseia em $\mathcal{MP}$ em 15 das 18 instâncias. Já com relação ao número de colunas geradas, $\mathcal{MP}_{\text{ex}}$ conseguiu melhores resultados do que $\mathcal{MP}$ em todas as instâncias ct-1-4-2000-c2 e ct-1-4-4000-c1. Ainda comparando as abordagens de GC baseadas em $\mathcal{MP}$ e $\mathcal{MP}_{\text{ex}}$, nota-se que na maioria das instâncias a primeira produziu limites inferiores em menos tempo que a segunda. Como a diferença entre os tempos das duas abordagens de GC é pequena e com base nas médias dos ao número de iterações e ao número de colunas geradas, pode-se concluir que a abordagem baseada em $\mathcal{MP}_{\text{ex}}$ mostrou melhor desempenho do que a baseada em $\mathcal{MP}$ ao encontrar limites inferiores no primeiro grupo de instâncias.

Tabela 5.3: Resultados obtidos por $\mathcal{F}_2$ e pelas abordagens de GC no primeiro grupo de instâncias

	$\mathcal{F}_2$		$\mathcal{MP}$				$\mathcal{MP}_{\text{ex}}$			
Instância	LB	Tempo	LB	Tempo	Iter	Cols	LB	Tempo	Iter	Cols
ct-1-4-1000-a1	**6082**	< 0,01	6079	0,12	84	1325	6079	**0,11**	**70**	**710**
ct-1-4-1000-b1	**7064**	< 0,01	7062	**0,06**	66	1011	7062	0,08	**56**	**504**
ct-1-4-1000-c1	**9181**	< 0,01	9176	**0,03**	**31**	342	9176	0,05	32	**302**
ct-1-4-1000-a2	**5712**	< 0,01	5710	0,11	83	1357	5710	**0,09**	**72**	**700**
ct-1-4-1000-b2	**7080**	< 0,01	7076	**0,06**	62	984	7076	0,08	**55**	**488**
ct-1-4-1000-c2	**9132**	< 0,01	9127	0,03	34	394	9127	**0,02**	**31**	**293**
ct-1-4-2000-a1	**12025**	< 0,01	12021	**0,08**	**77**	1127	12021	0,10	**77**	**754**
ct-1-4-2000-b1	**14609**	< 0,01	14602	**0,05**	**48**	558	14602	0,07	**48**	**458**
ct-1-4-2000-c1	**19572**	< 0,01	19571	**0,02**	24	429	19571	0,04	**23**	**378**
ct-1-4-2000-a2	**12087**	< 0,01	12084	**0,09**	81	1109	12084	**0,09**	**75**	**740**
ct-1-4-2000-b2	**14425**	< 0,01	14421	**0,05**	**47**	552	14421	0,07	48	**458**
ct-1-4-2000-c2	**19742**	< 0,01	**19742**	**0,03**	25	432	**19742**	0,04	**23**	**381**
ct-1-4-4000-a1	**23731**	< 0,01	23728	**0,10**	84	1126	23728	**0,10**	**79**	**870**
ct-1-4-4000-b1	**28918**	< 0,01	28911	**0,07**	55	691	28911	0,08	**53**	**604**
ct-1-4-4000-c1	**37251**	< 0,01	37247	0,04	33	583	37247	**0,03**	**27**	**514**
ct-1-4-4000-a2	**24058**	< 0,01	24055	**0,10**	82	1140	24055	0,11	**81**	**889**
ct-1-4-4000-b2	**28653**	< 0,01	28648	**0,05**	**49**	618	28648	0,08	51	**577**
ct-1-4-4000-c2	**36892**	< 0,01	36888	**0,04**	32	563	36888	0,05	**31**	**534**
Média		< 0,01		**0,06**	55,4	796,7		0,07	**51,8**	**564,1**

A Tabela 5.4 apresenta os resultados obtidos para o segundo grupo de instâncias. Ao contrário do primeiro grupo, as abordagens baseadas em GC obtiveram melhores limites inferiores do que a formulação $\mathcal{F}_2$. Assim como nas instâncias do primeiro grupo, a abordagem de GC baseada em $\mathcal{MP}_{\text{ex}}$ teve melhor performance do que a que se baseia em

$\mathcal{MP}$ com relação ao número de iterações em 13 das 18 instâncias. Semelhante também ao que ocorreu no primeiro grupo de instâncias, os resultados obtidos por $\mathcal{MP}_{\text{ex}}$ são melhores na maioria das instâncias em termos do número de colunas geradas. Além disso, os resultados de $\mathcal{MP}_{\text{ex}}$ são melhores também nas médias iterações e número de colunas geradas quando comparados aos obtidos por $\mathcal{MP}$.

