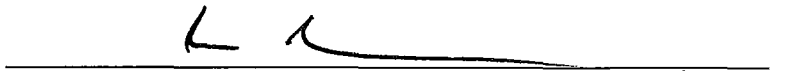
NOVAS RELAXAÇÕES LINEARES PARA O PROBLEMA DE SCHEDULING DE PROCESSOS EM MULTIPROCESSADORES COM RESTRIÇÕES DE ATRASO

Pedro Paulo Rodrigues Teixeira Filho

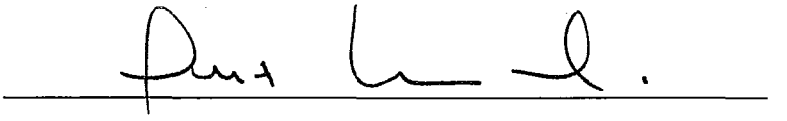
DISSERTAÇÃO SUBMETIDA AO CORPO DOCENTE DA COORDENAÇÃO DOS PROGRAMAS DE PÓS-GRADUAÇÃO DE ENGENHARIA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS EM ENGENHARIA DE SISTEMAS E COMPUTAÇÃO.

Aprovada por:



Prof. Nelson Maculan Filho, Ph.D.

Prof. Claudio Thomas Bornstein, Ph.D.



Prof. Luiz Satoru Ochi, D.Sc.

RIO DE JANEIRO, RJ - BRASIL

NOVEMBRO DE 2006

TEIXEIRA FILHO, PEDRO PAULO RODRIGUES

Novas Relaxações Lineares para o Problema de Scheduling de Processos em Multiprocessadores com Restrições de Atraso [Rio de Janeiro] 2006

XI, 65 p. 29,7 cm (COPPE/UFRJ, M.Sc., Engenharia de Sistemas e Computação, 2006)

Dissertação – Universidade Federal do Rio de Janeiro, COPPE

1 - *Scheduling* determinístico

2 - Multiprocessadores homogêneos

3 - Programação Inteira

I. COPPE/UFRJ II. Título (série)

Aos meus pais.

Agradecimentos

Antes de tudo, agradeço a Deus, criador e dono de toda a ciência, onisciente, onipotente, onipresente, e ainda assim se preocupa conosco, meras criaturas, mas incondicionalmente amadas por Ele e chamadas de *filhos*. A Ele, toda honra e glória.

Aos meus pais, pelo incentivo, apoio, exemplo, amor e carinho com que me educaram, nunca medindo esforços para me proporcionar o melhor que poderiam em todos os aspectos da vida, e ao meu irmão, pelo incentivo e companheirismo de sempre. Serei sempre grato a vocês. Amo vocês!

À minha Tia Ângela, por ter me acolhido em sua casa, nunca medindo esforços para que nada me faltasse, agindo como uma verdadeira mãe para mim. Te amo tia!

Ao Prof. Maculan, muito mais que um orientador, um amigo e às vezes até um "paizão", por aceitar me orientar diante das grandes dificuldades "logísticas" existentes, e mesmo assim forneceu dicas e conselhos essenciais para o desenvolvimento do trabalho, além de ter aberto de certa forma as portas do mundo para mim ao me dar a oportunidade de trabalhar com pessoas de diversos lugares do mundo.

Ao Prof. Leo Liberti, da *École Polytechnique de Paris*, por ter também me orientado no desenvolvimento do trabalho além de ter dado acesso às máquinas na Itália necessárias para a realização dos testes.

À pesquisadora Tatjana Davidovic, da Academia de Ciências e Artes Sérvias, por ter me fornecido o conjunto de instâncias usadas para realizar os testes necessários no trabalho.

Ao amigos do LabOtim, Adrya, Elivelton, Luidi, e outros, em especial Yuri Abitbol, pela enorme ajuda na revisão e correção deste texto, além das preciosas dicas durante a elaboração de todo o trabalho. À grande amiga Fátima, também pertecente à família LabOtim, por toda sua ajuda durante estes anos, em especial por ter resolvido na última hora um enorme problema para mim. Sou muito grato a vocês todos!

Finalmente, não poderia esquecer também de todos os meus parentes e outros amigos que de certa forma me apoiaram e incentivaram durante todo o mestrado, em especial o grande colega Glaydiston, cuja enorme ajuda e paciência para ensinar foram essenciais na disciplina de Álgebra Linear, e a amiga Carmen Lucia, que também ajudou na revisão e correção do texto.

Resumo da Dissertação apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Mestre em Ciências (M.Sc.)

NOVAS RELAXAÇÕES LINEARES PARA O PROBLEMA DE SCHEDULING DE PROCESSOS EM MULTIPROCESSADORES COM RESTRIÇÕES DE ATRASO

Pedro Paulo Rodrigues Teixeira Filho

Novembro/2006

Orientador: Nelson Maculan Filho
Programa: Engenharia de Sistemas e Computação

Descrevemos e avaliamos uma nova formulação matemática para o problema de escalonamento de tarefas em um sistema de multiprocessadores homogêneos com atrasos de comunicação. As formulações existentes envolvem um grande número de variáveis binárias com três índices. A formulação aqui discutida usa variáveis binárias com no máximo dois índices. Um conjunto de inequações válidas para a formulação é apresentada. Resultados numéricos de instâncias de tamanho pequeno e médio são apresentados.

Abstract of Dissertation presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Master of Science (M.Sc.)

NEW LINEAR RELAXATIONS FOR THE MULTIPROCESSOR SCHEDULING PROBLEM WITH COMMUNICATION DELAYS

Pedro Paulo Rodrigues Teixeira Filho

Novembro/2006

Advisor: Nelso Maculan Filho

Department: Systems Engineering and Computer Science

We describe and evaluate a new mixed-integer mathematical programming formulation for the Multiprocessor Scheduling Problem. Existing formulations involve a large number of binary variables with three indices. The formulation discussed here only uses binary variables with at most two indices. A set of valid inequalities is described. Numerical results of small and medium-scale problems are presented.

Conteúdo

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

Uma das mais importantes áreas de otimização aplicada consiste dos problemas de escalonamento, ou escalonamento, extensivamente estudados na literatura devido principalmente à sua aplicabilidade em quase todas situações do mundo real. Podemos modelar um problema de escalonamento em uma enorme gama de diferentes áreas, dentre elas: manufatura, logística, arquitetura de computadores e comunicações.

Na literatura, os problemas de escalonamento têm sido abordado em sua grande maioria atráves do uso de heurísticas e meta- heurísticas. O número de formulações matemáticas existentes é grande, mas não se compara ao enorme número de heurísticas existentes, já que elas proporcionam resultados razoáveis em um tempo satisfatório, além da facilidade de implementação que algumas delas oferecem. Porém, não devemos abrir mão do uso de formulações matemáticas, pois estas podem fornecer informações preciosas sobre determinado problema, como por exemplo limites inferiores e o valor da solução ótima. Destaco uma frase atribuída a A. Rinnooy Kan, onde ele diz: "Uma maneira natural de se atacar problemas de escalonamento é formulá-los como modelos de programação matemática."

Tendo isso em mente, o objetivo desta tese é estudar uma nova formulação proposta para um problema de escalonamento de tarefas em um sistema de multiprocessadores, onde o tempo gasto para transmitir dados entre dois processadores é considerado. O texto está organizado da seguinte forma: No capítulo 2 fazemos a introdução formal dos problemas de escalonamento e do problema de estudo deste trabalho. No Capítulo 3 descrevemos algumas formulações matemáticas para duas versões de problema de escalonamento em multiprocessadores. No Capítulo 4 apresentamos os resultados numéricos da nova formulação. No Capítulo 5 apresentamos resultados numéricos da nova formulação adaptada para o problema genérico de escalonamento em multiprocessadores. No último capítulo são expostas algumas conclusões e possíveis extensões e direcionamentos futuros.

Capítulo 2

O Problema de Scheduling

Neste capítulo fazemos a introdução ao problema genérico de escalonamento, descrevendo suas principais características, e descrevemos a variante dele para sistemas de multiprocessadores.

Na seção 2.1 fazemos uma introdução geral ao problema de escalonamento, descrevendo alguns conceitos básicos presentes em praticamente todas as variações do problema. Nas últimas duas seções tratamos especificamente dos problemas de *scheduling* em multiprocessadores, sendo que na seção 2.2 introduzimos o problema geral com multiprocessadores, enquanto que na seção 2.3 apresentamos uma variação do problema com multiprocessadores, problema este foco principal de estudo deste trabalho.

2.1 Introdução ao Problema de Escalonamento

Em geral, os problemas de escalonamento podem ser vistos como problemas de alocação de recursos sobre um determinado período de tempo, com o propósito de executar um conjunto de tarefas que faz parte de um ou mais processos. Essas tarefas competem individualmente por recursos que

podem ser de diferentes naturezas, como por exemplo, força humana, dinheiro, processadores ou máquinas, energia e ferramentas. As tarefas podem ser caracterizadas por: tempo de finalização, datas de término, pesos de urgência relativa e funções descrevendo o seu processamento. Também pode ser definida uma estrutura de um conjunto de tarefas refletindo restrições de precedência entre elas, além de diferentes critérios para medir a qualidade do desempenho do conjunto em questão.

Mais formalmente, podemos caracterizar os problemas de escalonamento por três conjuntos: conjunto $T = \{t_1, t_2, \ldots, t_n\}$ de n tarefas, conjunto $P = \{p_1, p_2, \ldots, p_m\}$ de m processadores (ou máquinas) e conjunto $V = \{v_1, v_2, \ldots, v_s\}$ de s tipos de recursos adicionais V. De forma geral, escalonamento significa atribuir processadores(ou máquinas) de P e possíveis recursos de V a tarefas de T, de forma a executar todas as tarefas de acordo com as restrições impostas. Um *schedule* pode ser entendido como um plano de programação do conjunto de tarefas, contendo o momento de execução de cada uma e o processador ou máquina onde será executada. Existem duas restrições gerais ne teoria clássica de escalonamento, as quais podem ser relaxadas, que dizem que cada tarefa deve ser processada por no máximo um processador em determinado período, e cada processador é capaz de processar no máximo uma tarefa por vez. A primeira delas pode ser relaxada, como será visto mais à frente.

Os processadores do conjunto P podem ser divididos em dois grupos: paralelos, que são aqueles que executam as mesmas funções, ou dedicados, que são aqueles especializados na execução de determinadas tarefas. Dentre o grupo dos processadores paralelos, podemos distinguir três tipos de acordo com a sua velocidade: *idênticos*, se todos possuem a mesma velocidade para executar as tarefas; *uniformes*, se possuem diferentes velocidades mas a ve-

locidade de cada processador é constante e não depende da tarefa em T; e finalmente *não-uniformes*, se a velocidade depende da tarefa particular a ser processada

Em geral, uma tarefa $t_j \in T$ pode ser caracterizada pelos seguintes dados:

(i) Vetor dos tempos de processamento $\Lambda_j = \{L_{1j}, \ldots, L_{mj}\}$, onde L_{ij} é o tempo necessário pelo processador p_i para processar t_j. No caso de processadores idênticos, temos que $L_{ij} = L_j$, $i = 1, 2, \ldots, m$.

(ii) Tempo de chegada v_j, que é o tempo necessário para que a tarefa t_j esteja pronta para ser processada. Se os tempos de chegada de todas as tarefas de T forem iguais, então assume-se que $v_j = 0 \ \forall j$.

(iii) Data de término d_j, que especifica um limite de tempo no qual a tarefa t_j deve ser completada. Geralmente, funções de penalidade são associadas a essas datas.

(iv) Prazo de término(*deadline*), que é um limite de tempo real mais rigoroso no qual a tarefa t_j deve ser completada.

(v) Peso, ou prioridade w_j, que expressa uma urgência relativa de t_j.

Apresentamos a seguir nos próximos parágrafos algumas definições sobre tarefas preemptivas, relações de precedência entre tarefas, plano de programação das tarefas(*schedules*) e critérios de otimalidade.

Um plano de programação é denominado *preemptivo* se cada tarefa pode ser interrompida em determinado momento e reiniciada posteriormente no mesmo ou em outro processador, sem nenhum custo. Caso nenhuma tarefa possa ser interrompida e reiniciada sem nenhum custo, então o plano de programação é chamado de *não-preemptivo*. Um relação de precedência entre duas tarefas t_i e t_j pertecentes ao conjunto T, por exemplo $t_i \prec t_j$, significa

que a tarefa t_j só pode ser processada após o término do processamento da tarefa t_i. Em outras palavras, uma relação de precedência $\prec$ é definida no conjunto T. As tarefas no conjunto T são chamadas *dependentes* se a ordem de execução de pelo menos duas delas é restringida por esta relação. Caso contrário, as tarefas são chamadas de *independentes*. Um conjunto de tarefas com relações de precedência é geralmente representado como um grafo acíclico direcionado, onde os nós representam as tarefas e os arcos as relações de precedência. Um exemplo é apresentado na figura 2.1.

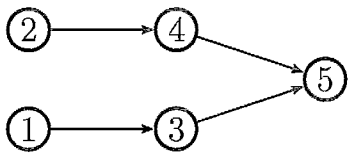


Figura 2.1: Exemplo de grafo de precedência

Mais formalmente, um plano de programação é basicamente uma atribuição de processadores do conjunto P (e possivelmente recursos do conjunto V) às tarefas do conjunto T em um determinado período de forma que as seguintes condições abaixo sejam satisfeitas:

- em todo momento cada processador está atribuído à no máximo uma tarefa e cada tarefa é processada por no máximo um processador;
- todas as tarefas são completadas;
- se as tarefas t_i, t_j estão na relação $t_i \prec t_j$, o processamento de t_j não é iniciado antes de t_i ter sido completada;

- no caso de um escalonamento não-preemptivo nenhuma tarefa é interrompida, caso contrário o número de interrupções de cada tarefa é finito;

- restrições de recursos, se houver, são satisfeitas.

No caso de um escalonamento determinístico, existem várias medidas de desempenho ou critérios de otimalidade que podemos usar, dentre eles:

- *Makespan.* O *makespan* é representado por W_{max} e é definido como o tempo que se leva para completar todas as n tarefas, ou seja, $W_{max} = max(W_1, \ldots, W_n)$.

- O tempo total ponderado de execução, representado por $\sum w_j W_j$.

- O máximo atraso, que é representado por A_{max}, e é formalmente definido como $A_{max} = max(A_1, \ldots, A_n)$, onde $A_j = W_j - d_j$.

A grande variedade de problemas de escalonamento existentes motivou a introdução de uma notação sistemática que pudesse servir de base para um esquema de classificação. Neste trabalho utilizamos a notação proposta por Graham et al. [10] e Blazewicz et al. [4], apresentada em [3], que é composta basicamente de três campos $\alpha \mid \beta \mid \gamma$. O primeiro campo α descreve as características do ambiente de processadores; o segundo campo β descreve as características das tarefa; o terceiro campo γ descreve o critério de otimalidade escolhido.

