Universidade Federal de Minas Gerais
Escola de Engenharia
Programa de Pós-Graduação em Engenharia de Produção

ALYSSON ANTHONY ALMEIDA MENDONÇA

Mestrado em Engenharia de Produção

Modelos e técnicas de *local branching* para o problema de abastecimento de linhas de montagem

Belo Horizonte – MG
Maio de 2011

Modelos e técnicas de *local branching* para o problema de abastecimento de linhas de montagem

Universidade Federal de Minas Gerais
Escola de Engenharia
Programa de Pós-Graduação em Engenharia de Produção

Modelos e técnicas de *local branching* para o problema de abastecimento de linhas de montagem

por

ALYSSON ANTHONY ALMEIDA MENDONÇA

Dissertação apresentada ao Programa de Pós-Graduação em Engenharia de Produção da Escola de Engenharia da Universidade Federal de Minas Gerais como requisito parcial à obtenção do título de Mestre em Engenharia de Produção.

Área de Concentração - Produção e Logística

Prof. Dr. Mauricio Cardoso de Souza
Orientador

Belo Horizonte – MG
Maio de 2011

Universidade Federal de Minas Gerais
Escola de Engenharia
Programa de Pós-Graduação em Engenharia de Produção

Modelos e técnicas de *local branching* para o problema de abastecimento de linhas de montagem

Este exemplar corresponde à versão final da dissertação.

Banca examinadora:

Prof. Dr. Alexandre Salles da Cunha - DCC-UFMG

Prof. Dr. Carlos Roberto Venâncio de Carvalho - DEP-UFMG

Prof. Dr. Mauricio Cardoso de Souza (orientador) - DEP-UFMG

Prof. Dr. Ricardo Saraiva de Camargo - DEP-UFMG

Belo Horizonte – MG
Maio de 2011

Este trabalho é dedicado a meus pais,
a meu irmão,
a meus amigos,
à Nathália
e a todos que nunca deixaram de me apoiar
durante a realização desta jornada.

Sanity and happiness are an impossible combination.

Mark Twain

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Siglas

JIT	Just-In-Time
LFP	Line Feed Problem
EOQ	Economic order quantity
ELSP	Economic lot scheduling problem
WW	Wagner-Whitin problem
CLSP	Capacitated Lot Sizing Problem
DLSP	Discrete lot sizing and scheduling problem
CSLP	Continuous setup lot sizing problem
PLSP	Proportional lot sizing and scheduling problem
VSBPP	Variable Size Bin Packing Problem
GRASP	Greedy Randomized Adaptive Search Procedure
AMPL	A Modeling Language for Mathematical Programming
MIP	Mixed Integer Problem
API	Application Programming Interface
RINS	Relaxation induced neighbourhood search
VNSB	Variable neighbourhood search branching
VNS	Variable neighbourhood search

Resumo

O problema de abastecimento de linhas de montagem pode ser observado na indústria automobilística, onde os sistemas de manufatura normalmente apresentam linhas de produção paralelas dedicadas à montagem de diferentes famílias de produtos. Neste contexto, um processo de abastecimento deve garantir o fornecimento dos itens necessários para as linhas de produção realizarem suas operações, sendo desejável a utilização de políticas de abastecimento que possam reduzir os custos associados ao processo. O problema objeto de estudo possui características de sequenciamento e dimensionamento de lotes, se apresentando como um problema de NP-difícil para o qual encontrar uma solução viável pode ser considerado um problema NP-completo. Este trabalho realiza uma comparação de diferentes formulações matemáticas para o problema, avaliando a influência das diferentes formulações em métodos de solução considerando instâncias de tamanho real. É realizado um ensaio sobre a utilização de desigualdades válidas para o problema propostas na literatura utilizando um algoritmo de planos de corte simples e avaliando o efeito da adição de cortes à formulação durante a solução do problema utilizando pacotes de otimização. São propostas três abordagens para a solução do problema baseadas na técnica de *local branching*, dentre as quais uma abordagem que gera um conjunto elite de soluções utilizando GRASP e que aproveita informações extraídas através da solução da relaxação linear do problema, abordagem que se destaca por apresentar resultados promissores e diversas oportunidades para a realização de trabalhos futuros.

Palavras-chave: abastecimento de linhas de montagem, modelagem matemática, otimização combinatória, *local branching*.

Abstract

The problem faced when feeding assembly lines can be seen in the automobile industry in which manufacturing systems usually have parallel production lines dedicated to assembling different product families. In this context, a feeding process must deliver the necessary items to ensure that the assembly lines can perform their operations, being highly desirable the use feeding policies that can reduce the costs associated with the feeding process. The problem evaluated in this study has similarities with lot sizing and sequencing problems, and it has been shown in the related literature that solving the problem is NP-hard whereas finding a valid solution can be considered NP-complete. This study makes a comparison of different mathematical formulations for the problem, assessing the influence of different formulations when solving real size instances of the problem. Experiments are made to evaluate the use of valid inequalities for the problem as proposed in the literature, using a simple cutting planes algorithm and evaluating the addition of cuts to the formulation during the solving process when optimization packages are used. The study proposes three approaches for solving the problem based on the local branching technique, one of which is an approach that generates a set of elite solutions using GRASP and also takes advantage of information extracted from solving the linear relaxation of the problem, an approach which stands out by showing promising results and several opportunities for conducting future work.

Keywords: line feeding problem, mathematical formulation, combinatorial optimization, local branching.

1 O problema de abastecimento de linhas de montagem

1.1 Contexto industrial

A padronização de métodos de trabalho em sistemas de manufatura é um importante fator a ser considerado em organizações que buscam atingir objetivos relacionados a alta produtividade ou redução de custo.

A organização do sistema de manufatura em linhas de produção e a utilização de metodologias como o *just-in-time* (JIT) são exemplos de estratégias de padronização que podem ser verificados em diversos segmentos da indústria, como por exemplo a indústria automobilística.

Em sistemas de manufatura da indústria automobilística as linhas de produção geralmente são dispostas em paralelo, onde cada linha de produção é dedicada à montagem de uma família de produtos. Cada linha possui centros de trabalho dispostos em série. Um processo de abastecimento deve fornecer todos os itens necessários para os centros de trabalho das linhas de realizarem suas operações. Os itens são transportados da área de estoque para as linhas de produção em containeres que possuem tamanhos pré-definidos.

As políticas utilizadas no processo de abastecimento podem afetar significativamente os custos relacionados ao manuseio de itens entre a área de estoque e os centros de trabalho. Tipicamente, os custos relevantes neste contexto possuem componentes relacionados à estocagem e ao manuseio dos itens.

No presente estudo foram utilizadas informações reais de uma indústria automobilística localizada no Brasil. O processo de produção é realizado em cinco estágios: mecânica, prensagem, funilaria, pintura e montagem final. O estágio de montagem final é composto por quatro linhas de produção independentes que produzem veículos de acordo com as características de cada família de produtos. Em cada linha, veículos que estão em produção permanecem um período pré-definido em cada centro de trabalho, sendo este

período denominado tempo de ciclo. O processo de abastecimento deve fornecer os itens necessários para as atividades de cada centro de trabalho considerando um horizonte de planejamento dividido em períodos. Por exemplo, um horizonte de planejamento considerando um intervalo de uma semana pode ser dividido em períodos de um dia. É possível formar estoque de itens na própria linha de produção, porém esta prática acarreta em custos que são calculados sobre a quantidade em estoque na linha de produção ao final de cada período.

O processo de abastecimento das linhas de produção da indústria possui as seguintes restrições operacionais:

1. Cada container transportado deve ser carregado com um único tipo de item.
2. Cada container transportado deve ser carregado com sua capacidade máxima, ou seja, a quantidade transportada por containeres de determinado tipo considerando um tipo de item específico é pré-definida.
3. Para cada tipo de item um único tipo de container deve ser utilizado para o transporte em todo o horizonte de programação.

A restrição de utilização da capacidade máxima de todos os containeres é típica de sistemas JIT [23]. Este conjunto de restrições é utilizado com o objetivo de diminuir variabilidades no processo de produção das linhas de produção.

O objetivo da política de abastecimento de linhas de produção é minimizar os custos de transporte de itens e de estocagem na linha de produção, atendendo à demanda de todos os centros de trabalho a cada período e observando as restrições operacionais. Apesar deste trabalho ser focado na indústria automobilística, a determinação de uma boa política de abastecimento de linhas de produção é relevante no contexto de diversas indústrias.

O problema como aqui descrito foi introduzido em [34] sendo denominado *Line Feed Problem* (LFP). O LFP possui importantes características relacionadas a outros problemas, como o empacotamento em mochilas e problemas de dimensionamento de lotes. O LFP é um problema de difícil solução computacional: a versão não-capacitada do LFP é NP-Difícil e mesmo encontrar uma solução viável para o LFP é um problema NP-Completo.

Uma formulação matemática inicial para o LFP foi apresentada em [34]. Uma formulação mais forte foi apresentada em [5], trabalho que propôs também uma família

de desigualdades para fortalecer a nova formulação. Até o presente momento não foram publicados trabalhos que solucionaram instâncias de tamanho real do LFP até a otimalidade.

Este estudo tem como objetivo avaliar diferentes formulações para o LFP, testar as desigualdades propostas em [5] no contexto de um algoritmo de Plano de Cortes simples e avaliar a utilização de técnicas de *local branching* na solução do problema.

As próximas seções deste trabalho estão organizadas da seguinte forma: a seção 2 apresenta uma revisão da bibliografia relacionada a problemas de dimensionamento de lotes e características do LFP; a seção 3 detalha diferentes modelos computacionais para o LFP e apresenta os resultados obtidos em instâncias de tamanho real apresentadas anteriormente na literatura; a seção 4 introduz a utilização de técnicas de *local branching* no contexto do LFP e descreve os resultados obtidos com sua utilização; a seção 5 sintetiza os resultados alcançados e sugere oportunidades para o desenvolvimento de trabalhos no futuro.

1.2 Objetivos

O primeiro objetivo para este trabalho é estudar e testar diferentes formulações matemáticas para o LFP, avaliando a influência das diferentes formulações em métodos de solução do problema considerando instâncias de tamanho real.

Um segundo objetivo é testar as desigualdades válidas para o LFP propostas em [5], analisando o impacto da utilização das desigualdades no processo de solução do LFP.

Adicionalmente, temos como objetivo propor e avaliar a utilização de técnicas de *local branching* na solução de instâncias de tamanho real do LFP.

2 Revisão da literatura

Nesta seção vamos apresentar a literatura relacionada ao LFP, assim como características do LFP que são similares a outros problemas.

2.1 Dimensionamento de lotes

Segundo Drexl e Kimms [8], sistemas de produção são frequentemente organizados em uma arquitetura composta por diversas células de produção, também chamadas de segmentos. As células de produção são por sua vez organizadas em linhas de produção ou em centros de trabalho, onde cada célula é capaz de realizar um conjunto de operações. Insumos e componentes fluem pelo sistema, onde são processados e manuseados até a entrega do produto final.

O planejamento da produção é um processo afetado por decisões de médio e curto prazo. As decisões de curto prazo determinam como planejar a produção de modo a atender pedidos e datas de entrega de todos os itens, sejam eles produtos finais ou componentes utilizados durante a produção. As decisões de médio prazo estão relacionadas à demanda, como por exemplo, tamanho e datas de entrega de pedidos, sendo relacionadas às decisões de curto prazo através do Plano Mestre de Produção.

Encontrar um plano de produção que atenda apenas à demanda e às restrições do sistema de produção normalmente não é suficiente, sendo necessário encontrar um plano que além de viável atenda a um ou mais objetivos, como por exemplo, minimizar custos associados à produção.

É necessário classificar os custos associados à produção de forma a identificar quais são mais relevantes aos sistemas de produção. Considerando custos de oportunidade do capital, é necessário diminuir a manutenção de estoques de forma a dirimir custos diretos de estocagem de itens. Se o sistema de produção possui recursos que são compartilhados, contendo máquinas por exemplo, pode ser necessário diminuir o número de

setups - alteração do item utilizado por determinado recurso - com o objetivo de diminuir atrasos na produção devido aos tempos de *setup*. Em sistemas onde custos de estocagem e de *setup* são relevantes, normalmente existe um *trade-off* entre produzir lotes maiores (diminuindo o número de *setups*) e produzir lotes menores (aumentando o número de *setups*). Se no sistema de produção existir a possibilidade de produzir determinado item ou o custo de *setup* for dependente da ordem em que os itens utilizam determinado recurso, teremos que decidir também a ordem em que os itens serão produzidos. Nesse caso, determinar a sequência e as quantidades em que os itens devem ser produzidos se torna um problema de sequenciamento e dimensionamento de lotes.

2.1.1 Características de problemas de dimensionamento de lotes

A complexidade de problemas de dimensionamento de lotes depende em grande parte das características do sistema de produção consideradas no modelo do problema. O trabalho de Karimi, Ghomi e Wilson [22] classifica as características que podem ser contempladas em um modelo de dimensionamento de lotes em horizonte de planejamento, número de níveis, número de produtos, restrições de capacidade ou recursos, deterioração de itens, demanda, estrutura de setup e falta de estoque.

O horizonte de planejamento determina o período de planejamento considerado pelo modelo, podendo ser classificado em finito ou infinito. Horizonte de planejamento finito normalmente é modelado em conjunto com demanda dinâmica, enquanto que o horizonte de planejamento infinito normalmente é utilizado em modelos que consideram a demanda estática. O sistema pode ser classificado também de acordo com a periodicidade de observações do sistema, podendo o sistema ser observado em períodos de tempo discretos ou contínuos. O período de tempo considerado categoriza o sistema como *small bucket* ou *big bucket*. Em sistemas *big bucket* os períodos de tempo geralmente têm duração da ordem de dias ou semanas, sendo possível produzir várias unidades dos itens em cada período. Em sistemas *small bucket* o período de tempo considerado é pequeno o suficiente para garantir que no máximo uma unidade de item poderá ser produzida a cada período, normalmente considerando períodos de tempo com duração de algumas horas. O horizonte de planejamento pode ser caracterizado também como horizonte rolante, normalmente utilizado quando as informações do modelo podem apresentar incerteza.

Sistemas de produção podem ser caracterizados quanto ao número de níveis como possuindo único nível ou múltiplos níveis. Em sistemas de único nível, os insumos são transformados em produtos finais após uma única operação, sem a necessidade de pro-

dução de itens intermediários. Em sistemas de múltiplos níveis, insumos precisam ser processados por diversas operações até serem transformados nos produtos finais, onde a saída de cada operação é a entrada para a próxima. Em geral, sistemas de múltiplos níveis são mais complexos do que sistemas de único nível.

Quanto ao número de itens, sistemas de produção são classificados como único produto ou múltiplos produtos, de acordo com o número de produtos finais considerados no sistema. Sistemas de múltiplos produtos geralmente são mais complexos do que sistemas com único produto.

Sistemas de produção podem ser classificados quanto a restrições sobre recursos, sendo então denominados capacitados ou não-capacitados. Problemas capacitados consideram um ou mais tipos de restrições sobre os recursos, como por exemplo, disponibilidade de mão-de-obra, equipamentos, máquinas e recursos financeiros. Modelos que consideram sistemas capacitados normalmente possuem solução mais difícil do que os de sistemas não-capacitados.

Sistemas que consideram a deterioração de itens possuem restrições sobre o período de tempo máximo no qual itens podem ser estocados. Modelos que consideram a deterioração de itens possuem maior complexidade do que sistemas que não consideram deterioração.

A demanda de um sistema pode ser caracterizada por diferentes critérios. Quando a demanda não varia com o tempo é denominada demanda estática, enquanto que se a demanda se altera com o tempo é denominada demanda dinâmica. A demanda é denominada determinística se o valor da demanda é conhecido previamente, independentemente de ser uma demanda estática ou dinâmica. Caso o valor da demanda seja dependente de probabilidades a demanda é denominada probabilística.

A demanda de cada item em sistemas com único nível pode ser obtida diretamente dos pedidos de clientes ou das previsões de demanda, sendo denominada então demanda independente. Em sistemas de múltiplos níveis, a demanda de itens em determinado nível depende da produção de outros níveis, sendo então denominada demanda dependente. Modelos que consideram sistemas com demanda dinâmica, probabilística ou dependente tendem a ser mais complexos do que sistemas que consideram demanda estática, determinística ou independente.

A existência de custos de *setup* em um sistema de produção normalmente leva à introdução de variáveis binárias no modelo, aumentando sua complexidade. Em sistemas onde custos de *setup* são considerados, a estrutura de *setups* pode ser classificada como

simples ou complexa. Se a duração e o custo de *setup* em um determinado período for independente da sequência e decisões dos períodos anteriores a estrutura de *setups* é denominada simples. Caso contrário, a estrutura de *setups* é denominada complexa. Sistemas que possuem estrutura complexa de *setups* normalmente levam a modelos de maior complexidade.

A possibilidade de falta de estoque em um sistema de produção pode ser modelada utilizando as abordagens de *backlogging* ou de *lost sales*. Em modelos que tratam a falta de estoque por *backlogging*, é possível atender a demanda de determinado período em períodos posteriores. Em modelos que tratam a falta de estoque por *lost sales*, existe a possibilidade de não atender à demanda. Existem modelos que utilizam as duas abordagens simultaneamente. Para modelar a falta de estoque, normalmente são adicionados à função objetivo *shortage costs*, custos que representam uma penalidade por postergar o atendimento ou não atender à demanda. Modelos que consideram a possibilidade de falta de estoque normalmente são mais complexos se comparados a modelos equivalentes que não consideram esta possibilidade.

2.1.2 Categorização de modelos de dimensionamento de lotes

Em 1913 foi proposto por Harris, F. o primeiro modelo para tratar o problema de dimensionamento de lotes, chamado de modelo do lote econômico (EOQ) [19] [18]. O modelo EOQ assume as premissas de que o sistema de produção possui nível único, item único, é não capacitado, possui demanda estática, apresentando horizonte de planejamento infinito e período de tempo contínuo. As premissas adotadas facilitam a solução do problema, porém são muito restritivas para representar grande parte dos sistemas de produção.

O modelo *Economic lot scheduling problem* (ELSP) foi proposto em 1958 por Rogers, J. [32] com o objetivo de representar melhor alguns sistemas de produção. Enquanto no EOQ o sistema de produção era não capacitado, no ELSP são consideradas restrições de capacidade do sistema de produção. Restrições de capacidade normalmente estão relacionadas a sistemas onde vários itens utilizam um número limitado de recursos compartilhados. O ELSP é utilizado para representar sistemas de produção com único nível, múltiplos produtos, demanda estática e horizonte de planejamento infinito com período de tempo contínuo. Encontrar a solução ótima do ELSP é um problema NP-Difícil [20].

O *Wagner-Whitin problem* (WW) foi proposto em 1958 por Wagner, H. e Whitin, T. [35] e considera um horizonte de planejamento finito, o qual é dividido em períodos

discretos. Considera também demanda dinâmica e é não capacitado, sendo o WW de único nível um problema de único produto.

As seguintes proposições são válidas para o WW:

- Existe uma solução ótima na qual produção é realizada apenas em períodos cujo estoque inicial é zero.
- Existe uma solução ótima na qual sempre que uma produção é realizada, sua quantidade irá atender exatamente à demanda de um conjunto de períodos consecutivos iniciados no período da produção.

O trabalho original apresentou um algoritmo baseado nas proposições para encontrar a solução ótima do WW utilizando programação dinâmica, com complexidade $O(n^2)$ onde n é o número de períodos considerados [35]. É possível transformar instâncias do WW em instâncias do problema de caminho mínimo, revelando que o WW pode ser resolvido de forma computacionalmente eficiente [1].

O *Problema de dimensionamento de lotes capacitado* (CLSP) pode ser visto como uma extensão do WW que considera restrições de capacidade. Assim como o ELSP, o CLSP é um problema que considera múltiplos produtos. O CLSP é um problema *big bucket* pois vários itens podem ser produzidos em um mesmo período, como por exemplo, em um período com duração de uma semana. Encontrar a solução ótima do CLSP é um problema NP-Difícil [12] [2]. Quando *setups* são adicionados ao modelo CLSP, encontrar uma solução viável se torna um problema NP-Completo [26].

Quando os períodos do CLSP são divididos em intervalos menores, encontramos o *Discrete lot sizing and scheduling problem* (DLSP). No DLSP, no máximo um item pode ser produzido a cada período, caracterizando o DLSP como um problema *small bucket*. Caso ocorra produção de item em um determinado período, esta produção deve utilizar a capacidade total do recurso. Devido à menor duração dos períodos (ex: horas), podem ser necessários vários períodos para realizar a produção de um único lote no DLSP. Para considerar *setups* no DLSP é necessário introduzir variáveis adicionais ao modelo indicado qual é a atribuição entre recursos e itens a cada período, observando que custos de *setup* são contabilizados apenas no início de cada lote e que a produção de um lote pode acontecer por vários períodos.

Assim como no CLSP, encontrar uma solução ótima para o DLSP é um problema NP-Difícil [33]. Encontrar uma solução viável tem complexidade polinomial quando *se-*

tups e máquinas paralelas não são consideradas no modelo, porém o DLSP apresenta complexidade NP-Completo caso algum destes fatores seja considerado.

Ao retirar do DLSP a necessidade de utilizar a capacidade total de recursos em períodos que possuem produção obtemos o *Continuous setup lot sizing problem* (CSLP). Uma diferença importante entre o DLSP e o CSLP é que quando existem períodos de ociosidade entre períodos de produção o DLSP contabiliza tempos de *setup* adicionais, perturbação que não ocorre no CSLP.

Permitir a existência de períodos onde ocorre produção sem a utilização da capacidade total dos recursos, como ocorre no CSLP, é equivalente a permitir capacidade ociosa em períodos de produção. O *Proportional lot sizing and scheduling problem* (PLSP), proposto por Drexl, A. e Haase, K. em 1995 [7], aborda este problema utilizando a capacidade ociosa de determinado período de produção para iniciar a produção de um segundo item no mesmo período.

