JOSÉ EURÍPEDES FERREIRA DE JESUS FILHO

ALGORITMOS DE BUSCA LOCAL APLICADOS AO PROBLEMA INTEGRADO DE DIMENSIONAMENTO DE LOTES E PROGRAMAÇÃO DA PRODUÇÃO

Monografia de graduação apresentada ao Departamento de Ciência da Computação da Universidade Federal de Lavras como parte das exigências do curso de Ciência da Computação para obtenção do título de Bacharel em Ciência da Computação.

LAVRAS - MG
2010

JOSÉ EURÍPEDES FERREIRA DE JESUS FILHO

ALGORITMOS DE BUSCA LOCAL APLICADOS AO PROBLEMA INTEGRADO DE DIMENSIONAMENTO DE LOTES E PROGRAMAÇÃO DA PRODUÇÃO

Monografia de graduação apresentada ao Departamento de Ciência da Computação da Universidade Federal de Lavras como parte das exigências do curso de Ciência da Computação para obtenção do título de Bacharel em Ciência da Computação.

Aprovada em ____ de _____ de _____

_____

Dr. Joaquim Quintero Uchôa UFLA

_____

Dr. Ricardo Silveira de Sousa UFLA

_____

Dr. Claudio Fabiano Motta Toledo

Orientador

LAVRAS - MG
2010

Aos meus pais, José Eurípedes e Gilda.

À minha irmã, Juliana.

Ao meu sobrinho, Iago.

DEDICO.

AGRADECIMENTOS

Ao Prof. Claudio, pela insistência e paciência, e cujos ensinamentos e amizade superam e muito uma simples orientação.

Aos meus pais, que sempre acreditaram e apoiaram, mesmo quando eu mesmo tinha dúvidas.

À minha irmã Juliana, pela força oferecida, pela confidência prestada, pelas longas conversas, pela fé depositada, pelo companheirismo e dedicação.

Aos meus amigos de graduação, Jônatas, Lucas de Luca, Lucas de Oliveira e Rodrigo, pelo humor às aulas, pela insistência nos ensinamentos e pela união durante todo o caminho de formação.

Aos amigos de república, João Ricardo, Tales, Guilherme e Filipe, pela paciência com as minhas súbitas mudanças de humor.

Aos amigos de Jataí, Osório, Elber, Gleuber, Max, Karen, Marcela e Micaela, pela total dedicação, pelos bons e maus momentos, pelas noites de brincadeiras ou de conversas sérias, pela confiança e apoio.

Aos Profs. Joaquim e Luiz Henrique, pelo empenho em suas disciplinas, pelos momentos de conversas nos corredores e pela atenção oferecida.

RESUMO

Os Problemas de Dimensionamento de Lotes e Programação da Produção se tornaram importantes temas de estudo devido à sua freqüente ocorrência em diversos contextos industriais. O presente trabalho propõe abordagens baseadas em métodos de busca local, *simulated annealing* e busca tabu, para solucionar o Problema Integrado de Dimensionamento de Lotes e Programação da Produção (PIDLPP). Os métodos propostos são avaliados utilizando conjuntos de instâncias do problema e comparados com outros métodos encontrados na literatura. Um estudo de diferentes abordagens para execução de busca em vizinhança também é realizado em diferentes instâncias do PIDLPP.

Palavras-chave: Busca tabu; *simulated annealing*; dimensionamento de lotes; programação da produção.

ABSTRACT

Lot Size and Scheduling Problems have become a very important literature study theme due to their high aplicability in real industry context. The present work proposes local search approaches as, simulated annealing and tabu search to solve the Synchronized and Integrated Two-Level Lot Size and Scheduling Problem (SITLSP). The proposed methods are evaluated over a set of problem instances and compared with other methods found in literature. A study of different neighborhood approaches is also made for SITLSP.

Keywords: Tabu search; simulated annealing; lot size, scheduling.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS

PDLPP	PROBLEMA DE DIMENSIONAMENTO DE LOTES E PROGRAMAÇÃO DA PRODUÇÃO
PIDLPP	PROBLEMA INTEGRADO DE DIMENSIONAMENTO DE LOTES E PROGRAMAÇÃO DA PRODUÇÃO
PCDLPP	PROBLEMA CAPACITADO DE DIMENSIONAMENTO DE LOTES E PROGRAMAÇÃO DA PRODUÇÃO
PDLPPSC	PROBLEMA DE DIMENSIONAMENTO DE LOTES E PROGRAMAÇÃO DA PRODUÇÃO COM SETUP CONTÍNUO
PCDLM	PROBLEMA CAPACITADO DE DIMENSIONAMENTO DE LOTES MULTI-NÍVEIS
SA	SIMULATED ANNEALING
BT	BUSCA TABU
AG	ALGORITMO GENÉTICO
SI	SISTEMA IMUNE
SIA	SISTEMA IMUNOLÓGICO ARTIFICIAL

SUMÁRIO

1 INTRODUÇÃO

1.1 Contextualização e Motivação

O presente trabalho desenvolve a pesquisa de métodos heurísticos baseados em busca local para solucionar o Problema Integrado de Dimensionamento de Lotes e Programação da Produção (PIDLPP).