Tabela 5.4: Resultados obtidos por $\mathcal{F}_2$ e pelas abordagens de GC no segundo grupo de instâncias

	$\mathcal{F}_2$		$\mathcal{MP}$				$\mathcal{MP}_{\text{ex}}$			
Instância	LB	Tempo	LB	Tempo	Iter	Cols	LB	Tempo	Iter	Cols
ct-2-4-1000-a1	30462	< 0,01	**30576**	**0,09**	91	969	**30576**	**0,09**	**87**	**775**
ct-2-4-1000-b1	35372	< 0,01	**35665**	**0,06**	59	730	**35665**	0,07	**53**	**581**
ct-2-4-1000-c1	45994	< 0,01	**46420**	**< 0,01**	40	685	**46420**	0,04	**36**	**655**
ct-2-4-1000-a2	28612	< 0,01	**28720**	**0,09**	**88**	973	**28720**	0,10	92	**828**
ct-2-4-1000-b2	35448	< 0,01	**35721**	**0,06**	61	735	**35721**	**0,06**	**56**	**612**
ct-2-4-1000-c2	45749	< 0,01	**46173**	**0,04**	**37**	675	**46173**	**0,04**	**37**	**657**
ct-2-4-2000-a1	60223	< 0,01	**60417**	**0,07**	**62**	893	**60417**	0,09	**62**	**837**
ct-2-4-2000-b1	73145	< 0,01	**73808**	**0,04**	42	883	**73808**	0,06	**38**	**829**
ct-2-4-2000-c1	97537	< 0,01	**99918**	**0,01**	25	1214	**99918**	0,02	**23**	**1204**
ct-2-4-2000-a2	60541	< 0,01	**60741**	**0,06**	61	906	**60741**	0,07	**60**	**826**
ct-2-4-2000-b2	72242	< 0,01	**72801**	**0,04**	41	871	**72801**	0,06	**38**	**815**
ct-2-4-2000-c2	98213	< 0,01	**100796**	**0,01**	**21**	**1197**	**100796**	0,03	23	1204
ct-2-4-4000-a1	119018	< 0,01	**119264**	**0,12**	88	1464	**119264**	**0,12**	**83**	**1370**
ct-2-4-4000-b1	145124	< 0,01	**146246**	**0,06**	**43**	1377	**146246**	0,08	45	**1347**
ct-2-4-4000-c1	186892	< 0,01	**190591**	**0,05**	31	1921	**190591**	**0,05**	**29**	**1890**
ct-2-4-4000-a2	120656	< 0,01	**120899**	0,14	95	1564	**120899**	**0,12**	**78**	**1337**
ct-2-4-4000-b2	143809	< 0,01	**144940**	**0,06**	**41**	1364	**144940**	0,07	46	**1336**
ct-2-4-4000-c2	185116	< 0,01	**188663**	**0,04**	**32**	**1924**	**188663**	**0,04**	33	1932
Média		< 0,01		**0,06**	53,22	1130,28		0,07	**51,06**	**1057,50**

Com base nos resultados apresentados nas Tabelas 5.3 e 5.4 é perceptível que a inclusão das variáveis de troca auxiliou na aceleração da convergência do algoritmo de GC, já que o número de iterações foi menor para a maioria das instâncias quando comparado com a versão do algoritmo que não possui tais variáveis. Em média, a adição destas variáveis ao problema mestre também resultou na redução do número de colunas geradas.

Vale notar também que ambos os algoritmos de GC mantêm a performance independentemente do tamanho da instância. Isto se deve ao esquema de agregação dos serviços por demanda, que faz com que a dimensão da instância não tenha influência sobre o desempenho do algoritmo, já que o número de linhas do problema será igual ao número de demandas diferentes ao invés do número de serviços.

Tendo como base os resultados apresentados nas Tabelas 5.3 e 5.4, foi escolhida a formulação $\mathcal{MP}_{\text{ex}}$ como base para as abordagens heurísticas HPMRI e HA, já que o procedimento de solução de ambas inclui o algoritmo de GC.

