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

UM MODELO DE OTIMIZAÇÃO PARA O PROBLEMA DE DIMENSIONAMENTO E PROGRAMAÇÃO DE LOTES DE PRODUÇÃO EM MÁQUINA ÚNICA

DISSERTAÇÃO DE MESTRADO

Cezaraugusto Gomes Scalcon

Santa Maria, RS, Brasil

2012

PPGEP/UFSM, RS SCALCON, Cezaraugusto Gomes Mestre 2012

UM MODELO DE OTIMIZAÇÃO PARA O PROBLEMA DE DIMENSIONAMENTO E PROGRAMAÇÃO DE LOTES DE PRODUÇÃO EM MÁQUINA ÚNICA

Cezaraugusto Gomes Scalcon

Dissertação apresentada ao Curso de Mestrado do Programa de Pós-Graduação em Engenharia de Produção, Área de concentração em Gerência de Produção, da Universidade Federal de Santa Maria (UFSM,RS), como requisito parcial para obtenção do grau de

Mestre em Engenharia de Produção

Orientador: Prof. Dr. Felipe Martins Müller

Santa Maria, RS, Brasil

2012

Scalcon, Cezaraugusto Gomes
UM MODELO DE OTIMIZAÇÃO PARA O PROBLEMA DE DIMENSIONAMENTO E PROGRAMAÇÃO DE LOTES DE PRODUÇÃO EM MÁQUINA ÚNICA / Cezaraugusto Gomes Scalcon.-2012.
64 f.; 30cm

Orientador: Felipe Martins Müller
Dissertação (mestrado) - Universidade Federal de Santa Maria, Centro de Tecnologia, Programa de Pós-Graduação em Engenharia de Produção, RS, 2012

1. programação inteira 2. dimensionamento de lotes 3. programação da produção I. Müller, Felipe Martins II. Título.

Ficha catalográfica elaborada através do Programa de Geração Automática da Biblioteca Central da UFSM, com os dados fornecidos pelo(a) autor(a).

Endereço: Rua Hipólito Ribeiro, 579. Centro – Bagé – RS – Brasil, CEP 96.400-430
Fone (0xx)53 3240-6888, Cel (0xx)53 9973-7932; email: cezarscalcon@ gmail.com

Universidade Federal de Santa Maria
Centro de Tecnologia
Programa de Pós-Graduação em Engenharia de Produção

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

UM MODELO DE OTIMIZAÇÃO PARA O PROBLEMA DE DIMENSIONAMENTO E PROGRAMAÇÃO DE LOTES DE PRODUÇÃO EM MÁQUINA ÚNICA

Elaborado por
Cezaraugusto Gomes Scalcon

como requisito parcial para a obtenção do grau de
Mestre em Engenharia de Produção

COMISÃO EXAMINADORA:

FELIPE MARTINS MÜLLER, Dr.
(Presidente/Orientador)

OLINTO CÉSAR BASSI DE ARAUJO, Dr.
(UFSM – POLITÉCNICO)

KAREN PAZ BASTOS, Dra.
(CEITEC)

Santa Maria, 2 de julho de 2012

DEDICATÓRIA

A Deus, a minha família, aos colegas e professores que me auxiliaram nesta trajetória. Sem os quais não teria alcançado o êxito necessário.

AGRADECIMENTOS

Ao Professor Dr. Felipe Martins Müller, pela compreensão e pelos ensinamentos.

Ao Professor Dr. Olinto Araújo, pela paciência e ensinamentos que direcionaram a elaboração desse trabalho.

A minha esposa Rejane e minhas filhas Stella e Andreza, pelos momentos distantes, pelo Amor e Carinho que nos une.

A meus pais, Suzel e Alfeu.

Meus irmãos Suze e Julio Cezar.

Ao Sr. Jorge e Dona Carmen.

EPÍGRAFE

ORA E CONFIA

Se um dia te encontrares em situações tão difíceis que a vida te pareça um cárcere sem portas; sob o cerco de perseguidores aparentemente imbatíveis; sofrendo a conspiração de intrigas domésticas; na trama de processos obsessivos; no campo de moléstias consideradas irreversíveis; no laço de paixões que te conturbem a mente; debaixo de provas que te induzam à desolação e ao desânimo; sob a pressão de hábitos infelizes; em extrema penúria, sem trabalho e sem meios de sobrevivência; de alma relegada a supremo abandono; na área de problemas criados pelos entes a que mais ames; não desesperes.

Ora em Silêncio e confia em Deus, esperando pela Divina Providência, porque Deus tem estradas, onde o mundo não tem caminhos.

É por isto que a tempestade pode rugir à noite, mas não existem forças na Terra que impeçam cada dia a chegada de novo amanhecer.

Pelo Espírito Meimei - Do livro: Amizade, Médium: Francisco Cândido Xavier.

ABSTRACT
Master's Degree Dissertation
Posgraduation Program in Engineering of Production
Federal University os Santa Maria, RS, Brazil

UM MODELO DE OTIMIZAÇÃO PARA O PROBLEMA DE DIMENSIONAMENTO E PROGRAMAÇÃO DE LOTES DE PRODUÇÃO EM MÁQUINA ÚNICA

Author: CEZARAUGUSTO GOMES SCALCON
Advisor: FELIPE MARTINS MÜLLER
Date and Place of Defense: Santa Maria July 2, 2012.

In this paper we proposed 0-1 integer programming formulation to model single batch processing machine. This problem deals with a set of jobs with non-identical sizes and processing times that has to be grouped to form batches according to the limited capacity of the machine. The processing time of a batch is the longest processing time of all jobs in the batch. The performance measure is the total time required to process all jobs (makespan). The formulation presented strengthens the model, i.e., it is closer to the optimal formulation than those proposed in the literature. Computational experiments demonstrate that the model is consistent and adequately represents the problem addressed.

Key Words: integer programming; lot-sizing; production planning.

RESUMO
Dissertação de Mestrado
Programa de Pós-Graduação em Engenharia de Produção
Universidade Federal de Santa Maria

UM MODELO DE OTIMIZAÇÃO PARA O PROBLEMA DE DIMENSIONAMENTO E PROGRAMAÇÃO DE LOTES DE PRODUÇÃO EM MÁQUINA ÚNICA

Autor: CEZARAUGUSTO GOMES SCALCON
Orientador: FELIPE MARTINS MÜLLER
Data e Local da Defesa: Santa Maria, dois de Julho 2012.

Neste trabalho é proposta uma formulação de programação inteira 0-1 para modelar o problema de programação e dimensionamento de lotes de produção em máquina única. Este problema considera um conjunto de tarefas com diferentes tamanhos e tempos de processamento que devem ser agrupadas em lotes de acordo com a capacidade limitada da máquina. O tempo de processamento de um lote é determinado pelo maior tempo de processamento dentre todas as tarefas que compõem o lote. A medida de desempenho é o tempo total necessário para processar todas as tarefas (*makespan*). A formulação apresentada é mais forte, ou seja, mais próxima da formulação ideal do que aquelas propostas na literatura. Experimentos computacionais demonstram que o modelo é consistente e representa adequadamente o problema tratado.

Palavras-chave: programação inteira; dimensionamento de lotes; programação da produção.

LISTA DE TABELAS

LISTA DE ABREVIATURAS

PL - Programação Linear
PIB - Programação Inteira Binária
PLI - Programação Linear Inteira
PLP - Programação Linear Pura
PPL - Problema de Programação Linear
PPI - Problema de Programação Inteira
PLIM - Programação Linear Inteira Mista
BPM - *Batch Processing Ma*chine
FFLPT - *First Fit Longest Processing Time*
BFLPT - *Best-Fit Longest Processing Time*
ACO - *Ant Colony Optimization*
PN - Poda do Nó

LISTA DE FIGURAS

SUMÁRIO

1 INTRODUÇÃO

Produzir um número crescente de produtos industriais, a fim de atender a demanda por produtos manufaturados, tem impulsionado o desenvolvimento tecnológico e proporcionado a implementação de processos cada vez mais complexos.

As indústrias envolvidas nestas atividades são de categoria mundial e mantêm suas operações em diversas partes do mundo, produzem em determinado país e montam o produto final em outro, aproveitando o potencial de cada ambiente, o que descentraliza suas atividades e aumenta a complexidade de suas operações. Este aspecto tem feito com que vários países operem em forma de blocos com o objetivo de facilitar o trânsito de produtos, potencializar a capacidade local, reduzir as dificuldades burocráticas de internacionalização e principalmente, conforme Potts e Kovalyov (2000, p. 228), diminuir custos.

Este processo permite que os consumidores possam adquirir produtos e serviços com sofisticada tecnologia a preços significativamente mais acessíveis, massificando o uso de determinados itens, como equipamentos eletrônicos, veículos e outros. Para atender a esta demanda, alguns fabricantes produzem milhões de unidades por ano.

Para suprir esta necessidade, são utilizados processos que executam múltiplas tarefas ou operações simultâneas e fazem parte do sistema produtivo como a mais simples ferramenta manual.

Para que o sistema de produção tenha o desempenho esperado processos auxiliares e de apoio devem estar envolvidos no planejamento da produção. E todos estes processos devem ser orquestrados de modo a garantir o atendimento da demanda. O processo produtivo é comparado por Chase, Jacobs e Aquilano (2008) a um funil, mostrado na Figura 1, no qual se o tempo de espera ultrapassar a capacidade do funil o líquido derrama. O mesmo acontece com um processo real. Se há muitos serviços em um processo, o tempo de espera aumentará.

A classificação tradicional dos sistemas, baseada no fluxo do material, conforme Moreira (2001), pode ser dividido em;

- Sistemas de Produção Contínuo.
- Sistemas de Produção Intermitente.

O Sistema de Produção Contínuo é aquele no qual há uma sequência linear de operações para um único ou poucos produtos.

Podendo ser subdividido em sistemas de produção em massa para linhas de montagem, e o sistema de processamento contínuo para a indústria de processo.

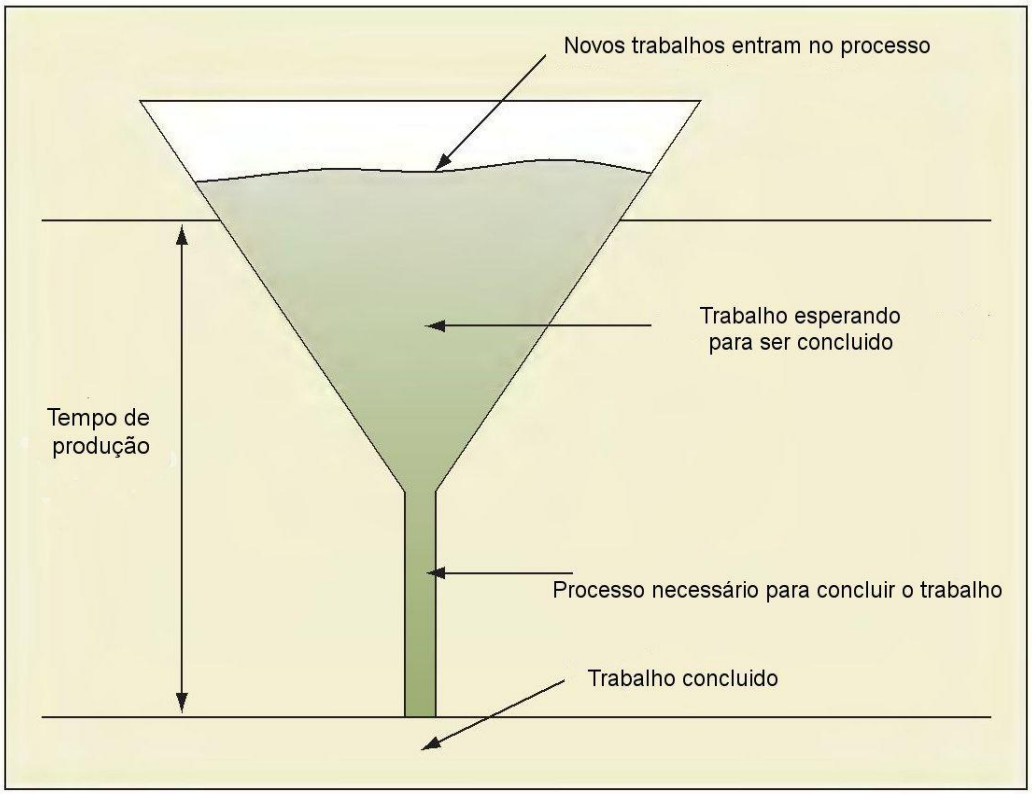


Figura 1 - Representação do processo produtivo.

O Sistema de Produção Intermitente, também pode ser subdividido em dois outros sistemas que são:

i. Sistemas de produção por encomenda, onde o cliente especifica seu produto, que é produzido individualmente.

ii. Sistema de produção em lote, quando consideráveis quantidades de um número relativamente grande de produtos, com certo grau de padronização, é produzido. Na produção em lote, ao término da fabricação de um produto, outros produtos tomam seu lugar nas máquinas, de maneira que o primeiro produto só voltará a ser fabricado depois de algum tempo.

A este respeito Gaither e Frazier (2005) acrescentam que, quando as estratégias de negócios são desenvolvidas para cada linha de produto, as determinações do volume da demanda esperada para cada produto e o número de modelos de produto, necessários para atrair mercado, são um fator importante na escolha do tipo de sistema de produção.

O que valida a afirmativa que o tipo de sistema de produção ou o projeto de processo apropriado depende do número de produtos e do tamanho dos lotes a serem produzidos num sistema de produção.

A Figura 2, como descrevem Chase, Jacobs e Aquilano (2008), mostra um esquema que relaciona o tamanho do lote com o número de produtos que devem ser produzidos. Os sistemas produtivos no ponto A são aqueles de produção em massa e que têm fluxo continuo e precisam ser operados 24 horas por dia, com água, petróleo, gás e energia elétrica. O ponto B são operações em uma linha relativamente estável de produtos, cada uma das quais sendo produzidas em lotes periódicos, que podem ser a produção de produtos químicos, componentes eletrônicos, circuitos integrados e processadores.

O sistema do ponto C é aquele em que os componentes andam de estação em estação de trabalho, seguindo uma sequência necessária para a fabricação do produto, são exemplos a montagem manual de componentes, que podem ser brinquedos e eletrodomésticos.

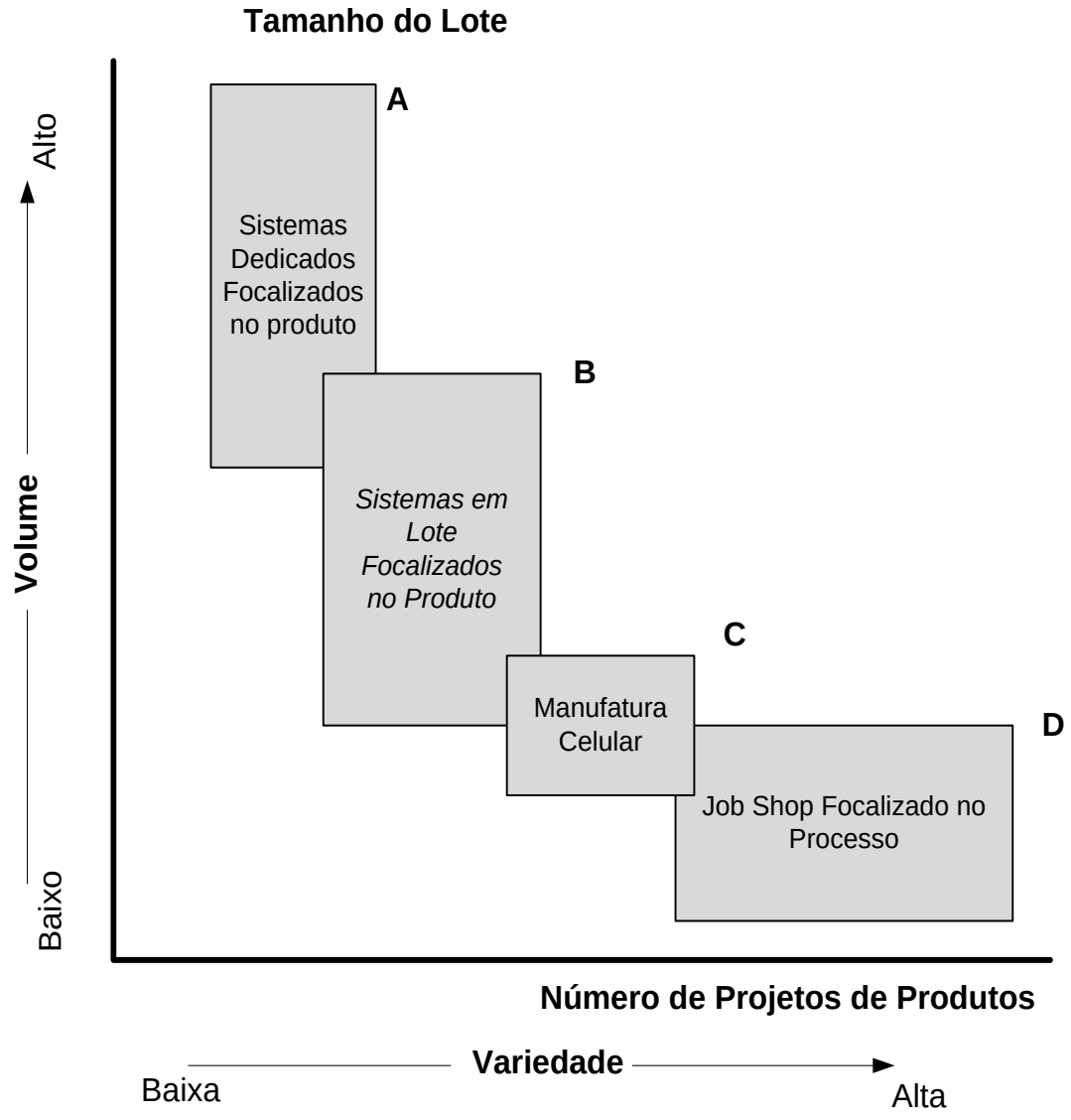


Figura 2 - Processo: dependente do produto e do tamanho do lote.