O PIDLPP é um problema de produção baseado em uma situação real encontrada em fábricas de refrigerantes. No Brasil, a produção de refrigerante encontra-se em ascendência. Em 2008, a indústria de refrigerantes faturou 9,05% mais do que em 2007. Em 2009, estudos revelam que até o mês de junho, o faturamento foi 3,89% maior em relação ao mesmo período de 2008 (ABIR, 2009).

Os problemas de produção se tornaram tema de pesquisa de vários autores nos últimos anos. Isso se deve à ocorrência de tais problemas nos mais diversos contextos industriais, onde a necessidade de otimizar recursos e reduzir custos é uma constante.

No meio industrial, as decisões de o quê, quanto, onde, como, quando, com o quê e com quem produzir devem ser tomadas. Além disso a demanda a ser atendida, a demanda futura, o tempo de produção, o custo de produção, os custos de estoque, a capacidade das máquinas e vários outros fatores que variam de acordo com o contexto de cada indústria também devem ser considerados

Estas decisões e restrições são fatores cruciais para o custo de produção. Um bom planejamento neste cenário passa pelo conhecimento de todas as variáveis e restrições envolvidas no processo produtivo. O objetivo em geral é minimizar os custos que podem impactar fortemente no sucesso ou fracasso financeiro de uma indústria.

O problema de produção possui complexidade muito alta, devido ao grande número de variáveis e restrições presentes no processo produtivo. Assim, formulações matemáticas complexas levam ao desenvolvimento de métodos exatos que muitas vezes consomem grande quantidade de tempo computacional para retornar uma

solução ótima ou mesmo factível. Por outro lado, as decisões de produção devem ser tomadas em curtos períodos de tempo em situações reais dentro de uma indústria.

Neste cenário, o uso de métodos de inteligência artificial torna-se uma ótima alternativa. O presente trabalho propõe o uso de heurísticas baseadas em busca local para a resolução deste tipo de problema de produção.

Os métodos propostos neste trabalho são avaliados em diversos cenários de complexidade, intitulados instâncias, e comparados com outras heurísticas e métodos exatos encontrados na literatura.

1.2 Objetivos do Trabalho

O presente trabalho tem como objetivo o estudo de heurísticas baseadas em busca local (*simulated annealing* e busca tabu) como solução para o Problema Integrado de Dimensionamento de Lotes e Programação da Produção (PIDLPP).

Outros objetivos:

- Estudar diferentes tipos de abordagens de busca em vizinhança para o PIDLPP.
- Avaliar o comportamento das buscas locais em diferentes níveis de complexidade (instâncias) dos problemas.
- Comparar os métodos desenvolvidos com outras heurísticas e métodos exatos já existentes na literatura.

1.3 Estrutura do Trabalho

O primeiro capítulo deste trabalho faz uma breve introdução e a contextualização do problema de produção. O segundo capítulo abrange o referencial teórico necessário para a compreensão do problema e dos métodos abordados. O terceiro capítulo descreve as características do Problema Integrado de Dimensionamento de Lotes e Programação da Produção e os métodos propostos para solucionar o PIDLPP. O quarto capítulo traz as medidas utilizadas para comparação de

desempenho e apresenta os resultados computacionais obtidos. O quinto capítulo destaca as conclusões do presente trabalho.

2 REFERENCIAL TEÓRICO

2.1 Introdução

O presente capítulo introduz alguns conceitos relacionados a métodos de busca local e a problemas de programação da produção. O objetivo é facilitar o entendimento dos métodos propostos e do problema estudado neste trabalho. Isso é feito através da apresentação de alguns métodos e problemas relacionados existentes na literatura. A seção 2.2 apresenta uma revisão focada em métodos de resolução que utilizam busca local. A seção 2.3 introduz conceitos relacionados a problemas de dimensionamento de lotes e programação da produção. Diversos autores que trabalham neste tipo de problema são citados.