2.2 Escalonamento em Multiprocessadores

Programas paralelos podem ser representados por um conjunto de tarefas interrelacionadas, onde essas tarefas são basicamente unidades sequenciais. Quando um conjunto de processadores está intrinsicamente acoplado,

temos então um sistema de multiprocessadores. Em um sistema desse não só temos que determinar quantos, mas quais processadores devem ser alocados para uma aplicação, e mais especificamente, quais vão ser responsáveis por executar cada tarefa. Esse problema é conhecido na literatura como **MSP**(*Multiprocessor Scheduling Problem*), ou **Problema de Escalonamento em Multiprocessadores**, e tem sido estudado por mais de 40 anos por estar relacionado ao uso eficiente dos sistemas de multiprocessadores.

Os problemas de escalonamento em multiprocessadores são NP-Difícies na maioria dos casos(e.g. a minimização do *makespan* em dois processadores uniformes, problema $Q2 \mid\mid W_{max}$ na notação de [1], já pertence à classe NP-Difícil [8, 9]) e são geralmente resolvidos usando-se métodos heurísticos em todos os casos, com exceção dos mais simples. Este trabalho tem como foco principal as formulações matemáticas desenvolvidas para o **MSP** e seu variante **MSPCD**, a ser descrito mais a frente, de forma que deixaremos de lado as muitas heurísticas existentes para esses problemas em geral.

Diversas formulações matemáticas para problemas de escalonamento são apresentadas em um *survey* por Blazewics et al. [2], onde podemos destacar algumas formulações para problemas similares aos aqui discutidos. Vale citar uma formulação para o problema de escalonamento em processadores paralelos uniformes com tempos de processamento padrão unitários ($Q \mid p_j = 1 \mid \gamma$), que foi baseada em um caso especial do problema de transporte, e assim permite que o problema seja resolvido em tempo polinomial. É apresentada também uma formulação para o problema de escalonamento preemptivo em máquinas paralelas com critério de otimalidade de minimização do *makespan*($P \mid pmtn \mid W_{max}$), cujo número de variáveis depende poliniomialmente do tamanho da entrada quando o número de processadores m é fixo, podendo-se assim aplicar o procedimento de Karmarkar

para resolver o problema, resultando na abordagem de complexidade polinomial denominada método de uma-fase. Tal abordagem pode ser generalizada para cobrir os casos das máquinas serem uniformes($Q \mid pmtn \mid W_{max}$) ou não-uniformes($R \mid pmtn \mid W_{max}$).

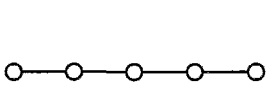
Maculan et al.[15] propuseram uma nova formulação para o **MSP** com relações de precedências entre as tarefas ($R|prec|W_{max}$), onde o número de variáveis binárias é polinomial, melhorando uma formulação de Blazewics et al.[4] baseada na discretização do horizonte de *schedule* em unidades de período de tempo, o que acarreta no uso de um número não-polinomial de varáveis 0-1. Esta formulação será descrita com detalhes no próximo capítulo, na seção 3.2.

Pablo et al.[5] abordam o mesmo problema e propõem uma nova formulação para o problema, visto que a formulação proposta em [15] não é indicada para resolver problemas de grande escala através do uso de técnicas padrão de programação inteira, como *branch-and-bound* e *branch-and-cut*. Nesta nova formulação, um subconjunto das inequações possue uma forte estrutura combinatorial e assim tornou possível a definição de um *politopo de partições em ordens lineares*, cuja investigação de sua estrutura facial resultou em inequações que definem facetas. Estas inequações foram usadas na implementação de um algoritmo de *branch-and-cut* com o qual foi possível resolver na otimalidade diversas instâncias da vida real, além de poderem ser usadas para melhorar formulações matemáticas de outras variantes de problemas de escalonamento em multiprocessadores. Outros dois conjuntos de inequações válidas são também propostos, dos quais podemos adaptá-los à formulação estudada neste trabalho, descrita no próximo capítulo na seção 3.3.2.

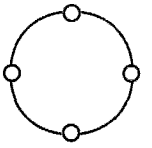
2.3 Escalonamento em Multiprocessadores com Atrasos de Comunicação

Nesta seção focaremos em uma versão ligeiramente modificada do problema apresentado na seção anterior, onde questões relacionadas à comunicação entre os processadores e sua arquitetura não foram levadas em conta.

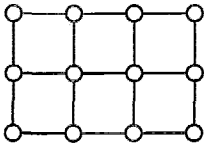
Com relação à arquitetura de um sistema de multiprocessadores, temos que as propriedades de uma rede de processadores com respeito ao desempenho de comunicação entre eles dependem fortemente de sua topologia, que pode ser: vetor linear, anel, malha, hipercubo, árvore, dentre outras, ilustradas na figura 2.2.



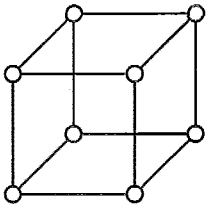
(a) Vetor Linear



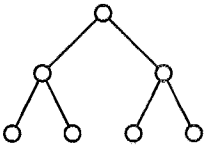
(b) Anel



(c) Malha



(d) Hipercubo 3D



(e) Árvore

Figura 2.2: Exemplos de topologia para sistemas de multiprocessadores

Já com respeito à comunicação entre os processadores, temos que tarefas executadas em diferentes processadores devem trocar dados, e consequentemente tais trocas introduzem atrasos de comunicação. Podemos lidar com esses atrasos de forma implícita, quando os tempos de comunicação já estão incluídos no tempo de execução das tarefas, ou explícita, onde podemos distinguir dois subcasos. No primeiro caso, cada tarefa requer somente um processador por vez e envia uma mensagem(e.g. resultados de sua execução) para todos seus sucessores imediatos após o término de sua execução. Se duas tarefas, t_j e t_k, estão em uma relação de precedência, i.e. $t_j \prec t_k$, então t_k provavelmente usará informações produzidas pela execução da tarefa t_j. Portanto, um atraso de comunicação somente irá ocorrer somente se ambas tarefas forem executadas em processadores diferentes. No segundo caso, trabalhamos com o conceito de tarefas divisíveis, que nada mais é que a alocação de uma tarefa divisível a vários processadores.

No paradigma de escalonamento determinístico, assumimos que os atrasos de comunicação são conhecidos de antemão, e que podem ser calculados através de alguns parâmetros, como a quantidade de dados a ser transmitida e a distância entre os processadores que executam o par de tarefas em questão.

Dessa forma, podemos definir em linhas gerais o **Problema de Escalonamento em Multiprocessadores com Atrasos de Comunicação** (*Multiprocessor Scheduling Problem With Communication Delays*, ou **MSPCD**) como o problema de escalonar tarefas dependentes em sistemas de multiprocessadores homogêneos, cujos processadores estão conectados arbitrariamente, contabilizando-se o tempo gasto na transferência de dados entre tarefas alocadas a processadores diferentes. Com respeito aos critérios de otimalidade existentes, neste trabalho vamos nos ater somente ao *makespan*. Portanto, de acordo com a notação apresentada em [3], o **MSPCD**

é descrito como $P, conn \mid prec, c_{jk} \mid W_{max}$, onde P representa processadores idênticos; *conn* indica que a topologia que os processadores estão conectados é arbitrária; *prec* indica a existência de restrições de precedência entre as tarefas; c_{jk} indica que se existe uma relação de precedência entre as tarefas j e k e elas são executadas em processadores diferentes, então ocorre um atraso de comunicação de de duração c_{jk} devido à transferência de dados; e finalmente W_{max} que representa o critério de otimalidade escolhido, que no caso é a minimização do *makespan*.

Tanja et al.[7] propuseram uma formulação para este problema baseada na formulação apresentada em [15]. Como a formulação original, esta apresenta a grande desvantagem de ser somente capaz de resolver problemas de pequena escala, e com o objetivo de tentar contornar essa limitação os autores utilizaram a heurística VNS(*Variable Neighborhood Search*) para o cálculo de limites superiores, e os métodos de balanço de carga e caminho crítico na tentativa de melhorar os limites inferiores. Com a introdução dos atrasos de comunicação, a modelagem matemática passou a ter um termo não-linear, produto de duas variáveis binárias. Além do conjunto de restrições adicionais padrão acrescidas à formulação com o propósito de linearizá-la, foi utilizada um outro conjunto de restrições apresentada em [13] como alternativa, chamado de *restrições de redução*. O uso deste outro conjunto de restrições reduziu o número de variáveis na formulação, melhorando o tempo e a qualidade da solução das instâncias testadas. Esta formulação é apresentada com mais detalhes no próximo capítulo, na seção 3.3.1.

Capítulo 3

Formulações Matemáticas

Neste capítulo apresentamos duas formulações matemáticas para o problema **MSPCD** e uma formulação para um problema correlato, denominado **MSP**. É interessante fazer um estudo um pouco mais detalhado dessas formulações para entendermos melhor a evolução delas até a forma mais recente, denominada de formulação *Packing*.

A primeira seção apresenta uma notação geral a ser usada por todas as formulações neste trabalho. A seção 3.2 apresenta uma formulação desenvolvida para o problema **MSP**, onde os processadores não são idênticos nem uniformes e não há atrasos de comunicação. Já a seção 3.3.1 apresenta uma formulação para o **MSPCD** baseada na primeira formulação apresentada, e finalmente na seção 3.3.2 é descrita a formulação mais recente para o **MSPCD**, a qual é objeto de estudo neste trabalho.

3.1 Notação Geral

Apresentamos abaixo uma notação básica usada pelas três formulações apresentadas neste capítulo. Esta notação é suficiente para descrevermos

a primeira formulação, porém necessita da introdução de alguns parâmetros para descrevermos as formulações para o **MSPCD**, além de termos de acrescentar novas variáveis de decisão e redefinir uma delas para a formulação *Packing*. Essas modificações serão dsecritas nas seções das formulações relativas ao **MSPCD**.

Definimos primeiramente os parâmetros e logo após as variáveis de decisão:

- T representa o conjunto parcialmente ordenado de tarefas, *i.e.* $T = \{t_1, \ldots, t_n\}$.
- P representa o conjunto de processadores, *i.e.* $P = \{p_1, \ldots, p_m\}$.
- E representa o conjunto das arestas que representam as ordens de precedência entre as tarefas, *i.e.* $E = \{e_{ij} \mid i, j \in T\}$.
- Λ representa o conjunto dos tempos de execução de cada tarefa em cada processador, *i.e.* $\Lambda = \{L_{11}, \ldots, L_{mn}\}$.
- $\delta^-(j)$ representa o conjunto dos predecessores imediatos da tarefa $j \in T$, *i.e.* $\delta^-(j) = \{i \in T \mid (i,j) \in E\}$.
- α coeficiente de penalidade suficientemente grande, *i.e.* $\alpha \gg 0$.

As seguintes variáveis de decisão são usadas:

- x_i representa o tempo de início de execução da tarefa $i \in T$.
- z^s_{jk} indica se a tarefa $j \in T$ é a s-ésima tarefa executada pelo processador $k \in P$.
- W_{max} representa o tempo total de execução de todas as tarefas, *i.e. makespan*.

3.2 Formulação para MSP

Esta formulação, descrita em [15], foi desenvolvida para o problema de escalonamento de tarefas em processadores não-idênticos com restrições de precedência.

Apesar desta formulação não tratar exatamente do **MSPCD**, julgamos importante dar-lhe um enfoque um pouco mais detalhado por dela ter sido estendida a primeira formulação para o **MSPCD**, descrita na seção 3.3.1.

O conjunto de tarefas a ser escolanado pode ser representado por um grafo acíclico direcionado de precedência definido pela tripleta $\mathcal{G} = (T, E, \Lambda)$. Apresentamos abaixo a formulação matemática para o **MSP**.

3.2.1 Formulação Matemática

$$(MSP) \qquad \min \; W_{max} \tag{3.1}$$

sujeito a:

$$\sum_{k=1}^{m}\sum_{s=1}^{n} z_{jk}^{s} \;=\; 1, \quad \forall j \in T \tag{3.2}$$

$$\sum_{j=1}^{n} z_{jk}^{1} \;\leq\; 1, \quad \forall k \in P \tag{3.3}$$

$$\sum_{j=1}^{n} z_{jk}^{s} \;\leq\; \sum_{j=1}^{n} z_{jk}^{s-1}, \quad \forall k \in P, \; \forall s = 2, \ldots, n \tag{3.4}$$

$$x_j \;\geq\; x_i + \sum_{k=1}^{m}\sum_{s=1}^{n} L_{ik} z_{ik}^{s}, \quad \forall i \in \delta^{-}(j), \; \forall j \in T \tag{3.5}$$

$$x_j \;\geq\; x_i + L_{ik} - \alpha \left[2 - \left(z_{ik}^{s} + \sum_{r=s+1}^{n} z_{jk}^{r} \right) \right],$$
$$\forall k \in T, \; \forall s = 1, \ldots, n-1, \; \forall \, (i,j) \; \in T \times T \tag{3.6}$$

$$W_{max} \geq x_j + \sum_{k=1}^{m} \sum_{s=1}^{n} L_{jk}\, z_{jk}^{s}, \quad \forall j \in T \tag{3.7}$$

$$z_{jk}^{s} \in \{0,1\}, \quad \forall j \in T, \quad \forall k \in P, \quad \forall s = 1, \ldots, n \tag{3.8}$$

$$x_j \geq 0, \quad \forall j \in T \tag{3.9}$$

$$W_{max} \geq 0 \in \mathbb{R} \tag{3.10}$$

A função objetivo dada por (3.1) procura minimizar o *makespan*. As equações (3.2) garantem que cada tarefa é atribuída a exatamente um processador, enquanto que as inequações (3.3-3.4) garantem que cada processador não vai executar simultaneamente mais de uma tarefa. Vale ressaltar que a tarefa $s + 1$ só será atribuída a um processador se a tarefa s já tiver sido. As inequações (3.5) expressam as relações de precedência entre as tarefas, as quais garantem que uma tarefa só pode ser executada se todas as suas predecessoras já o tiverem sido. As restrições (3.6) definem a sequência de tempo inicial de execução do conjunto de tarefas atribuídas a um mesmo processador, e finalmente, as restrições (3.7) definem o *makespan*.

3.3 Formulações para o MSPCD

Nesta seção apresentamos duas formulações para o **MSPCD**, sendo que a primeira delas desenvolvida por Tanja et al.[7] é uma adaptação da formulação descrita na seção anterior para o problema em questão, e por conseguinte apresenta as mesmas desvantagens já discutidas no capítulo anterior. Portanto, Liberti et al. [14] propuseram uma nova formulação para este problema, que procura contornar as desvantagens encontradas na primeira formulação.

Como o **MSPCD** é uma versão particular do **MSP**, devemos introduzir alguns conceitos novos bem como redefinir alguns já existentes, necessários

à modelagem deste problema. Como o conjunto de processadores é formado por processadores idênticos, podemos redefinir o conjunto dos tempos de execução das tarefas, que passa a ser $\Lambda = \{L_1, \ldots, L_n\}$, pois agora uma tarefa qualquer t_i leva o mesmo tempo para executar em todos processadores. Devemos definir o conjunto $C = \{c_{ij} \mid e_{ij} \in E\}$ que corresponde ao conjunto dos custos das arestas de comunicação, onde o custo c_{ij} denota a quantidade de dados a ser transmitida entre as tarefas $i \in T$ e $j \in T$ caso elas sejam executadas em processadores diferentes. Quando ambas as tarefas são executadas no mesmo processador, o custo de comunicação é igual a zero. Uma tarefa não pode ser executada a não ser que todas suas tarefas predecessoras tiverem sua execução completada e todos os dados relevantes estejam disponíveis. Para esta versão do problema aqui abordada, não é permitida a interrupção da execução das tarefas para reinício posterior além de execuções redundantes de tarefas, isto é, permitir a execução de uma mesma tarefa em mais de um processador.