No ponto D a produção é de pequenos lotes, que abrangem um grande número de produtos, onde a maioria necessita de um conjunto ou uma sequência diferente de etapas de processamento, como oficinas, fábricas e gráficas.

Os sistemas de produção em lote são largamente utilizados, vários métodos e técnicas são empregados a fim de aperfeiçoar este sistema de produção.

No nível operacional o mesmo ocorre, uma vez que máquinas em linhas de produção que processam várias tarefas em lote promovem uma significativa vantagem em relação a outros sistemas, quando aplicados adequadamente. Existe a necessidade de programar de forma correta a produção dessas máquinas, uma vez que invariavelmente representam gargalos na produção. Neste caso especifico, o que se busca não é otimizar um sistema completo, mas sim em uma etapa do processo ou mesmo uma máquina do processo.

Dobson e Nambimadom (1992) estudaram o caso de dimensionamento e programação de lotes em máquina única com minimização do tempo total de conclusão ponderado com o critério de otimização. A definição do problema abordado considera que as tarefas são agrupadas em famílias incompatíveis, ou seja, duas tarefas de famílias diferentes não podem compor um mesmo lote. Azizoglu e Webster (2001) tratam o caso em que todas as tarefas são compatíveis.

A Figura 3 apresenta uma representação esquemática do problema de dimensionamento e programação de lotes de produção em máquina única. Em cada lote diversas tarefas, com diferentes tamanhos, são processadas. O tempo de processamento de um lote é dado pelo maior tempo de processamento das tarefas que compõem o lote, esse tempo é conhecido em língua inglesa por *makespan.* Um critério comum de otimização é a minimização do tempo máximo de conclusão das tarefas, o que significa minimizar a soma do *makespan* de todos os lotes designados à máquina.

Esse problema é bem representado na prática no processo de fabricação de circuitos integrados e processadores. Na fabricação destes componentes existe uma operação chamada *burn-in* (teste de confiabilidade) que é realizado em câmaras térmicas, onde estes circuitos são submetidos a uma temperatura constante de aproximadamente 120ºC, durante um tempo determinado, conforme descrição de Parsa, Karimi e Kashan (2009, p. 1722).

Estes componentes são submetidos a testes em lotes, em cada máquina térmica, que dependendo do fabricante podem ser mais ou menos complexos. Esta complexidade está relacionada à confiabilidade exigida do produto, que para muitas empresas agrega valor ao produto final, pois garante produtos de qualidade superior.

Uma programação eficaz dessas operações é de grande importância para o desempenho geral de uma empresa. Na operação de *burn-in*, diferentes tipos de circuito integrado são carregados em placas e, em seguida, colocados em uma máquina térmica para teste, conforme Uzsoy (1994, p. 1615)

A operação de *burn-in*, como descrevem Sung e Choung (1998, p. 559), é um procedimento relativamente demorado para os padrões de fabricação destes produtos, os lotes a serem testados devem ser perfeitamente escolhidos. Xu e Bean (2007, p. 143) esclarecem que os tempos de processamento das operações de *burn-in* são, geralmente, extremamente longos se comparados como os outros testes de operação, por exemplo, 120h x 4-5h.

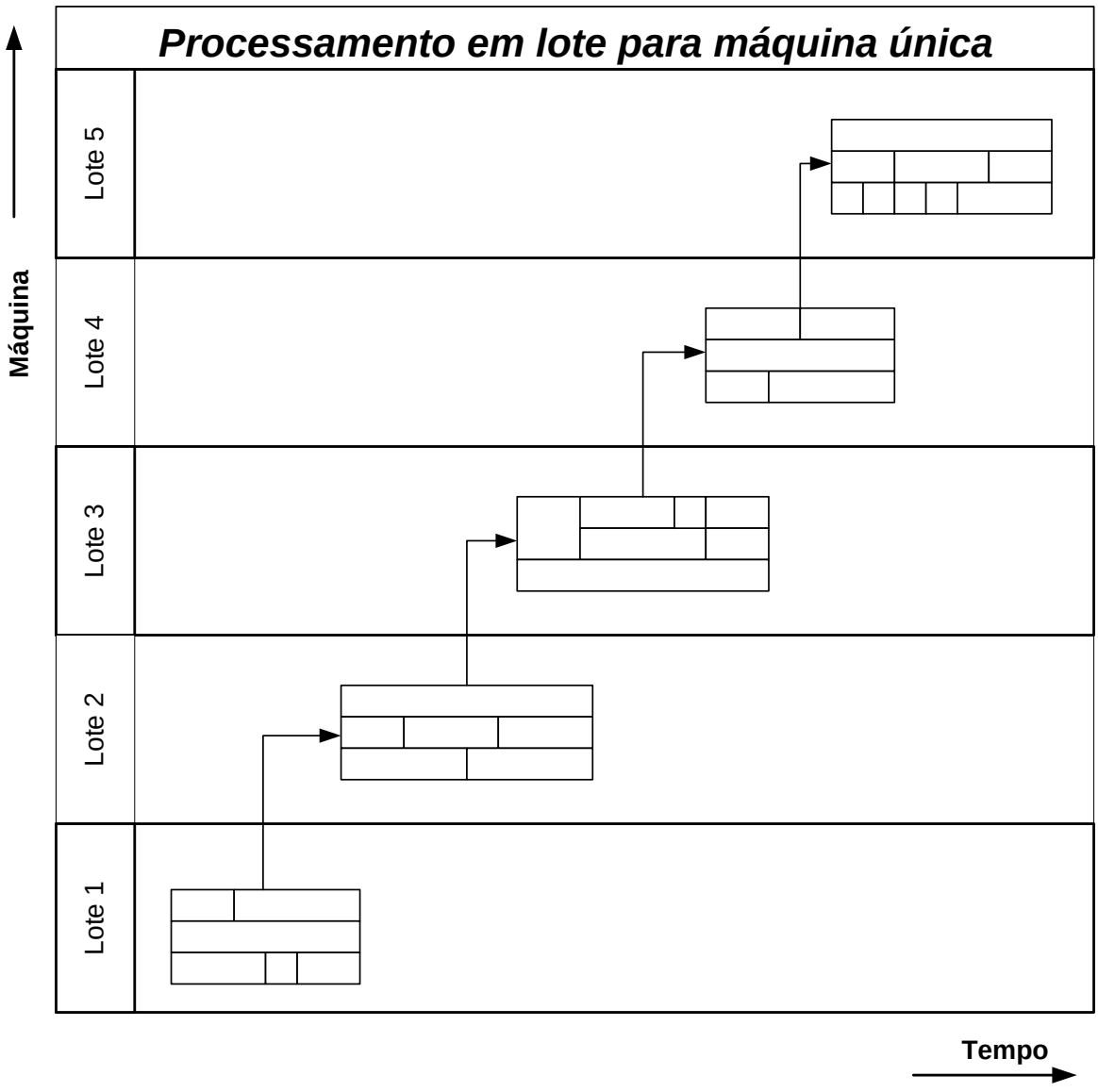


Figura 3 - Produção em lote para máquina única.

A câmara ou máquina térmica, que aplica o teste de *burn-in*, é mantida em temperatura elevada e constante, durante um período de tempo. O tempo de teste e a temperatura para cada circuito integrado são determinados pela especificação do produto e são conhecidos *a priori*. A ideia para o teste de *burn-in* é expor o circuito integrado a um *stress* térmico, de modo que qualquer destes componentes em teste que apresente defeito latente possa ser identificado.

O teste de estresse térmico simula as condições de operação a que um componente estaria submetido em condições normais de trabalho, e muitos dos defeitos, que só seriam perceptíveis no uso comercial, manifestam-se na operação de *burn-in*, permitindo uma seleção adequada dos componentes defeituosos.

Posteriormente o componente é submetido a um teste funcional rápido, que testa o componente e descarta aqueles que não oferecem condições comerciais. Os componentes que apresentam condições de utilização são então submetidos a um teste funcional completo. A capacidade de um equipamento de *burn-in* é limitada. Sung e Choung (1998, p. 560) e Parsa (2009, p. 1721) descrevem que agrupar em lotes os componentes a serem testados é essencial, para que todo o espaço útil das máquinas térmicas seja utilizado.

Um circuito integrado pode ser mantido na máquina mais do que o tempo pré-especificado de *burn-in*, mas não pode ser retirado da máquina antes que o tempo pré-determinado de teste tenha decorrido. Uma vez que o processamento de um lote inicia, não pode ser inserida nenhuma nova tarefa. Fica claro, nessa aplicação, que o tempo de processamento de um lote é determinado pelo tempo de processamento mais longo das tarefas contidas no lote.

O teste de *burn-in* frequentemente é o gargalo da fabricação, pois ocorre no final do processo e tem papel vital no prazo de entrega dos produtos acabados. Um processo eficaz, dessas operações, é de grande importância para o desempenho global de uma empresa, segundo Sung e Choung (1998, p. 559).

É importante observar que diversos componentes exigem tempo mínimo diferente de processamento (*burn-in*), assim a programação da produção destes lotes é não trivial, como argumenta Ghazvini (1998, p. 275), o que pode afetar diretamente a taxa de produção. De fato, programar a produção de uma grande quantidade de componentes que recebem muitas operações durante o processo de produção, segundo Sung e Choung (1998, p. 560), é uma tarefa extremamente difícil, praticamente impossível de ser realizada de forma eficiente se não forem considerados métodos computacionais de otimização.

Uma programação eficaz dessas operações é de grande importância para o desempenho geral de uma empresa. Conforme descrito anteriormente, na operação de *burn-in*, diferentes tipos de circuito integrado são carregados em placas e, em seguida, colocados em uma máquina térmica para teste, Figura 4.

Neste trabalho são explorados os aspectos relacionados à otimização de processos relativos ao dimensionamento e programação de lotes de produção em máquina única. Especificamente, são propostas formulações para modelos de programação inteira 0-1. Na definição do problema é considerado um conjunto de tarefas com diferentes tamanhos e tempos de processamento que devem ser agrupadas em lotes de acordo com a capacidade limitada da máquina.

A nova formulação incorpora restrições para quebra de simetria e um procedimento de pré-processamento que ordena as tarefas segundo o tempo de processamento. Em comparação ao modelo que usualmente aparece na literatura cientifica, os resultados apresentados são melhores. O que comprova que a formulação proposta é mais forte no sentido que a relaxação contínua do modelo considerado fornece melhores limitantes e não altera a solução ótima do problema original.

Figura 4 - Equipamento de *burn-in.*

A apresentação deste trabalho é dividida da seguinte forma;

A primeira seção descreve a introdução e postula os parâmetros gerais do trabalho; na segunda é feita a revisão bibliográfica. A terceira seção trata do problema de programação e dimensionamento de lotes de produção em máquina única e as formulações propostas são descritas. A quarta seção apresenta a metodologia empregada para gerar o conjunto de instâncias para teste e os resultados computacionais comparativos. Na quinta seção constam as conclusões e, na sexta seção as referências bibliográficas utilizadas. A sétima seção traz os apêndices que incluem gráficos e análises adicionais.

2 REVISÃO DE BIBLIOGRÁFICA

A Programação Matemática, nas observações de Goldbarg e Luna (2005), é fortemente direcionada ao apoio à tomada de decisão, principalmente, no gerenciamento de sistemas de grande porte. Esta técnica permite a modelagem de inter-relações de variáveis que dificilmente seriam visíveis de forma intuitiva. Nos modelos matemáticos, conforme Damodaran, Manjeshwar e Srihari (2006, p. 883), a solução de determinado sistema é geralmente realizada por um conjunto de equações ou expressões matemáticas.

Se existem n decisões quantificáveis a serem tomadas, pode-se associar a cada decisão uma variável do modelo matemático, que usualmente é denominada variável de decisão, cujos valores o próprio modelo deve determinar.

Diferente de uma análise qualitativa, esta metodologia objetiva determinar grandezas mensuráveis e a análise para se chegar ao fim desejado pode ser descrita sucintamente conforme o fluxo da Figura 5.

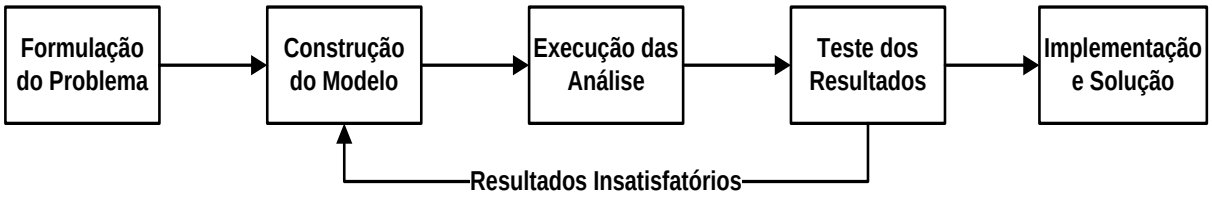


Figura 5 - Fluxo da análise quantitativa

Modelos de grandes dimensões são análogos, segundo Hillier e Lieberman (2010), sob certos aspectos, ao desenvolvimento de um programa de computador muito extenso. Quando a primeira versão do programa de computador estiver terminada, ele inevitavelmente conterá muitos defeitos - *bugs*. Finalmente, após uma longa sucessão de programas mais aperfeiçoados, o programador (ou equipe de programação) concluirá que o programa atual, de forma geral, está dando resultados razoavelmente válidos. Embora alguns pequenos *bugs* indubitavelmente ainda permaneçam ocultos no programa (e talvez jamais venham a ser detectados), os principais *bugs* foram suficientemente eliminados de maneira que o programa possa ser usado de forma confiável.

De modo semelhante, a primeira versão de um modelo matemático de grandes dimensões contém falhas. Alguns fatores ou inter-relacionamentos relevantes, sem dúvida, não foram incorporados ao modelo e alguns parâmetros não foram estimados corretamente. Isso é inevitável dada à dificuldade de comunicação e de compreensão de todos os aspectos e

sutilezas de um problema operacional complexo, bem como em razão da dificuldade em coletar dados confiáveis. O processo de teste e aperfeiçoamento de um modelo para aumentar sua validade é comumente referido como validação do modelo.

Um número considerável de autores tem utilizado programação matemática para estudar o problema de minimizar o tempo de processamento total em uma máquina de processamento em lote único ou até mesmo de máquinas paralelas. Autores como Dupont e Dhaenens (2000, p. 807), Xu, Chen e Li (2011, p. 582), Ghazvini (1998, p. 274), Parsa (2009, p. 1721) e Damodaram (2006, p. 882) adotam métodos para buscar a solução de problemas de programação utilizando, principalmente, a técnica de enumeração *Branch-and-bound* e uma de suas várias adaptações e estratégias de implementação.

No trabalho de Chen, Du e Huang (2011, p. 5755), intitulado *Scheduling a batch processing machine with non-identical job sizes: a clustering perspective*, os autores tratam do problema de minimizar o tempo total de produção de máquinas de processamento em lote (BPM – Batch Processing Machines) e, a partir de uma motivação baseada no processo prático a operação de *burn-in*, propõem um método de resolução sob a perspectiva de uma heurística de clusterização ou agrupamento.