2.1.3 Problema de dimensionamento de lotes não capacitado

O *Problema de dimensionamento de lotes não capacitado* (USILSP) é um dos problemas clássicos de dimensionamento de lotes. O problema trata da decisão de quando e quanto produzir de determinado item para atender a demandas em um horizonte de programação finito, sem restrições de capacidade, visando minimizar os custos de produção e estocagem. Wolsey, L. [36] apresenta o USILSP como uma reformulação inteira-mista do WW.

A seguir apresentamos a formulação para o USILSP descrita em [29]:

$$Min \sum_{t=1}^{n} (p_t x_t + f_t y_t + h_t s_t)$$

s.a.

$$\begin{aligned} s_{t-1} + x_t &= d_t + s_t && \forall t \\ s_0 &= s_n = 0 && \\ x_t &\leq M y_t && \forall t \\ x_t, s_t &\in \mathbb{R}^+ && \forall t \\ y_t &\in \{0,1\} && \forall t \end{aligned}$$

Nesta formulação x_t é o tamanho do lote produzido no período t, y_t indica se há

produção no período t, s_t indica a quantidade em estoque ao final do período t, d_t indica a demanda no período t, p_t é o custo de produção unitário no período t, f_t é o custo fixo se houver produção no período t, h_t é o custo unitário de estocagem ao final do período t. Existem diversas referências recentes sobre o USILSP ou suas variações.

Desigualdades que definem facetas da casca convexa da variação do USILSP que considera dois itens são estudadas em [37].

Uma descrição explícita da casca convexa da variação do USILSP que considera falta de estoque com *backlogging* é encontrada em [24].

A revisão apresentada em [3] descreve formulações, extensões e técnicas de solução para o USILSP.

2.2 Problema de empacotamento

Outro problema relacionado ao LFP é um tipo especial de problema de empacotamento com tamanho de *bin* variável (VSBPP). O VSBPP pode ser descrito como uma generalização do problema de empacotamento onde *bins* de diferentes capacidades e custos estão disponíveis para o empacotamento dos itens [4]. Uma formulação direta do VSBPP é apresentada a seguir:

$$Min \sum_{i \in I} f_i y_i$$

s.a.

$$\sum_{i \in I} x_{ij} = 1 \qquad j \in J$$

$$\sum_{j \in J} w_j x_{ij} \leq b_i y_i \qquad i \in I$$

$$x_{ij} \in \{0,1\} \qquad i \in I, j \in J$$

$$y_i \in \{0,1\} \qquad i \in I$$

Nesta formulação I é o conjunto de *bins*, J é o conjunto de itens, y_i é variável binária que indica se o *bin* i é utilizado na solução, f_i é o custo de utilizar o *bin* i na solução, x_{ij} é a variável que indica se o item$j$ será armazenado no *bin* i, w_j é o peso do item j, b_i é a capacidade do *bin* i.

Uma comparação entre seis heurísticas para o VSBPP e é realizada em [17], no qual é apresentada uma heurística baseada em *set covering* que apresenta boa qualidade das soluções a custo computacional reduzido.

Várias formulações e inequações para o VSBPP são apresentadas em [4] objetivando melhorar os limites da Relaxação Linear.

2.3 Abastecimento de linhas de montagem

Segundo Souza et al. [34], o LFP é o problema encontrado ao decidir como empacotar itens nos containeres disponíveis com o objetivo de atender às necessidades dos centros de trabalho a um custo mínimo no horizonte de planejamento considerado. O LFP pode ser considerado um VSBP que possui características especiais: restrição da quantidade disponível de cada *bin* e uma estrutura de custos tal que o custo de cada *bin* varia de acordo com os itens empacotados. A versão não-capacitada do LFP é NP-Difícil, tendo como um caso especial o *Change Making Problem* o qual é NP-Difícil mesmo em sua versão não-capacitada. Um caso especial do LFP, equivalente ao problema de decisão do *Maximum Cardinality Bin Packing Problem*, pode ser utilizado para demonstrar que encontrar uma solução viável para o LFP é um problema NP-Completo [34]. O trabalho propõe uma heurística GRASP para resolver o LFP, obtendo soluções de melhor qualidade do que a heurística gulosa utilizada na indústria estudada, além de apresentar soluções de melhor qualidade do que o pacote comercial CPLEX.

Em [5] é proposta uma formulação mais forte para o LFP do que a presente em [34], obtendo melhores limites na relaxações lineares e em algoritmos *branch-and-bound*. O trabalho propõe também desigualdades para o LFP, baseadas em desigualdades do USILSP, com o objetivo de fortalecer ainda mais a formulação proposta. É proposto um algoritmo para resolver o problema de separação das desigualdades para o LFP com complexidade $O(IKT^2)$ onde I é o número de itens, K o número de containeres e T o número de períodos considerados no horizonte de planejamento.

3 Modelos para o problema de abastecimento de linhas de montagem

3.1 Modelo original

O primeiro modelo proposto especificamente para o LFP foi apresentado em [34]. Uma instância do problema possui I tipos de item, K tipos de containeres e T períodos a serem considerados no horizonte de programação. São dados de entrada a demanda de cada item em cada período(d_{it}), o custo de transporte de cada tipo de container em cada período ($b_k t$), o custo de armazenamento na linha de produção por unidade de cada item (c_i) a capacidade de cada container considerando cada item (q_{ik}), a quantidade máxima de cada tipo de container a ser utilizada simultaneamente em qualquer período (l_k). As variáveis de decisão atribuem quais tipos de containeres devem ser utilizados para transportar quais tipos de item (x_{ik}), o número de viagens que o container do tipo k deve fazer para transportar o item i no período t (f_{ikt}) e quantas unidade de determinado tipo de item devem ser estocadas ao final de cada período (s_{it}). O número de containeres de determinado tipo necessários para atender à demanda em determinado período de um tipo de item é calculado por $w_{ikt} = \lceil d_{it} r_{ik}/q_i k \rceil$, onde r_{ik} é o *lead time* médio de abastecimento da linha de montagem utilizando o item i e o container k expresso em períodos. A constante M é um valor calculado para cada instância e corresponde ao maior número de viagens simultâneas que o menor container precisa realizar para atender à demanda acumulada de todos os períodos de algum tipo de item. O valor de M é calculado como $M = max\{\sum_{t=1}^{T} \lceil d_{it}/q_{i\bar{k}} \rceil : i = 1,..,I\}$, onde $\bar{k}$ é o tipo de container de menor capacidade.

A formulação matemática, doravante denominada Mod_1, é apresentada a seguir:

$$Min \sum_{t=1}^{T} \sum_{i=1}^{I} (c_i sit + \sum_{k=1}^{K} (b_{kt} f_{ikt})) \tag{3.1}$$

s.a.

$$s_{i,t-1} - s_{it} + \sum_{k=1}^{K} q_{ik} f_{ikt} = d_{it} \qquad \forall i, \forall t \tag{3.2}$$

$$\sum_{k=1}^{K} x_{ik} = 1 \qquad \forall i \tag{3.3}$$

$$\sum_{t=1}^{T} f_{ikt} - M x_{ik} \leq 0 \qquad \forall i, \forall k \tag{3.4}$$

$$\sum_{i=1}^{I} w_{ikt} x_{ik} \leq l_k \qquad \forall k, \forall t \tag{3.5}$$

$$f_{ikt} \in \mathbb{N}, x_{ik} \in \{0,1\}, s_{it} \in \mathbb{R}^+ \qquad \forall i, \forall k, \forall t \tag{3.6}$$

O objetivo 3.1 minimiza o somatório dos custos de estocagem e dos custos de transporte, observando que os custos de transporte podem ser constantes ou variar a cada período. A família de restrições 3.2 garante que a quantidade em estoque somada à quantidade transportada é suficiente para atender a demanda em todos os períodos. A família de restrições 3.3 define que cada tipo de item será transportado por um único tipo de container em todos os períodos. A família de restrições 3.4 faz o acoplamento entre as variáveis que determinam a quantidade transportada por período de determinado tipo de item em determinado tipo de container e as variáveis que atribuem um tipo de container a um tipo de item, restringindo também a quantidade máxima de containeres utilizada em qualquer período para qualquer tipo de container e qualquer tipo de item. A família de restrições 3.5 limita o número de containeres de determinado tipo que podem ser utilizados simultaneamente em qualquer período. A família de restrições 3.6 define o domínio das variáveis de decisão do modelo.

3.2 Modelo com eliminação de variáveis de estoque

A seguir apresentamos a reformulação do modelo Mod_1 eliminando as variáveis de estoque, como apresentado em [5]. O objetivo de utilizar este modelo foi verificar o impacto da eliminação de variáveis de estoque na Relaxação Linear e na solução do problema inteiro misto.

Pochet e Wolsey [30] mostram que a eliminação das variáveis de estoque no USILSP pode ser realizada substituindo cada variável de estoque s_t por $\sum_{k=1}^{t} x_k - \sum_{k=1}^{t} d_k$, onde s_t é o estoque ao final do período t, x_k é a quantidade produzida no período k e d_k é a demanda no período k. Considerando o LFP, a eliminação das variáveis de estoque pode ser realizada substituindo cada variável de estoque s_{it} por $\sum_{k=1}^{K} \sum_{t'=1}^{t} q_{ik} f_{ikt'} - \sum_{t'=1}^{t} d_{it'}$, resultado que pode ser verificado de forma construtiva:

Demonstração. O estoque de um item i ao final do primeiro período é definido por:

$$s_{i1} = \sum_{k=1}^{K} q_{ik} f_{ik1} - d_{i1}$$

Expressando s_{i2} em função de s_{i1} temos:

$$\begin{aligned}
s_{i2} &= \sum_{k=1}^{K} q_{ik} f_{ik2} - d_{i2} + s_{i1} \\
s_{i2} &= \sum_{k=1}^{K} q_{ik} f_{ik2} - d_{i2} + (\sum_{k=1}^{K} q_{ik} f_{ik1} - d_{i1}) \\
s_{i2} &= (\sum_{k=1}^{K} q_{ik} f_{ik2} + \sum_{k=1}^{K} q_{ik} f_{ik1}) - (d_{i2} + d_{i1})
\end{aligned}$$

Os períodos subsequentes podem então ser expressos por:

$$\begin{aligned}
s_{i3} &= (\sum_{k=1}^{K} q_{ik} f_{ik3} + \sum_{k=1}^{K} q_{ik} f_{ik2} + \sum_{k=1}^{K} q_{ik} f_{ik1}) - (d_{i3} + d_{i2} + d_{i1}) \\
s_{i4} &= (\sum_{k=1}^{K} q_{ik} f_{ik4} + \sum_{k=1}^{K} q_{ik} f_{ik3} + \sum_{k=1}^{K} q_{ik} f_{ik2} + \sum_{k=1}^{K} q_{ik} f_{ik1}) - (d_{i4} + d_{i3} + d_{i2} + d_{i1}) \\
&\vdots \\
s_{it} &= \sum_{k=1}^{K} q_{ik} f_{ikt} + \sum_{k=1}^{K} q_{ik} f_{ikt_{-1}} + \ldots + \sum_{k=1}^{K} q_{ik} f_{ik1}) - (d_{it} + d_{it_{-1}} + \ldots + d_{i1}) \\
s_{it} &= \sum_{k=1}^{K} \sum_{t'=1}^{t} q_{ik} f_{ikt'} - \sum_{t'=1}^{t} d_{it'}
\end{aligned}$$

□

A formulação do modelo após eliminação das variáveis de estoque, doravante denominado Mod_2, é apresentada a seguir:

$$Min \sum_{t=1}^{T} \sum_{i=1}^{I} \sum_{k=1}^{K} ((b_{kt} + (T-t+1)c_i q_{ik}) f_{ikt}) \tag{3.7}$$

s.a.

$$\sum_{t'=1}^{t} \sum_{k=1}^{K} q_{ik} f_{ikt'} \geq \sum_{t'=1}^{t} d_{it'} \qquad \forall i, \forall t \tag{3.8}$$

$$\sum_{k=1}^{K} x_{ik} = 1 \qquad \forall i \tag{3.9}$$

$$\sum_{t=1}^{T} f_{ikt} - M x_{ik} \leq 0 \qquad \forall i, \forall k \tag{3.10}$$

$$\sum_{i=1}^{I} w_{ikt} x_{ik} \leq l_k \qquad \forall k, \forall t \tag{3.11}$$

$$f_{ikt} \in \mathbb{N}, x_{ik} \in \{0,1\} \forall i, \forall k, \forall t \tag{3.12}$$

A função objetivo 3.7 e a restrição de atendimento à demanda são afetadas pela eliminação das variáveis de estoque. De maneira informal, a eliminação pode ser descrita como a substituição do estoque em determinado período pela demanda acumulada até o período subtraída da produção acumulada até o período. Por ser constante, o termo $-c_i(\sum_{t'=1}^{t} d_{it'})$ não foi considerado na função objetivo.

3.3 Refinamento do cálculo do parâmetro M

Com o objetivo de tornar a formulação Mod_1 mais forte, [5] apresenta uma desagregação da constante M para cada item, período e container. A nova família de constantes é dada por $M_{ikt} = \lceil \sum_{t'}^{T} = d_{it'}/q_{ik} \rceil$.

Considerando como modelo referência Mod_2 e substituindo a constante M presente na restrição 3.10 por M_{ikt} obtemos o modelo doravante denominado Mod_3, apresentado a seguir.

$$Min \sum_{t=1}^{T} \sum_{i=1}^{I} \sum_{k=1}^{K} ((b_{kt} + (T-t+1)c_i q_{ik}) f_{ikt}) \tag{3.13}$$

s.a.

$$\sum_{t'=1}^{t} \sum_{k=1}^{K} q_{ik} f_{ikt'} \geq \sum_{t'=1}^{t} d_{it'} \qquad \forall i, \forall t \tag{3.14}$$

$$\sum_{k=1}^{K} x_{ik} = 1 \qquad \forall i \tag{3.15}$$

$$\sum_{t=1}^{T} f_{ikt} - M_{ik1} x_{ik} \leq 0 \qquad \forall i, \forall k \tag{3.16}$$

$$f_{ikt} - M_{ikt} x_{ik} \leq 0 \qquad \forall i, \forall k, \forall t \tag{3.17}$$

$$\sum_{i=1}^{I} w_{ikt} x_{ik} \leq l_k \qquad \forall k, \forall t \tag{3.18}$$

$$f_{ikt} \in \mathbb{N}, x_{ik} \in \{0,1\}, \forall i, \forall k, \forall t \tag{3.19}$$

A desagregação da constante M descrita anteriormente resulta na família de restrições 3.17, que explicita limites mais justos à quantidade máxima de containeres necessária para atender à demanda de qualquer item utilizando qualquer container em qualquer período. A família de restrições 3.17 é suficiente para realizar o acoplamento entre as variáveis de atribuição e de transporte, porém é adicionada também a família de restrições 3.16 com o objetivo de melhorar o *upper bound* obtido durante a computação.

3.4 Variáveis de atribuição desagregadas por período

O modelo apresentado em [5] é baseado em modificações do modelo descrito na seção 3.1. As modificações sobre o modelo original consistiram da eliminação das variáveis de estoque, desagregação do limite de containeres transportados (M) por período e desagregação das variáveis de atribuição x em variáveis de atribuição por período, z.

A formulação doravante denominada Mod_4 é apresentada abaixo:

$$Min \sum_{t=1}^{T} \sum_{i=1}^{I} \sum_{k=1}^{K} ((b_{kt} + (T - t + 1)c_i q_{ik}) f_{ikt}) \tag{3.20}$$

s.a.

$$\sum_{t'=1}^{t} \sum_{k=1}^{K} q_{ik} f_{ikt'} \geq \sum_{t'=1}^{t} d_{it'}, \qquad \forall i, \forall t \tag{3.21}$$

$$\sum_{k=1}^{K} z_{ikt} \leq 1 \qquad \forall i, \forall t \tag{3.22}$$

$$z_{ikt} = z_{ikt-1} \qquad \forall i, \forall k, \forall t \in 2, ..., T \tag{3.23}$$

$$\sum_{t=1}^{T} f_{ikt} - M_{ik1} z_{ik1} \leq 0 \qquad \forall i, \forall k \tag{3.24}$$

$$f_{ikt} - M_{ikt} z_{ikt} \leq 0 \qquad \forall i, \forall k, \forall t \tag{3.25}$$

$$\sum_{i=1}^{I} w_{ikt} z_{ikt} \leq l_k \qquad \forall k, \forall t \tag{3.26}$$

$$f_{ikt} \in \mathbb{N}, z_{ikt} \in \{0, 1\}, \forall i, \forall k, \forall t \tag{3.27}$$

A função objetivo 3.20 e a família de restrições de atendimento à demanda são iguais às apresentadas no modelo 3.2. As famílias de restrições 3.22 e 3.24 são equivalentes à 3.3 do modelo original, onde as restrições 3.22 definem que apenas um container seja atribuído a um tipo de item em determinado período, enquanto as restrições 3.24 definem que mesma a atribuição de tipo de container a tipo de item deve ser utilizada em todos os períodos. As famílias de restrições 3.24 e 3.25 substituem as restrições 3.4, realizando o acoplamento entre as variáveis de atribuição e de quantidade transportada, além de restringir a quantidade máxima de containeres utilizada em qualquer período para qualquer tipo de container e qualquer tipo de item. As restrições 3.24 desagregam a quantidade máxima de containeres transportados com informação específica para cada tupla de período, tipo de container e tipo de item. As famílias de restrições 3.26 e 3.27 são equivalentes respectivamente às restrições 3.5 e 3.6, contendo a substituição das variáveis de atribuição x por variáveis de atribuição parametrizáveis por período, z.

3.5 Parâmetro M refinado incluindo variáveis de estoque

Após testes iniciais foi identificada a necessidade de verificar se os resultados obtidos com o modelo Mod_3 são mais influenciados pelo refinamento do parâmetro M (discutido em 3.3) ou pela eliminação das variáveis de estoque (discutida em 3.2).

A formulação baseada em Mod_1 incluindo o refinamento do parâmetro M, porém sem eliminar variáveis de estoque, doravante denominada Mod_5, é apresentada a seguir:

$$Min \sum_{t=1}^{T} \sum_{i=1}^{I} \sum_{k=1}^{K} b_{kt} f_{ikt} \tag{3.28}$$

s.a.

$$s_{i,t-1} - s_{it} + \sum_{k=1}^{K} q_{ik} f_{ikt} = d_{it} \qquad \forall i, \forall t \tag{3.29}$$

$$\sum_{k=1}^{K} x_{ik} = 1 \qquad \forall i \tag{3.30}$$

$$\sum_{t=1}^{T} f_{ikt} - M_{ik1} x_{ik} \leq 0 \qquad \forall i, \forall k \tag{3.31}$$

$$f_{ikt} - M_{ikt} x_{ik} \leq 0 \qquad \forall i, \forall k, \forall t \tag{3.32}$$

$$\sum_{i=1}^{I} w_{ikt} x_{ik} \leq l_k \qquad \forall k, \forall t \tag{3.33}$$

$$f_{ikt} \in \mathbb{N}, x_{ik} \in \{0,1\}, s_{it} \in \mathbb{R}^{+} \qquad \forall i, \forall k, \forall t \tag{3.34}$$

O modelo Mod_5 pode ser obtido a partir do modelo Mod_1 com a substituição da família de restrições 3.4 pelas famílias de restrições 3.31 e 3.32.

3.6 Desigualdades válidas para o LFP

O trabalho de Cunha e Souza [5] propõe uma família de desigualdades para fortalecer a formulação do LFP apresentada no mesmo trabalho. As denominadas desigualdades (i,L,S) são baseadas nas desigualdades que definem facetas no USILSP. As desigualdades (i,L,S) são definidas como:

$$\sum_{j\in L}\sum_{S}\sum_{k=1}^{K} f_{ikj}q_{ik} + \sum_{j\in S}\sum_{k=1}^{K} D(i,j,l)z_{ikj} \geq D(i,1,l) \tag{3.35}$$

Na equação 3.35 L é um conjunto de períodos $L = \{1,...,l\}$, S é um conjunto de períodos tal que $S \subseteq L$, f_{ikj} é a família de variáveis de transporte, z_{ikj} é a família de variáveis de atribuição, q_{ik} é a capacidade do container k ao transportar o item i e $D(i,t',t)$ é a demanda acumulada do item i a partir do período t', a qual é dada por $D(i,t',t) = \sum_{j=t'}^{t}(d_{ij})$.

Existe um número exponencial de desigualdades (i,L,S) para o LFP. Ao invés de explicitar todas as desigualdades na formulação é possível resolver em tempo polinomial um problema de separação para identificar as desigualdades (i,L,S) violadas por determinada solução da relaxação linear do UFLP.

Dado um vetor $(\bar{z},\bar{f})$ solução da relaxação linear do LFP o problema de separação pode ser resolvido em $O(IKT^2)$ operações, calculando-se para cada tupla de item i e período l o valor de α_{il}:

$$\alpha_{il} = \sum_{j=1}^{l} min\{\sum_{k=1}^{K} \bar{f}_{ikj}q_ik; \sum_{k=1}^{K} D(i,j,l)\bar{z}_{ikj}\} \tag{3.36}$$

Se para um determinado α_{il} temos que $\alpha_{il} < D(i,1,l)$ então a solução atual viola a desigualdade (i,L,S) dada por i, $L = \{1,...l\}$ e $S = \{j \in L : \sum_{k=1}^{K} \bar{f}_{ikj}q_ik > \sum_{k=1}^{K} D(i,j,l)\bar{z}_{ikj}\}$.

3.7 Experimentos Computacionais

3.7.1 Plataforma de testes

Os experimentos desta seção foram realizados em uma estação com processador Core2Duo 1.3Ghz, 4gb de memória Ram, sistema operacional Windows7 64 bits. Os modelos foram implementados em linguagem AMPL [13]. O pacote de otimização utilizado em conjunto com o AMPL para resolver as relaxações lineares e os MIP's foi o CPLEX 11 versão 11.1.1.

