UNIVERSIDADE FEDERAL DE SANTA MARIA
CENTRO DE TECNOLOGIA
PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA

MODELOS MATEMÁTICOS PARA OS PROBLEMAS DE DIMENSIONAMENTO E PROGRAMAÇÃO DE BATELADAS EM MÁQUINA ÚNICA E MÁQUINAS PARALELAS

DISSERTAÇÃO DE MESTRADO

Renan Spencer Trindade

Santa Maria, RS, Brasil

2014

MODELOS MATEMÁTICOS PARA OS PROBLEMAS DE DIMENSIONAMENTO E PROGRAMAÇÃO DE BATELADAS EM MÁQUINA ÚNICA E MÁQUINAS PARALELAS

Renan Spencer Trindade

Dissertação apresentada ao Curso de Mestrado do
Programa de Pós-Graduação em Informática, Área de Concentração em
Computação Aplicada, da Universidade Federal de Santa Maria (UFSM, RS),
como requisito parcial para obtenção do grau de
Mestre em Ciência da Computação

Orientador: Felipe Martins Müller, Dr.

Santa Maria, RS, Brasil

2014

UNIVERSIDADE FEDERAL DE SANTA MARIA
CENTRO DE TECNOLOGIA
PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA

A Comissão Examinadora, abaixo assinada,
aprova a Dissertação de Mestrado

MODELOS MATEMÁTICOS PARA OS PROBLEMAS DE DIMENSIONAMENTO E PROGRAMAÇÃO DE BATELADAS EM MÁQUINA ÚNICA E MÁQUINAS PARALELAS

elaborada por
Renan Spencer Trindade

como requisito parcial para a obtenção do grau de
Mestre em Ciência da Computação

COMISSÃO EXAMINADORA:

_____
Prof. Felipe Martins Müller, Dr. (UFSM)
(Presidente/Orientador)

_____
Prof. Viviane Cátia Köhler, Dr. (UFSM)

_____
Prof. Marcia Helena Costa Fampa, Dr. (UFRJ)

Santa Maria, 19 de Março de 2014.

DEDICATÓRIA

Dedico esta dissertação ao meu pai, por me mostrar que mesmo nas lutas mais difíceis, devemos manter a confiança e a esperança de que tudo dará certo. Obrigado meu pai por continuar aqui comigo me mostrando que não podemos perder batalhas que podemos vencer.

AGRADECIMENTOS

Quero agradecer primeiramente a minha família, que em todos os momentos esteve ao meu lado, mesmo que a distância não permitisse o convívio diário. Aos meus pais, Isac e Elisabeth, aos meus irmãos e cunhados, muito obrigado por estarem sempre na base da minha vida. Com certeza a maior parte da minha dedicação foi provida da inspiração de vocês.

Quero agradecer a minha namorada Anelise pela paciência nos momentos difíceis e pela ajuda, mesmo com o nosso tempo de convivência desfavorecido e com nossa implacável condição de distância.

Quero agradecer ao professor Felipe Martins Müller por esta grande oportunidade que está se formalizando neste documento. Obrigado por acreditar em mim e na minha capacidade.

Quero agradecer aos meus amigos por estarem sempre batendo à minha porta, ligando ou teclando pra conversar sobre nossas rotinas, novidades e anseios. Mesmo que para alguns a distância possa criar esta saudade de sempre, lembro de que tudo isso nada mais é do que um grande estímulo para o nosso próximo show.

Seguindo esta linha, não posso deixar de agradecer ao amigo Olinto, que esteve sempre correndo ao meu lado nestes últimos anos, abdicando do seu tempo pra estar me auxiliando quando preciso. Obrigado por toda a ajuda e cobrança, por acreditar em mim e por todo o conhecimento que me passou.

RESUMO

Dissertação de Mestrado
Programa de Pós-Graduação em Informática
Universidade Federal de Santa Maria

MODELOS MATEMÁTICOS PARA OS PROBLEMAS DE DIMENSIONAMENTO E PROGRAMAÇÃO DE BATELADAS EM MÁQUINA ÚNICA E MÁQUINAS PARALELAS

AUTOR: RENAN SPENCER TRINDADE
ORIENTADOR: FELIPE MARTINS MÜLLER, DR.
Data e Local da defesa: Santa Maria, 19 de Março de 2014.

Problemas de minimização do *makespan* no dimensionamento e programação de bateladas em máquinas de processamento são extensamente explorados pela literatura acadêmica, motivados principalmente por testes de confiabilidade na indústria de semicondutores. Estes problemas consistem em agrupar tarefas em bateladas e programar o processamento em uma ou mais máquinas em paralelo. As tarefas possuem tempos de processamento e tamanhos não idênticos e o tamanho total da batelada não pode exceder a capacidade da máquina. Para cada batelada é definido um tempo de processamento que será igual ao maior tempo de processamento das tarefas que foram alocadas a ela. As tarefas podem considerar também tarefas com tempos de liberação não idênticos, neste caso as bateladas só poderão ser processadas depois que a tarefa com o maior tempo de liberação for disponibilizada. Este trabalho aborda quatro diferentes problemas de dimensionamento e programação de bateladas com tarefas de tamanhos não idênticos, que consideram diferentes características: máquina de processamento única ($1|s_j,B|C_{max}$), máquina de processamento única e tarefas com tempos de liberação não idênticos ($1|r_j,s_j,B|C_{max}$), máquinas de processamento paralelas idênticas ($P_m|s_j,B|C_{max}$) e máquinas de processamento paralelas idênticas e tarefas com tempos de liberação não idênticos ($P_m|r_j,s_j,B|C_{max}$). São propostos novos modelos matemáticos com formulações que exploram características de cada problema. Os modelos matemáticos são resolvidos utilizando CPLEX e os resultados computacionais comprovam que os modelos propostos possuem um desempenho melhor do que outros modelos da literatura. Os modelos propostos para $1|s_j,B|C_{max}$ e $1|r_j,s_j,B|C_{max}$ são comparados com meta-heurísticas previamente publicadas e os resultados mostram que os novos modelos oferecem soluções melhores com tempos computacionais competitivos.

Palavras-chave: máquina de processamento em batelada, dimensionamento e programação, modelos matemáticos, programação inteira mista.

ABSTRACT

Master of Science Dissertation
Post-Graduate Program in Informatics
Federal University of Santa Maria

MATHEMATICAL MODELS FOR SCHEDULING A SINGLE AND PARALLEL IDENTICALS BATCH PROCESSING MACHINES WITH NON-IDENTICAL JOB SIZES

AUTHOR: RENAN SPENCER TRINDADE
ADVISOR: FELIPE MARTINS MÜLLER, DR.
Date and Local of presentation: Santa Maria, January 19, 2014.

Problems of scheduling on batch processing machines to minimize makespan are widely exploited by academic literature, mainly motivated by reliability testing in the semiconductor industry. These problems consist in grouping jobs as a batch and scheduling the processing in single or parallel machines. The jobs have non-identical processing times and non-identical sizes and the total size of the batch cannot exceed the machine capacity. The processing time of a batch is given by the longest processing time of any job in the batch. Jobs with non-identical release times can also be considered, and in this case a batch can only be processed after the job with the longest release time in the batch is available. We consider four different problems of scheduling on batch processing machines with non-identical job size and different characteristics: single batch processing machine ($1|sj,B|C_{max}$), single batch processing machine with non-identical job release times ($1|r_j,s_j,B|C_{max}$), identical parallel batch processing machines ($P_m|s_j,B|C_{max}$), and identical parallel batch processing machines with non-identical job release times ($P_m|r_j,s_j,B|C_{max}$). New mathematical models are proposed with formulations that exploit characteristics of each problem. The mathematical models are solved using CPLEX and the computational results show that the proposed models performed better than other models from literature. The new models for $1|s_j,B|C_{max}$ and $1|r_j,s_j,B|C_{max}$ are compared with previously published meta-heuristics and the results show that the models provide better solutions than meta-heuristics methods with competitive computational times.

Keywords: batch processing machine, scheduling, mathematical models, mixed integer programming

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

Máquinas de processamento em bateladas (BPM, do inglês: *Batch Processing Machine*) são responsáveis por realizar uma mesma tarefa em um grupo de produtos simultaneamente, visando evitar configurações em equipamentos e facilitar o manuseio de material. Estas máquinas são utilizadas nos mais diferentes seguimentos da indústria, tais como: fabricação de calçados (FANTI et al.,1996), na aeronáutica (ZEE, VAN HARTEN E SCHUUR, 1997), fabricação de tubos de metal (RAM E PATEL, 1998), fundição (MATHIRAJAN, SIVAKUMAR E CHANDRU, 2004), fabricação de móveis (YUAN et al., 2004), entre outros. Embora existam muitas variações do problema envolvendo máquinas de processamento em bateladas, as abordagens neste estudo são motivadas por testes de confiabilidade na indústria de semicondutores, em operações chamadas *burn-in*, apresentadas em Uzsoy (1994).

Nestas, circuitos eletrônicos integrados (CIs) são sujeitos ao estresse térmico por um período prolongado e posteriormente levados a testes de integridade, que podem identificar defeitos latentes, levando ao seu descarte. Isto é feito através da manutenção do circuito a uma temperatura constante, geralmente em um forno com cerca de 120ºC. Outra prática semelhante descrita, em Chung, Tai e Pearn (2009), é a utilização de máquinas de processamento em batelada para testes de envelhecimento de telas de cristal líquido que utilizam transistores de película fina (TFT-LCD). Neste caso, o forno é geralmente regulado a uma temperatura de 55ºC e os painéis a serem testados possuem tempos de montagem diferentes, não estando todos disponíveis no momento em que o processo de testes é iniciado. Como os tempos de montagem são conhecidos, os tempos de liberação dos produtos podem ser definidos *a priori*.

Cada circuito ou painel, denominado de tarefa, necessita de um tempo mínimo dentro de um forno (máquina de processamento), definido a partir de requisitos do fornecedor. As tarefas não podem ser processadas diretamente na máquina, necessitando estarem contidas em bandejas ou impressas em placas (do inglês: *wafers*), denominada batelada, que podem conter um número máximo de tarefas. É possível manter uma tarefa no forno por mais tempo que o definido *a priori*, mas esta não pode ser retirada antes que seu tempo mínimo exigido seja cumprido. Esta característica permite que uma máquina processe tarefas que necessitam de tempos de processamentos diferentes e o tempo de processamento de uma batelada é determinado pelo requisito de tempo de processamento mais longo entre todas as tarefas da

batelada. Em casos que os tempos de liberação (do inglês *release time*) das tarefas não são idênticos, a batelada só poderá começar a ser processada quando todas as tarefas atribuídas a ela estiverem disponíveis. Isto porque uma vez que o processamento é iniciado, o processamento da batelada não pode ser interrompido até que o teste seja concluído.

Os tempos de processamento das operações *burn-in* podem ser longos quando comparados com outras operações de teste, por exemplo, 120h contra 4-5h para outras operações. O processo completo para a produção de um chip pode levar até três meses. Em painéis TFT-LCD os testes levam geralmente 6h e uma máquina pode suportar cerca de 450 partes de painéis. Assim, os testes *burn-in* consistem em um gargalo na operação final de testes e os agendamentos eficientes destas operações visam maximizar a produtividade e reduzir o tempo de fluxo no estoque, o que caracteriza uma grande preocupação para a gestão.

Neste trabalho são abordados quatro variações do problema de programação de máquinas de processamento em bateladas como problemas de otimização, que consideram diferentes premissas. Os problemas são diferenciados a partir da definição formal, descrita a seguir: no problema classificado como $1|s_j,B|C_{max,}$ em um sistema de três campos proposto por Graham, R. L, et al. (1979), existem J tarefas a serem processadas; cada tarefa $j \in J$ exige um tempo mínimo de processamento p_j na máquina; cada tarefa $j \in J$ possui um tamanho s_j; a máquina de processamento possui uma capacidade de espaço B que não pode ser violada; a máquina pode processar somente uma batelada k por vez; a soma dos tamanhos s_j das tarefas j que pertencem à batelada k deve ser menor ou igual à capacidade B; o tempo de processamento de uma batelada k é dado por $\max\{p_j\}$, se a tarefa j pertencer à batelada k. O objetivo do problema é minimizar o tempo de conclusão do da última tarefa j processada (*makespan*, na língua inglesa), dado por C_{max}.

O problema classificado como $1|r_j,s_j,B|C_{max}$ assume todas as premissas do problema anterior, incluindo as que seguem: cada tarefa j possui um tempo de liberação r_j; o tempo de liberação de uma batelada k é dado por $\max\{r_j\}$, se a tarefa j pertencer à batelada k.

Os problemas classificados como $P_m|s_j,B|C_{max}$ e $P_m|r_j,s_j,B|C_{max}$ assumem todas as premissas do primeiro e segundo problemas respectivamente, considerando máquinas de processamento em batelada paralelas idênticas e as seguintes premissas: existem M máquinas de processamento disponíveis; todas as M máquinas possuem a mesma capacidade B; cada batelada k pode ser processada em qualquer máquina $m \in M$.

Os quatro problemas abordados neste trabalho possuem complexidade NP-Difícil, conforme evidenciado em Uzsoy (1994) para o problema $1|s_j,B|C_{max}$, Xu, Chen e Li (2012)

para o problema problema $1|r_j,s_j,B|C_{max}$, Chang, Damodaran e Melouk (2004) para o problema $P_m|s_j,B|C_{max}$ e Damodaran, Vélez-Gallego e Maya (2009) para o problema $P_m|r_j,s_j,B|C_{max}$.

1.1 Objetivos

Neste trabalho são propostas novas formulações matemáticas para quatro variações do problema de dimensionamento e programação de bateladas em máquinas de processamento: $1|s_j,B|C_{max}$, $1|r_j,s_j,B|C_{max}$, $P_m|s_j,B|C_{max}$ e $P_m|r_j,s_j,B|C_{max}$. Os modelos são resolvidos utilizando o resolvedor comercial CPLEX[1] e comparados com outros modelos presentes na literatura, como em Melouk, Damodaran e Chang (2004), Xu, Chen e Li (2012), Chang, Damodaran e Melouk (2004) e Vélez-Gallego, M. C. (2009), que abordam os respectivos problemas indicados.

Este trabalho apresenta um contraponto para utilização de meta-heurísticas nos problemas $1|s_j,B|C_{max}$ e $1|r_j,s_j,B|C_{max}$, mostrando uma comparação direta entre os modelos propostos e abordagens da literatura, como em Chen, Du e Huang (2011) e Xu, Chen e Li (2012), indicando que a utilização de abordagens exatas podem ser melhoradas e aplicadas a estes problemas, demonstrando sua competitividade e, em muitos casos, a superioridade sobre métodos meta-heurísticos.

1.2 Estrutura do trabalho

O trabalho está dividido em seis capítulos. O Capítulo 2 apresenta uma revisão da literatura acadêmica referente à taxonomia dos problemas de programação de tarefas em geral e a apresentação de trabalhos relacionados aos problemas considerados nesta pesquisa.

Nos capítulos posteriores são apresentados os modelos matemáticos publicados na literatura e os propostos neste trabalho para cada um dos quatro problemas de otimização considerados, conforme detalhado a seguir.

[1] IBM CPLEX Optimizer. Acesso: http://www.ibm.com/software/commerce/optimization/cplex-optimizer/

O Capítulo 3 apresenta o problema de minimização do *makespan* de dimensionamento e programação de bateladas em máquina única com tarefas de tamanhos não idênticos $1|s_j,B|C_{max}$. Neste capítulo também é apresentado uma comparação direta entre os modelos apresentados e meta-heurísticas encontradas na literatura.

O Capítulo 4 apresenta o problema de minimização do *makespan* no dimensionamento e programação de bateladas em máquina única com tarefas de tamanhos e tempos de liberação não idênticos $1|r_j,s_j,B|C_{max}$. Neste capítulo também é apresentado uma comparação direta entre os modelos apresentados e meta-heurísticas encontradas na literatura.

O Capítulo 5 apresenta o problema de minimização do *makespan* no dimensionamento e programação de bateladas em máquinas paralelas idênticas com tarefas de tamanhos não idênticos $P_m|s_j,B|C_{max}$. Neste capítulo também é apresentado uma comparação direta entre o modelo apresentado e o encontrado na literatura.

O Capítulo 6 apresenta o problema de minimização do *makespan* no dimensionamento e programação de bateladas em máquinas paralelas idênticas com tarefas de tamanhos e tempos de liberação não idênticos $P_m|r_j,s_j,B|C_{max}$. Neste capítulo também é apresentado uma comparação direta entre o modelo apresentado e o encontrado na literatura.

Os seus resultados computacionais comparativos de cada modelo são apresentados e discutidos em seu respectivo capítulo, em conjunto com suas conclusões. Ao final, são apresentadas conclusões gerais sobre este trabalho e as referências bibliográficas.

2 REVISÃO BIBLIOGRÁFICA

Neste capítulo é apresentada uma revisão de trabalhos encontrados na literatura científica que abordam os problemas tratados nesta pesquisa. A revisão bibliográfica começa apresentando as classificações e a taxonomia dos problemas de programação de máquinas na seção 2.1. As revisões específicas de cada uma das variações do problema de programação de bateladas são apresentadas nas seções 2.2, 2.3, 2.4 e 2.5. Ao final, são levantadas as considerações gerais deste capítulo em 2.6.

2.1 Taxonomia dos problemas de máquinas de processamento

Várias características do problema de dimensionamento e programação de máquinas de processamento podem ser classificadas em um sistema de três campos $\alpha \mid \beta \mid \gamma$, proposto em Graham, et al. (1979). Outras revisões como Potts e Kovalyov (2000), Mathirajan e Sivakumar (2006) e Mönch, et al. (2011) abordam o sistema de três campos e estendem para problemas de dimensionamento e programação de bateladas.

No sistema de classificação em três campos, o campo α refere-se ao ambiente de programação, indicando o tipo e a quantidade das máquinas de processamento envolvidas. Este campo pode conter os atributos tais como:

1 : Máquina única,

F : *Flow shop*,

FJ : *Job shop* flexível,

J : *Job shop*,

O : *Open shop*,

P : Máquinas paralelas idênticas,

Q : Máquinas paralelas uniformes,

R : Máquinas paralelas não relacionadas,

RS : Conjunto de unidades de transporte.

O campo β refere-se às características das tarefas, mostrando informações sobre a existência de relações entre as tarefas ou características individuais. É encontrado neste campo:

B : considera formação de bateladas em máquina(s) com um limite de capacidade,

B_i : considera formação de bateladas em máquinas com capacidades diferentes,

$d_j = d$: considera que todas as tarefas possuem prazos de entrega idênticos,

$F = k$: considera que o número de famílias é uma constante,

$n_f = n/F$: considera que todas as famílias possuem o mesmo número de tarefas,

p_j : considera tarefas com tempos de processamento não idênticos,

p_{jm} : considera tarefas com tempos de processamento não idênticos para cada máquina,

$p_j = p$: considera que todas as tarefas possuem tempos de processamento idênticos,

pmtn : permite a preempção das tarefas,

r_j : considera tarefas com tempos de liberação não idênticos,

s_j : considera tarefas com tamanhos não idênticos,

w_j : considera tarefas com pesos não idênticos,

d_j : considera tarefas com prazos de entrega não idênticos,

s-batch : considera dimensionamento de bateladas por tamanho das tarefas,

batch-incompatible: considera *p-batch* com famílias de tarefas incompatíveis,

$\overline{d}_j = \overline{d}$: considera que todas as tarefas possuem datas de validade idênticas,

$\overline{d}_j$: considera tarefas com datas de validade não idênticas,

s_f= s : considera que todas as famílias de tarefas possuem o mesmo tempo de preparação.

s_f : considera famílias de tarefas com tempos de preparação independentes da sequência e da máquina,

s_{fg} : considera famílias de tarefas com tempos de preparação independentes da máquina,

s_{if} :considera famílias de tarefas com tempos de preparação independentes da sequência,

s_{ifg} : considera famílias de tarefas com tempos de preparação.

Finalmente, o campo γ apresenta o critério de otimalidade considerado pelo problema, contendo a função de custo a ser minimizado. Neste campo pode ser encontrado:

C_{max} : minimizar o *makespan* (tempo de conclusão do processamento da última tarefa processada),

L_{max} : minimizar o *maximum lateness* (maior atraso das tarefas),

$\sum C_j$: minimizar o *total flow time* (tempo total de fluxo),

$\sum T_j$: minimizar o *total tardiness* (atraso total),

$\sum U_j$: minimizar o *number of tardy jobs* (número de tarefas atrasadas),

$\sum w_j C_j$: minimizar o *total weighted flow time* (tempo total de fluxo ponderado),

$\sum w_j E_j$: minimizar o *total weighted earliness* (precocidade total ponderado),

$\sum w_j T_j$: minimizar o *total weighted tardiness* (atraso total ponderado),

$\sum w_j U_j$: minimizar o *weighted number of tardy jobs* (número ponderado de tarefas atrasadas).

Neste estudo são utilizadas as notações abaixo:

$1|s_j,B|C_{max}$: Minimização do *makespan* no problema de dimensionamento e programação de bateladas em máquina única com tarefas de tamanhos não idênticos;

$1|r_j,s_j,B|C_{max}$: Minimização do *makespan* no problema de dimensionamento e programação de bateladas em máquina única com tarefas de tamanhos e tempos de liberação não idênticos;

$P_m|s_j,B|C_{max}$: Minimização do *makespan* no problema de dimensionamento e programação de bateladas em máquinas paralelas idênticas com tarefas de tamanhos não idênticos;

$P_m|r_j,s_j,B|C_{max}$: Minimização do *makespan* no problema de dimensionamento e programação de bateladas em máquinas paralelas idênticas com tarefas de tamanhos e tempos de liberação não idênticos.

2.2 Revisão do problema $1|s_j,B|C_{max}$

A minimização do *makespan* no problema de dimensionamento e programação de bateladas em máquina única com tarefas de tamanhos não idênticos ($1|s_j,B|C_{max}$) é extensamente investigado pela literatura acadêmica. São encontradas várias abordagens para a resolução deste problema utilizando vários métodos computacionais, incluindo heurísticas, meta-heurísticas, algoritmo aproximativo e métodos exatos.

Uzsoy (1994) foi o primeiro trabalho a considerar o problema $1|s_j,B|C_{max}$, afirmando a sua complexidade NP-Difícil. A partir de trabalhos prévios sobre o problema *bin-packing*, são propostas heurísticas baseadas no método *first-fit* utilizando diferentes métodos de ordenações das tarefas: por tempo de processamento, tamanho e proporção entre tempo e tamanho das tarefas.

Ghazvini e Dupont (1998) propõe a utilização de heurísticas baseadas no método *best-fit* para a resolução deste problema. São apresentadas variações que necessitam de uma ordenação prévia das tarefas, assim como Uzsoy (1994). Também é proposta uma heurística

iterativa chamada DYNA, que apresentou as melhores soluções na análise comparativa. Este trabalho trata instâncias com até cem tarefas.

Em Zhang et al. (2001) é proposto um algoritmo aproximativo para casos gerais e outro para casos especiais. Este trabalho considera instâncias nas quais a máquina apresenta capacidade unitária e os tamanhos das tarefas são distribuídos no intervalo (0, 1]. Foi considerado um caso especial quando os tempos de processamento das tarefas grandes (maiores que 0,5) não são menores do que das tarefas pequenas (com tamanhos não maiores que 0,5). Neste caso, o algoritmo prova que obtém soluções com razão de aproximação igual a 3/2 da solução ótima, para o pior caso. Para o algoritmo de casos gerais, é provada a aproximação de 7/4 da solução ótima, no pior caso.

Em Dupont e Dhaenens-Flipo (2002) é apresentada uma abordagem exata para o problema. É proposto neste trabalho um sistema de enumeração reduzida para o método *branch-and-bound*, que utiliza regras de dominância a partir de propriedades do problema. Este trabalho considera instâncias que possuem a máquina com capacidade igual a cem. Os resultados demonstram que a enumeração proposta tem um bom desempenho em instâncias que possuem tarefas pequenas com tamanhos mais distribuídos ([1, 10]), provando a otimalidade das soluções. Para instâncias com uma distribuição de tamanhos das tarefas menor ([5, 10]), o algoritmo enfrenta mais dificuldade, mas encontra soluções de boa qualidade. Em instâncias com tarefas maiores ([5,20] e [5,30]) o método exige um custo computacional maior, principalmente em instâncias com um número maior de tarefas. Este trabalho trata instâncias de até quatrocentas tarefas.

Melouk, Damodaran e Chang (2004) aborda o problema propondo a utilização da meta-heurística *Simulated Annealing* (SA). Neste trabalho, a solução inicial é gerada aleatoriamente, mas formando bateladas que respeitam a capacidade da máquina. A vizinhança é definida pelo movimento de troca entre duas tarefas j e k de bateladas diferentes. O movimento de troca é efetuado sempre que duas condições são satisfeitas: se a tarefa j possuir um tempo de processamento menor que a tarefa k, e se a tarefa k tiver o mesmo tempo de processamento da batelada a qual está contida. Este trabalho propõe configurações para a geração de instâncias utilizadas amplamente na literatura científica. Os resultados computacionais são comparados com o CPLEX 7.1, utilizando um modelo de programação inteira mista, proposto pelos autores. Com o SA proposto é possível encontrar soluções melhores que CPLEX com um tempo limite de 1800 segundos, principalmente em instâncias de maior porte. Este trabalho trata instâncias de até cem tarefas.

Damodaran, Manjeshwar e Srihari (2006) propõe a utilização de um Algoritmo Genético (AG). Na codificação utilizada, cada tarefa é representada por um gene e cada cromossomo constitui uma solução completa, representada por uma sequência de tarefas. Na decodificação da solução é utilizada a heurística *best-fit* (GHAZVINI E DUPONT, 1998) a partir da ordem das tarefas contida em cada cromossomo para formação de bateladas. Uma parte da população inicial é criada de forma aleatória e a outra pelas heurísticas *first-fit* (UZSOY, 1994) e *best-fit*, com ordenação decrescente pelos tempos de processamento. A função de aptidão (*fitness*) é calculada pelo valor do C_{max}. A seleção de soluções para cruzamento (*crossover*) utiliza o método de seleção por torneio (*tournament selection*). É adotado o mecanismo de cruzamento de ponto único, onde cada cromossomo sorteado é dividido em duas partes. A parte esquerda do cromossomo é mantida e as demais tarefas são inseridas novamente de acordo com a ordem em que são encontradas no outro cromossomo sorteado. O processo de mutação consiste em efetuar a troca de posição de duas tarefas adjacentes. Os resultados computacionais são comparados com os do algoritmo SA proposto por Melouk, Damodaran e Chang (2004) e com o CPLEX 7.1. A abordagem proposta demonstra ser mais robusta, encontrando soluções melhores de forma mais consistente. Quando comparado ao CPLEX 7.1, utilizando o modelo proposto por Melouk, Damodaran e Chang (2004), a abordagem proposta encontra soluções melhores com menor tempo de processamento em todas as instâncias. Este trabalho trata instâncias de até cem tarefas. As médias de tempo de execução do AG nas instâncias de maior porte variam entre 76,88s até 79,22s, enquanto o SA apresenta tempos entre 338,88s até 2404,16s em um Pentium 3, 500 MHz com 261MB de memória RAM.

Em Kashan, Karimi, e Jolai (2006) é encontrada outra abordagem que utiliza um Algoritmo Genético e um Algoritmo Genético Híbrido. Na proposta do Algoritmo Genético, são utilizadas chaves aleatórias (do inglês: *random keys*) para representar as soluções e formar os cromossomos. Nesta representação, é gerada uma chave aleatória correspondente para cada tarefa e posteriormente é realizada uma ordenação destas tarefas por ordem crescente de chaves que define a sequência com a qual um algoritmo construtivo, do tipo *best-fit*, considera as tarefas para gerar uma solução. Para o Algoritmo Genético Híbrido, cada gene de um cromossomo representa a batelada na qual a tarefa correspondente está contida. A população inicial para os dois algoritmos é gerada aleatoriamente. Para o procedimento do cruzamento, duas soluções são selecionadas utilizando roleta ponderada (*Roulette wheel selection*), de acordo com a função de aptidão calculada pelo valor do C_{max}. O procedimento de cruzamento utiliza o método de cruzamento uniforme parametrizado (*parameterized uniform crossover*).

O processo de mutação escolhe aleatoriamente dois cromossomos, efetuando a troca de posições. Uma heurística de factibilização é utilizada para garantir que nenhuma batelada exceda a capacidade da máquina, como também uma heurística de busca local gulosa para reduzir o valor do C_{max} de uma parcela da população. Os resultados computacionais obtidos pelos métodos propostos são comparados com o SA proposto em Melouk, Damodaran e Chang (2004), demonstrando que Algoritmo Genético Híbrido é superior em tempo e qualidade de solução.