Na Tabela 5.5 são sumarizados os resultados obtidos pelas abordagens heurísticas, com tempo limite de 10 segundos, nas instâncias do primeiro grupo. Pode-se observar que a abordagem HA obteve melhores limites superiores em 17 das 18 instâncias com tempo médio de execução de 3,09 segundos em comparação à média de 10 segundos da heurística HPMRI, bem como menor diferença percentual média entre o limite superior obtido e o melhor limite inferior.

Tabela 5.5: Resultados obtidos pelas heurísticas propostas sobre o primeiro grupo de instâncias

		HPMRI ($\mathcal{MP}_{\text{ex}}$)			HA		
Instância	Melhor LB	UB	Tempo	Gap (%)	UB	Tempo	Gap (%)
ct-1-4-1000-a1	6082	6122	10,00	0,66	**6095**	4,15	0,21
ct-1-4-1000-b1	7064	7110	10,00	0,65	**7101**	1,89	0,52
ct-1-4-1000-c1	9181	9208	10,00	0,29	**9194**	0,96	0,14
ct-1-4-1000-a2	5712	5786	10,00	1,30	**5733**	3,97	0,37
ct-1-4-1000-b2	7080	7098	10,00	0,25	**7080**	2,13	0,00
ct-1-4-1000-c2	9132	**9162**	10,00	0,33	9206	0,74	0,81
ct-1-4-2000-a1	12025	12134	10,00	0,91	**12076**	5,27	0,42
ct-1-4-2000-b1	14609	14658	10,00	0,34	**14628**	2,64	0,13
ct-1-4-2000-c1	19572	19598	10,00	0,13	**19578**	3,18	0,03
ct-1-4-2000-a2	12087	12169	10,00	0,68	**12113**	4,44	0,22
ct-1-4-2000-b2	14425	14485	10,00	0,42	**14438**	2,77	0,09
ct-1-4-2000-c2	19742	19792	10,00	0,25	**19752**	2,45	0,05
ct-1-4-4000-a1	23731	23933	10,00	0,85	**23829**	5,64	0,41
ct-1-4-4000-b1	28918	28999	10,00	0,28	**28961**	2,32	0,15
ct-1-4-4000-c1	37251	37341	10,00	0,24	**37257**	1,94	0,02
ct-1-4-4000-a2	24058	24237	10,00	0,74	**24095**	5,43	0,15
ct-1-4-4000-b2	28653	28718	10,00	0,23	**28713**	2,99	0,21
ct-1-4-4000-c2	36892	36934	10,00	0,11	**36897**	2,65	0,01
Média			10,00	0,48		3,09	0,22

Os resultados encontrados pelas heurísticas no segundo grupo de instâncias são apresentados na Tabela 5.6 também com tempo limite de 10 segundos. Como no primeiro grupo de instâncias, as abordagens se mostraram capazes de encontrar soluções de qualidade com a heurística HA obtendo vantagem em termos de tempo computacional médio.

Com os resultados mostrados nas Tabelas 5.5 e 5.6 é possível afirmar que as abordagens propostas são eficientes em termos de qualidade de solução e tempo computacional. Tais resultados se mostram importantes no que diz respeito ao fato de que a alta qualidade das soluções encontradas fundamenta o uso destes algoritmos no auxílio à operação de *clusters* com maior eficiência em termos de energia elétrica, bem como demonstra que as abordagens mantêm o bom desempenho com o crescimento das instâncias permitindo que sejam aplicadas no apoio à consolidação de milhares de serviços quando utilizadas na

Tabela 5.6: Resultados obtidos pelas heurísticas propostas sobre o segundo grupo de instâncias