Para obter a solução das relaxações lineares o CPLEX foi configurado para ignorar a integralidade de variáveis de decisão utilizando a seguinte diretiva AMPL:

```
option cplex_options 'relax'
```

Inicialmente foram utilizadas as opções padrão do CPLEX para resolver os MIPS relacionados às formulações do LFP, porém para qualquer combinação de instância e modelo a execução do algoritmo era interrompida por falta de memória. Este problema foi solucionado com a adição da opção *nodefile=2*, a qual determina que a informação sobre nós da árvore de *branch-and-bound* deve ser guardada em disco. Com a nova opção qualquer combinação de instância e modelo pôde ser executada até em um limite de tempo de 2 horas, porém a maioria das instâncias finalizava sem que o CPLEX encontrasse uma solução inteira viável.

Após a realização de testes empíricos adicionamos as novas opções *mipemphasis=4* e *nodesel=0* ao CPLEX. A opção *mipemphasis=4* leva a uma estratégia alternativa de busca por soluções viáveis de alta qualidade, enquanto a opção *nodesel=0* prioriza a busca em profundidade na árvore *branch-and-bound* [21]. Foram adicionadas também as opções *return_mipgap=2* e *bestbound* para prover mais informações ao final da execução do *solver* e a opção *time=7200* para limitar o tempo de execução a 2 horas. Com as opções utilizadas, é possível que o CPLEX aplique os seguintes tipos de cortes: *Clique*, *Cover*, *Disjunctive*, *Flow Cover*, *Flow Path*, *Gomory*, *Generalized Upper Bound* (GUB), *Implied Bound*, *Mixed Integer Rounding* (MIR) e *Zero-Half*.

Para realizar a solução dos MIP's analisados e verificar o *upper bound* e *gap* resultantes foi utilizada a seguinte diretiva AMPL:

```
option cplex_options 'time=7200 return_mipgap=2 bestbound mipemphasis=4 nodefile=2 nodesel=0'
```

As instâncias avaliadas possuem 191 itens, 3 tipos de containeres e 7 períodos no horizonte de programação. As instâncias são descritas de forma detalhada em [34]. O nome de cada instância possui o formato *VVVBtest191B_EKY*, onde *VVV*, *B*, *E*, *K*, *Y* indicam as seguintes características da instância:

- *VVV*: Instância que consideram custos de transporte variáveis são identificadas pelo prefixo *var*. Instâncias que consideram custos de transporte constantes não possuem este prefixo.
- *B*: Em instâncias sem *b* a razão $(l_k \div a_k)$ possui o mesmo valor para cada container k, onde l_k é o percentual do número total a_k de containeres do tipo k necessários para abastecer a linha de montagem considerando que apenas containeres do tipo k

sejam utilizados. Em instâncias com b a razão $(l_k \div a_k)$ pode variar para cada tipo de container k.

- E: Pode assumir os valores 1 ou 2, indicando qual dos dois tipos de parametrização da função de demanda diária das linhas de montagem foi utilizada na instância.
- K: Pode assumir os valores 1 ou 2, indicando qual políticas para os custos de transporte foi utilizada na instância. A política 1 assume um mesmo custo de transporte para todos os containeres enquanto a política 2 assume que o custo de transporte de um container é proporcional à sua capacidade.
- Y: Pode assumir os valores 1, 2 ou 3 e indica qual de três conjuntos de configuração do parâmetro l_k foi utilizado na instância.

3.7.2 Resultados computacionais: custos de transporte constantes

As instâncias consideradas nesta seção resultam em 5921 variáveis para o Mod_1 (1337 não-inteiras); 4584 variáveis inteiras para o Mod_2, Mod_3 e Mod_5; 8022 variáveis inteiras para o Mod_4. As instâncias avaliadas possuem custos de transporte constantes.

Para cada modelo apresentamos nas tabelas Tabela 1, Tabela 2, Tabela 3, Tabela 4 e Tabela 5 o valor da relaxação linear (RL), o limite superior (UB), o limite inferior (LB) e o *gap* de otimalidade relativo em percentual (GAP) após a execução do CPLEX.

Para facilitar a comparação em termos de valores de relaxação linear apresentamos na Tabela 6 a razão entre os valores obtidos por cada modelo.

Instância	**RL**	**UB**	**LB**	**gap (%)**
test191_113	38.913,9	261.274,2	153.392,2	41,3
test191_121	69.268,4	202.029,8	147.689,6	26,9
test191_213	40.317,0	378.925,3	168.582,3	55,5
test191_222	71.767,7	223.990,3	158.886,3	29,1
btest191b_111	38.913,9	248.441,5	139.683,5	43,8
btest191b_113	38.913,9	510.509,5	127.941,5	74,9
btest191b_211	40.317,0	311.584,6	155.545,6	50,1
btest191b_213	40.317,0	367.910,9	140.401,9	61,8

Tabela 1: Resultados Mod_1 - custos de transporte constantes

O modelo Mod_5 não encontrou solução viável para nenhuma das oito instâncias dentro do tempo limite. O modelo Mod_5 é baseado em Mod_1, contendo adicionalmente

Instância	RL	UB	LB	gap (%)
test191_113	38.913,9	193.735,3	180.947,4	6,6
test191_121	69.268,4	177.881,7	175.172,0	1,5
test191_213	40.317,0	205.558,9	193.837,7	5,7
test191_222	71.767,7	190.478,5	186.038,0	2,3
btest191b_111	38.913,9	190.288,4	170.135,7	10,6
btest191b_113	38.913,9	197.879,7	168.732,5	14,7
btest191b_211	40.317,0	204.878,6	188.886,6	7,8
btest191b_213	40.317,0	205.784,9	179.870,4	12,6

Tabela 2: Resultados Mod_2 - custos de transporte constantes

Instância	RL	UB	LB	gap (%)
test191_113	67.964,2	194.207,2	186.746,9	3,8
test191_121	69.268,4	177.739,3	175.530,5	1,2
test191_213	71.313,5	205.494,8	199.903,9	2,7
test191_222	71.767,7	190.083,2	187.420,3	1,4
btest191b_111	54.274,4	188.236,1	174.918,7	7,1
btest191b_113	38.913,9	194.081,1	175.742,8	9,5
btest191b_211	66.457,2	203.387,9	194.185,3	4,5
btest191b_213	40.317,0	204.075,8	187.226,6	8,3

Tabela 3: Resultados Mod_3 - custos de transporte constantes

Instância	RL	UB	LB	gap (%)
test191_113	67.964,2	193.195,3	185.973,0	3,7
test191_121	69.268,4	177.750,4	175.468,6	1,3
test191_213	71.313,5	206.288,1	199.164,0	3,5
test191_222	71.767,7	190.170,7	187.397,3	1,5
btest191b_111	54.274,4	188.990,9	175.509,7	7,1
btest191b_113	38.913,9	194.277,0	175.381,4	9,7
btest191b_211	66.457,2	202.995,1	194.400,3	4,2
btest191b_213	40.317,0	204.897,3	185.253,6	9,6

Tabela 4: Resultados Mod_4 - custos de transporte constantes

Instância	RL5	UB	LB	GAP (%)
test191_113	67.964.2	N/A	N/A	N/A
test191_121	69.268.4	N/A	N/A	N/A
test191_213	71.313.5	N/A	N/A	N/A
test191_222	71.767.7	N/A	N/A	N/A
btest191b_111	54.274.4	N/A	N/A	N/A
btest191b_113	38.913,9	N/A	N/A	N/A
btest191b_211	66.457,2	N/A	N/A	N/A
btest191b_213	40.317,0	N/A	N/A	N/A

Tabela 5: Resultados Mod_5 - custos de transporte constantes

Instância	Mod_2/Mod_1	Mod_3/Mod_1	Mod_4/Mod_1	Mod_5/Mod_1
test191_113	1,00	1,75	1,75	1,75
test191_121	1,00	1,00	1,00	1,00
test191_213	1,00	1,77	1,77	1,77
test191_222	1,00	1,00	1,00	1,00
btest191b_111	1,00	1,39	1,39	1,39
btest191b_113	1,00	1,00	1,00	1,00
btest191b_211	1,00	1,65	1,65	1,65
btest191b_213	1,00	1,00	1,00	1,00
Valor Médio	1,00	1,32	1,32	1,32

Tabela 6: Comparação da relaxação linear - custos de transporte constantes

as restrições com refinamento de cálculo do parâmetro M existentes em Mod_3 e em Mod_4.

Os modelos Mod_1, Mod_2, Mod_3, Mod_4 obtiveram um *gap* de otimalidade médio nas instâncias testadas respectivamente de $47,9\%$, $7,7\%$, $4,8\%$ e $5,1\%$. Por não ter encontrado soluções viáveis para as instâncias consideradas não é possível analisar o *gap* de otimalidade de Mod_5.

Comparando o valor das relaxações lineares obtidas com Mod_1 e Mod_2 podemos observar que a eliminação das variáveis de estoque não alterou o valor da relaxação linear. Este resultado era esperado pois ao realizar a eliminação não adicionamos novas informações ao problema ou às variáveis de decisão remanescentes. As formulações Mod_3, Mod_4 e Mod_5 mostraram-se mais fortes que as anteriores, provendo limites da relaxação linear em média 32% superiores aos limites de Mod_1 e Mod_2. Este resultado era esperado pois estas formulações utilizam limites mais justos nas constantes M_{ikt} das restrições de acoplamento entre as variáveis de atribuição e movimentação (ver equações 3.16 e 3.17). Todas as relaxações lineares foram executadas em menos de 1 segundo, granularidade mínima de nossa medição.

Ao comparar o *gap* relativo de otimalidade percebemos que Mod_3 e Mod_4 apresentam *gaps* menores que Mod_1, Mod_2. Mod_3 obteve *gaps* melhores que Mod_4 em seis das oito instâncias testadas (test191_121, test191_213, test191_222, btest191b_111, btest191b_113, btest191b_213) o que pode ser explicado pelo maior número de variáveis presentes em Mod_4, dificultando a solução do problema. Não podemos afirmar com certeza que Mod_3 continuaria com *gaps* melhores que Mod_4 se o tempo de execução fosse aumentado nas instâncias com custos de transporte constantes.

Um comportamento inesperado foi mostrado nas execuções de Mod_2 pois estas

apresentaram um *gap* muito menor que Mod_1. Inicialmente suspeitamos que o menor número de variáveis de Mod_2 em relação a Mod_1 pudesse influenciar na convergência do CPLEX. É possível notar que as variáveis de estoque eliminadas eram não-inteiras, indício que esta eliminação não deveria afetar significativamente o processo de *branching*. Uma segunda hipótese é que a relaxação linear tornou-se mais rápida, permitindo que um número maior de nós da árvore de *branch-and-bound* fosse explorado no mesmo tempo, porém nos parece improvável que uma pequena diminuição no tempo de execução das relaxações lineares seja o responsável por uma diminuição de até uma ordem de grandeza (instâncias test191_121 e test191_222) no *gap*. Uma terceira hipótese é que a eliminação das variáveis de estoque tenha tornado a estrutura da formulação do Mod_2 mais adequada à aplicação de cortes mais efetivos pelo CPLEX.

3.7.3 Resultados computacionais: custos de transporte variáveis

As instâncias consideradas nesta seção resultam em 5921 variáveis para o Mod_1 (1337 não-inteiras); 4584 variáveis inteiras para o Mod_2, Mod_3 e Mod_5; 8022 variáveis inteiras para o Mod_4. As instâncias avaliadas possuem custos de transporte variáveis.

Para cada modelo apresentamos nas tabelas Tabela 7, Tabela 8, Tabela 9, Tabela 10 e Tabela 11 o valor da relaxação linear (RL), o limite superior (UB), o limite inferior (LB) e o *gap* de otimalidade relativo em percentual (GAP) após a execução do CPLEX. É apresentada também uma comparação em termos de valores de relaxação linear dos modelos na Tabela 12.

Instância	**RL1**	**UB**	**LB**	**GAP (%)**
vartest191_113	226.465,0	528.468,5	384.432,5	27,3
vartest191_121	354.999,2	N/A	N/A	N/A
vartest191_213	245.827,3	550.542,2	407.848,2	25,9
vartest191_222	383.573,9	N/A	N/A	N/A
varbtest191b_111	226.465,0	476.088,5	350.983,5	26,3
varbtest191b_113	226.465,0	N/A	N/A	N/A
varbtest191b_211	245.827,3	1.064.462,48	384.371,5	63,9
varbtest191b_213	245.827,3	501.032,4	367.259,4	26,7

Tabela 7: Resultados Mod_1 - custos de transporte variáveis

Os modelos Mod_1, Mod_2, Mod_3, Mod_4 e Mod_5 obtiveram um *gap* de otimalidade médio nas instâncias testadas respectivamente de $34,0\%$, $8,9\%$, $6,1\%$, $6,1\%$ e $16,6\%$.

Comparando os resultados apresentados para as instâncias que consideram custos

Instância	RL2	UB	LB	GAP (%)
vartest191_113	226.465,0	485.044,3	431.377,5	11,1
vartest191_121	354.999,2	465.864,9	460.355,7	1,2
vartest191_213	245.827,3	520.668,0	467.889,4	10,1
vartest191_222	383.573,9	504.462,4	497.229,2	1,4
varbtest191b_111	226.465,0	456.649,4	394.192,6	13,7
varbtest191b_113	226.465,0	429.528,8	380.272,2	11,5
varbtest191b_211	245.827,3	510.762,7	449.759,4	11,9
varbtest191b_213	245.827,3	455.077,3	408.760,9	10,2

Tabela 8: Resultados Mod_2 - custos de transporte variáveis

Instância	RL3	UB	LB	GAP (%)
vartest191_113	286.734,9	489.496,8	444.984,5	9,1
vartest191_121	354.999,2	465.251,5	461.192,4	0,9
vartest191_213	311.119,6	519.856,4	482.182,7	7,3
vartest191_222	383.573,9	503.675,3	498.260,4	1,1
varbtest191b_111	254.371,1	452.346,6	411.091,1	9,1
varbtest191b_113	226.465,0	427.356,6	393.143,4	8,0
varbtest191b_211	291.891,8	506.052,1	468.614,8	7,4
varbtest191b_213	245.827,3	452.795,4	425.354,4	6,1

Tabela 9: Resultados Mod_3 - custos de transporte variáveis

Instância	RL4	UB	LB	GAP (%)
vartest191_113	286.734,9	487.933,4	447.862,5	8,2
vartest191_121	354.999,2	465.227,6	461.438,9	0,8
vartest191_213	311.119,6	518.906,1	479.463,7	7,6
vartest191_222	383.573,9	504.516,2	498.333,8	1,2
varbtest191b_111	254.371,1	455.202,5	411.696,3	9,6
varbtest191b_113	226.465,0	427.531,9	393.817,0	7,9
varbtest191b_211	291.891,8	503.697,1	463.734,1	7,9
varbtest191b_213	245.827,3	452.823,9	428.220,2	5,4

Tabela 10: Resultados Mod_4 - custos de transporte variáveis

Instância	RL5	UB	LB	GAP (%)
vartest191_113	N/A	N/A	N/A	N/A
vartest191_121	N/A	N/A	N/A	N/A
vartest191_213	541.999,2	541.999,2	453.241,6	16,4
vartest191_222	N/A	N/A	N/A	N/A
varbtest191b_111	N/A	N/A	N/A	N/A
varbtest191b_113	N/A	N/A	N/A	N/A
varbtest191b_211	523.226,7	523.226,7	439.463,8	16,0
varbtest191b_213	468.752,8	468.752,8	387.233,6	17,4

Tabela 11: Resultados Mod_5 - custos de transporte variáveis

Instância	Mod_2/Mod_1	Mod_3/Mod_1	Mod_4/Mod_1	Mod_5/Mod_1
vartest191_113	1,00	1,27	1,27	1,27
vartest191_121	1,00	1,00	1,00	1,00
vartest191_213	1,00	1,27	1,27	1,27
vartest191_222	1,00	1,00	1,00	1,00
varbtest191b_111	1,00	1,12	1,12	1,12
varbtest191b_113	1,00	1,00	1,00	1,00
varbtest191b_211	1,00	1,19	1,19	1,19
varbtest191b_213	1,00	1,00	1,00	1,00
Valor Médio	1,00	1,11	1,11	1,11

Tabela 12: Comparação da relaxação linear - custos de transporte variáveis

de transporte variáveis com os resultados das instâncias que consideram custos de transporte constantes (ver seção 3.7.2), podemos perceber que o *gap* de otimalidade médio é maior nas instâncias que consideram custos de transportes variáveis, indicando que estas instâncias representam problemas de solução computacional mais difícil.

Assim como nas instâncias com custos de transporte constantes, comparando os valores das relaxações lineares de Mod_1 e Mod_2 podemos observar que a eliminação das variáveis de estoque não tem influência nos valores de relaxação linear. As formulações Mod_3, Mod_4 e Mod_5 apresentaram limites de relaxação linear que são em média 11% melhores que as formulações Mod_1 e Mod_2, percentual menor do que o observado ao na análise das instâncias com custos de transporte constantes. Todas as relaxações foram executadas em menos de 1 segundo, granularidade mínima de nossa medição.

Ao comparar o *gap* relativo de otimalidade percebemos que Mod_3 e Mod_4 apresentam *gaps* menores que Mod_1, Mod_2 e Mod_5. Mod_3 obteve *gaps* melhores que Mod_4 em quatro das oito instâncias testadas (test191_213, test191_222, btest191b_111, btest191b_211), não sendo possível inferir qual modelo apresentaria os melhores *gaps* caso o tempo de execução fosse aumentado nas instâncias com custos de transporte variáveis.

De forma semelhante ao observado nas instâncias com custos de transporte constantes, as execuções de Mod_2 apresentaram *gaps* menores que Mod_1.

3.7.4 Testes computacionais com desigualdades válidas do LFP

Algoritmo de planos de cortes simples

Para avaliar as desigualdades para o LFP propostas em [5], desenvolvemos um algoritmo de planos de cortes simples para resolver o modelo Mod_4. O algoritmo foi

desenvolvido em linguagem *C#* utilizando a API *ILOG CPLEX Concert* disponibilizada no CPLEX.

O algoritmo consiste em resolver a relaxação linear do modelo Mod_4 e resolver o problema de separação das desigualdades. Caso existam desigualdades violadas, estas são adicionadas como cortes à formulação original e o processo se repete; caso contrário o algoritmo termina.

Segue um pseudocódigo do algoritmo desenvolvido:

```
solucaoAtual = resolverRelaxacao( ModeloLFP );
cortes       = resolverSeparacao(solucaoAtual);

ENQUANTO( cortes > 0 )
{
    adicionarCortes(ModeloLFP, cortes);

    solucaoAtual = resolverRelaxacao( ModeloLFP );
    cortes       = resolverSeparacao(solucaoAtual);
}

FIM.
```

A função *resolverRelaxacao* do pseudocódigo obtém a solução ótima para a relaxação linear do modelo do LFP, sendo este modelo inicialmente igual a Mod_4. A função *resolverSeparacao* recebe como entrada a solução ótima da relaxação linear do LFP e identifica as desigualdades válidas violadas, utilizando o algoritmo de separação apresentado na seção 3.6. A função *adicionarCortes* adiciona as desigualdades identificadas como novas restrições ao modelo do LFP, sendo as restrições acumuladas entre iterações.

A seguir são apresentados os resultados obtidos realizando três iterações do algoritmo para a instância *test191_113*. Na Tabela 13 *Iteração* é o número da iteração do algoritmo de planos de corte, *Novos cortes* é o número total de cortes adicionados ao modelo na iteração, *Lower bound* é o valor da solução da relaxação linear na solução corrente.

É possível realizar a seguinte análise dos resultados apresentados na Tabela 13:

1. Após a solução da relaxação linear na primeira iteração, o algoritmo de separação de desigualdades identifica 1288 desigualdades violadas e adiciona as restrições correspondentes ao modelo.

Iteração	Novos cortes	Lower bound
1	1288	67964
2	21	67964
3	0	67964

Tabela 13: Utilização das desigualdades em um algoritmo de planos de corte simples

2. Após a solução da relaxação linear na segunda iteração, podemos perceber que 21 novas desigualdades violadas foram identificadas, totalizando um total de 1309 restrições adicionadas ao modelo. Os valores das variáveis de decisão na solução da segunda iteração são diferentes dos valores observados na solução da primeira iteração, porém a função objetivo apresenta o mesmo valor.

3. Após a solução da relaxação linear na terceira iteração, podemos perceber não foram identificadas novas desigualdades violadas. Os valores das variáveis de decisão na solução da terceira iteração são diferentes dos valores observados na solução da segunda iteração, porém a função objetivo ainda apresenta o mesmo valor.

Utilizamos como critério de parada a realização de três iterações pois ao final da terceira iteração não eram identificadas novas desigualdades violadas. O comportamento apresentado se repete para todas as instâncias que também foram utilizadas em testes nas seções 3.7.2 e 3.7.3.

Utilização das desigualdades para melhorar os limites durante a solução do MIP

O CPLEX disponibiliza uma interface que permite a consulta de informações e alteração do comportamento do algoritmo de solução. A interface permite a criação de *localCuts*: cortes que ficam armazenados em um pool e que podem ser aplicados ao modelo a qualquer momento durante o processo de otimização. A escolha de quais cortes presentes no pool devem ser aplicados e o momento em que os cortes devem ser adicionados ao modelo é realizada por um algoritmo interno do CPLEX.

Utilizando a interface provida pelo CPLEX, foi desenvolvida uma implementação na qual o CPLEX é utilizado para resolver a formulação inteira-mista Mod_4, porém após cada solução de relaxação linear realizada pelo algoritmo de solução do CPLEX as desigualdades violadas são identificadas e as restrições correspondentes são adicionadas ao pool de *localCuts*.