Em Meng e Tang (2010) é encontrado um estudo sobre o problema que utiliza o método Busca Tabu para melhorar uma solução inicial, encontrada a partir de uma heurística gulosa baseada em programação dinâmica. Na Busca Tabu são consideradas duas vizinhanças básicas distintas. A primeira efetua a troca entre duas tarefas de bateladas diferentes. A segunda efetua a troca de uma tarefa por outras duas de uma batelada diferente. Além destas, são utilizadas outras três vizinhanças compostas por uma série de movimentos realizados entre bateladas adjacentes. Estes movimentos são realizados entre as K primeiras bateladas, em que K representa a profundidade da vizinhança. A primeira série realiza movimentos de troca de uma tarefa da batelada k por outra tarefa da batelada k+1. A segunda série realiza movimentos de inserção de uma tarefa da batelada k na batelada k+1. A terceira série realiza os mesmos movimentos de troca da segunda série, porém incluindo uma nova batelada vazia. A profundidade K é definida dinamicamente a partir da estratégia *filter and fan*, que realiza uma busca por amplitude até que seja encontrada uma solução melhor que a solução incumbente ou o número máximo para K seja atingido. Os autores consideraram a abordagem eficaz e compararam os tempos computacionais com o CPLEX utilizando um modelo matemático similar ao proposto por Melouk, Damodaran e Chang (2004), comparando instâncias com até cem tarefas.

Em Parsa, Karimi e Kashan (2011) é apresentada uma formulação para o problema utilizando a decomposição de Dantzing-Wolfe como um problema de particionamento, utilizando o método geração de colunas para proporcionar um limitante inferior de boa qualidade. O método *branch-and-price* é utilizado obter a solução inteira ótima para o problema. Os resultados computacionais são comparados com o método *branch-and-bound* proposto em Dupont e Dhaenens-Flipo (2002) e revelam tempos computacionais menores e soluções melhores. Este trabalho trata instâncias de até quinhentas tarefas.

Em Chen, Du e Huang (2011) é resolvido este problema utilizando uma heurística baseada em um caso especial do problema de agrupamento (do inglês: *Clustering Problem*). É introduzida a definição de proporção de resíduos da batelada, provando que sua minimização

é equivalente ao objetivo original do problema. Os resultados computacionais mostraram que o algoritmo proposto alcança resultados superiores aos da heurística proposta por Ghazvini e Dupont (1998) e aos do Algoritmo Genético proposto por Damodaran, Manjeshwar e Srihari (2006), principalmente em instâncias com maior número de tarefas. Este trabalho trata instâncias de até quinhentas tarefas.

Em Lee e Lee (2013) são propostos dois tipos de heurísticas baseadas em uma decomposição do modelo original do problema, criando duas heurísticas que utilizam modelos matemáticos relaxados. Estes modelos selecionam as tarefas que irão compor as bateladas criadas. Na primeira, várias bateladas são geradas simultaneamente a cada iteração do algoritmo, até que todas as tarefas tenham sido alocadas. Na segunda, cada batelada é gerada separadamente de maneira gulosa. Os modelos utilizam o critério de minimização da proporção dos resíduos nas bateladas, proposto por Chen, Du e Huang (2011). A abordagem proposta encontra soluções melhores quando comparadas com os algoritmos propostos por Ghazvini e Dupont (1998), mas necessitam de um maior custo computacional. Os autores concluem que este tempo computacional é aceitável, mesmo sendo mais alto.

Em Damodaran, Ghrayeb e Guttikonda (2013) é proposta a utilização do método GRASP para tratar o problema. Para encontrar a solução inicial é utilizado o algoritmo *first-fit* com um grau de aleatorização no critério guloso de escolha, assumindo que as tarefas foram pré-ordenadas de forma decrescente pelos seus tempos de processamento. A solução é melhorada utilizando um algoritmo de busca local que utiliza operadores de troca (realiza a troca de duas tarefas de bateladas diferentes) e inserção (retira a tarefa de uma batelada e a insere em outro). Os resultados computacionais são comparados com outras abordagens como: CPLEX e *Simulated Annealing* (MELOUK, DAMODARAN e CHANG, 2004) e Algoritmo Genético (DAMODARAN, MANJESHWAR e SRIHARI, 2006), revelando a superioridade da abordagem proposta em instâncias testadas que possuem mais de 20 tarefas, ressaltando que seus tempos computacionais são semelhantes aos do Algoritmo Genético, mas encontra soluções melhores. Este trabalho trata instâncias de até cem tarefas.

2.3 Revisão do problema 1| $r_j,s_j,B|C_{max}$

Existem poucos trabalhos na literatura científica que investigam o problema de dimensionamento e programação de bateladas em máquina única com minimização do

makespan no qual as tarefas têm tamanhos e tempos de liberação não idênticos ($1|r_j,s_j,B|C_{max}$). Até o presente momento, no melhor conhecimento do autor, são encontrados apenas dois trabalhos que utiliza meta-heurísticas para sua resolução.

O estudo apresentado por Chou, Chang e Wang (2005) trata o problema propondo duas abordagens de Algoritmos Genéticos Híbridos para o problema. O problema é separado em três estágios: sequenciamento das tarefas, alocação das tarefas em bateladas e alocação das bateladas na máquina. O Algoritmo Genético é utilizado para determinar a sequência de tarefas. As duas abordagens se diferenciam no segundo estágio, em que uma delas utiliza um algoritmo de refinamento, chamado de *Merge-Split* (Mesclar e Dividir), também sugerido neste trabalho. No terceiro estágio é utilizada uma ordenação das bateladas por tempo de liberação para determinar o escalonamento na máquina de processamento. A utilização dos algoritmos propostos apresenta uma boa qualidade nas soluções encontradas, mas se mostrou pouco satisfatória em tempos computacionais quando o número de tarefas aumenta. Foram utilizadas instâncias de até cem tarefas.

Em Xu, Chen e Li (2012) é proposto um método Colônia de Formigas. Este trabalho traz a indicação de que o problema tratado é NP-Difícil. É introduzida a definição de tempo ocioso e espaço desperdiçado (do inglês: *Waste and Idle Space*), provando que sua minimização é equivalente ao objetivo original do problema. Neste estudo também é possível encontrar uma proposta válida para a criação de um limitante inferior. Para a geração das instâncias é ressaltado que quando o limite de distribuição dos tempos de liberação das tarefas é curto, ou seja, próximos à zero, o problema se torna muito parecido com $1|s_j,B|C_{max}$. Em contraponto, quando os tempos de liberação possuem um longo intervalo, o espaço solução do problema se torna pequeno e, como consequência, a solução do problema é trivial. Esta abordagem apresenta resultados satisfatórios em comparação com o CPLEX 11.0 utilizando um modelo matemático sugerido neste trabalho, e o Algoritmo Genético Híbrido publicado em Chou, Chang e Wang (2005), encontrando soluções melhores em tempos razoáveis.

2.4 Revisão do problema $P_m|s_j,B|C_{max}$

O problema de dimensionamento e programação de bateladas em máquinas paralelas idênticas com minimização do *makespan* no qual as tarefas têm tamanhos não idênticos

($P_m|s_j,B|C_{max}$) é abordado nesta seção. Os trabalhos que investigam esse problema, em sua maioria, são extensões de trabalhos já publicados para o problema $1|s_j,B|C_{max.}$

Em Chang, Damodaran e Melouk (2004) é utilizado o método *Simulated Annealing*. Este trabalho traz a indicação de que o problema tratado é NP-Difícil. A implementação desta abordagem é feita de modo similar à encontrada em Melouk, Damodaran e Chang (2004), mas acrescentando uma etapa que sequencia as bateladas nas máquinas paralelas conforme sua disponibilidade, similar ao *first-fit*. É encontrada também uma formulação de um modelo matemático de programação inteira mista. Os resultados computacionais comprovam que, quando comparado com o CPLEX 7.1 utilizando o modelo matemático, esta abordagem encontra melhores soluções em um menor tempo. Este trabalho também relata dificuldades na utilização do CPLEX 7.1 como falta de memória e soluções de baixa qualidade quando o número de tarefas aumenta, com o tempo máximo de execução limitado em uma hora. Este trabalho trata instâncias de até cinquenta tarefas.

Em Kashan, Karimi e Jenabi (2008) é utilizado um Algoritmo Genético Híbrido. Esta abordagem utiliza uma representação similar à encontrada em Kashan, Karimi, e Jolai (2006) para o problema em máquina única. Duas heurísticas baseadas em busca local são sugeridas para melhorar a qualidade das soluções geradas. As bateladas geradas são ordenadas de forma crescente pelos tempos de processamento e escalonadas nesta ordem nas máquinas. Os resultados desta abordagem são comparados com o método *Simulated Annealing* porposto por Chang, Damodaran e Melouk (2004), apresentando soluções superiores em tempos razoáveis. Este trabalho trata instâncias com até cem tarefas e com duas e quatro máquinas paralelas.

Em Damodaran, Hirani e Galego (2009) é proposta uma nova abordagem para o Algoritmo Genético, muito similar à encontrada em Damodaran, Manjeshwar e Srihari (2006) que trata o problema com máquina única, com adição de uma heurística baseada no método *first-fit* para sequenciar as bateladas às máquinas paralelas, utilizando uma ordenação prévia de forma crescente por tempo de processamento. Os resultados desta abordagem são comparados com o Algoritmo Genético Híbrido de Kashan, Karimi e Jenabi (2008), com Algoritmo Genético e com o CPLEX 7.1. Os resultados demonstram que a abordagem proposta consome um tempo computacional muito inferior às demais abordagens, mas encontra soluções piores que o Algoritmo Genético Híbrido. Segundo os autores, esta diferença na qualidade das soluções é pequena. Este trabalho trata instâncias com até cem tarefas e com duas e quatro máquinas paralelas.

Em Cheng et al. (2012) é apresentado um modelo de programação inteira mista e proposto um algoritmo aproximativo para o tratamento do respectivo problema. É provado que o algoritmo obtém soluções com razão de aproximação igual a 2 da solução ótima, para o pior caso.

Em Cheng et al. (2013) é encontrada a aplicação do método Colônia de Formigas para este problema. Nesta abordagem, cada solução é representada pela ordem com que as tarefas são distribuídas em um vetor, formando um caminho (*patch*). As bateladas são formadas pelo algoritmo *first-fit* e posteriormente ordenadas de forma crescente de acordo com o seus tempos de processamento para então serem sequenciadas, de forma que cada batelada seja atribuída à primeira máquina ociosa disponível. Os resultados computacionais são apresentados e comparados com outros trabalhos publicados como *Simulated Annealing* proposto por Chang, Damodaran e Melouk (2004) e Algoritmo Genético Híbrido proposto por Kashan, Karimi e Jenabi (2008), revelando que a Colônia de Formigas encontra melhores soluções em menos tempo computacional. Este trabalho trata instâncias com até quinhentas tarefas e com quatro e oito máquinas paralelas.

2.5 Revisão do problema $P_m|r_j,s_j,B|C_{max}$

O problema de dimensionamento e programação de bateladas em máquinas paralelas idênticas com minimização do *makespan* no qual as tarefas têm tamanhos e tempos de liberação não idênticos ($P_m|r_j,s_j,B|C_{max}$) teve o primeiro trabalho publicado por Chung, Tai e Pearn (2009). Neste estudo os autores propõem um modelo matemático de Programação Inteira Mista para a resolução do problema. É sugerido também um algoritmo combinado em que inicialmente é calculado um número mínimo de bateladas necessárias para a solução do problema e, posteriormente, o modelo matemático é resolvido utilizando este número como parâmetro. Na sequência o número de bateladas é incrementado e o modelo é resolvido novamente. O processo é iterado até que a solução corrente seja igual ou maior que a solução da iteração anterior ou o número de bateladas seja maior que o número de tarefas. São propostas também três heurísticas para tratar instâncias de maior porte (com quinze tarefas), na qual a terceira é apenas a escolha da melhor solução entre as outras duas. Os autores consideram que os resultados computacionais do modelo matemático proposto são

superados pelos do algoritmo combinado e as heurísticas encontram boas soluções em pouco tempo computacional.

Em sua tese de doutorado, Vélez-Gallego, M. C. (2009) prova que o problema em estudo possui complexidade NP-Difícil. O problema é abordado utilizando um modelo matemático de programação mista, cinco heurísticas e duas meta-heurísticas. Também é apresentada uma proposta de geração de colunas, que obtém limitantes inferiores não inteiros. São propostas as meta-heurísticas SA e GRASP para a resolução do problema. Na comparação, o método GRASP encontra soluções melhores, mas com um tempo maior. O modelo matemático proposto é comparado ao modelo proposto em Chung, Tai e Pearn (2009), utilizando o resolvedor XPRESS - IVE ® em instâncias pequenas de até 10 tarefas, demonstrando sua superioridade. Os testes do modelo proposto são apresentados para instâncias de até 50 tarefas. Este trabalho considera a utilização de três e cinco máquinas paralelas. Os resultados deste trabalho são publicados em Damodaran, Vélez-Gallego e Maya (2009).

2.6 Considerações finais do capítulo

Neste capítulo foram apresentados trabalhos relacionados aos problemas estudados nesta pesquisa, trazendo um breve resumo dos conteúdos ali contidos. Algumas observações podem ser feitas a partir deste levantamento.

Uma destas observações diz respeito à quantidade de métodos heurísticos e meta-heurísticos abordados. Poucos métodos exatos são propostos e, invariavelmente, apresentam desempenho muito ruim em resultados comparativos com meta-heurísticas. Vários trabalhos citam a dificuldade em tratar os problemas em questão quando o número de tarefas aumenta, impedindo a utilização das abordagens exatas encontradas na literatura científica em um tempo computacional aceitável.

A distribuição de tempos de liberação em instâncias para problemas em que este fator é considerado também é passível de uma observação importante encontrada em Xu, Chen e Li (2012). Nele, é constatado que quando a distribuição dos tempos de liberação das tarefas é menos esparsa e próximos à zero, o problema se torna muito parecido com o problema em que os tempos de liberação não são considerados.

3 MODELOS MATEMÁTICOS PARA O PROBLEMA $1|s_j,B|C_{max}$

Neste capítulo é proposto um modelo matemático de programação binária para minimização do *makespan* no problema de dimensionamento e programação de bateladas em máquina única com tarefas de tamanhos não idênticos ($1|s_j,B|C_{max}$). Suas definições, exemplos, modelo matemático encontrado na literatura e observações relevantes são apresentados na seção 3.1. Um novo modelo matemático de programação binária é proposto na seção 3.2. Os resultados computacionais dos modelos matemáticos são comparados e discutidos na seção 3.3. As conclusões deste capítulo são apresentadas na seção 3.4.

3.1 Descrição do problema e modelo da literatura

Formalmente, este problema pode ser definido como: dado um conjunto J de tarefas, cada elemento $j \in J$ pode ser definido por uma dupla de atributos $\{p_j, s_j\}$, que representam o tempo de processamento e o tamanho da tarefa, respectivamente. Cada tarefa $j \in J$ deve ser designada a uma batelada $k \in K$, respeitando o limite de capacidade B da máquina. Cada batelada k será designada a máquina de processamento. O tempo de processamento de cada batelada $k \in K$ é definido como $P_k = \max\{p_j, j \in k\}$. Todas as tarefas são disponíveis no tempo zero, ou seja, o problema não considera tarefas com tempos de liberação diferentes, $\{r_j = 0, j \in J\}$. As tarefas não podem ser divididas entre as bateladas, não sendo possível adicioná-las ou removê-las da máquina durante o seu processamento. O objetivo é encontrar uma configuração de bateladas $k \in K$ e uma ordem sequenciamento na máquina de processamento, de modo que o tempo de processamento da máquina C_{max} (*makespan*) seja minimizado. O número de bateladas utilizadas depende do número de tarefas e da capacidade da máquina consideradas na instância, sendo que no pior caso, este número será igual à quantidade de tarefas, $|K| = |J|$.

Um exemplo pode ser montado a partir de uma instância de dez tarefas, apresentado na Tabela 3.1. A representação gráfica da solução ótima deste exemplo é ilustrada na Figura 3.1, onde cada tarefa é representada por um retângulo, cujas dimensões de altura e largura representam os valores de tamanho e tempo de processamento da tarefa, respectivamente. Estas dimensões são destacadas na Figura 3.1 através da tarefa 1. A capacidade B da máquina

adotada para este exemplo é de 8 unidades. A solução ótima deste exemplo apresenta cinco bateladas utilizadas e o C_{max} é de 23 unidades de tempo. Pode-se destacar que as bateladas 1 e 3 não utilizam a sua capacidade máxima, caracterizando um desperdício inevitável de espaço.

Tabela 3.1 - Dados para instância exemplo do problema $1|s_j,B|C_{max}$.

j	1	2	3	4	5	6	7	8	9	10
p_j	9	5	8	2	6	3	4	1	2	5
s_j	3	6	1	2	4	3	2	5	3	6

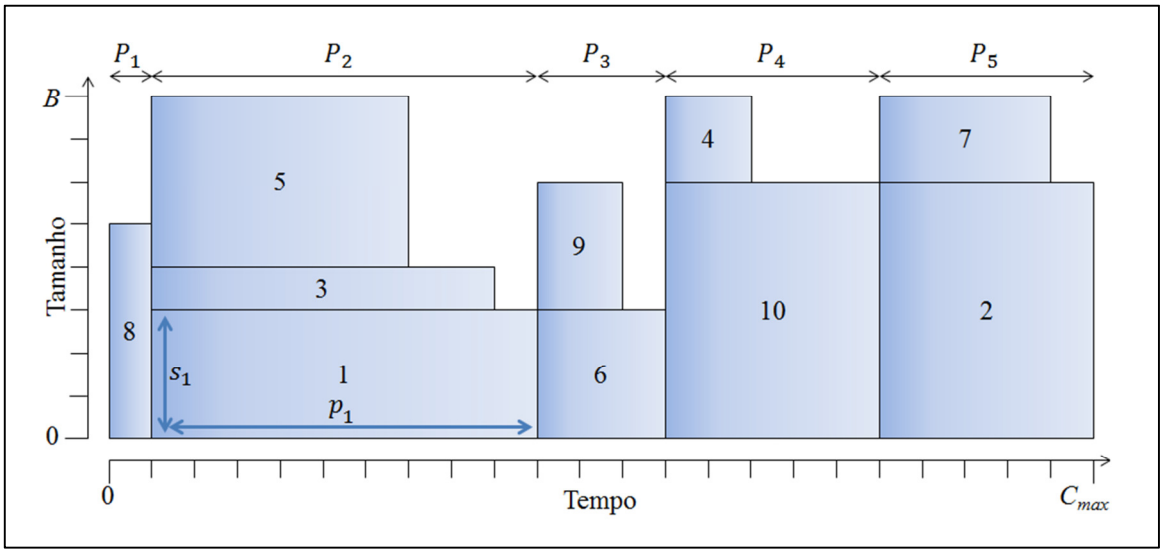


Figura 3.1 - Ilustração da instância exemplo do problema $1|s_j,B|C_{max}$.

A complexidade deste problema é considerada NP-Difícil, uma vez que quando todas as tarefas possuem tempos de processamento idênticos $\{p_j = P, \forall j \in J\}$, o problema $1|s_j,B|C_{max}$ se torna equivalente ao problema *bin packing* com capacidade do *bin* igual a B e tamanho dos itens a_j (UZSOY, 1994).

Em Melouk, Damodaran e Chang (2004) é proposto um modelo de programação linear inteira mista para a resolução do problema. Modelos muito similares ou iguais a este são utilizados por outros trabalhos posteriores como uma base comparativa nos experimentos computacionais. O modelo é definido a seguir:

Conjuntos:

J : conjunto de tarefas, $j \in J$

K: conjunto de bateladas, $k \in K$

Parâmetros:

B : capacidade da máquina

p_j: tempo de processamento da tarefa j

s_j: tamanho da tarefa j

Variáveis:

$x_{jk} = \begin{cases} 1 \text{ se a tarefa } j \in J \text{ for atribuída à batelada } k \in K \\ 0 \text{ caso contrário} \end{cases}$

$y_k = \begin{cases} 1 \text{ se a batelada } k \in K \text{ for utilizada} \\ 0 \text{ caso contrário} \end{cases}$

P_k : tempo de processamento da batelada $k \in K$

Modelo 3.1

$$Min \sum_{k \in K} P_k \tag{3.1}$$

S.a.

$$\sum_{j \in J} s_j x_{jk} \leq B y_k \qquad \forall k \in K \tag{3.2}$$

$$\sum_{k \in K} x_{jk} = 1 \qquad \forall j \in J \tag{3.3}$$

$$P_k \geq p_j x_{jk} \qquad \forall k \in K, \forall j \in J \tag{3.4}$$

$$x_{jk} \leq y_k \qquad \forall k \in K, \forall j \in J \tag{3.5}$$

$$P_k \geq 0 \qquad \forall k \in K, \forall j \in J \tag{3.6}$$

$$y_k \in \{0,1\} \qquad \forall k \in K, \forall j \in J \tag{3.7}$$

$$x_{jk} \in \{0,1\} \qquad \forall k \in K, \forall j \in J \tag{3.8}$$

A função objetivo (3.1) minimiza o tempo de utilização da máquina (*makespan*). A restrição (3.2) determina que cada batelada, se utilizada, não exceda a capacidade da máquina. A restrição (3.3) determina que cada tarefa seja designada somente a uma batelada. A restrição (3.4) determina o tempo de processamento de cada batelada. A restrição (3.5) é redundante com a (3.2), mas é inserida porque melhora o desempenho computacional em resolvedores baseados em relaxações lineares. As restrições (3.6), (3.7) e (3.8) determinam o domínio das variáveis.

É comum encontrar na literatura vários trabalhos que utilizam os resultados computacionais obtidos através do resolvedor CPLEX na comparação de suas abordagens. O Modelo 3.1 é utilizado para este fim em Melouk, Damodaran e Chang (2004) e Damodaran, Ghrayeb e Guttikonda (2013). Modelos similares, que não consideram a restrição (3.5) na formulação, também são utilizados em Damodaran, Manjeshwar e Srihari (2006) e Rafiee Parsa, Karimi e Kashan (2011).

O problema $1|s_j,B|C_{max}$, pode ser considerado altamente simétrico em relação a ordem de programação das bateladas na máquina. Isto ocorre porque uma mesma solução pode ser representada de diferentes formas, apenas permutando a sequência das bateladas. Estas permutações representam soluções equivalentes e, consequentemente, o mesmo valor para C_{max}. O Modelo 3.1 considera estas representações como soluções distintas, gerando um grande espaço solução onde várias soluções possíveis possuem equivalência.

Na Figura 3.2 é ilustrado um exemplo em que duas soluções equivalentes, (a) e (b), são formadas a partir da instância apresentada na Tabela 3.1. As duas soluções apresentadas possuem as mesmas formações de bateladas e C_{max}. A diferença entre as soluções é o sequenciamento das bateladas na máquina. Várias outras permutações são facilmente identificadas a partir deste exemplo.

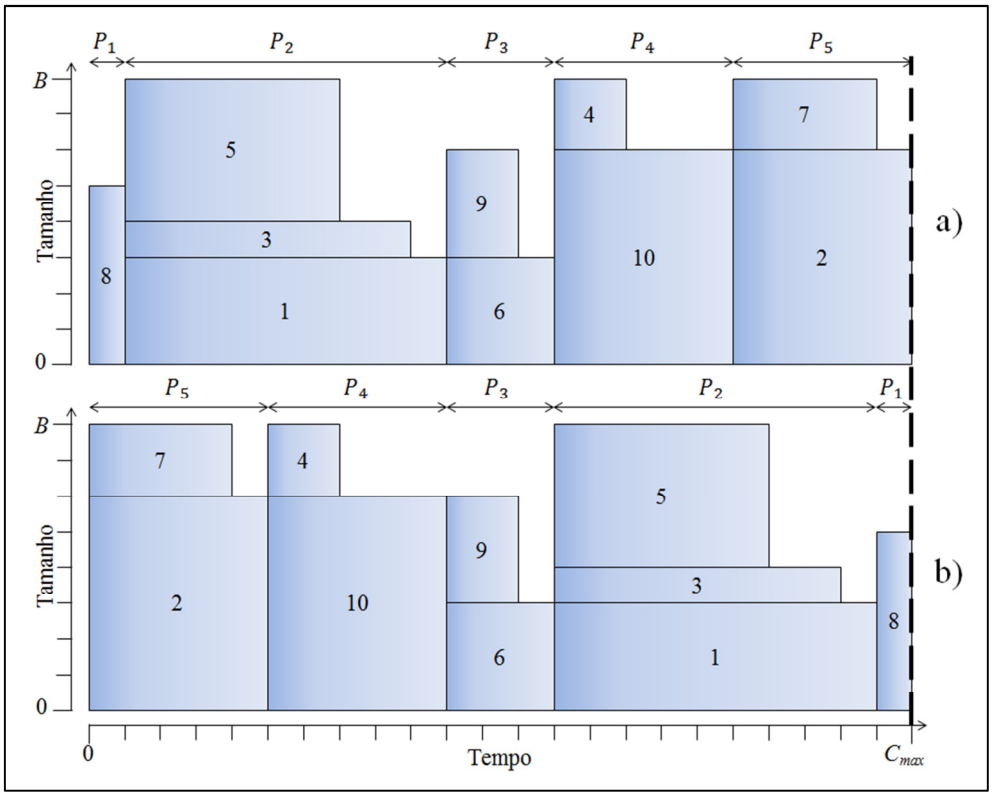


Figura 3.2 - Exemplo de soluções simétricas do o problema.

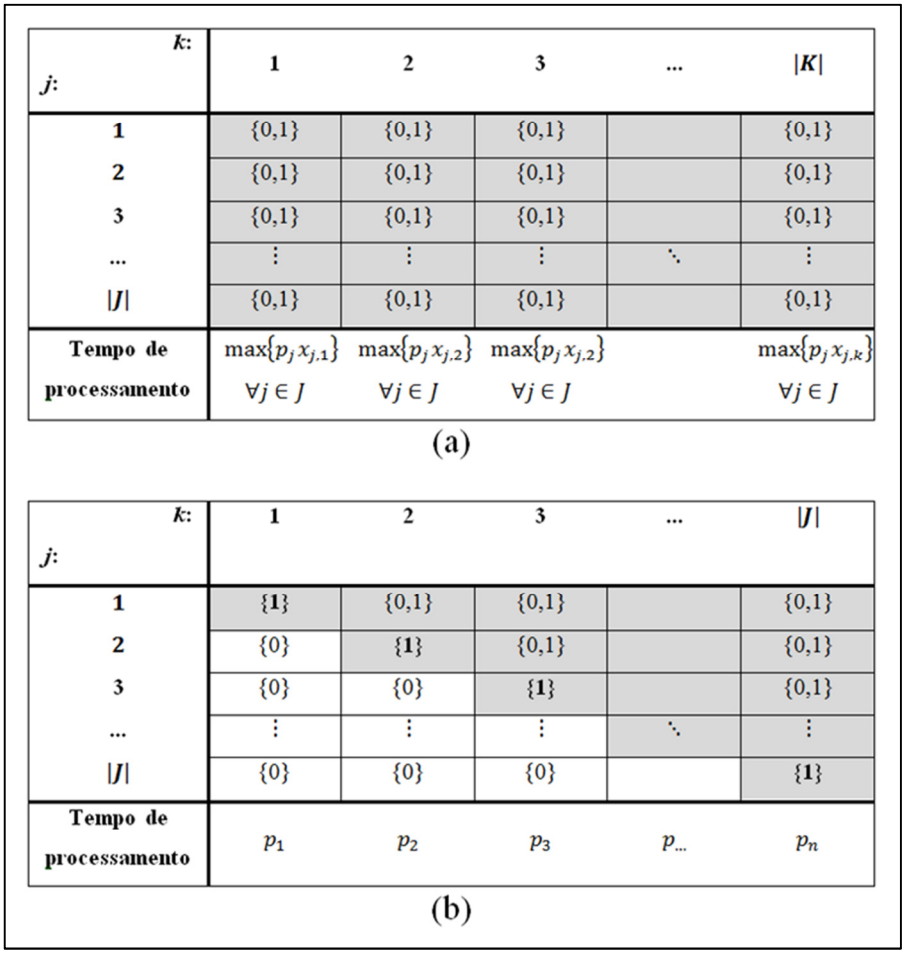
3.2 Modelo proposto

Visando reduzir o espaço solução deste problema, é proposto um modelo baseado em restrições que retiram soluções simétricas do espaço solução, fornecendo melhores limitantes pela relaxação contínua do problema, não alterando a solução ótima do problema original. O novo conjunto de restrições necessita que as tarefas sejam ordenadas pelos seus tempos de processamento de forma não decrescente.

Considerando o número bateladas m igual ao número de tarefas n, pode-se definir que a batelada k só será utilizada se a tarefa k for atribuída a ele e não conter as tarefas $j: j > k$. Ou seja, para que a batelada k seja utilizada, a variável x_{kk} deve ser igual a 1 e as variáveis $x_{jk}, j > k$ são desconsideradas. Com isso, tendo as tarefas já ordenadas pelos seus tempos de processamento, pode ser definido explicitamente que a batelada k sempre tem a tarefa k como a de maior tempo de processamento.

A Figura 3.3 ilustra a possibilidade de configuração de cada batelada, se utilizada, e seu respectivo tempo de processamento. Nota-se que para o modelo original (a), é utilizada a

matriz completa das variáveis x_{jk} e, no modelo proposto (b), é utilizada somente a matriz triangular superior, destacando a diagonal principal como variáveis obrigatórias se a batelada for utilizada. Também é ilustrada em (b) a definição explícita do tempo de processamento de cada batelada, se utilizada.



k: / j:	1	2	3	...	$\|K\|$
1	{0,1}	{0,1}	{0,1}		{0,1}
2	{0,1}	{0,1}	{0,1}		{0,1}
3	{0,1}	{0,1}	{0,1}		{0,1}
...	⋮	⋮	⋮	⋱	⋮
$\|J\|$	{0,1}	{0,1}	{0,1}		{0,1}
Tempo de processamento	$\max\{p_j x_{j,1}\}$ $\forall j \in J$	$\max\{p_j x_{j,2}\}$ $\forall j \in J$	$\max\{p_j x_{j,2}\}$ $\forall j \in J$		$\max\{p_j x_{j,k}\}$ $\forall j \in J$

(a)

k: / j:	1	2	3	...	$\|J\|$
1	{1}	{0,1}	{0,1}		{0,1}
2	{0}	{1}	{0,1}		{0,1}
3	{0}	{0}	{1}		{0,1}
...	⋮	⋮	⋮	⋱	⋮
$\|J\|$	{0}	{0}	{0}		{1}
Tempo de processamento	p_1	p_2	p_3	$p_{...}$	p_n

(b)

Figura 3.3 - Ilustração da formação das bateladas.