		HPMRI ($\mathcal{MP}_{\text{ex}}$)			HA		
Instância	Melhor LB	UB	Tempo	Gap (%)	UB	Tempo	Gap (%)
ctag-2-4-1000-a1	30576	**30589**	10,00	0,04	30594	0,67	0,06
ctag-2-4-1000-b1	35665	35685	10,00	0,06	**35677**	0,85	0,03
ctag-2-4-1000-c1	46420	**46422**	0,25	0,00	46435	0,22	0,03
ctag-2-4-1000-a2	28720	**28735**	10,00	0,05	28740	0,80	0,07
ctag-2-4-1000-b2	35721	**35728**	10,00	0,02	35784	0,51	0,18
ctag-2-4-1000-c2	46173	**46177**	0,16	0,01	46221	0,24	0,10
ctag-2-4-2000-a1	60417	**60424**	10,00	0,01	60430	0,83	0,02
ctag-2-4-2000-b1	73808	**73817**	2,36	0,01	73818	0,47	0,01
ctag-2-4-2000-c1	99918	**99925**	0,38	0,01	**99925**	0,15	0,01
ctag-2-4-2000-a2	60741	**60764**	10,00	0,04	60804	0,64	0,10
ctag-2-4-2000-b2	72801	**72812**	10,00	0,02	72833	0,68	0,04
ctag-2-4-2000-c2	100796	**100809**	1,19	0,01	100810	0,18	0,01
ctag-2-4-4000-a1	119264	**119293**	10,00	0,02	119303	0,97	0,03
ctag-2-4-4000-b1	146246	**146252**	0,92	0,00	146287	0,54	0,03
ctag-2-4-4000-c1	190591	**190601**	0,09	0,01	190602	0,30	0,01
ctag-2-4-4000-a2	120899	**120909**	7,99	0,01	120920	0,84	0,02
ctag-2-4-4000-b2	144940	**144952**	0,42	0,01	**144952**	0,57	0,01
ctag-2-4-4000-c2	188663	188681	0,10	0,01	**188663**	0,16	0,00
Média			5,21	0,02		0,53	0,04

configuração de *clusters* de grande escala.

Uma possível maneira de obter resultados ainda melhores seria através da utilização de arquiteturas multi-*thread*, onde cada *thread* executaria diferentes versões dos algoritmos durante um tempo limite e, a partir disso seria escolhida a melhor solução encontrada dentre as execuções. Além disso, uma possível forma de avaliar as abordagens propostas com situações mais próximas da realidade seria por meio de simulações com instâncias que apresentem alterações ao longo do tempo de forma dinâmica com variações do número de serviços e suas demandas.

Capítulo 6

Conclusão

Neste trabalho foi apresentado o PCCVH, que consiste em encontrar a melhor configuração global dos servidores de um *cluster* virtualizado heterogêneo, onde deve-se decidir quais servidores estarão ligados e em qual frequência de CPU de forma que todas as demandas pelos serviços executados por aplicações hospedadas no *cluster* sejam atendidas e o consumo energético necessário para isso seja minimizado.

Para tratar o PCCVH foram propostas duas abordagens heurísticas baseadas em GC com o objetivo de obter soluções inteiras viáveis de boa qualidade em tempo computacional razoável. A primeira delas (HPMRI) consiste na resolução do problema de PLI que se obtém ao final do algoritmo de GC e a segunda (HA) é baseada em PL e busca encontrar um solução inteira viável a partir de uma solução fracionária arredondando os valores das variáveis desta última por um procedimento iterativo e, depois disso tenta-se melhorar a solução encontrada com a aplicação de uma busca local.

Tais abordagens foram testadas em experimentos realizados sobre instâncias geradas com características variadas que buscam retratar cenários distintos com relação à dimensão do *cluster*, ao número de serviços que devem ser executados e variedade das demandas dos serviços, além de contar com dados realistas de frequências de operação, capacidade e consumo energético dos servidores. Foram geradas 36 instâncias divididas em dois grupos que se diferenciam pela relação entre as demandas dos serviços e as capacidades dos servidores do *cluster*.

A partir dos resultados obtidos dos experimentos sobre estas instâncias é possível concluir que os objetivos das abordagens propostas neste trabalho foram alcançados. Estes resultados mostram que os algoritmos desenvolvidos são capazes de obter soluções

próximas do valor ótimo e ainda de manter a eficiência independentemente da dimensão da instância, o que também sugere que as abordagens são potencialmente aplicáveis na prática.

A abordagem HA possui uma importante vantagem prática em relação a abordagem HPMRI: não necessitar de um resolvedor de PLI. Atualmente, todos os pacotes para a resolução de PLI com alto desempenho, como o CPLEX (usado neste trabalho), o XPRESS-MP (Xpress MP, 2006) ou o Gurobi (Gurobi Optimization, 2010), são de código proprietário. Suas licenças para uso não-acadêmico são bastante caras, da ordem de dezenas de milhares de dólares. Existem pacotes para PLI de código aberto e gratuitos, como o GLPK[1] ou o COIN-/BCP (Ralphs e Ladányi, 2001), porém com desempenho bastante inferior. Por outro lado, a abordagem HA apenas necessita de um resolvedor de PL. Nesse caso, o desempenho dos pacotes abertos e gratuitos é bem mais próximo do desempenho dos pacotes comerciais.