Neste ponto é importante observar que a resolução de problemas com o uso de computadores é classificada de acordo com seu grau de dificuldade. A teoria que trata desta questão é denominada Teoria da Complexidade Computacional e, grosso modo, procura classificar os problemas em fáceis ou difíceis (Hartmanis e Stearns, 1965). É de fácil aceitação que existem cálculos que são impraticáveis para serem levados a cabo por um ser humano. O estudo nesse caso recai na discussão se existem problemas que sejam difíceis para computadores ideais resolverem, ou seja, computadores teóricos que transcedam questões tecnológicas e considerem tão somente limites físicos da natureza. Desta forma, um problema pode ser considerado naturalmente difícil se para sua solução forem necessários recursos computacionais significativos, independente do tipo de algoritmo utilizado. Para quantificar os recursos necessários à resolução destes problemas, um meio, dentre outros, para quantificar a dificuldade, é o tempo de processamento.

A composição de um problema computacional consiste em instâncias e a solução para estas instâncias. As instâncias são os elementos que compõem um determinado problema e um problema computacional pode ter infinitas instâncias, já as soluções podem ser determinadas como um resultado único, tipo sim ou não ou (0-1).

Para medir a dificuldade de resolver um problema computacional, pode-se desejar ver quanto tempo o melhor algoritmo necessita para resolver o problema. No entanto, o tempo

de execução pode, em geral, depender da instância. Em particular, instâncias maiores exigirão mais tempo para resolver. Assim, o tempo necessário para resolver um problema é calculado em função do tamanho da instância.

A pergunta primordial, neste caso, é se existe um problema cuja resposta possa ser obtida em tempo polinomial em função de parâmetros relativos à instância. Caso afirmativo, o problema é dito de fácil resolução e que pertence à classe P. Caso não se conheça um algoritmo para resolver um dado problema em tempo polinomial, então este problema é considerado difícil e pertence à classe NP, acrônimo em inglês para tempo polinomial não determinístico – *Non-Deterministic Polynomial time* (Garey e Johnson,1979).

Considerando problemas combinatórios, uma ideia simplista para encontrar a solução ótima corresponde à enumeração completa de todas as soluções. A partir desta enumeração basta comparar o valor atribuído a cada solução e selecionar a melhor.

A dificuldade ou complexidade destes problemas pode ser vislumbrada com um exemplo relacionado a problemas de ordenação de n entidades (lotes, peças, produtos ou componentes), no qual o número de soluções possíveis é $n!$. O caso mais complexo pode ser o de uma fábrica com m máquinas e n entidades e o número de soluções possíveis neste caso é de $(n!)^m$. A Tabela 1 mostra de que forma varia o número possível de sequências com a variação do número de entidades e o número de máquinas em uma fábrica que pode processar componentes em lote.

Tabela 1 – Complexidade; número de soluções.

Número de entidades n	Número de máquinas m	Número de soluções
5	1	120
5	3	1,7 milhões
5	5	25.000 milhões
10	10	$3{,}96^{10^{65}}$

Como se pode observar pela Tabela 1, o número de sequências cresce para números extremamente elevados, mesmo para problemas relativamente pequenos de programação da produção.

Apenas para dar a ideia da complexidade de tais problemas, se um computador demorar 0,000001 segundos para avaliar cada solução, para o caso n = 10 e m = 10 na Tabela 1 seriam necessários $3.96 * 10^{59}$ segundos, ou seja $1{,}26 * 10^{35}$ milhões de bilhões de séculos

para avaliar todas as soluções. Para esses problemas não se conhece até o presente momento se existem algoritmos eficientes de resolução, ou seja, pertencentes à classe P.

Segundo Uzsoy (1994 *apud* Parsa, 2009, p. 1721), problemas de programação em lote com tarefas de diferentes tamanhos em máquina única ou máquinas paralelas são considerados NP - difícil ou NP - *hard*. Li e Lee (1997 *apud* Dupont e Dhaenens, 2000, p. 808) demonstram que este tipo de problema é fortemente NP – *hard*, assim como Meng e Tang (2010, p. 1703).

Em razão da dificuldade da programação de produção em lote é que muitas pesquisas têm sido feitas sobre este caso. Sung e Choung (1998, p. 561) relatam que estas pesquisas podem ser divididas em duas categorias de processamento.

A primeira é que o tempo de processamento de cada lote é fixo e independente do emprego do lote, a segunda é que o tempo de processamento é dependente dos atributos de cada lote. Na indústria de semicondutores, a primeira categoria, onde o tempo de processamento de cada lote é fixo, pode ser aplicado principalmente na fabricação do *wafer* – elemento utilizado na fabricação de processadores, memórias e circuitos integrados. Ver Apêndice A.

A segunda categoria, onde o tempo de processamento depende de cada lote corresponde ao tempo de processamento da maior tarefa, é especialmente aplicada na operação de *burn-in*.

2.1 PROBLEMAS DE PROGRAMAÇÃO DE TAREFAS EM LOTE

A formulação inicial do problema, para uma única máquina, tem a seguinte proposta:

Uma função objetivo para minimizar o problema. Um conjunto de restrições para garantir que a cada tarefa é atribuído apenas um lote. Um conjunto de restrições para garantir que a capacidade da máquina não é ultrapassada, quando são atribuídas tarefas ao lote. Um conjunto de restrições para determinar o tempo de processamento do lote em estudo e um conjunto de restrições que especifica as variáveis de decisão. Ver Apêndice A.

Dupont e Dhaenens (2000, p. 808), em seu trabalho, consideram o problema de se minimizar o tempo total de conclusão das tarefas de uma máquina de processamento em lote.

E para isto adotam as seguintes hipóteses de trabalho;

i. A existência de um conjunto $N = \{1,...,n\}$ de tarefas que devem ser processadas. Para cada tarefa é definido o tempo de processamento mínimo exigido, denotado por p_i. É assumido que cada tarefa i está disponível no tempo $r_i = 0$.

ii. A máquina tem uma capacidade B e cada tarefa tem um tamanho s_i. A soma dos tamanhos das tarefas contidas num lote não podem exceder B. Por definição, o tamanho de uma tarefa não pode exceder a capacidade da máquina, ou seja, $s_i \leq B$ para toda tarefa i.

iii. Uma vez que o processamento do lote é iniciado ele não pode ser interrompido e nem outras tarefas podem ser introduzidas na máquina até que o processamento seja concluído. O tempo de processamento de um lote é dado pelo maior tempo de processamento de uma tarefa contida no lote.

iv. O objetivo é minimizar o tempo máximo de conclusão das tarefas ou *makespan* (C_{max}).

Conjuntamente com estas hipóteses é utilizado um algoritmo *Branch-and-bound.* Para calcular o valor inicial são utilizadas as heurísticas FFLPT - *First Fit Longest Processing Time* e BFLPT - Best-Fit Longest Processing Time.

Melouk, Damodaran e Chang (2003, p. 143) propõem a utilização de um algoritmo *simulated annealing,* onde uma solução inicial é escolhida de forma aleatorizada e em seguida uma solução vizinha é gerada. Para os testes computacionais, instâncias aleatórias foram

geradas e a abordagem testada foi comparada com o pacote comercial CPLEX, com o modelo que consta no Apêndice C.

Diversos autores têm utilizado as mais variadas abordagens. Sung e Choung (1998, p. 567) exploram, para a minimização do *makespan*, duas alternativas, que foram chamadas de estática e dinâmicas. A abordagem estática explorou a solução com tamanho de tarefas iguais, e a dinâmica o algoritmo *Branch-and-bound* com algumas outras heurísticas, sendo que a abordagem estática foi apresentada mais como uma forma de embasar o trabalho. Chandru et al (1993 *apud* Sung e Choung, 1998, p. 571) também abordam de forma semelhante o problema do *makespan* em duas alternativas.

Dupont e Dhaenens (2000, p. 815) utilizam o método *Branch-and-bound* em uma máquina de processamento em lote único. Graças às propriedades de dominância do método e a enumeração reduzida, instâncias com um grande número de tarefas foram utilizadas. Este trabalho mostra que quanto maior a tarefa mais difícil é a solução do problema, mas mesmo para estes problemas a enumeração utilizada como um método heurístico foi capaz de obter soluções de muito boa qualidade. O que demonstra que a utilização de métodos exatos pode ter aplicações industriais muito importantes e, principalmente, para agrupar em lotes uma linha de produção de produtos diferentes.

Xu e Bean (2007, p. 145) utilizam um algoritmo genético para desenvolver o problema de programação de lotes em máquina única com critério de otimização *makespan.* Os autores sugerem que uma atenção especial é requerida a fim de evitar que uma representação inadequada seja proposta, não levando aos resultados esperados e criando um problema de inviabilidade. Cada tarefa é codificada como um gene, assim um cromossomo para n tarefas contém m genes, para obter o gene de cada tarefa gera-se um número aleatório inteiro em $\{1, \dots, m\}$ e adiciona-se uma variável $(0,1)$ aleatória uniforme. Este número serve como a chave aleatória. No mapeamento, a parte inteira da chave aleatória é definida como a atribuição da máquina para cada tarefa e a parte fracionária de todas as chaves é classificada para fornecer a sequência de trabalho em cada máquina. Para cada máquina a formação dos lotes é dada a partir da sequência de trabalho de tal forma que o tamanho total do lote é igual ou inferior à capacidade da máquina.

Domadaran (2006, p.884) também utiliza um algoritmo genético para a minimização do *makespan* mediante a aplicação de operadores para cruzamento e mutação com dois pais escolhidos aleatoriamente.

Parsa (2009, p. 1722) utiliza a decomposição *Dantzig-Wolfe* para formular sua proposta, e um algoritmo *Branch-and-Price*, afirmando que os resultados apresentados são bons.

Meng e Tang (2010, p. 1703) utilizam como método para minimizar o *makespan*, uma heurística de busca tabu, que tem como um de seus principais componentes o uso de memória adaptativa para criar uma busca mais flexível. Meng e Tang também fazem uso de uma heurística gulosa, com base em programação dinâmica, que de acordo com o problema executa buscas na vizinhança para obtenção de uma solução inicial com a finalidade de minimizar o número de lotes formados. Além da busca nas vizinhanças, uma busca em profundidade em até três níveis é realizada, de forma dinâmica, procurando melhorar o resultado. Posteriormente a heurística de busca tabu é introduzida com o objetivo de melhorar o resultado obtido, o que segundo os autores resultou em excelentes resultados.

No trabalho proposto por Chen, Du e Huang (2011, p. 5755), os autores estudam um modelo de clusterização ou agrupamento para programar uma máquina de processamento em lote único com tamanhos não idênticos de tarefas classificadas em três níveis. O problema acima pode ser indicado $1 \mid batch,\ s^j \leq C \mid C_{máx.}$ Onde se pode obter a seguinte formulação;

Minimizar

$$\text{Min } C_{\max} = \sum_{b \in B} p^b \tag{a}$$

s.a.

$$\sum_{b \in B} x_{jb} = 1 \quad \forall j \in J \tag{b}$$

$$\sum_{j \in J} s_j x_{jb} \leq C \quad \forall b \in B \tag{c}$$

$$p^b \geq p_j x_{jb} \quad \forall j \in J, b \in B \tag{d}$$

$$x_{jb} \in \{0,1\} \quad \forall j \in J, b \in B \tag{e}$$

$$\left\lceil \sum s_j \,/\, C \right\rceil \leq |B| \leq n \tag{f}$$

A função objetivo (a) é minimizar o *makespan*. A restrição (b) garante que uma tarefa *j* é atribuída a apenas um lote. A restrição (c) assegura que o tamanho total do lote não pode exceder a capacidade da máquina. A restrição (d) determina o tempo de processamento para o lote *b*. A restrição (e) denota a restrição do binário da variável x_{jb}. A última restrição (f) dá o número mínimo e o máximo de lotes necessários para agrupar todas as tarefas, onde o símbolo $\lceil B \rceil$ denota o número de elementos no conjunto *B* e o elemento $\lceil x \rceil$, o menor número inteiro maior do que ou igual *x*.

O problema descrito anteriormente consiste em duas tarefas. A primeira tarefa é a de formar lotes de tal forma que a restrição de capacidade da máquina não é violada, isto é, o problema de lotes. A segunda tarefa é determinar uma sequência na qual os lotes serão processados, ou seja, o problema de programação.

Xu, Chen e Li (2011, p. 582), investigam o problema de processamento de máquina em lote único, onde esta máquina pode processar diversas tarefas simultaneamente. Os autores exploram o problema ACO - *Ant Colony Optimization*, onde os seguintes argumentos são utilizados;

i. Existem *n* tarefas a serem processadas. Cada tarefa *j* tem um tempo de liberação, r_j; um tempo de processamento, p_j; e um tamanho correspondente, $s_{j.}$
ii. A máquina de processamento em lote tem uma capacidade de *B*, e a soma dos tamanhos das tarefas, de um lote, deve ser menor do que ou igual a *B*. Admite-se que nenhuma tarefa tem um tamanho superior a *B* e todas as tarefas são compatíveis para serem agrupadas em um lote.
iii. Uma vez que o processamento de um lote é iniciado, ele não pode ser interrompido, e outros trabalhos não podem ser introduzidos na máquina até o processamento ser concluído.
iv. O número de lotes *k* é igual ao número de tarefas *n*. Um lote é aberto, se houver pelo menos uma tarefa atribuida a um lote. Em contraste, um lote é fechado sem qualquer tarefa atribuída, quando o lote *b* for $P^b = 0$.
v. Seu critério objetivo é minimizar o makespan.

2.2 OTIMIZAÇÃO DISCRETA

A formulação de um problema de Programação Linear – PL pode levar a uma solução ótima em que o resultado seja um número fracionário. Nem sempre este tipo de resultado atende a uma determinada necessidade quando o resultado deve envolver uma solução inteira.

O que caracteriza um problema de otimização discreta é exatamente este ponto; a solução do problema deve contemplar determinadas variáveis de decisão que devem assumir apenas valores "inteiros".

Como exemplos podem ser citados os seguintes casos de otimização discreta ou programação inteira;

i. Dimensionamento e programação de lotes. Consiste na divisão de um macro período em vários micro períodos. A hipótese deste problema é de que somente um item é produzido por período e, se produzido, utiliza a capacidade total.

ii. Os casos de otimização discreta que trabalham com problemas de decisão de sequências, programas e itinerários, têm como exemplo clássico o problema do caixeiro viajante, onde para n cidades existem $(n-1)!$ diferentes percursos. Também os problemas de programação de máquinas onde para n itens a serem produzidos em cada uma de k máquinas existem $(n!)^k$ sequências possíveis.

Os modelos de programação inteira têm restrições e uma função objetivo idêntica àquelas formuladas pela PL. com a seguinte forma:

$$z = max\, cx + dy$$
$$Ax + Dy \leq b$$
$$x \in R^n_{+,} \quad y \in Z^p_+$$

A única diferença é que uma ou mais das variáveis de decisão precisam ter um valor inteiro no final da solução. Render, Hanna e Stair (2010) descrevem os três tipos de problemas de programação inteira da seguinte forma:

1. Problemas de programação inteira pura são casos em que todas as variáveis devem ter valores inteiros.

2. Problemas de programação inteira mista são os casos em que algumas, mas não todas, variáveis de decisão devem ter valores inteiros, tratada como Programação Inteira Mista – PIM.
3. Problema de programação inteira 0 - 1 são casos especiais em que todas as variáveis de decisão devem ter valores inteiros de 0 a 1 como solução. Também conhecida como Programação Inteira Binária – PIB.

A resolução de um problema de programação inteira é muito mais difícil do que resolver um problema de PL, conforme Hillier e Lieberman (2010). Dupont e Dhaenens (2000, p. 808) observam que muitas vezes este problema de Programação Linear Inteira deve ser relaxado, são ignoradas as condições de integralidade e o problema é resolvido como um problema de PL padrão. Em razão desta dificuldade é necessária muita atenção na aplicação desta rotina, tendo em vista que para números muito pequenos não é possível arredondar a variável não inteira ou também definir em que sentido deve-se fazer o arredondamento para que a viabilidade seja mantida. Portanto, esta regra não é um procedimento robusto para solucionar problemas de programação inteira. Uma diferença ainda mais acentuada da programação inteira com relação à PL é de que não existe procedimento suficientemente seguro para afirmar que uma solução viável é ótima.

Já a modelagem com variáveis binárias é especialmente aplicada, para problemas de otimização, quando um evento deve ser avaliado na sua ocorrência ou não, e nos casos de decisão entre duas alternativas. De acordo com Arenales et al (2008) esta dicotomia é modelada por uma variável binária x tal que

$$x = \begin{cases} 1 \text{ se o evento ocorre} \\ 0 \text{ se o evento não ocorre} \end{cases}$$

As variáveis binárias são também reconhecidamente utilizadas e importantes na literatura para definição de modelos de lotes como o descrito por Domadaran (2006, p. 884).