Desta forma a simetria é tratada, pois a batelada k não poderá ter uma configuração semelhante a nenhuma outra batelada $j: j \neq k$, além de ser definida uma ordem para seu escalonamento na máquina, pelo tempo de processamento das bateladas de forma não crescente.

O Modelo 3.2 proposto definido a seguir utiliza os mesmos parâmetros do Modelo 3.1:

Modelo 3.2

$$Min \sum_{k \in K} p_k x_{kk} \tag{3.9}$$

S.a.

$$\sum_{\substack{j \in J: \\ j \leq k}} s_j x_{jk} \leq B x_{kk} \qquad \forall k \in K \tag{3.10}$$

$$\sum_{\substack{k \in K: \\ j \leq k}} x_{jk} = 1 \qquad \forall j \in J \tag{3.11}$$

$$x_{jk} \leq x_{kk} \qquad \forall j \in J, \forall k \in K: j < k \tag{3.12}$$

$$x_{jk} \in \{0,1\} \qquad \forall j \in J, \forall k \in K: j \leq k \tag{3.13}$$

A função objetivo (3.9) minimiza o tempo de utilização da máquina (*makespan*). A restrição (3.10) determina que cada batelada, se utilizada, não exceda a capacidade da máquina. A restrição (3.11) determina que cada tarefa só pode ser designada a uma batelada com índice maior ou igual ao índice da própria tarefa. A restrição (3.12) é redundante com a restrição (3.10), no entanto foi inserida porque, quando utilizada em resolvedores baseados em relaxações lineares, o desempenho computacional é melhorado. A restrição (3.13) determina o domínio binário das variáveis de decisão.

A Tabela 3.2 apresenta um exemplo montado a partir da instância descrita na Tabela 3.1, em que as tarefas e seus atributos são ordenados de forma crescente a partir dos tempos de processamento. Os índices das tarefas j são alterados para j' quando utilizados no Modelo 3.2. Depois de encontrada a solução final, um mapeamento é necessário para que se descubra o índice original de cada tarefa.

Tabela 3.2 - Dados para instância exemplo do problema $1|s_j,B|C_{max}$ com ordenação das tarefas por tempo de processamento.

$\boldsymbol{j}$	8	4	9	6	7	2	10	5	3	1
$\boldsymbol{j'}$	1	2	3	4	5	6	7	8	9	10
$\boldsymbol{p_{j'}}$	1	2	2	3	4	5	5	6	8	9
$\boldsymbol{s'_{j'}}$	5	2	3	3	2	6	6	4	1	3

Para demonstrar a diferença na representação de uma solução entre os modelos, este exemplo estende-se a seguir: dada uma batelada Q_2 que contenha três tarefas j, $Q_2 = \{1,3,5\}$, esta poderá ser representada de diversas formas como ilustrado na Figura 3.4 (a) quando utilizado o Modelo 3.1 e assumindo que $|K| = |J|$. Para que seja representada no Modelo 3.2, esta mesma batelada Q_2 terá os índices das tarefas alterados para j'. A batelada conterá então as tarefas $Q'_2 = \{10,9,8\}$ de acordo com a Tabela 3.2. Neste modelo, esta batelada só terá uma única representação no espaço solução, como ilustrado na Figura 3.4 (b). As representações mostram as variáveis que assumem o valor igual a 1 e para todas as outras variáveis de cada conjunto o valor é igual a 0. Além disso, o tempo de processamento da batelada será imediatamente igual a $p'_{10} = 9$ no Modelo 3.2, dispensando o uso da variável P_k do Modelo 3.1.

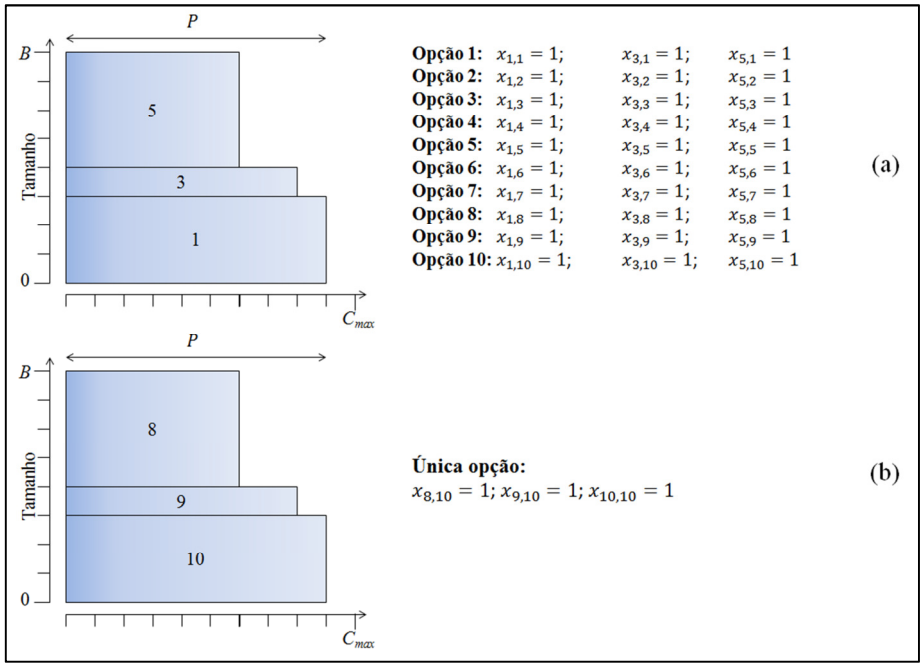


Figura 3.4 - Exemplo para diferenças de representações dos modelos para uma batelada.

Claramente o Modelo 3.2 apresenta um número de variáveis menor quando comparado com o Modelo 3.1. Além de não apresentar nenhuma variável contínua, o Modelo 3.2 apresenta a metade de variáveis binárias que o modelo anterior. Por exemplo, em uma instância com 10 tarefas, o Modelo 3.1 apresenta 110 variáveis binárias, que incluem as variáveis x_{jk} e y_k, enquanto o Modelo 3.2 apresenta 55 variáveis x_{jk}. Além disso, o número

de restrições lineares também é menor. A relação de número de variáveis dos dois modelos é apresentada na Tabela 3.3:

Tabela 3.3 - Relação de números de variáveis entre o Modelo 3.1 e Modelo 3.2.

	Modelo 3.1	Modelo 3.2
Variáveis contínuas	$\|J\|$	0
Variáveis binárias	$\|J\|^2 + \|J\|$	$(\|J\|^2 + \|J\|)/2$
Restrições lineares	$2(\|J\|^2 + \|J\|)$	$(\|J\|^2 + 3\|J\|)/2$

3.3 Resultados computacionais

Esta seção apresenta os resultados computacionais e a metodologia dos testes conduzidos para atestar a qualidade do modelo matemático proposto (Modelo 3.2), comparando-o com o modelo Modelo 3.1 e heurísticas encontrados na literatura.

Os modelos matemáticos foram testados neste trabalho, utilizando CPLEX 12.5 em um computador com processador Core 2, 2 GHz, com 2GB de memória RAM, semelhante ao utilizado nos testes publicados em Chen, Du e Huang (2011). O tempo máximo de execução para cada instância foi definido arbitrariamente como 1800 segundos. O resolvedor CPLEX foi configurado para executar em apenas um fluxo (thread), para que não fosse beneficiado com o paralelismo do processador. As heurísticas comparadas não foram implementadas neste trabalho e seus resultados foram compilados de Chen, Du e Huang (2011).

3.3.1 Instâncias utilizadas e limitantes inferiores

Este trabalho contou com o mesmo conjunto de instâncias utilizado nos testes computacionais de Chen, Du e Huang (2011), que foi gentilmente fornecido pelos seus autores e utilizado nos testes computacionais deste trabalho. A geração deste conjunto de instâncias obedece aos mesmos fatores adotados por Melouk, Damodaran e Chang (2004), mas acrescentando instâncias com 200, 300 e 500 tarefas. Na metodologia utilizada são

definidos diferentes números de tarefas, limites de tempo de processamento, limites de tamanho das tarefas e uma capacidade para a máquina. O tamanho das tarefas e o tempo de processamento são aleatoriamente escolhidos com distribuição uniforme, dentro de um respectivo intervalo. Considerando os parâmetros apresentados na Tabela 3.4, ao todo, foram geradas 42 (7*2*3) categorias. Para cada categoria foram geradas 100 instâncias diferentes, totalizando 4200 instâncias testadas.

Tabela 3.4 - Fatores utilizados na geração do conjunto de instâncias.

Número de tarefas:	10, 20, 50, 100, 200, 300 e 500
Tempo de processamento:	p_1: [1, 10] p_2: [1, 20]
Tamanho das tarefas:	s_1: [1, 10] s_2: [2, 4] s_3: [4, 8]
Capacidade da máquina:	10

Um limitante inferior (LI) proposto por Kashan, Karimi e Jolai (2006) é utilizado como referência nos testes computacionais deste trabalho. Este método requer a ordenação prévia das tarefas em modo decrescente por tempo de processamento e é uma modificação do método publicado em Uzsoy (1994). O cálculo chamado de C_{LI} permite que as tarefas sejam divididas e processadas em bateladas diferentes de maneira similar ao método FFLPT (*Firs Fit Longest Processing Time*). As tarefas do conjunto J são distribuídas nas bateladas de acordo com a ordem estipulada. Se o tamanho da tarefa for menor que o restante da capacidade da batelada disponível, a tarefa é dividida, de modo que parte é utilizada para preencher este espaço restante e a outra parte é atribuída a uma nova batelada.

A modificação realizada por Kashan, Karimi, e Jolai (2006) remove do conjunto J e insere em J' as tarefas que não podem ser agrupadas com qualquer outra tarefa em uma mesma batelada, de acordo com a equação (3.14).

$$J' = \{\forall k \in J : B - s_k < min_{i \in J}\{s_i\}\} \tag{3.14}$$

O novo limitante é calculado a partir do cálculo C_{LI} descrito utilizando o conjunto de tarefas J e posteriormente somado ao tempo de processamento das tarefas do conjunto J'. O limitante inferior modificado pode ser obtido de acordo com (3.15):

$$LI = C_{LI} + \sum_{i \in J'} p_i \tag{3.15}$$

3.3.2 Comparação entre Modelo 3.1 e Modelo 3.2

A partir da Tabela 3.5 até a Tabela 3.8 são apresentados os resultados computacionais na média para as 100 instâncias testadas. A primeira coluna apresenta a categoria das instâncias testadas, conforme mostrado na Tabela 3.4. A coluna 2 apresenta o LI representado pela equação (3.15). As colunas 3, 4, 5 e 6 mostram os resultados computacionais do Modelo 3.1, como a média das melhores soluções encontradas C_{max}, a média dos tempos totais de execução T_t(s), a média dos tempos das melhores soluções encontradas T_M(s) e a média dos GAPs fornecidos pelo CPLEX. As colunas 7, 8, 9 e 10 mostram os resultados computacionais Modelo 3.2 como a média das melhores soluções encontradas C_{max}, a média dos tempos totais de execução T_t(s), a média dos tempos das melhores soluções encontradas T_M(s) e a média dos GAPs fornecidos pelo CPLEX. Os resultados computacionais são representados pela média das 100 instâncias testadas em cada categoria.

Tabela 3.5 - Resultados comparativos entre Modelo 3.1 e Modelo 3.2 para instâncias com 10 tarefas.

		Modelo 3.1				Modelo 3.2			
Instância	LI	C_{max}	T_t(s)	T_M(s)	GAP	C_{max}	T_t(s)	T_M(s)	GAP
p_1s_1	35,33	**36,86**	0,12	0,10	0,00	**36,86**	**0,01**	**0,01**	**0,00**
p_1s_2	20,16	**20,38**	0,06	0,06	0,00	**20,38**	**0,02**	**0,02**	**0,00**
p_1s_3	43,04	**43,79**	0,18	0,12	0,00	**43,79**	**0,00**	**0,00**	**0,00**
p_2s_1	64,83	**67,62**	0,11	0,10	0,00	**67,62**	**0,01**	**0,01**	**0,00**
p_2s_2	39,78	**40,22**	0,06	0,06	0,00	**40,22**	**0,02**	**0,02**	**0,00**
p_2s_3	79,52	**81,05**	0,15	0,12	0,00	**81,05**	**0,00**	**0,00**	**0,00**

Tabela 3.6 - Resultados comparativos entre Modelo 3.1 e Modelo 3.2 para instâncias com 20 tarefas.

		Modelo 3.1				Modelo 3.2			
Instância	LI	C_{max}	T_t(s)	T_M(s)	GAP	C_{max}	T_t(s)	T_M(s)	GAP
p_1s_1	64,88	**68,07**	588,79	1,77	1,15	**68,07**	**0,03**	**0,03**	**0,00**
p_1s_2	36,96	**37,13**	1297,88	18,67	19,76	**37,13**	**0,18**	**0,18**	**0,00**
p_1s_3	81,36	**83,69**	478,03	1,99	0,93	**83,69**	**0,01**	**0,01**	**0,00**
p_2s_1	125,71	**133,09**	633,27	2,19	0,85	**133,09**	**0,03**	**0,03**	**0,00**
p_2s_2	72,02	**72,88**	1459,43	31,31	21,87	**72,88**	**0,21**	**0,19**	**0,00**
p_2s_3	155,02	**159,11**	741,51	2,48	1,56	**159,11**	**0,01**	**0,01**	**0,00**

Tabela 3.7 - Resultados comparativos entre Modelo 3.1 e Modelo 3.2 para instâncias com 50 tarefas.

		Modelo 3.1				Modelo 3.2			
Instância	LI	C_{max}	T_t(s)	T_M(s)	GAP	C_{max}	T_t(s)	T_M(s)	GAP
p_1s_1	156,21	164,43	1800,00	341,79	51,84	**164,08**	**1,03**	**0,21**	**0,00**
p_1s_2	86,87	88,37	1800,00	472,65	69,35	**87,39**	**415,40**	**3,88**	**0,23**
p_1s_3	196,76	202,09	1800,00	352,51	61,64	**202,03**	**0,04**	**0,10**	**0,00**
p_2s_1	298,7	315,43	1800,00	362,54	55,00	**314,57**	**0,61**	**0,27**	**0,00**
p_2s_2	166,66	170,22	1800,00	371,67	70,02	**168,11**	**265,89**	**17,42**	**0,08**
p_2s_3	371,85	384,40	1800,00	397,58	66,59	**384,13**	**0,04**	**0,10**	**0,00**

Tabela 3.8 - Resultados comparativos entre Modelo 3.1 e Modelo 3.2 para instâncias com 100 tarefas.

		Modelo 3.1				Modelo 3.2			
Instância	LI	C_{max}	T_t(s)	T_M(s)	GAP	C_{max}	T_t(s)	T_M(s)	GAP
p_1s_1	307,04	346,95	1800,00	283,16	89,77	**318,99**	**82,41**	**1,50**	**0,02**
p_1s_2	170,18	208,96	1800,00	292,26	88,39	**170,59**	**1630,65**	**96,89**	**1,26**
p_1s_3	388	414,51	1800,00	315,37	91,13	**396,96**	**0,10**	**0,10**	**0,00**
p_2s_1	587,84	670,27	1800,00	314,30	90,40	**610,64**	**51,01**	**9,16**	**0,01**
p_2s_2	324,62	417,73	1800,00	351,37	90,20	**326,26**	**1659,79**	**235,46**	**0,97**
p_2s_3	748,62	805,43	1800,00	270,74	91,92	**766,91**	**0,11**	**0,11**	**0,00**

Os testes comparativos entre o Modelo 3.1 e o Modelo 3.2 foram realizados em instâncias com no máximo 100 tarefas. É possível identificar nas tabelas Tabela 3.7 e Tabela 3.8 que o Modelo 3.1 atinge o tempo limite de 1800 segundos em todas as instâncias testadas, deixando notório que este tempo também será atingido se o número de tarefas das instâncias aumentar. Os resultados evidenciam que o Modelo 3.2 é superior em tempo computacional e

apresenta melhores soluções em geral. Além disso, os GAPs encontrados nos resultados do Modelo 3.1 revelam a dificuldade enfrentada pelo resolvedor para encontrar limitantes inferiores de qualidade. Esta dificuldade é amenizada com a utilização do Modelo 3.2, encontrando bons limitantes mesmo quando a otimalidade da solução não é provada.

A Tabela 3.5 revela que os dois modelos são eficientes quando o tamanho da instância é pequeno (10 tarefas). Ambos os modelo provaram a otimalidade de todas as instâncias em tempos muito reduzidos. Mesmo assim, o Modelo 3.2 é em média mais eficiente em todas as categorias de instâncias. Na Tabela 3.6, com resultados para instâncias com 20 tarefas, já é possível perceber que os tempos computacionais do Modelo 3.1 aumentam significativamente, não sendo possível provar a otimalidade de todas as instâncias testadas no tempo máximo definido, mesmo que todas as soluções ótimas sejam encontradas. Já o Modelo 3.2 prova a otimalidade de todos os casos em um tempo computacional não superior a um segundo. Conforme o número de tarefas aumenta para 50 e 100 tarefas, o desempenho dos dois modelos continua revelando a superioridade do Modelo 3.2, que apresenta sempre um tempo computacional menor e soluções melhores.

Através dos tempos computacionais de ambos os modelos, é possível identificar que instâncias do tipo "s_2" tendem a consumir um maior esforço computacional, podendo ser consideradas mais difíceis para ambos os modelos. Isto ocorre porque as tarefas geradas para estas instâncias possuem tamanhos pequenos quando comparados com a capacidade da máquina, aumentando o número de combinações possíveis de tarefas na formação das bateladas. Com isso, o espaço solução de problemas com instâncias deste tipo tende a ser maior do que nos outros tipos de instâncias, fazendo com que os resolvedores demorem em provar a otimalidade das soluções encontradas. Isto pode ser confirmado na comparação das colunas de tempo computacional total $T_t(s)$ e tempo da melhor solução inteira encontrada $T_M(s)$. Os resolvedores encontram uma solução de boa qualidade em um tempo muito inferior ao tempo em que o mesmo consome para provar as soluções.

Seguindo esta premissa, pode-se concluir que ambos os modelos apresentados tendem a tratar com maior facilidade instâncias do tipo "s_3". É observado então uma ordem de dificuldade dos tipos das instâncias de acordo com o tamanho das tarefas gerados, sendo "s_2" mais difícil que "s_3", que são mais difíceis que "s_1".

Uma segunda análise considerando os tipos de instâncias "p_1" e "p_2" revela que os tempos computacionais dos modelos tendem a crescer de acordo com o aumento dos limites em que os tempos de processamento das tarefas são gerados. Com isso, é observado que instâncias do tipo "p_2" são mais difíceis que as do tipo "p_1". Porém, é constatado para estas

instâncias que o aumento de dificuldade entre instâncias "p_1" e "p_2" é irrisório comparado ao aumento de dificuldade observado entre instâncias "s_1", "s_2" e "s_3".

3.3.3 Comparação entre o Modelo 3.2 e as heurísticas

Para que fossem realizadas comparações de forma justa e fidedigna, as condições dos testes dos modelos testados neste trabalho são similares aos publicados em Chen, Du e Huang (2011). Para tanto, foi utilizado mesmo conjunto de instâncias e um computador similar ao reportado no estudo original. Os resultados computacionais das heurísticas foram extraídos de Chen, Du e Huang (2011), para fins de comparação com os modelos testados neste trabalho.

A partir da Tabela 3.9 até a Tabela 3.15 são apresentados os resultados computacionais. A primeira coluna apresenta a categoria das instâncias, a segunda coluna apresenta o LI e as colunas três e quatro apresentam resultados da heurística BFLPT (do inglês: *Best-Fit Longest Processing Time*) proposto por Ghazvini e Dupont (1998). As colunas cinco e seis apresentam resultados do Algoritmo Genético (AG) proposto por Damodaran, Manjeshwar e Srihari (2006). As colunas sete e oito apresentam resultados do algoritmo de agregação CACB (do inglês: *Constrained Agglomerative Clustering of Batches*) proposto por Chen, Du e Huang (2011). Estes resultados são retirados de Chen, Du e Huang (2011). As colunas 9, 10, 11 e 12 mostram os resultados computacionais do Modelo 3.2, a saber: média das melhores soluções encontradas C_{max}, média dos tempos totais de execução $T_t(s)$, média dos tempos das melhores soluções encontradas $T_M(s)$, média dos GAPs fornecidos pelo CPLEX e o número de soluções ótimas provadas #O.

Tabela 3.9 - Resultados para instâncias com 10 tarefas.

		BFLPT		AG		CACB		Modelo 3.2				
Instância	LI*	C_{max}*	$T_t(s)$*	C_{max}*	$T_t(s)$*	C_{max}*	$T_t(s)$*	C_{max}	$T_t(s)$	$T_M(s)$	GAP	#O
p_1s_1	35,33	37,06	0,02	**36,86**	2,38	37,62	0,03	**36,86**	0,01	0,01	0,00	100
p_1s_2	20,16	20,60	0,05	20,39	2,03	21,58	0,02	**20,38**	0,02	0,02	0,00	100
p_1s_3	43,04	44,04	0,05	43,87	2,39	44,17	0,03	**43,79**	0,00	0,00	0,00	100
p_2s_1	64,83	67,99	0,02	67,63	2,34	68,22	0,02	**67,62**	0,01	0,01	0,00	100
p_2s_2	39,78	40,43	0,02	**40,22**	2,06	43,35	0,03	**40,22**	0,02	0,02	0,00	100
p_2s_3	79,52	81,53	0,03	**81,05**	2,33	82,10	0,02	**81,05**	0,00	0,00	0,00	100

*Resultados compilados do trabalho Chen, Du e Huang (2011).

Tabela 3.10 - Resultados para instâncias com 20 tarefas.

		BFLPT		AG		CACB		Modelo 3.2				
Instância	LI*	C_{max}*	T_t(s)*	C_{max}*	T_t(s)*	C_{max}*	T_t(s)*	C_{max}	T_t(s)	T_M(s)	GAP	#O
p_1s_1	64,88	69,17	0,02	68,55	3,83	69,73	0,05	**68,07**	0,03	0,03	0,00	100
p_1s_2	36,96	37,93	0,02	37,82	3,27	39,10	0,07	**37,13**	0,18	0,18	0,00	100
p_1s_3	81,36	84,53	0,03	83,78	4,16	84,45	0,05	**83,69**	0,01	0,01	0,00	100
p_2s_1	125,71	135,25	0,05	134,05	3,97	136,06	0,05	**133,09**	0,03	0,03	0,00	100
p_2s_2	72,02	73,91	0,03	73,53	3,23	77,19	0,06	**72,88**	0,21	0,19	0,00	100
p_2s_3	155,02	160,91	0,03	159,44	4,13	160,73	0,03	**159,11**	0,01	0,01	0,00	100

*Resultados compilados do trabalho Chen, Du e Huang (2011).

Tabela 3.11 - Resultados para instâncias com 50 tarefas.

		BFLPT		AG		CACB		Modelo 3.2				
Instância	LI*	C_{max}*	T_t(s)*	C_{max}*	T_t(s)*	C_{max}*	T_t(s)*	C_{max}	T_t(s)	T_M(s)	GAP	#O
p_1s_1	156,21	166,61	0,03	166,07	10,59	166,12	0,14	**164,08**	1,03	0,21	0,00	100
p_1s_2	86,87	89,77	0,08	89,57	8,28	91,22	0,32	**87,39**	415,40	3,88	0,23	86
p_1s_3	196,76	204,45	0,05	203,71	11,33	203,02	0,20	**202,03**	0,04	0,04	0,00	100
p_2s_1	298,70	319,72	0,05	319,27	10,59	318,83	0,25	**314,57**	0,61	0,27	0,00	100
p_2s_2	166,66	171,97	0,03	171,32	8,41	175,70	0,30	**168,11**	265,89	17,42	0,08	92
p_2s_3	371,85	388,29	0,03	387,20	11,47	386,35	0,09	**384,13**	0,04	0,04	0,00	100

*Resultados compilados do trabalho Chen, Du e Huang (2011).

Tabela 3.12 - Resultados para instâncias com 100 tarefas.

		BFLPT		AG		CACB		Modelo 3.2				
Instância	LI*	C_{max}*	T_t(s)*	C_{max}*	T_t(s)*	C_{max}*	T_t(s)*	C_{max}	T_t(s)	T_M(s)	GAP	#O
p_1s_1	307,04	324,67	0,13	324,33	27,58	321,67	0,91	**318,99**	82,41	1,50	0,02	96
p_1s_2	170,18	175,24	0,08	175,26	20,47	175,93	1,29	**170,59**	1630,65	96,89	1,26	21
p_1s_3	388,00	400,88	0,11	400,55	30,66	397,67	0,53	**396,96**	0,10	0,10	0,00	100
p_2s_1	587,84	620,88	0,06	620,04	26,97	618,34	0,86	**610,64**	51,01	9,16	0,01	98
p_2s_2	324,62	334,76	0,05	334,49	19,98	337,93	1,33	**326,26**	1659,79	235,46	0,97	17
p_2s_3	748,62	774,45	0,06	773,94	29,97	770,22	0,41	**766,91**	0,11	0,11	0,00	100

*Resultados compilados do trabalho Chen, Du e Huang (2011).

Tabela 3.13 - Resultados para instâncias com 200 tarefas.

		BFLPT		AG		CACB		Modelo 3.2				
Instância	LI*	C_{max}*	T_t(s)*	C_{max}*	T_t(s)*	C_{max}*	T_t(s)*	C_{max}	T_t(s)	T_M(s)	GAP	#O
p_1s_1	609.97	639,42	0,16	638,88	82,59	632,68	4,77	**629,43**	191,47	14,97	0,02	92
p_1s_2	332.96	343,07	0,14	342,31	57,86	341,11	14,23	**333,76**	1634,52	280,11	0,96	10
p_1s_3	773.36	792,12	0,22	792,57	93,30	787,14	2,59	**786,19**	21,40	0,43	0,00	99
p_2s_1	1162.02	1215,48	0,17	1215,89	82,42	1212,11	6,63	**1197,48**	262,14	33,48	0,02	92
p_2s_2	635.63	655,09	0,25	654,80	57,59	654,82	15,30	**638,63**	1635,79	377,39	0,95	8
p_2s_3	1479.14	1517,77	0,09	1518,07	92,86	1509,62	2,67	**1505,11**	0,58	0,47	0,00	100

*Resultados compilados do trabalho Chen, Du e Huang (2011).

Tabela 3.14 - Resultados para instâncias com 300 tarefas.

		BFLPT		AG		CACB		Modelo 3.2				
Instância	LI*	C_{max}*	T_t(s)*	C_{max}*	T_t(s)*	C_{max}*	T_t(s)*	C_{max}	T_t(s)	T_M(s)	GAP	#O
p_1s_1	907,34	943,11	0,28	942,60	163,16	934,70	19,31	**928,68**	557,85	65,79	0,06	75
p_1s_2	495,23	509,87	0,30	510,20	113,91	506,34	34,95	**497,01**	1499,80	427,03	0,58	21
p_1s_3	1157,18	1182,22	0,28	1182,28	188,00	1174,94	8,20	**1174,46**	36,50	1,48	0,00	99
p_2s_1	1746,82	1819,52	0,22	1819,00	164,50	1811,67	17,64	**1793,54**	530,28	119,89	0,03	80
p_2s_2	961,56	990,17	0,25	990,14	114,23	985,20	33,44	**966,69**	1532,10	418,01	0,83	17
p_2s_3	2216,52	2265,02	0,25	2264,67	187,48	2253,95	8,94	**2247,39**	24,44	1,66	0,00	99

*Resultados compilados do trabalho Chen, Du e Huang (2011).

Tabela 3.15 - Resultados para instâncias com 500 tarefas.

		BFLPT		AG		CACB		Modelo 3.2				
Instância	LI*	C_{max}*	T_t(s)*	C_{max}*	T_t(s)*	C_{max}*	T_t(s)*	C_{max}	T_t(s)	T_M(s)	GAP	#O
p_1s_1	1517,92	1563,12	0,64	1564,05	417,31	1551,54	101,23	**1544,36**	1171,28	247,34	0,10	45
p_1s_2	830,63	855,48	0,42	855,20	281,58	848,92	152,22	**835,00**	1637,21	223,91	0,68	17
p_1s_3	1925,68	1962,28	0,48	1962,03	504,30	1952,61	40,75	**1949,76**	64,22	6,11	0,00	97
p_2s_1	2904,48	2999,58	0,66	3000,50	423,19	2985,65	102,88	**2964,93**	1277,59	337,54	0,09	36
p_2s_2	2904,48	1631,42	0,27	1632,13	279,20	1620,94	147,73	**1599,44**	1750,57	284,53	1,12	5
p_2s_3	3665,03	3728,16	0,61	3726,52	498,23	3710,81	42,98	**3701,79**	67,96	5,94	0,00	97

*Resultados compilados do trabalho Chen, Du e Huang (2011).