Segue pseudocódigo da implementação desenvolvida:

```
DECLARAR API_RELAXACAO_LINEAR(solucaoRelaxacao)
{
    cortes = resolverSeparacao(solucaoRelaxacao);
    adicionarLocalCuts(cortes);
}

CPLEX.usar(API_RELAXACAO_LINEAR);
CPLEX.resolverMIP( ModeloLFP );

FIM.
```

A diretiva *DECLARAR API_RELAXACAO_LINEAR* no pseudocódigo define um trecho de código para realizar a separação das desigualdades violadas e adicioná-las ao pool de *localCuts*. A diretiva *CPLEX.usar(API_RELAXACAO_LINEAR)* configura o CPLEX para executar o trecho de código declarado em *API_RELAXACAO_LINEAR* após qualquer relaxação linear realizada pelo algoritmo de solução do CPLEX. A diretiva *CPLEX.resolverMIP* inicia o processo de solução do LFP considerando a formulação inteira-mista.

Os resultados obtidos ao utilizar a implementação para resolver a instância *test191_113* utilizando um tempo limite de 1500 segundos são apresentados a seguir:

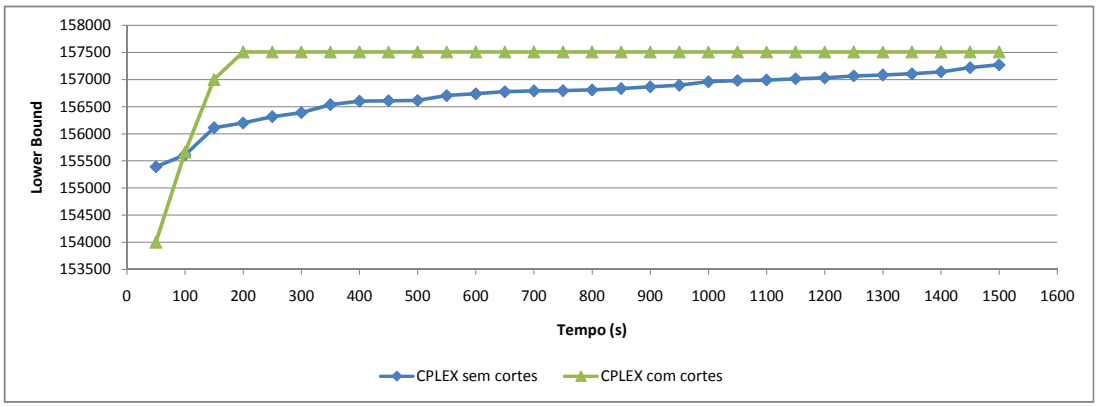

Figura 1: Efeito da utilização das desigualdades no lower bound

É possível perceber que ao adicionar *localCuts* utilizando as desigualdades identificadas na solução da relaxação linear, os limites inferiores (*lower bounds*) melhoram mais rapidamente. Por outro lado, os limites superiores (*upper bounds*) não são alterados, indicando que nenhuma solução inteira viável foi identificada durante a execução do CPLEX.

4 Local Branching

4.1 Referencial Teórico

4.1.1 Local branching clássico

O *local branching*, técnica proposta por Fischetti e Lodi [11], é um método de solução exato de MIPs. O *local branching* controla algum outro algoritmo de solução de MIPs exato, priorizando a exploração parcial ou completa da vizinhança de soluções viáveis já identificadas.

Mesmo considerando os pacotes de solução de MIPs mais recentes, instâncias de tamanho real de alguns problemas não são resolvidas otimamente em um tempo de computação aceitável. Nesse contexto, é preferível uma estratégia de solução que apresente boas soluções viáveis ainda no início da computação, possibilitando que soluções que estejam a uma distância aceitável da solução ótima sejam encontradas utilizando um limite de tempo menor.

Um aspecto importante é que o *local branching* melhora o comportamento heurístico do *solver* sem prejuízo à otimalidade do método, ou seja, utilizado em conjunto com o *solver* e dado o tempo de computação necessário o *local branching* irá encontrar uma solução comprovadamente ótima.

Heurísticas para fixação de variáveis

Uma das técnicas utilizadas em heurísticas para solução de MIPs é denominada *hard variable fixing*. Utilizando um método de solução de MIPs (heurístico ou exato) uma solução inicial para o problema é encontrada rapidamente parametrizando o método para interromper a execução após pouco esforço computacional. A solução obtida pode ser até mesmo inviável para o MIP, como por exemplo, a solução ótima da relaxação linear do problema. Observando os valores das variáveis na solução inicial parte das variáveis é fixada em valor inteiro, por exemplo, fixando variáveis nas quais o valor inteiro mais

próximo é não-nulo. O processo pode então ser repetido, observando que a cada iteração mais variáveis são fixadas em valores inteiros, diminuindo o tamanho do problema a cada iteração.

Um aspecto importante na técnica de *hard variable fixing* é a escolha de quantas e quais variáveis devem ser fixadas a cada iteração. Boas soluções normalmente são encontradas apenas após várias iterações fixando variáveis, sendo difícil determinar quais escolhas de fixação de variáveis das iterações anteriores foram ruins. Mesmo que escolhas de fixação ruins sejam percebidas, a criação de mecanismos para desfazer ou dirimir o impacto de escolhas de fixação ruins é uma tarefa complexa.

Como alternativa ao *hard variable fixing*, Fischetti e Lodi [11] propuseram uma técnica denominada *soft variable fixing*. Ao invés de escolher parte das variáveis e fixar seus valores, a técnica determina que uma proporção das variáveis de uma determinada solução viável deve ser fixada, delegando a tarefa de escolher as variáveis a serem fixadas para o *solver*. A motivação para realização do *soft variable fixing* é realizar a fixação de variáveis de forma tão efetiva quanto no *hard variable fixing*, porém permitindo um maior grau de liberdade para o *solver* com o objetivo de encontrar melhores soluções.

Supondo que exista uma solução viável para um MIP e que um percentual α das variáveis binárias não-nulas da solução devem ser fixadas, a seguinte restrição pode ser adicionada ao problema para realizar *soft variable fixing*:

$$\sum_{j=1}^{n} \bar{x}_j x_j \geq \lceil \alpha \sum_{j=1}^{n} \bar{x}_j \rceil \tag{4.1}$$

Na restrição 4.1 o conjunto de variáveis binárias possui n variáveis $x_j : j \in 1..n$ e os valores das variáveis na solução viável é dado pela família de constantes $\bar{x}_j : j \in 1..n$. A restrição força que ao menos um percentual α das variáveis binárias não-nulas permaneçam com o mesmo valor nas novas soluções, o que é equivalente a dizer que apenas um percentual $(1-\alpha)$ das variáveis binárias não-nulas na solução viável continuam livres para terem seu valor determinado pelo *solver*.

Detalhes do método

Para detalhar o funcionamento do *local branching* utilizamos a notação apresentada em [16] e o exemplo de formulação genérica de MIP proposto em [11]:

$$Minc^T x \tag{4.2}$$

s.a.

$$Ax \geq b \tag{4.3}$$

$$x_j \in \{0,1\} \forall j \in \beta \neq \emptyset \tag{4.4}$$

$$x_j \geq 0, inteiro \forall j \in \mathbb{G} \tag{4.5}$$

$$x_j \geq 0, \forall j \in \mathbb{C} \tag{4.6}$$

No modelo (4.2)-(4.6) o conjunto de variáveis $N = \{x_1, x_2, x_3, ..., x_n\}$ foi particionado nos subconjuntos β, $\mathbb{G}$ e $\mathbb{C}$ que correspondem respectivamente ao conjunto de variáveis binárias, conjunto de variáveis inteiras e conjunto de variáveis contínuas. O subconjunto β deve ser não-vazio, quanto os subconjuntos $\mathbb{G}$ e $\mathbb{C}$ podem ser vazios.

Considerando uma solução viável de referência $\bar{x}$ para (4.2)-(4.6), o conjunto $\bar{S} = \{j \in \beta : \bar{x}_j = 1\}$ define o suporte binário de $\bar{x}$. Para qualquer parâmetro k inteiro positivo, a k-ésima visinhança $N(\bar{x}, k)$ de $\bar{x}$ define um subproblema contendo o conjunto de soluções viáveis de (4.2)-(4.6) que satisfazem a seguinte restrição de *local branching*:

$$\delta(x, \bar{x}) : \sum_{j \in \bar{S}} (1 - x_j) + \sum_{j \in \beta \setminus \bar{S}} x_j \leq k \tag{4.7}$$

Na restrição 4.7 são contabilizadas as variáveis binárias que eram unitárias na solução de referência e se tornaram nulas, assim como as variáveis binárias que eram nulas e se tornaram unitárias. O número de "trocas"de valor nas variáveis binárias deve ser menor ou igual ao parâmetro inteiro k.

Para problemas onde a cardinalidade do suporte binário de qualquer solução viável de (4.2)-(4.6) é uma constante, a restrição 4.7 pode ser reescrita de forma simplificada:

$$\delta(x, \bar{x}) : \sum_{j \in \bar{S}} (1 - x_j) \leq k' \tag{4.8}$$

Na restrição 4.8 são contabilizadas apenas as variáveis que eram unitárias na solução de referência e posteriormente se tornaram nulas. Em problemas onde a cardinalidade

do suporte binário de qualquer solução viável é constante, sempre que é atribuído valor nulo a uma variável binária que antes era unitária é necessário também atribuir valor unitário a uma variável binária que antes era nulo. Devido a este comportamento a utilização da restrição 4.7 nestes problemas resultaria em uma contabilização de alterações em múltiplos de 2. Para estes problemas o efeito da utilização da restrição 4.7 ou 4.8 é o mesmo se $k' = k/2$.

A restrição de *local branching* pode ser utilizada como critério de *branching* em métodos enumerativos como o *branch-and-bound* e o *branch-and-cut*. Dada uma solução viável $\bar{x}$ o espaço de busca pode ser particionado utilizando a seguinte disjunção:

$$\delta(x,\bar{x}) \leq k(ramoesquerdo) OU \delta(x,\bar{x}) > k(ramodireito) \tag{4.9}$$

O tamanho inicial das vizinhanças, dado pelo parâmetro k, deve ser suficientemente grande para possibilitar que a vizinhança da solução viável contenha soluções melhores, porém precisa ser suficientemente pequeno para que seja possível explorar a vizinhança adequadamente considerando um pequeno limite de tempo.

O método de *local branching* é iniciado com a formulação original do problema e uma solução viável $\bar{x}_1$. É adicionada então uma restrição de *local branching* (ver restrição 4.8) centrada em $\bar{x}_1$ para limitar o espaço de busca à vizinhança de $\bar{x}_1$, o que corresponde ao ramo esquerdo da disjunção 4.9. O *solver* é então chamado para encontrar a solução ótima da vizinhança $\bar{x}_1$, $N(\bar{x}_1,k)$. Após a completa exploração da vizinhança a restrição de *local branching* 4.8 centrada em $\bar{x}_n$ é substituída pelo seu complemento, o que corresponde ao ramo direito da disjunção 4.9, evitando que a vizinhança seja explorada novamente nas iterações seguintes. No *local branching* clássico existem duas alternativas para o prosseguimento do método após a completa exploração de uma vizinhança qualquer $N(\bar{x}_n,k)$.

- Caso o ótimo local $\bar{x}_{n+1}$ da vizinhança $N(\bar{x}_n,k)$ seja melhor do que $\bar{x}_n$:

  É adicionada então uma nova restrição de *local branching* 4.8 centrada em $\bar{x}_{n+1}$ e o *solver* é chamado para explorar completamente a nova vizinhança $N(\bar{x}_{n+1},k)$ em uma nova iteração controlada pelo *local branching*.

- Caso o ótimo local $\bar{x}_{n+1}$ da vizinhança $N(\bar{x}_n, k)$ não seja melhor do que $\bar{x}_n$:

  O *solver* é chamado para explorar o restante do espaço de busca, deixando de ter ser comportamento controlado pelo *local branching*.

4.1.2 Extensões para o Local Branching

O trabalho de Fischetti e Lodi [11] propõe diversas extensões ao método de *local branching* clássico descrito na seção 4.1.1:

- *Limite de tempo e intensificação*: um novo parâmetro t_{max} limita o tempo de exploração das vizinhanças. Caso o limite de tempo t_{max} seja atingido sem encontrar a solução ótima da vizinhança $N(\bar{x_n}, k)$ podem acontecer dois casos:

  Caso a exploração tenha identificado uma solução viável $\bar{x}_{n+1}$ melhor que a solução atual $\bar{x}_n$, é criada uma nova vizinhança centrada em $\bar{x}_{n+1}$. É adicionada a restrição de *local branching* 4.8 centrada em $\bar{x}_{n+1}$ e removida a restrição centrada em $\bar{x}_n$ sem adicionar seu complemento. A adição do complemento da restrição centrada $\bar{x}_n$ iria remover a vizinhança $N(\bar{x}_n, k)$ do espaço de busca, o que seria matematicamente incorreto pois a vizinhança ainda pode conter o ótimo global.

  Caso a exploração não tenha identificado uma solução viável $\bar{x}_{n+1}$ melhor que a solução atual $\bar{x}_n$ o parâmetro k é reduzido, facilitando a determinação do ótimo local dentro do limite de tempo por realizar a exploração de uma vizinhança menor.

- *Diversificação*: pode ser utilizado quando a exploração de uma vizinhança $N(\bar{x}_n, k)$ não encontrou uma solução viável $\bar{x}_{n+1}$ melhor que a solução atual $\bar{x}_n$. Ao invés de chamar o *solver* para explorar o restante do espaço de busca o parâmetro k é aumentado, possibilitando a identificação de uma solução viável melhor que a atual por realizar a exploração de uma vizinhança maior. Este tipo de diversificação é denominado *soft diversification*.

  Após o aumento do parâmetro k é possível que ainda assim não seja encontrada uma solução viável melhor que a atual. Nesse caso o *local branching* é reiniciado utilizando uma vizinhança centrada em uma solução pior do que a solução atual, permitindo a exploração de uma região diferente do espaço de buscas. Este tipo de diversificação é denominado *strong diversification*.

- *Utilização de soluções não-viáveis*: para problemas onde determinar uma solução viável inicial é difícil, é sugerida a criação de variáveis de folga para parte do conjunto de restrições, penalizando a inviabilidade na função objetivo.

- *local branching em variáveis inteiras não-binárias*: para que seja possível realizar o *branching* em variáveis inteiras não-binárias é necessário criar variáveis de folga para representar a variação de cada variável inteira dentro de limites superior e inferior. A restrição de *local branching* precisa ser adaptada para contemplar esta alteração como detalhado em [11].

- *utilização de solução inicial inviável*: em sua descrição original, é necessário uma solução viável para iniciar a computação utilizando o *local branching*. O método pode ser adaptado para tratar soluções iniciais inviáveis.

Uma outra extensão proposta para o *local branching* é o *Relaxation induced neighbourhood search* (RINS), apresentado por Danna et al. [6]. O RINS realiza a fixação de variáveis comparando os valores atribuídos às variáveis binárias em determinada solução viável e os valores que estas variáveis assumem na relaxação linear do subproblema atual. RINS explora a idéia de que fixar o valor de variáveis que possuem o mesmo valor na solução viável e na relaxação do subproblema atual provavelmente leva a boas soluções, concentrando o esforço computacional do *solver* em variáveis que diferem na solução viável e na relaxação linear, sendo as variáveis que permaneceram livres as que intuitivamente podem causar uma melhoria mais significativa na função objetivo.

Local branching e Variable neighborhood search

Segundo Lazic et al. [25], o *local branching* pode ser visto também como uma especialização do *variable neighbourhood search branching* (VNSB), método apresentado por Hansen et al. [16] para resolver MIPs baseando-se em *local branching* e *variable neighborhood search* (VNS) [28].

O VNS realiza a troca de vizinhanças de uma forma sistemática em um algoritmo de busca local, explorando vizinhanças cada vez mais distantes da melhor solução inteira conhecida e ao mesmo tempo evitando que o algoritmo fique estagnado na vizinhança de ótimos locais. O VNSB utiliza idéias do VNS para direcionar a exploração de vizinhanças, porém as vizinhanças são definidas adicionando restrições ao problema de forma semelhante ao *local branching*.

As diferenças entre VNSB e *local branching* podem ser enumeradas em:

1. No passo de busca local do *local branching* é explorada uma vizinhança de tamanho $K > 1$, onde K é um parâmetro do *local branching*. No VNSB é utilizado implicitamente $K = 1$.

2. A intensificação da busca no *local branching* é realizada reduzindo a vizinhança a ser explorada pela metade, ou seja, a sequência de valores de K será dada por $K, K/2, ..., K/2^n$. O equivalente no VNSB é a utilização de *backward VNS*, onde o K inicial possui valor máximo e é reduzido uma unidade por iteração até atingir o valor unitário, resultando na sequência de valores de K dada por $K_max, K_{max-1}, ..., K_n = 1$.

3. A diversificação no *local branching* escolhe aleatoriamente uma nova solução viável a partir de uma vizinhança dada por um disco no espaço de busca com centro na solução atual e raio $R_{lb} = \{r \in \Re : 1 \leq r \leq K + dv(K/2)\}$, onde dv é o número de diversificações realizadas. A diversificação no VNSB escolhe aleatoriamente uma nova solução viável a partir de uma vizinhança dada por um disco no espaço de busca com centro na solução atual e raio $R_{vnsb} = \{r \in \mathbb{R}^+ : K_{atual} \leq r \leq K_{atual} + K_{passo}\}$, onde K_{atual} define a vizinhança atual e K_{passo} é um parâmetro do VNSB.

4.2 Local Branching aplicado ao LFP

Com o objetivo de verificar o comportamento do *local branching* quando aplicado ao LFP, foram realizadas três implementações utilizando o modelo Mod_3 do LFP:

- *Local branching clássico*: implementação de *local branching* utilizando a abordagem proposta por Fischetti e Lodi em [11].
- *Local branching com fixação de variáveis*: implementação de *local branching* utilizando idéias da heurística para fixação de variáveis *Hard Variable Fixing*.
- *Local branching com intensificação baseada em soluções GRASP*: implementação de *local branching* proposta no trabalho que realiza a fixação de variáveis e intensificação observando soluções obtidas através do algoritmo GRASP específico para o LFP proposto em [34].

4.2.1 Local Branching clássico

A implementação *local branching clássico* utilizou como base o framework *ALB++* proposto por Martinez e Cunha [27]. O *ALB++* é um framework desenvolvido em C++ com o objetivo de permitir a implementação de algoritmos baseados em *local branching* a problemas combinatórios.

O *ALB++* fornece uma implementação de *local branching* já contendo a extensão *Limite de tempo e intensificação* e a extensão *Diversificação* por *soft diversification* (ver seção 4.1.2). No contexto deste trabalho, a implementação de referência do *local branching* existente no *ALB++* foi modificada, sendo adicionadas as seguintes extensões:

1. *Restrição de local branching para suporte binário de cardinalidade constante*: a implementação original do *ALB++* utiliza a restrição de *local branching* na forma geral, dada por 4.7. O LFP apresenta suporte binário com cardinalidade constante pois em qualquer solução viável todos os itens serão atribuídos a um único container, sendo possível então utilizar para este trabalho a restrição de *local branching* simplificada dada por 4.8.

2. *Formulação do LFP*: foram implementadas classes no padrão do framework *ALB++* contendo a formulação Mod_3 do LFP (ver seção 3.3).

3. *Mudança de coordenadas das variáveis binárias do LFP*: As variáveis binárias de atribuição da formulação Mod_3 do LFP, x_{ik}, possuem as dimensões de *item* e *container*. A implementação do framework *ALB++* permite o *branching* apenas sobre uma família de variáveis binárias unidimensional, sendo necessário então alterar as coordenadas da família de variáveis x_{ik}.

Para realizar a mudança de coordenadas das variáveis binárias do LFP as coordenadas bidimensionais das variáveis de atribuição x_{ik} precisam ser transformadas para coordenadas unidimensionais, x_j. Esta transformação foi realizada utilizando a seguinte equação:

$$j = i + (k * I) \tag{4.10}$$

Na equação 4.10 j representa o índice unidimensional de uma variável de atribuição x_{ik}, i é o número do item, k é o número do container e I é o número total de itens.

A mudança de coordenadas da família de variáveis f_{ikt} também foi necessária internamente na implementação do trabalho, resultando na família de restrições f_h. A transformação foi realizada através da seguinte equação:

$$h = i + I * (k + t * K) \tag{4.11}$$

Na equação 4.11 h representa o índice unidimensional de uma variável de atribuição f_{ikt}, i é o número do item, I é o número total de itens, k é o número do container, t é o número do período e K é o número total de containeres.

$$\sum_{j \in J} (1 - x_j) \leq k' \tag{4.12}$$

Nas implementações de *local branching* para o LFP propostas neste trabalho a equação 4.12 é utilizada como restrição de *local branching*, onde j é a coordenada das variáveis de atribuição após a mudança de coordenadas da função 4.10, J é o subconjunto das variáveis x_j que foram atribuídas ao valor 1 na melhor solução viável encontrada até o momento e k' é o tamanho inicial das vizinhanças a serem exploradas.

Utilizando como exemplo o modelo Mod_3 (ver 3.13) é possível perceber que a cardinalidade do conjunto J sempre é igual ao número de itens existentes na instância do LFP. De acordo com a família de restrições 3.15 cada item deve ser atribuído necessariamente a um único container durante todo o horizonte de planejamento. Considerando a restrição de *local branching* 4.12, cada vez que uma variável de atribuição $x_{j'}$ associada a um item i tem seu valor alterado de 1 para 0, necessariamente uma outra variável de atribuição $x_{j''}$ que considera o mesmo item i deve ter seu valor alterado de 0 para 1, indicando uma nova atribuição do item i a algum container. No contexto do *local branching*, a restrição 4.12 indica que cada vizinhança a ser explorada é definida por uma região centrada na melhor solução viável encontrada até o momento, e que a região contém soluções que possuam até k' atribuições de item a container diferentes da melhor solução viável encontrada até o momento.