Assumimos que os m processadores idênticos que compõem o sistema de multiprocessadores possuem suas próprias memórias locais. Os processadores se comunicam trocando mensagens em dois canais bidirecionais de igual capacidade, e sua arquitetura, ou topologia, é modelada usando-se uma *matriz de distância*. O elemento (k, l) da matriz de distância $D = [d_{kl}]_{m \times m}$ é igual ao número mínimo de ligações(ou *links*) ligando os processadores p_k e p_l, e portanto a matriz de distância é simétrica com os elementos da diagonal iguais a zero. Denominamos γ_{ij}^{kl} o tempo gasto para a troca de dados entre as tarefas $i \in T$ e $j \in T$ quando executadas nos processadores $k \in P$ e $l \in P$ respectivamente. Pode ser calculado da seguinte forma: $\gamma_{ij}^{kl} = c_{ij}\, d_{kl}\, ccr$, onde ccr representa a razão entre o tempo gasto para se transferir uma unidade de dado e o tempo gasto para realizar uma única operação computacional.

Finalmente, assumimos também que cada processador pertencente ao sistema de multiprocessadores possue unidades de processamento de E/S para todos os elos(ou *links*) de comunicação, de forma que cada processador possa efetuar simultaneamente a execução da tarefa e a comunicação necessária.

O conjunto de tarefas a ser escalonado pode ser representado por um grafo acíclico direcionado de precedência definido pela quadrupleta $\mathcal{G} = (T, E, C, \Lambda)$.

3.3.1 Primeira Formulação para MSPCD

Apresentamos a seguir a primeira formulação estudada para o **MSPCD**, baseada na formulação apresentada na seção anterior e descrita originalmente em [7].

Formulação Matemática

$$(MSPCD) \qquad \min_{z,x} \max_{j \in T} \{x_j + L_j\} \tag{3.11}$$

sujeito a:

$$\sum_{k=1}^{m} \sum_{s=1}^{n} z_{jk}^{s} = 1, \quad \forall j \in T \tag{3.12}$$

$$\sum_{j=1}^{n} z_{jk}^{1} \leq 1, \quad \forall k \in P \tag{3.13}$$

$$\sum_{j=1}^{n} z_{jk}^{s} \leq \sum_{j=1}^{n} z_{jk}^{s-1}, \quad \forall k \in P, \forall s \in \{2, \ldots, |T|\} \tag{3.14}$$

$$x_j \geq x_i + L_i + \sum_{k=1}^{m} \sum_{s=1}^{n} \sum_{l=1}^{m} \sum_{r=1}^{n} \gamma_{ij}^{kl} z_{ik}^{s} z_{jl}^{r},$$
$$\forall i \in \delta^{-}(j), \forall j \in T \tag{3.15}$$

$$x_j \geq x_i + L_i - \alpha \left[2 - \left(z_{ik}^s + \sum_{r=s+1}^{n} z_{jk}^r \right) \right],$$
$$\forall k \in P,\ \forall s \in \{1, \ldots, |T| - 1\},\ \forall i,\ j \in T \tag{3.16}$$

$$z_{jk}^s \in \{0, 1\},\ \forall j \leq T, s \in \{1, \ldots, |T|\}, \quad \forall k \in P \tag{3.17}$$

$$x_j \geq 0, \forall j \in T \tag{3.18}$$

As equações (3.12) garantem que cada tarefa é atribuída a exatamente um processador, enquanto que as inequações (3.13-3.14) garantem que cada processador não pode ser usado simultaneamente por mais de uma tarefa. As inequações (3.15) expressam as relações de precedência entre as tarefas junto com o tempo de comunicação necessário quando duas tarefas são executadas em processadores diferentes. As restrições (3.16) definem a sequência de tempo inicial de execução do conjunto de tarefas atribuídas a um mesmo processador.

3.3.2 Formulação *Packing*

Nesta seção descrevemos uma nova formulação proposta em [14] para o **MSPCD**. Esta nova formulação é baseada em uma idéia descrita em [11], onde o problema em questão é a alocação de espaço em um porto para os navios ancorarem.

Antes de apresentarmos a formulação matemática, faz-se necessário introdozir um novo conjunto de variáveis de decisão usado especificamente por este novo modelo programação matemática associadao ao **MSPCD**. As novas variáveis são descritas abaixo.

Notação Estendida

y_i representa o ID do processador onde a tarefa $i \in T$ será executada;

σ_{ij} indica a relação entre a ordem do término das tarefas $i \in T$ e $j \in T$, mais precisamente, se a tarefa i termina antes da tarefa j iniciar.

ϵ_{ij} indica a relação entre os índices dos processadores que executam as tarefas $i \in T$ e $j \in T$, mais precisamente, se o índice do processador que executará a tarefa i é estritamente menor que aquele que executará a tarefa j.

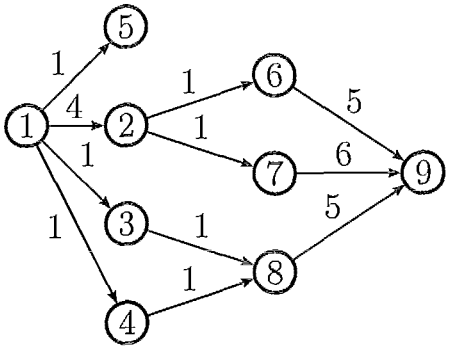
z_{ik} indica se a tarefa $i \in T$ é executada no processador $k \in P$.

Discutiremos abaixo a formulação matemática, onde inicialmente faremos uma apresentação sucinta da idéia da qual esta nova modelagem foi concebida e logo após descreveremos a formulação do modelo de programação matemática propriamente dita. Finalmente, descreveremos um conjunto de inequações válidas para esta formulação.

Formulação Matemática

O **MSPCD** pode ser representado por um diagrama espaço-tempo onde o eixo horizontal e o eixo vertical representam as unidades de tempo e os processadores respectivamente, e onde cada tarefa i a ser executada em um processador pode ser vista como um retângulo de tamanho L_i, e altura 1 (indicando que cada tarefa poderá ser executada somente por um processador). Uma solução para o problema consiste em "empacotar" todos os retângulos representando as tarefas dentro de um retângulo maior de altura $|P|$ e tamanho W_{max}. Como nosso objetivo é obter o menor *makespan* devemos então minimizar W_{max}, ou seja, devemos obter o menor retângulo capaz de empacotar todos os retângulos representando as tarefas.

Seja o grafo de precedência da figura 3.1 descrito em [12]. Vamos ilustrar uma solução para o **MSPCD** do grafo dado usando o diagrama espaço-tempo.



i	1	2	3	4	5	6	7	8	9
L_i	2	3	3	4	5	4	4	4	1

Figura 3.1: Grafo de precedência da instância *kwok*

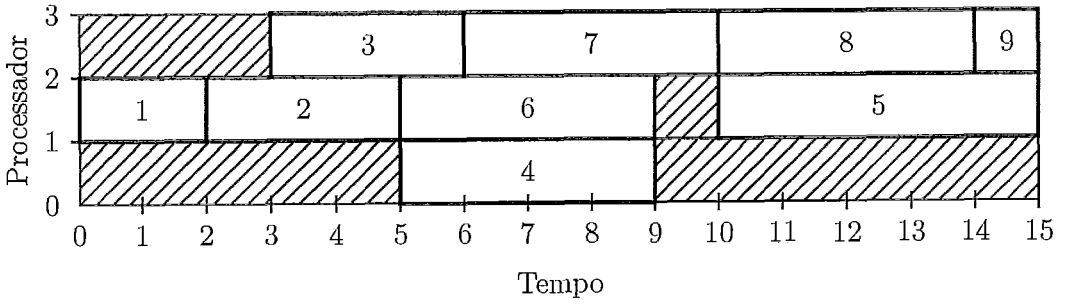


Figura 3.2: Diagrama espaço-tempo da solução ótima da instância *kwok*

Baseado nessa idéia, podemos interpretar a variável x_i como a coordenada do eixo horizontal da parte de baixo à esquerda do retângulo representando a tarefa i, e a variável y_i como a coordenada do eixo vertical da parte de baixo à esquerda do retângulo representando a tarefa i.

Também podemos dar uma outra interpretação para as variáveis binárias σ_{ij} e ϵ_{ij} :

$$\forall\, i,j \in T\ \sigma_{ij} \;=\; \begin{cases} 1 & \text{se o retângulo da tarefa } i \text{ está completamente à esquerda} \\ & \text{do retângulo da tarefa } j, \text{ e os dois retângulos não se} \\ & \text{sobrepõem em relação ao eixo horizontal} \\ 0 & \text{c.c.} \end{cases} \tag{3.19}$$

$$\forall\, i,j \in V\ \epsilon_{ij} \;=\; \begin{cases} 1 & \text{se o retângulo da tarefa } i \text{ está completamente abaixo} \\ & \text{do retângulo da tarefa } j, \text{ e os dois retângulos não se} \\ & \text{sobrepõem em relação ao eixo vertical} \\ 0 & \text{c.c.} \end{cases} \tag{3.20}$$

Sendo assim, apresentamos abaixo a nova formulação para **MSPCD**:

$$(MSPCD_{packing}) \qquad \min_{x,y,\sigma,\epsilon} W_{max} \tag{3.21}$$

sujeito a:

$$x_i + L_i \leq W_{max}, \forall\, i \in T \tag{3.22}$$

$$x_j - x_i - L_i - (\sigma_{ij} - 1)\alpha \geq 0 \quad ,\forall\, i \neq j \in T \tag{3.23}$$

$$y_j - y_i - 1 - (\epsilon_{ij} - 1)|P| \geq 0 \quad ,\forall\, i \neq j \in T \tag{3.24}$$

$$\sigma_{ij} + \sigma_{ji} + \epsilon_{ij} + \epsilon_{ji} \geq 1 \quad ,\forall\, i \neq j \in T \tag{3.25}$$

$$\sigma_{ij} + \sigma_{ji} \leq 1 \quad ,\forall\, i \neq j \in T \tag{3.26}$$

$$\epsilon_{ij} + \epsilon_{ji} \leq 1 \quad ,\forall\, i \neq j \in T \tag{3.27}$$

$$\sigma_{ij} = 1 \quad ,\forall\, j \in T\ :\ i \in \delta^-(j) \tag{3.28}$$

$$x_i + L_i + \sum_{h,k \in P} \gamma_{ij}^{hk} z_{ih} z_{jk} \leq x_j \quad ,\forall\, j \in V\ :\ i \in \delta^-(j) \tag{3.29}$$

$$\tag{3.30}$$

$$\sum_{k \in P} k z_{ik} = y_i \quad ,\forall\, i \in T \tag{3.31}$$

$$\sum_{k \in P} z_{ik} = 1 \quad ,\forall\, i \in T \tag{3.32}$$

$$W_{max} \geq 0 \tag{3.33}$$

$$x_i \geq 0 \quad ,\forall\, i \in T \tag{3.34}$$

$$y_i \in \{1, \ldots, |P|\}, \forall\, i \in T \tag{3.35}$$

$$z_{ik} \in \{0,1\} \quad ,\forall\, i \in T, k \in P \tag{3.36}$$

$$\sigma_{ij}, \epsilon_{ij} \in \{0,1\} \quad ,\forall\, i,j \in T \tag{3.37}$$

A função objetivo dada por (3.21) procura minimizar o *makespan*. A restrição (3.22) define o *makespan*, impondo que o valor de W_{max} tenha que ser pelo menos maior ou igual ao tempo de início de cada tarefa acrescido de seu tempo de execução. As restrições (3.23) definem a sequência de tempo de início do conjunto de tarefas atribuídas ao mesmo processador. Ela expressa o fato de que se as tarefas i e j são executadas no mesmo processador, então a tarefa j deve ser iniciada pelo menos L_i unidades de tempo depois do início da tarefa i. Caso as tarefas i e j não sejam atribuídas ao mesmo processador e sejam mutuamente independentes, ou seja, não existindo uma relação de precedência entre elas, o raciocínio anterior não se aplica, pois pode ocorrer que a tarefa j comece a executar antes de i terminar. Daí o uso do termo $(\sigma_{ij} - 1)W_{\max}$ em (3.23), onde $W_{\max}$ é um limite superior do *makespan*, garantindo que a inequação seja satisfeita. As restrições (3.24) asseguram que se duas tarefas são mutuamente independentes e se sobrepõem, elas devem ser atribuídas a processadores diferentes.

Vamos analisar as restrições (3.25) caso a caso: (i) caso as tarefas i e j sejam mutuamente independentes e se sobreponham(o que implica em

$\sigma_{ij} = \sigma_{ji} = 0$), então deve existir uma relação entre os ID's dos processadores que vão executá-las, e junto com a restrição (3.24), asseguramos que elas devem ser executadas em processadores diferentes; (ii) caso as tarefas não se sobreponham, então podemos afirmar que ou a tarefa i termina antes da tarefa j começar, ou vice-versa, o que implica $\sigma_{ij} = 1$ ou $\sigma_{ji} = 1$, o que já satisfaz as restrições independente da escolha dos processadores; (iii) caso as tarefas sejam dependentes, teremos uma relação de precedência pré-estabelecida entras elas ($i \rightarrow j$ ou $j \rightarrow i$), o que novamente implica que $\sigma_{ij} = 1$ ou $\sigma_{ji} = 1$, satisfazendo a restrição independente da escolha dos processadores que irão executar tais tarefas.

Os próximos dois conjuntos de restrições garantem que o modelo não fique inconsistente. As restrições (3.26) garantem que se a tarefa i termina antes da tarefa j, então j não poderá terminar antes de i (e vice-versa), enquanto que as restrições (3.27) garantem que se o ID do processador atribuído à tarefa i for estritamente menor que o ID do processador atribuído à tarefa j, então o ID do processador de j não pode ser estritamente menor que o ID do processador de i (e vice-versa).

As restrições (3.28) garantem que se existe um arco ligando as tarefas i e j no grafo G de precedência, então a tarefa j só pode ser executada depois que a tarefa i terminar, ou seja, $\sigma_{ij} = 1$, ao passo que a restrições (3.29) expressam as restrições de precedência entre as tarefas junto com o tempo de comunicação requerido para as tarefas atribuídas a processadores diferentes. Já as restrições (3.31) relacionam o ID do processador onde a tarefa i vai ser executada à variável y, e finalmente as restrições (3.32) garantem que cada tarefa seja atribuída a somente um processador.