Existem diversas técnicas desenvolvidas para buscar a solução de problemas de programação inteira, segundo Goldbarg e Luna (2005) que podem ser:

Técnicas de Enumeração:

→ Separação e avaliação progressiva ou *Branch-and-bound* .

→ Enumeração implícita.

→ Restrições *surrogate*.

Técnicas de Corte:

→ Cortes inteiros (primais e duais).

→ Cortes combinatóriais.

→ Cortes de intersecção.

→ Métodos de decomposição de Benders.

Técnicas Híbridas:

→ *Branch and Cut*.

→ Teoria de grupo.

Na maioria das ocasiões as técnicas de solução são especializadas para cada tipo de problema de programação inteira.

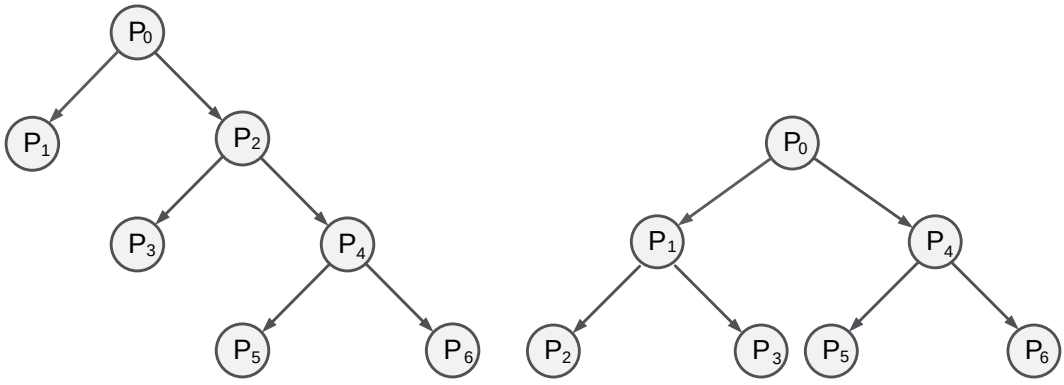
Um dos problemas clássicos da Otimização Combinatória é o problema do Caixeiro Viajante, onde para n cidades existem $(n-1)!$ diferentes percursos.

Será apresentada resumidamente a teoria que suporta a técnica *Branch-and-bound*.

Esta última estratégia pode ser aplicada, de forma auxiliar, utilizando o método de *Branch-and-bound* , que consta de duas operações básicas: bifurcação ("*Branch*") e limitação ("*bound*").

O método denominado *Branch-and-bound* é definido por Goldbarg e Luna (2005) como uma ideia que desenvolve uma enumeração inteligente dos pontos candidatos à solução ótima inteira de um problema. O termo *Branch* refere-se ao fato de que o método efetua partições no espaço de soluções (bifurcação). O termo *Bound* ressalta que a prova da otimalidade da solução utiliza-se de limites calculados ao longo da enumeração (limitação).

A técnica do *Branch-and-bound* tem uma excelente qualidade do limite gerado pela solução inteira. Goldbarg e Luna complementam que a qualidade do limite alcançado normalmente depende, para cada problema, da estratégia de ramificação da árvore de busca. Existem basicamente duas estratégias de divisão ou *Branch*. A Figura 7 apresenta o aspecto das estratégias das árvores desenvolvidas pela busca em profundidade e pela busca em largura.



Busca em Profundidade Busca em Largura

Figura 6 - Estratégias de ramificação da árvore *Branch-and-bound*

Existem várias adaptações e estratégias de implementação da técnica *Branch-and-bound* que podem ser, dentre outras:

Técnica de desenvolvimento de busca;

→ Busca em profundidade

→ Busca em largura

→ Variante híbrida

Técnicas de formação de busca;

→ Método das Penalidades

→ Estratégias dinâmicas

→ Outras variantes como a proposta por Parsa (2009, p. 1723)

Técnicas para obtenção dos limites;

→ Relaxação linear

→ Relaxação lagrangeana

→ Algoritmos heurísticos e metaheurísticos

→ Cortes

As técnicas de solução são especializadas para os diversos tipos de problemas de programação inteira, desenvolvendo-se abordagens e algoritmos específicos para cada situação.

O exemplo a seguir demonstra a técnica *Branch-and-bound* .

Seja;

Máx. $21x_1 + 11x_2$

sujeito a $7x_1 + 4x_2 \leq 13$

$x_1, x_2 \geq 0$

x_1, x_2 $inteiros$

A Figura 8 demonstra os pontos de solução factíveis do problema, onde o Problema de PL apresentado é o Problema de Programação Inteira – PPI, desconsiderando as restrições de integralidade das variáveis inteiras, que é a relaxação do PPI.

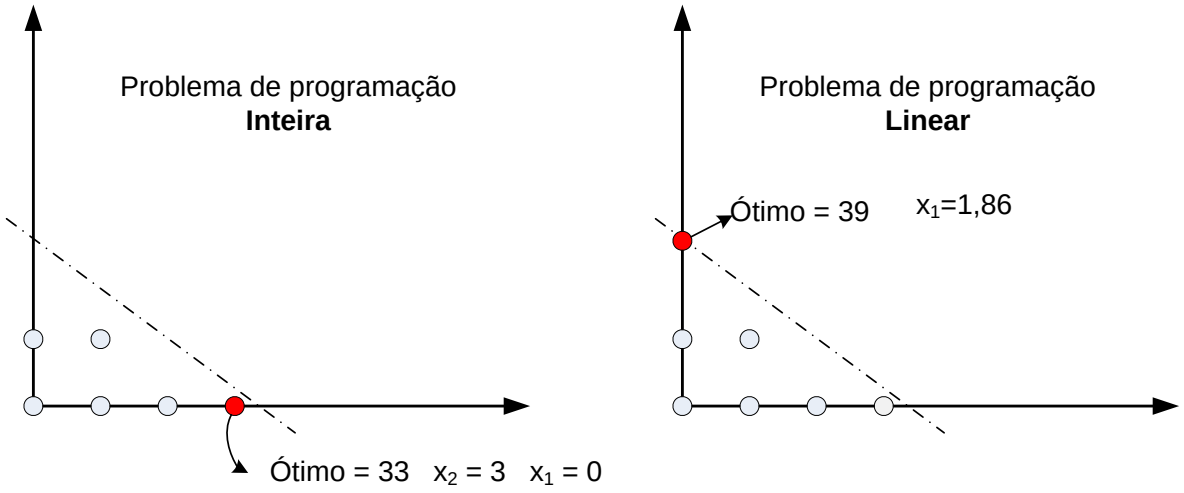


Figura 7 - Soluções Factíveis

Como podemos observar, a solução do Problema de Programação Linear - PPL é sempre maior ou igual à solução do PPI, pois o problema relaxado é composto por todas as soluções inteiras e também as soluções reais do problema, logo são formadas por um conjunto de soluções factível mais abrangente.

Se a solução do Problema de Programação Inteira relaxado corresponde a uma solução do Problema de Programação Linear, quando possui todas as variáveis inteiras, então a solução encontrada é a solução ótima do Problema de Programação Inteira.

Assim temos que, para um problema de maximização $Z_{PPL} \geq Z_{PPI}$, ou seja, a solução ótima da relaxação linear de um problema inteiro (Z_{PPL}) é sempre maior ou igual à solução ótima do problema inteiro (Z_{PPI}).

A divisão do problema é interrompida quando uma das condições a seguir é satisfeita. Essas condições são chamadas de testes de sondagem ou Poda do Nó - PN.

i. **PN1** ou poda por infactibilidade. O problema relaxado é infactível.
ii. **PN2** ou poda por otimalidade. A solução ótima do problema relaxado é inteira.
iii. **PN3** ou poda por qualidade. O valor de qualquer solução factível do problema relaxado é pior que o valor da melhor solução atual (solução incumbente).

Quando uma dessas três condições ocorre, o subproblema pode ser descartado (sondado ou podado), pois todas as suas soluções factíveis estão implicitamente enumeradas.

Resolvendo o problema relaxado tem-se o nó inicial, na Figura 9:

a) Valor ótimo da solução: 39

b) Valor das variáveis: $x_1 = 1{,}86 \quad x_2 = 0$

$Z = 39$
$x_1 = 1{,}86$
$x_2 = 0$

Figura 8 - Nó Inicial

Logo o valor de x_1 não é inteiro, então dividimos o problema em dois subproblemas, Tabela 2:

i. um onde considerou o valor de $x_1 \geq 2$ que vamos chamar de subproblema A.
ii. outro considerou $x_1 \leq 1$ chamado de subproblema B.

Tabela 2 – Subproblema A - B

Subproblema A	Subproblema B
$\max\ 21x_1 + 11x_2$ $sujeito\ a\ \ 7x_1 + 4x_2 \leq 13$ $\boldsymbol{x_1 \geq 2}$ $x_1, x_2 \geq 0$	$\max\ 21x_1 + 11x_2$ $sujeito\ a\ \ 7x_1 + 4x_2 \leq 13$ $\boldsymbol{x_1 \leq 1}$ $x_1, x_2 \geq 0$

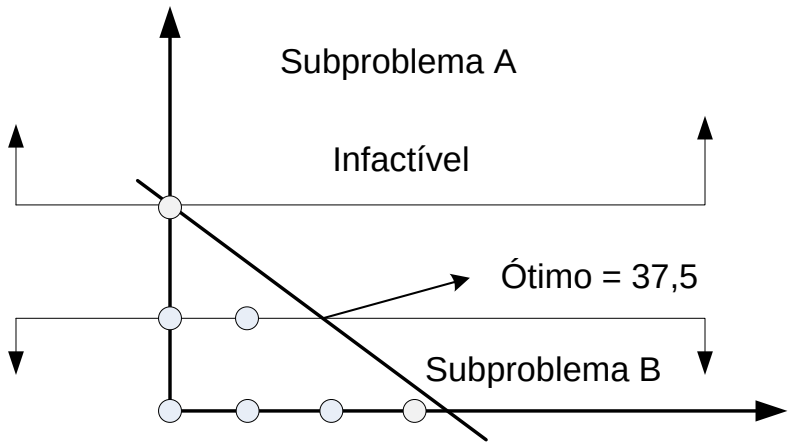


Figura 9 - Subproblema A - B

Não encontramos solução factível ao resolver o problema A, então aplicando o critério para poda podemos eliminá-lo (PN1), o problema relaxado é infactível, Figura 10.

Resolvendo o subproblema B temos $Z = 37{,}5 \quad x_1 = 1 \ e \ x_2 = 1{,}5$

Agora x_2 não é inteiro, logo particionamos o problema em dois, considerando o subproblema C com a variável $x_2 \leq 1$ e o subproblema D com $x_2 \geq 2$, Tabela 3 subproblema C – D.

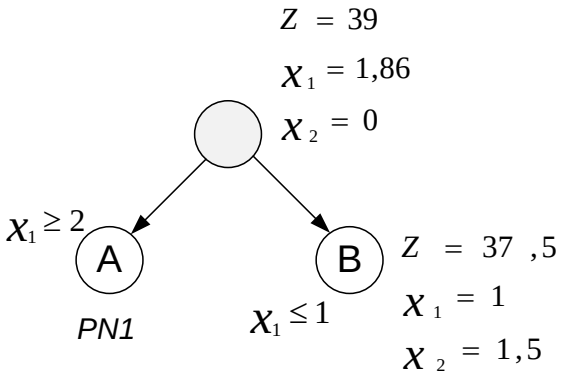


Figura 10 - Árvore do subproblema A - B

Tabela 3 – Subproblema C - D

Subproblema C	Subproblema D
max $21x_1 + 11x_2$ $sujeito\ a \quad 7x_1 + 4x_2 \leq 13$ $\boldsymbol{x_1 \leq 2}$ $\boldsymbol{x_2 \leq 1}$ $x_1, x_2 \geq 0$	max $21x_1 + 11x_2$ $sujeito\ a \quad 7x_1 + 4x_2 \leq 13$ $\boldsymbol{x_1 \leq 1}$ $\boldsymbol{x_2 \geq 2}$ $x_1, x_2 \geq 0$

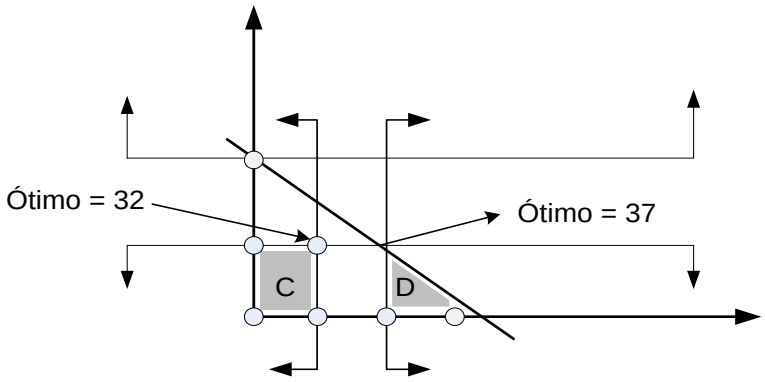


Figura 11 - Subproblema C - D

PN2 a solução ótima do problema relaxado é inteira, Figura 12.

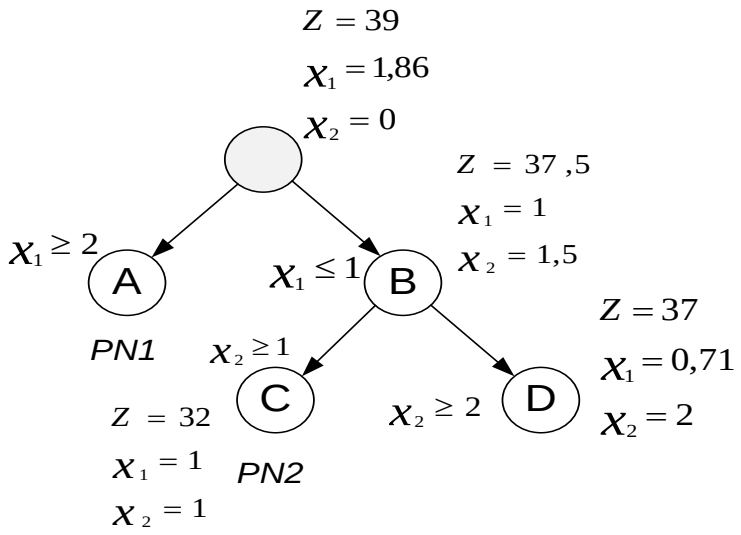


Figura 12 - Árvore do subproblema C - D

A solução do subproblema C é igual a 32, $x_1 = 1 \;\; e \;\; x_2 = 1$, as duas variáveis são inteiras, logo considerando o teste de sondagem (PN2) este problema pode ser sondado por otimalidade.

Resolvendo o subproblema D, temos $Z = 37$, $x_1 = 0{,}71 \;\; e \;\; x_2 = 2$, Figura 13 árvore do subproblema C - D.

A variável x_1 novamente não é inteira, particiona-se o subproblema gerando dois novos subproblemas como demonstrado a seguir, Tabela 4 e Figura 14.

Tabela 4 – Subproblema E - F

Subproblema E	Subproblema F
$\max\ 21x_1 + 11x_2$ $sujeito\ a \quad 7x_1 + 4x_2 \leq 13$ $\boldsymbol{x_1 \leq 1}$ $\boldsymbol{x_2 \leq 2}$ $\boldsymbol{x_1 \leq 0}$ $x_1, x_2 \geq 0$	$\max\ 21x_1 + 11x_2$ $sujeito\ a \quad 7x_1 + 4x_2 \leq 13$ $\boldsymbol{x_1 \leq 1}$ $\boldsymbol{x_2 \geq 2}$ $\boldsymbol{x_2 \leq 1}$ $x_1, x_2 \geq 0$

Na etapa de solução do subproblema E - F é definido que no passo F, aplicando o critério de poda (PN1), este pode ser eliminado.