Como trabalho futuro sugere-se a utilização de arquiteturas multi-*thread* para obter um conjunto de soluções distintas para que se possa escolher a que leva à melhor configuração do *cluster*. Também pode ser explorada a ideia dos servidores com processadores que possuem mais de um núcleo onde as configurações poderiam assumir diferentes frequências em cada um dos núcleos de um mesmo servidor. Além disso, para avaliar os algoritmos em situações mais próximas do real, seria interessante realizar simulações com eventos de variação dinâmica das demandas dos serviços em intervalos de tempo que façam com que reconfigurações periódicas do *cluster* sejam necessárias.

No âmbito das abordagens de solução, sugere-se investigar a combinação do algoritmo de GC com um procedimento de *branch-and-bound*, resultando no chamado *branch-and-price*. Desta forma, poderia-se ter um algoritmo capaz de obter soluções provadamente ótimas. No entanto, dadas as características do PCCVH em uma situação real, este algoritmo também deveria ser capaz de obter tais soluções em um baixo tempo computacional.

[1]Informações em `http://www.gnu.org/software/glpk/`

Referências Bibliográficas

AGARWAL, Y.; MATHUR, K.; SALKIN, H. M. A set-partitioning-based exact algorithm for the vehicle routing problem. *Networks*, v. 19, n. 7, p. 731–749, 1989.

ALVES, C.; VALÉRIO DE CARVALHO, J. M. Accelerating column generation for variable sized bin-packing problems. *European Journal of Operational Research*, v. 183, n. 3, p. 1333–1352, 2007.

ALVES, C.; VALÉRIO DE CARVALHO, J. M. A stabilized branch-and-price-and-cut algorithm for the multiple length cutting stock problem. *Computers & Operations Research*, v. 35, n. 4, p. 1315–1328, 2008.

ARENALES, M.; ARMENTANO, V.; MORABITO, R.; YANASSE, H. *Pesquisa operacional*. Rio de Janeiro: Elsevier, 2007.

BARROSO, L. A.; HÖLZLE, U. The case for energy-proportional computing. *Computer*, v. 40, n. 12, p. 33–37, 2007.

BELOV, G.; LETCHFORD, A. N.; UCHOA, E. A node-flow model for 1d stock cutting: Robust branch-cut-and-price. *RPEP, Relatório Técnico*, v. 5, n. 7, 2005.

BELOV, G.; SCHEITHAUER, G. A cutting plane algorithm for the one-dimensional cutting stock problem with multiple stock lengths. *European Journal of Operational Research*, v. 141, n. 2, p. 274–294, 2002.

BELOV, G.; SCHEITHAUER, G. A branch-and-cut-and-price algorithm for one-dimensional stock cutting and two-dimensional two-stage cutting. *European journal of operational research*, v. 171, n. 1, p. 85–106, 2006.

BERL, A.; GELENBE, E.; DI GIROLAMO, M.; GIULIANI, G.; DE MEER, H.; DANG, M.; PENTIKOUSIS, K. Energy-efficient cloud computing. *The Computer Journal*, v. 53, n. 7, p. 1045, 2010.

BERTINI, L.; LEITE, J.; MOSSÉ, D. Statistical QoS guarantee and energy-efficiency in web server clusters. *Euromicro Conference on Real-Time Systems*, p. 83–92, 2007.

BETTINELLI, A.; CESELLI, A.; RIGHINI, G. A branch-and-cut-and-price algorithm for the multi-depot heterogeneous vehicle routing problem with time windows. *Transportation Research Part C: Emerging Technologies*, 2010.

BIANCHINI, R.; RAJAMONY, R. Power and energy management for server systems. *Computer*, v. 37, n. 11, p. 68–76, 2004.

BICHLER, M.; SETZER, T.; SPEITKAMP, B. Capacity planning for virtualized servers. *Workshop on Information Technologies and Systems (WITS), Milwaukee, Wisconsin, USA*, 2006.

BISCHOFF, E. E.; WÄSCHER, G. Cutting and packing. *European Journal of Operational Research*, v. 84, n. 3, p. 503–505, 1995.