4.2.2 Local branching com fixação de variáveis

Foi realizada uma segunda implementação de *local branching*, utilizando a abordagem *Hard Variable Fixing* de fixação de variáveis, com o objetivo de diminuir o tamanho dos problemas a serem resolvidos durante a exploração das vizinhanças das soluções viáveis.

O *local branching* foi dividido em passos:

1. Definição do conjunto de variáveis que podem ser fixadas, doravante denominado $\bar{X}$.
2. Obtenção de uma solução inicial viável para o LFP.
3. Fixação de parte das variáveis do conjunto $\bar{X}$.
4. Execução do *local branching*.

No passo 1 é encontrada a solução para relaxação linear da instância de LFP. O conjunto $\bar{X}$ é definido de forma que contenha as variáveis binárias que são estritamente positivas na solução da relaxação linear, ou seja, dado o conjunto de variáveis binárias da solução da relaxação linear $\tilde{X}$ podemos definir $\bar{X} = \{x \in \tilde{X} : x > 0\}$. Uma descrição do conjunto $\bar{X}$ é persistida em arquivo-texto para consulta posterior. Este passo é realizado uma única vez para cada instância do LFP.

No passo 2 é encontrada uma solução inteira viável para a instância do LFP executando o CPLEX com uma parametrização que interrompa a otimização quando a primeira solução inteira viável é encontrada, comportamento determinado pela opção $IntSolLim = 1$ do CPLEX. Uma descrição da solução inteira encontrada é persistida em arquivo-texto para consulta posterior. Este passo é realizado uma única vez para cada instância do LFP.

No passo 3 o conjunto $\bar{X}$ é recuperado do arquivo-texto persistido no passo 1 e parte das variáveis do conjunto é escolhida para ser fixada. Um parâmetro para o método de *local branching* apresentado é o fator percentual de variáveis binárias a serem fixadas inicialmente. Outro parâmetro para o método de *local branching* apresentado é a semente utilizada para realizar a escolha aleatória de variáveis a serem fixadas.

Cada variável escolhida no passo 3 é fixada no valor atribuído pela solução encontrada no passo 2, sendo a fixação realizada pela adição de restrições ao modelo da forma $\{\bar{x}_i \in \bar{X} : \bar{x}_i = \bar{s}_i\}$ onde $\bar{s}_i$ é o valor atribuído à variável $\bar{x}_i$ na solução inteira encontrada no passo 2. Este passo é realizado para cada execução do método.

No passo 4 a solução inteira encontrada no passo 2 é utilizada como solução inicial do *local branching* clássico. A partir deste ponto o método se comporta como o *local branching* clássico, observando que um número menor de variáveis binárias continua livre para a execução do algoritmo.

Segue o pseudocódigo do algoritmo desenvolvido:

```
// Passo1
solucaoRelaxacao = ResolverRelaxacaoMod_3()
variaveisAFixar = ObterVariaveisAFixar(solucaoRelaxacao)

// Passo2
solucaoViavel = ObterSolucaoViavelMod_3()

// Passo3
FixarVariaveis(variaveisAFixar, solucaoViavel)

// Passo4
ExecutarLocalBranching(solucaoViavel)

FIM.
```

4.2.3 Local branching com intensificação baseada em soluções GRASP

Foi realizada uma terceira implementação de *local branching*, baseada livremente nas idéias de *Hard Variable Fixing* (ver seção 4.1.1) e *Path-Relinking*, técnica proposta inicialmente por Glover [14] e discutida em [15] e [31], como uma estratégia de intensificação para explorar trajetórias definidas a partir de pares de soluções do conjunto elite. A técnica de *Path-Relinking* apresenta bons resultados em várias problemas, como descrito no trabalho de Resende e Ribeiro [31].

A abordagem de *local branching* proposta realiza intensificação sobre pares de soluções do conjunto elite e diversificação observando informações da relaxação linear. A abordagem contempla vários passos:

1. *Passo1 - Computação do conjunto de elite*:

   Computação prévia de um conjunto de elite contendo soluções encontradas utilizando *GRASP* [34].

2. *Passo2 - Determinação do conjunto inicial de variáveis fixadas*:

   Escolha de um par de soluções do conjunto de elite e definição do conjunto inicial de variáveis fixadas.

3. *Passo3 - Execução do local branching utilizando nova diversificação*:

   Solução da relaxação linear e determinação da lista ordenada de itens candidatos a

diversificação. Execução do *local branching* utilizando a diversificação baseada na lista ordenada de itens candidatos à diversificação computada no passo Passo2.

Segue o pseudocódigo do algoritmo desenvolvido:

```
// Passo1 - Computação do conjunto de elite
conjuntoEliteMod_3 = ConverterConjuntoElite(conjuntoEliteMod_1)

// Passo2 - Determinação do conjunto inicial de variáveis fixadas
PARA CADA PAR DE SOLUÇÕES (solucao1, solucao2) EM conjuntoEliteMod_3
REPETE
    listaVariaveisAFixar = ObterListaVariaveisAFixar(solucao1, solucao2)
    restricoesFixacao = FixarVariaveis(listaVariaveisAFixar)

    // Passo3 - Execução do local branching utilizando nova diversificação
    solucaoRelaxacao = ResolverRelaxacaoMod_3()
    listaItensDiversificacao = CriarListaItensDiversificacao(solucaoRelaxacao)

    melhorSolucao = EscolherMelhorSolucao(solucao1, solucao2)
    ExecutarLocalBranching(melhorSolucao, listaItensDiversificacao, restricoesFixacao)
FIM REPETE

FIM.
```

Cada passo da abordagem proposta é detalhado nas seções subsequentes.

Passo1 - Computação do conjunto de elite

Neste passo é computado um conjunto de elite utilizando a heurística *GRASP* proposta para o LFP por Souza et al. em [34], no contexto da formulação Mod_1 do LFP. *GRASP* é uma técnica proposta por Feo e Resende [9] [10] que possui características gulosa, probabilística e adaptativa, tendo sido utilizada com sucesso para obter boas soluções em vários problemas combinatórios.

Durante a execução do *GRASP* duas etapas principais são repetidas: a etapa de construção de uma solução viável e a etapa de busca local.

A etapa de construção de uma solução viável para o LFP consiste de um *loop* que é executado até que todos os itens tenham sido atribuídos a algum container ou até que seja determinado que a solução em construção não é viável, o que ocorre quando são necessários mais containeres do que os disponíveis (restrição 3.5) para finalizar a atribuição de itens

a containeres. Durante a atribuição de itens a containeres é realizada uma escolha gulosa e aleatória, escolhendo aleatoriamente uma atribuição dentre as atribuições que possuem menor custo incremental. A complexidade computacional da construção de uma solução viável para o LFP é $O(I^2KT)$, onde I é o número de itens, K é o número de containeres e T é o número de períodos.

A etapa de busca local consiste de, para cada item, verificar se é possível substituir o container atribuído na etapa de construção de solução viável por um container diferente. Caso não existam containeres suficientes para realizar a substituição, o procedimento verifica se é possível trocar o container atribuído ao item analisado pelo container atribuído a algum outro item. A busca local continua até que um ótimo local seja encontrado. A complexidade computacional do procedimento é de $O(I^2KT)$.

O *GRASP* foi executado para cada instância do LFP com um limite máximo de 500 iterações. Durante cada execução foi separado um conjunto de soluções denominado conjunto elite, contendo soluções a serem utilizadas como solução inicial do *local branching*. O conjunto elite é formado com no máximo 5 soluções. Os critérios para decidir se uma solução candidata, encontrada na iteração atual do *GRASP*, deve ou não ser adicionada ao conjunto elite depende do número de soluções que já compõem o conjunto elite.

Enquanto o conjunto elite possui menos de 5 soluções, uma solução candidata é admitida no conjunto elite caso atenda a pelo menos um dos seguintes critérios:

1. A solução candidata possui o melhor valor de função objetivo encontrado até a iteração atual do *GRASP*.

2. As atribuições de item a container são diferentes em ao menos 10% quando as atribuições da solução candidata são comparadas às atribuições de cada solução já presente no conjunto elite.

   Supondo instâncias onde são considerados 191 itens, uma solução candidata atende a este critério se ao menos 39 dos 191 itens estão atribuídos a containeres diferentes quando a solução candidata é comparada par-a-par a cada solução que já pertence ao conjunto elite.

Enquanto o conjunto elite possui 5 soluções, uma solução candidata é admitida no conjunto elite caso atenda a pelo menos um dos seguintes critérios:

1. A solução candidata possui valor de função objetivo melhor do que todas as soluções que pertencem ao conjunto elite.

2. As atribuições de item a container são diferentes em ao menos 10% quando as atribuições da solução candidata são comparadas às atribuições de cada solução já presente no conjunto elite. Adicionalmente, é necessário que a solução candidata possua valor de função objetivo melhor do que ao menos uma solução do conjunto elite.

Para o caso em que uma solução candidata é admitida no conjunto elite quando este já possui 5 soluções, é retirada do conjunto elite a solução de pior função objetivo, mantendo a cardinalidade do conjunto elite constante.

Utilizando os critérios apresentados anteriormente, é possível que para determinada instância a execução do *GRASP* finalize com um conjunto elite contendo uma única solução. Este comportamento indica que a solução encontrada na primeira iteração do *GRASP* possui o melhor valor de função objetivo dentre todas as soluções encontradas durante a execução. Adicionalmente, o comportamento indica que a partir da segunda iteração não foram encontradas soluções com diferença de ao menos 10% nas atribuições de item a container quando comparadas à solução encontrada na primeira iteração.

Como última etapa da formação do conjunto elite, para cada solução a abordagem aproveita as atribuições de item a container (família de variáveis x_{ik}) fixando seus valores e chama o CPLEX para encontrar os valores ótimos das variáveis de movimentação (família de variáveis f_{ikt}), sendo esta computação realizada tipicamente em menos de um segundo para cada solução. A solução com valores ótimos das variáveis de movimentação então entra no conjunto elite substituindo a solução do conjunto na qual foi baseada.

Passo2 - Determinação do conjunto inicial de variáveis fixadas

Neste passo são utilizadas idéias do *Path-Relinking* (ver seção 4.2.3) para determinar uma lista de variáveis a serem fixadas inicialmente. O procedimento cria uma lista de variáveis de atribuição a serem fixadas de forma a intensificar a exploração do espaço de busca em trajetórias definidas a partir de pares de soluções do conjunto elite.

Para cada par de soluções do conjunto elite, é adicionada na lista de variáveis a serem fixadas as variáveis que possuem o mesmo valor atribuído nas duas soluções. Para cada item $\hat{i}$:

- Caso as duas soluções tenham atribuído o item $\hat{i}$ ao mesmo container, todas as variáveis $x_{\hat{i}k}$ possuirão o mesmo valor e serão adicionadas à lista de variáveis a serem

fixadas, indicando que não será permitido alterar a atribuição do item durante a computação do *local branching*.

- Caso as duas soluções tenham atribuído o item $\hat{i}$ a containeres diferentes, duas variáveis $x_{\hat{i}k}$ possuirão valor diferente se zero em uma das soluções. Serão adicionadas à lista de variáveis a serem fixadas apenas as variáveis $x_{\hat{i}k}$ que possuírem valor zero nas duas soluções, indicando que será permitido alterar a atribuição do item durante a computação do *local branching*, porém limitando a alteração à trajetória entre as duas soluções.

Para exemplificar a fixação podemos considerar um exemplo de instância na qual existem os containeres a, b e c. Caso uma das soluções tenha atribuído o item $\hat{i}$ ao container a e outra solução tenha atribuído o item $\hat{i}$ ao container b, a fixação de variáveis permitiria que o container $\hat{i}$ fosse atribuído ao container a ou b durante a computação do *local branching*, porém proibiria a atribuição do item $\hat{i}$ ao container c.

A lista de variáveis a serem fixadas inicialmente deve ser então percorrida e as restrições para fixação das variáveis devem ser adicionadas ao modelo. Como a lista contém apenas variáveis que possuem o mesmo valor nas duas soluções do conjunto elite, o valor de cada variável da lista pode ser atribuído apenas a um dos valores atribuídos à variável nas duas soluções consideradas.

A fixação de variáveis é realizada adicionando restrições ao modelo da forma $x_{ik} = \hat{x}$, onde $\hat{x}$ é o valor atribuído à variável x_{ik} nas soluções.

Apesar de conceitualmente simples, um detalhe importante de implementação é que todas as restrições criadas neste passo devem ter sua referência guardada para posterior utilização no método.

Passo3 - Execução do local branching utilizando nova diversificação

Inicialmente são utilizadas idéias do RINS (ver seção 4.1.2) para criar uma lista de itens ordenada por uma função que tenta quantificar a expectativa de melhoria da função objetivo se as variáveis de atribuição associadas ao item forem deixadas livres para a determinação de valor pelo *solver*. O procedimento considera que variáveis binárias que tiveram um valor atribuído na solução da relaxação linear mais próximo de $0,5$ possuem uma expectativa maior de melhoria da solução objetivo quando forem deixadas livres pois as variáveis que possuem valor próximo de 0 ou 1 na relaxação linear e na solução viável de Mod_3 provavelmente já levam a uma boa solução.

O procedimento resolve a relaxação linear de Mod_3 e ordena os itens em ordem não-decrescente de acordo com a seguinte função *deltaRL*:

$$deltaRL_i = Min(|\bar{x}_{ik} - 0,5|) \tag{4.13}$$

Na função 4.13 i é o número do item, k é o número do container e $\bar{x}_{ik}$ é o valor atribuído pela relaxação linear à variável de atribuição do item i ao container k.

A seguir é executado o *local branching* para cada par de soluções do conjunto elite. Como solução inicial do *local branching* é utilizada a solução do par de soluções consideradas que possui a melhor função objetivo.

Uma diferença importante do algoritmo proposto para a implementação *local branching clássico* detalhada na seção 4.2.1 está relacionada à estratégia de diversificação. Na implementação *local branching clássico*, quando a exploração de uma vizinhança não encontrou solução viável encontrada o tamanho da vizinhança é aumentado. Nos experimentos realizados foi percebido que esta estratégia de diversificação não encontrava uma solução melhor do que a melhor solução viável já existente antes da diversificação dado o limite de tempo dos experimentos.

A diversificação proposta nesta implementação utiliza uma abordagem alternativa. Quando a exploração de uma vizinhança não encontrou solução viável encontrada e o tamanho da vizinhança já foi aumentado em iteração anterior, parte das variáveis fixadas no passo Passo2 é liberada através da remoção das restrições que fixavam estas variáveis. Cada vez que esta diversificação ocorre o procedimento libera as restrições associadas a 10% dos itens, utilizando a ordem de liberação de itens definida através lista ordenada pela função 4.13.

Para exemplificar a liberação de variáveis podemos considerar um exemplo de instância na qual existem os containeres a, b e c. Para cada item i as variáveis liberadas vão depender do procedimento realizado no passo Passo2:

- Caso as duas soluções do conjunto elite tenham atribuído o item i a um mesmo container, todas as variáveis de atribuição do item i (x_{ia}, x_{ib}, x_{ic}) foram fixadas no passo Passo2 e são então liberadas durante a diversificação com a remoção das restrições correspondentes criadas no mesmo passo.

- Caso as duas soluções do conjunto elite tenham atribuído o item i a containeres

diferentes, apenas uma das de atribuição do item i terão sido fixadas. Supondo em uma solução o item i foi atribuído ao container a e em outra solução foi atribuído ao container b, apenas a variável de atribuição x_{ic} foi fixada no passo Passo2, sendo então liberadas durante a diversificação com a remoção das restrições correspondente.

Este passo é repetido durante a computação do *local branching* sempre que deve ser realizada uma nova diversificação após uma diversificação de aumento do tamanho de vizinhança, de forma similar à abordagem de *strong diversification* apresentada na seção 4.1.2.

4.3 Experimentos Computacionais

4.3.1 Plataforma de testes

Os experimentos desta seção foram realizados em uma estação com processador Core2Duo 1.3Ghz, 4gb de memória Ram, sistema operacional Ubuntu Linux 10 versão 64 bits. Todas as implementações foram realizadas em linguagem C++ utilizando o compilador G++ versão 4.3. O pacote de otimização CPLEX utilizado para resolver os subproblemas nas implementações e para comparação foi o CPLEX 12 versão 12.2.

A implementação *local branching* clássico foi avaliada apenas para as instâncias testadas na seção 3.7. As demais implementações foram avaliadas utilizando todas as instâncias apresentadas em [5]. Todas as implementações foram implementadas e avaliadas utilizando custos de transporte variáveis.

4.3.2 Resultados do local branching clássico

Os testes da implementação *local branching* clássico (ver seção 4.2.1) foram baseados em uma parametrização obtida empiricamente, aplicando o método às instâncias avaliadas na seção 3.7. Os parâmetros utilizados na parametrização do *local branching* foram uma vizinhança de tamanho (K) igual a **5**, realização de no máximo 20 diversificações, limite de tempo 300 segundos para exploração de cada vizinhança e limite de tempo de execução para cada instância de 7200 segundos.

Os resultados do CPLEX foram obtidos utilizando a parametrização descrita na seção 3.7.

Instância	Local branching clássico	CPLEX
varbtest191b_111	458246	450328
varbtest191b_113	429765	427576
varbtest191b_211	520211	503689
varbtest191b_213	453683	452881
vartest191_113	488095	484971
vartest191_121	465551	465227
vartest191_213	529025	518152
vartest191_222	518864	503489

Tabela 14: Resultados da implementação *local branching* clássico

A Tabela 14 apresenta a instância avaliada, o melhor *upper bound* obtido durante a execução do *local branching* clássico e o melhor *upper bound* obtido durante a execução do CPLEX.

Para todas as instâncias avaliadas o *upper bound* obtido com a utilização do *local branching* clássico foi pior do que o *upper bound* com a utilização do CPLEX utilizando o mesmo limite de tempo.

Com o objetivo de avaliar as diferenças na convergência dos métodos, são apresentados gráficos para ilustrar a convergência do *upper bound* durante a execução de cada método.

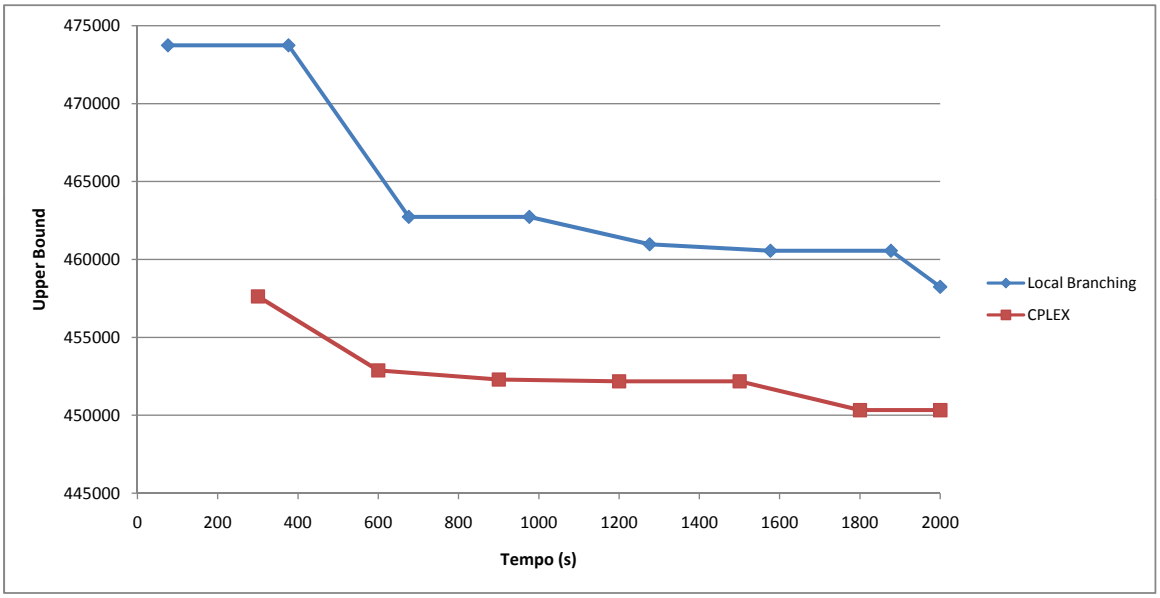

Figura 2: Convergência do *local branching* clássico na instância tb191b_111

Nos gráficos apresentados a medição do *upper bound* durante a execução do método *local branching* clássico foi realizada ao final da exploração de cada vizinhança, sendo este intervalo de no máximo 300 segundos. A medição do *upper bound* para o método