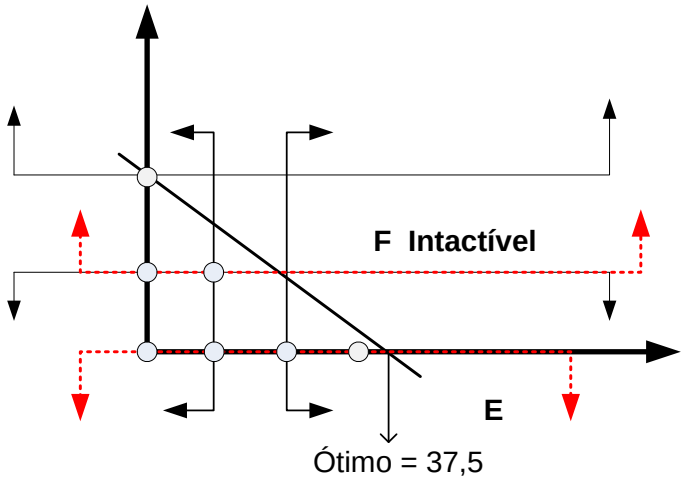


Figura 13 - Subproblema E - F

Resolvendo o subproblema G, Figura 15, obtemos $Z = 33,\ x_1 = 0\ e\ x_2 = 3$, logo a solução é inteira, portanto aplicando o PN2 este problema pode ser sondado. O subproblema H não tem solução factível e também pode ser sondado por PN1.

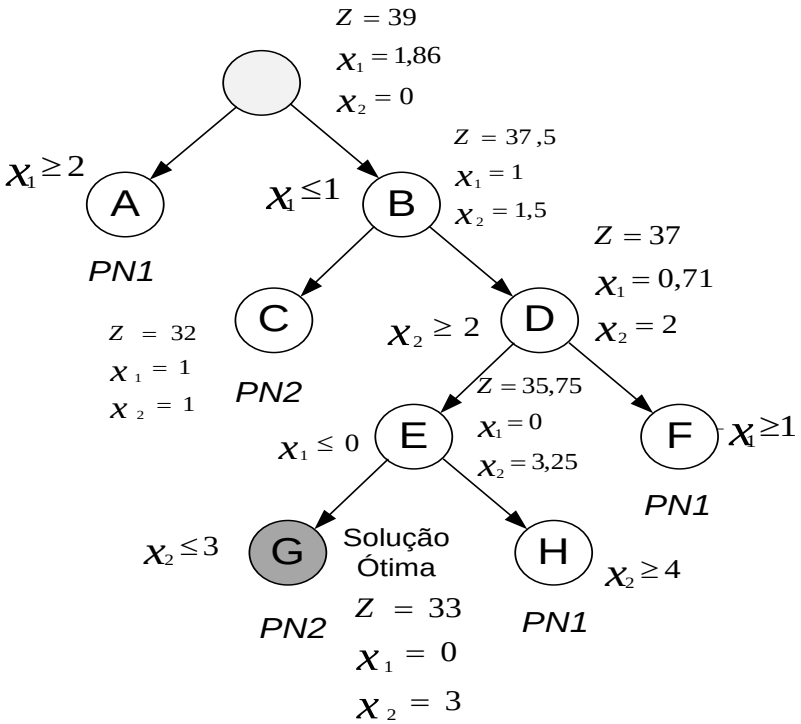


Figura 14 - Subproblema G - H

Nenhum nó pode ser ramificado, portanto, a melhor solução inteira encontrada é dada pelo problema G e é a solução ótima do problema.

Na solução do exemplo, utilizando a técnica *Branch-and-bound* , evidencia-se que muitas soluções não precisam de avaliação explicita. Isto fica claro na solução de problemas maiores.

Para que estes problemas sejam organizados e processados de acordo com uma das técnicas de resolução já listadas, e principalmente para resolução de problemas com maiores dimensões, fica evidente que um resolvedor é necessário.

A principal *interface* utilizada, nos resolvedores de PL inteira mista – PLIM comerciais, conforme Linderoth e Lodi (2010?), é a linguagem C e C++, além de também ser utilizado o Matlab, NET e Java.

Os resolvedores listados a seguir são comerciais:

CPLEX - É um resolvedor de PLIM desenvolvido pela IBM, trabalha com a técnica *Branch-and-bound* e possui várias técnicas de corte e estratégias de pesquisa. Possui um algoritmo de busca dinâmica especializado para encontrar soluções viáveis.

Gurobi - Também contém um solucionador para Programação Linear Inteira Mista – PLIM é um dos mais atuais resolvedores desenvolvidos para uso com processadores de múltiplos núcleos. Contém recursos avançados de corte e técnicas de pesquisa. Além de possuir varias linguagens de programação.

Lindo - É um solucionador PLIM que tem quinze diferentes tipos de planos de corte, seis nós diferentes e várias regras de seleção, e uma variedade de técnicas de pré-processamento e bifurcações.

Mosek - É um software que tem um solucionador e um solver com seis regras de seleção de nós, onze tipos de planos de corte e outras heurísticas, bem como metodologias avançadas de bifurcação.

X-Press MP - Tem uma solução *Branch-and-bound* para resolver PLIM, possui muitos planos de corte, técnicas de pré-processamento e tem recursos para enumerar várias soluções viáveis, e também para ajustar os parâmetros de algoritmos.

Os resolvedores listados a seguir são não comerciais:

BLIS - É um solucionador PLIM de plataforma aberta projetado para rodar em memoria distribuída e possui uma biblioteca de planos de corte.

CBC - Destina-se a resolver um grande número de problemas de PL e PLIM, e outros problemas relacionados. É um conjunto de rotinas escritas em C e organizadas sob a forma de uma biblioteca.

GLPK – É um programa para PL que utiliza um conjunto de sub-rotinas e uma biblioteca para solucionar problemas de PLIM, este software se diferencia pelo grande número de usuários e comunidades que utilizam a interface disponível para uso. O pacote GLPK inclui os seguintes componentes principais: Simplex, *Branch-and-bound* e *branch-and-cut.*

lp_solve – Também é um PLIM que admite um grande número de interfaces, como Java, AMPL, MATLAB, O-Matrix, Sysquake, Scilab, Octave, Freemat, Euler, Python, Sage, PHP e R.

MINTO – Mixed Integer Optimizer é um solucionador de PLIM, desenvolvido na década de 1990. Para a época possuía um excelente código de pré-processamento, e é utilizado como software externo para resolver o relaxamento dos problemas de PL.

SCIP – O SCIP é um pacote de software desenvolvido por pesquisadores de Berlim, com biblioteca fechada, e que recebeu diversos prêmios pela qualidade do produto. Provavelmente é o software não comercial de PLIM mais rápido existente e pode ser utilizado com várias interfaces.

SYMPHONY – É um solucionador com biblioteca fechada. Que tem como metodologia central de solução a utilização de uma bifurcação personalizável, para execução de um algoritmo *Branch-and-bound.* Algumas funções do Symphony podem ser destacadas, como a implementação de um algoritmo para PLIM bi critério e um método de aproximação de resultados. Também tem uma função para análise de sensibilidade e outra função capaz de previamente calcular a árvore de geração inicial de um algoritmo *Branch-and-bound*.

3 PROBLEMA DE DIMENSIONAMENTO E PROGRAMAÇÃO DE LOTES DE PRODUÇÃO EM MÁQUINA ÚNICA

O problema de programação e dimensionamento de lotes de produção em máquina única, tratado neste trabalho, tem como objetivo minimizar o tempo total de produção ($C_{\max}$). Cada tarefa *j* a ser processada tem um tempo de processamento p_j e um tamanho correspondente s_j, j = 1,2, ..., n. O tempo de processamento de um lote b, b = 1,2, ..., m, é dado pelo maior tempo de processamento do lote, ou seja, $C_b = \max\{p_j : x_{jb} = 1\}$.

O tamanho total do lote deve ser menor ou igual à capacidade *S* da máquina. Todas as tarefas estão disponíveis no momento 0 (zero) e nenhuma atividade pode ser dividida entre os lotes. A partir da definição de um lote, acrescentar ou retirar uma tarefa não é permitido e o processamento do lote não pode ser interrompido depois que ele é iniciado.

Dado que cada tarefa pode se constituir em um lote, será considerado que existem tantos lotes quantas tarefas ($m = n$). No caso em que o maior tamanho das tarefas for menor que a capacidade máxima da máquina é possível calcular um limitante para o número de lotes a partir da expressão $n \cdot s_{\max} / S$, em que $s_{\max}$ é o maior tamanho das tarefas.

A Figura 15 corresponde a tarefas de 1 a 10 e ao forno de *burn in*. A representação das tarefas foi estabelecida da seguinte forma; comprimento corresponde ao tempo de processamento e a altura o consumo de recurso, no caso das tarefas, e a capacidade máxima, no caso da máquina.

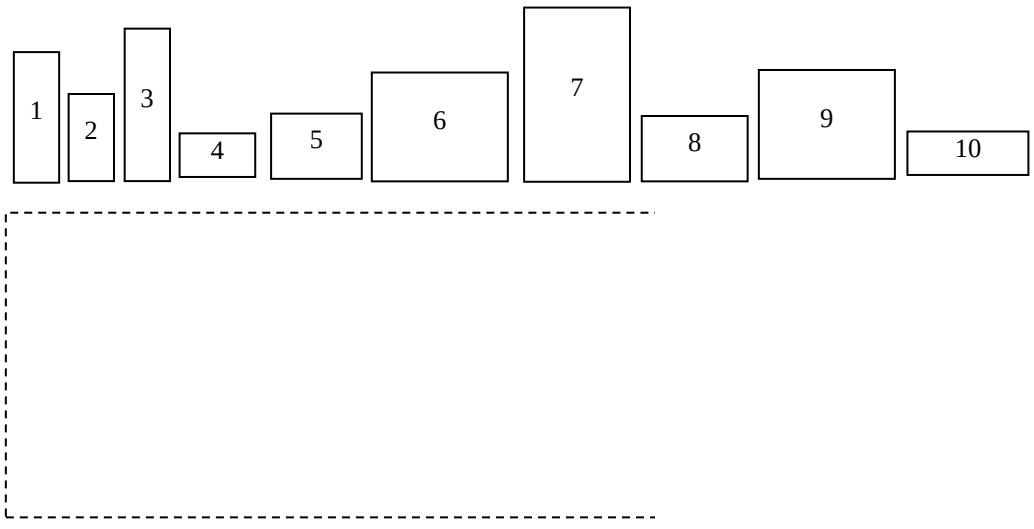


Figura 15 - Representação das Tarefas

A Figura 16 representa uma solução composta por quatro lotes. Uma característica importante a ser considerada é que o problema em questão é altamente simétrico, no sentido que uma mesma solução pode ser construída de diferentes formas.

A solução da Figura 17 é composta pelos mesmos quatro lotes da Figura 16, com a diferença que estes estão dispostos sequencialmente em uma ordem diferente. De fato, qualquer permutação na ordem dos lotes produz o mesmo valor de *makespan*.

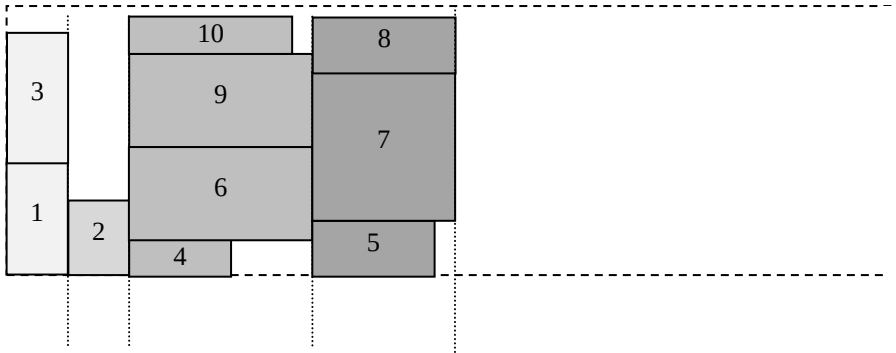


Figura 16 - *Makespan*

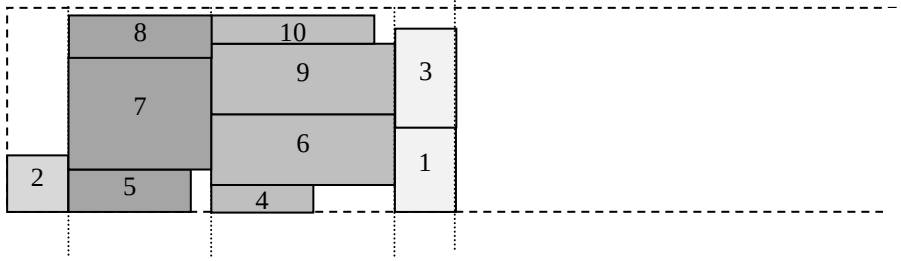


Figura 17 - *Makespan* em ordem arbitrária

Para modelar matematicamente esse problema, restrições de quebra de simetrias são importantes para reduzir o espaço de soluções. Na seção a seguir, são propostos dois conjuntos de desigualdades válidas e um procedimento de pré-processamento para tratar simetria.

3.1 MODELOS MATEMÁTICOS

O modelo apresentado a seguir é o que mais comumente aparece na literatura científica, como em Meng e Tang (2010, p. 1704) e Dupont e Dhaenens (2000, p. 809), e é denominado no Modelo 1.

Variáveis

$$x_{jb} = \begin{cases} 1 & \text{se a tarefa } j \text{ é processada no lote } b \\ 0 & \text{caso contrário} \end{cases}$$

$$y_b = \begin{cases} 1 & \text{se o lote } b \text{ é utilizado} \\ 0 & \text{caso contrário} \end{cases}$$

C_b : tempo de processamento do lote *b*

Parâmetros

s_j : tamanho da tarefa *j*

p_j: tempo de processamento da tarefa *j*

S : capacidade máxima da máquina para processar um lote

Modelo 1

$$\text{Min } C_{\max} = \sum_{b=1}^{m} C_b \tag{1}$$

s.a

$$\sum_{b=1}^{m} x_{jb} = 1 \quad j = 1, 2, ..., n \tag{2}$$

$$\sum_{j=1}^{n} s_j x_{jb} \leq S\, y_b \quad b = 1, ..., n \tag{3}$$

$$x_{jb} \leq y_b \quad j = 1, 2, ..., n; \quad b = 1, ..., n \tag{4}$$

$$p_j x_{jb} \leq C_b \quad j = 1, 2, ..., n; \quad b = 1, ..., n \tag{5}$$

$$y_b \in \{0,1\} \quad b = 1, ..., n \tag{6}$$

$$x_{jb} \in \{0,1\} \quad j = 1, 2, ..., n; \quad b = 1, ..., n \tag{7}$$

$$C_b \geq 0 \quad b = 1, ..., n \tag{8}$$

A função objetivo (1) minimiza o *makespan*, $C_{\max}$. O conjunto de restrições (2) garante que uma tarefa *j* é atribuída a um único lote *b*. O conjunto de restrições (3) garante que a capacidade *S* da máquina seja respeitada quando tarefas são atribuídas a um lote. O conjunto de restrições (5) determina o tempo de processamento do lote *b*. Os conjuntos de restrições (6), (7) e (8) especificam o tipo das variáveis de decisão. O conjunto de restrições (4) é redundante, uma vez que é uma implicação direta de (3). No entanto, é sabido ser uma desigualdade válida que, quando incluída no modelo, melhora o desempenho de resolvedores baseados em relaxações lineares.

O Modelo 1 não discrimina as soluções simétricas, para tratar a estrutura de simetria ilustrada na seção anterior são propostos dois conjuntos de desigualdades válidas semelhantes àquelas propostas Köhler (2012) para um problema de agrupamento em grafo. O primeiro é formulado como segue.

$$\sum_{b=j}^{m} x_{jb} = 1 \quad j = 1, 2, ..., n \tag{9}$$

Esse conjunto de restrições força que cada tarefa seja designada para um lote com índice maior ou igual ao índice da própria tarefa. Desta forma, a tarefa *n* só pode ser designada para o lote *n*, a tarefa *n* – 1 para os lotes *n* – 1 e *n*, e assim por diante. Desta forma, uma ordem é implicitamente definida para o uso dos lotes e, assim, somente uma das soluções com o mesmo valor de função objetivo, que diferem apenas no índice dos lotes, é considerada.

O segundo conjunto de desigualdades válidas, conforme definido em (10), faz uso do fato que (9) força o índice de um cluster para que seja igual ao maior índice das tarefas que compõem o cluster. Deste modo, se a tarefa *k* não está no lote *k*, então o lote *k* deve ser considerado vazio.

$$x_{jb} \leq x_{bb} \quad j = 1, 2, ..., n; b = 1, ..., n \tag{10}$$

Considerando estes conjuntos de restrições no exemplo da Figuras 15, 16 e 17, uma solução possível é expressa por b_2 ={2}, b_3 = {1,3}, b_8 = {5,7,8}, b_{10} = {4,6,9,10}, que apresenta o mesmo valor de *makespan.*

O modelo que inclui as restrições de quebra de simetria é formalizado a seguir e denominado Modelo 2.