A análise dos resultados mostra que as heurísticas, em geral, apresentam boas soluções em um tempo reduzido. Em instâncias pequenas, até 20 tarefas, a heurística AG apresenta um tempo computacional mais elevado, mas encontra melhores soluções. Com o aumento da quantidade de tarefas nas instâncias, a partir de 50 tarefas, a heurística CACB se destaca apresentando soluções de melhor qualidade em um tempo menor do que a heurística AG. A heurística BFLPT consome um tempo constantemente ínfimo em todos os testes, porém a qualidade de suas soluções se deteriora significativamente com o aumento da quantidade de tarefas, assim como ocorre com a heurística AG.

O CPLEX utilizando o Modelo 3.2 apresenta em geral uma superioridade em média na qualidade das soluções em todas as categorias de instâncias testadas em comparação com as heurísticas. Além disso, com o Modelo 3.2 é possível provar a otimalidade de todas as instâncias de até vinte tarefas, além de apresentar um tempo computacional compatível com o consumido pelas heurísticas, como observado nas tabelas Tabela 3.9 e Tabela 3.10. Em instâncias de cinquenta tarefas, com exceção das pertencentes ao tipo "s_2", o modelo proposto continua provando a otimalidade, apresentando um tempo computacional análogo ao apresentado pela heurística CACB, como apresentado na Tabela 3.11. Em instâncias do tipo "s_3", o Modelo 3.2 prova a otimalidade de praticamente todas as instâncias testadas, deixando apenas 9 das 1200 instâncias deste tipo com um GAP muito pequeno.

O comportamento dos tipos de instâncias testadas nesta seção respeita as observações descritas na seção 3.3.2, para instâncias de até cem tarefas. Com isso, o Modelo 3.2 apresenta tempos discrepantes para instâncias do tipo "s_2" a partir de cinquenta tarefas, e do tipo "s_1" a partir de cem tarefas. Este tempo elevado é explicado pela dificuldade que o CPLEX enfrenta para provar a otimalidade das soluções encontradas. No entanto, como heurísticas não possuem a finalidade de realizar a prova da solução ótima em suas soluções, é sensato comparar seus tempos computacionais com o tempo em que o CPLEX encontra a melhor solução em cada instância.

A Figura 3.5 apresenta gráficos para cada categoria das instâncias, comparando os tempos computacionais consumidos por cada método testado. É possível analisar o crescimento do consumo de tempo de cada método conforme o aumento do número de tarefas. Os resultados apresentados para o Modelo 3.2 são do tempo em que o CPLEX encontra a melhor solução $T_M(s)$.

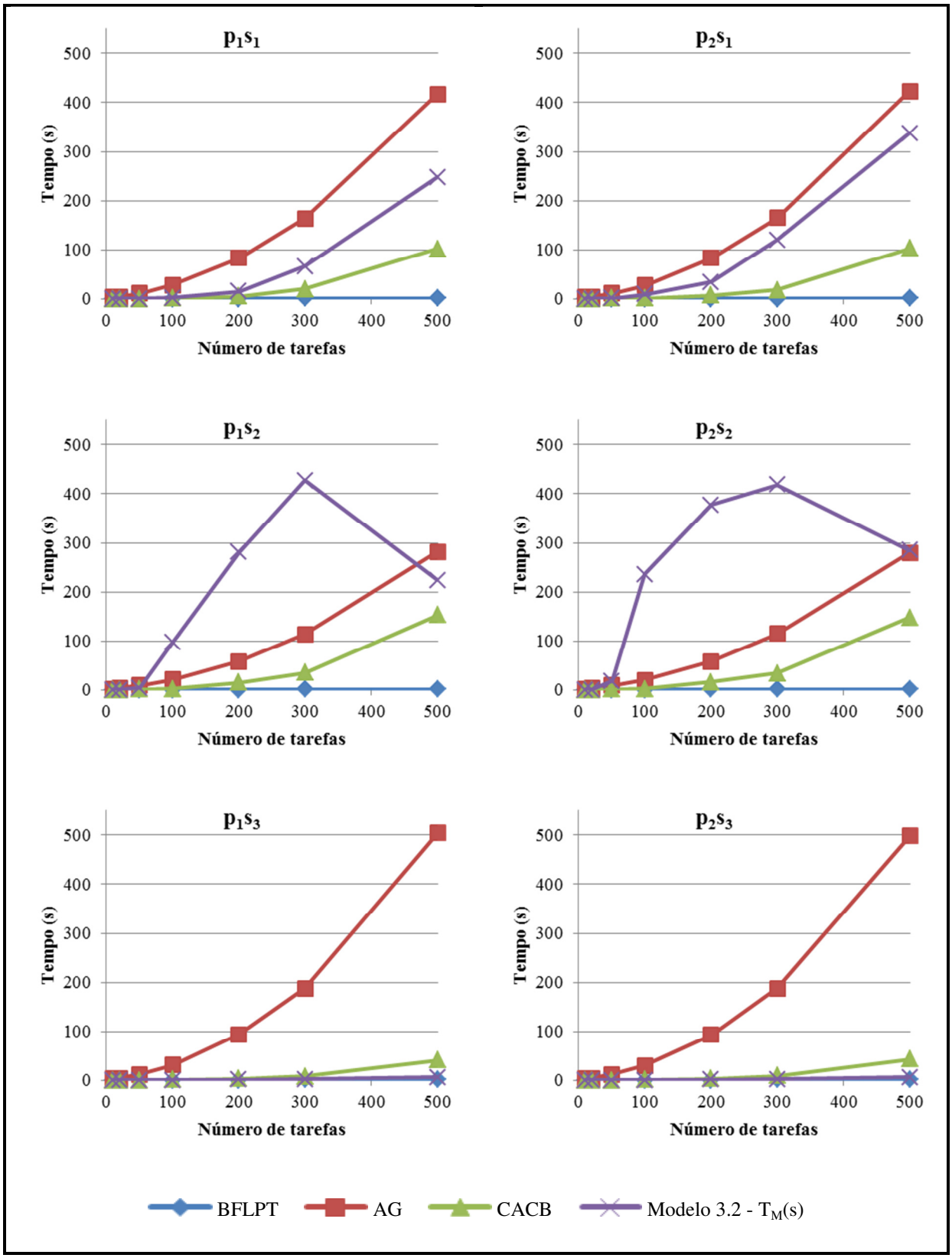


Figura 3.5 - Gráficos comparativos dos tempos computacionais consumidos em cada método pela categoria das instâncias testadas.

É possível confirmar através da Figura 3.5 que a diferença de desempenho dos métodos entre instâncias do tipo "p_1" e "p_2" é mínima. O comportamento de tempo computacional entre estes dois tipos de instância tende a ser similar em todos os métodos.

Ainda de acordo com a Figura 3.5, as heurísticas testadas apresentam uma clara diferença de tempos computacionais entre si, indicando em todos os gráficos que a heurística AG consome sempre o maior tempo computacional, seguido na ordem pela heurística CACB e por último a heurística BFLPT, que consome um tempo computacional ínfimo. Já o Modelo 3.2 encontra soluções boas em um tempo compatível com as heurísticas testadas, mas seu desempenho comparativo varia de acordo com as categorias "s_1", "s_2" e "s_3" das instâncias testadas.

Os gráficos "p_1s_1" e "p_2s_1" na Figura 3.5 indicam que, em instâncias do tipo "s_1", o tempo computacional em que o Modelo 3.2 encontra a melhor solução é constantemente menor que o da heurística AG e superior à heurística CACB. Já os gráficos "p_1s_2" e "p_2s_2" indicam que o tempo do Modelo 3.2 cresce em instâncias com até trezentas tarefas, sendo superior a todas as heurísticas testadas. No entanto, este tempo diminui com quinhentas tarefas, ficando abaixo do tempo da heurística AG em instâncias "p_1s_2" e semelhante em instâncias "p_2s_2". Já em instâncias "p_1s_2" e "p_2s_2", o tempo computacional do Modelo 3.2 é muito menor do que os apresentados pelas heurísticas AG e CACB, aparecendo nos gráficos juntamente com a linha que representa a heurística BFLPT.

A Figura 3.6 apresenta gráficos do o GAP relativo (GR) para cada categoria de instância e método testado, tendo como base as soluções fornecidas pelo Modelo 3.2. Esta distância é calculada pela fórmula a seguir:

$$GR = \frac{(Solução\ da\ heurística) - (Solução\ fornecida\ pelo\ Modelo\ 3.2)}{(Solução\ da\ heurística)} \times 100$$

Com isso, a solução do Modelo 3.2 é representada nos gráficos sempre com o com valor zero.

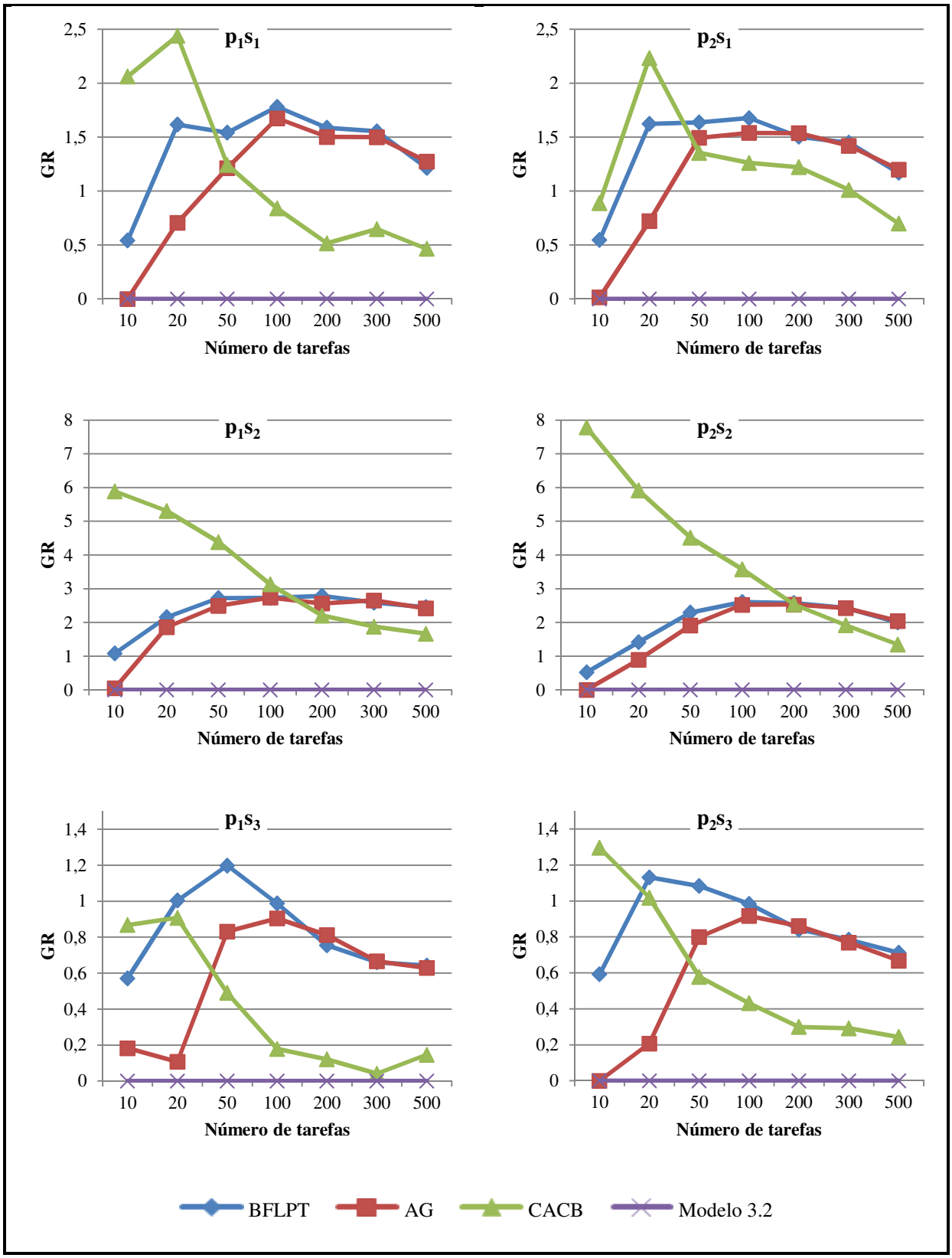


Figura 3.6 - Gráficos comparativos da qualidade das soluções fornecidas em relação ao Modelo 3.2 pelo tipo de instância testada.

Os GAPs relativos das soluções encontradas pelas heurísticas e pelo Modelo 3.2, apresentadas na Figura 3.6, aumentam de acordo com a dificuldade de cada categoria de instâncias testadas. É perceptível que em instâncias do tipo "s_3", já atestadas como sendo as mais fáceis, as diferenças de soluções não são tão grandes quanto em instâncias do tipo "s_1" e "s_2". Em instâncias do tipo "p_2s_2", a distância entre as soluções da heurística CACB pode variar entre 1,31% com quinhentas tarefas, até 7,74% com dez tarefas. No entanto, em instâncias do tipo "p_2s_3" esta diferença começa com 1,29% em instâncias com dez tarefas e termina com 0,24% em instâncias com quinhentas tarefas.

Analisando em conjunto a Figura 3.5 e a Figura 3.6, observa-se que em instâncias do tipo "s_2", o Modelo 3.2 consome um tempo computacional mais elevado em média, mas produz soluções consideravelmente melhores que as apresentadas pelas heurísticas. Já em instâncias do tipo "s_3" a diferença de qualidade das soluções geradas entre as heurísticas e o Modelo 3.2 é pequena. Considerando que o Modelo 3.2 consegue provar a otimalidade de quase todas as instâncias do tipo "s_3", evidencia-se que a qualidade de soluções produzidas pelas heurísticas para este tipo de instância é muito próxima ao ótimo. Porém, o tempo consumido pelo Modelo 3.2 nestas instâncias é muito menor do que nas heurísticas AG e CACB, comprovando um mérito do Modelo 3.2.

3.4 Conclusões do capítulo

Neste capítulo foram revisados os conceitos e observações que envolvem a minimização do *makespan* no problema de dimensionamento e programação de bateladas em máquina única ($1|s_j,B|C_{max}$), apresentando suas definições e o modelo matemático de programação mista encontrado na literatura. É proposto um modelo matemático de programação binária para a solução deste problema, que foi comparado com o modelo e heurísticas encontradas na literatura.

É possível atestar através dos resultados computacionais que o modelo proposto é superior ao modelo encontrado na literatura, encontrando soluções melhores em tempo computacional menor, além de provar a otimalidade de um número muito maior de instâncias testadas.

Comparando com as heurísticas, foi utilizado o tempo em que o modelo proposto encontrou a melhor solução, dentro de um tempo limite estipulado. Com isso, é possível

atestar que o modelo proposto consegue encontrar médias de soluções melhores em todos os tipos de instâncias testadas. Foram elaborados gráficos que fornecem a distância entre as soluções encontradas no modelo e as encontradas pelas heurísticas, revelando que esta distância aumenta quando a dificuldade das instâncias também aumenta. O tempo computacional consumido pelo modelo é superior em instâncias mais difíceis, mas é compensado pela melhora da qualidade das soluções em relação às encontradas pelas heurísticas.

4 MODELOS MATEMÁTICOS PARA O PROBLEMA $1|r_j,s_j,B|C_{max}$

Neste capítulo é proposto um modelo matemático de programação inteira mista para a minimização do *makespan* no problema de dimensionamento e programação de bateladas em máquina única, com tarefas de tamanhos e tempos de liberação não idênticos ($1|r_j,s_j,B|C_{max}$). Suas definições, exemplos, modelo matemático encontrado na literatura e observações relavantes são apresentados na seção 4.1. Um novo modelo matemático é proposto na seção 4.2. Os resultados computacionais dos modelos matemáticos são comparados e discutidos na seção 4.3. As conclusões deste capítulo são apresentadas na seção 4.4.

4.1 Descrição do problema e modelo da literatura

Formalmente, este problema pode ser definido como: dado um conjunto J de tarefas, cada elemento $j \in J$ pode ser descrito por um terno de atributos $\{r_j, p_j, s_j\}$, que representam o tempo de liberação, tempo de processamento e o tamanho da tarefa, respectivamente. Cada tarefa $j \in J$ deve ser designada a uma batelada $k \in K$, respeitando o limite de capacidade B da máquina. O tempo de processamento de cada batelada $k \in K$ é definida como $P_k = \max\{p_j, j \in k\}$. Cada tarefa $j \in J$ é disponível no tempo r_j, ou seja, o problema considera tarefas com tempos de liberação diferentes. O tempo de liberação de cada batelada $k \in K$ é definida como $R_k = \max\{r_j, j \in k\}$. As tarefas não podem ser divididas entre as bateladas, não sendo possível adicioná-las ou removê-las da máquina durante o seu processamento. O objetivo é encontrar uma configuração de bateladas $k \in K$ de modo que o tempo de processamento da máquina C_{max} (*makespan*) seja minimizado. O número de bateladas utilizadas depende do número de tarefas e da capacidade da máquina consideradas na instância, sendo que no pior caso, este número será igual à quantidade de tarefas, $|K| = |J|$.

Na Tabela 4.1 é apresentada uma instância com sete tarefas, utilizada como exemplo. A representação gráfica da solução ótima desta instância é ilustrada na Figura 4.1, na qual cada tarefa é representada por uma seta e um retângulo. A seta em cinza à esquerda representa o tempo de liberação da tarefa e o retângulo à direita representa os valores de tamanho (altura do retângulo) e tempo de processamento (largura do retângulo) da tarefa. Estes valores são destacados na ilustração através da tarefa 1. A capacidade B da máquina adotada para este

exemplo é de 14 unidades. A solução ótima deste exemplo apresenta duas bateladas, k_1 e k_2. A última tarefa é processada após 20 unidades de tempo (C_{max}). Pode-se destacar que a batelada k_1 não utiliza a sua capacidade máxima, mesmo sendo a primeira a ser processada, caracterizando um desperdício de espaço inevitável para que a programação ocupe o menor tempo de processamento possível na máquina.

Tabela 4.1 - Dados para instância exemplo do problema $1|r_j,s_j,B|C_{max}$.

j	1	2	3	4	5	6	7
r_j	8	9	3	6	10	5	1
p_j	9	5	8	5	6	3	4
s_j	3	6	1	6	4	3	2

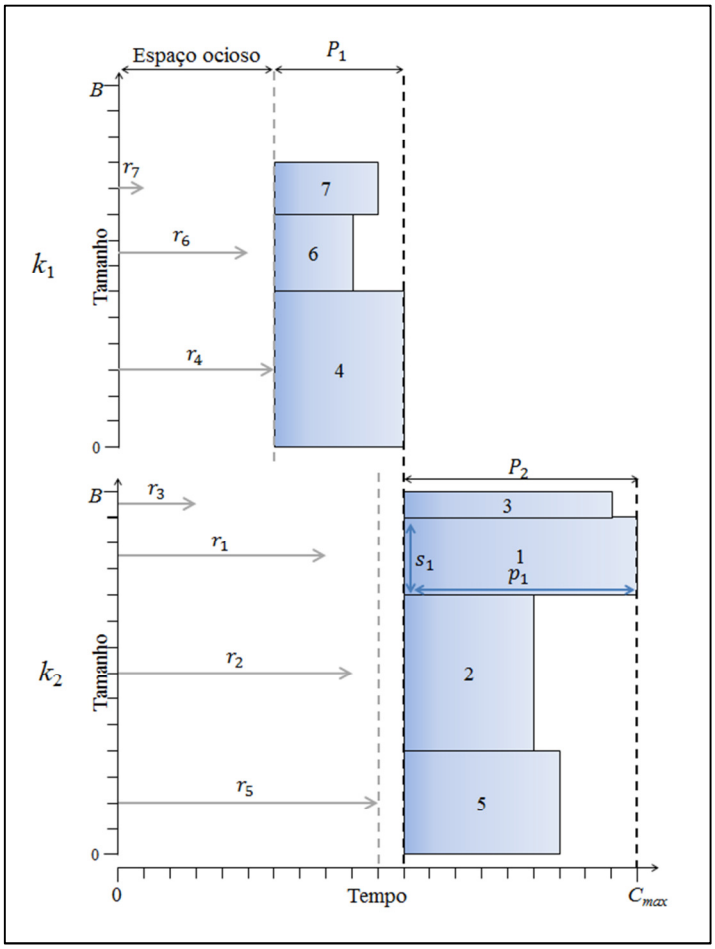


Figura 4.1 - Solução ótima a partir da instância exemplo do problema $1|r_j,s_j,B|C_{max}$.

Ao contrário do problema em que os tempos de liberação não são considerados ($1|s_j,B|C_{max}$), a ordem em que as bateladas são programadas à máquina pode proporcionar soluções diferentes para este problema. Isto acontece por causa dos diferentes tempos de liberação das tarefas, que fazem com que uma determinada permutação de bateladas possa proporcionar um tempo diferente de ociosidade na máquina.

Uma solução não ótima para a instância da Tabela 4.1 é ilustrada na Figura 4.2. As bateladas formadas nesta solução são idênticas às da solução ótima ilustrada na Figura 4.1. As soluções ilustradas diferem pela ordem em que as bateladas são sequenciadas na máquina. Na solução ótima, são programadas as bateladas na ordem k_1 e k_2, apresentando o C_{max} = 20. A Figura 4.2 sequência as bateladas na ordem k_2 e k_1, apresentando o C_{max} = 24. O fator determinante nesta diferença de soluções está no aumento do tempo ocioso, que na Figura 4.1 é de 6 unidades de tempo, e na Figura 4.2 é de 10 unidades de tempo.

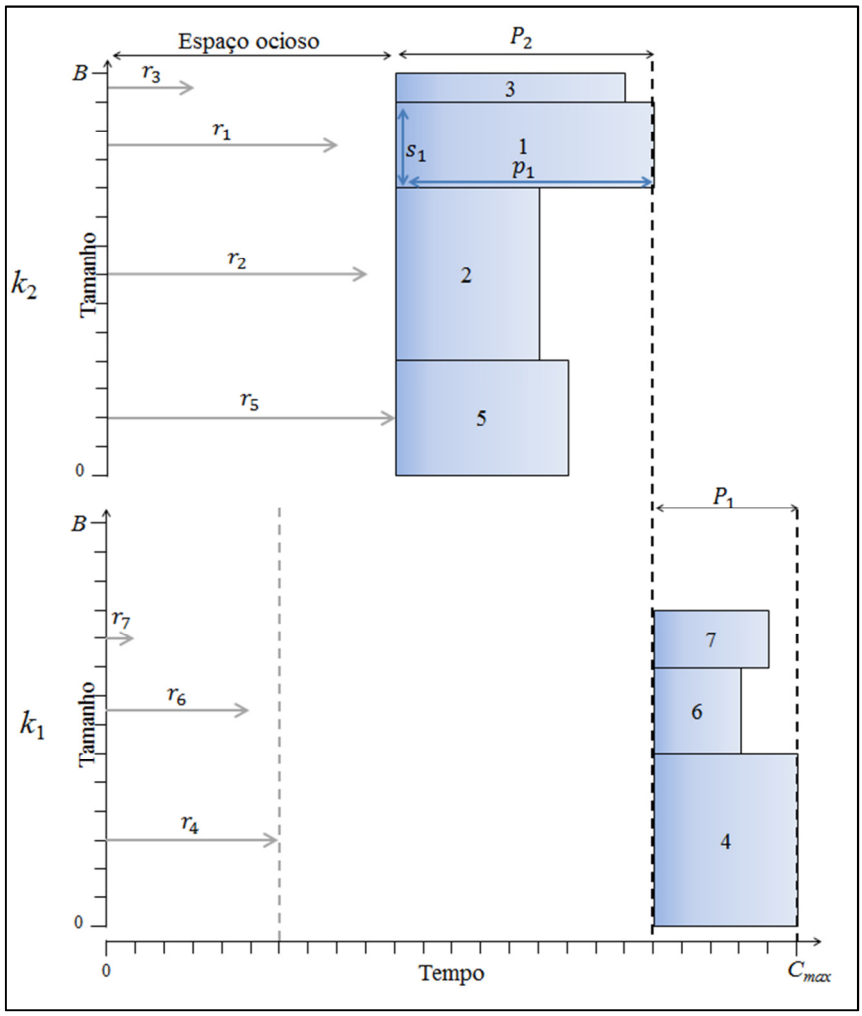


Figura 4.2 - Solução alternativa a partir da instância exemplo do problema $1|r_j,s_j,B|C_{max}$.

A complexidade deste problema é definida como NP-Difícil, como provado em Xu, Chen e Li (2012) a partir de um caso especial do problema $1|s_j,B|C_{max}$. Quando todas as tarefas possuem tempos de liberação iguais a zero $\{r_j = 0, \forall j \in J\}$, o problema $1|r_j,s_j,B|C_{max}$ se torna equivalente ao problema $1|s_j,B|C_{max}$, que é NP-Difícil (UZSOY, 1994).

Em Xu, Chen e Li (2012) é discutida a relação entre a dificuldade do problema e a variação dos tempos de liberação das tarefas dentro da instância. Se a variação de distribuição dos tempos de liberação for pequena, as tarefas estarão disponíveis em um momento parecido, fazendo com que a interferência dos tempos de liberação seja anulada, tornando-se este problema semelhante ao $1|s_j,B|C_{max}$. Se esta variação de tempos de liberação das tarefas for grande, as tarefas serão distribuídas em intervalos longos, reduzindo o tamanho do espaço solução do problema. Em uma situação extrema, a solução ótima pode determinar que cada tarefa seja designada a uma batelada diferente.

Um modelo de programação linear inteira mista é proposto em Xu, Chen e Li (2012) para a solução do problema. Os parâmetros e variáveis definidos para o Modelo 3.1 são utilizados por este modelo. Parâmetros e variáveis adicionais, em conjunto com o modelo são definidos a seguir.

Parâmetros:

r_j: tempo de liberação da tarefa j

Variáveis:

S_k : tempo em que a batelada $k \in K$ começa a ser processada

Modelo 4.1

$$Min\ S_{|K|} + P_{|K|} \tag{4.1}$$

S.a.

$$\sum_{j \in J} s_j x_{jk} \leq B \qquad \forall k \in K \tag{4.2}$$

$$\sum_{k \in K} x_{jk} = 1 \qquad \forall j \in J \tag{4.3}$$

$$P_k \geq p_j x_{jk} \qquad \forall k \in K, \forall j \in J \qquad (4.4)$$

$$S_k \geq r_j x_{jk} \qquad \forall k \in K, \forall j \in J \qquad (4.5)$$

$$S_k \geq S_{k-1} + P_{k-1} \qquad \forall k \in K: k > 1 \qquad (4.6)$$

$$P_k \geq 0 \qquad \forall k \in K, \forall j \in J \qquad (4.7)$$

$$S_k \geq 0 \qquad \forall k \in K, \forall j \in J \qquad (4.8)$$

$$x_{jk} \in \{0,1\} \qquad \forall k \in K, \forall j \in J \qquad (4.9)$$

A função objetivo (4.1) minimiza o tempo de utilização da máquina (*makespan*). Neste modelo, o tempo de utilização da máquina é definido pelo tempo de conclusão da última batelada. A restrição (4.2) determina que cada batelada não exceda a capacidade da máquina. A restrição (4.3) determina que cada tarefa será designada a somente uma batelada. A restrição (4.4) determina o tempo de processamento da batelada *k.* As restrições (4.5) e (4.6) determinam tempo em que a batelada k começa a ser processada. As restrições (4.7), (4.8) e (4.9) definem os domínios das variáveis.

4.2 Modelo proposto

Como destacado na seção anterior, a ordem com que as bateladas são sequenciadas na máquina pode gerar soluções com resultados diferentes de C_{max}. A razão desta diferença é explicada pelo aumento do tempo de ociosidade da máquina, que pode aumentar devido aos diferentes tempos de liberação das bateladas.

Para garantir que este tempo de ociosidade seja o mínimo, as bateladas podem ser sequenciadas na máquina em ordem crescente pelos seus tempos de liberação, para que aquelas que possuem menor tempo de liberação sejam processadas antes.

O espaço solução deste problema pode ser reduzido para garantir esta ordem, através de uma nova formulação proposta, baseada em restrições que garantem sequenciamento das bateladas por ordem crescente pelos seus tempos de liberação. Este novo modelo fornece melhores limitantes pela relaxação contínua do problema, não alterando a solução ótima do problema original.

O novo conjunto de restrições necessita que as tarefas sejam ordenadas pelos seus tempos de liberação de forma crescente.

Tendo o número de bateladas *m* igual ao número de tarefas *n*, pode-se definir que a batelada *k* só será utilizada se a tarefa k for atribuída a ele, ou seja, a variável x_{kk} define a utilização da batelada *k* seja igual a 1. A batelada *k* não deve conter as tarefas $j: j > k$ e as variáveis $x_{jk}: j > k$ não são consideradas na formulação. Pode ser definido explicitamente que a batelada k sempre possui a tarefa k como a de maior tempo de liberação. Desta forma uma batelada k, se utilizada, tem obrigatoriamente um tempo de liberação maior ou igual a qualquer batelada $j: j < k$.