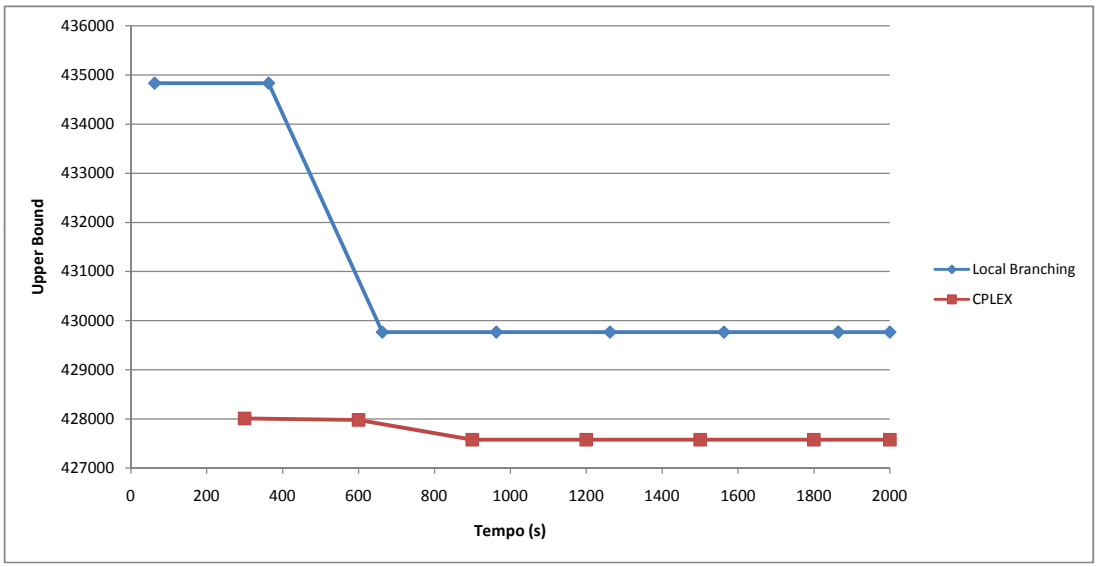

Figura 3: Convergência do *local branching* clássico na instância tb191b_113

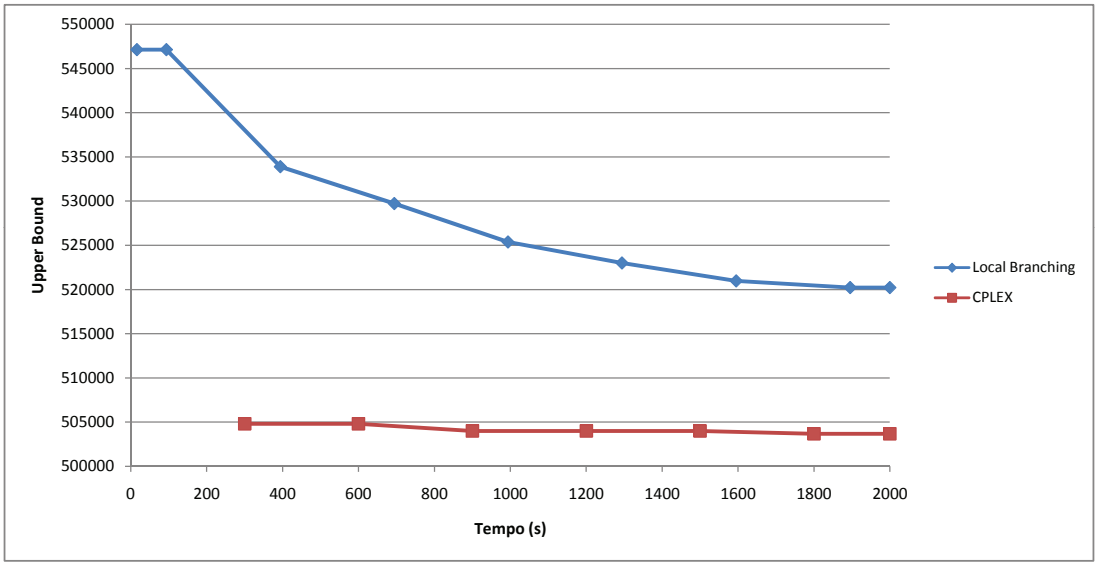

Figura 4: Convergência do *local branching* clássico na instância tb191b_211

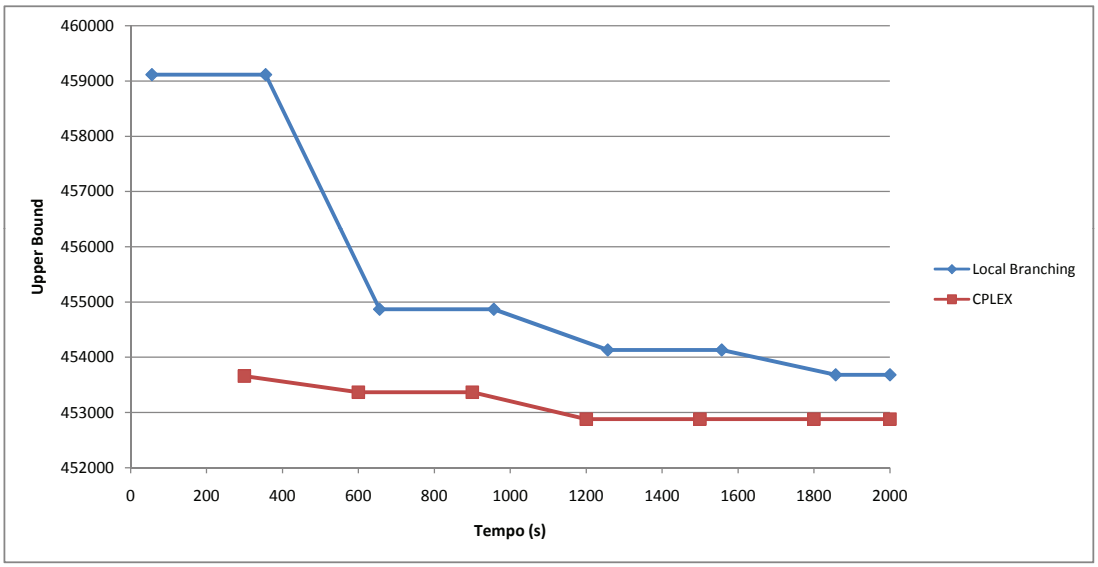

Figura 5: Convergência do *local branching* clássico na instância tb191b_213

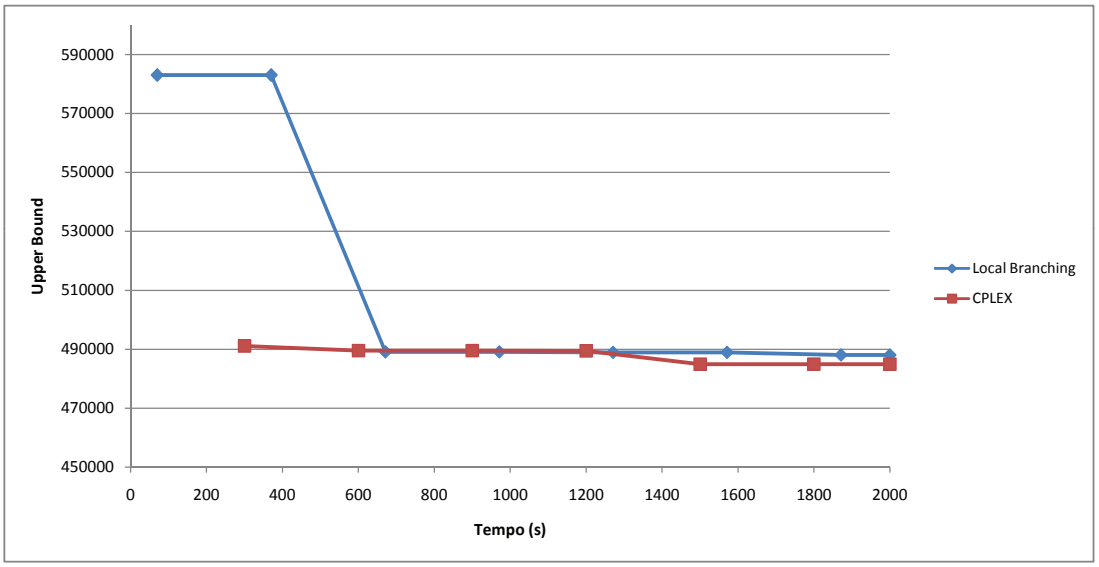

Figura 6: Convergência do *local branching* clássico na instância t191_113

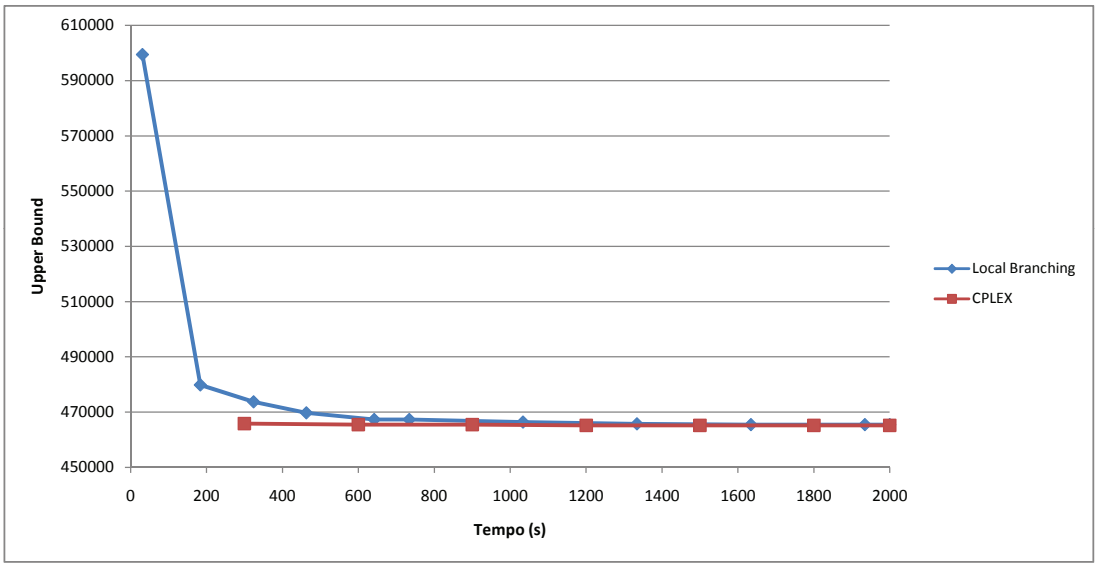

Figura 7: Convergência do *local branching* clássico na instância t191_121

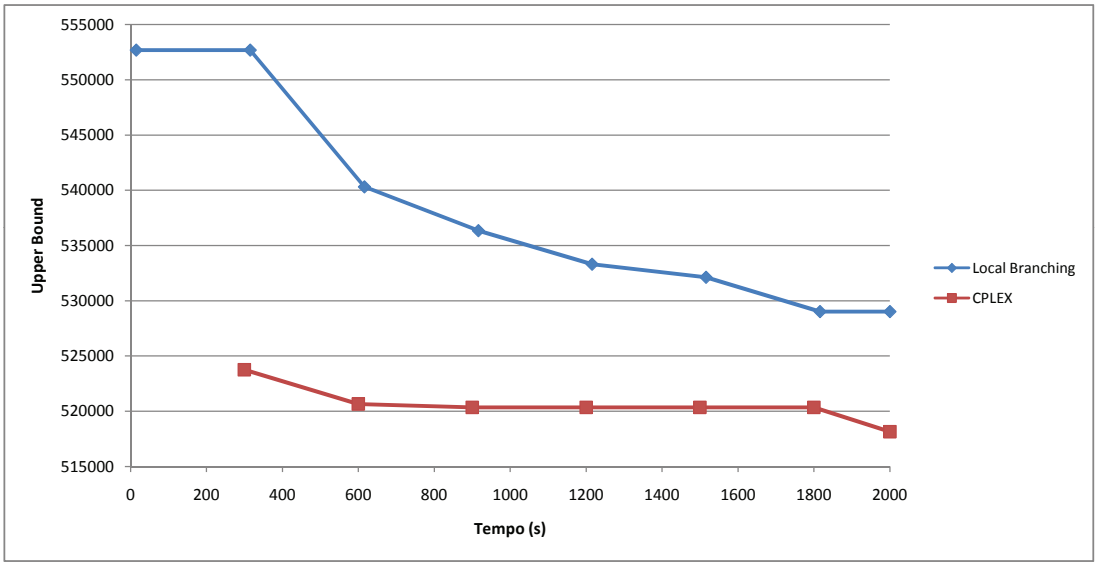

Figura 8: Convergência do *local branching* clássico na instância t191_213

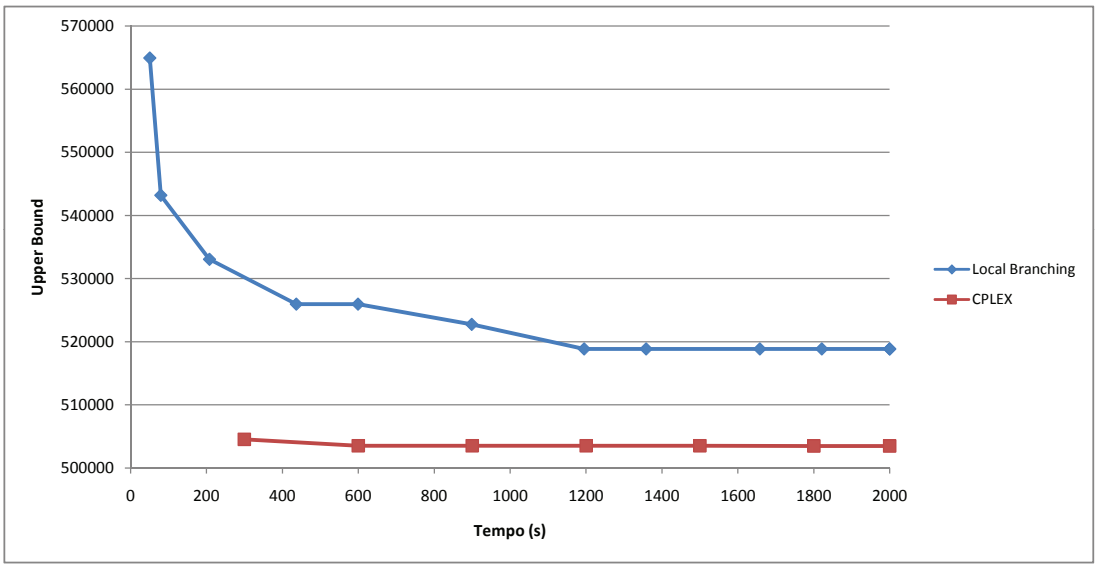

Figura 9: Convergência do *local branching* clássico na instância t191_222

CPLEX foi realizada em intervalos fixos de 300 segundos, não existindo medição no tempo zero por não haver solução viável.

Em duas instâncias (t191_113 e t191_121) o *upper bound* do método *local branching* clássico ficou próximo do *upper bound* do método CPLEX durante a maior parte do tempo de execução. Observando os gráficos de convergência de todas as outras instâncias percebemos que o valor de *upper bound* obtido ao final do tempo limite de execução do método *local branching* clássico é maior do que o *upper bound* obtido pelo método CPLEX após a medição de 300 segundos.

Uma hipótese para a convergência *upper bound* observada na implementação *local branching* clássico seria de que o grande número de variáveis binárias no problema dificulta a efetiva exploração de vizinhanças pelo método. Quando parametrizado para explorar vizinhanças pequenas (parâmetro K), pouca melhoria do *upper bound* era observada após a exploração de cada vizinhança, enquanto que a parametrização de exploração de vizinhanças grandes leva o algoritmo a não determinar a otimalidade de cada vizinhança no tempo limite de exploração.

4.3.3 Local branching com fixação de variáveis

Foram realizados dois conjuntos de testes para a implementação apresentada na seção *local branching* clássico 4.2.2. O primeiro conjunto de testes teve como objetivo analisar o efeito de diferentes parametrizações de tamanho de vizinhança (K) e de percentual de variáveis fixas na convergência do método. O segundo conjunto de testes teve como

objetivo comparar a influência da escolha de diferentes soluções iniciais no resultado do método.

Observando os resultados apresentados na seção 4.3.2 e objetivando priorizar a análise do comportamento heurístico descrito na seção 4.3.2 o tempo total dos testes foi alterado para 400 segundos por instância.

Efeito de diferentes parametrizações na convergência do método

Percebemos durante o trabalho que a parametrização do *solver* tem grande influência nos resultados obtidos pelo método. Em particular, percebemos que ao utilizar o CPLEX como *solver* para explorar as vizinhanças definidas pelo *local branching* diferentes combinações das opções *MIPEmphasis* e *NodeSelect* afetam significativamente os resultados obtidos.

Inicialmente estudamos como diferentes parametrizações do CPLEX afetam a convergência da solução quando o CPLEX é utilizado isoladamente sem o controle do *local branching*. Após testes empíricos, quatro parametrizações foram escolhidas como as mais promissoras no contexto específico das instâncias do LFP que consideram custos de transporte variáveis. O Gráfico 10 apresenta o comportamento das quatro parametrizações do CPLEX quando este é utilizado isoladamente sem controle do *local branching* para resolver a instância tb191b_111 do LFP.

A parametrização *Default* corresponde à estratégia padrão de solução do CPLEX, parametrizada como *MIPEmphasis=BALANCED* e *NodeSelect=DEPHT FIRST SEARCH*.

A parametrização *Alternativo* corresponde à melhor estratégia de solução do CPLEX encontrada empiricamente quando o critério de parada do CPLEX é um tempo limite de 7200 segundos. Esta parametrização é a mesma utilizada para comparar os diferentes modelos para o LFP na seção 3.7 e é configurada como *MIPEmphasis=FEASIBILITY* e *NodeSelect=DEPHT FIRST SEARCH*.

A parametrização *Solução Inicial Local Branching* corresponde à melhor estratégia determinada empiricamente para obter a solução inicial do LFP. Para as instâncias testadas esta estratégia sempre obtém uma solução viável para o CPLEX em até 50 segundos; outras configurações do CPLEX obtêm uma solução viável entre 50 e 150 segundos. A solução inicial encontrada por esta estratégia apresentou melhor valor para a função objetivo (menor *upper bound*) para as instâncias testadas, porém o CPLEX não é capaz de melhorar a solução inicial obtida em um limite de até 1600 segundos. A estratégia é

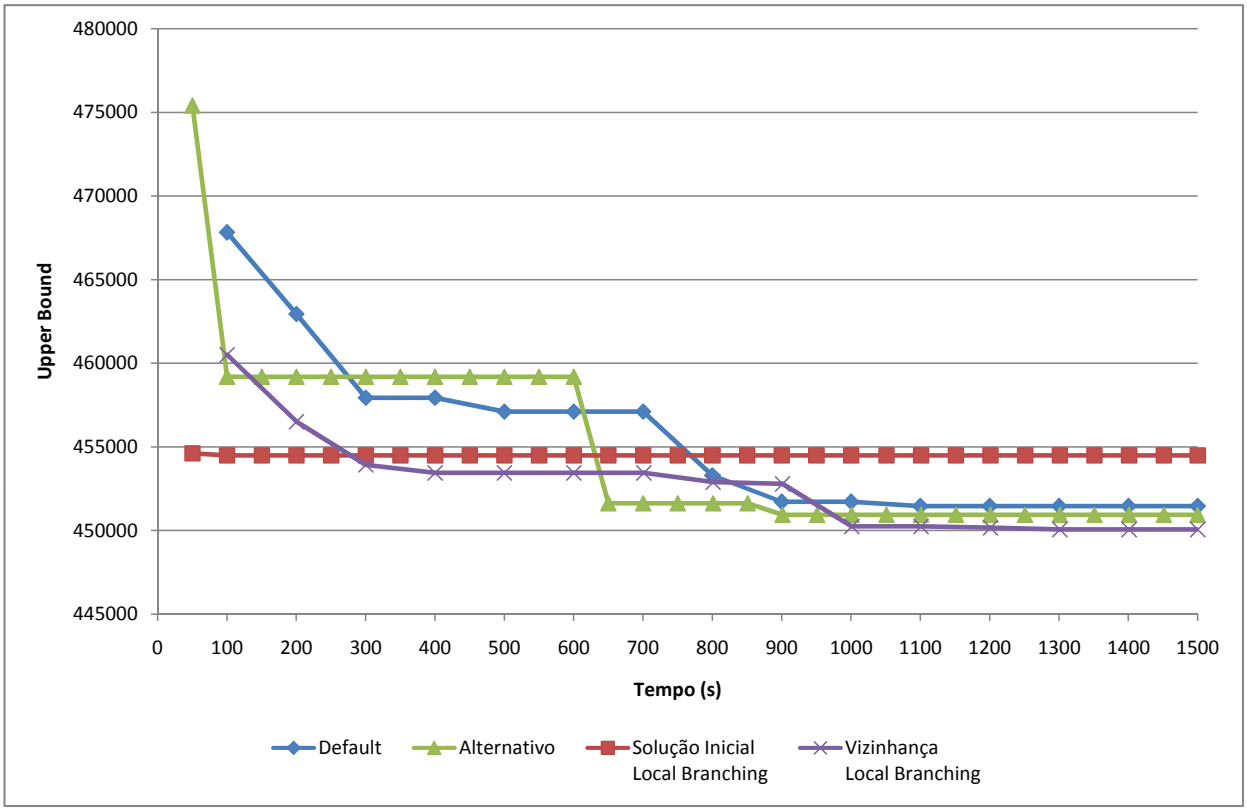

Figura 10: Convergência de diferentes parametrizações do CPLEX

parametrizada como *MIPEmphasis=FEASIBILITY* e *NodeSelect=ALTERNATE BEST ESTIMATE SEARCH*.

A parametrização *Vizinhança Local Branching* corresponde à melhor estratégia encontrada empiricamente quando o critério de parada do CPLEX é um tempo limite de 1600 segundos, sendo parametrizada como *MIPEmphasis=HIDDEN FEASIBILITY* e *NodeSelect=ALTERNATE BEST ESTIMATE SEARCH*.

Com o objetivo de investigar a influência do parâmetro K (tamanho inicial das vizinhanças) e o percentual de variáveis fixadas inicialmente na convergência do *local branching*, foram realizados testes variando estes parâmetros considerando execuções do *local branching* onde o critério de parada era um tempo limite da ordem de 400 segundos. Neste contexto, foi utilizada como solução inicial para o *local branching* a primeira solução encontrada pelo CPLEX utilizando sua parametrização *DEFAULT* para a instância tb191b_111 do LFP.

A quantidade total de variáveis inteiras fixadas seguiu percentuais pré-definidos em 40%, 60% e 80% do número total de variáveis, sendo as variáveis a serem fixadas escolhidas aleatoriamente de acordo com uma distribuição uniforme. Para cada percentual de fixação de variáveis foram realizados testes utilizando três sementes pré-definidas

diferentes para o gerador de números aleatórios utilizado para a escolha do conjunto de variáveis a serem fixadas.