CARDELLINI, V.; CASALICCHIO, E.; COLAJANNI, M.; YU, P. S. The state of the art in locally distributed web-server systems. *ACM Computing Surveys*, v. 34, n. 2, p. 263–311, 2002.

CATTRYSSE, D.; SALOMON, M.; KUIK, R.; VAN WASSENHOVE, L. N. A dual ascent and column generation heuristic for the discrete lotsizing and scheduling problem with setup times. *Management Science*, v. 39, n. 4, p. 477–486, 1993.

CHUNG, F.; GRAHAM, R.; MAO, J.; VARGHESE, G. Parallelism versus memory allocation in pipelined router forwarding engines. *Theory of Computing Systems*, v. 39, n. 6, p. 829–849, 2006.

CHURCH, K.; GREENBERG, A.; HAMILTON, J. On delivering embarrassingly distributed cloud services. *HotNets*, 2008.

COFFMAN JR, E. G.; GAREY, M. R.; JOHNSON, D. S. Approximation algorithms for bin packing: A survey. *Approximation algorithms for NP-hard problems*, p. 46–93. PWS Publishing Co., 1996.

DANTZIG, G.; THAPA, M. *Linear Programming 2: Theory and Extensions*. Springer-Verlag, 2003.

DANTZIG, G. B.; WOLFE, P. Decomposition principle for linear programs. *Operations research*, v. 8, n. 1, p. 101–111, 1960.

DESROCHERS, M.; DESROSIERS, J.; SOLOMON, M. A new optimization algorithm for the vehicle routing problem with time windows. *Operations Research*, v. 40, n. 2, p. 342–354, 1992.

DESROSIERS, J.; SOUMIS, F.; DESROCHERS, M. Routing with time windows by column generation. *Networks*, v. 14, n. 4, p. 545–565, 1984.

DYCKHOFF, H. A typology of cutting and packing problems. *European Journal of Operational Research*, v. 44, n. 2, p. 145–159, 1990.

ELECTRONIC EDUCATIONAL DEVICES. Watts Up PRO. Disponível em: <http://www.wattsupmeters.com/>, 2010.

ELNOZAHY, E. N.; KISTLER, M.; RAJAMONY, R. Energy-efficient server clusters. *Power-Aware Computer Systems*, volume 2325 of *Lecture Notes in Computer Science*, p. 179–197, 2003.

ERTEK, G.; KILIC, K. Decision support for packing in warehouses. *Computer and Information Sciences–ISCIS 2006*, p. 115–124, 2006.

FAN, X.; WEBER, W.-D.; BARROSO, L. A. Power provisioning for a warehouse-sized computer. *ISCA '07: Proceedings of the 34th annual international symposium on Computer architecture*, p. 13–23, New York. ACM, 2007.

FEKETE, S. P.; SCHEPERS, J. New classes of fast lower bounds for bin packing problems. *Mathematical programming*, v. 91, n. 1, p. 11–31, 2001.

FORD, L. R.; FULKERSON, D. R. A suggested computation for multicommodity flows. *Management Science*, v. 5, n. 1, p. 97–101, 1958.

FRIESEN, D. K.; LANGSTON, M. A. Variable sized bin packing. *SIAM journal on computing*, v. 15, p. 222, 1986.

GILMORE, P. C.; GOMORY, R. E. A linear programming approach to the cutting stock problem. *Operations research*, v. 9, n. 6, p. 849–859, 1961.

GILMORE, P. C.; GOMORY, R. E. A linear programming approach to the cutting stock problem-Part II. *Operations research*, v. 11, n. 6, p. 863–888, 1963.

GUROBI OPTIMIZATION. *Gurobi optimizer reference manual version 3.0*, 2010.

HAOUARI, M.; SERAIRI, M. Relaxations and exact solution of the variable sized bin packing problem. *Computational Optimization and Applications*, p. 1–24, 2009.

HAYES, B. Cloud computing. *Communications of the ACM*, v. 51, n. 7, p. 9–11, 2008.

HEATH, T.; DINIZ, B.; CARRERA, E. V.; MEIRA JR., W.; BIANCHINI, R. Energy conservation in heterogeneous server clusters. *Proceedings of the tenth ACM SIGPLAN symposium on Principles and practice of parallel programming*, p. 186–195. ACM, 2005.