O Modelo 4.2 proposto definido a seguir utiliza os mesmos parâmetros do Modelo 4.1:

Modelo 4.2

$$Min\ S_{|K|} + P_{|K|} \tag{4.10}$$

S.a.

$$\sum_{\substack{j \in J: \\ j \leq k}} s_j x_{jk} \leq B x_{kk} \qquad \forall k \in K \tag{4.11}$$

$$\sum_{\substack{k \in K: \\ j \leq k}} x_{jk} = 1 \qquad \forall j \in J \tag{4.12}$$

$$P_k \geq p_j x_{jk} \qquad \forall j \in J, \forall k \in K: j < k \tag{4.13}$$

$$S_k \geq r_k x_{kk} \qquad \forall k \in K \tag{4.14}$$

$$S_k \geq S_{k-1} + P_{k-1} \qquad \forall k \in K: k > 1 \tag{4.15}$$

$$x_{jk} \leq x_{kk} \qquad \forall j \in J, \forall k \in K: j < k \tag{4.16}$$

$$S_k \geq 0 \qquad \forall k \in K, \forall j \in J \tag{4.17}$$

$$x_{jk} \in \{0,1\} \qquad \forall k \in K, \forall j \in J \tag{4.18}$$

A função objetivo (4.10) minimiza o tempo de utilização da máquina (*makespan*). Neste modelo, o tempo de utilização da máquina é definido pelo tempo de conclusão da última batelada. A restrição (4.11) determina que cada batelada, se utilizada, não exceda a capacidade da máquina. A restrição (4.12) determina que cada tarefa seja designada a somente uma batelada. A restrição (4.13) determina o tempo de processamento da batelada *k.* A restrição (4.14) e (4.15) determinam o instante em que a batelada k começa a ser processada.

A restrição (4.16) é redundante com a restrição (4.11), no entanto foi inserida porque, quando utilizados em resolvedores baseados em relaxações lineares, o desempenho computacional é melhorado. As restrições (4.17) e (4.18) definem os domínios das variáveis.

O Modelo 4.2 desconsidera soluções factíveis do Modelo 4.1 em que o sequenciamento das bateladas não é dado de forma não decrescente pelos tempos de liberação, como ilustrado na Figura 4.2. Porém, o Modelo 4.2 pode representar qualquer solução ótima do Modelo 4.1, de acordo a Proposição 4.1.

■ **Proposição 4.1** - Qualquer solução ótima para o Modelo 4.1 pode ser expressa considerando as bateladas em ordem não decrescente do maior tempo de liberação das tarefas que compõe cada batelada.

Prova: No caso em que as bateladas possuem tempos de liberação iguais, a alteração da ordem de sequenciamento não modifica o valor de *makespan*. Para o caso em que os tempos de liberação são diferentes, basta considerar duas situações possíveis:
1) Em uma dada solução existe uma batelada k que tem tempo de liberação menor que uma batelada k' que a precede na sequência com que estão designadas à maquina. Se a máquina não ficar ociosa em um momento imediatamente anterior ao início do processamento da batelada k', então sempre é possível permutar as bateladas k e k', sem alterar o valor de *makespan*.
2) Em uma dada solução existe uma batelada k que tem tempo de liberação menor de uma batelada k' que a precede na sequência com que estão designadas à maquina. Se a máquina ficar ociosa em um momento imediatamente anterior ao início do processamento da batelada k', então é possível diminuir o *makespan* ao permutar as bateladas k e k', e, portanto, a solução não é ótima. □

A Proposição 4.1 é construída a partir de três casos possíveis para permutação entre duas bateladas k e k', sendo que k' é sequenciada precedendo a batelada k na máquina de processamento. Os casos descritos estão ilustrados na Figura 4.3. O caso 1 considera que as duas bateladas possuem o mesmo tempo de liberação. Neste caso, a permuta no sequenciamento das duas bateladas não afetará o C_{max}. O caso 2 considera que a batelada k possui um tempo de liberação menor que o da batelada k e a máquina de processamento não está ociosa no momento em que as bateladas são liberadas. Neste caso, a permuta no sequenciamento das duas bateladas não afetará o C_{max}. O caso 3 também considera que a

batelada k possui um tempo de liberação menor que o da batelada k e a máquina de processamento não está ociosa no momento em que as bateladas são liberadas.

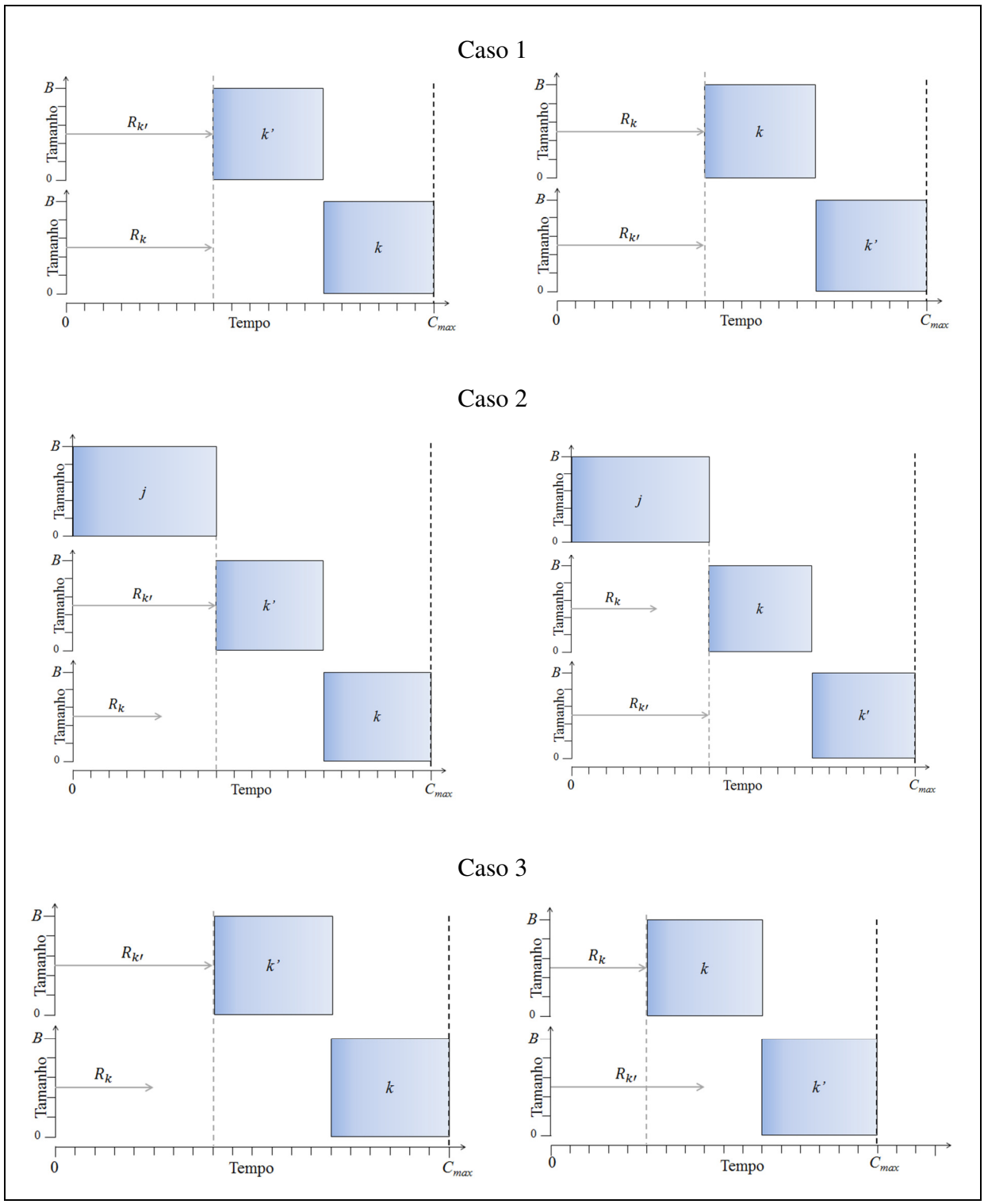


Figura 4.3 - Ilustração da troca de sequenciamento de duas bateladas para o problema com tempos de liberação não idênticos.

Neste caso, a permuta no sequenciamento das duas bateladas melhora o C_{max} e a nova solução não é ótima.

No Modelo 4.2 o número de variáveis é reduzido quando comparado com o Modelo 4.1. Por exemplo, em uma instância com 10 tarefas, o Modelo 4.1 apresenta 100 variáveis binárias x_{jk}, enquanto o Modelo 4.2 apresenta 55 variáveis x_{jk}. Além disso, o número de restrições lineares também é menor. A relação de número de variáveis e de restrições lineares dos dois modelos é apresentada na Tabela 4.2:

Tabela 4.2 - Relação de números de variáveis entre o Modelo 4.1 e Modelo 4.2.

	Modelo 4.1	Modelo 4.2
Variáveis contínuas	$2\|J\|$	$2\|J\|$
Variáveis binárias	$\|J\|^2$	$(\|J\|^2 + \|J\|)/2$
Restrições lineares	$2\|J\|^2 + 3\|J\| - 1$	$\|J\|^2 + 5\|J\| - 1$

4.3 Resultados computacionais

Esta seção apresenta os resultados computacionais e a metodologia dos testes conduzidos para atestar a qualidade do modelo matemático proposto (Modelo 4.2), comparando-o com o modelo Modelo 4.1 e a heurística encontrada na literatura.

4.3.1 Fatores gerais das instâncias utilizadas e limitantes inferiores

A geração de instâncias para os testes computacionais obedece à mesma parametrização empregada em Chou, Chang e Wang (2005), com exceção do limite de distribuição para o tempo de liberação. Esta metodologia é aplicada em Xu, Chen e Li (2012), em que são definidos diferentes números de tarefas, limites de tamanho das tarefas, um limite de tempo de processamento, um limite de tempo de tempo de liberação e uma capacidade para a máquina. Os tamanhos, tempos de processamento e de liberação das tarefas são escolhidos

aleatoriamente, com distribuição uniforme dentro de um respectivo intervalo. Diferentes instâncias são montadas conforme a Tabela 4.3.

Tabela 4.3 - Fatores gerais utilizados na geração do conjunto de instâncias.

Tempo de liberação	r_1: [0, C]
Tempo de processamento:	p_1: [8, 48]
Tamanho das tarefas:	s_1: [1, 15] s_2: [15, 35]
Capacidade da máquina:	40

O valor de C é obtido através da solução do problema sem considerar os tempos de liberação das tarefas. O autor não deixa claro quanto ao método utilizado a sua definição.

O limitante inferior (LI) utilizado nas tabelas comparativas foi proposto por Xu, Chen e Li (2012). O LI é calculado ordenando as tarefas de forma crescente pelos seus tempos de liberação, sendo possível concluir que a tarefa *n* tem o maior tempo de liberação. Considerando que esta tarefa será alocada à última batelada processada na máquina, o LI poderá ser calculado a partir de três casos, como descritos a seguir:

$$LI = \max_{\forall i \in J}\{F_1, F_2, F_3\} \tag{4.19}$$

Onde,

$$F_1 = r_n + p_n \qquad i | r_i + p_i \leq r_n \tag{4.20}$$

$$F_2 = r_i + p_i + p_n \qquad i | r_i + p_i > r_n; s_j + s_n > B \tag{4.21}$$

$$F_3 = \min\{r_n + \max\{p_i, p_n\}, r_i + p_i + p_n\} \qquad i | r_i + p_i > r_n; s_j + s_n \leq B \tag{4.22}$$

O Limitante Inferior pode ser calculado pela equação (4.19) que define o maior valor entre os três casos (F_1, F_2 e F_3) descritos nas equações (4.20), (4.21) e (4.22) respectivamente.

O caso F_1, expresso na equação (4.20), determina que se a tarefa i puder ser processada antes que a tarefa n seja liberada, o limitante inferior para este caso será $r_n + p_n$.

O caso F_2, expresso na equação (4.21), determina que se a tarefa i não puder ser processada antes que a tarefa n seja liberada e as duas não puderem ser alocadas juntas na mesma batelada, o limitante inferior para este caso será $r_i + p_i + p_n$.

O caso F_3, expresso na equação (4.22), determina que se a tarefa i não puder ser processada antes que a tarefa n seja liberada e as duas puderem ser alocadas juntas nas mesma batelada, haverá duas possibilidades: as duas poderão ser alocadas juntas, definida por $r_n + \max\{p_i, p_n\}$; e, as duas serão alocadas separadamente em duas bateladas, definida por $r_i + p_i + p_n$. O limitante inferior neste caso será o menor valor dentre estas duas condições. O LI considera apenas a relação entre a última tarefa liberada e as demais contidas no conjunto.

4.3.2 Comparação entre Modelo 4.1 e Modelo 4.2

Neste trabalho é proposto um conjunto de instâncias gerado a partir da metodologia apresentada na seção 4.3.1. O valor para limites de geração do tempo de liberação foi definido entre [0, C], onde C é o valor obtido através da heurística BFFLPT (*Batch First Fit Lowest Precessing Time*). Nesta heurística, as tarefas são ordenadas a partir de seus tempos de liberação, de forma crescente. Posteriormente, cada tarefa é alocada à primeira batelada a qual a capacidade não é excedida, de forma similar à heurística *first-fit*.

Além das duas categorias "s_1" e "s_2" descritas na Tabela 4.3, sete configurações adicionais para o conjunto de instâncias foram gerados a partir do número de tarefas. O conjunto de número de tarefas é {10, 20, 50, 100, 200, 300, 500}. Para cada categoria foram geradas 20 instâncias, totalizando 280 (2*7*20) instâncias.

Os testes foram realizados utilizando CPLEX 12.5 em um computador com processador Intel Quad-Core Xeon X3360 2.83 GHz, com 8GB de memória RAM. O tempo máximo de execução para cada instância foi definido arbitrariamente como 1800 segundos.

A Tabela 4.4 mostra os resultados computacionais em média para cada categoria de instância testada. A primeira e a segunda coluna apresentam a categoria de instância testada. Para o Modelo 4.1, as colunas 3, 4, 5 e 6 apresentam a média das melhores soluções encontradas C_{max}, a média dos tempos totais de execução T_t(s), a média dos tempos das melhores soluções encontradas T_M(s) e a média dos GAPs fornecidos pelo CPLEX, respectivamente. Para o Modelo 4.2, as colunas 7, 8, 9 e 10 apresentam a média das melhores soluções encontradas C_{max}, a média dos tempos totais de execução T_t(s), a média dos tempos das melhores soluções encontradas T_M(s) e a média dos GAPs fornecidos pelo CPLEX, respectivamente. Os resultados computacionais são representados pela média das 20 instâncias testadas em cada categoria.

Tabela 4.4 - Resultados computacionais comparativos entre Modelo 4.1 e Modelo 4.2.

Instância		Modelo 4.1				Modelo 4.2			
Tarefas	Tipo	C_{max}	T_t(s)	T_M(s)	GAP	C_{max}	T_t(s)	T_M(s)	GAP
10	s_1	**117,80**	0,16	0,09	0,00	**117,80**	**0,02**	**0,02**	**0,00**
10	s_2	**316,95**	0,57	0,20	0,00	**316,95**	**0,01**	**0,00**	**0,00**
20	s_1	**193,80**	192,34	27,78	0,03	**193,80**	**0,33**	**0,32**	**0,00**
20	s_2	**560,70**	270,13	7,97	0,44	**560,70**	**0,01**	**0,01**	**0,00**
50	s_1	396,50	1800,00	1674,20	40,64	**389,45**	**456,99**	**129,53**	**0,63**
50	s_2	1351,80	1800,00	1725,32	75,01	**1298,55**	**0,06**	**0,06**	**0,00**
100	s_1	920,05	1800,00	1744,30	94,50	**760,45**	**1744,24**	**327,94**	**5,51**
100	s_2	3368,70	1800,00	1640,57	94,90	**2578,35**	**0,39**	**0,36**	**0,00**
200	s_1	-	-	-	-	1576,75	1800,00	1761,36	15,38
200	s_2	-	-	-	-	5049,35	1,19	1,17	0,00
300	s_1	-	-	-	-	2526,05	1800,00	1388,04	21,49
300	s_2	-	-	-	-	7483,25	1,99	1,93	0,00
500	s_1	-	-	-	-	9327,80	1800,00	1568,47	57,02
500	s_2	-	-	-	-	12589,80	3,50	3,43	0,00

Os testes comparativos entre o Modelo 4.1 e o Modelo 4.2 foram realizados em instâncias com no máximo 100 tarefas. Na Tabela 4.4 é possível verificar que o Modelo 4.1, em todas as instâncias a partir de cinquenta tarefas, atinge o tempo limite de 1800 segundos. Esta observação permite que se conclua que este tempo limite também será atingido conforme o número de tarefas das instâncias aumente.

Os resultados computacionais mostram que o Modelo 4.2 é superior ao modelo Modelo 4.1, tanto em tempo de execução quanto em qualidade de soluções encontradas. A dificuldade encontrada pelo resolvedor CPLEX, em conjunto com o Modelo 4.1, é destacada pelos altos GAPS encontrados em instâncias com número de tarefas igual ou maiores a cinquenta. Em contrapartida, o Modelo 4.2 proporcionando bons limitantes mesmo quando a solução ótima não é encontrada.

Os dois modelos se mostram eficientes em instâncias com o número de tarefas reduzido. A otimalidade é provada em todas as instâncias com número de tarefas igual a dez, em tempos computacionais pequenos. Quando o número de tarefas passa a ser vinte, não é possível provar a otimalidade com o Modelo 4.1 das soluções em todas as instâncias testadas, mesmo que o resultado ótimo sempre seja encontrado. Já o Modelo 4.2 prova a otimalidade de todas as instâncias com até vinte tarefas em tempo reduzido. Conforme o número de tarefas aumenta para 50 e 100 tarefas, o desempenho dos dois modelos continua revelando a

superioridade do Modelo 4.2, que apresenta sempre um tempo computacional menor e qualidade de soluções melhor.

É possível identificar através dos resultados computacionais de ambos os modelos que instâncias do tipo "s_1" tendem a consumir um maior esforço computacional, podendo ser consideradas mais difíceis para ambos os modelos. Este comportamento também pode ser observado na seção 3.3.2 no problema em que os tempos de liberação não são considerados, em instâncias do tipo "s_2", para o respectivo caso. Esta diferença entre as instâncias pode ser explicada pelo tamanho pequeno das tarefas quando comparado com a capacidade da máquina, que elevam o número de combinações de possíveis de bateladas e, consequentemente, gera um espaço solução maior. Com isso, é possível observar uma relação de dificuldade para as instâncias testadas, em que instâncias do tipo "s_1" são mais difíceis que instâncias do tipo "s_2".

O Modelo 4.2 consegue provar a otimalidade de todas as instâncias do tipo "s_2" em instâncias de até quinhentas tarefas com tempos muito pequenos. Já em instâncias do tipo "s_1" o Modelo 4.2 deixa de provar a otimalidade de todas as instâncias quando o número de tarefas é igual a cinquenta, mas continua gerando soluções e limitantes de boa qualidade. Quando o número de tarefas aumenta, a qualidade das soluções piora e o Modelo 4.2 passa a atingir o tempo limite definido em todas as instâncias a partir de duzentas tarefas, em instâncias do tipo "s_1". A comparação entre as colunas $T_t(s)$ e $T_M(s)$ indicam que em ambos os modelos, soluções novas são geradas em praticamente todo o tempo computacional consumido.

4.3.3 Comparação entre Modelo 4.2 e as heurísticas

Este trabalho contou com o mesmo conjunto de instâncias utilizado nos testes computacionais de Xu, Chen e Li (2012), que foi gentilmente fornecido pelos seus autores e utilizado nos testes computacionais deste trabalho. Este conjunto obedece à metodologia descrita na seção 4.3.1. O conjunto de número de tarefas utilizado é {10, 20, 50, 100}. Para cada categoria descrita na Tabela 4.3, em conjunto com o conjunto de quantidade de tarefas, foram geradas 5 instâncias, totalizando 40 (2*4*5) instâncias.

Os testes do modelo matemático foram realizados em um computador com processador Pentium-4 2,99 GHz, com 3GB de memória RAM, utilizando CPLEX 12.5. O tempo máximo de execução para cada instância foi definido arbitrariamente como 1800

segundos. O resolvedor CPLEX foi configurado para executar em apenas um fluxo (*thread*), para que não fosse utilizado o paralelismo.

Para que as comparações fossem realizadas de forma justa e fidedigna, as condições publicadas em Xu, Chen e Li (2012) foram reproduzidas da maneira mais próxima possível, sendo utilizado o mesmo conjunto de instâncias e um computador semelhante. A heurística comparada não foi implementada neste trabalho e os resultados computacionais foram compilados de Chen e Li (2012).

As tabelas a seguir mostram os resultados computacionais para cada instância testada. A primeira coluna apresenta o tipo de instância testada. Os resultados para o limitante inferior proposto por Xu, Chen e Li (2012) estão apresentados na segunda coluna. As colunas 3, 4, 5 e 6 apresentam os resultados de tempo, tempo em que melhor solução foi encontrada, solução e GAP, respectivamente, do modelo proposto. As colunas 7, 8 e 9 apresentam resultados para média de soluções, média de tempo e desvio padrão, respectivamente, utilizando algoritmo de Colônia de Formigas proposto por Xu, Chen e Li (2012), testado 10 vezes. A coluna 10 apresenta o GAP relativo (GR) das soluções, que utiliza a fórmula a seguir:

$$GR = \frac{(Solução\ ACO) - (Solução\ Modelo\ Proposto)}{(Solução\ ACO)} \times 100\%$$

Tabela 4.5 - Resultados para instâncias com 10 tarefas.

		Modelo 4.2				ACO			Melhora
Instância	LI*	C_{max}	T_t(s)	T_M(s)	GAP	C_{max}*	T_t(s)*	Desv. Padr.*	Solução(%)
s_1-1	126	**126**	0,225	0,225	0,000	**126**	0,006	0,000	0,000
s_1-2	245	**254**	0,049	0,049	0,000	**254**	0,233	0,000	0,000
s_1-3	186	**186**	0,038	0,038	0,000	**186**	0,005	0,000	0,000
s_1-4	235	**235**	0,023	0,023	0,000	**235**	0,005	0,000	0,000
s_1-5	241	**266**	0,062	0,062	0,000	**266**	0,226	0,000	0,000
Média	*207,000*	***213,400***	*0,079*	*0,079*	*0,000*	***213,400***	*0,095*	*0,000*	*0,000*
s_2-1	228	**273**	0,012	0,003	0,000	**273**	0,197	0,000	0,000
s_2-2	271	**271**	0,003	0,000	0,000	**271**	0,004	0,000	0,000
s_2-3	260	**294**	0,024	0,024	0,000	**294**	0,216	0,000	0,000
s_2-4	316	**405**	0,002	0,000	0,000	**405**	0,195	0,000	0,000
s_2-5	260	**271**	0,008	0,008	0,000	**271**	0,195	0,000	0,000
Média	*267,000*	***302,800***	*0,010*	*0,007*	*0,000*	***302,800***	*0,161*	*0,000*	*0,000*

*Resultados compilados do trabalho Xu, Chen e Li (2012).

Tabela 4.6 - Resultados para instâncias com 20 tarefas.

		Modelo 4.2				ACO			Melhora
Instância	LI*	C_{max}	T_t(s)	T_M(s)	GAP	C_{max}*	T_t(s)*	Desv. Padr.*	Solução(%)
s_1-1	399	**421**	0,440	0,306	0,000	**421**	0,777	0,000	0,000
s_1-2	349	**371**	0,151	0,151	0,000	**371**	0,808	0,000	0,000
s_1-3	400	**433**	0,318	0,307	0,000	439	0,849	1,760	1,387
s_1-4	372	**385**	0,427	0,427	0,000	395	1,110	5,790	2,532
s_1-5	339	**339**	0,279	0,279	0,000	**339**	0,090	0,000	0,000
Média		***389,800***	*0,323*	*0,294*	*0,000*	*393,000*	*0,727*	*1,510*	*0,780*
s_2-1	427	**498**	0,046	0,046	0,000	**498**	0,400	0,000	0,000
s_2-2	548	**585**	0,016	0,016	0,000	**585**	0,271	0,000	0,000
s_2-3	562	**612**	0,022	0,022	0,000	**612**	0,296	0,000	0,000
s_2-4	532	**550**	0,012	0,012	0,000	**550**	0,270	0,000	0,000
s_2-5	556	**609**	0,017	0,017	0,000	**609**	0,282	0,000	0,000
Média		***570,800***	*0,023*	*0,023*	*0,000*	***570,800***	*0,304*	*0,000*	*0,000*

*Resultados compilados do trabalho Xu, Chen e Li (2012).

Tabela 4.7 - Resultados para instâncias com 50 tarefas.

		Modelo 4.2				ACO			Melhora
Instância	LI*	C_{max}	T_t(s)	T_M(s)	GAP	C_{max}*	T_t(s)*	Desv. Padr.*	Solução(%)
s_1-1	841	**868**	356,511	183,204	0,000	884	5,125	5,300	1,810
s_1-2	579	**588**	181,090	181,083	0,000	639	6,163	5,130	7,981
s_1-3	888	**946**	41,780	41,670	0,000	990	4,941	11,110	4,444
s_1-4	716	**752**	1034,800	685,313	0,000	807	6,783	12,530	6,815
s_1-5	752	**794**	25,287	25,287	0,000	824	9,130	3,690	3,641
Média		***789,600***	*327,894*	*223,311*	*0,000*	*828,800*	*6,428*	*7,552*	*4,938*
s_2-1	1213	**1439**	0,132	0,132	0,000	**1439**	2,235	0,000	0,000
s_2-2	1247	**1247**	0,342	0,342	0,000	1253	4,004	2,000	0,479
s_2-3	1297	**1371**	0,189	0,189	0,000	**1371**	2,361	0,000	0,000
s_2-4	1342	**1449**	0,062	0,062	0,000	**1449**	1,903	0,000	0,000
s_2-5	1176	**1258**	0,177	0,177	0,000	1260	2,527	2,350	0,159
Média		***1352,800***	*0,180*	*0,180*	*0,000*	*1354,400*	*2,606*	*0,870*	*0,128*

*Resultados compilados do trabalho Xu, Chen e Li (2012).

Tabela 4.8 - Resultados para instâncias com 100 tarefas.

		Modelo 4.2				ACO			Melhora
Instância	LI*	C_{max}	T_t(s)	T_M(s)	GAP	C_{max}*	T_t(s)*	Desv. Padr.*	Solução(%)
s_1-1	1521	**1593**	1800,000	1746,065	0,251	1644	20,612	5,530	3,102
s_1-2	1560	**1563**	97,694	97,670	0,000	1669	28,112	19,210	6,351
s_1-3	1444	**1593**	1800,000	1718,368	6,340	1708	26,375	16,640	6,733
s_1-4	1460	**1500**	1800,000	1783,640	0,670	1633	22,608	12,740	8,145
s_1-5	1662	**1728**	258,364	258,342	0,000	1796	21,158	9,890	3,786
Média		***1595,400***	*1151,212*	*1120,817*	*1,452*	*1690,000*	*23,773*	*12,802*	*5,623*
s_2-1	2526	**2585**	9,093	8,722	0,000	2647	13,885	4,430	2,342
s_2-2	2622	**2646**	0,187	0,187	0,000	**2646**	6,313	0,000	0,000
s_2-3	2539	**2550**	0,257	0,257	0,000	**2550**	7,640	0,000	0,000
s_2-4	2487	**2538**	0,415	0,415	0,000	2539	12,307	2,700	0,039
s_2-5	2577	**2607**	2,056	0,706	0,000	2623	13,384	3,830	0,610
Média		***2585,200***	*2,402*	*2,057*	*0,000*	*2601,000*	*10,706*	*2,192*	*0,598*

*Resultados compilados do trabalho Xu, Chen e Li (2012).

Os resultados computacionais revelam que o CPLEX utilizando o Modelo 4.2 encontra soluções melhores ou iguais a heurística ACO em todas as instâncias testadas. Em casos de até vinte tarefas, o Modelo 4.2 além de provar a otimalidade de todas as instâncias testadas, consome um tempo computacional menor que a heurística ACO. Em instâncias com cinquenta tarefas o Modelo 4.2 também prova a otimalidade em todos os casos, porém o tempo computacional do tipo "s_1" aumenta consideravelmente, tornando-se maior que o da heurística ACO . Já em instâncias do tipo "s_2", o Modelo 4.2 atinge tempos computacionais menores que a heurística ACO em todos os casos com até cem tarefas. O GAP do Modelo 4.2 revela que o CPLEX encontra soluções de qualidade, mesmo quando o CPLEX não consegue provar a otimalidade nas condições testadas.

O comportamento dos tipos de instâncias testadas nesta seção respeita as observações descritas na seção 4.3.2, que consideram instâncias do tipo "s_1" como mais difíceis. Neste caso, o tempo computacional gasto pelo Modelo 4.2 é exorbitantemente maior que o consumido pela heurística ACO. Como o CPLEX encontra soluções ao longo de todo o processo, o tempo da última solução encontrada T_M(s) tende a apresentar valor semelhante ao tempo total T_t(s).

A Figura 4.4 apresenta gráficos para cada categoria de instância, comparando os tempos computacionais consumidos por cada método testado. É possível analisar o crescimento do consumo de tempo de cada método conforme o aumento do número de

tarefas. Os resultados apresentados para o Modelo 4.2 são do tempo em que o CPLEX encontra a melhor solução $T_M(s)$.