O identificador de cada conjunto de variáveis fixadas tem como prefixo o percentual de variáveis fixadas e como sufixo a semente utilizada para a escolha aleatória das variáveis. Em decorrência da utilização de sementes pré-definidas para o gerador de números aleatórios conjuntos de variáveis fixadas gerados a partir de uma mesma semente apresentarão a mesma sequência de variáveis fixadas, ficando então o menor conjunto de variáveis fixadas completamente contido no maior conjunto de variáveis fixadas.

Fixado (%)	Vizinhança (K)	Conjunto Fixado	Upper Bound	Tempo (s)
40	3	40_1	453947	451
40	3	40_2	451735	522
40	3	40_3	453293	428
40	5	40_1	454298	450
40	5	40_2	451723	450
40	5	40_3	452883	451
40	10	40_1	454813	450
40	10	40_2	451723	450
40	10	40_3	452883	451
60	3	40_1	454107	538
60	3	40_2	451986	464
60	3	40_3	454625	549
60	5	40_1	454113	451
60	5	40_2	451278	483
60	5	40_3	454625	462
60	10	40_1	454099	450
60	10	40_2	451272	450
60	10	40_3	454813	482

Tabela 15: Comparação de parametrizações do *local branching*

A Tabela 15 apresenta o primeiro valor de *upper bound* registrado após 400 segundos pois o mecanismo implementado para persistir as informações de *upper bound* é executado apenas ao final da busca de cada vizinhança. Os testes foram realizados considerando um tempo limite de 150 segundos para exploração de cada vizinhança.

Com o objetivo de comparar a convergência do método com diferentes parametrizações são apresentados gráficos da variação do melhor *upper bound* encontrado durante a execução do método para cada combinação de tamanho de vizinhança K e percentual F do conjunto de variáveis inteiras fixadas.

Foram realizados testes fixando 80% das variáveis binárias no início da execução, porém para todos os tamanhos de vizinhança as regiões exploradas eram inviáveis, sendo a

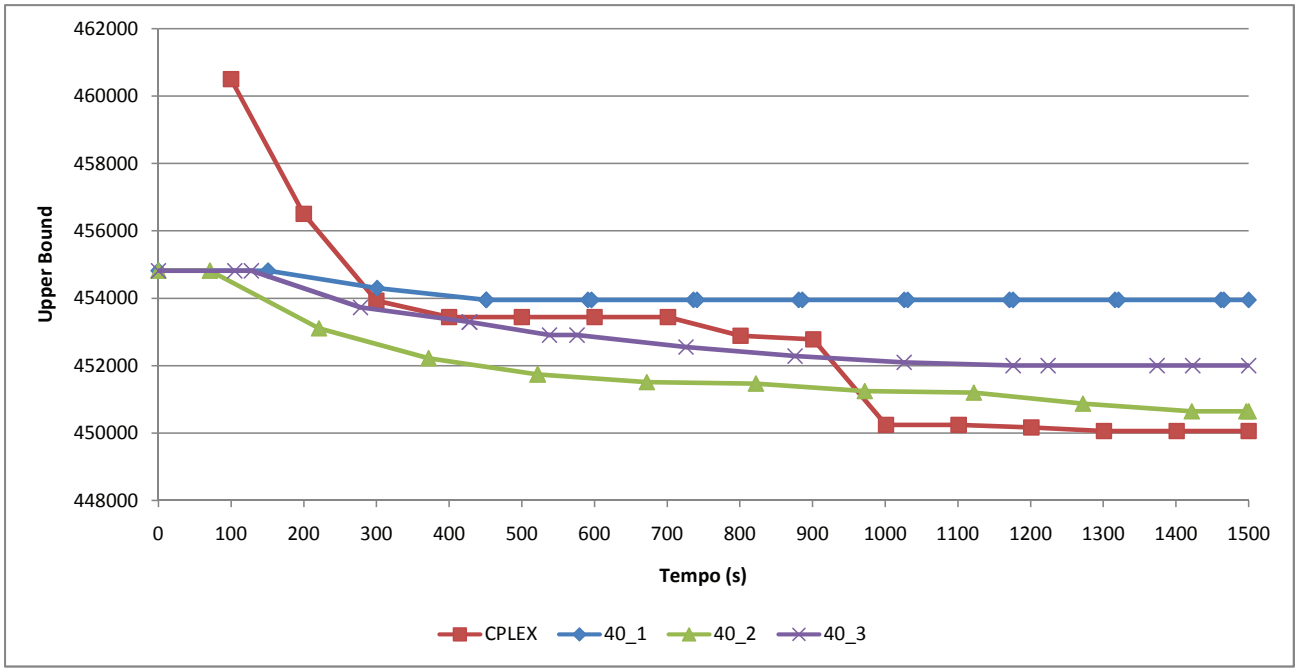

Figura 11: Convergência da parametrização com K=3 e 40% das variáveis fixas

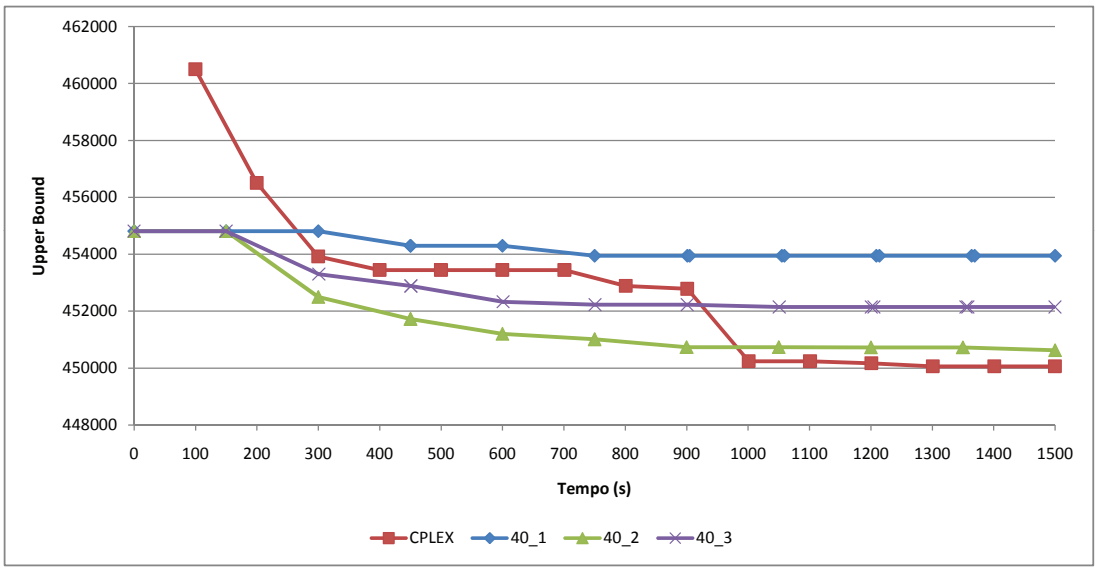

Figura 12: Convergência da parametrização com K=5 e 40% das variáveis fixas

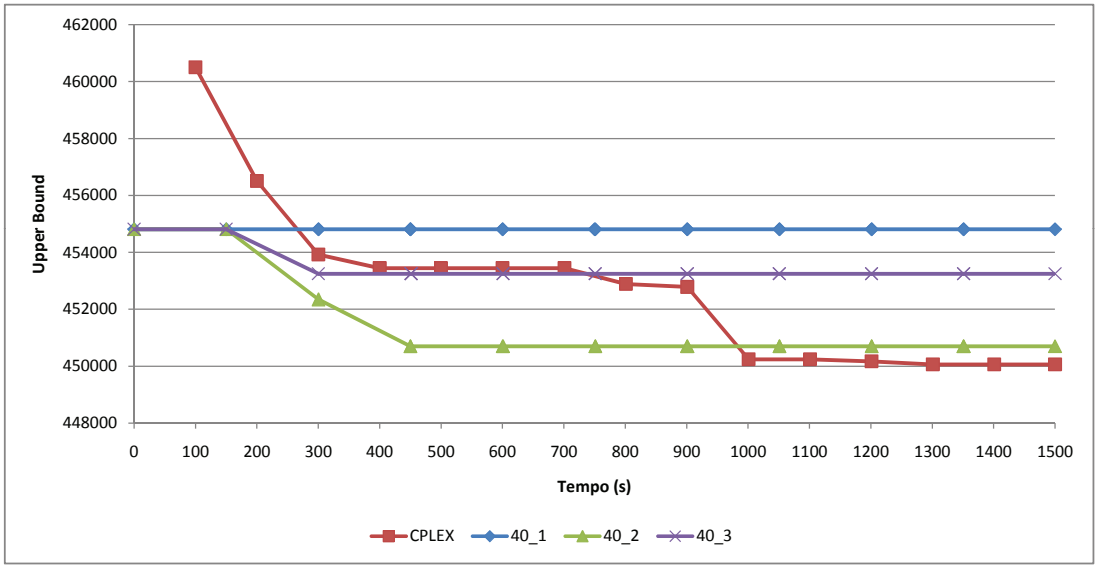

Figura 13: Convergência da parametrização com K=10 e 40% das variáveis fixas

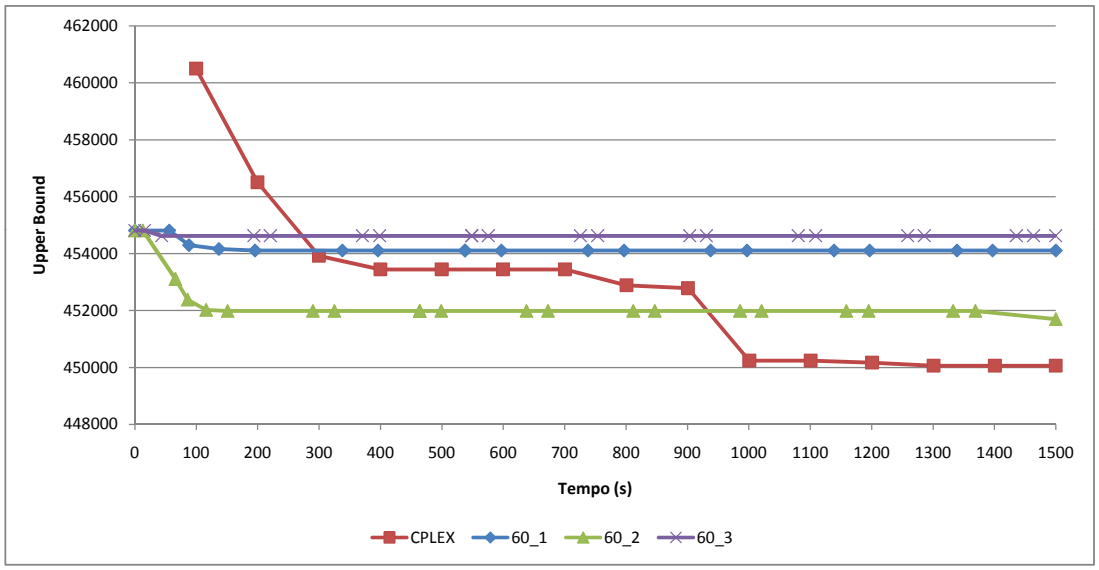

Figura 14: Convergência da parametrização com K=3 e 60% das variáveis fixas

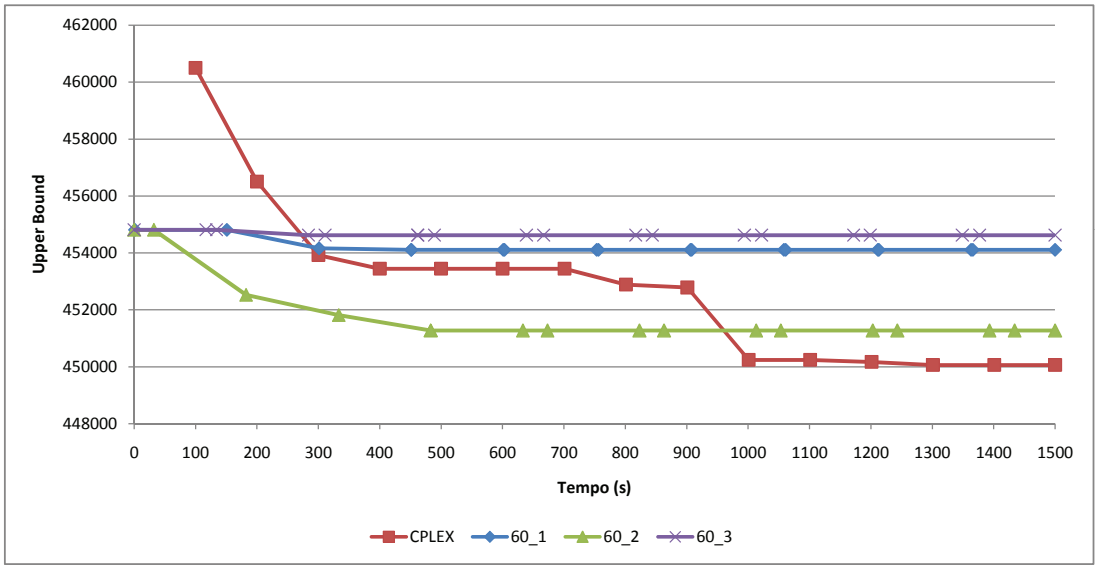

Figura 15: Convergência da parametrização com K=5 e 60% das variáveis fixas

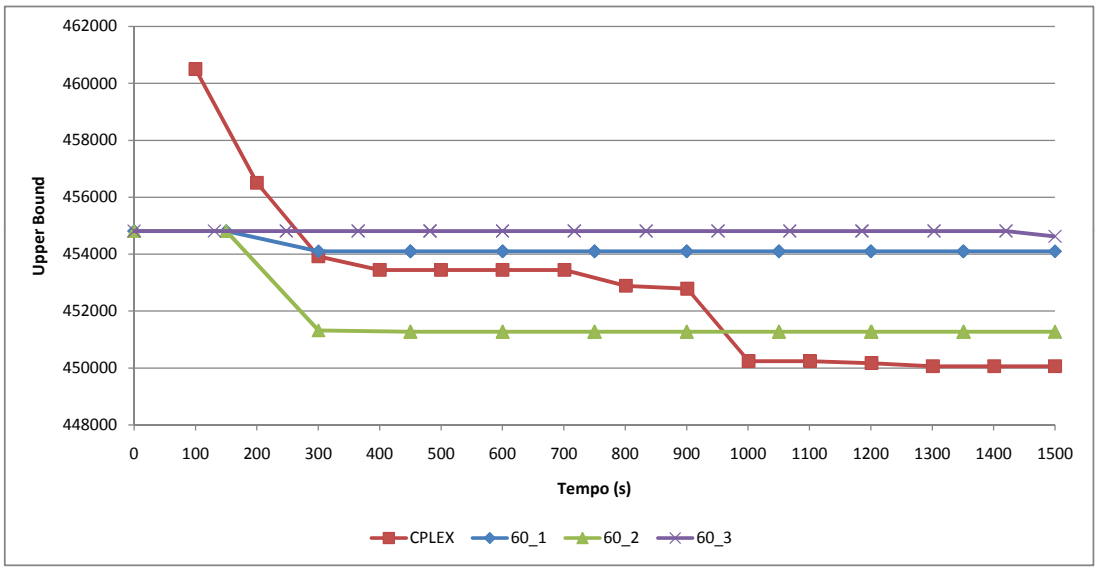

Figura 16: Convergência da parametrização com K=10 e 60% das variáveis fixas

execução do algoritmo interrompida após a realização de 20 diversificações do tipo *strong diversification* como descrito na seção 4.2.1.

Os resultados obtidos indicam que a escolha das variáveis fixadas inicialmente influenciam fortemente a convergência do método, sendo possível observar que a sequência de fixação de variáveis identificada pelo sufixo _2 leva a uma melhor convergência do método independentemente das parametrizações de tamanho de vizinhança e de percentual de variáveis fixadas.

A influência da escolha de variáveis a serem fixadas inicialmente na convergência observada nesta abordagem de *local branching* para o LFP pode ser considerada um importante aspecto negativo, dado que para a efetiva utilização da abordagem a escolha da melhor semente para o gerador de números aleatórios deveria ser realizada empiricamente para cada instância do problema.

Comparação da utilização de diferentes soluções iniciais no resultado do método

Observando os resultados apresentados na seção 4.3.3 a parametrização mantida para as implementações de *local branching* foi uma vizinhança de tamanho (K) igual a 5, utilização da semente de gerador de números aleatórios identificada pelo sufixo _2, tempo limite para exploração de vizinhanças de 60 segundos, limite de tempo de execução para cada instância de 400 segundos, máximo de 20 diversificações e um percentual de fixação de 40% das variáveis positivas presentes na solução da relaxação linear de cada instância.

A Tabela 16 apresenta a instância e os resultados obtidos utilizando como solução inicial para o método a melhor solução do conjunto elite obtido através de GRASP (coluna Local branching Hardfix - GRASP) e a primeira solução viável do CPLEX (coluna Local branching Hardfix - CPLEX) obtida através da parametrização *Solução Inicial Local Branching* descrita na seção 4.3.3. Para cada solução inicial são apresentados o valor da função objetivo na solução inicial (coluna UB Inicial), o valor da função objetivo da melhor solução encontrada até a parada do método (coluna Melhor UB) e o tempo decorrido em segundos para a identificação da melhor solução encontrada (coluna Tempo UB).

Não foi apresentado o tempo de melhoria de *upper bound* para a instância tb191b_123 relacionado à execução com solução inicial do GRASP pois o *local branching* não melhorou a solução inicial dentro do limite de tempo de 400 segundos.

Uma primeira observação pode ser realizada quanto ao *upper bound* inicial das soluções. Em apenas 4 instâncias (tb191b_111, tb191b_123, t191_113, t191_213) a solução

Instância	Local branching Hardfix - GRASP			Local branching Hardfix - CPLEX		
	UB inicial	Melhor UB	Tempo (s)	UB inicial	Melhor UB	Tempo (s)
varbtest191b_111	471904	470935	395	454814	451718	360
varbtest191b_112	429062	428670	390	467629	433553	328
varbtest191b_113	429062	428184	180	448039	432291	300
varbtest191b_121	475179	475092	360	528944	486936	393
varbtest191b_122	513886	513281	400	586083	534644	365
varbtest191b_123	525054	525054	-	557251	529928	363
varbtest191b_211	518603	514294	400	517966	511412	180
varbtest191b_212	455355	454902	300	477947	460887	382
varbtest191b_213	454503	454498	400	473029	456962	400
varbtest191b_221	514077	513870	328	525768	515383	300
varbtest191b_222	552749	552139	400	579495	561334	371
varbtest191b_223	563463	563449	180	578144	569315	300
vartest191_111	433942	428873	373	441879	433085	330
vartest191_112	454999	453846	400	479119	459081	300
vartest191_113	511517	502684	378	491978	489537	60
vartest191_121	465279	465266	209	478467	469927	400
vartest191_122	474479	474460	180	476699	471255	289
vartest191_123	479061	477588	180	497011	487485	301
vartest191_211	469147	456825	370	482887	471015	360
vartest191_212	479747	479447	377	498588	487477	400
vartest191_213	542080	535516	400	530560	523480	367
vartest191_221	503053	502949	120	527796	512248	400
vartest191_222	512535	503828	258	563371	528237	400
vartest191_223	517714	517103	321	524093	516103	360

Tabela 16: Resultados da implementação *local branching* com fixação de variáveis

inicial do CPLEX apresentou *upper bound* melhor do que o *upper bound* da solução inicial obtida com a utilização do GRASP (ver seção 4.2.3).

É possível observar também que nas execuções utilizando solução inicial do GRASP a utilização do *local branching* levou a uma diminuição média do *upper bound* de $0,5\%$, diminuição menor do que a diminuição de $3,2\%$ observada com a utilização do método em execuções que utilizaram a solução inicial do CPLEX. Uma justificativa seria que a maioria das soluções inicias do GRASP apresentam *upper bound* de menor valor, estando então mais próximas do valor ótimo para o problema, sendo então esperado que a melhoria destes *upper bounds* seja computacionalmente mais difícil.

4.3.4 Resultados do local branching com intensificação baseada em soluções GRASP

A implementação de *local branching* que utiliza intensificação baseada em soluções GRASP, apresentada na seção 4.2.3, foi testada utilizando como parâmetros uma vizinhança de tamanho (K) igual a 5, tempo limite para exploração de vizinhanças de 60 segundos e limite de tempo de execução para cada iteração de 400 segundos, sem limite máximo de realização de diversificações.

Foram testadas duas estratégias para fixar variáveis. Na primeira, são comparadas para fixação as variáveis de atribuição das duas soluções do conjunto de elite como o melhor valor de função objetivo. Já na segunda, são comparadas para fixação as variáveis de atribuição das duas soluções mais distantes, sendo a distância entre duas soluções de elite dada pelo número de variáveis de atribuição de item a container que são diferentes nas duas soluções. Nas duas estratégias a solução do par considerado que possui melhor *upper bound* é utilizada como solução inicial para a computação.

Os valores obtidos para o CPLEX utilizaram a parametrização *Solução Inicial Local Branching* descrita na seção 4.3.3. Para instância o CPLEX foi executado com um tempo limite de 2000 segundos, sendo computados o *upper bound*, *lower bound* e GAP relativo durante a execução.

A Tabela 17 está dividida em seções. A seção *Entrada* apresenta a identificação de cada instância (Instância), número de soluções no conjunto elite (#Elite) e o *upper bound* da melhor solução do conjunto elite (UB Elite). A seção *LB (melhor par)* contém os resultados obtidos utilizando a estratégia que considerou as duas soluções de melhor função objetivo no conjunto elite, sendo apresentados o *upper bound* (UB), número de variáveis fixadas inicialmente (#Fix) e tempo decorrido quando o *upper bound* foi identificado (t).