HORVATH, T.; ABDELZAHER, T.; SKADRON, K.; LIU, X. Dynamic voltage scaling in multitier web servers with end-to-end delay control. *IEEE Transactions on Computers*, p. 444–458, 2007.

JOHNSON, E. Modelling and strong linear programs for mixed integer programming. *Algorithms and Model Formulations in Mathematical Programming*, p. 1–43. Springer-Verlag New York, Inc., 1989.

KANDASAMY, N.; ABDELWAHED, S.; HAYES, J. P. Self-optimization in computer systems via on-line control: Application to power management. *ICAC '04*, p. 54–61, 2004.

KANTOROVICH, L. Mathematical methods of organising and planning production (translated from a report in russian, dated 1939). *Management Science*, v. 6, n. 4, p. 366–422, 1960.

KAPLAN, J.; FORREST, W.; KINDLER, N. Revolutionizing data center energy efficiency. *McKinsey & Company, Relatório Técnico*, 2008.

KHANAFER, A.; CLAUTIAUX, F.; TALBI, E. New lower bounds for bin packing problems with conflicts. *European Journal of Operational Research*, v. 206, n. 2, p. 281–288, 2010.

KHANNA, G.; BEATY, K.; KAR, G.; KOCHUT, A. Application performance management in virtualized server environments. *10th IEEE/IFIP Network Operations and Management Symposium*, p. 373–381, 2006.

KHARGHARIA, B.; HARIRI, S.; YOUSIF, M. S. Autonomic power and performance management for computing systems. *Cluster Computing*, v. 11, n. 2, p. 167–181, 2008.

KIYANCLAR, N. A survey of virtualization techniques focusing on secure on-demand cluster computing. *Arxiv preprint cs.OS/0511010*, 2005.

KOOMEY, J. G. Estimating total power consumption by servers in the U.S. and the world. http://enterprise.amd.com/Downloads/ svrpwrusecompletefinal.pdf, 2007.

KUSIC, D.; KEPHART, J. O.; HANSON, J. E.; KANDASAMY, N.; JIANG, G. Power and performance management of virtualized computing environments via lookahead control. *Cluster Computing*, v. 12, n. 1, p. 1–15, 2009.

LASDON, L. *Optimization theory for large systems*. Dover Publications, 2002.

MALKEVITCH, J. Bin packing and machine scheduling. *Feature Column from the AMS: Monthly Essays on Mathematical Topics*, 2004.

MONACI, M. Algorithms for packing and scheduling problems. *4OR: A Quarterly Journal of Operations Research*, v. 1, n. 1, p. 85–87, 2003.

MORENO, L.; DE ARAGÃO, M. P.; UCHOA, E. Improved lower bounds for the Split Delivery Vehicle Routing Problem. *Operations Research Letters*, 2010.

MOSBERGER, D.; JIN, T. httperf – a tool for measuring web server performance. *SIGMETRICS Performance Evaluation Review*, v. 26, n. 3, p. 31–37, 1998.

OLIVEIRA, J. F.; WÄSCHER, G. Cutting and packing. *European Journal of Operational Research*, v. 183, n. 3, p. 1106–1108, 2007.

PESSOA, A.; DE ARAGÃO, M. P.; UCHOA, E. Robust branch-cut-and-price algorithms for vehicle routing problems. *The vehicle routing problem: Latest advances and new challenges*, p. 297–325, 2008.

PETRUCCI, V.; CARRERA, E. V.; LOQUES, O.; LEITE, J.; MOSSÉ, D. Optimized management of power and performance for virtualized heterogeneous server clusters. *11th IEEE/ACM International Symposium on Cluster, Cloud and Grid (CCGrid'11)*, 2011.

PETRUCCI, V.; LOQUES, O.; MOSSÉ, D. A dynamic configuration model for power-efficient virtualized server clusters. *11th Brazillian Workshop on Real-Time and Embedded Systems (WTR)*, volume 2. Citeseer, 2009.

PETRUCCI, V.; LOQUES, O.; MOSSÉ, D. A dynamic optimization model for power and performance management of virtualized clusters. *Proceedings of the 1st International Conference on Energy-Efficient Computing and Networking*, p. 225–233. ACM, 2010.

PIGATTI, A.; DE ARAGÃO, M. P.; UCHOA, E. Stabilized branch-and-cut-and-price for the generalized assignment problem. *Annals of GRACO*, v. 5, p. 389–395, 2005.