Modelo 2

Min (1)

s.a

(3), (4), (5), (6),(7),(8)

$$\sum_{b=j}^{m} x_{jb} = 1 \quad j = 1, 2, ..., n \tag{12}$$

$$x_{jb} \leq x_{bb} \quad j = 1, 2, ..., n; b = 1, ..., n \tag{13}$$

Ainda, é possível lançar mão de uma rotina de pré-processamento para efetuar uma ordenação das tarefas de acordo como tamanho (ordem crescente). Esse procedimento simples, associado a (12) e (13), faz com que as tarefas com maior probabilidade de determinarem o *makespan* de cada lote tenham menor possibilidade de designação para diferentes lotes. O modelo com pré-processamento é denominado **Modelo 3**.

No Modelo 3, caso o lote *b* seja definido uma das tarefas que determina o tempo de processamento deste lote é a tarefa correspondente à variável x_{bb}. A partir desta observação é possível formalizar um modelo no qual não são necessárias as variáveis C_b. Esse modelo é definido a seguir e denominado Modelo 4.

Modelo 4

$$\text{Min} \quad C_{\max} = \sum_{b=1}^{m} p_b x_{bb} \tag{14}$$

s.a

(9), (10), (7)

$$\sum_{j=1}^{n} s_j x_{jb} \leq S\, x_{bb} \quad b = 1, ..., n \tag{15}$$

O Modelo 4 apresenta características semelhantes ao problema de *bin packing*, Dhaenens (2000, p. 809), uma vez que é necessário definir cada lote e o valor deste lote é dado por x_{bb}.

Como já mencionando antes na definição do problema, o limitante inferior para o número de lotes é calculado pela expressão $n \cdot s_{\max} / S$, em que $s_{\max}$ é o maior valor do tempo

de processamento. Este limitante corresponde ao número mínimo de *bins* necessários para "empacotar" todas as tarefas. A forma mais simples de incluir expressamente esse limitante no modelo corresponde à restrição (15), definida a seguir.

$$\sum_{b=1}^{m} x_{bb} \leq \frac{n \cdot s_{\max}}{S} \tag{16}$$

É importante observar que o uso de qualquer lote além do limite estipulado na restrição (16) obrigatoriamente aumenta o valor de *makespan* e, por isso, pode ser descartado.

4 RESULTADOS COMPUTACIONAIS

Os testes computacionais foram processados em um computador Intel Quad Core Xeon X3360 2.83 GHz e o otimizador CPLEX 12.1, com configuração padrão, foi utilizado para resolver as quatro formulações estudadas.

O conjunto de instâncias testes foi gerado de acordo com Parsa et al. (2010). Quatro fatores são considerados para gerar as instâncias: número de tarefas, intervalo do tamanho das tarefas, intervalo do tempo de processamento e capacidade da máquina. O tempo de processamento e o tamanho das tarefas foram gerados aleatoriamente a partir de uma distribuição uniforme, de acordo com os valores dos fatores apresentados na Tabela 5.

Tabela 5 – Valores dos fatores considerados para gerar instâncias aleatórias

Fatores	Valores
Número de tarefas (n)	20, 40, 60, 80, 100
Tamanho das tarefas (s_j)	[1,10] ; [4,8] ; [1,5] ; [2,4]
Tempo de processamento das tarefas (p_j)	[1,10] ; [1,5]
Capacidade da máquina (S)	5, 10

A Tabela 6 apresenta seis diferentes tipos de problemas combinando os quatro intervalos de tamanho de tarefas e os dois níveis de capacidade para a máquina. Para cada tipo, os cinco diferentes números de tarefas e dois níveis de tempo de processamento são utilizados, possibilitando $6 \times 2 \times 5 = 60$ possíveis combinações. Para cada combinação foram geradas 10 instâncias testes, o que totaliza 600 instâncias testes. Em todos os testes o tempo computacional máximo foi limitado em 1800 segundos.

Tabela 6 – Tipos de instâncias

Tipo da instância	Tamanho das tarefas (s_j)	Capacidade da máquina (B)
1	[1,10]	10
2	[4,8]	10
3	[1,5]	10
4	[2,4]	10
5	[1,5]	5
6	[2,4]	5

Os resultados para os modelos 2, 3 e 4, considerando os tempos de processamento $p_j \in [1,10]$ e $p_j \in [1,5]$, são apresentados na Tabela 7 e Tabela 8, respectivamente. Nessas tabelas, as colunas apresentam o tipo da instância, o número de tarefas, o tempo de processamento, o número de soluções comprovadamente ótimas e o *gap* final. Os valores, à

exceção do número de ótimos, são médias das dez instâncias de cada configuração. É possível verificar que o Modelo 4 apresenta o melhor desempenho médio em todos os quesitos avaliados e possibilitou a obtenção de todos os ótimos que os demais modelos foram hábeis para encontrar, à exceção para o conjunto de instâncias do tipo 3 com 40 tarefas. O tempo computacional dos modelos 3 e 4 foi bastante semelhante e notadamente menor que o tempo computacional do modelo 2.

O pior desempenho, considerando todos os três modelos, ocorreu para os experimentos com as instâncias do tipo 3 e 4. As tarefas que compõem essas instâncias apresentam tamanhos dentro dos intervalos [1,5] e [2,4] e capacidade máxima da máquina igual a 10. A diferença entre o maior tamanho das tarefas e a capacidade da máquina pode ser investigada como uma possível explicação para este desempenho pior. Ainda assim, o *gap* obtido a partir do Modelo 4 não foi superior a 2%.

Os testes computacionais com restrição para o número máximo de lotes demonstraram que este tipo de restrição tem impacto praticamente nulo nos resultados finais. De fato, em alguns casos o tempo computacional aumenta e a qualidade da solução deteriora. Uma explicação possível para esse fato reside no pré-processamento e nas algoritmos para encontrar cortes que o pacote CPLEX utiliza. Muitas vezes a inclusão de uma restrição quebra uma determinada estrutura facilmente reconhecida e, com isso, o resolvedor não consegue eliminar variáveis/restrições durante o pré-processamento ou ainda encontrar cortes durante a evolução da busca em árvore.

Devido ao melhor desempenho do Modelo 4, os demais resultados analisados consideram este modelo e o Modelo 1, utilizado na literatura científica.

Tabela 7 – Resultados dos modelos 2,3, e 4 para os seis tipos de instâncias com $p_j \in [1,10]$

Inst	***n***	**Modelo 2**			**Modelo 3**			**Modelo 4**		
		T(s)	**#O**	**G(%)**	**T(s)**	**#O**	**G(%)**	**T(s)**	**#O**	**G(%)**
1	20	0,1	10	0,00	0,0	10	0,00	0,0	10	0,00
	40	187,4	10	0,00	0,6	10	0,00	0,1	10	0,00
	60	304,7	10	0,00	1,7	10	0,00	14,3	10	0,00
	80	583,3	9	0,07	14,6	10	0,00	25,8	10	0,00
	100	1213,3	5	1,40	85,1	10	0,00	182,6	10	0,00
2	20	0,0	10	0,00	0,0	10	0,00	0,0	10	0,00
	40	0,2	10	0,00	0,2	10	0,00	0,0	10	0,00
	60	29,4	10	0,00	0,9	10	0,00	0,0	10	0,00
	80	128,4	10	0,00	1,1	10	0,00	0,0	10	0,00
	100	326,0	10	0,00	2,4	10	0,00	0,1	10	0,00
3	20	234,1	10	0,00	1,0	10	0,00	0,2	10	0,00
	40	1800,0	1	6,81	1462,1	3	3,97	160,6	10	0,00
	60	1800,0	0	23,56	1800,0	0	16,66	1207,4	4	0,92

	80	1800,0	0	33,83	1800,0	0	22,49	1646,2	1	1,98
	100	1800,0	0	31,30	1800,0	0	20,28	1800,0	0	1,50
4	20	309,7	10	0,00	2,3	10	0,00	0,1	10	0,00
	40	1800,0	0	15,04	1696,9	2	8,62	252,2	10	0,00
	60	1800,0	0	30,30	1800,0	0	24,20	832,9	6	0,65
	80	1800,0	0	35,55	1800,0	0	27,80	1225,4	4	0,80
	100	1800,0	0	41,24	1800,0	0	32,07	1629,0	1	1,44
5	20	0,1	10	0,00	0,0	10	0,00	0,0	10	0,00
	40	65,5	10	0,00	1,1	10	0,00	0,1	10	0,00
	60	543,0	9	0,30	181,4	9	0,11	21,1	10	0,00
	80	493,5	10	0,00	1,8	10	0,00	0,1	10	0,00
	100	906,4	6	0,31	192,7	9	0,06	48,2	10	0,00
6	20	0,0	10	0,00	0,0	10	0,00	0,0	10	0,00
	40	0,3	10	0,00	0,2	10	0,00	0,0	10	0,00
	60	31,2	10	0,00	0,7	10	0,00	0,0	10	0,00
	80	270,8	10	0,00	1,9	10	0,00	0,0	10	0,00
	100	572,6	10	0,00	5,8	10	0,00	0,1	10	0,00

A Figura 18 demonstra o Gráfico do tempo de processamento para a instância 6, da Tabela 7. É bastante claro que a velocidade de processamento do Modelo 4 é muito superior a dos Modelos 2 e superior a do modelo 3. Verificar o Apêndice D.

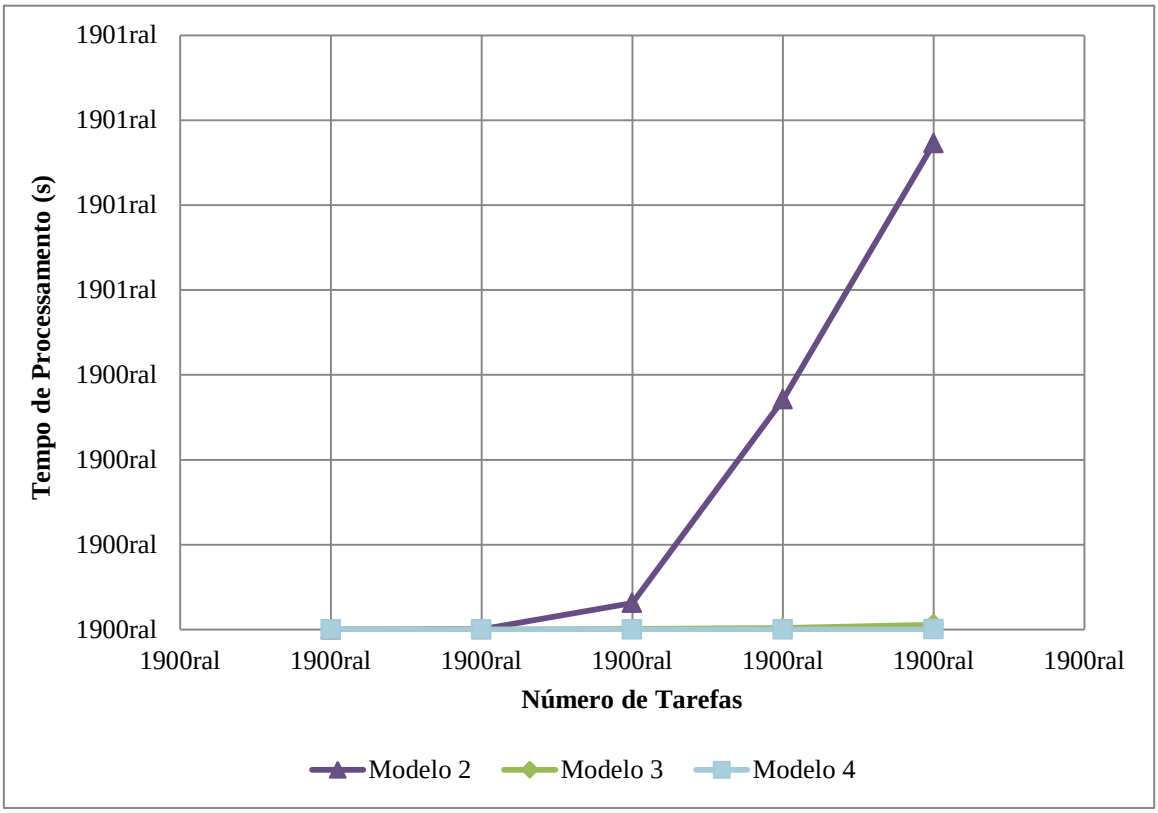


Figura 18 - Resultados dos modelos 2, 3 e 4 para a instância 6 com $p_j \in [1,10]$

Tabela 8 – Resultados dos modelos 2,3, e 4 para os seis tipos de instâncias com $p_j \in [1,5]$

Inst	*n*	Modelo 2			Modelo 3			Modelo 4		
		T(s)	#O	G(%)	T(s)	#O	G(%)	T(s)	#O	G(%)
1	20	0,1	10	0,00	0,0	10	0,00	0,0	10	0,00
	40	148,7	10	0,00	0,3	10	0,00	0,1	10	0,00
	60	714,1	8	0,49	3,1	10	0,00	0,2	10	0,00
	80	449,5	9	0,37	12,3	10	0,00	196,1	9	0,08
	100	1088,6	5	1,55	263,8	9	0,45	103,4	10	0,00
2	20	0,0	10	0,00	0,0	10	0,00	0,0	10	0,00
	40	0,2	10	0,00	0,1	10	0,00	0,0	10	0,00
	60	52,5	10	0,00	0,6	10	0,00	0,0	10	0,00
	80	153,0	10	0,00	1,1	10	0,00	0,0	10	0,00
	100	312,3	10	0,00	2,2	10	0,00	0,1	10	0,00
3	20	71,5	10	0,0	0,6	10	0,00	0,0	10	0,00
	40	1786,8	1	11,39	1232,7	4	4,26	782,2	7	1,00
	60	1800,0	0	17,85	1800,0	0	12,72	1335,5	3	1,72
	80	1800,0	0	21,35	1800,0	0	14,31	1474,3	2	2,13
	100	1800,0	0	27,54	1800,1	0	16,67	1679,1	1	1,64
4	20	190,5	10	0,00	1,4	10	0,00	13,9	10	0,00
	40	1686,3	1	10,61	1328,8	3	5,40	370,4	9	0,32
	60	1800,0	0	22,62	1800,0	0	15,43	1096,8	5	1,16
	80	1800,0	0	26,58	1800,0	0	21,09	1800,0	0	2,69
	100	1800,1	0	32,53	1800,1	0	21,59	1800,0	0	2,50
5	20	0,1	10	0,0	0,0	10	0,00	0,0	10	0,00
	40	95,6	10	0,00	0,6	10	0,00	0,0	10	0,00
	60	371,0	10	0,00	1,5	10	0,00	12,1	10	0,00
	80	819,9	8	0,49	171,8	10	0,00	184,5	9	0,07
	100	409,9	9	0,22	6,9	10	0,00	27,2	10	0,00
6	20	0,0	10	0,00	0,0	10	0,00	0,0	10	0,00
	40	0,3	10	0,00	0,1	10	0,00	0,0	10	0,00
	60	43,2	10	0,00	0,4	10	0,00	0,0	10	0,00
	80	204,7	10	0,00	0,4	10	0,00	0,0	10	0,00
	100	436,4	10	0,00	0,7	10	0,00	0,0	10	0,00

As Tabelas 9 e 10 apresentam resultados comparativos referentes aos Modelos 1 e 4. Os dados analisados são os mesmos considerados nas tabelas anteriores. Os resultados obtidos com o Modelo 4 são claramente superiores em qualidade, mesmo para instâncias pequenas (20 e 40 tarefas) o tempo computacional é menor. Considerando tempos de processamento dentro do intervalo [0,10], para todos os tipos de instâncias com 100 tarefas, não foi possível provar a otimalidade de nenhuma das soluções encontradas com o Modelo 1 e o *gap* médio sempre é maior que 60% . Para as instâncias com tempo de processamento dentro do intervalo [0,5], os resultados foram bastante semelhantes.