O modelo descrito acima é um modelo de programação matemática misto não-linear, cujos únicos termos bilineares são $z_{ih}z_{jk}$ nas restrições (3.29), que

representam o atraso de comunicação. A constante α pode ser calculada da seguinte forma:

$$\alpha = \sum_{i \in T} L_i + \sum_{i,j \in T} c_{ij} \max\{d_{hk} \mid h, k \in P\}.$$

Podemos linearizar este modelo introduzindo um novo conjunto de variáveis contínuas $\zeta \in [0,1]$ as quais substituem os termos não-lineares citados acima, de forma que:

$$\zeta_{ij}^{hk} = z_{ih}\, z_{jk}, \quad \forall\, j \in T, i \in \delta^-(j), h, k \in P.$$

As variáveis ζ devem satisfazer as restrições de linearização abaixo, daqui para frente chamadas de restrições de linearização padrão, que garantem também a integralidade de ζ.

$$\left.\begin{array}{rcll} z_{ih} & \geq & \zeta_{ij}^{hk} & ,\forall\, j \in T, i \in \delta^-(j), h, k \in P \\ z_{jk} & \geq & \zeta_{ij}^{hk} & ,\forall\, j \in T, i \in \delta^-(j), h, k \in P \\ z_{ih} + z_{jk} - 1 & \leq & \zeta_{ij}^{hk} & ,\forall\, j \in T, i \in \delta^-(j), h, k \in P \end{array}\right\} \tag{3.38}$$

Podemos também utilizar um outro conjunto de restrições para linearizar o modelo, descrito em [13]. Este conjunto é denominado *restrições de redução* e é apresentado abaixo:

$$\zeta_{ij}^{hk} = \zeta_{ji}^{kh} \quad ,\forall\, i \neq j \in T, h, k \in P \tag{3.39}$$

$$\sum_{h \in P} \zeta_{ij}^{hk} = z_{jk}. \quad ,\forall\, i \neq j \in T, k \in P \tag{3.40}$$

A seguir, mostramos que uma reformulação do problema contendo o sistema de restrições de redução (ao invés das retrições de linearização (3.38)) é exata.

Proposição 3.3.1. *As restrições (3.32), (3.39) e (3.40) implicam as restrições de linearização (3.38).*

Prova. Por (3.39) e (3.40) temos

$$\zeta_{ij}^{hk} \leq z_{ih} \quad \text{e} \quad \zeta_{ij}^{hk} \leq z_{jk} \tag{3.41}$$

Por (3.41), temos que para qualquer conjunto J de índices $f \in P$, $\sum_{f\in J} \zeta_{ij}^{fk} \leq \sum_{f\in J} z_{if}$. Escolha um $h \in P$ e considere o conjunto $J = P\backslash\{h\}$. Para cada $i, j \in T$ e $k \in P$, temos:

$$\sum_{f\in J} z_{if} \geq \sum_{f\in J} \zeta_{ij}^{fk} \Rightarrow \text{(adicione e subtraia } \zeta_{ij}^{kl})$$

$$\sum_{f\in J} z_{if} \geq \sum_{f\in J} \zeta_{ij}^{fk} + \zeta_{ij}^{hk} - \zeta_{ij}^{hk}$$

$$\sum_{f\in J} z_{if} \geq \sum_{f\in P} \zeta_{ij}^{fk} - \zeta_{ij}^{hk} \Rightarrow \text{(substitua } \textstyle\sum_{f\in P} \zeta_{ij}^{fk} \text{ por } z_{jk})$$

$$\sum_{f\in J} z_{if} \geq z_{jk} - \zeta_{ij}^{hk} \Rightarrow \text{(some e subtraia } z_{ih})$$

$$\sum_{f\in J} z_{if} + z_{ih} - z_{ih} \geq z_{jk} - \zeta_{ij}^{hk} \rightarrow \sum_{f\in P} z_{if} - z_{ih} \geq z_{jk} - \zeta_{ij}^{hk}$$

$$\zeta_{ij}^{hk} \geq z_{jk} + z_{ih} - \sum_{f\in P} z_{if} \Rightarrow \text{(substitua } \textstyle\sum_{f\in P} z_{if} \text{ por (3.32))}$$

$$\zeta_{ij}^{hk} \geq z_{jk} + z_{ih} - 1$$

□

Inequações Válidas

Nesta seção descrevemos dois conjuntos de inequações válidas que podem ser usadas com a formulação apresentada na seção anterior. Elas foram descritas originalmente em [5], e foram adaptadas para a formulação *Packing* aqui descrita.

Dado o grafo acíclico direcionado $\mathcal{G} = (T, E, C, \Lambda)$, definimos os seguintes conjuntos para cada tarefa $j \in T$:

- $S_j = \{i \in T$: existe um caminho em $\mathcal{G}$ de i para $j\}$, *i.e*, S_j é o conjunto dos predecessores da tarefa i;
- $Q_j = \{i \in T$: existe um caminho em $\mathcal{G}$ de j para $i\}$, *i.e*, Q_j é o conjunto de sucessores da tarefa j;
- $R_j = \{i \in T$: não existe caminho em $\mathcal{G}$ de i para j ou de j para $i\}$.

(i) **Inequações dos Sucessores.** Seja uma tarefa j e um processador k. Estas inequações dizem que a soma dos tempos de processamento de todas as tarefas do conjunto de sucessores de j que são atribuídas ao processador k mais o tempo de se completar a tarefa j não pode exceder o *makespan*. Elas são apresentadas abaixo:

$$x_j + L_j + \sum_{i \in Q_j} L_j z_{ik} \leq W_{max} \quad \forall j \in T, k \in P \tag{3.42}$$

(ii) **Inequações dos Predecessores.** O objetivo destas inequações é levar em conta o tempo total de processamento das tarefas atribuídas ao processador k ao se computar o tempo de início de execução da tarefa j. Se j for também atribuída ao processador k, seu tempo de início é no mínimo igual à soma dos tempos de processamento de todas as tarefas já atribuídas ao processador k. Esta classe de inequações é dada por:

$$\sum_{i \in S_j} L_i(z_{jk} + z_{ik} - 1) \quad +$$

$$\sum_{i \in R_j} L_i(z_{jk} + z_{ik} - \sigma_{ji} - 1) \quad \leq \quad x_j \quad \forall j \in T,\, k \in P \qquad (3.43)$$

Lema 3.1. *A inequação (3.43) é válida.*

Prova. Vamos assumir que $i \in S_j$ e ambas as tarefas i e j são executadas no processador k. Neste caso, o valor do primeiro somatório é L_i enquanto que em qualquer outro caso este valor é não-positivo. No segundo somatório, quando i está em R_j, se a tarefa j não é executada no processador k, então $z_{jk} = 0$ e $z_{jk} + z_{ik} - \sigma_{ji} - 1 \leq 0$ o que torna a inequação redundante. Por outro lado, se a tarefa j é executada no processador k, então $z_{jk} = 1$ e as três situações abaixo podem ocorrer:

(a) Se i é executada no processador k antes de j, então $z_{jk}+z_{ik}-\sigma_{ji}-1 = 1$, e a contribuição da tarefa i para a soma é L_i.

(b) Se i é executada no processador k e depois de j, então $z_{jk} + z_{ik} - \sigma_{ji} - 1 = 0$, e a tarefa i não contribui para a soma.

(c) Se i não é executada no processador k, então $z_{jk}+z_{ik}-\sigma_{ji}-1 \leq 0$ o que torna a inequação redundante.

Os argumentos acima garantem a validade da inequação. ∎

Vamos explorar mais um pouco o argumento do item (c), para explicar o motivo da inequação descrita não poder ser uma igualdade. Caso as

tarefas i e j sejam sobrepostas , isto é $\sigma_{ij} + \sigma_{ji} = 0$, então teremos que $z_{jk}+z_{ik}-\sigma_{ji}-1 = 0$. Já se tivermos $\sigma_{ji} = 1$, teremos que $z_{jk}+z_{ik}-\sigma_{ji}-1 < 0$, ou seja, um termo que contribui negativamente na inequação, tornando-a por conseguinte menos eficiente. Isso acontece porque a variável σ_{ji} não garante que as tarefas i e j sejam executadas na mesma máquina. Para contornar esse problema, devemos multiplicar a variável σ_{ji} pela variável z_{jk}, de forma que a inequação fique da seguinte forma:

$$\begin{aligned} \sum_{i \in P_j} L_i(z_{jk} + z_{ik} - 1) \quad + \\ \sum_{i \in R_j} L_i(z_{jk} + z_{ik} - \sigma_{ji} z_{jk} - 1) \quad \leq \quad x_j \quad \forall j \in T,\, k \in P \end{aligned} \tag{3.44}$$

Novamente, temos um termo não-linear ($\sigma_{ji} z_{jk}$) dado pelo produto de duas variáveis binárias, e portanto devemos linearizar (3.44) da mesma forma feita com as restrições (3.29), criando uma nova variável e acrescentando os mesmos tipos de restrições usadas em (3.38). Dessa forma, temos novamente uma formulação composta somente de termos lineares.

Apesar de agora não ser possível que em determinados casos o conjunto de inequações deixe de atuar da forma mais eficiente, testes preliminares mostraram o tempo de solução dos problemas pelo CPLEX piorou, se considerarmos o tempo gasto com a formulação sem as inequações e com as inequações descritas inicialmente. Isso se deve principalmente à adição do novo termo não-linear, e consequentemente uma nova variável e um novo conjunto de restrições, o que devido à linearização acarretou em aumento do tamanho da matriz usada em cada iteração do simplex e consequentemente o processo de resolução ficou mais lento. Como o ganho esperado não foi

alcançado, optamos por usar os conjuntos de inequações na forma em que foram apresentados de início.

Capítulo 4

Resultados Computacionais

Neste capítulo apresentamos os resultados computacionais obtidos ao resolvermos instâncias do **MSPCD** utilizando a formulação *Packing*. Descrevemos as instâncias de teste na próxima seção, e os detalhes de implementação na seção 4.2. Finalmente, fazemos um resumo geral dos resultado apresentados neste capítulo na seção 4.3

4.1 Descrição das instâncias

Procuramos utilizar um grande número de instâncias de teste com diferentes características cada, de forma que possamos avaliar melhor o comportamento desta nova formulação mediante a variação de parâmetros do grafo de tarefas representando a instância. Podemos enumerar alguns parâmetros relevantes de acordo com [7] :

- n - número de tarefas;
- ρ - densidade das arestas do grafo de tarefas;
- f_p - nível de paralelismo, isto é, valor médio de tarefas mutuamente independentes;

- f_c - razão entre o tempo de computação e o tempo de comunicação, representado pelo parâmetro γ na formulação usada.

Outro parâmetro muito importante é a arquitetura da máquina com multiprocessadores, dado pelo número de processadores p e a matriz de distância D (ver seção 3.3 descrevendo a rede de interconexão entre os processadores). Das diversas arquiteturas de processadores existentes, resolvemos usar a arquitetura onde todos os processadores estão completamente conectados, e a arquitetura hipercubo 2D quando possível, por ela ser amplamente utilizada. O valor para f_p é calculado baseado no número de processadores, número de níveis do grafo de precedência e o número médio de tarefas por nível. O parâmetro f_c é calculado da seguinte forma:

$$f_c = \frac{\sum_j L_j}{\sum_i \sum_j c_{ij}} \tag{4.1}$$

Abaixo apresentamos as matrizes de distância para as duas arquiteturas usadas e seus respectivos números de processadores:

$$D = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, \text{ para } p = 2; \qquad D = \begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix}, \text{ para } p = 3;$$

$$D = \begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix}, \text{ para } p = 4,$$

para a arquitetura onde os processadores estão completamente conectados, e

$$D = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, \text{ para } p = 2; \qquad D = \begin{bmatrix} 0 & 1 & 1 & 2 \\ 1 & 0 & 2 & 1 \\ 1 & 2 & 0 & 1 \\ 2 & 1 & 1 & 0 \end{bmatrix}, \text{ para } p = 4,$$

para a arquitetura hipercubo de uma e duas dimensões respectivamente.

A densidade ρ do grafo de tarefas também mostrou ser um parâmetro importante a considerar de acordo com [7], e como veremos mais a frente ao analisarmos as tabelas dos testes. Dessa forma, procuramos selecionar instâncias com o mesmo número de tarefas, processadores e arquitetura, mas com densidades diferentes.

Em nossos testes utilizamos três grupos de instâncias diferentes: algumas instâncias pequenas já conhecidas na literatura, instâncias geradas aleatoriamente com valor de solução ótima conhecido (instâncias com prefixo *ogra* de *optimal graphs*) e instâncias geradas aleatoriamente com o valor da solução ótima desconhecido (instâncias cujo nome se iniciam com a letra "t"). O grupo de instâncias *ogra* possui uma característica interessante que é o fato de independente da densidade do grafo e do número de processadores, o valor da solução ótima é sempre o mesmo. O gerador dos dois últimos grupos de instâncias é descrito em [6].

4.2 Detalhes de Implementação

Descrevemos detalhadamente nesta seção a implementação realizada para testarmos as instâncias descritas na seção anterior. O modelo matemático *Packing* foi implementado usando a linguagem AMPL (*A Modeling Language for Mathematical Programming*) que por sua vez chamava o software CPLEX

9.1 para resolver a instância em questão. Os testes foram conduzidos em uma máquina com quatro processadores Intel *Xeon* cada um com 2.80 GHz e com uma memória total de 4 GB de RAM, e o limite de tempo definido para todos os testes foi de um dia. Vale ressaltar que cada processo do CPLEX ocupava somente um processador, não havendo em hipótese alguma a possibilidade de processamento paralelo de um processo do resolvedor.

Com o propósito de se obter dados para melhor analisar a nova formulação, foi decidido variar alguns parâmetros importantes, como o tipo de linearização usado, acréscimo de inequações válidas e arquitetura do sistema de multiprocessadores. As duas primeiras variações citadas dão-se na formulação matemática, ou seja, o modelo em si é alterado, enquanto que a última dá-se em um determinado conjunto de parâmetros(mais especificamente na matriz de distâncias D).

Para o uso das inequações válidas, faz-se necessário definir novos conjuntos (ver seção 3.3.2 do capítulo 3). A fim de obtermos os dados desses novos conjuntos, precisamos calcular o fecho transitivo do grafo de precedência, então implementamos o algoritmo de Warshall na propria linguagem AMPL, pois era o mais simples de se implementar. Uma desvantagem dele é sua ordem de complexidade, que é de $O(|V|^3)$, onde $|V|$ é o número de vértices do grafo. Isso pode afetar o tempo de solução de instâncias com um número de tarefas significativo, então por isso decidimos usar o tempo gasto pelo CPLEX para resolver determinado problema, ao invés de usar o tempo total, pois nosso objetivo aqui é simplesmente avaliar o impacto das inequações na formulação.

Com relação a análise dos resultados, vamos focar na comparação dos resultados obtidos usando-se a formulação normal, ou seja, sem as inequações, e a formulação com as inequações. O parâmetro principal de comparação é

o tempo gasto na solução das instâncias, e quando os tempos forem iguais, fato comum no grupo de instâncias com 30 e 40 tarefas, o parâmetro usado será o valor do *gap* de dualidade.