PISINGER, D. A minimal algorithm for the 0-1 knapsack problem. *Operations Research*, v. 45, n. 5, p. 758–767, 1997.

PISINGER, D. A minimal algorithm for the bounded knapsack problem. *INFORMS Journal on Computing*, v. 12, n. 1, p. 75, 2000.

RALPHS, T. K.; LADÁNYI, L. COIN-OR: Common optimization interface for operations research. COIN/BCP user's manual. *International Business Machines Corporation Report*, 2001.

RANGANATHAN, P. Recipe for efficiency: principles of power-aware computing. *Communications of the ACM*, v. 53, n. 4, p. 60–67, 2010.

RUSU, C.; FERREIRA, A.; SCORDINO, C.; WATSON, A.; MOSSÉ, D. Energy-efficient real-time heterogeneous server clusters. *Real-Time and Embedded Technology and Applications Symposium, 2006. Proceedings of the 12th IEEE*, p. 418–428. IEEE, 2006.

SAVELSBERGH, M. A branch-and-price algorithm for the generalized assignment problem. *Operations Research*, v. 45, n. 6, p. 831–841, 1997.

SHARMA, V.; THOMAS, A.; ABDELZAHER, T.; SKADRON, K.; LU, Z. Power-aware QoS management in web servers. *In Proceedings of the 24th IEEE Real-Time systems Symposium (RTSS'03)*, p. 63–72, 2003.

SKITT REUVEN, R.; ROBIN, A. Vehicle routing via column generation. *European Journal of Operational Research*, v. 21, n. 1, p. 65–76, 1985.

SRIKANTAIAH, S.; KANSAL, A.; ZHAO, F. Energy aware consolidation for cloud computing. *Proceedings of the 2008 conference on power aware computing and systems*, HotPower'08, p. 10–10, Berkeley. USENIX Association, 2008.

SWEENEY, P. E.; PATERNOSTER, E. R. Cutting and packing problems: a categorized, application-orientated research bibliography. *Journal of the Operational Research Society*, v. 43, n. 7, p. 691–706, 1992.

VALÉRIO DE CARVALHO, J. M. Exact solution of bin-packing problems using column generation and branch-and-bound. *Annals of Operations Research*, v. 86, p. 629–659, 1999.

VALÉRIO DE CARVALHO, J. M. LP models for bin packing and cutting stock problems. *European Journal of Operational Research*, v. 141, n. 2, p. 253–273, 2002.

VALÉRIO DE CARVALHO, J. M. Using extra dual cuts to accelerate column generation. *INFORMS Journal on Computing*, v. 17, n. 2, p. 175, 2005.

VANDERBECK, F. *Decomposition and column generation for integer programs*. Ph.d. thesis, Université Catholique de Louvain, 1994.

VANDERBECK, F. Lot-sizing with start-up times. *Management Science*, p. 1409–1425, 1998.

VANDERBECK, F. Computational study of a column generation algorithm for bin packing and cutting stock problems. *Mathematical Programming*, v. 86, n. 3, p. 565–594, 1999.

VANDERBECK, F. On Dantzig-Wolfe decomposition in integer programming and ways to perform branching in a branch-and-price algorithm. *Operations Research*, v. 48, n. 1, p. 111–128, 2000.

VANDERBECK, F. Implementing mixed integer column generation. DESAULNIERS, G.; DESROSIERS, J.; SOLOMON, M. M. (Eds.), *Column Generation*, p. 331–358. Springer US, 2005.

WANG, P. Y.; WÄSCHER, G. Cutting and packing. *European Journal of Operational Research*, v. 141, n. 2, p. 239–240, 2002.

WANG, Y.; WANG, X.; CHEN, M.; ZHU, X. Power-efficient response time guarantees for virtualized enterprise servers. *Proceedings of the 2008 Real-Time Systems Symposium*, p. 303–312, Washington, DC, USA. IEEE Computer Society, 2008.

WÄSCHER, G.; HAUSSNER, H.; SCHUMANN, H. An improved typology of cutting and packing problems. *European Journal of Operational Research*, v. 183, p. 1109–1130, 2007.

WILHELM, W. E. A technical review of column generation in integer programming. *Optimization and Engineering*, v. 2, n. 2, p. 159–200, 2001.

XPRESS MP. Xpress MP software development. *Dash optimization, Release*, 2006.