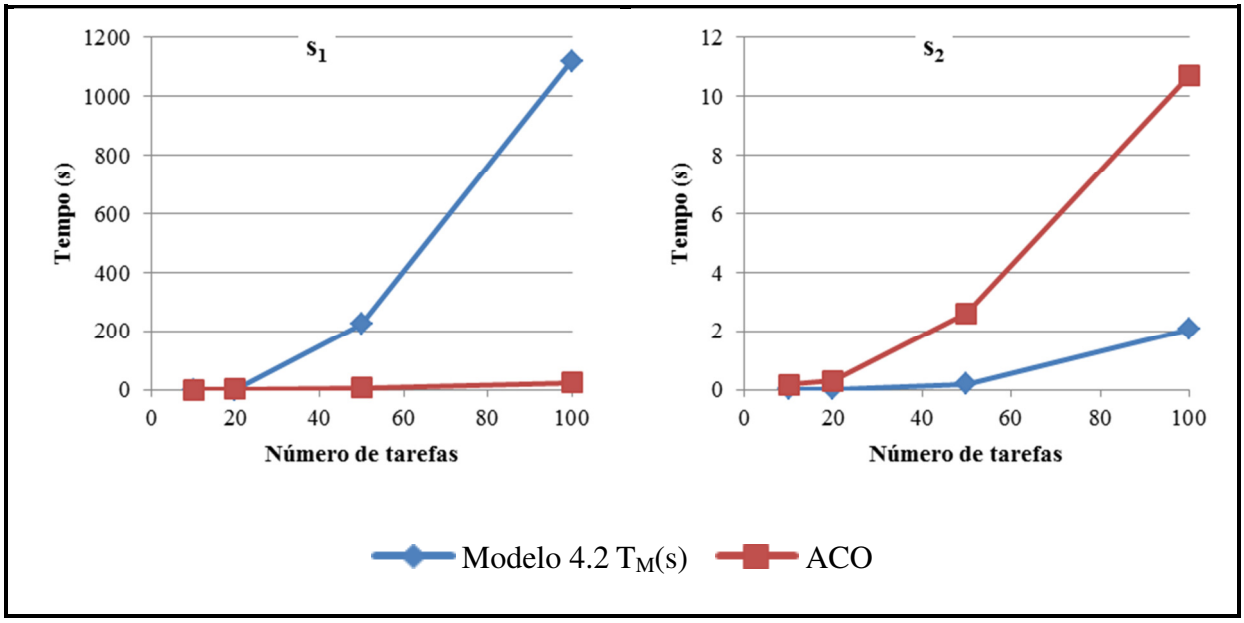


Figura 4.4 - Gráficos comparativos dos tempos computacionais médios consumidos em cada método pela categoria das instâncias testadas.

É possível analisar através da Figura 4.4 que em instâncias do tipo "s_2", o Modelo 4.2 consome um tempo computacional menor em todas as instâncias. A tendência das linhas sugere que este tempo aumente cada vez mais, conforme o número de tarefas também aumente. Já em instâncias do tipo "s_1", o tempo computacional do Modelo 4.2 é muito maior que o gasto pela heurística ACO.

A Figura 4.5 apresenta gráficos do o GAP relativo (GR) para cada categoria de instância e método testado, tendo como base as soluções fornecidas pelo Modelo 4.2. Com isso, a solução do Modelo 4.2 é representada nos gráficos sempre com o com valor zero.

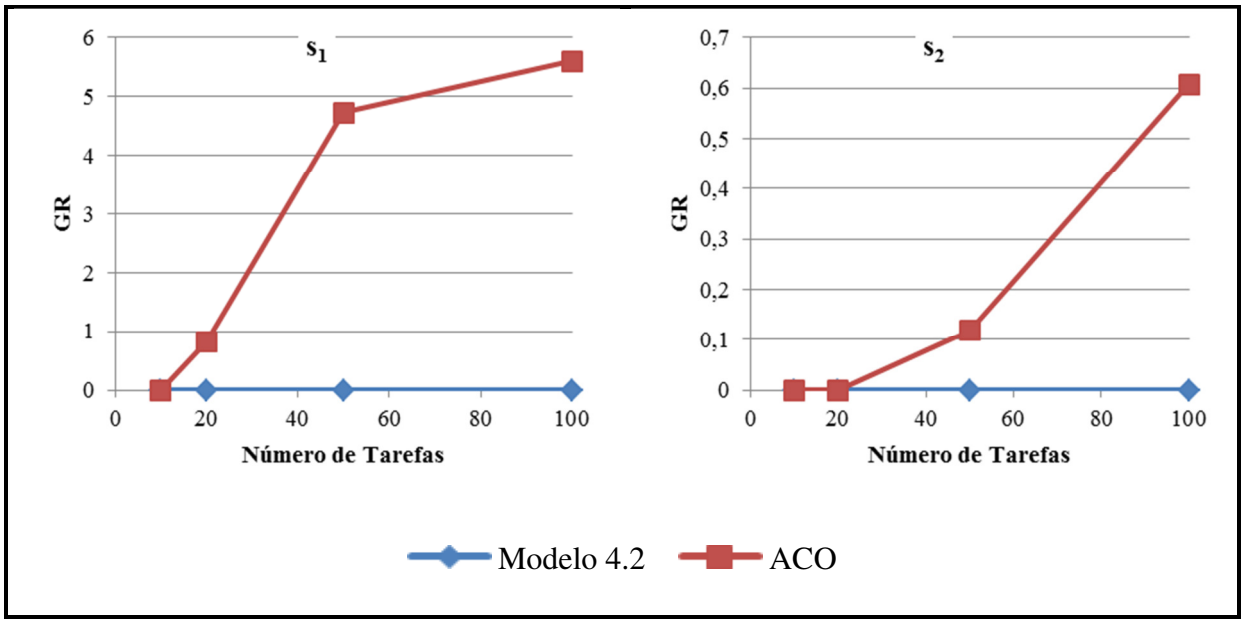


Figura 4.5 - Gráficos comparativos das soluções fornecidas em cada método pelo tipo de instância testada.

As diferenças de soluções encontradas pela heurística ACO e pelo Modelo 4.2 são muito maiores em instâncias do tipo mais difícil, "s_1". Esta diferença tende a aumentar de acordo com o crescimento do tamanho da instância e pode chegar a 6,733% em média, em instâncias com cem tarefas. Já em instâncias do tipo "s_2" esta diferença não é tão acentuada, revelando uma boa qualidade das soluções encontradas na heurística ACO nestes casos. Porém, além do Modelo 4.2 provar a otimalidade de todos estes casos, o tempo computacional revelado na Figura 4.4 indica sua superioridade em ambas as análises.

É constatado que em instâncias do tipo "s_1" o Modelo 4.2 encontra soluções melhores que a heurística ACO, porém consome um tempo muito mais alto. Neste caso, um novo teste foi elaborado para atestar a compatibilidade dos dois métodos para este tipo de instância.

Para as instâncias mais difíceis, com cinquenta e cem tarefas que utilizam a distribuição "s_1", foram recolhidas as melhores soluções encontradas pelo CPLEX com o modelo proposto, utilizando os tempos médios de execução do ACO como tempo limite. As soluções podem ser comparadas conforme a Tabela 4.9 e a Tabela 4.10.

Tabela 4.9 - Resultados para instâncias com 50 tarefas testadas com tempo limite.

			Modelo 4.2		ACO		Melhora
Instância	LI*	Tempo limite	C_{max}	GAP	C_{max}*	Desv. Padr.*	Solução(%)
s_1-1	841	5,125	916	9,830	**884**	5,300	-3,620
s_1-2	579	6,163	658	12,010	**639**	5,130	-2,973
s_1-3	888	4,941	**989**	10,210	990	11,110	0,101
s_1-4	716	6,783	**795**	10,160	807	12,530	1,487
s_1-5	752	9,13	**824**	3,640	**824**	3,690	0,000
Média			*836,400*	*9,170*	*828,800*	*7,552*	*-1,001*

*Resultados compilados do trabalho Xu, Chen e Li (2012).

Tabela 4.10 - Resultados para instâncias com 100 tarefas testadas com tempo limite.

			Modelo 4.2		ACO		Melhora
Instância	LI*	Tempo limite	C_{max}	GAP	C_{max}*	Desv. Padr.*	Solução(%)
s_1-1	1521	20,612	1689	6,470	**1644**	5,530	-2,737
s_1-2	1560	28,112	**1590**	3,140	1669	19,210	4,733
s_1-3	1444	26,375	**1703**	15,610	1708	16,640	0,293
s_1-4	1460	22,608	1639	9,090	**1633**	12,740	-0,367
s_1-5	1662	21,158	**1778**	2,810	1796	9,890	1,002
Média			*1679,800*	*7,424*	*1690,000*	*12,802*	*0,585*

*Resultados compilados do trabalho Xu, Chen e Li (2012).

Os resultados mostrados na Tabela 4.9 e na Tabela 4.10 indicam que em instâncias do tipo "s_1", o Modelo 4.2 é compatível com a heurística ACO ao longo de sua execução. Isso significa que o Modelo 4.2 encontra em média soluções tão boas quanto a heurística ACO dentro do tempo limite estipulado, porém o Modelo 4.2 possui a capacidade de seguir encontrando soluções melhores se houver a mais disponibilidade de tempo. A Tabela 4.9 mostra que a qualidade média da solução do Modelo 4.2 é um pouco pior quando comparado com a heurística ACO em instâncias com cinquenta tarefas, porém encontra soluções melhores ou iguais na maioria dos casos. Já a Tabela 4.10 indica que para instâncias com cem tarefas o Modelo 4.2 encontra em média soluções melhores que a heurística ACO, também encontrando soluções melhores ou iguais na maioria dos casos.

4.4 Conclusões do capítulo

Neste capítulo foram revisados os conceitos e observações que envolvem a minimização do *makespan* no problema de dimensionamento e programação de bateladas em máquina única com tarefas de tamanhos e tempos de liberação não idênticos ($1|r_j,s_j,B|C_{max}$), apresentando suas definições e o modelo matemático de programação mista encontrado na literatura. É proposto um modelo matemático de programação inteira mista para a resolução deste problema, que foi comparado com o modelo e a heurística ACO encontrada na literatura.

Os resultados comprovam que o modelo proposto é superior ao modelo encontrado na literatura, tanto em tempo computacional quanto em qualidade de soluções encontradas, além de provar a otimalidade de um número muito maior de instâncias e conseguir melhores limitantes para soluções quando isso não ocorre.

O modelo proposto foi comparado com uma abordagem proposta por Xu, Chen e Li (2012), que utiliza a heurística ACO para resolver o problema. O modelo proposto encontrou soluções melhores que a heurística ACO dentro de um limite de tempo estipulado. Além disso, em instâncias mais fáceis (da categoria "s_2") o modelo proposto consegue provar a otimalidade das soluções em tempo inferior ao consumido pela heurística. Já em instâncias mais difíceis (da categoria "s_1"), o modelo proposto encontra soluções compatíveis às soluções encontradas pela heurística ACO com o mesmo tempo computacional, porém o modelo proposto consegue melhorar a qualidade da solução encontrada quando disposto de mais tempo.

5 MODELOS MATEMÁTICOS PARA O PROBLEMA $P_m|s_j,B|C_{max}$

Neste capítulo é proposto um modelo matemático de programação inteira mista para a minimização do *makespan* no problema de dimensionamento e programação de bateladas em máquinas paralelas idênticas, com tarefas de tamanhos não idênticos ($P_m|s_j,B|C_{max}$). Definições, exemplos, modelo matemático encontrado na literatura e observações relevantes são apresentados na seção 5.1. Um novo modelo matemático é proposto na seção 5.2. Os resultados computacionais dos modelos matemáticos são comparados e discutidos na seção 5.3. As conclusões deste capítulo são apresentadas na seção 5.4.

5.1 Descrição do problema e modelo da literatura

Formalmente, este problema pode ser definido como: dado um conjunto J de tarefas, cada elemento $j \in J$ pode ser definido por uma dupla de atributos $\{p_j, s_j\}$, que representam o tempo de processamento e o tamanho da tarefa, respectivamente. Cada tarefa $j \in J$ deve ser designada a uma batelada $k \in K$, respeitando o limite de capacidade B das máquinas de processamento. Cada batelada $k \in K$ utilizada deve ser designada a uma máquina de processamento $m \in M$. O tempo de processamento de cada batelada $k \in K$ é definida como $P_k = \max\{p_j, j \in k\}$. Todas as tarefas são disponíveis no tempo zero, ou seja, o problema não considera tarefas com tempos de liberação diferentes, $\{r_j = 0, j \in J\}$. As tarefas não podem ser divididas entre as bateladas, não sendo possível adicioná-las ou removê-las da máquina durante o seu processamento. O tempo total de processamento C_{max} (*makespan*) será igual ao tempo da última máquina a terminar o processamento. O objetivo é encontrar uma configuração de bateladas $k \in K$ e uma programação destas bateladas às máquinas $m \in M$, de modo que o C_{max} seja minimizado. O número de bateladas utilizadas depende do número de tarefas e da capacidade da máquina consideradas na instância, sendo que no pior caso, este número será igual à quantidade de tarefas, $|K| = |J|$.

Um exemplo pode ser montado a partir de uma pequena instância de dez tarefas, apresentada na Tabela 5.1. A representação gráfica da solução ótima deste exemplo é ilustrada na Figura 5.1, na qual cada tarefa é representada por um retângulo, cujas dimensões de altura e largura representam os valores de tamanho e tempo de processamento da tarefa,

respectivamente. Estas dimensões são destacadas na Figura 5.1 através da tarefa 1. Neste exemplo são utilizadas três máquinas paralelas (m_1, m_2 e m_3) com capacidade B de 8 unidades. A solução ótima deste exemplo apresenta cinco bateladas utilizadas e o C_{max} é de 9 unidades de tempo.

Tabela 5.1 - Dados para instância exemplo do problema $P_m|s_j,B|C_{max}$.

j	1	2	3	4	5	6	7	8	9	10
p_j	9	5	8	2	6	3	4	1	2	5
s_j	3	6	1	2	4	3	2	5	3	6

Neste problema é observado que a disponibilidade de novas máquinas paralelas não necessariamente se reflete na melhora da solução ótima. Esta situação ocorre quando o C_{max} possui o valor igual ao da tarefa que contém o maior tempo de processamento. Observando a Figura 5.1 identifica-se que a máquina "m_3" apresenta tempo de ociosidade e o C_{max} é definido a partir das bateladas atribuídas às máquinas "m_1" e "m_2". O tempo de execução da máquina "m_1" é definido pela tarefa 1, que possui o maior tempo de processamento entre todas as tarefas. A tarefa 1 não pode ser dividida, portanto mesmo que o número de máquinas paralelas aumente o C_{max} do problema não pode ser melhorado.

Com isso, pode-se dizer que o menor valor para C_{max} no problema $P_m|s_j,B|C_{max}$ quando o número de máquinas paralelas aumenta tende a ser $\max_{\forall i \in J}\{p_j\}$.

Assim como o problema $P_m|s_j,B|C_{max}$, a complexidade deste problema também é considerada fortemente NP-Difícil, como reportado em Chang, Damodaran e Melouk(2004), a partir de um caso especial do problema *bin packing*.

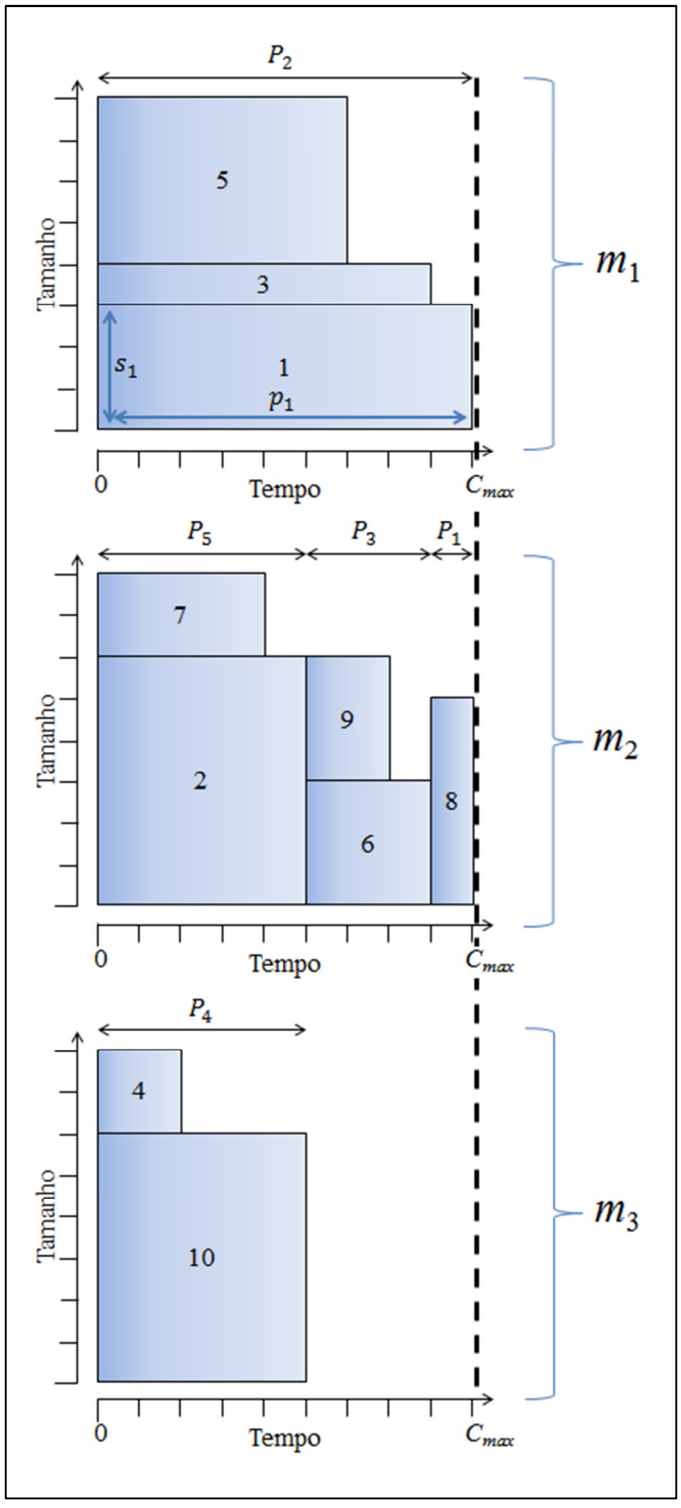


Figura 5.1 - Ilustração da instância exemplo do problema $P_m|s_j,B|C_{max}$.

Em Chang, Damodaran e Melouk (2004) é proposto um modelo de programação linear inteira mista para a resolução do problema. O modelo é definido a seguir:

Conjuntos:

J : conjunto de tarefas, $j \in J$

K: conjunto de bateladas, $k \in K$

M: conjunto de máquinas paralelas, $m \in M$

Parâmetros:

B : capacidade da máquina

p_j: tempo de processamento da tarefa j

s_j: tamanho da tarefa j

Variáveis:

$x_{jkm} = \begin{cases} 1 \text{ se a tarefa } j \in J \text{ for atribuída à batelada } k \in K \text{ na máquina } m \in M \\ 0 \text{ caso contrário} \end{cases}$

P_{km} : tempo de processamento da batelada $k \in K$ pela máquina $m \in M$

C_{max} : tempo de utilização da última máquina a terminar o processamento (*makespan*)

Modelo 5.1

$$Min\ C_{max} \tag{5.1}$$

S.a.

$$\sum_{j \in J} \sum_{m \in M} s_j x_{jkm} \leq B \qquad \forall k \in K \tag{5.2}$$

$$\sum_{k \in K} \sum_{m \in M} x_{jkm} = 1 \qquad \forall j \in J \tag{5.3}$$

$$P_{km} \geq p_j x_{jkm} \qquad \forall k \in K, \forall j \in J, \forall m \in M \tag{5.4}$$

$$C_{max} \geq \sum_{k \in K} P_{km} \qquad \forall m \in M \tag{5.5}$$

$$x_{jkm} \in \{0,1\} \qquad \forall k \in K, \forall j \in J \tag{5.6}$$

$$P_k \geq 0 \qquad \forall k \in K, \forall j \in J \tag{5.7}$$

$$C_{max} \geq 0 \tag{5.8}$$

A função objetivo (5.1) minimiza o tempo de utilização da última máquina a terminar o processamento (*makespan*). A restrição (5.2) determina que cada batelada não exceda a capacidade da máquina. A restrição (5.3) determina que cada tarefa seja designada somente a uma batelada e a uma única máquina. A restrição (5.4) determina o tempo de processamento de cada batelada. A restrição (5.5) determina o tempo de utilização da última máquina a terminar o processamento (*makespan*). As restrições (5.6),(5.7) e (5.8) determinam o domínio das variáveis.

Assim como o problema $1|s_j,B|C_{max}$, este problema pode ser considerado altamente simétrico em relação a ordem de programação das bateladas em cada uma das máquinas paralelas. Isto ocorre porque uma mesma solução pode ser representada de diferentes formas, apenas trocando a ordem de sequenciamento das bateladas, assim como o caso explorado na seção 3.1.

Além disso, o Modelo 5.1 quando comparado ao Modelo 3.1 apresenta uma nova dimensão às variáveis de decisão que representam a formação das bateladas. Neste caso, a variável x_{jkm} é responsável não só pelo dimensionamento das bateladas, mas também pelo sequenciamento à máquina de processamento. Esta variável possui uma nova dimensão $|M|$, que acentua ainda mais a característica de apresentar soluções equivalentes.

5.2 Modelo proposto

Um novo conjunto de restrições que visa diminuir o espaço solução deste problema foi utilizado de forma semelhante ao aplicado no Modelo 3.2, fornecendo melhores limitantes pela relaxação contínua do problema, não alterando a solução ótima do problema original.

O novo conjunto de restrições necessita que as tarefas sejam ordenadas pelos seus tempos de processamento de forma não decrescente.

A formulação apresentada para o problema $P_m|s_j,B|C_{max}$ consiste em dois tipos de decisão: o dimensionamento das bateladas e a programação destas nas máquinas de processamento. O Modelo 5.2 considera estas decisões utilizando dois conjuntos de variáveis diferentes. O dimensionamento das bateladas é atribuído às variáveis x_{jk} e a programação destas bateladas as máquinas de processamento é atribuída às variáveis y_{km}.

Tendo o número de bateladas m igual ao número de tarefas n, pode-se definir que a batelada k só será utilizada se a tarefa k for atribuída a ele, ou seja, a variável x_{kk} define a

utilização da batelada k se igual a 1. A variável y_{km} define a designação da batelada k a máquina m, se $x_{kk} = 1$. Com isso, as variáveis que definem a batelada k uma nova dimensão, como ocorre no Modelo 5.1. A batelada k não deve conter as tarefas $j: j > k$ e as variáveis $x_{jk}: j > k$ não são consideração na formulação. A partir da ordenação das tarefas de forma crescente de acordo com os tempos de processamento, pode ser definido explicitamente que a batelada k sempre tem a tarefa k como a de maior tempo de processamento.

A Tabela 3.4 ilustra para cada modelo a possibilidade de configuração de bateladas, se utilizadas. Nota-se que para o Modelo 5.1 (a) é utilizada a matriz completa das variáveis x_{jk} replicada para cada máquina de processamento. No Modelo 5.2 (b) é utilizada a matriz triangular superior somente uma vez, destacando a diagonal principal como variáveis obrigatórias se a batelada for utilizada.

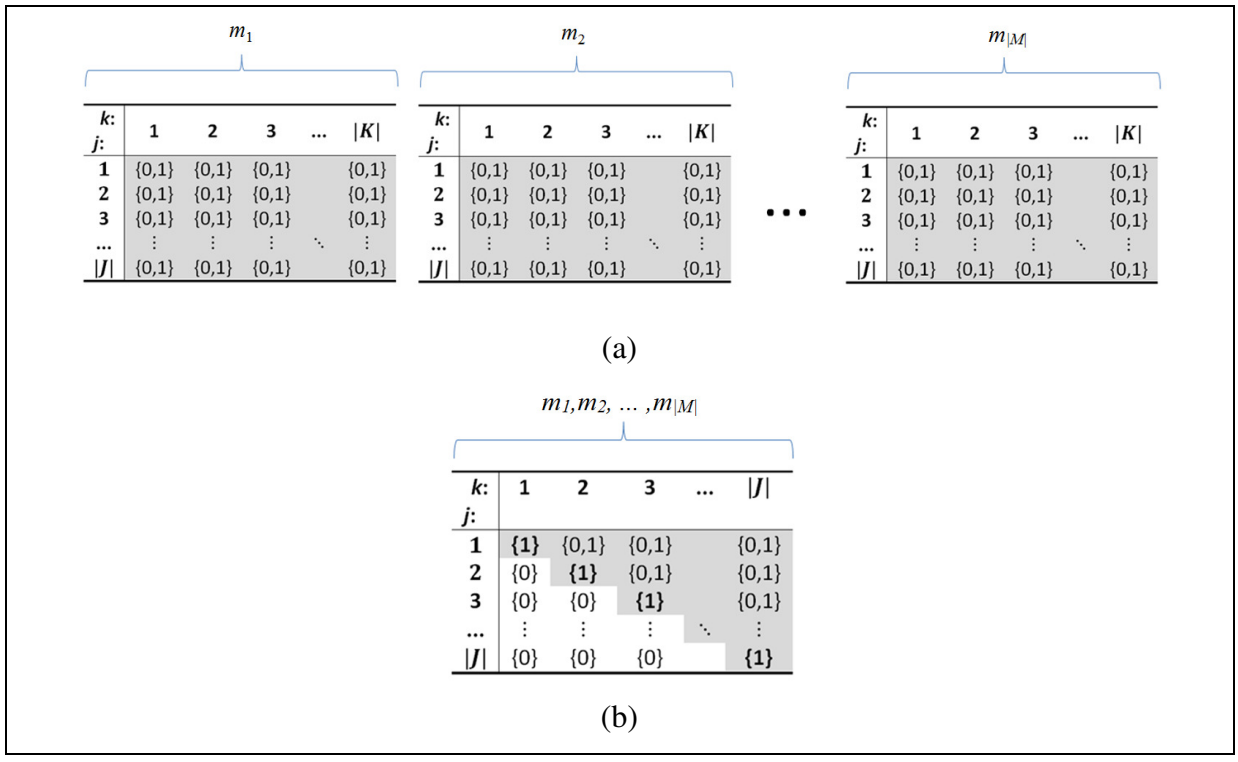


Figura 5.2 - Ilustração da formação de lotes.

O Modelo 5.2 definido a seguir utiliza os mesmos parâmetros do Modelo 5.1:

Variáveis:

$$x_{jk} = \begin{cases} 1 \text{ se a tarefa } j \in J \text{ for atribuída à batelada } k \in K \\ 0 \text{ caso contrário} \end{cases}$$

$y_{km} = \begin{cases} 1 & \text{se a batelada } k \in K \text{ for processada na máquina } m \in M \\ 0 & \text{caso contrário} \end{cases}$

C_m : tempo de utilização da máquina $m \in M$

C_{max} : tempo de utilização da última máquina a terminar o processamento (*makespan*)

Modelo 5.2

$$Min\ C_{max} \tag{5.9}$$

S.a.

$$\sum_{\substack{j \in J: \\ j \leq k}} s_j x_{jk} \leq B x_{kk} \qquad \forall k \in K \tag{5.10}$$

$$\sum_{\substack{k \in K: \\ j \leq k}} x_{jk} = 1 \qquad \forall j \in J \tag{5.11}$$

$$x_{jk} \leq x_{kk} \qquad \forall j \in J, \forall k \in K: j < k \tag{5.12}$$

$$x_{kk} \leq \sum_{m \in M} y_{km} \qquad \forall k \in K \tag{5.13}$$

$$C_m \geq \sum_{k \in K} p_k y_{km} \qquad \forall m \in M \tag{5.14}$$

$$C_{max} \geq C_m \qquad \forall m \in M \tag{5.15}$$

$$x_{jk} \in \{0,1\} \qquad \forall j \in J, \forall k \in K: j \leq k \tag{5.16}$$

$$C_m \geq 0 \qquad \forall m \in M \tag{5.17}$$

$$C_{max} \geq 0 \tag{5.18}$$

A função objetivo (5.9) minimiza o tempo de utilização da última máquina a terminar o processamento (*makespan*). A restrição (5.10) determina que cada batelada não exceda a capacidade da máquina. A restrição (5.11) determina que cada tarefa seja designada somente a uma única batelada. A restrição (5.12) determina que cada tarefa só é designada a uma batelada com índice maior ou igual ao índice da própria tarefa. A restrição (5.13) determina que cada batelada seja designada a uma máquina. A restrição (5.14) define o tempo de utilização de cada máquina. A restrição (5.15) determina o tempo de utilização da última

máquina a terminar o processamento (*makespan*). As restrições (5.16), (5.17) e (5.18) determinam o domínio das variáveis.

5.3 Resultados computacionais

O conjunto de instâncias utilizado nos testes computacionais do problema $(1|s_j,B|C_{max})$, fornecido por Chen, Du e Huang (2011), já especificado na seção 3.3.1. Nos testes foram utilizadas somente instâncias com até cem tarefas. Além disso, foram adicionadas três novas categorias correspondentes aos números de máquinas paralelas, sendo elas 2, 4 e 8 máquinas.

A geração deste conjunto obedece aos mesmos fatores adotados por Melouk, Damodaran e Chang (2004) para problemas com máquina única e Chang, Damodaran, Melouk (2004) para problemas com máquinas paralelas, com exceção do conjunto de números de tarefas utilizado.