As tabelas de resultados mostradas nas próximas duas seções apresentam todas o mesmo formato. A primeira e a segunda coluna contém respectivamente os nomes das instâncias de teste e o número de processadores na arquitetura usada. As próximas três colunas: número de tarefas n, densidade ρ do grafo de precedência e a razão entre o tempo de comunicação e o tempo de computação f_c. As próximas seis colunas dizem respeito aos dados da solução da instância, onde a letra subscrita o identifica o uso da formulação em sua forma normal, enquanto a letra subscrita l identifica o uso na formulação das inequações válidas. São elas: valor da melhor solução inteira encontrada dentro do limite de tempo estipulado, tempo de solução gasto pelo resolvedor CPLEX e finalmente, o *gap* de dualidade existente ao fim do processo de solução.

As próximas duas seções apresentam os resultados obtidos usando a arquitetura completamente conectada e a arquitetura hipercubo respectivamente. Já na seção 4.7, apresentamos os valores da relaxação linear de todas as instâncias testadas usando a arquitetura completamente conectada, além de algumas instâncias com mais de 100 tarefas.

4.2.1 Arquitetura Completamente Conectada

Apresentamos duas tabelas de resultados de testes rodados assumindo-se que todos os processadores estão conectados entre si, o que se configura na denominada arquitetura completamente conectada. A tabela 4.1 apresenta dos dados dos testes rodados usando a linearização padrão, enquanto que a tabela 4.2 apresenta dos dados dos testes rodados usando a linearização

denominada restrições de redução (*reduction constraints*). Abaixo fazemos uma análise dos resultados apresentados em cada uma das tabelas.

Na tabela 4.1, pode-se observar rapidamente que nas 17 primeiras instâncias testadas, todas elas foram resolvidas, e a grande maioria em um período de tempo inferior a um minuto. Isso deve-se principalmente ao baixo número de tarefas, que variaram entre nove e 12, e ao uso de no máximo três processadores na maioria das instâncias, salvo todas instâncias iniciadas pela letra *t*. A diferença no tempo gasto para resolver essas instâncias usando a formulação normal e a formulação com inequações chega a ser irrisória, onde na maioria delas a diferença é de apenas alguns segundos, e em duas delas a formulação normal demorou um minuto a mais para resolver. Já nas instâncias iniciadas por *t20*, o uso das inequações válidas fez com que o tempo de solução fosse menor em todas elas, com destaque para as instâncias **t20_25**, cuja diferença de tempo foi de 35 minutos, e **t20_75**, cuja diferença de tempo foi ligeiramente superior a duas horas.

No grupo de instâncias com 30 tarefas, temos que em seis delas o *gap* foi menor usando as inequações válidas, em outras seis ele foi igual a zero usando-se as formulações normal e a com inequações, e em somente uma o *gap* foi ligeiramente maior com o uso das inequações(diferença menor que 1%). Dentre as instâncias cujo *gap* foi menor, a diferença de valores do *gap* foi em média de 2%, e podemos destacar o fato das instâncias **t30_30_2** e **t30_60_2** terem sido resolvidas, o que não aconteceu quando foi usada a formulação em sua forma normal. No caso das instâncias que foram resolvidas com ambas versões de formulação, temos que em cinco casos o tempo de solução com o uso das inequações foi menor, onde podemos destacar as instâncias **t30_75_1** com diferença de tempo de 18 horas, **t30_90_1** com diferença de seis horas

e a instância **t30_60_1** com diferença de quatro horas. No único caso que o tempo foi pior, a diferença foi de apenas quatro minutos.

Finalmente, no grupo de instâncias com 40 tarefas podemos observar que somente em um caso foi possível obter a solução ótima, fato ligado ao uso das inequações válidas. Vale ressaltar que nesse caso o valor da solução ótima foi encontrado usando-se a formulação padrão, contudo sua otimalidade não foi provada dentro do tempo estipulado. Analisando os valores de *gap* obtidos, nota-se que em oito instâncias ele foi menor com o uso das inequações, e em outras cinco instâncias ele foi maior. Em ambos os casos, a diferença dos valores do *gap* variou entre 1% e 3%.

Tabela 4.1: Tabela comparativa de resultados para a arquitetura completamente conectada usando a linearização padrão. Tempo de CPU expresso em [horas]:minutos:segundos.[centésimos]

Instância	p	n	ρ	f_c	x_o^*	x_l^*	CPU_o	CPU_l	Gap_o	Gap_l
kwok	3	9	33.33	1.00	15	15	00:06.87	00:01.19	0.00%	0.00%
sih91	2	8	25.00	2.00	16	16	00:00.20	00:00.24	0.00%	0.00%
ss91	2	9	41.67	3.28	250	250	00:00.00	00:00.13	0.00%	0.00%
mt91	2	10	31.11	3.24	53	53	00:00.40	00:00.56	0.00%	0.00%
ogra9_30	3	9	13.88	1.46	20	20	00:00.00	00:01.69	0.00%	0.00%
ogra9_60	3	9	27.78	1.46	20	20	00:00.62	00:01.19	0.00%	0.00%
ogra9_90	3	9	41.67	1.05	20	20	00:00.86	00:01.05	0.00%	0.00%
ogra10_30	3	10	13.33	3.00	25	25	01:41.32	00:51.51	0.00%	0.00%
ogra10_60	3	10	26.67	1.23	25	25	00:01.59	00:01.44	0.00%	0.00%
ogra10_90	3	10	40.00	0.86	25	25	00:01.18	00:01.24	0.00%	0.00%
ogra12_20	3	12	7.58	4.54	19	19	00:00.01	00:02.14	0.00%	0.00%
gra12	3	12	16.7	3.23	135	135	19:09.12	18:06.46	0.00%	0.00%
t10_10	4	10	8.89	2.29	71	71	00:20.15	00:01.07	0.00%	0.00%
t10_25	4	10	28.9	0.84	60	60	00:09.31	00:04.08	0.00%	0.00%
t10_50	4	10	46.66	0.23	112	112	00:09.50	00:06.09	0.00%	0.00%
t10_75	4	10	80.0	0.22	105	105	00:02.81	00:00.46	0.00%	0.00%
t10_90	4	10	86.67	0.22	102	102	00:04.79	00:09.10	0.00%	0.00%
t20_10	4	20	8.95	1.57	110	110	00:29.83	00:18.04	0.00%	0.00%
t20_25	4	20	23.68	0.42	151	151	38:09.38	03:24.38	0.00%	0.00%
t20_50	4	20	53.16	0.24	221	221	03:24.47	02:02.86	0.00%	0.00%
t20_75	4	20	76.32	0.15	241	241	11:59:29	09:51:30	0.00%	0.00%
t20_90	4	20	80.52	0.12	242	242	01:26.84	01:05.44	0.00%	0.00%

continua na próxima página

Tabela 4.1: Tabela comparativa de resultados para a arquitetura completamente conectada usando a linearização padrão(continuação)

Instância	p	n	ρ	f_c	x_o^*	x_l^*	CPU_o	CPU_l	Gap_o	Gap_l
t30_10_1	4	30	9.88	0.79	137	137	24:00:00	24:00:00	7.30%	5.11%
t30_10_2	4	30	8.96	0.82	155	152	10:48:27	24:00:00	28.33%	25.00%
t30_25_1	4	30	22.29	0.50	209	209	24:00:00	24:00:00	5.26%	3.35%
t30_25_2	4	30	22.98	0.29	282	285	24:00:00	24:00:00	4.14%	1.05%
t30_30_1	4	30	30.34	0.34	414	414	00:19:45	00:02:05	0.00%	0.00%
t30_30_2	4	30	30.57	0.22	262	262	24:00:00	11:16:51	0.38%	0.00%
t30_50_1	4	30	53.10	0.18	370	370	00:02:25	00:06:11	0.00%	0.00%
t30_50_2	4	30	47.58	0.14	470	470	24:00:00	24:00:00	7.12%	7.59%
t30_60_1	4	30	56.32	0.21	467	467	06:44:23	02:33:26	0.00%	0.00%
t30_60_2	4	30	62.98	0.10	451	445	24:00:00	05:29:44	5.31%	0.00%
t30_75_1	4	30	75.86	0.14	544	544	23:11:21	04:37:50	0.00%	0.00%
t30_75_2	4	30	74.02	0.09	607	607	24:00:00	24:00:00	5.52%	3.73%
t30_90_1	4	30	89.19	0.10	562	562	08:58:31	02:47:50	0.00%	0.00%
t30_90_2	4	30	88.27	0.08	697	697	00:31:16	00:18:24	0.00%	0.00%
t40_10_1	4	40	12.43	0.65	233	233	24:00:00	24:00:00	2.15%	2.15%
t40_10_2	4	40	9.35	0.58	236	245	24:00:00	24:00:00	22.88%	25.31%
t40_25_1	4	40	25.00	0.33	270	270	24:00:00	01:24:08	1.11%	0.00%
t40_25_2	4	40	25.25	0.20	355	369	24:00:00	24:00:00	21.29%	24.39%
t40_30_1	4	40	29.74	0.24	463	457	24:00:00	24:00:00	9.22%	7.98%
t40_30_2	4	40	29.61	0.17	399	397	24:00:00	24:00:00	11.52%	10.58%
t40_50_1	4	40	46.92	0.14	501	505	24:00:00	24:00:00	19.76%	20.24%
t40_50_2	4	40	51.92	0.09	606	590	24:00:00	24:00:00	7.75%	4.04%
t40_60_1	4	40	60.76	0.15	776	754	24:00:00	24:00:00	2.57%	3.21%
t40_60_2	4	40	58.97	0.08	635	615	24:00:00	24:00:00	17.64%	14.63%
t40_75_1	4	40	73.84	0.12	776	777	24:00:00	24:00:00	0.97%	2.08%
t40_75_2	4	40	74.23	0.06	777	772	24:00:00	24:00:00	12.80%	12.37%
t40_90_1	4	40	88.58	0.09	694	693	24:00:00	24:00:00	7.90%	6.61%
t40_90_2	4	40	88.71	0.04	601	601	24:00:00	24:00:00	4.15%	2.60%

Na tabela 4.2, novamente pode-se observar que as 17 primeiras instâncias foram resolvidas em um pequeno período de tempo, com exceção de uma delas. Diferentemente do caso da formulação com a linearização padrão, aqui temos que nas 12 primeiras instâncias o tempo de solução com o uso das inequações válidas foi maior, onde podemos destacar os casos da instância **gra12**, cuja diferença de tempo foi de quatro horas, e da instância **ogra10_30**,

cuja diferença de tempo foi de 12 minutos. No restante das instâncias, a diferença foi da ordem de segundos. Já nas instâncias iniciadas por *t20*, novamente o uso das inequações válidas fez com que o tempo de solução fosse menor em todas elas, com destaque para as instâncias **t20_25**, cuja diferença foi de duas horas, e **t20_75**, cuja diferença foi de uma hora e meia aproximadamente.

No grupo de instâncias com 30 tarefas, temos que em cinco delas o *gap* foi menor usando as inequações válidas, em outras seis ele foi igual a zero usando-se as formulações normal e a com inequações, e em outras três foi maior com o uso das inequações. Dentre as instâncias cujo *gap* foi menor, podemos destacar o caso cuja diferença foi de quase 9% e dois outros casos cuja diferença foi de 5% aproximadamente. No caso das instâncias que foram resolvidas em ambas versões de formulação, temos que em quatro casos o tempo de solução com o uso das inequações foi menor, onde podemos destacar as instâncias **t30_30_1**, **t30_50_1** e **t30_75_1** cujos tempos com as inequações diferem dos tempos obtidos sem as inequações em um pouco mais de três horas dos nos dois primeiros casos e aproximadamente duas horas no último. Nos casos cujo tempo de solução foi maior, a diferença de tempo foi apenas de no máximo dois minutos. Nas três instâncias restantes cujo valor do *gap* foi maior com o uso das inequações, a diferença de seus valores não chegou a 2%. Um fato interessante que ocorreu nesse grupo de instâncias foi a instância **t30_60_2**, que foi resolvida com a formulação normal, mas não foi resolvida com o acréscimo das inequações. O contrário ocorreu com a instância **t30_25_2**, cujo tempo de solução foi razoavelmente baixo se comparado às outras instâncias do grupo. Vale ressaltar que em ambos os casos, apesar do *gap* não ter chegado a zero ao usar ou não as inequações, o valor

da solução encontrado era o valor da solução ótima, ou seja, somente não foi possível provar a otimalidade dentro do limite de tempo.

Já no grupo de instâncias com 40 tarefas, destacamos o fato de duas instâncias terem sido resolvidas com o uso das inequações válidas, sendo que em uma delas o valor da solução ótima foi encontrado ao usar a formulação normal, mas a otimalidade não foi provada. Analisando-se os valores de *gap* desse grupo, temos que em sete instâncias o valor dele foi menor com o uso das inequações, e nas outras sete o valor foi maior. Em ambos os casos, a diferença nos valores do *gap* com a formulação normal e com a formulação com as inequações foi em média de 1%, com exceção de alguns casos onde a diferença chegou a 5% e 8%.

Tabela 4.2: Tabela comparativa de resultados para a arquitetura completamente conectada usando as restrições de redução. Tempo de CPU expresso em [horas]:minutos:segundos.[centésimos]

Instância	p	n	ρ	f_c	x_o^*	x_l^*	CPU_o	CPU_l	Gap_o	Gap_l
kwok	3	9	33.33	1.00	15	15	00:00.40	00:01.66	0.00%	0.00%
sih91	2	8	25.00	2.00	16	16	00:00.05	00:00.12	0.00%	0.00%
ss91	2	9	41.67	3.28	250	250	00:00.07	00:00.13	0.00%	0.00%
mt91	2	10	31.11	3.24	53	53	00:00.12	00:00.52	0.00%	0.00%
ogra9_30	3	9	27.78	1.00	20	20	00:00.90	00:05.30	0.00%	0.00%
ogra9_60	3	9	27.78	1.46	20	20	00:00.15	00:02.21	0.00%	0.00%
ogra9_90	3	9	41.67	1.05	20	20	00:00.17	00:01.20	0.00%	0.00%
ogra10_30	3	10	13.33	3.00	25	25	00:58.12	13:36.53	0.00%	0.00%
ogra10_60	3	10	26.67	1.23	25	25	00:00.27	00:02.25	0.00%	0.00%
ogra10_90	3	10	40.00	0.86	25	25	00:00.14	00:02.00	0.00%	0.00%
ogra12_20	3	12	7.58	4.54	19	19	00:00.88	00:04.47	0.00%	0.00%
gra12	3	12	16.7	3.23	135	135	02:53:16	06:35:02	0.00%	0.00%
t10_10	4	10	8.89	2.29	71	71	00:04.64	00:00.04	0.00%	0.00%
t10_25	4	10	28.9	0.84	60	60	00:04.38	00:02.93	0.00%	0.00%
t10_50	4	10	46.66	0.23	112	112	00:01.81	00:04.22	0.00%	0.00%
t10_75	4	10	80.0	0.22	105	105	00:00.07	00:00.04	0.00%	0.00%
t10_90	4	10	86.67	0.22	102	102	00:00.56	00:00.38	0.00%	0.00%
t20_10	4	20	8.95	1.57	110	110	03:53.49	00:35.95	0.00%	0.00%
t20_25	4	20	23.68	0.42	151	151	02:45:03	23:14.97	0.00%	0.00%
t20_50	4	20	53.16	0.24	221	221	01:53.40	01:03.79	0.00%	0.00%

continua na próxima página

Tabela 4.2: Tabela comparativa de resultados para a arquitetura completamente conectada usando as restrições de redução(continuação)