Entrada			LB (melhor par)			LB (par distante)			CPLEX 400s		
Instância	#Elite	UB Elite	UB	#Fix	t (s)	UB	#Fix	t (s)	UB	LB	Gap (%)
varbtest191b_111	2	471904	471283	573	269	471283	573	269	457069	407703	10,8
varbtest191b_112	2	429062	427580	573	349	427580	573	349	433284	395236	8,8
varbtest191b_113	2	429062	427646	573	392	427646	573	392	428652	390683	8,9
varbtest191b_121	4	475159	475159	569	0	475095	565	380	471195	461586	2,0
varbtest191b_122	5	513401	513182	563	300	513323	555	9	512070	506269	1,1
varbtest191b_123	5	525054	523746	551	392	524397	545	11	526799	507581	3,6
varbtest191b_211	4	518030	514226	569	322	515198	565	301	507444	463994	8,6
varbtest191b_212	1	455319	-	-	-	-	-	-	454442	419886	7,6
varbtest191b_213	1	454341	-	-	-	-	-	-	453449	420441	7,3
varbtest191b_221	5	514077	514036	559	1	514036	555	1	513391	498886	2,8
varbtest191b_222	5	552749	552749	573	3	552392	561	100	552201	540521	2,1
varbtest191b_223	5	563463	563187	555	398	562702	541	400	564364	544237	3,6
vartest191_111	2	433942	433488	573	384	433488	573	384	437501	391522	10,5
vartest191_112	4	452228	449351	569	362	449250	563	87	446169	409288	8,3
vartest191_113	5	511517	509367	573	400	505124	563	391	495149	444568	10,2
vartest191_121	2	465279	465279	573	-	465279	573	-	465252	460761	1,0
vartest191_122	2	474479	474479	573	-	474479	573	-	466039	460738	1,1
vartest191_123	5	479061	478915	565	3	477283	561	380	475252	463875	2,4
vartest191_211	2	469147	468212	573	396	468212	573	396	457564	419012	8,4
vartest191_212	3	479363	477346	567	400	477630	565	263	476662	440297	7,6
vartest191_213	3	540590	534720	559	287	537562	557	400	529707	478614	9,6
vartest191_221	2	503053	502942	573	385	502942	573	385	503271	496411	1,4
vartest191_222	1	552734	-	-	-	-	-	-	505361	496370	1,8
vartest191_223	5	516833	514787	569	400	515142	559	207	510878	499742	2,2

Tabela 17: Resultados da implementação local branching com intensificação baseada em soluções GRASP (400 segundos)

A seção *LB (par mais distante)* contém os resultados obtidos utilizando a estratégia que considerou as duas soluções mais distantes entre si no conjunto elite, sendo apresentados o *upper bound* (UB), número de variáveis fixadas inicialmente (#Fix) e tempo decorrido quando o *upper bound* foi identificado (t). A seção *CPLEX 400s* contém os resultados obtidos pelo CPLEX após 400 segundos de computação, sendo apresentados o *upper bound* (UB), o *lower bound* (LB) e o GAP relativo percentual (Gap).

As colunas *#Fix* da Tabela 17 apresentam o número de variáveis da família x_{ik} fixadas inicialmente antes do início da execução do *local branching*. Nas instâncias avaliadas existem 191 itens e 3 containeres, totalizando 573 variáveis x_{ik}. Como descrito na seção 4.2.3, sempre são fixadas as variáveis que possuem mesmo valor nas duas soluções do conjunto elite consideradas durante a iteração. Cada vez que um item é atribuído a containeres diferentes nas duas soluções, duas variáveis x_{ik} ficarão livres, sendo fixada a variável x_{ik} que corresponde à atribuição do item ao container não utilizado em nenhuma das soluções. Desta forma, o valor informado na coluna *Variáveis fixadas* corresponde a $(573 - 2i)$, onde i representa o número de itens que foram atribuídos a containeres diferentes nas soluções consideradas.

Não foi possível executar o método para instâncias tb191b_212, tb191b_213 e t191_222 pois o conjunto elite destas instâncias continha uma única solução.

Em oito instâncias o conjunto elite continha duas soluções, sendo realizada então uma execução do método utilizando o par de soluções existente. Para estes casos os resultados das seções *LB (melhor par)* e *LB (par mais distante)* na Tabela 17 são os mesmos, correspondendo aos valores obtidos na mesma execução.

Em treze instâncias o conjunto de elite continha mais do que duas soluções. Para estas instâncias foram realizados testes utilizando as duas estratégias de fixação de variáveis. Considerando este subconjunto das instâncias, em uma instância (tb191b_221) as duas estratégias obtiveram o mesmo *upper bound*, em seis instâncias (tb191b_122, tb191b_123, tb191b_211, t191_212, t191_213, t191_223) a estratégia de fixação de variáveis que considera o melhor par de soluções do conjunto elite obteve o melhor *upper bound* do *local branching*, enquanto nas demais seis instâncias (tb191b_121, tb191b_222, tb191b_223, t191_112, t191_113, t191_123) a estratégia de fixação de variáveis de melhor *upper bound* obteve os melhores limites.

Em seis instâncias o *upper bound* obtido utilizando uma das estratégias de fixação de variáveis foi melhor do que o obtido pelo CPLEX após 400 segundos de computação. Considerando o subconjunto destas seis instâncias, não foi possível identificar um padrão

de parâmetros ou condições que pudessem ser identificados como determinantes para uma convergência mais rápida do *local branching*. Quatro instâncias do subconjunto possuem conjunto elite com a cardinalidade mínima de duas soluções para a execução do método (tb191b_112, tb191b_113, t191_111, t191_221), indicando que para conjuntos elite com cardinalidade máxima de cinco soluções o número de soluções existentes no conjunto não é fator determinante para a convergência do *local branching*.

Com o objetivo de validar a efetividade da realização de apenas duas iterações para cada instância escolhendo apenas os pares de soluções do conjunto elite compostos pelas melhores soluções ou pelas soluções mais distantes, foram realizados testes computando o *upper bound* obtido por *local branching* para todas as combinações de pares de soluções do conjunto elite de cada instância. Como consequência, o tempo de execução total do *local branching* varia de acordo com a cardinalidade do conjunto elite de cada instância, sendo realizado um número de iterações correspondente a $\binom{n}{2}$, onde n é o número de soluções existentes no conjunto elite. Cada iteração do *local branching* é executada com um tempo limite de 400 segundos e os resultados obtidos são comparados com a execução do CPLEX por 2000 segundos. Os resultados são apresentados na Tabela 18, a qual contém a identificação de cada instância (Instância), número de soluções no conjunto elite (#Elite), *upper bound* da melhor solução do conjunto elite (UB Elite), *upper bound* obtido na melhor iteração de *local branching* (UB), número de variáveis fixadas inicialmente na melhor iteração de *local branching* (#Fix), tempo decorrido em na iteração quando o *upper bound* foi identificado na melhor iteração de *local branching* (t), *upper bound* obtido pelo CPLEX após 2000 segundos (CPX UB), *lower bound* obtido pelo CPLEX após 2000 segundos (CPX LB) e o GAP relativo percentual obtido pelo CPLEX após 2000 segundos (Gap).

Comparando a Tabela 18 com a Tabela 17 é possível observar que em apenas três instâncias (tb191b_122, tb191b_221, tb191b_222) a melhor iteração do *local branching* não considerava o melhor par de soluções nem o par de soluções mais distante. Podemos concluir então que o aumento do tempo total de computação necessário para realizar as iterações com as outras combinações dos pares de solução do conjunto elite não ocasiona uma melhoria significativa dos *upper bounds* obtidos nas instâncias de LFP consideradas. Acreditamos que refinamentos na abordagem proposta, como por exemplo, alterações na formação do conjunto elite e na estratégia de fixação de variáveis, possam resultar em um método competitivo para a solução do LFP.

Entrada			LB (melhor iteração)			CPLEX 2000s		
Instância	#Elite	UB Elite	UB	#Fix	t s	CPX UB	CPX LB	Gap (%)
varbtest191b_111	2	471904	471283	573	269	450499	408048	9,4
varbtest191b_112	2	429062	427580	573	349	427770	395235	7,6
varbtest191b_113	2	429062	427646	573	392	427027	390681	8,5
varbtest191b_121	4	475159	475095	565	380	469527	461616	1,7
varbtest191b_122	5	513401	513017	557	351	512032	506201	1,1
varbtest191b_123	5	525054	523746	551	392	524142	507624	3,2
varbtest191b_211	4	518030	514226	569	322	503528	465261	7,6
varbtest191b_212	1	455319	-	-	-	453933	420462	7,4
varbtest191b_213	1	454341	-	-	-	452804	420548	7,1
varbtest191b_221	5	514077	514031	563	26	509314	498884	2,0
varbtest191b_222	5	552749	552175	567	392	552179	540505	2,1
varbtest191b_223	5	563463	562702	541	400	562601	544235	3,3
vartest191_111	2	433942	433488	573	384	430507	391960	9,0
vartest191_112	4	452228	449250	563	87	445654	409281	8,2
vartest191_113	5	511517	505124	563	391	486397	445427	8,4
vartest191_121	2	465279	465279	573	-	465228	460761	1,0
vartest191_122	2	474479	474479	573	-	465831	460732	1,1
vartest191_123	5	479061	477283	561	380	473042	464073	1,9
vartest191_211	2	469147	468212	573	396	454785	420126	7,6
vartest191_212	3	479363	477346	567	400	474685	440590	7,2
vartest191_213	3	540590	534720	559	287	517835	480437	7,2
vartest191_221	2	503053	502942	573	385	503271	496411	1,4
vartest191_222	1	552734	-	-	-	503941	496374	1,5
vartest191_223	5	516833	514787	569	400	510223	499742	2,1

Tabela 18: Resultados da implementação local branching com intensificação baseada em soluções GRASP (2000 segundos)

5 Conclusão e propostas para trabalhos futuros

O problema de abastecimento de linhas de montagem possui aplicação direta na indústria automobilística, possuindo importantes implicações econômicas e operacionais no contexto do processo de produção. O problema possui aspectos que se assemelham a vários problemas clássicos como o problema de dimensionamento de lotes não capacitado e o problema de empacotamento com tamanho de *bin* variável, apresentando assim uma mescla de elementos relativos ao sequenciamento e ao dimensionamento de lotes.

Este trabalho realizou uma comparação dos modelos já existentes na literatura e algumas de suas variações, observando as diferenças nas relaxações lineares e nos resultados obtidos ao resolver instâncias de tamanho real em um pacote de solução de *MIPs* comercial. Em particular, o modelo Mod_3 proposto no trabalho apresentou resultados favoráveis quando comparado ao modelo mais recente apresentado na literatura, obtendo resultados melhores nas instâncias que consideram custos de transporte fixos e resultados equivalentes nas instâncias que consideram custos de transporte variáveis.

Foi realizado um ensaio do sobre a viabilidade de utilização de desigualdades válidas existentes na literatura para o problema. A implementação do plano de cortes simples utilizando as desigualdades válidas apresentou dificuldade de convergência, sendo realizadas várias iterações do método sem que o valor da função objetivo fosse alterado. Por outro lado, a utilização das desigualdades válidas como cortes adicionados durante a exploração da árvore de *branch-and-bound* teve como efeito uma melhoria mais rápida dos *lower bounds* durante a solução das instâncias, sendo observada em contrapartida uma maior dificuldade para a melhoria dos *upper bounds*.

O trabalho apresentou três abordagens para a aplicação da técnica de *local branching* ao problema de abastecimento de linhas de montagem.

A abordagem *local branching* clássico, mais próxima do método de *local branching* originalmente apresentado na literatura, obteve resultados desfavoráveis quando comparada à utilização do pacote de solução de *MIPs* comercial pois foi observada convergência

mais lenta do que o pacote em todas as instâncias testadas.

Os resultados da segunda abordagem de *local branching* descrita no trabalho foram apresentados em conjunto com uma análise da influência da variação de parâmetros na convergência do método. Os resultados obtidos com a utilização da abordagem são fortemente influenciados pela parametrização utilizada, sendo sugerido inclusive que a escolhe de boas parametrizações precise ser realizada empiricamente para cada instância.

A terceira abordagem de *local branching* proposta no trabalho utiliza um conjunto elite formado à partir de soluções obtidas com a utilização de um algoritmo GRASP. A abordagem obteve resultados favoráveis em parte das instâncias quando comparado à utilização do pacote de solução de *MIPs* comercial, sendo a abordagem considerada pelo autor como a mais promissora para refinamento em trabalhos futuros.

Durante a realização deste trabalho foram identificadas algumas oportunidades para a realização de trabalhos futuros. O problema de abastecimento de linhas de montagem possui características que possivelmente são válidas também no contexto do processo de produção de indústrias não relacionadas à área automobilística. O estudo do problema em contextos diferentes poderia levar à necessidade de consideração de novas características e restrições estendendo os modelos existentes na literatura e os modelos propostos neste trabalho.

A utilização das desigualdades válidas propostas na literatura poderia ser estudada em um contexto de métodos de planos de corte mais sofisticados buscando uma utilização mais eficiente dos cortes, por exemplo, limitando a criação apenas a cortes que sejam ortogonais entre si, sendo possível ainda um estudo voltado para o refinamento das desigualdades.

A utilização de abordagens para solução do problema utilizando técnicas de *local branching* mostrou-se promissora, sendo destacadas as seguintes oportunidades:

- Refinamentos na criação do conjunto elite: Existem diversas oportunidades para refinamento do conjunto elite, como por exemplo a adaptação do GRASP para gerar soluções viáveis para o modelo Mod_3, alteração dos critérios para entrada de soluções no conjunto elite objetivando a aceitação de mais soluções, utilização d estratégias para a geração de soluções com maior diversidade e a melhoria das soluções do conjunto elite através da realização de mais iterações do GRASP.

- Refinamentos na fixação de variáveis: Dentre as oportunidades para o refinamento das estratégias de fixação de variáveis podemos destacar a elaboração de estratégias

para fixar apenas parte das atribuições que são iguais em um par de soluções do conjunto elite, a utilização de informações da relaxação linear para determinar um subconjunto de variáveis mais promissor a ser fixado e a utilização das informações de três ou mais soluções do conjunto elite para determinar quais variáveis devem ser fixadas em uma determinada iteração do método.

Uma última oportunidade identificada é a utilização de abordagens diferentes para a solução do problema de abastecimento de linhas de montagem, como por exemplo, a utilização de técnicas baseadas em programação dinâmica.

Referências Bibliográficas

[1] AGGARWAL, A.; PARK, J. K. Improved algorithms for economic lot size problems. *Operations Research*, INFORMS, v. 41, n. 3, p. 549–571, 1993.

[2] BITRAN, G. R.; YANASSE, H. H. Computational complexity of the capacitated lot size problem. *Management Science*, INFORMS, v. 28, n. 10, p. 1174–1186, 1982.

[3] BRAHIMI, N. et al. Single item lot sizing problems. *European Journal of Operational Research*, Elsevier B.V., v. 168, n. 1, p. 1 – 16, 2006.

[4] CORREIA, I.; GOUVEIA, L.; GAMA, F. S. da. Solving the variable size bin packing problem with discretized formulations. *Computers & Operations Research*, Elsevier Science Ltd., v. 35, n. 6, p. 2103–2113, 2008.

[5] CUNHA, A. S. da; SOUZA, M. C. de. Stronger upper and lower bounds for a hard batching problem to feed assembly lines. *Electronic Notes in Discrete Mathematics*, Elsevier B.V., v. 30, p. 159–164, 2008.

[6] DANNA, E.; ROTHBERG, E.; PAPE, C. L. Exploring relaxation induced neighborhoods to improve mip solutions. *Mathematical Programming*, Springer Berlin / Heidelberg, v. 102, n. 1, p. 71–90, 2005.

[7] DREXL, A.; HAASE, K. Proportional lotsizing and scheduling. *International Journal of Production Economics*, Elsevier B.V., v. 40, n. 1, p. 73 – 87, 1995.

[8] DREXL, A.; KIMMS, A. Lot sizing and scheduling – survey and extensions. *European Journal of Operational Research*, Elsevier B.V., v. 99, n. 2, p. 221 – 235, 1997.

[9] FEO, T. A.; RESENDE, M. G. A probabilistic heuristic for a computationally difficult set covering problem. *Operations Research Letters*, Elsevier B.V., v. 8, n. 2, p. 67 – 71, 1989.

[10] FEO, T. A.; RESENDE, M. G. Greedy randomized adaptive search procedures. *Journal of Global Optimization*, Springer-Verlag, v. 6, n. 2, p. 109–133, 3 1995.

[11] FISCHETTI, M.; LODI, A. Local branching. *Mathematical Programming*, Springer Berlin / Heidelberg, v. 98, n. 1, p. 23–47, 2003.

[12] FLORIAN, M.; LENSTRA, J. K.; KAN, A. H. G. R. Deterministic production planning: Algorithms and complexity. *Management Science*, INFORMS, v. 26, n. 7, p. 669–679, 1980.

[13] FOURER, R.; GAY, D. M.; KERNIGHAN, B. W. *AMPL: A Modeling Language for Mathematical Programming*. 1st. ed. Pacific Grove: Duxbury Press, 2002.

[14] GLOVER, F. Tabu search and adaptive memory programing - advances, applications and challenges. In: *Interfaces in Computer Science and Operations Research*. [S.l.]: Kluwer, 1996. p. 1–75.

[15] GLOVER, F. et al. Fundamentals of scatter search and path relinking. *Control and Cybernetics*, v. 39, p. 653–684, 2000.

[16] HANSEN, P.; MLADENOVIC, N.; UROSEVIC, D. Variable neighborhood search and local branching. *Computers & Operations Research*, Elsevier B.V., v. 33, n. 10, p. 3034 – 3045, 2006.

[17] HAOUARI, M.; SERAIRI, M. Heuristics for the variable sized bin-packing problem. *Computers & Operations Research*, Elsevier Science Ltd., v. 36, n. 10, p. 2877–2884, 2009.

[18] HARRIS, F. How many parts to make at once. *Factory, The Magazine of Management*, v. 10, n. 2, p. 135 – 136, 1913.

[19] HAX, C.; CANDEA, D. *Production and inventory management*. 1a. ed. Englewood Cliffs, N.J.: Prentice-Hall, 1984.

[20] HSU, W.-L. On the general feasibility test of scheduling lot sizes for several products on one machine. *Management Science*, INFORMS, v. 29, n. 1, p. 93–105, 1983.

[21] ILOG AMPL CPLEX System Version 11.0 User's Guide. ILOG Inc. Disponível em: <http://www.netlib.org/ampl/solvers/cplex/ampl110.pdf>. Acesso em: 03 jul. 2010.

[22] KARIMI, B.; GHOMI, S. M. T. F.; WILSON, J. M. The capacitated lsizing problem: a review of models and algorithms. *Omega*, Elsevier B.V., v. 31, n. 5, p. 365 – 378, 2003.

[23] KRAJEWSKI, L. J.; RITZMAN, L. P. *Operations Management: Strategy and Analysis*. 3a. ed. Reading: Addison-Wesley, 1993.

[24] KUCUKYAVUZ, S.; POCHET, Y. Uncapacitated lot sizing with backlogging: the convex hull. *Mathematical Programming*, Springer-Verlag, New York, v. 118, n. 1, p. 151–175, 2009.

[25] LAZIC, J. et al. Variable neighbourhood decomposition search for 0-1 mixed integer programs. *Computers & Operations Research*, Elsevier B.V., v. 37, n. 6, p. 1055–1067, 2010.

[26] MAES, J.; MCCLAIN, J. O.; WASSENHOVE, L. N. V. Multilevel capacitated lot-sizing complexity and lp-based heuristics. *European Journal of Operational Research*, Elsevier B.V., v. 53, n. 2, p. 131–148, 1991.

[27] MARTINEZ, L. C.; CUNHA, A. S. Um arcabouço local branching para problemas de otimização combinatória aplicado ao problema da Árvore de custo mínimo com k arestas. In: *XLI Simposio Brasileiro de Pesquisa Operacional*. Porto Seguro - Brasil: SBPO, 2009.

[28] MLADENOVIC, N.; HANSEN, P. Variable neighborhood search. *Computers & Operations Research*, Elsevier B.V., v. 24, n. 11, p. 1097 – 1100, 1997.

[29] POCHET, Y. Mathematical programming models and formulations for deterministic production planning problems. In: *Computational Combinatorial Optimization, Optimal or Provably Near-Optimal Solutions*. New York: Springer-Verlag, 2001. p. 57–111.

[30] POCHET, Y.; WOLSEY, L. A. *Production Planning by Mixed Integer Programming*. 1a. ed. New York: Springer-Verlag, 2006.

[31] RESENDE, M. G. C.; RIBEIRO, C. C. GRASP and path-relinking: Recent advances and applications. In: *Proceedings of the Fifth Metaheuristics International Conference (MIC2003)*. Kyoto - Japão: Toshihide Ibaraki and Yasunari Yoshitomi, 2003. p. T6–1 – T6–6.

[32] ROGERS, J. A computational approach to the economic lot scheduling problem. *Management Science*, INFORMS, v. 4, n. 3, p. 264–291, 1958.

[33] SALOMON, M. et al. Some extensions of the discrete lotsizing and scheduling problem. *Management Science*, INFORMS, v. 37, n. 7, p. 801–812, 1991.

[34] SOUZA, M. C. de; CARVALHO, C. R. V. de; BRIZON, W. B. Packing items to feed assembly lines. *European Journal of Operational Research*, Elsevier B.V., v. 184, n. 2, p. 480–489, 2008.

[35] WAGNER, H. M.; WHITIN, T. M. Dynamic version of the economic lot size model. *Management Science*, INFORMS, v. 5, n. 1, p. 89–96, 1958.

[36] WOLSEY, L. A. Progress with single-item lot-sizing. *European Journal of Operational Research*, Elsevier B.V., v. 86, n. 3, p. 395 – 401, 1995.

[37] YAMAN, H. Polyhedral analysis for the two-item uncapacitated lot-sizing problem with one-way substitution. *Discrete Applied Mathematics*, Elsevier Science Publishers B. V., v. 157, n. 14, p. 3133–3151, 2009.