Na metodologia utilizada, diferentes instâncias foram montadas conforme a Tabela 5.2. Com isso, foram geradas 24 (4*2*3) categorias para o conjunto. Para cada categoria, foram geradas 100 instâncias diferentes, totalizando 2400 instâncias. Cada instância é testada em três configurações de máquinas paralelas diferentes, totalizando 9600 testes realizados.

Tabela 5.2 - Fatores utilizados na geração do conjunto de instâncias.

Número de tarefas:	10, 20, 50, 100
Número de máquinas paralelas:	2, 4, 8
Tempo de processamento:	p_1: [1, 10] p_2: [1, 20]
Tamanho das tarefas:	s_1: [1, 10] s_2: [2, 4] s_3: [4, 8]
Capacidade da máquina:	10

Os testes foram realizados utilizando CPLEX 12.5 em um computador com processador Intel Quad-Core Xeon X3360 2.83 GHz, com 8GB de memória RAM. O tempo máximo de execução para cada instância foi definido arbitrariamente como 1800 segundos.

O limitante inferior utilizado como referência nos testes computacionais deste trabalho é calculado a partir do limitante inferior modificado (C_{LIM}) calculado para o problema em que a máquina única é considerado $1|s_j,B|C_{max}$, abordado na seção 3.3.1. O novo limitante calculado para o problema $P_m|s_j,B|C_{max}$ é calculado pela fórmula abaixo:

$$C_{LI} = \frac{C_{LIM}}{|M|} \tag{5.19}$$

As tabelas a seguir mostram os resultados computacionais. A primeira coluna apresenta a categoria de instância testada. A segunda coluna apresenta o limitante inferior (LI) calculado. As colunas 3, 4, 5 e 6 apresentam a média das melhores soluções encontradas C_{max}, a média dos tempos totais de execução T_t(s), a média dos tempos das melhores soluções encontradas T_M(s) e a média dos GAPs fornecidos pelo CPLEX, respectivamente, do Modelo 5.1. As colunas 7, 8, 9 e 10 apresentam a média das melhores soluções encontradas C_{max}, a média dos tempos totais de execução T_t(s), a média dos tempos das melhores soluções encontradas T_M(s) e a média dos GAPs fornecidos pelo CPLEX, respectivamente, do Modelo 5.2. Os resultados são representados pela média das 100 instâncias testadas em cada categoria.

Tabela 5.3 - Resultados comparativos entre o Modelo 5.1 e o Modelo 5.2 para instâncias com 2 máquinas paralelas.

		Modelo 5.1				Modelo 5.2			
Instância	LI	C_{max}	T_t(s)	T_M(s)	GAP	C_{max}	T_t(s)	T_M(s)	GAP
10 Tarefas									
p_1s_1	17,67	**18,76**	0,13	0,12	**0,00**	**18,76**	**0,01**	**0,01**	**0,00**
p_1s_2	10,08	**11,03**	0,05	0,05	**0,00**	**11,03**	**0,02**	**0,02**	**0,00**
p_1s_3	21,52	**22,13**	0,19	0,15	**0,00**	**22,13**	**0,01**	**0,01**	**0,00**
p_2s_1	32,42	**34,50**	0,12	0,11	**0,00**	**34,50**	**0,01**	**0,01**	**0,00**
p_2s_2	19,89	**21,71**	0,05	0,05	**0,00**	**21,71**	**0,02**	**0,02**	**0,00**
p_2s_3	39,76	**40,87**	0,17	0,14	**0,00**	**40,87**	**0,01**	**0,01**	**0,00**
20 Tarefas									
p_1s_1	32,44	**34,27**	1308,41	45,82	5,54	**34,27**	**0,03**	**0,03**	**0,00**
p_1s_2	18,48	**18,83**	884,08	21,46	8,16	**18,83**	**0,11**	**0,11**	**0,00**
p_1s_3	40,68	**42,13**	1412,74	27,48	6,27	**42,13**	**0,02**	**0,02**	**0,00**
p_2s_1	62,86	**66,79**	1287,70	27,99	4,35	**66,79**	**0,03**	**0,03**	**0,00**
p_2s_2	36,01	**36,87**	651,70	24,56	7,05	**36,87**	**0,15**	**0,15**	**0,00**
p_2s_3	77,51	**79,82**	1395,83	46,57	5,60	**79,82**	**0,02**	**0,02**	**0,00**

Tabela 5.4 - Resultados comparativos entre o Modelo 5.1 e o Modelo 5.2 para instâncias com 4 máquinas paralelas.

		Modelo 5.1				Modelo 5.2			
Instância	LI	C_{max}	T_t(s)	T_M(s)	GAP	C_{max}	T_t(s)	T_M(s)	GAP
10 Tarefas									
p_1s_1	8,83	**10,87**	0,16	0,14	**0,00**	**10,87**	**0,02**	**0,02**	**0,00**
p_1s_2	5,04	**9,49**	0,10	0,09	**0,00**	**9,49**	**0,01**	**0,01**	**0,00**
p_1s_3	10,76	**12,18**	0,25	0,22	**0,00**	**12,18**	**0,02**	**0,02**	**0,00**
p_2s_1	16,21	**20,26**	0,16	0,12	**0,00**	**20,26**	**0,02**	**0,02**	**0,00**
p_2s_2	9,95	**18,68**	0,11	0,10	**0,00**	**18,68**	**0,01**	**0,01**	**0,00**
p_2s_3	19,88	**22,67**	0,23	0,21	**0,00**	**22,67**	**0,02**	**0,02**	**0,00**
20 Tarefas									
p_1s_1	16,22	**17,47**	1316,19	87,17	8,11	**17,47**	**0,05**	**0,05**	**0,00**
p_1s_2	9,24	**10,43**	56,14	1,59	0,49	**10,43**	**0,32**	**0,32**	**0,00**
p_1s_3	20,34	**21,29**	1629,93	95,67	11,78	**21,29**	**0,03**	**0,03**	**0,00**
p_2s_1	31,43	**33,95**	1122,49	73,44	5,29	**33,95**	**0,07**	**0,07**	**0,00**
p_2s_2	18,01	**20,51**	92,62	7,26	0,64	**20,51**	**0,35**	**0,35**	**0,00**
p_2s_3	38,76	**40,21**	1731,24	145,29	12,27	**40,21**	**0,05**	**0,05**	**0,00**

Tabela 5.5 - Resultados comparativos entre o Modelo 5.1 e o Modelo 5.2 para instâncias com 8 máquinas paralelas.

		Modelo 5.1				Modelo 5.2			
Instância	LI	C_{max}	T_t(s)	T_M(s)	GAP	C_{max}	T_t(s)	T_M(s)	GAP
10 Tarefas									
p_1s_1	4,42	**9,54**	0,23	0,10	0,00	**9,54**	**0,01**	**0,01**	**0,00**
p_1s_2	2,52	**9,49**	0,25	0,10	0,00	**9,49**	**0,01**	**0,01**	**0,00**
p_1s_3	5,38	**9,42**	0,33	0,13	0,00	**9,42**	**0,01**	**0,01**	**0,00**
p_2s_1	8,10	**18,55**	0,21	0,09	0,00	**18,55**	**0,01**	**0,01**	**0,00**
p_2s_2	4,97	**18,68**	0,24	0,10	0,00	**18,68**	**0,01**	**0,01**	**0,00**
p_2s_3	9,94	**18,27**	0,34	0,12	0,00	**18,27**	**0,01**	**0,01**	**0,00**
20 Tarefas									
p_1s_1	8,11	**10,51**	276,62	15,22	2,44	**10,51**	**0,09**	**0,09**	**0,00**
p_1s_2	4,62	**9,81**	2,76	0,23	0,00	**9,81**	**0,07**	**0,07**	**0,00**
p_1s_3	10,17	11,61	760,27	28,35	7,24	**11,60**	**0,15**	**0,15**	**0,00**
p_2s_1	15,71	**20,76**	328,01	14,35	3,34	**20,76**	**0,13**	**0,13**	**0,00**
p_2s_2	9,00	**19,52**	2,80	0,23	0,00	**19,52**	**0,08**	**0,08**	**0,00**
p_2s_3	19,38	22,31	958,29	37,31	8,28	**22,30**	**0,26**	**0,26**	**0,00**

Uma análise geral comprova que o Modelo 5.2 encontra soluções melhores ou iguais às encontradas pelo Modelo 5.1 em todas as instâncias testadas, além de consumir tempos computacionais menores em todos os casos testados. Na Tabela 5.3, na Tabela 5.4 e na Tabela 5.5 são observados que ambos os modelos consomem um baixo tempo computacional em instâncias com até dez tarefas, provando a otimalidade de todas as instâncias deste tipo. Já em

instâncias de vinte tarefas, é possível verificar a superioridade do Modelo 5.2, que prova a otimalidade destas instâncias em tempos reduzidos. Já o Modelo 5.1 enfrenta dificuldades em instâncias com vinte tarefas. Mesmo encontrando a solução ótima da maioria das instâncias, os tempos computacionais e GAP do Modelo 5.1 são elevados, principalmente quando é utilizado um número menor de máquinas paralelas. Na Tabela 5.5 é possível destacar que o Modelo 5.1 não encontra a solução ótima em todas as instâncias com vinte tarefas do tipo "p_1s_3" e "p_2s_3".

Os testes atestam que o Modelo 5.1 encontra menos dificuldade em instâncias do tipo "s_2" e mais dificuldade nas do tipo "s_3". É constatado também que o aumento do número de máquinas paralelas resulta em uma diminuição do tempo computacional para o Modelo 5.1.

Os testes computacionais para o Modelo 5.2 foram estendidos utilizando instâncias de até cem tarefas, conforme Tabela 5.6, Tabela 5.7 e Tabela 5.8.

Tabela 5.6 - Resultados computacionais do Modelo 5.2 para instâncias com 2 máquinas paralelas.

		Modelo 5.2			
Instância	LI	C_{max}	T_t(s)	T_M(s)	GAP
50 Tarefas					
p_1s_1	78,11	82,30	1,63	0,19	0,00
p_1s_2	43,44	43,94	564,00	1,34	0,53
p_1s_3	98,38	101,30	0,06	0,10	0,00
p_2s_1	149,35	157,52	10,08	0,32	0,00
p_2s_2	83,33	84,32	614,12	10,13	0,36
p_2s_3	185,93	192,34	0,08	0,11	0,00
100 Tarefas					
p_1s_1	153,52	159,78	179,96	0,94	0,06
p_1s_2	85,09	85,56	1748,21	48,64	1,64
p_1s_3	194,00	198,75	0,22	0,11	0,00
p_2s_1	293,92	305,58	97,68	10,55	0,02
p_2s_2	162,31	163,36	1779,00	136,21	1,17
p_2s_3	374,31	383,73	0,94	0,17	0,00

Tabela 5.7 - Resultados computacionais do Modelo 5.2 para instâncias com 4 máquinas paralelas.

		Modelo 5.2			
Instância	LI	C_{max}	T_t(s)	T_M(s)	GAP
50 Tarefas					
p_1s_1	39,05	41,43	4,75	0,22	0,00
p_1s_2	21,72	22,18	284,93	2,51	0,62
p_1s_3	49,19	50,90	0,13	0,13	0,00
p_2s_1	74,68	78,97	0,68	0,33	0,00
p_2s_2	41,67	42,40	432,65	17,45	0,49
p_2s_3	92,96	96,40	0,09	0,14	0,00
100 Tarefas					
p_1s_1	76,76	80,09	59,56	1,09	0,04
p_1s_2	42,55	43,06	1390,48	33,26	1,74
p_1s_3	97,00	99,64	0,27	0,18	0,00
p_2s_1	146,96	153,03	63,29	6,74	0,02
p_2s_2	81,16	81,98	1573,48	141,58	1,17
p_2s_3	187,16	192,11	0,42	0,29	0,00

Tabela 5.8 - Resultados computacionais do Modelo 5.2 para instâncias com 8 máquinas paralelas.

		Modelo 5.2			
Instância	LI	C_{max}	T_t(s)	T_M(s)	GAP
50 Tarefas					
p_1s_1	19,53	20,96	18,25	0,26	0,05
p_1s_2	10,86	11,77	384,46	2,30	1,43
p_1s_3	24,60	25,71	0,12	0,17	0,00
p_2s_1	37,34	39,72	6,82	1,12	0,00
p_2s_2	20,83	22,46	637,25	9,08	1,45
p_2s_3	46,48	48,45	0,19	0,24	0,00
100 Tarefas					
p_1s_1	38,38	40,34	41,19	1,60	0,05
p_1s_2	21,27	21,77	781,35	53,69	1,71
p_1s_3	48,50	50,07	0,34	0,28	0,00
p_2s_1	73,48	76,81	86,97	8,65	0,03
p_2s_2	40,58	41,40	1302,40	153,53	1,62
p_2s_3	93,58	96,34	0,97	0,94	0,00

Os resultados estendidos demonstram que o Modelo 5.2 segue provando a otimalidade de todas as instâncias do tipo "s_3" testadas, apresentando bons tempos computacionais. Para os tipos "s_1" e "s_2", o Modelo 5.2 não consegue provar a otimalidade de todas as instâncias, porém oferece boas soluções e limitantes.

Instâncias do tipo "s_2" tendem a consumir um maior esforço computacional, podendo ser consideradas mais difíceis para o Modelo 5.2. Este comportamento também pode ser observado na seção 3.3.2 no problema em que a máquina única é considerada $1|s_j,B|C_{max}$. Esta diferença entre as instâncias pode ser explicada pelo tamanho pequeno das tarefas, que elevam o número de combinações de possíveis de bateladas, gerando um espaço solução maior para o problema. Com isso, é possível definir uma relação de dificuldade para as instâncias testadas, onde instâncias do tipo "s_2" mais difíceis que "s_3", que são mais difíceis que "s_1". Também é constatado que o resolvedor, utilizando o Modelo 5.2, encontra boas soluções em um tempo muito inferior ao tempo em que o mesmo consome para confirmar a otimalidade. Isto pode ser confirmado na comparação das colunas de tempo computacional total $T_t(s)$ e tempo da melhor solução inteira encontrada $T_M(s)$.

5.4 Conclusões do capítulo

Neste capítulo foram revisados os conceitos e observações que envolvem a minimização do *makespan* no problema de dimensionamento e programação de bateladas em máquinas paralelas idênticas com tarefas de tamanhos não idênticos ($P_m|s_j,B|C_{max}$), apresentando suas definições e o modelo matemático de programação mista encontrado na literatura. É proposto um modelo matemático de programação inteira mista para a resolução deste problema, que foi comparado com um modelo e heurísticas encontradas na literatura.

É possível atestar através dos resultados computacionais que o modelo proposto é superior ao modelo encontrado na literatura, encontrando soluções melhores em tempo computacional menor, além de provar a otimalidade de um número muito maior de instâncias testadas.

6 MODELOS MATEMÁTICOS PARA O PROBLEMA $P_m|r_j,s_j,B|C_{max}$

Neste capítulo é proposto um modelo matemático de programação inteira mista para a minimização do *makespan* no problema de dimensionamento e programação de bateladas em máquinas paralelas idênticas, com tarefas de tamanhos e tempos de liberação não idênticos ($P_m|r_j,s_j,B|C_{max}$). Definições, exemplos, modelo matemático encontrado na literatura e observações relevantes são apresentados na seção 6.1. Um novo modelo matemático é proposto na seção 6.2. Os resultados computacionais dos modelos matemáticos são comparados e discutidos na seção 6.3 e as conclusões deste capítulo são apresentadas na seção 6.4.

6.1 Descrição do problema e modelo da literatura

Formalmente, este problema pode ser definido como: dado um conjunto J de tarefas, cada elemento $j \in J$ pode ser definido por um terno de atributos $\{r_j, p_j, s_j\}$, que representam o tempo de liberação, o tempo de processamento e o tamanho da tarefa, respectivamente. Cada tarefa $j \in J$ deve ser designada a uma batelada $k \in K$, respeitando o limite de capacidade B das máquinas de processamento. Cada batelada $k \in K$ utilizada deve ser designada a uma máquina de processamento $m \in M$. O tempo de processamento de cada batelada $k \in K$ é definida como $P_k = \max\{p_j, j \in k\}$. Cada tarefa $j \in J$ é disponível no tempo r_j, ou seja, o problema considera tarefas com tempos de liberação diferentes. O tempo de liberação de cada batelada $k \in K$ é definida como $R_k = \max\{r_j, j \in k\}$. As tarefas não podem ser divididas entre as bateladas, não sendo possível adicioná-las ou removê-las da máquina durante o seu processamento. O tempo total de processamento C_{max} (*makespan*) será igual ao tempo da última máquina a terminar o processamento. O objetivo é encontrar uma configuração de bateladas $k \in K$ e uma programação destas bateladas às máquinas $m \in M$, de modo que o C_{max} seja minimizado. O número de bateladas utilizadas depende do número de tarefas e da capacidade da máquina consideradas na instância, sendo que no pior caso, este número será igual à quantidade de tarefas, $|K| = |J|$.

Na Tabela 6.1 é apresentada uma pequena instância com sete tarefas, utilizada como exemplo. A representação gráfica da solução ótima desta instância é ilustrada na Figura 6.1,

na qual cada tarefa é representada por uma seta e um retângulo. A seta em cinza à esquerda representa o tempo de liberação da tarefa e o retângulo à direita representa os valores de tamanho (altura do retângulo) e tempo de processamento (largura do retângulo) da tarefa. Estes valores são destacados na ilustração através da tarefa 1. Neste exemplo, são utilizadas duas máquinas paralelas (m_1 e m_2) e a capacidade B das máquinas adotadas para este exemplo são de 10 unidades. A solução ótima deste exemplo apresenta quatro bateladas, "k_1", "k_2", "k_3" e "k_4". A última tarefa é processada após 17 unidades de tempo (C_{max}).

Tabela 6.1 - Dados para instância exemplo do problema $P_m|r_j,s_j,B|C_{max}$.

j	1	2	3	4	5	6	7
r_j	8	9	3	6	10	5	1
p_j	9	5	8	5	6	3	4
s_j	3	6	1	6	4	3	2

Uma particularidade deste problema é que a utilização de um número maior de máquinas paralelas não necessariamente se converte em uma melhora na solução ótima para o problema. Este fato pode ser facilmente constatado neste exemplo. Como observado na Figura 6.1, as bateladas "k_2" e "k_4" terminam simultaneamente no final do processamento das duas máquinas paralelas. No caso da batelada "k_4", a disponibilidade de uma nova máquina poderia antecipar o início do seu processamento. Porém, a batelada "k_2" não seria favorecida desta hipótese, fazendo com que a solução ótima não fosse alterada.

Com isso, pode-se dizer que o menor valor para C_{max} no problema $P_m|r_j,s_j,B|C_{max}$ quando o número de máquinas paralelas aumenta tende a ser $\max_{\forall i \in J}\{r_j + p_j\}$.

A complexidade do problema $P_m|r_j,s_j,B|C_{max}$ é considerada NP-Difícil, como provado em Damodaran, Gallego e Maya (2009), mostrando que quando assumido que $\{r_j = 0, j \in J\}$ e $\{s_j = S\,, j \in J\}$, este problema é reduzido a um caso particular do problema clássico $P_m|\,|C_{max}$, comprovadamente NP-Difícil.

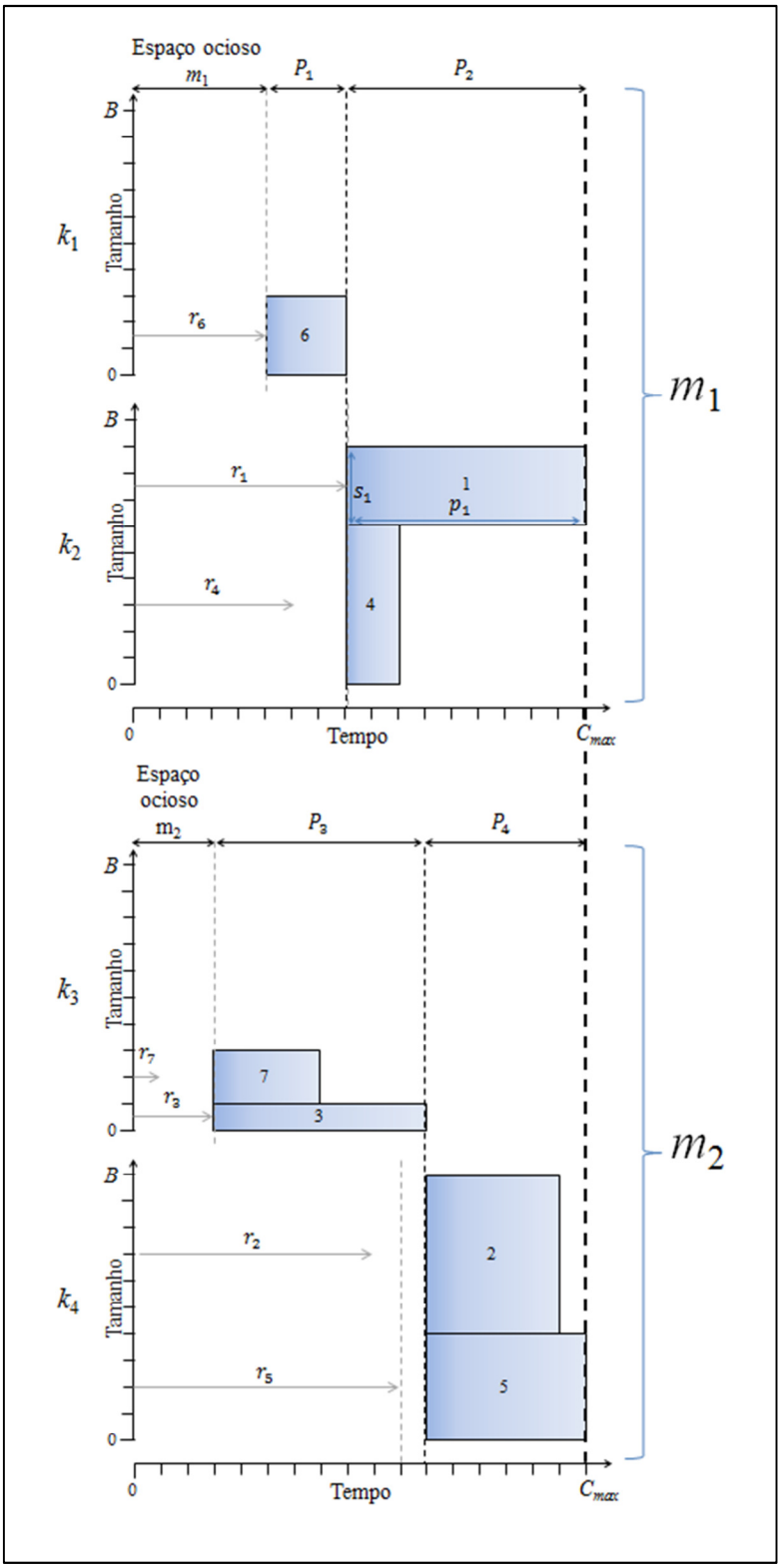


Figura 6.1 - Solução ótima a partir da instância exemplo do problema $P_m|r_j,s_j,B|C_{max}$.

Um modelo de programação linear inteira mista é proposto em Vélez-Gallego (2009) para a solução do problema. Os parâmetros e variáveis definidos para o Modelo 5.1 são

utilizados por este modelo. Parâmetros e variáveis adicionais, em conjunto com o modelo são definidos a seguir.

Parâmetros:

r_j: tempo de liberação da tarefa j

Variáveis:

S_{km}: tempo em que a batelada $k \in K$ começa a ser processada na máquina $m \in M$

Modelo 6.1

$$Min\ C_{max} \tag{6.1}$$

S.a.

$$\sum_{j \in J} s_j x_{jkm} \leq B \qquad \forall k \in K, \forall m \in M \tag{6.2}$$

$$\sum_{k \in K} \sum_{m \in M} x_{jkm} = 1 \qquad \forall k \in K, \forall m \in M \tag{6.3}$$

$$S_{km} \geq r_j x_{jkm} \qquad \forall k \in K, \forall j \in J, \forall m \in M \tag{6.4}$$

$$S_{km} \geq S_{k-1,m} + p_j x_{j,k-1,m} \qquad \forall k \in K \backslash \{1\}, \forall j \in J, \forall m \in M \tag{6.5}$$

$$C_{max} \geq S_{|K|,m} + p_j x_{j,|K|,m} \qquad \forall j \in J, \forall m \in M \tag{6.6}$$

$$x_{jkm} \in \{0,1\} \qquad \forall k \in K, \forall j \in J, \forall m \in M \tag{6.7}$$

$$S_{km} \geq 0 \qquad \forall k \in K, \forall m \in M \tag{6.8}$$

$$C_{max} \geq 0 \tag{6.9}$$

A função objetivo (6.1) minimiza o tempo de utilização da última máquina a terminar o processamento (*makespan*). A restrição (6.2) determina que cada batelada não exceda a capacidade das máquinas. A restrição (6.3) determina que cada tarefa seja designada somente a uma batelada e a uma única máquina. As restrições (6.4) e (6.5) determinam o tempo em que a batelada k começa a ser processada na máquina designada. A restrição (6.6) determina o tempo de utilização da última máquina a terminar o processamento (*makespan*). As restrições (6.7),(6.8) e (6.9) determinam o domínio das variáveis.

Vélez-Gallego utiliza um número máximo de bateladas que uma máquina pode processar na formulação do modelo. Este limitante diminui o número de variáveis e o espaço solução do modelo publicado, melhorando o desempenho computacional dos resolvedores. A definição é apresentada abaixo:

$$|K| = \frac{|J|}{|M|} \tag{6.10}$$

No entanto, este corte de variáveis pode levar o Modelo 6.1 a não encontrar a solução ótima em certos casos. Um exemplo pode ser montado a partir da instância representada na Tabela 6.2, em que aparecem várias tarefas com tempos de processamentos pequenos e poucas com tempos de processamento grandes.

Tabela 6.2 - Dados para a segunda instância exemplo do problema $P_m|r_j,s_j,B|C_{max}$.

$\boldsymbol{j}$	1	2	3	4	5	6	7
$\boldsymbol{r_j}$	1	1	3	5	7	0	0
$\boldsymbol{p_j}$	2	2	2	2	2	10	10
$\boldsymbol{s_j}$	5	5	8	8	7	5	5

A Figura 6.2 representa a solução ótima da instância exemplo apresentada na Tabela 6.2. Neste exemplo é identificado que a solução ótima apresenta seis bateladas, "k_1", "k_2", "k_3", "k_4", "k_5" e "k_6". Para a primeira máquina foram designadas cinco destas bateladas, sendo que na segunda máquina apenas uma. Como a instância exemplo apresenta sete tarefas, o número máximo de bateladas por máquina definido pela fórmula (6.10) seria quatro, arredondado o valor discreto para cima. Com isso, é notado que esta regra não pode ser aplicada na solução ótima do exemplo da Tabela 6.2 e, verificando a programação em geral, é possível perceber que não há uma solução ótima para o problema que utilize apenas quatro bateladas por máquina paralela.

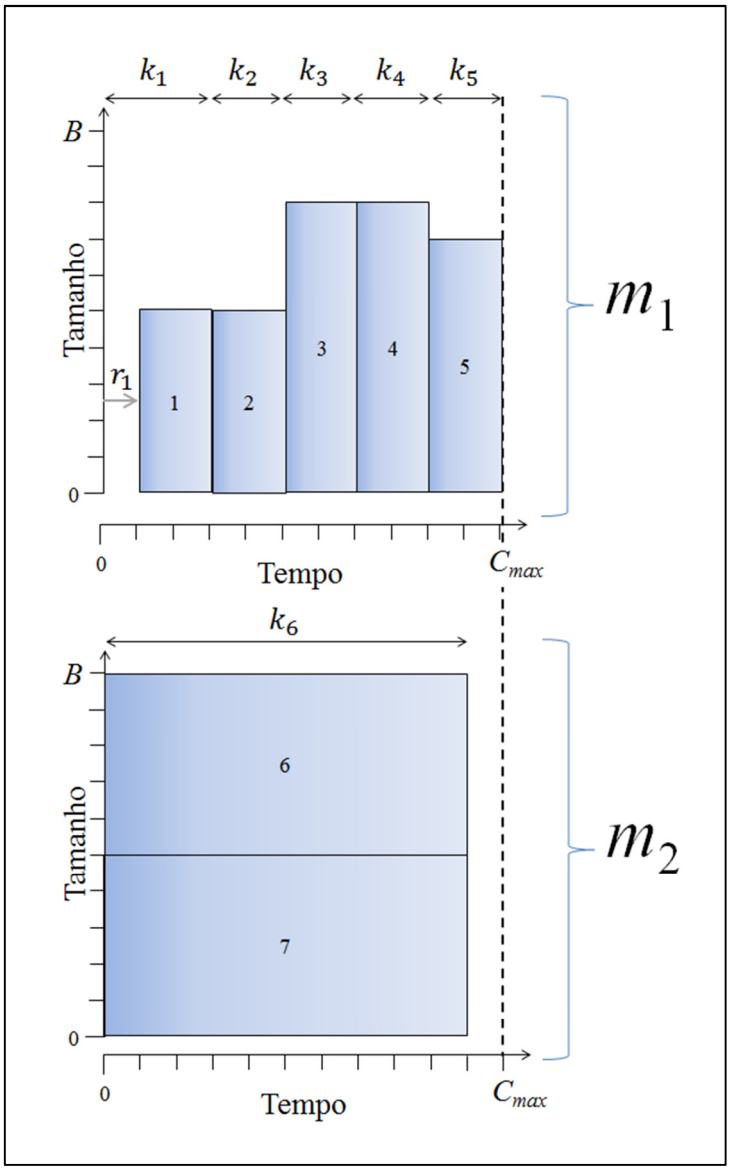


Figura 6.2 - Solução ótima a partir da instância exemplo apresentada na Tabela 6.2.