Instância	p	n	ρ	f_c	x_o^*	x_l^*	CPU_o	CPU_l	Gap_o	Gap_l
t20_75	4	20	76.32	0.15	241	241	02:18:46	00:44:39	0.00%	0.00%
t20_90	4	20	80.52	0.12	242	242	01:07.10	00:13.98	0.00%	0.00%
t30_10_1	4	30	9.88	0.79	150	138	24:00:00	24:00:00	16.00%	7.10%
t30_10_2	4	30	8.96	0.82	160	155	24:00:00	24:00:00	31.25%	26.70%
t30_25_1	4	30	22.29	0.50	209	209	24:00:00	24:00:00	8.01%	5.13%
t30_25_2	4	30	22.98	0.29	282	282	24:00:00	03:13:04	5.67%	0.00%
t30_30_1	4	30	30.34	0.34	414	414	03:30:48	00:11:34	0.00%	0.00%
t30_30_2	4	30	30.57	0.22	262	262	24:00:00	24:00:00	8.40%	7.29%
t30_50_1	4	30	53.10	0.18	370	370	03:29:44	00:01:32	0.00%	0.00%
t30_50_2	4	30	47.58	0.14	470	470	24:00:00	24:00:00	3.96%	5.07%
t30_60_1	4	30	56.32	0.21	467	467	00:23:37	00:17:56	0.00%	0.00%
t30_60_2	4	30	62.98	0.10	445	445	13:47:35	24:00:00	0.00%	1.53%
t30_75_1	4	30	75.86	0.14	544	544	05:34:36	03:40:45	0.00%	0.00%
t30_75_2	4	30	74.02	0.09	607	607	24:00:00	24:00:00	1.08%	1.99%
t30_90_1	4	30	89.19	0.10	562	562	00:00:39	00:01:10	0.00%	0.00%
t30_90_2	4	30	88.27	0.08	697	697	00:03:47	00:05:22	0.00%	0.00%
t40_10_1	4	40	12.43	0.65	254	246	24:00:00	24:00:00	10.24%	5.28%
t40_10_2	4	40	9.35	0.58	265	286	24:00:00	24:00:00	31.32%	36.36%
t40_25_1	4	40	25.00	0.33	293	270	24:00:00	12:54:31	8.87%	0.00%
t40_25_2	4	40	25.25	0.20	380	370	24:00:00	24:00:00	27.30%	25.68%
t40_30_1	4	40	29.74	0.24	459	466	24:00:00	24:00:00	9.44%	10.73%
t40_30_2	4	40	29.61	0.17	397	402	24:00:00	24:00:00	11.08%	12.09%
t40_50_1	4	40	46.92	0.14	485	491	24:00:00	24:00:00	16.70%	17.52%
t40_50_2	4	40	51.92	0.09	591	595	24:00:00	24:00:00	3.78%	4.77%
t40_60_1	4	40	60.76	0.15	753	753	24:00:00	24:00:00	2.66%	3.15%
t40_60_2	4	40	58.97	0.08	612	603	24:00:00	24:00:00	14.05%	12.06%
t40_75_1	4	40	73.84	0.12	776	776	24:00:00	24:00:00	1.30%	0.97%
t40_75_2	4	40	74.23	0.06	772	772	24:00:00	24:00:00	9.35%	9.92%
t40_90_1	4	40	88.58	0.09	688	689	24:00:00	24:00:00	5.58%	4.84%
t40_90_2	4	40	88.71	0.04	601	601	24:00:00	17:50:06	0.87%	0.00%

De acordo com [14], quando a densidade do grafo de precedências é alta, o conjunto de restrições de linearização denominado como *retrições de redução* é menor que o conjunto de restrições de linearização padrão, o que indica que o tempo de solução das instâncias cujo ρ seja alto é menor ao usar as restrições de redução ao invés da linearização padrão. Quando a instância

não foi resolvida, podemos notar que o valor do *gap* também é menor em quase todos os casos. Quando a densidade do grafo de precedências é baixa ocorre o contrário, ou seja, tanto o tempo de solução quanto o valor do *gap*(quando a instância não for resolvida no ótimo) são menores ao se usar a linearização padrão ao invés das restrições de redução. Ambas situações podem ser observadas em quase todas as instâncias testadas, comparando os resultados da tabela 4.1 com a tabela 4.2, seja usando a formulação normal ou a formulação com as inequações.

Tendo isso em mente, é interessante que se faça uma análise do impacto do uso das inequações de acordo com a densidade do grafo da instância em questão. No caso do conjunto de instâncias com 30 tarefas, usando o conjunto de restrições de linearização padrão pode-se observar na tabela 4.1 que nas instâncias em que $\rho \geq 50$, onde o uso da linearização padrão não produz bons resultados como argumentado acima, as inequações melhoraram bastante o tempo de solução ou o valor do *gap*, permitindo inclusive que em uma delas fosse encontrada a solução ótima. Usando o conjunto de restrições de redução para a linearização, pode-se observar na tabela 4.2 que nas instâncias em que $\rho < 50$, onde o uso desse conjunto de restrições normalmente não produz bons resultados, as inequações melhoraram o tempo das instâncias que já tinham sido resolvidas como também reduziram o valor do *gap* final.

Já no caso do conjunto de instâncias com 40 tarefas, o uso das inequações não apresentou o mesmo desempenho apresentado no conjunto de instâncias com 30 tarefas tanto para a linearização padrão como para as restrições de redução. Usando a linearização padrão, quando $\rho \geq 50$, em quatro instâncias o *gap* foi menor com uma diferença média de 3%, e em três ele foi maior com diferença média de 1%. Usando as restrições de redução, quando $\rho < 50$, em três delas o *gap* foi menor e em outras três ele foi maior, sendo que em ambos

os casos a diferença média no valor do *gap* foi também de 1%. Quando $\rho < 50$, no caso da linearização padrão, e $\rho \geq 50$, no caso das restrições de redução, o ganho com o uso das inequações não foi tão expressivo, principalmente com o grupo de instâncias de 40 tarefas.

Apresentamos abaixo a tabela 4.3, que resume quantitativamente o desempenho do uso das inequações, onde a linha "<" mostra o número de instâncias cujo tempo de solução ou *gap* de dualidade foi menor, e a linha ">" o número de instâncias cujo tempo de solução ou *gap* de dualidade foi maior. As inequações obtiveram mais êxito quando a linearização padrão foi usada, e apesar do número de instâncias cujo tempo de solução foi maior usando as inequações ter sido maior, vale lembrar que isso ocorreu principalmente com as instâncias pequenas, que foram resolvidas dentro de segundos. Nas instâncias com número maior de tarefas, as inequações apresentaram melhores resultados de forma geral.

	Lin. Padrão		Rest. Redução	
Grupo	Tempo	Gap	Tempo	Gap
<	17	15	13	12
>	7	6	15	13

Tabela 4.3: Tabela comparativa com número de instâncias

4.2.2 Arquitetura Hipercubo

Nesta seção apresentamos os resultados obtidos usando a arquitetura hipercubo 2D, onde nem todas instâncias disponíveis puderam ser usadas, pois algumas possuíam número de processadores inferior a quatro. O objetivo de realizar testes com essa arquitetura é de poder avaliar qual a influência da

arquitetura do sistema de multiprocessadores na solução final do problema, ou seja, saber se essa mudança torna o problema mais fácil ou mais difícil de ser resolvido, e caso seja possível, dizer sob quais condições isso acontece.

Como o objetivo principal aqui não é mais avaliar o uso das linearizações, decidiu-se realizar os testes com a formulação normal e a formulação com as inequações usando somente a linearização padrão. Não faremos também uma análise dos resultados com as inequações como foi feito na seção anterior, já que o foco da análise aqui feita é o impacto da mudança de arquitetura dos multiprocessadores.

O primeira fato que pode ser observado foi que no conjunto de instâncias usados para teste, a mudança para a arquitetura hipercubo 2D não acarretou mudança no valor da solução ótima de todas as instâncias que puderam ser resolvidas, com exceção de uma única delas. A única mudança significativa que pôde ser observada foi o tempo gasto para resolver estas instâncias, e no caso daquelas que não foram resolvidas, o valor do *gap*.

Dentre aquelas que foram resolvidas, o tempo gasto para resolver as instâncias com arquitetura hipercubo foi na grande maioria dos casos menor que para resolver aquelas com arquitetura completamente conectada, tanto para a formulação normal quanto para a formulação com inequações. Podemos destacar três casos interessantes: a instância **t30_25_2**, que não foi resolvida na arquitetura completamente conectada mas foi resolvida na arquitetura hipercubo 2D; a instância **t30_30_2**, que foi não foi resolvida na arquitetura hipercubo 2D mas na arquitetura completamente conectada, e a instância **t20_10** que foi resolvida na arquitetura hipercubo 2D usando as inequações, mas não foi usando a formulação normal.

Com respeito às instâncias que não foram resolvidas, pode-se observar que no conjunto com 30 tarefas o valor do *gap* foi menor na maioria dos casos,

seja para a formulação normal quanto para a formulação com as inequações. Já no grupo de instâncias com 40 tarefas, nenhuma instância foi resolvida na arquitetura hipercubo, ao passo que na arquitetura completamente conectada tivemos uma delas resolvida. Com relação ao valor do *gap*, na grande maioria dos casos ele foi maior na arquitetura hipercubo que na outra arquitetura testada.

As tabelas abaixo resumem quantitativamente as informações apresentadas acima, onde podemos ver que à medida que o número de tarefas vai crescendo, os resultados da arquitetura hipercubo 2D vão ficando piores que os da arquitetura completamente conectada, como pode ser o observado para o caso das instâncias com 40 tarefas, onde o valor do *gap* foi maior em 12 instâncias. A tabela 4.4 apresenta os dados referente ao tempo gasto pelas instâncias resolvidas, enquanto a tabela 4.5 apresenta os dados referentes ao *gap* das instâncias que não foram resolvidas.

Num. Tarefas	Caso	Normal	Ineq.
10	<	4	4
	>	0	0
20	<	4	3
	>	1	2
30	<	5	5
	>	1	2

Tabela 4.4: Tabela referente ao tempo

Num. Tarefas	Caso	Normal	Ineq.
30	<	5	4
	>	2	3
40	<	2	2
	>	12	12

Tabela 4.5: Tabela referente ao *gap*

Tabela 4.6: Tabela comparativa de resultados para a arquitetura hipercubo 2D usando a linearização padrão. Tempo de CPU expresso em [horas]:minutos:segundos.[centésimos]

Instância	p	n	ρ	f_c	x_o^*	x_l^*	CPU_o	CPU_l	Gap_o	Gap_l
t10_10	4	10	8.89	2.29	71	71	00:04.52	00:00.08	0.00%	0.00%
t10_25	4	10	28.9	0.84	60	60	00:02.47	00:00.95	0.00%	0.00%
t10_50	4	10	46.6	0.23	112	112	00:02.70	00:03.89	0.00%	0.00%
t10_75	4	10	80.0	0.22	105	105	00:00.09	00:00.47	0.00%	0.00%
t10_90	4	10	86.67	0.22	102	102	00:01.64	00:02.14	0.00%	0.00%
t20_10	4	20	8.95	1.57	110	110	24:00:00	00:06.14	0.91%	0.00%
t20_25	4	20	23.68	0.42	151	151	36:25.78	08:06.01	0.00%	0.00%
t20_50	4	20	53.16	0.24	221	221	01:07.56	00:54.74	0.00%	0.00%
t20_75	4	20	76.32	0.15	241	241	03:47:22	01:33:46	0.00%	0.00%
t20_90	4	20	80.52	0.12	242	242	01:25.21	01:32.16	0.00%	0.00%
t30_10_1	4	30	9.88	0.79	138	138	24:00:00	24:00:00	2.17%	0.72%
t30_10_2	4	30	8.96	0.82	154	152	24:00:00	24:00:00	28.57%	26.97%
t30_25_1	4	30	22.29	0.50	209	209	24:00:00	24:00:00	6.90%	7.66%
t30_25_2	4	30	22.98	0.29	282	282	01:05:36	02:10.74	0.00%	0.00%
t30_30_1	4	30	30.34	0.34	415	415	10:41.89	05:24.94	0.00%	0.00%
t30_30_2	4	30	30.57	0.22	263	263	24:00:00	24:00:00	3.42%	0.38%
t30_50_1	4	30	53.10	0.18	370	370	15:49.60	02:42.32	0.00%	0.00%
t30_50_2	4	30	47.58	0.14	470	470	24:00:00	24:00:00	8.25%	6.83%
t30_60_1	4	30	56.32	0.21	467	467	05:54:04	04:39:38	0.00%	0.00%
t30_60_2	4	30	62.98	0.10	445	445	24:00:00	05:06:14	0.45%	0.00%
t30_75_1	4	30	75.86	0.14	544	544	07:32:04	02:03:03	0.00%	0.00%
t30_75_2	4	30	74.02	0.09	607	607	24:00:00	24:00:00	3.29%	2.57%
t30_90_1	4	30	88.19	0.10	562	562	01:08:28	01:24:17	0.00%	0.00%
t30_90_2	4	30	88.27	0.08	697	697	13:53.78	12:57.61	0.00%	0.00%
t40_10_1	4	40	12.43	0.65	256	260	24:00:00	24:00:00	10.94%	12.31%
t40_10_2	4	40	9.35	0.58	257	256	24:00:00	24:00:00	29.18%	28.52%
t40_25_1	4	40	25.00	0.33	275	270	24:00:00	24:00:00	2.91%	0.37%
t40_25_2	4	40	25.25	0.20	378	378	24:00:00	24:00:00	26.72%	26.69%
t40_30_1	4	40	29.74	0.24	478	477	24:00:00	24:00:00	12.99%	12.60%
t40_30_2	4	40	29.61	0.17	399	411	24:00:00	24:00:00	11.19%	13.87%
t40_50_1	4	40	46.92	0.14	511	499	24:00:00	24:00:00	21.67%	19.23%
t40_50_2	4	40	51.92	0.09	609	601	24:00:00	24:00:00	8.06%	6.90%
t40_60_1	4	40	60.76	0.15	754	753	24:00:00	24:00:00	3.55%	3.36%
t40_60_2	4	40	58.97	0.08	643	647	24:00:00	24:00:00	18.51%	18.81%
t40_75_1	4	40	73.84	0.12	778	777	24:00:00	24:00:00	2.79%	2.18%
t40_75_2	4	40	74.23	0.06	795	772	24:00:00	24:00:00	15.22%	12.30%
t40_90_1	4	40	88.58	0.09	691	704	24:00:00	24:00:00	7.59%	8.46%
t40_90_2	4	40	88.71	0.04	601	601	24:00:00	24:00:00	4.62%	3.94%

4.2.3 Relaxação Linear

A análise dos resultados obtidos através da relaxação linear do problema é de extrema importância, pois nos dá uma pista sobre a qualidade da formulação matemática desenvolvida. Se o valor da relaxação linear tende a assumir valores próximos do valor da solução ótima da instância em questão, indica que a formulação gera bons limites inferiores (no caso do problema atual, que é de minimização) e é indicada para aplicar técnicas de decomposição, como a relaxação lagrangeana.