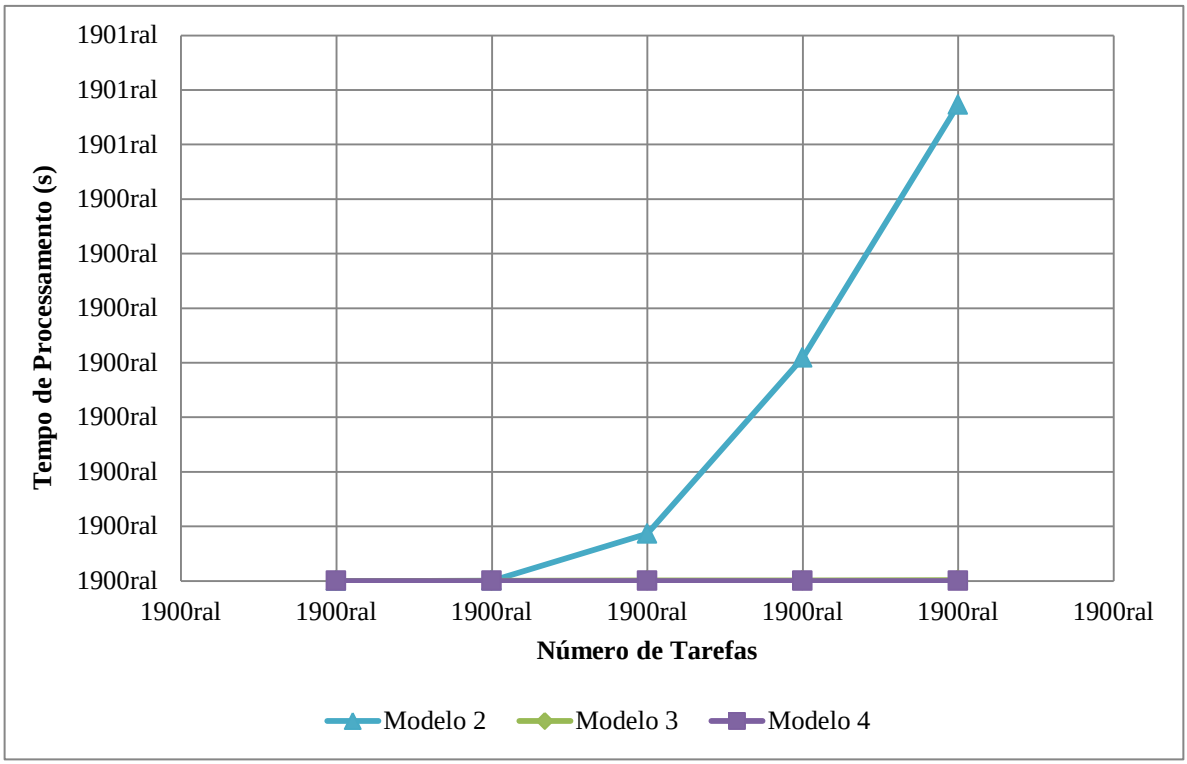


Figura 19 - Resultados dos modelos 2, 3 e 4 para a instância 6 com $p_j \in [1,5]$

A Figura 19 demonstra o Gráfico do tempo de processamento para a instância 6, da Tabela 8. É bastante claro que a velocidade de processamento do Modelo 4 é muito superior a dos Modelos 2 e superior a do modelo 3. O número de soluções comprovadamente ótimas nesta instância para todas as tarefas é de 100%, além de uma grande velocidade de processamento todas as soluções ótimas são alcançadas para qualquer número de tarefas da instância. Verificar o Apêndice E.

Tabela 9 – Resultados dos modelos 1 e 4 para os seis tipos de instâncias com $p_j \in [1,10]$

Inst	***n***	**Modelo 1**			**Modelo 4**		
		T(s)	**#O**	**G(%)**	**T(s)**	**#O**	**G(%)**
1	20	241,2	9	0,133	0,0	10	0,00
	40	1800,0	0	8,573	0,1	10	0,00
	60	1651,1	1	10,955	14,3	10	0,00
	80	1800,0	0	10,844	25,8	10	0,00
	100	1800,0	0	67,446	182,6	10	0,00
2	20	194,4	9	0,130	0,0	10	0,00
	40	1630,0	1	6,008	0,0	10	0,00
	60	1800,0	0	13,638	0,0	10	0,00
	80	1800,0	0	12,908	0,0	10	0,00
	100	1800,0	0	61,802	0,1	10	0,00
3	20	234,1	9	1,890	0,2	10	0,00

	40	1800,0	0	49,063	160,6	10	0,00
	60	1800,0	0	71,808	1207,4	4	0,92
	80	1800,0	0	81,582	1646,2	1	1,98
	100	1800,0	0	86,292	1800,0	0	1,50
4	20	205,9	10	0,000	0,1	10	0,00
	40	1800,0	0	51,661	252,2	10	0,00
	60	1800,0	0	68,591	832,9	6	0,65
	80	1800,0	0	81,411	1225,4	4	0,80
	100	1800,0	0	87,305	1629,0	1	1,44
5	20	555,6	7	0,460	0,0	10	0,00
	40	1800,0	10	7,144	0,1	10	0,00
	60	1800,0	0	8,161	21,1	10	0,00
	80	1276,6	4	6,331	0,1	10	0,00
	100	1800,0	0	71,367	48,2	10	0,00
6	20	731,4	6	2,279	0,0	10	0,00
	40	1267,9	3	3,996	0,0	10	0,00
	60	1306,6	3	3,538	0,0	10	0,00
	80	1578,2	3	3,299	0,0	10	0,00
	100	1800,0	0	66,015	0,1	10	0,00

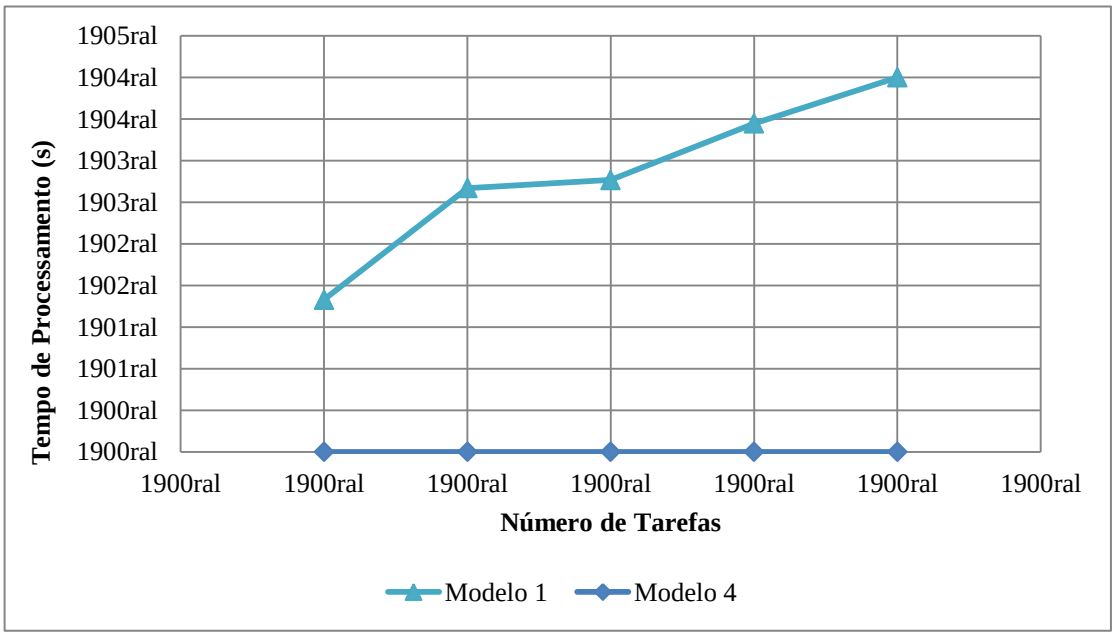


Figura 20 - Resultados dos modelos 1 e 4 para a instâncias 6 com $p_j \in [1,10]$

O gráfico da Figura 20, que compara os resultados dos Modelos 1 e 4, demonstra a superioridade do Modelo 4 em todas as análises. O tempo de processamento para a tarefa 100 no Modelo 1 ultrapassa a 1800 s e o tempo do Modelo 4 para a mesma tarefa, chega a 0,1 s.

Um tempo de processamento do modelo muito superior ao do Modelo 1, o que comprova mais uma vez a superioridade do modelo proposto.

Tabela 10 – Resultados dos modelos 1 e 4 para os seis tipos de instâncias com $p_j \in [1,5]$

Inst	***n***	**Modelo 1**			**Modelo 4**		
		T(s)	**#O**	**G(%)**	**T(s)**	**#O**	**G(%)**
1	20	486,6	9	0,26	0,0	10	0,00
	40	1444,7	2	7,18	0,1	10	0,00
	60	1800,0	0	14,82	0,2	10	0,00
	80	1800,1	0	12,23	196,1	9	0,08
	100	1800,3	0	39,39	103,4	10	0,00
2	20	217,1	9	0,20	0,0	10	0,00
	40	922,3	5	3,04	0,0	10	0,00
	60	1693,3	1	11,78	0,0	10	0,00
	80	1800,1	0	11,90	0,0	10	0,00
	100	1800,1	0	39,79	0,1	10	0,00
3	20	757,0	7	5,61	0,0	10	0,00
	40	1800,0	0	59,66	782,2	7	1,00
	60	1800,0	0	74,04	1335,5	3	1,72
	80	1800,0	0	82,44	1474,3	2	2,13
	100	1800,1	0	86,80	1679,1	1	1,64
4	20	880,3	6	8,38	13,9	10	0,00
	40	1800,0	0	53,02	370,4	9	0,32
	60	1800,0	0	71,13	1096,8	5	1,16
	80	1800,0	0	80,86	1800,0	0	2,69
	100	1800,1	0	87,51	1800,0	0	2,50
5	20	186,8	9	0,26	0,0	10	0,00
	40	1621,8	1	8,86	0,0	10	0,00
	60	1459,7	2	6,88	12,1	10	0,00
	80	1670,6	1	8,41	184,5	9	0,07
	100	1790,1	1	38,91	27,2	10	0,00
6	20	281,8	9	0,25	0,0	10	0,00
	40	732,5	6	2,29	0,0	10	0,00
	60	1141,6	4	3,75	0,0	10	0,00
	80	1040,1	6	1,43	0,0	10	0,00
	100	1760,9	3	36,79	0,0	10	0,00

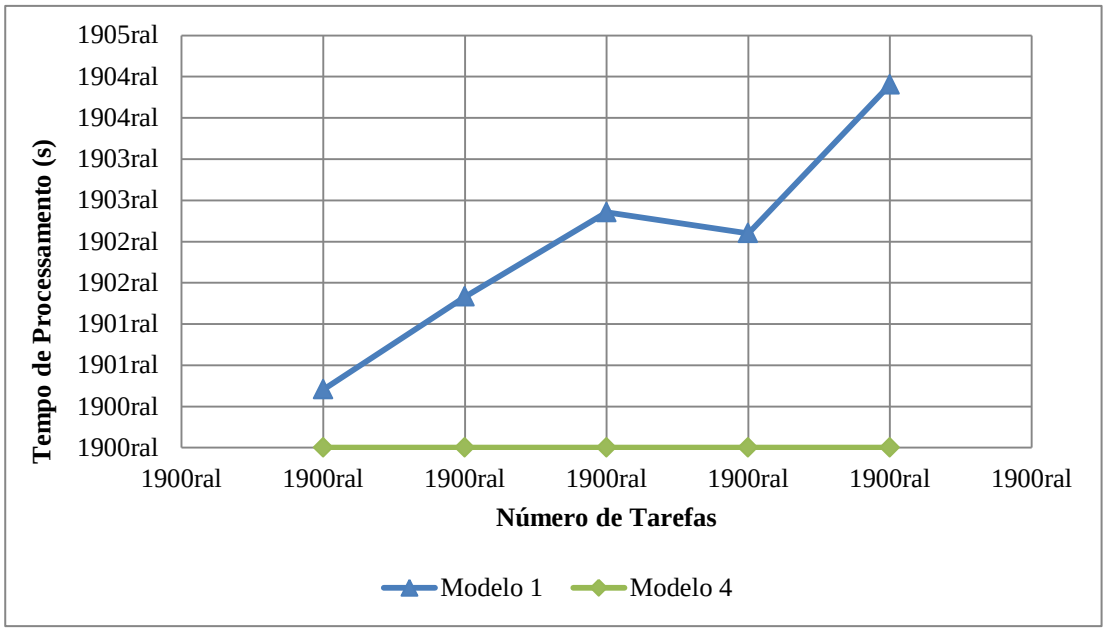


Figura 21 - Resultados dos modelos 1 e 4 para a instância 6 com $p_j \in [1,5]$

Como o gráfico da Figura 20, o resultado dos Modelos 1 e 4 para a instância 6 e $p_j \in [1,5]$, são muito semelhantes, o Modelo 4 apresenta superioridade de desempenho na comparação com o Modelo 1.

A Tabela 11 exibe resultados comparativos para os Modelos 1 e 4, nessa tabela são apresentados o número total médio de nós avaliados e o tempo médio para encontrar a melhor solução reportada pelo respectivo modelo. Esses resultados demonstram o efeito da simetria quando se observa a grande diferença no número de nós avaliados pelos modelos e, por conseguinte, o tempo para encontrar uma solução ou provar que é ótima.

Para instâncias com 80 ou 100 tarefas, eventualmente, o Modelo 1 avalia menos nós que o Modelo 4, mas isso ocorre porque o tempo de processamento de cada nó é maior no primeiro caso. É possível comprovar esse fato a partir do tempo para encontrar a solução ótima, ou a melhor solução (incumbente), que é menor para o Modelo 4. A exceção a essa regra ocorre nas instâncias do tipo 3 e 4, para as quais o número de nós avaliados para o Modelo 4 é bem superior. No entanto, a explicação é a mesma, uma vez que essas instâncias foram aquelas que apresentaram maior dificuldade para obtenção da solução ótima.

Devido ao tempo computacional gasto para encontrar a melhor solução a partir do Modelo 4, é possível especular que o uso de um pacote de otimização como o CPLEX pode ser utilizado com tempo computacional limite e, desta forma, produzir soluções heurísticas de boa qualidade em um tempo computacional razoável par fins práticos.

Tabela 11 – Número de nós e tempo computacional para encontrar a melhor solução dos modelos 1 e 4.

Inst	***n***	**[1,10]**				**[1,5]**			
		Modelo 1		**Modelo 4**		**Modelo 1**		**Modelo 4**	
		Nós	Tempo I (s)	Nós	Tempo I (s)	Nó	Tempo I (s)	Nós	Tempo I (s)
1	20	128284	1,3	30	0,0	796819	0,6	10	0,0
	40	490418	58,0	947	0,0	433238	27,5	844	0,1
	60	111008	485,5	2209	0,1	132692	482,0	1673	0,1
	80	20255	1325,7	2174	0,2	44883	985,0	2065805	0,3
	100	394	779,8	8878	36,4	2351	851,4	11048	51,6
2	20	167265	0,7	2	5,0	512713	0,5	3	7,0
	40	631222	45,3	22	4,0	464006	26,6	11	2,0
	60	130188	324,2	536	1,0	111600	227,6	249	1,0
	80	17902	1256,4	70	0,0	36201	739,5	27	0,0
	100	976	930,7	164	0,1	1194	1343,5	109	0,1
3	20	303808	13,0	3942	0,0	1257184	0,5	386	0,1
	40	216297	623,0	97328	0,1	221048	355,2	6168010	25,9
	60	57965	1279,8	4737327	0,4	105467	1078,2	2944296	25,3
	80	14415	910,8	4068218	37,2	38845	944,4	2492150	254,1
	100	4826	1202,0	3125717	204,5	8010	1181,8	2822518	366,7
4	20	365556	3,7	1388	0,0	1372957	1,1	8069	0,0
	40	238880	1124,3	330392	0,3	261852	678,0	4061364	63,8
	60	60838	1516,0	5965295	89,5	99290	1357,0	5403492	167,5
	80	14433	1198,3	5063635	138,7	35903	1378,3	4586096	236,6
	100	3814	1369,3	4373614	160,5	7363	1069,1	4534252	296,5
5	20	920134	0,7	126	1,0	311928	0,5	9	3,0
	40	571035	40,1	3643	0,0	654636	32,9	67	1,0
	60	243105	378,0	116262	1,1	181448	177,7	3263	0,1
	80	32019	859,2	447	0,1	49261	758,7	4930175	0,1
	100	1596	1416,5	842857	12,1	2018	1320,3	411801	20,3
6	20	1276645	1,1	47	7,0	557400	0,4	0	6,0
	40	717905	28,8	9	9,0	445392	15,0	6	5,0
	60	192822	178,5	4	7,0	219118	136,7	0	7,0
	80	47645	963,4	10	8,0	45986	482,2	17	6,0
	100	397	862,7	147	7,0	6276	1598,9	30	3,1

5 CONCLUSÕES

Neste trabalho foram proposta novas formulações de programação inteira 0-1 para modelar o problema de programação e dimensionamento de lotes de produção em máquina única. As formulações propostas fazem uso de dois conjuntos de desigualdades válidas e são mais forte do que aquela comumente utilizada na literatura como base de comparação para outros métodos. Adicionalmente, foi utilizado, em conjunto com as desigualdades válidas, um procedimento de pré-processamento que ordena as variáveis segundo o tempo de processamento.