6.2 Modelo proposto

Este modelo apresenta melhorias muito similares as encontradas nos problemas $1|r_j,s_j,B|C_{max}$ e $P_m|s_j,B|C_{max}$, exploradas nas seções 4.2 e 5.2. Entre elas, pode ser destacada a estratégia que garante que os tempos de ociosidade das máquinas sejam mínimos, designando as bateladas em ordem crescente pelos seus tempos de liberação. Esta melhoria é baseada em

restrições que garantem este sequenciamento, diminuindo o espaço solução do problema original e garantindo a otimalidade da solução.

Outra estratégia adotada é da representação das bateladas no problema de forma idêntica a utilizada primeiramente neste trabalho no problema $1|s_j,B|C_{max}$, sem a replicação das variáveis para cada máquina paralela. Como discutido nos resultados computacionais, esta representação melhora bastante o seu desempenho computacional.

O novo conjunto de restrições necessita que as tarefas sejam ordenadas pelos seus tempos de liberação de forma não decrescente.

Tendo o número de bateladas *m* igual ao número de tarefas *n*, pode-se definir que a batelada *k* só será utilizada se a tarefa k for atribuída a ele, ou seja, a variável x_{kk} define a utilização da batelada *k* se igual a 1. A variável y_{km} define o destino da batelada *k* a máquina m, se $x_{kk} = 1$. Com isso, as variáveis que definem a batelada *k* não são replicadas para cada máquina paralela, como ocorre no Modelo 6.1. A batelada k não deve conter as tarefas $j: j > k$ e as variáveis $x_{jk}: j > k$ não são consideradas na formulação. A partir da ordenação das tarefas de forma crescente de acordo com tempos de liberação, pode ser definido explicitamente que a batelada k sempre tem a tarefa k como a de maior tempo de liberação. Desta forma uma batelada k, se utilizada, tem obrigatoriamente um tempo de liberação maior ou igual a qualquer batelada $j: j < k$.

O Modelo 6.2 proposto definido a seguir utiliza os mesmos parâmetros do Modelo 6.1:

Variáveis:

$x_{jk} = \begin{cases} 1 \text{ se a tarefa } j \in J \text{ for atribuída à batelada } k \in K \\ 0 \text{ caso contrário} \end{cases}$

$y_{km} = \begin{cases} 1 \text{ se a batelada } k \in K \text{ for processada na máquina } m \in M \\ 0 \text{ caso contrário} \end{cases}$

P_{km} : tempo de processamento da batelada $k \in K$, processada na máquina $m \in M$

S_{km} : tempo em que a batelada $k \in K$ começa a ser processada na máquina $m \in M$

C_m : tempo de utilização da máquina $m \in M$

C_{max} : tempo de utilização da última máquina a terminar o processamento (*makespan*)

Modelo 6.2

$$Min\ C_{max} \tag{6.11}$$

S.a.

$$\sum_{\substack{j\in J:\\ j\le k}} s_j x_{jk} \le B x_{kk} \qquad \forall k \in K \tag{6.12}$$

$$\sum_{\substack{k\in K:\\ j\le k}} x_{jk} = 1 \qquad \forall j \in J \tag{6.13}$$

$$x_{jk} \le x_{kk} \qquad \forall j \in J, \forall k \in K: j < k \tag{6.14}$$

$$x_{kk} \le \sum_{m\in M} y_{km} \qquad \forall k \in K \tag{6.15}$$

$$P_{km} \ge p_j(x_{jk} + y_{km} - 1) \qquad \forall j \in J, \forall k \in K: j < k, \forall m \in M \tag{6.16}$$

$$S_{km} \ge r_j y_{km} \qquad \forall k \in K, \forall m \in M \tag{6.17}$$

$$S_{km} \ge S_{k-1,m} + P_{j,k-1} \qquad \forall k \in K, \forall m \in M \tag{6.18}$$

$$C_m \ge S_{|K|,m} + p_j x_{j,|K|} \qquad \forall m \in M \tag{6.19}$$

$$C_{max} \ge C_m \qquad \forall m \in M \tag{6.20}$$

$$x_{jk} \in \{0,1\} \qquad \forall j \in J, \forall k \in K: j \le k \tag{6.21}$$

$$P_{km} \ge 0 \qquad \forall k \in K, \forall m \in M \tag{6.22}$$

$$S_{km} \ge 0 \qquad \forall k \in K, \forall m \in M \tag{6.23}$$

$$C_m \ge 0 \qquad \forall m \in M \tag{6.24}$$

$$C_{max} \ge 0 \tag{6.25}$$

A função objetivo (6.11) minimiza o tempo de utilização da última máquina a terminar o processamento (*makespan*). A restrição (6.12) determina que cada batelada não exceda a capacidade da máquina. A restrição (6.13) determina que cada tarefa seja designada somente a uma batelada. A restrição (6.14) determina que cada tarefa é designada somente a uma batelada com índice maior ou igual ao índice da própria tarefa. A restrição (6.15) determina que todas as bateladas utilizadas sejam designadas a uma máquina. A restrição (6.16) determina o tempo de processamento de cada batelada. As restrições (6.17) e (6.18)

determinam tempo em que a batelada k começa a ser processada na máquina designada. A restrição (6.19) define o tempo de utilização de cada máquina. A restrição (6.20) determina o tempo de utilização da última máquina a terminar o processamento (*makespan*). As restrições (6.21), (6.22), (6.23), (6.24) e (6.25) determinam o domínio das variáveis.

6.3 Resultados computacionais

Para os testes computacionais é utilizado o mesmo conjunto de instâncias gerado neste trabalho, empregado no problema $1|r_j,s_j,B|C_{max}$ e proposto na seção 4.3.2. Os fatores aplicados na geração das instâncias são similares aos utilizados em Xu, Chen e Li (2012), com a adição de novos valores de números de tarefas, como 200, 300 e 500. Além disso, foram adicionadas três novas categorias correspondentes aos números de máquinas paralelas, sendo elas 2, 4 e 8 máquinas.

Os tamanhos, tempos de processamento e de liberação das tarefas são escolhidos aleatoriamente, com distribuição uniforme dentro de um respectivo intervalo. É utilizado o mesmo conjunto de instâncias para os diferentes números de máquinas propostos. Isto significa que uma mesma instância será testada nas diferentes configurações de máquinas paralelas.

Conforme a Tabela 6.3 foram geradas 14 categorias de instâncias, sendo elas a combinação entre sete diferentes números de tarefas e dois limites de tamanho das tarefas. Para cada categoria, foram geradas 20 instâncias diferentes, totalizando 280 instâncias. Cada instância é testada em três configurações de máquinas paralelas diferentes, totalizando 840 testes realizados.

O valor para limites de geração do tempo de liberação foi definido entre [0, C], onde C é o valor obtido através da heurística BFFLPT (*Batch First Fit Lowest Precessing Time*). Nesta heurística, as tarefas são ordenadas a partir de seus tempos de liberação, de forma crescente. Posteriormente, cada tarefa é alocada à primeira batelada a qual a capacidade não é excedida, de forma similar à heurística *first-fit*.

Tabela 6.3 - Fatores gerais utilizados na geração do conjunto de instâncias.

Número de tarefas:	10, 20, 50, 100, 200, 300, 500
Número de máquinas paralelas:	2, 4, 8
Tempo de liberação	r_1: [0, *C*]
Tempo de processamento:	p_1: [8, 48]
Tamanho das tarefas:	s_1: [1, 15] s_2: [15, 35]
Capacidade da máquina:	40

Os testes foram realizados utilizando CPLEX 12.5 em um computador com processador Intel Quad-Core Xeon X3360 2.83 GHz, com 8GB de memória RAM. O tempo máximo de execução para cada instância foi definido arbitrariamente como 1800 segundos.

As tabelas a seguir apresentam os resultados computacionais. A primeira e a segunda coluna apresentam a categoria de instância testada. As colunas 3, 4, 5 e 6 apresentam a média das melhores soluções encontradas C_{max}, a média dos tempos totais de execução $T_t(s)$, a média dos tempos das melhores soluções encontradas $T_M(s)$ e a média dos GAPs fornecidos pelo CPLEX, respectivamente, do Modelo 6.1. As colunas 7, 8, 9 e 10 apresentam a média das melhores soluções encontradas C_{max}, a média dos tempos totais de execução $T_t(s)$, a média dos tempos das melhores soluções encontradas $T_M(s)$ e a média dos GAPs fornecidos pelo CPLEX, respectivamente, do Modelo 6.2. Os resultados computacionais são representados pela média das 20 instâncias testadas em cada categoria.

Tabela 6.4 - Resultados comparativos entre Modelo 6.1 e Modelo 6.2 para instâncias com 10 tarefas.

Instância		Modelo 6.1				Modelo 6.2			
Máquinas	Tipo	C_{max}	$T_t(s)$	$T_M(s)$	GAP	C_{max}	$T_t(s)$	$T_M(s)$	GAP
2	s_1	**106,800**	0,033	0,033	**0,000**	**106,800**	**0,027**	**0,027**	**0,000**
2	s_2	**281,150**	0,058	0,043	**0,000**	**281,150**	**0,012**	**0,012**	**0,000**
4	s_1	**106,600**	**0,018**	**0,018**	**0,000**	**106,600**	0,031	0,031	**0,000**
4	s_2	**278,100**	0,032	0,027	**0,000**	**278,100**	**0,014**	**0,014**	**0,000**
8	s_1	**106,600**	0,091	0,076	**0,000**	**106,600**	**0,066**	**0,055**	**0,000**
8	s_2	**278,100**	0,035	0,025	**0,000**	**278,100**	**0,020**	**0,020**	**0,000**

Tabela 6.5 - Resultados comparativos entre Modelo 6.1 e Modelo 6.2 para instâncias com 20 tarefas.

Instância		Modelo 6.1				Modelo 6.2			
Máquinas	Tipo	C_{max}	T_t(s)	T_M(s)	GAP	C_{max}	T_t(s)	T_M(s)	GAP
2	s_1	**177,050**	1,284	0,478	**0,000**	**177,050**	**0,134**	**0,114**	**0,000**
2	s_2	**521,250**	10,275	1,085	**0,000**	**521,250**	**0,018**	**0,018**	**0,000**
4	s_1	**176,700**	0,669	0,153	**0,000**	**176,700**	**0,279**	**0,132**	**0,000**
4	s_2	**518,350**	0,887	0,225	**0,000**	**518,350**	**0,025**	**0,025**	**0,000**
8	s_1	**176,700**	0,472	0,128	**0,000**	**176,700**	**1,394**	**0,228**	**0,000**
8	s_2	**518,350**	0,986	0,131	**0,000**	**518,350**	**0,037**	**0,037**	**0,000**

Tabela 6.6 - Resultados comparativos entre Modelo 6.1 e Modelo 6.2 para instâncias com 50 tarefas.

Instância		Modelo 6.1				Modelo 6.2			
Máquinas	Tipo	C_{max}	T_t(s)	T_M(s)	GAP	C_{max}	T_t(s)	T_M(s)	GAP
2	s_1	**362,150**	706,722	47,888	0,736	**362,150**	**2,293**	**2,009**	**0,000**
2	s_2	1254,050	1653,546	182,581	40,691	**1253,400**	**0,056**	**0,056**	**0,000**
4	s_1	**361,550**	367,337	10,226	0,000	**361,550**	**2,537**	**1,111**	**0,000**
4	s_2	**1251,400**	1624,649	12,463	8,234	**1251,400**	**0,187**	**0,128**	**0,000**
8	s_1	**361,550**	336,810	7,954	0,011	**361,550**	**6,457**	**1,922**	**0,000**
8	s_2	**1251,400**	950,792	7,753	1,454	**1251,400**	**0,465**	**0,253**	**0,000**

Tabela 6.7 - Resultados comparativos entre Modelo 6.1 e Modelo 6.2 para instâncias com 100 tarefas.

Instância		Modelo 6.1				Modelo 6.2			
Máquinas	Tipo	C_{max}	T_t(s)	T_M(s)	GAP	C_{max}	T_t(s)	T_M(s)	GAP
2	s_1	707,400	1800,000	1659,145	94,787	**691,000**	**53,534**	**28,296**	**0,000**
2	s_2	2666,950	1800,000	1558,003	90,790	**2501,450**	**0,183**	**0,179**	**0,000**
4	s_1	**690,050**	1800,000	451,207	73,248	**690,050**	**28,760**	**9,313**	**0,000**
4	s_2	**2498,550**	1636,681	599,112	74,420	**2498,550**	**1,322**	**0,663**	**0,000**
8	s_1	**690,050**	1653,785	108,068	30,617	**690,050**	**45,427**	**12,394**	**0,000**
8	s_2	**2498,550**	1657,189	322,266	83,481	**2498,550**	**4,125**	**1,463**	**0,000**

Tabela 6.8 - Resultados computacionais do Modelo 6.2 para instâncias com 200 tarefas.

Instância		Modelo 6.2			
Máquinas	Tipo	C_{max}	T_t(s)	T_M(s)	GAP
2	s_1	1318,700	338,466	243,030	0,000
2	s_2	4993,850	0,755	0,720	0,000
4	s_1	1317,250	322,922	115,818	0,008
4	s_2	4990,850	10,989	6,619	0,000
8	s_1	1317,250	256,722	142,416	0,000
8	s_2	4990,200	34,558	21,045	0,000

Tabela 6.9 - Resultados computacionais do Modelo 6.2 para instâncias com 300 tarefas.

Instância		Modelo 6.2			
Máquinas	Tipo	C_{max}	T_t(s)	T_M(s)	GAP
2	s_1	1981,700	1338,117	1147,109	0,313
2	s_2	7414,500	2,257	1,562	0,000
4	s_1	1977,100	804,331	442,762	0,010
4	s_2	7410,150	18,773	7,381	0,000
8	s_1	1977,050	1234,044	564,378	0,047
8	s_2	7410,150	82,080	49,809	0,000

Tabela 6.10 - Resultados computacionais do Modelo 6.2 para instâncias com 500 tarefas.

Instância		Modelo 6.2			
Máquinas	Tipo	C_{max}	T_t(s)	T_M(s)	GAP
2	s_1	3432,900	1797,005	1531,461	6,207
2	s_2	12533,500	5,311	4,826	0,000
4	s_1	3220,450	1797,192	1520,196	26,040
4	s_2	12530,450	52,878	47,955	0,000
8	s_1	850774,500	1800,000	1339,095	97,446
8	s_2	12530,450	329,547	239,956	0,000

Uma análise geral comprova que o Modelo 6.2 encontra soluções melhores ou iguais às encontradas pelo Modelo 6.1 em todas as instâncias testadas. Em muitos casos, as soluções encontradas são de qualidade superior, além de consumir tempos computacionais menores em todos os casos testados.

Analisando instâncias pequenas de até vinte tarefas é possível atestar que os dois modelos provam a otimalidade das soluções e apresentam tempos semelhantes. Mesmo assim, o Modelo 6.2 obtém tempos computacionais melhores em todos os testes, principalmente na categoria "s_2" com duas máquinas paralelas, como verificado na Tabela 6.4 e na Tabela 6.5.

Em instâncias com cinquenta tarefas, o Modelo 6.1 passa a não encontrar a solução ótima na categoria "s_2" com duas máquinas paralelas, apresentando um alto GAP, como indicado na Tabela 6.6. Além disso, mesmo que o Modelo 6.1 encontre as soluções ótimas em todas as outras categorias de instâncias, o mesmo não consegue provar a otimalidade, com exceção de instâncias do tipo "s_1" com quatro máquinas paralelas. Já o Modelo 6.2 prova a otimalidade de todas as instâncias consumindo um tempo inferior.

Os testes comparativos entre o Modelo 6.1 e o Modelo 6.2 foram realizados em instâncias com no máximo 100 tarefas. Na Tabela 6.7 é possível verificar que o Modelo 6.1 atinge o tempo limite de 1800 segundos na maioria das instâncias apresentadas, não encontrando a solução ótima em instâncias com duas máquinas paralelas. O Modelo 6.1 apresenta limitantes muito ruins para as demais instâncias, fornecendo valores altos para os GAPs. Já o Modelo 6.2 prova a otimalidade de todas as instâncias desta tabela em tempos computacionais muito reduzidos. Com isso, torna-se evidente a superioridade do Modelo 6.2 em relação ao Modelo 6.1.

Os testes computacionais para o Modelo 6.2 foram estendidos em instâncias de até quinhentas tarefas, apresentadas na Tabela 6.8, na Tabela 6.9 e na Tabela 6.10. Estes resultados demonstram que o Modelo 6.2 segue provando a otimalidade de todas as instâncias do tipo "s_2", apresentando bons tempos computacionais. Para o tipo "s_1" o Modelo 6.2 não consegue provar a otimalidade das instâncias, porém oferece bons limitantes em instâncias de até trezentas tarefas.

Como ocorrido nos testes computacionais do problema $1|r_j,s_j,B|C_{max}$, descritos na seção 4.3.2, a comparação entre as colunas $T_t(s)$ e $T_M(s)$ indicam que soluções novas são geradas em praticamente todo o tempo computacional consumido no Modelo 6.2. Porém, o Modelo 6.1 apresenta uma significativa diferença entre estes valores, como observados na Tabela 6.6 e na Tabela 6.7.

É possível verificar ao longo dos resultados computacionais, que a utilização de um número maior de máquinas paralelas não necessariamente se converte em uma melhora na solução ótima para o problema. Este evento pode ser confirmado comparando as instâncias com até cem tarefas, quando é aumentado o número de quatro para oito máquinas paralelas.

Este caso específico ocorre porque sempre que a última batelada é disponível para o processamento, pelo menos uma das quatro primeiras máquinas utilizadas já foi desocupada.

Em relação aos tipos de instâncias testadas, é possível afirmar que os resultados computacionais do Modelo 6.2 confirmam que instâncias do tipo "s_1" são mais difíceis que instâncias do tipo "s_2", como já descritas na seção 4.3.2. Porém, esta observação não se concretiza para o Modelo 6.1, apresentando tempos computacionais maiores para instâncias do tipo "s_2" em todos os casos, além do GAP maior na maioria dos casos.

O problema relatado na seção 6.1 sobre o limitante para ao número de bateladas descrito pela fórmula (6.10) não é identificado nos testes realizados. Isto ocorre porque a geração aleatória das instâncias não criou nenhum caso específico, similar ao descrito neste trabalho.

6.4 Conclusões do capítulo

Neste capítulo foram revisados os conceitos e observações que envolvem a minimização do *makespan* no problema de dimensionamento e programação de bateladas em máquinas paralelas idênticas, com tarefas de tamanhos e tempos de liberação não idênticos ($P_m|r_j,s_j,B|C_{max}$), apresentando suas definições e o modelo matemático de programação mista encontrado na literatura. É proposto um modelo matemático de programação inteira mista para a solução deste problema, que foi comparado com um modelo encontrado na literatura.

É possível atestar através dos resultados computacionais que o modelo proposto é superior ao modelo publicado na literatura, encontrando soluções melhores em tempo computacional menor, além de provar a otimalidade de um número muito maior de instâncias testadas.

É possível destacar que o modelo publicado na literatura apresenta um limitante para o número de bateladas que pode fazer com que a solução ótima não seja encontrada em alguns casos. Este problema é relatado e explorado neste trabalho, porém, nos testes elaborados este caso não foi identificado.

7 CONCLUSÕES

Neste trabalho são propostos quatro novos modelos matemáticos para a resolução de quatro problemas para o dimensionamento e programação de bateladas em máquinas de processamento: $1|s_j,B|C_{max}$, $1|r_j,s_j,B|C_{max}$, $P_m|s_j,B|C_{max}$ e $P_m|r_j,s_j,B|C_{max}$. Através de extensivos experimentos computacionais utilizando o CPLEX é demonstrado que as novas formulações são mais eficientes do que os modelos encontrados na literatura, fornecendo melhores soluções com um tempo computacional menor, além de provar a otimalidade de um número maior de instâncias testadas. Os novos modelos possibilitaram o tratamento de instâncias com um número significativamente maior de tarefas do que outros modelos.

O modelo proposto para o problema $1|s_j,B|C_{max}$ foi testado para instâncias com até quinhentas tarefas, apresentando bons resultados computacionais. O modelo para este problema publicado por Melouk, Damodaran e Chang (2004) se mostrou competitivo em instâncias com até dez tarefas. A partir de instâncias com vinte tarefas o modelo proposto se mostra mais eficiente. O novo modelo também foi comparado com métodos heurísticos, como o Algoritmo Genético publicado em Damodaran, Manjeshwar e Srihari (2006) e o algoritmo CACB publicado em Chen, Du e Huang (2011). Os testes revelaram a superioridade na qualidade das soluções oferecidas pelo modelo proposto, além de revelar tempos computacionais competitivos em vários casos.

O modelo proposto para o problema $1|r_j,s_j,B|C_{max}$ foi testado para instâncias com até quinhentas tarefas, apresentando bons resultados computacionais. O modelo publicado por Xu, Chen e Li (2012) se mostrou competitivo em instâncias com até dez tarefas. A partir de instâncias com vinte tarefas o modelo proposto se mostra mais eficiente. O novo modelo também foi comparado com a abordagem do método Colônia de Formigas publicado em Xu, Chen e Li (2012). Os testes revelaram a superioridade na qualidade das soluções oferecidas pelo modelo proposto, além de revelar tempos computacionais competitivos em vários casos.

O modelo proposto para o problema $P_m|s_j,B|C_{max}$ foi testado para instâncias com até cem tarefas, apresentando bons resultados computacionais. O modelo publicado por Chang, Damodaran e Melouk (2004) se mostrou competitivo em instâncias com até dez tarefas. A partir de instâncias com vinte tarefas o modelo proposto se mostra mais eficiente.

O modelo proposto para o problema $P_m|r_j,s_j,B|C_{max}$ foi testado para instâncias com até quinhentas tarefas, apresentando bons resultados computacionais. O modelo publicado por

Vélez-Gallego (2009) se mostrou competitivo em instâncias com até vinte tarefas. A partir de instâncias com cinquenta tarefas o modelo proposto se mostra mais eficiente.

Em trabalhos futuros é possível explorar as características utilizadas na construção dos novos modelos para criar métodos do tipo *matheuristics*.

REFERÊNCIAS BIBLIOGRÁFICAS

CHANG, P.-Y., DAMODARAN, P., e MELOUK, S. (2004). **Minimizing makespan on parallel batch processing machines**. International Journal of Production Research, 42(19), 4211-4220.

CHEN, H., DU, B., e HUANG, G. Q. (2011). **Scheduling a batch processing machine with non-identical job sizes: a clustering perspective**. International Journal of Production Research, 49(19), 5755-5778.

CHENG, B., WANG, Q., YANG, S., e HU, X. (2013). **An improved ant colony optimization for scheduling identical parallel batching machines with arbitrary job sizes**. Applied Soft Computing, 13(2), 765-772.

CHENG, B., YANG, S., HU, X., e CHEN, B. (2012). **Minimizing makespan and total completion time for parallel batch processing machines with non-identical job sizes**. Applied Mathematical Modelling, 36(7), 3161-3167.

CHOU, F.-D., CHANG, P.-C., e WANG, H.-M. (2005). **A hybrid genetic algorithm to minimize makespan for the single batch machine dynamic scheduling problem**. The International Journal of Advanced Manufacturing Technology, 31(3-4), 350-359.

CHUNG, S. H., TAI, Y. T., e PEARN, W. L. (2009). **Minimising makespan on parallel batch processing machines with non-identical ready time and arbitrary job sizes**. International Journal of Production Research, 47(18), 5109-5128.

DAMODARAN, P., GHRAYEB, O., e GUTTIKONDA, M. C. (2013). **GRASP to minimize makespan for a capacitated batch-processing machine**. The International Journal of Advanced Manufacturing Technology, 68(1-4), 407-414.

DAMODARAN, P., HIRANI, N. S., e GALLEGO, M. C. V. (2009). **Scheduling identical parallel batch processing machines to minimise makespan using genetic algorithms**. European J. of Industrial Engineering, 3(2), 187.

DAMODARAN, P., MANJESHWAR, P. K., e SRIHARI, K. (2006). **Minimizing makespan on a batch-processing machine with non-identical job sizes using genetic algorithms**. International Journal of Production Economics, 103(2), 882-891.

DAMODARAN, P., VÉLEZ-GALLEGO, M. C., e MAYA, J. (2009). **A GRASP approach for makespan minimization on parallel batch processing machines**. Journal of Intelligent Manufacturing, 22(5), 767-777.

DUPONT, L., e DHAENENS-FLIPO, C. (2002). **Minimizing the makespan on a batch machine with non-identical job sizes: an exact procedure**. Computers & Operations Research, 29(7), 807-819.

FANTI, M. P., MAIONE, B., PISCITELLI, G., e TURCHIANO, B. (1996), **Heuristic scheduling of jobs on a multi-product batch processing machine**. Int J Prod Res, 34(8), 2163-2186.

GHAZVINI, F. J. e DUPONT, L. (1998). **Minimizing mean flow times criteria on a single batch processing machine with non-identical jobs sizes**. International Journal of Production Economics, 55, 273-280.

GRAHAM, R. L., LAWLER, E. L., LENSTRA, J. K., e RINNOOY KAN, A. H. G. (1979). **Optimization and approximation in deterministic sequencing and scheduling : a survey**. Proceedings of the Advanced Research Institute on Discrete Optimization and Systems Applications of the Systems Science Panel of NATO and of the Discrete Optimization Symposium co-sponsored by IBM Canada and SIAM Banff, Aha. and Vancouver, 5, 287-326.

KASHAN, A. H., KARIMI, B., e JOLAI, F. (2006). **Effective hybrid genetic algorithm for minimizing makespan on a single-batch-processing machine with non-identical job sizes**. International Journal of Production Research, 44(12), 2337-2360.

KASHAN, A., KARIMI, B., e JENABI, M. (2008). **A hybrid genetic heuristic for scheduling parallel batch processing machines with arbitrary job sizes**. Computers & Operations Research, 35(4), 1084-1098.

LEE, Y. H., e LEE, Y. H. (2013). **Minimising makespan heuristics for scheduling a single batch machine processing machine with non-identical job sizes**. International Journal of Production Research, 51(12), 3488-3500.

MATHIRAJAN, M., e SIVAKUMAR, A. I. (2006). **A literature review, classification and simple meta-analysis on scheduling of batch processors in semiconductor**. The International Journal of Advanced Manufacturing Technology, 29(9-10), 990-1001.

MATHIRAJAN, M., SIVAKUMAR, A.I., e CHANDRU, V. (2004), **Scheduling algorithms for heterogeneous batch processors with incompatible job families**. J Intell Manuf, 15, 787-803

MELOUK, S., DAMODARAN, P., e CHANG, P.-Y. (2004). **Minimizing makespan for single machine batch processing with non-identical job sizes using simulated annealing**. International Journal of Production Economics, 87(2), 141-147.

MENG, Y., e TANG, L. (2010). **A tabu search heuristic to solve the scheduling problem for a batch-processing machine with non-identical job sizes**. In 2010 International Conference on Logistics Systems and Intelligent Management (ICLSIM) (Vol. 3, pp. 1703-1707). IEEE.

MÖNCH, L., FOWLER, J. W., DAUZÈRE-PÉRÈS, S., MASON, S. J., e ROSE, O. (2011). **A survey of problems, solution techniques, and future challenges in scheduling semiconductor manufacturing operations**. Journal of Scheduling, 14(6), 583-599.

PARSA, N. R., KARIMI, B., e HUSSEINZADEH KASHAN, A. (2010). **A branch and price algorithm to minimize makespan on a single batch processing machine with non-identical job sizes**. Computers & Operations Research, 37(10), 1720-1730.

POTTS, C. N., e KOVALYOV, M. Y. (2000). **Scheduling with batching: A review**. European Journal of Operational Research, 120(2), 228-249.

RAM, B., e PATEL, G. (1998) **Modeling furnace operations using simulation and heuristics**. Proc 1998 winter simulation conference, 957-963.

UZSOY, R. (1994). **Scheduling a single batch processing machine with non-identical job sizes**. International Journal of Production Research, 32(7), 1615-1635.

VÉLEZ-GALLEGO, M. C. (2009). **Algorithms for scheduling parallel batch processing machines with non-identical job ready times**. Ph.D. dissertation, Florida International University, Florida, United States.

XU, R., CHEN, H., e LI, X. (2012). **Makespan minimization on single batch-processing machine via ant colony optimization**. Computers & Operations Research, 39(3), 582-593.

YUAN, J. J., LIU, Z. H., NG, C.T., e CHENG, T.C.E. (2004), **The unbounded single machine parallel batch scheduling problem with family jobs and release dates to minimize makespan**. Theor Comput Sci, 320(2-3), 199-213.

ZEE, D. J. VAN DER., VAN HARTEN, A., e SCHUUR P. C. (1997), **Dynamic job assignment heuristics for multi-server batch operations - A cost based approach**. Int J Prod Res, 35(11), 3063-3093.

ZHANG, G., CAI, X., LEE, C.-Y., e WONG, C. (2001). **Minimizing makespan on a single batch processing machine with nonidentical job sizes**. Naval Research Logistics, 48(3), 226-240.