As próximas tabelas apresentam os dados referentes à solução da relaxação linear de todas as instâncias testadas previamente para a arquitetura completamente conectada, além de novas instâncias com um número de tarefas muito maior, onde fica claro que o uso das inequações válidas propicia um limite inferior melhor. Em todas elas, usou-se a formulação padrão e a formulação com as inequações, ambas usando as restrições de linearização padrão. Decidimos não colocar outras tabelas com os mesmos resultados usando as restrições de redução como alternativa de linearização, pois os valores da relaxação linear foram absolutamente idênticos, além da diferença de tempo ser desprezível. O formato dessas tabelas é similar ao das tabelas anteriores. Gostaria de chamar atenção para a coluna *Gap*, que neste caso é a diferença relativa entre o valor da relaxação linear e o valor da solução ótima do problema. Nas instâncias onde não é conhecida a solução ótima, o espaço é preenchido com o símbolo "-".

A tabela 4.7 apresenta os resultados do conjunto de instâncias usado para testar a arquitetura completamente conectada, enquanto que a tabela 4.8 apresenta os resultados do conjunto de instâncias com mais de 100 tarefas.

Tabela 4.7: Tabela comparativa de resultados da relaxação linear para a arquitetura completamente conectada. Tempo de CPU expresso em minutos : segundos . centésimos.

Instância	p	n	ρ	$\bar{x}_o$	x_n^*	CPU_o	CPU_n	Gap
kwok	3	9	33.33	11	11.33	00:00.00	00:00.02	26.66%
sih91	2	8	25.00	16	16	00:00.01	00:00.01	0.00%
ss91	2	9	41.67	250	250	00:00.00	00:00.01	0.00%
mt91	2	10	31.11	51	51	00:00.00	00:00.02	3.92%
ogra9_30	3	9	13.88	13	13	00:00.00	00:00.01	35.00%
ogra9_60	3	9	27.78	20	20	00:00.00	00:00.01	0.00%
ogra9_90	3	9	41.67	20	20	00:00.00	00:00.03	0.00%
ogra10_30	3	10	13.33	19	19	00:00.00	00:00.01	24.00%
ogra10_60	3	10	26.67	25	25	00:00.01	00:00.02	0.00%
ogra10_90	3	10	40.00	25	25	00:00.01	00:00.02	0.00%
ogra12_20	3	12	7.58	17	17	00:00.00	00:00.03	10.52%
gra12	3	12	16.7	100	100	00:00.02	00:00.01	25.92%
t10_10	4	10	8.89	71	71	00:00.00	00:00.02	0.00%
t10_25	4	10	28.9	49	49	00:00.01	00:00.03	18.33%
t10_50	4	10	80.0	94	94	00:00.01	00:00.03	16.07%
t10_75	4	10	8.95	105	105	00:00.02	00:00.05	0.00%
t10_90	4	10	86.67	92	92	00:00.02	00:00.05	9.80%
t20_10	4	20	8.95	109	109	00:00.01	00:00.34	0.90%
t20_25	4	20	23.68	129	129	00:00.04	00:00.53	14.56%
t20_50	4	20	53.16	214	214	00:00.15	00:00.39	3.16%
t20_75	4	20	76.32	209	209	00:00.22	00:00.67	13.27%
t20_90	4	20	80.52	240	240	00:00.09	00:00.36	0.82%
t30_10_1	4	30	9.88	126	126	00:00.12	00:00.76	-
t30_10_2	4	30	8.96	110	110	00:00.12	00:01.05	-
t30_25_1	4	30	22.29	191	191	00:00.50	00:01.63	-
t30_25_2	4	30	22.98	266	266	00:00.28	00:01.19	5.67%
t30_30_1	4	30	30.34	410	410	00:00.13	00:01.06	0.96%

continua na próxima página

Tabela 4.7: Tabela comparativa de resultados da relaxação linear para a arquitetura completamente conectada(continuação)

Instância	p	n	ρ	$\bar{x}_o$	$\bar{x}_l$	CPU_o	CPU_n	Gap
t30_30_2	4	30	30.57	234	234	00:00.27	00:00.74	10.68%
t30_50_1	4	30	53.10	369	369	00:00.10	00:00.69	0.27%
t30_50_2	4	30	47.58	419	419	00:00.12	00:00.47	-
t30_60_1	4	30	56.32	448	448	00:00.37	00:00.65	4.06%
t30_60_2	4	30	62.98	421	421	00:00.12	00:00.53	5.39%
t30_75_1	4	30	75.86	522	522	00:00.10	00:00.23	4.04%
t30_75_2	4	30	74.02	557	557	00:00.10	00:00.17	-
t30_90_1	4	30	89.19	545	545	00:00.58	00:00.21	3.02%
t30_90_2	4	30	88.27	677	677	00:00.21	00:00.67	2.86%
t40_10_1	4	40	12.43	228	228	00:00.48	00:04.45	-
t40_10_2	4	40	9.35	182	182	00:00.88	00:01.59	-
t40_25_1	4	40	25.00	267	267	00:01.14	00:02.54	1.11%
t40_25_2	4	40	25.25	274	274	00:01.33	00:02.11	-
t40_30_1	4	40	29.74	413	413	00:04.34	00:01.85	-
t40_30_2	4	40	29.61	353	353	00:00.68	00:01.72	-
t40_50_1	4	40	46.92	400	400	00:00.53	00:01.52	-
t40_50_2	4	40	51.92	559	559	00:01.05	00:01.84	-
t40_60_1	4	40	60.76	721	721	00:00.42	00:01.12	-
t40_60_2	4	40	58.97	521	521	00:00.53	00:02.39	-
t40_75_1	4	40	73.84	756	756	00:00.60	00:01.47	-
t40_75_2	4	40	74.23	673	673	00:00.74	00:01.20	-
t40_90_1	4	40	88.58	636	636	00:00.37	00:01.80	-
t40_90_2	4	40	88.71	571	571	00:00.24	00:00.60	-

Ao analisarmos a tabela 4.7, pode-se observar que em todos os casos o valor da relaxação linear foi o mesmo independente do uso ou não das

inequações. Como todas as instâncias apresentam um número relativamente pequeno de tarefas, o tempo gasto para resolver a relaxação linear foi muito pequeno. Sobre o tempo gasto para se obter o valor da relaxação, pode-se esperar de antemão que o tempo gasto usando-se as inequações válidas sempre maior na maioria dos casos, pois o acréscimo das restrições torna a matriz usada pelo simplex maior.

Tabela 4.8: Tabela comparativa de resultados da relaxação linear para instâncias com mais de 100 tarefas

Instância	p	n	ρ	$\bar{x}_o$	x_n^*	CPU_o	CPU_n	Gap_o	Gap_l
ogra100_1	2	100	10	323	593	00:02.56	00:19.11	59.62%	25.87%
ogra100_2	2	100	40	585	762	00:03.22	00:04.89	26.87%	4.75%
ogra100_3	2	100	70	675	785	00:04.42	00:06.05	15.62%	1.87%
ogra100_4	2	100	90	715	788	00:04.90	00:12.35	10.62%	1.5%
ogra150_1	2	100	10	362	788.5	00:09.24	00:26.88	63.8%	21.15%
ogra150_2	2	100	40	693	943	00:10.56	00:31.46	30.7%	5.7%
ogra150_3	2	100	70	874	992.5	00:10.52	00:27.48	12.6%	0.75%
ogra150_4	2	150	90	919	999	00:17.33	00:26.29	8.1%	0.01%

Como mostrado nas seções anteriores, o uso das inequações válidas melhorou o tempo de solução de um número significante das instâncias testadas, apesar dessa melhoria não ter sido causada por um valor de limite inferior melhor. O uso de inequações válidas está intimamente ligado à esperança de uma melhoria no valor do limite inferior obtido principalmente através da relaxação linear. Mas essa melhoria não foi observada no conjunto de instâncias usadas, que está limitado à 40 tarefas, já que acima disso a com-

plexidade do problema não permite que a grande maioria das instâncias seja resolvida em pelo menos um dia de processamento.

Sendo assim, decidimos investigar o valor da relaxação linear para as instâncias com número de tarefas superior a 100, pertencente ao grupo de instâncias denominado *ogra*. Para esse conjunto, decidimos usar o algoritmo de pontos interiores do CPLEX para tornar o processo de solução mais rápido, ao invés de se usar a opção padrão do CPLEX que é o algoritmo dual simplex.

A tabela 4.8 apresenta os resultados obtidos para este novo conjunto de instâncias. Pode-se observar agora que o uso das inequações válidas acarretou numa melhoria significativa no valor da relaxação linear, principalmente para as instâncias cuja densidade do grafo de precedência era baixa. Um fato interessante de se observar neste grupo de instâncias é que à medida que a densidade do grafo vai aumentando, o valor do limite inferior vai aumentando também, o que pode ser explicado pelo fato do problema ficar mais "amarrado" com o grafo de precedências mais denso.

4.3 Conclusões

Neste capítulo foram apresentados os resultados das instâncias do **MSPCD** testadas na formulação *Packing*. Com o intuito de avaliar esta nova formulação, foi usado nos testes dois tipos de linearização para a formulação, além de um conjunto de inequações válidas. Das arquiteturas existentes de sistemas de multiprocessadores, escolhemos usar a arquitetura completamente conectada e a arquitetura hipercubo 2D, por serem bastante populares. Também são apresentados os resultados da relaxação linear da formulação usando as restrições de linearização padrão.

Sendo o uso do conjunto de inequações válidas na formulação uma das principais contribuições deste trabalho, fizemos uma descrição e análise detalhada do impacto do uso delas na formulação usando ambos conjuntos de restrições de linearização. De forma geral, podemos dizer que o uso das inequações válidas traz ganhos razoáveis na solução do problema, quer seja no tempo gasto para resolvê-lo, quer seja no valor do *gap* de dualidade ao fim do processo. Os testes realizados com instâncias com um grande número de tarefas encorajam ainda mais seu uso, pois indicam uma melhoria considerável no valor do limite inferior obtido através da relaxação linear.

Finalmente, os testes usando a arquitetura hipercubo 2D mostraram que pelo menos para este conjunto de instâncias usado, a mudança de arquitetura imposta praticamente não causou impacto no valor do *makespan*. Observou-se que o aumento do número de tarefas tornou o problema mais difícil de se resolver do que na arquitura completamente conectada, o que pode ser explicado pelo fato de existir um número maior de alternativas a se considerar sobre onde a tarefa será executada.

Capítulo 5

Formulação Packing para o MSP

Após termos avaliado os resultados obtidos com a formulação *Packing* para o **MSPCD**, julgamos relevante avaliar o comportamento desta nova formulação na ausência dos termos não-lineares, que carecterizam os atrasos de comunicação, e suas respectivas restrições de linearização. Dessa forma, estaremos usando a formulação *Packing* em uma versão ligeiramente adaptada para resolver instâncias do problema **MSP** com processadores uniformes.

Para que possamos fazer uma análise válida do desempenho desta formulação, decidimos testar o mesmo conjunto de instâncias usando a formulação de Pablo et al.[5], já mencionada no capítulo 2, por esta configurar-se como uma das melhores formulações encontradas na literatura para o **MSP**.

Na seção 5.1 apresentamos as duas formulações matemáticas, enquanto que na seção 5.2 apresentamos os resultados computacionais obtidos. Finalmente, apresentamos as conclusões na seção 5.3.

5.1 Formulações Matemáticas

5.1.1 Formulação Packing

A seguir, apresentamos a formulação *Packing* adaptada para o **MSP** com processadores uniformes. A notação é a mesma utilizada no capítulo 3.

$$(MSP_{packing}) \qquad \min_{x,y,\sigma,\epsilon} W_{max} \tag{5.1}$$

sujeito a:

$$x_i + L_i \leq W_{max} \quad ,\forall\, i \in T \tag{5.2}$$

$$x_j - x_i - L_i - (\sigma_{ij} - 1)\alpha \geq 0 \quad ,\forall\, i \neq j \in T \tag{5.3}$$

$$y_j - y_i - 1 - (\epsilon_{ij} - 1)|P| \geq 0 \quad ,\forall\, i \neq j \in T \tag{5.4}$$

$$\sigma_{ij} + \sigma_{ji} + \epsilon_{ij} + \epsilon_{ji} \geq 1 \quad ,\forall\, i \neq j \in T \tag{5.5}$$

$$\sigma_{ij} + \sigma_{ji} \leq 1 \quad ,\forall\, i \neq j \in T \tag{5.6}$$

$$\epsilon_{ij} + \epsilon_{ji} \leq 1 \quad ,\forall\, i \neq j \in T \tag{5.7}$$

$$\sigma_{ij} = 1 \quad ,\forall\, j \in T : i \in \delta^-(j) \tag{5.8}$$

$$x_i + L_i + \leq x_j \quad ,\forall j \in V : i \in \delta^-(j) \tag{5.9}$$

$$\sum_{k \in P} k z_{ik} = y_i \quad ,\forall\, i \in T \tag{5.10}$$

$$\sum_{k \in P} z_{ik} = 1 \quad ,\forall\, i \in T \tag{5.11}$$

$$W_{max} \geq 0 \tag{5.12}$$

$$x_i \geq 0 \quad ,\forall\, i \in T \tag{5.13}$$

$$y_i \in \{1, \ldots, |P|\}, \forall\, i \in T \tag{5.14}$$

$$z_{ik} \in \{0,1\} \quad ,\forall\, i \in T, k \in P \tag{5.15}$$

$$\sigma_{ij}, \epsilon_{ij} \in \{0,1\} \quad ,\forall\, i,j \in T \tag{5.16}$$

5.1.2 Formulação de Coll

Apresentamos aqui a formulação descrita em [5] para o MSP com processadores uniformes. Vale ressaltar que a formulação original abrange processadores não-uniformes, e que por isso decidimos fazer algumas modificações a fim de que seja levado em consideração somente os processsadores uniformes, como é feito na formulação *Packing*.