As desigualdades válidas propostas quebram a estrutura de simetria inerente ao problema, decorrente do fato que a ordem na qual os lotes são processados é irrelevante de acordo com a definição do problema.

Um conjunto de instâncias testes foi gerado de acordo com especificações retiradas da literatura científica. Esse conjunto abrange 600 instâncias com diferentes características no que se refere a número de tarefas, tempo de processamento, tamanho das tarefas e capacidade da máquina. Os resultados computacionais demostraram o desempenho superior das formulações propostas em todos os quesitos avaliados, em que os melhores resultados foram obtidos com o modelo que considera na formulação o pré-processamento em conjunto com as desigualdades válidas. O modelo denominado Modelo 4 apresentou o melhor desempenho dentre todos os analisados e o tempo computacional para encontrar a solução ótima, ou a melhor solução, quando não foi possível provar a otimalidade, sugere que o este pode ser utilizado para produzir soluções de boa qualidade utilizando tempos computacionais razoáveis para situações reais. Isso cresce em importância quando consideramos que o projeto e implementação de heurísticas especializadas para esse problema pode demandar um longo tempo de desenvolvimento e ajuste de parâmetros. Diferente do tempo necessário para a implementação computacional dos modelos matemáticos propostos para serem resolvidos com os pacotes de otimização.

Vislumbra-se como trabalho futuro a possibilidade de estender as formulações utilizadas para o caso de máquinas paralelas, bem como a inclusão de restrições de liberação das tarefas (*ready time*).

6 REFERÊNCIAS

ARENALES, Marcos (et al). **Pesquisa Operacional**. Rio de Janeiro, Elsevier, 2008

AZIZOGLU M, WEBSTER S. Scheduling a batch processing machine with incompatible job families. Computers & Industrial Engineering 2001;39: 325–35.

CHEN, Huaping; DU, Bing; HUANG, George Q. Scheduling a batch processing machine with non-identical job sizes: a clustering perspective, **International Journal of Production Research**, v. 49, n. 19, p. 5755- 5778, 2011.

CHASE, Richard B.; JACOBS, F. Robert; AQUILANO, Nicholas J. **Administração da Produção para Vantagem Competitiva.** Porto Alegre. Bookman, 2008.

DIAS, Joana Matos. **Localização Dinâmica Modelos e Técnica.** Coimbra – Portugal: Imprensa da Universidade de Coimbra, 2010.

DAMODARAN, Purushothaman; MANJESHWAR, Praveen Kumar: SRIHARI, Krishnaswami. Minimizing makespan on a batch-processing machine with non-identical job sizes using genetic algorithms, **International Journal of Production Economics,** v. 103, p. 882-891, fevereiro 2006.

DOBSON, G.; NAMBIMADOM R-S. The batch loading and scheduling problem. **Research Report**, Simon School of Business Administration, University of Rochester, Rochester, NY, USA; 1992.

DUPONT, Lionel; DHAENENS-FLIPO, Clarisse; Minimizing the makespan on a batch machine with non-identical job sizes: an exact procedure. **Computers & Operations research,** v. 29, p. 807-819, julho 2000.

GHAZVINI, Fariborz; DUPONT, Lionel. Minimizing mean flow times criteria on a single batch processing machine with non-identical jobs sizes. **International Journal of Production Economics,** v. 55, p. 273-280, março 1998.

GOLDBARG, Marco Cesar; LUNA, Henrique Pacca. **Otimização Combinatória e Programação Linear**. Rio de Janeiro, Elsevier, 2005.

GAREY, Michael R.; JOHNSON, David S. **Computers and Intractability: A Guide to the Theory of NP-Completeness**. New York, W. H. Freeman, 1979.

GAITHER, Norman; FRAZIER, Greg. **Administração da Produção e Operações.** São Paulo. Thomson Learning, 2005.

HARTMANIS, J; STEARNS, R. E. On the computational complexity of algorithms. **Trans. American Mathematical Society**, 117:285--306, Maio 1965.

HILLIER, Frederick S.; LIEBERMAN, Gerald J. **Introdução à Pesquisa Operacional**. Porto Alegre, AMGH, 2010.

KÖHLER, V. **Programação Matemática Aplicada ao Problema de Clusterização com Aplicação em Engenharia de Software**. Tese de Doutorado, Universidade Federal do Rio de Janeiro- COPPE/UFRJ, 2012.

LINDEROTH, Jeffrey T.; LODI, Andrea; **MILP Software**. [S.l. : s.n.]; [2010?] Department of Industrial and Systems Engineering University of Wisconsin-Madison.

MELOUK, Sharif; DAMODARAN, Purushothaman; CHANG, Ping-YU. Minimizing makespan for single machine batch processing with non-identical job sizes using simulated annealing. **International Journal of Production Economics,** v. 87, p. 141-147, fevereiro 2003.

MENG, Ying; TANG, Lixin; A tabu search heuristic to solve the scheduling problem for a batch-processing machine with non-identical job sizes. **Logistic Systems and Intelligent Management,** v. 3, p. 1703 - 1707, janeiro 2010.

MOREIRA, Daniel Augusto. **Administração da Produção e Operações.** São Paulo. Thomson Learning, 2001.

MATHIRAJAN, M.; SIVAKUMAR, A. I. A literature review, classification and simple meta-analysis on scheduling of batch processors in semiconductor. **International Journal of Advance Manufacturing,** v. 29, p. 990-100, 2006.

PARSA, N. Rafiee; KARIMI, B.; KASHAN, A. Husseinzadeh. A branch and price algorithm to minimize makespan on a single batch processing machine with non-identical job sizes, **Computers & Operations Research**, v. 37, p. 1720- 1730, 2009.

POTTS, C. N.; KOVALYOV, M. Y. Scheduling with batching: a review, **European Journal of Operational Research,** v. 120, p. 228-249, 2000.

RENDER, Barry; HANNA, Michel E.; STAIR, Ralph M. **Análise Quantitativa Para Administração.** Porto Alegre. Bookman, 2010.

SUNG, C. S.; CHOUNG, Y. I.; Minimizing makespan on a single burn-in over in semiconductor manufacturing. **European Journal of Operational Research,** v. 120, p. 559 - 574, setembro 1998.

UZSOY, R. A single batch processing machine with non-identical job sizes, **International Journal of Production Research**, v. 32, n. 7, p. 1615–1635, 1994.

XU, Shubin; BEAN, Janes C.; A Genetic Algorithm for Scheduling Parallel Non-identical Batch Processing Machines**. Symposium on Computational Intelligence In Scheduling**, v. 07, p. 143-150, 2007.

XU, Rui; CHEN, Huaping; LI, Xueping; *Makespan* minimization on single batch-processing machine via ant colony optimization**.** . **Computers & Operations research,** v. 39, p. 582-593, maio 2011.

7 APÊNDICE A

O *wafer* é o principal elemento usado na fabricação dos chips. O *wafer* "virgem" é feito de silício puro, que é extraído da areia da praia. O *wafer* é criado através de um método chamado *Czochralski*, onde um pedaço de cristal de silício é colocado em uma vareta e então mergulhado em silício derretido. A vareta é suspensa e girada ao mesmo tempo, formando um grande cilindro de cristal de silício, também conhecido como lingote (ou "*ingot*", em inglês).

O lingote resultante deste processo mede de um a dois metros de comprimento e pode ter até 300 mm de diâmetro (é daí que vem termos como "*wafer* de 300 mm"). O lingote é então "fatiado" em *wafers*. Esses *wafers* são polidos e enviados para a fabricação do *chip*. Em cima deste *wafer* "virgem" é que os *chips* serão fabricados.

8 APÊNDICE B

A formulação de um problema, para uma única máquina, na literatura revisada tem comumente a seguinte proposta:

A variável de decisão x_{jb} é definida por:

$$x_{jb} = \begin{cases} 1 \ se\ a\ tarefa\ j\ é\ processada\ no\ lote\ b \\ 0 \qquad\qquad\qquad caso\ contrário \end{cases}$$

A formulação matemática inicial do problema proposto é:

$$Min\ C_{max} = \sum_{b=1}^{m} C_b \tag{1}$$

s.a.:

$$\sum_{b=1}^{m} x_{jb} = 1 \qquad j = 1,2,\dots,n \tag{2}$$

$$\sum_{j=1}^{n} s_j x_{jb} \leq S \qquad b = 1,2,\dots,n \tag{3}$$

$$p_j x_{jb} \leq C_b \qquad j = 1,2,\dots,n \quad b = 1,2,\dots,m \tag{4}$$

$$x_{jb} \in \{0,1\} \qquad j = 1,2,\dots,n \quad b = 1,2,\dots,m \tag{5}$$

$$C_b \geq 0 \qquad b = 1,2,\dots,m \tag{6}$$

A função objetivo (1) minimiza o *makespan*. O conjunto de restrições (2) garante que j é atribuído a apenas um lote. O conjunto de restrições (3) garante que a capacidade da máquina não é ultrapassada, quando são atribuídas tarefas ao lote. O conjunto de restrições (4) determina o tempo de processamento do lote b. O conjunto de restrições (5) e (6) especifica o tipo de restrição das variáveis de decisão.

9 APÊNDICE C

A capacidade total da máquina é *S*. O tempo de processamento do lote *b* é P^b e é dado pelo maior tempo de processamento entre todas as tarefas que constituem o lote; $P^b = max\,(p_j \ \forall j \in b)$. As variáveis de decisão X_{jb} e Y_b são definidas por;

X_{jb} = 1 se *j* é atribuído ao lote *b*, 0 caso contrário.

Y_b = 1 se o lote *b* é formado, 0 caso contrário.

Tendo a seguinte formulação matemática;

Minimizar

$$\sum_{b \in B} p^b \qquad \text{(i)}$$

Sujeito a:

$$\sum_{b \in B} X_{jb} = 1 \qquad \forall j \in J \qquad \text{(ii)}$$

$$\sum_{j \in J} s_j X_{jb} \leq S Y_b \qquad \forall b \in B \qquad \text{(iii)}$$

$$P^b \geq X_{jb} P_j \qquad \forall j \in J,\ b \in B \qquad \text{(iv)}$$

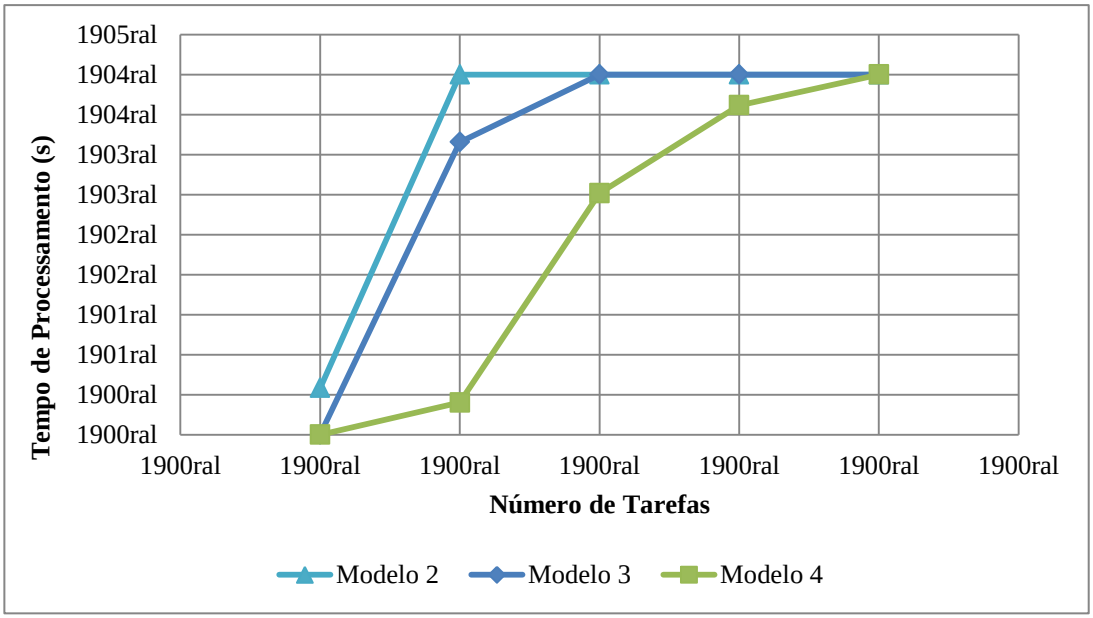
$$X_{jb} \leq Y_b \qquad \forall j \in J,\ b \in B \qquad \text{(v)}$$

$$X_{jb} \in \{0,1\} \qquad \forall j \in J,\ b \in B \qquad \text{(vi)}$$

$$Y_b \in \{0,1\} \qquad \forall j \in J,\ b \in B \qquad \text{(vii)}$$

A função objetivo (i) representa minimizar o *makespan*. O conjunto de restrições (ii) garante que a tarefa *j* só será atribuída a apenas um lote. O conjunto de restrições (iii) garante que a capacidade da máquina não é ultrapassada quando as tarefas são atribuídas a um lote. O conjunto de restrições (iv) encontra o tempo de processamento em lote para *b*. O conjunto de restrições (v) assegura que a tarefa *j* será atribuída a um único lote. As restrições (vi) e (vii) reforçam o binário de restrições sobre as variáveis de decisão.

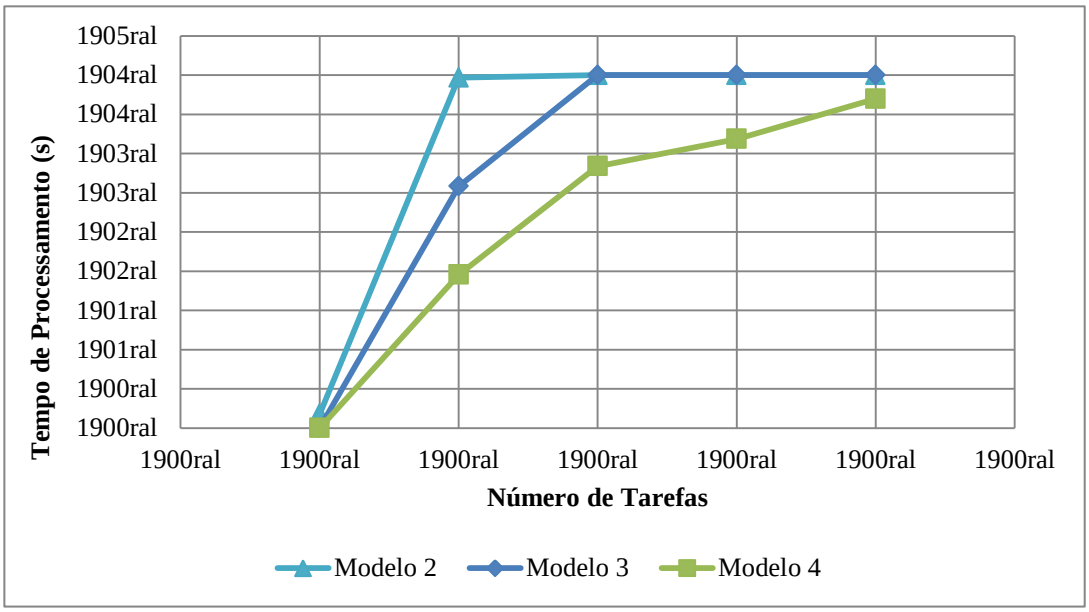
10 APÊNDICE D



Resultados dos modelos 2,3, e 4 para a instâncias 3 com $p_j \in [1,10]$

Neste gráfico, que representa a instância 3 para os Modelos 2, 3 e 4, também há superioridade do Modelo 4, onde somente na tarefa 100 é que o tempo extrapola os 1800 s ajustados, com tempo máximo de processamento. Presume-se que se o tempo máximo de processamento fosse um pouco maior o Modelo 4 também conseguiria encontrar o número de soluções comprovadamente ótimas.

11 APÊNDICE E



Resultados dos modelos 2,3, e 4 para a instâncias 3 com $p_j \in [1,5]$

Neste gráfico, que representa a instância 3 para os Modelos 2, 3 e 4, também há superioridade do Modelo 4. Embora muito semelhante ao gráfico do Apêndice C a tarefa 100 atinge o tempo de processamento de 1679,1 s e encontra uma solução comprovadamente ótima. O que vem de encontro com o exposto no Apêndice anterior, onde um tempo um pouco maior de *stup* pode levar o modelo a encontrar alguma solução.