Com respeito à notação, faz-se necessário definir uma nova variável binária:

$$\sigma_{ij} = \begin{cases} 1 & \text{tarefa } i \text{ é executada antes da tarefa } j \\ & \text{no mesmo processador} \\ 0 & \text{c.c.} \end{cases} \tag{5.17}$$

Segue abaixo a formulação:

$$(MSP) \qquad \min \ W_{max} \tag{5.18}$$

sujeito a:

$$\sum_{k=1}^{m} z_{jk} = 1 \quad \forall j \in T \tag{5.19}$$

$$\sigma_{ij} + \sigma_{ji} + z_{ik} - z_{jk} \leq 1 \quad \forall i \in T,\ \forall i \in R_j,\ \forall k \in P \tag{5.20}$$

$$\sigma_{ij} + z_{ik} - z_{jk} \leq 1 \quad \forall j \in T,\ i \in S_j,\ \forall k \in P \tag{5.21}$$

$$z_{ik} + z_{jk} - \sigma_{ij} - \sigma_{ji} \leq 1 \quad \forall i \in T,\ \forall i \in R_j,\ \forall k \in P \tag{5.22}$$

$$z_{ik} + z_{jk} - \sigma_{ij} \leq 1 \quad \forall j \in T,\ i \in S_j,\ \forall k \in P \tag{5.23}$$

$$x_i - x_j + \sum_{k=1}^{m} L_i z_{ik} \leq 0 \quad \forall j \in T,\ \forall i \in \delta^{-}(j) \tag{5.24}$$

$$x_j - W_{max} + \sum_{k=1}^{m} L_i z_{ik} \leq 0 \quad \forall j \in T \tag{5.25}$$

$$x_i - x_j + \sum_{k=1}^{m} L_i z_{ik} \leq \alpha(1 - z_{ij}) \quad \forall i \in T, \forall i \in R_j \tag{5.26}$$

$$x_j \geq 0, \forall j \in T \tag{5.27}$$

$$z_{jk} \in \{0, 1\} \quad \forall (j, k) \in T \times P \tag{5.28}$$

$$\sigma_{ij} \in \{0, 1\} \quad \forall i \in T, \forall i \in R_j \cup S_j \tag{5.29}$$

As equações (5.19) garantem que cada tarefa é processada e atribuída a exatamente um processador. As inequações (5.24) expressam as restrições de precedência entre as tarefas. As inequações (5.26) definem a sequência de tempos de início das tarefas atribuídas a um mesmo processador, garantindo a inexistência de sobreposição delas, enquanto que as inequações (5.25) definem o *makespan*. A relação correta entre as variáves z e σ é dada pelas inequações (5.20)-(5.23).

5.2 Resultados Computacionais

As instâncias utilizadas para testar ambas formulações são as mesmas usadas na seção anterior, sendo que os dados referentes à arquitetura do sistema de multiprocessadores e ao tempo de transferência de dados entre os processadores foram retirados, já que nesta versão do problema o tempo de transferência de dados é considerado irrelevante, independente de como o conjunto de processadores está interconectado.

Ambos modelos foram implementados na linguagem AMPL, sendo que o resolvedor utilizado para resolver a instância em questão foi o CPLEX 9.1. Os testes foram conduzidos em uma máquina com quatro processadores Intel *Xeon* cada um com 2.80 GHz e com uma memória total de 4 GB de RAM, com limite de tempo de solução de um dia.

Na tabela 5.1 apresentamos os resultados dos testes rodados usando ambas formulações. A primeira e a segunda coluna contém respectivamente os nomes das instâncias e o número de processadores, enquanto que as próximas duas colunas indicam respectivamente o número de tarefas e a densidade do grafo de precedência. As próximas seis colunas dizem respeito aos dados da solução da instância, onde a letra subscrita s identifica um resultado pertencente à formulação de Coll et al., enquanto a letra subscrita p identifica um resultado pertencente à formulação *Packing*. São elas: valor da melhor solução inteira encontrada dentro do limite de tempo estipulado, tempo de solução gasto pelo *solver* CPLEX e finalmente, o *gap* de dualidade existente ao fim do processo de solução.

Analisando a tabela, pode-se notar que a formulação de Coll et al. saiu-se ligeiramente melhor nas instâncias com número pequeno de tarefas, enquanto que a formulação *Packing* saiu-se melhor nas instâncias com 30 e 40 tarefas. Podemos destacar três casos interessantes onde a formulaçãoPacking sobressaiu-se de forma expressiva: a instância **t40_25_2**, onde a solução ótima pôde ser encontrada, as instâncias **t40_10_1** e **t40_25_2**, cuja diferença de tempo foi de aproximadamente 57 e 38 minutos respectivamente.

Nota-se facilmente também que a grande maioria das instâncias foi resolvida em um intervalo de tempo significantemente pequeno, se comparado com a versão do problema com atrasos de comunicação. Somente duas instâncias do grupo com 30 tarefas e duas do grupo com 40 tarefas não foram resolvidas. Essa fato deixa claro que a retirada dos termos não-lineares, e consequentemente das restrições de linearização, tornou o problema mais fácil de ser resolvido.

Tabela 5.1: Tabela comparativa de resultados. Tempo de CPU expresso em [horas]:minutos:segundos.[centésimos]

Instância	p	n	ρ	x_s^*	x_p^*	CPU_s	CPU_p	Gap_s	Gap_p
kwok	3	9	33.33	13	13	00:00.11	00:00.70	0.00%	0.00%
sih91	2	8	25.00	16	16	00:00.08	00:00.09	0.00%	0.00%
ss91	2	9	41.67	250	250	00:00.07	00:00.05	0.00%	0.00%
mt91	2	10	31.11	51	51	00:00.11	00:00.10	0.00%	0.00%
ogra9_30	3	9	13.88	20	20	00:30.30	00:18.27	0.00%	0.00%
ogra9_60	3	9	27.78	20	20	00:00.07	00:00.37	0.00%	0.00%
ogra9_90	3	9	41.67	20	20	00:00.07	00:00.16	0.00%	0.00%
ogra10_30	3	10	13.33	25	25	00:42.38	04:39.27	0.00%	0.00%
ogra10_60	3	10	26.67	25	25	00:00.17	00:00.25	0.00%	0.00%
ogra10_90	3	10	40.00	25	25	00:00.10	00:00.23	0.00%	0.00%
ogra12_20	3	12	7.58	17	17	00:00.44	00:00.57	0.00%	0.00%
gra12	3	12	16.7	134	134	09:17.50	05:39.98	0.00%	0.00%
t10_10	4	10	8.89	71	71	00:00.05	00:00.08	0.00%	0.00%
t10_25	4	10	28.9	49	49	00:00.07	00:00.07	0.00%	0.00%
t10_50	4	10	46.66	94	94	00:00.01	00:00.10	0.00%	0.00%
t10_75	4	10	80.0	105	105	00:00.00	00:00.01	0.00%	0.00%
t10_90	4	10	86.67	92	92	00:00.01	00:00.01	0.00%	0.00%
t20_10	4	20	8.95	109	109	00:04.33	00:02.05	0.00%	0.00%
t20_25	4	20	23.68	129	129	00:00.82	00:00:85	0.00%	0.00%
t20_50	4	20	53.16	214	214	00:00.13	00:00.23	0.00%	0.00%
t20_75	4	20	76.32	209	209	00:00:06	00:00:08	0.00%	0.00%
t20_90	4	20	80.52	240	240	00:00.02	00:00.08	0.00%	0.00%
t30_10_1	4	30	9.88	130	133	24:00:00	24:00:00	3.08%	5.26%
t30_10_2	4	30	8.96	153	153	24:00:00	24:00:00	28.10%	28.10%
t30_25_1	4	30	22.29	191	191	02:58.54	00:30.00	0.00%	0.00%
t30_25_2	4	30	22.98	266	266	00:33.48	00:06.21	0.00%	0.00%
t30_30_1	4	30	30.34	410	410	01:51.64	00:03.48	0.00%	0.00%
t30_30_2	4	30	30.57	234	234	01:37.82	00:06.42	0.00%	0.00%
t30_50_1	4	30	53.10	369	369	00:51.64	00:56.13	0.00%	0.00%

continua na próxima página

Tabela 5.1: Tabela comparativa de resultados(continuação)

Instância	p	n	ρ	x_s^*	x_p^*	CPU_s	CPU_p	Gap_s	Gap_p
t30_50_2	4	30	47.58	419	419	00:21.29	00:02.12	0.00%	0.00%
t30_60_1	4	30	56.32	448	448	00:11.11	00:00.09	0.00%	0.00%
t30_60_2	4	30	62.98	421	421	00:46.67	00:01.21	0.00%	0.00%
t30_75_1	4	30	75.86	522	522	00:10.62	00:00.33	0.00%	0.00%
t30_75_2	4	30	74.02	557	557	00:41.60	00:00.70	0.00%	0.00%
t30_90_1	4	30	89.19	545	545	00:00.53	00:00.05	0.00%	0.00%
t30_90_2	4	30	88.27	677	677	00:03.13	00:00.15	0.00%	0.00%
t40_10_1	4	40	12.43	228	228	00:59:21	00:02:04	0.00%	0.00%
t40_10_2	4	40	9.35	244	237	24:00:00	24:00:00	25.41%	23.21%
t40_25_1	4	40	25.00	267	267	39:00.37	00:47.92	0.00%	0.00%
t40_25_2	4	40	25.25	276	274	24:00:00	01:13.86	0.72%	0.00%
t40_30_1	4	40	29.74	413	413	01:52.81	00:15.22	0.00%	0.00%
t40_30_2	4	40	29.61	353	353	02:29.70	00:30.36	0.00%	0.00%
t40_50_1	4	40	46.92	400	400	01:40.88	00:06.56	0.00%	0.00%
t40_50_2	4	40	51.92	559	559	01:14.54	00:04.35	0.00%	0.00%
t40_60_1	4	40	60.76	721	721	00:51.40	00:02.91	0.00%	0.00%
t40_60_2	4	40	58.97	521	521	01:25.69	00:04.94	0.00%	0.00%
t40_75_1	4	40	73.84	756	756	00:56.20	00:02.49	0.00%	0.00%
t40_75_2	4	40	74.23	673	673	01:02.13	00:01.32	0.00%	0.00%
t40_90_1	4	40	88.58	636	636	01:03.94	00:00.75	0.00%	0.00%
t40_90_2	4	40	88.71	571	571	01:10.68	00:00.67	0.00%	0.00%

5.3 Conclusões

Neste capítulo, adaptamos a formulação *Packing* para resolver problemas de escalonamento em sistemas de multiprocessadores onde não há atrasos de comunicação entre eles, versão esta do problema denominada anteriormente

de **MSP**. Para que se possa ter uma idéia de seu desempenho, foi usada a formulação de Coll et al. adaptada para o mesmo problema.

A formulação *Packing* mostrou bom desempenho em quase todas as instâncias testadas, destacando-se em algumas delas. Infelizmente, com o conjunto de instâncias usado não foi possível detectar casos onde a formulação não tenha um desempenho bom e a outra o tenha. Seria interessante usar instâncias com um número maior de tarefas, a fim de se descobrir o número de tarefas onde a formulação não consegue mais se sair tão bem.

Capítulo 6

Conclusões

Neste trabalho apresentamos algumas formulações matemáticas para dois tipos de problemas de escalonamento em sistemas de multiprocessadores. Demos enfoque inicialmente a duas delas por razões históricas, por delas terem sido a motivação para o desenvolvimento de uma nova formulação mais compacta para a versão do problema com atrasos de comunicação, o MSPCD.

O objetivo principal deste trabalho era avaliar o comportamento dessa nova formulação, portanto era necessário realizar-se uma quantidade significativa de testes nesta nova formulação. Utilizou-se desde instâncias com número pequeno de tarefas até instâncias com um número razoável para ser resolvido utilizando um modelo matemático. Os resultados foram bastante satisfatórios ao se comparar com a outra formulação existente, apesar desta formulação já começar a mostras suas limitações com instâncias com mais de 40 tarefas.

Com o propósito de melhorar o tempo de solução das instâncias e o valor do limite inferior, foi apresentado um conjunto de inequações válidas para a formulação, constituindo nossa principal contribuição. Os testes com as inequações mostraram que de uma forma geral elas melhoraram os resulta-

dos obtidos sem o seu uso. Dessa forma, o uso dessas inequações com a formulação é encorajado.

Futuramente, seria interessante de se tentar implementar algum algoritmo de decomposição para esta formulação a fim de se resolver instâncias com um número maior de tarefas e valor de solução ótima desconhecido. Uma alternativa seria buscar inequações definidoras de facetas para o politopo de partições em ordens lineares ou algum outro similar, que de acordo com o trabalho de Pablo et al [5], pode ser usado para derivar inequações válidas para várias variantes de problemas de escalonamento. De posse dessas inequações, não fica difícil de se implementar um algoritmo de *branch-and-cut*. Uma outra alternativa seria tentar adaptar a formulação para que se possa usar uma abordagem de geração de colunas.

Bibliografia

[1] BLAZEWICZ, J., CELLARY, W., SLOWINSKI, R., *et al.*, *Scheduling Under Resource Constraints - Deterministic Models.* Basel, J.C. Baltzer AG, 1986.

[2] BLAZEWICZ, J., DROR, M., WEGLARZ, J., "Mathematical Rrogramming Formulations for Machine Scheduling: A survey", *European Journal of Operational Research*, v. 51, pp. 283–300, 1991.

[3] BLAZEWICZ, J., ECKER, K. H., PESCH, E., *et al.*, *Scheduling Computer and Manufacturing Processes.* 1 ed. Berlin, Springer, 1996.

[4] BLAZEWICZ, J., LENSTRA, J. K., KAN, A. H. G. R., "Scheduling subject to resource constraints: classification and complexity", *Discrete Applied Mathematics*, v. 5, pp. 11–24, 1983.

[5] COLL, P. E., RIBEIRO, C. C., SOUZA, C. C., "Multiprocessor Scheduling Under Precedence Constraints: Polyhedral Results", *Discrete Applied Mathematics*, v. 154, pp. 770–801, 2006.

[6] DAVIDOVIC, T., CRAINIC, T. G., "Benchmark-problem instances for static scheduling of task graphs with communication delays on homogeneous multiprocessor systems", *Computer & Operations Research*, v. 33, n. 8, pp. 2155–2177, 2006.

[7] DAVIDOVIC, T., LIBERTI, L., MACULAN, N., *et al.*, "Mathematical Programming-Based Approach to Scheduling of Communicating Tasks", *Les Cahiers du GERAD*, v. G-2004-99, 2004.

[8] GAREY, M. R., JOHNSON, D. S., "Strong NP-Completeness Results: Motivation, Examples and Implications", *Journal of the ACM*, v. 25, pp. 499–508, 1978.

[9] GAREY, M. R., JOHNSON, D. S., *Computers and Intractability: A Guide to the Theory of NP-Completeness*. San Franciso, W. H. Freeman and Company, 1979.

[10] GRAHAM, R. L., LAWLER, E. L., LENSTRA, J. K., *et al.*, "Optimization and approximation in deterministic sequencing and scheduling theory: a survey", *Annals Discrete Math.*, v. 5, pp. 287–326, 1979.

[11] GUAN, Y., CHEUNG, R. K., "The Berth Allocation Problem: Models and Solution Methods", *OR Spectrum*, v. 26, n. 1, pp. 75–92, 2004.

[12] KWOK, Y. K., AHMAD, I., "Benchmarking and Comparison of the Task Graph Scheduling Algorithms", *J. Parallel and Distributed Computing*, v. 59, n. 3, pp. 381–422, 1999.

[13] LIBERTI, L., "Compact Linearization for Bilinear Mixed-Integer Problems", working paper, DEI, Politecnico di Milano, 2004.

[14] LIBERTI, L., DAVIDOVIC, T., MACULAN, N., *et al.*, "A Compact Model for the Multiprocessor Scheduling Problem with Communication Delays", note del polo, Politecnico di Milano, 2005.

[15] MACULAN, N., RIBEIRO, C. C., PORTO, S., *et al.*, "A New Formulation for Scheduling Unrelated Processor Under Precedence Constraints", *RAIRO Recherche Opérationelle*, v. 33, n. 1, pp. 87–92, 1999.