2.2 Métodos de Busca Local

Bertsimas e Tsitsiklis (1997) classificam os algoritmos para resolução de problemas de otimização em três principais grupos:

1. **Algoritmos Exatos -** Algoritmos capazes de garantir a solução ótima. Geralmente desprendem muito tempo computacional e normalmente são inviáveis em aplicações reais. São exemplos desse grupo de algoritmos: *branch and bound*, *branch and cut*, algoritmo *Simplex* e programação dinâmica.
2. **Algoritmos de Aproximação -** São algoritmos capazes de encontrar soluções factíveis e que determinam o seu grau de otimalidade, ou seja, classificam quão boa é a solução. Geralmente, as soluções encontradas não são ótimas.
3. **Heurísticas -** Algoritmos capazes de encontrar soluções sem a garantia de factibilidade ou otimalidade. Geralmente, desprendem pouco tempo computacional. São exemplos deste grupo de algoritmos: métodos de busca local e algoritmos evolutivos.

Heurística é uma simplificação do domínio de busca, para facilitar a descoberta de soluções para um problema. Métodos heurísticos são algoritmos baseados nesta

simplificação que avaliam e fazem aproximações para encontrar soluções para o problema. Esta técnica nem sempre garante a melhor solução do domínio.

A classe de algoritmos heurísticos começou a ser estudada amplamente a partir dos anos 70, com o desenvolvimento da teoria da complexidade e a prova de que a maioria dos problemas combinatoriais são NP-Difícil. Isso significa que dificilmente se encontraria algoritmos exatos capazes de retornar soluções para estes problemas em um tempo computacional razoável (GENDREAU, 2003).

Dessa forma, diversas técnicas heurísticas surgiram com o decorrer do tempo, e entre elas, as buscas locais se tornaram uma classe de métodos bastante popular.

Gendreau (2003) define busca local como um procedimento interativo que progressivamente melhora uma solução inicial factível, utilizando diversos operadores ou movimentos. De acordo com MacFarlane (2009), buscas locais podem ser utilizadas na resolução de problemas computacionalmente complexos.

O sucesso de uma busca local está ligado diretamente à definição de sua vizinhança. A vizinhança de uma solução consiste em todas as possíveis soluções que podem ser atingidas, a partir da aplicação dos operadores de busca disponíveis sobre outra solução. Operadores, ou movimentos, são pequenas perturbações que modificam alguma solução a fim de se atingir outra diferente. Com isso, a vizinhança e os tipos de operadores estão estreitamente ligados ao tipo de codificação utilizada para representar uma solução.

Em 1983, uma impressionante técnica de busca local foi descrita, o *Simulated Annealing* (SA) de Kirkpatrick *et al*. (1983). Tratava-se de um método baseado no sistema de resfriamento dos corpos e que tinha convergência para o ótimo global garantida quando o número de iterações da busca tendia ao infinito.

SA é um método de busca local que aceita soluções de pior qualidade baseada em uma curva logarítmica controlada por um parâmetro chamado temperatura. A temperatura está diretamente ligada à probabilidade de aceitação de soluções de pior qualidade. Geralmente, no início do procedimento, a temperatura possui um valor alto,

e este valor diminui com o decorrer das iterações de acordo com uma função de resfriamento, caracterizando assim a convergência da busca.

Este método desde então se popularizou bastante pela facilidade de implementação e pelos bons resultados alcançados. Jin *et al.* (2009) apresenta uma aplicação de *simulated annealing* em um problema de programação de máquinas simples baseados em uma indústria de fabricação de correntes da China. As definições clássicas de operadores de busca (como o movimento de troca simples, por exemplo) não surtiriam efeito neste problema tendo em vista a existência de famílias de produtos. Isso significa que produtos da mesma família não apresentam tempo de troca de um produto para o outro. Dessa forma, Jin *et al.* (2009) desenvolvem novos operadores de busca baseados na destruição e construção de famílias de produtos, onde utilizam um procedimento guloso na montagem de novos lotes conseguindo resultados melhores que outros métodos da literatura. Barbarosoğlu e Özdamar (2000) fazem uma análise do espaço de busca de soluções iniciais e movimentos factíveis e infactíveis para o Problema Capacitado de Dimensionamento de Lotes Multi-níveis (PCDLM) utilizando *simulated annealing*.

Em 1986, Fred Glover propõe uma nova meta-heurística, que conta com uma estrutura de memória para guiar a busca local, conhecida como Busca Tabu (BT) (GLOVER, 1986). O termo meta-heurística desde então é comumente usado para designar métodos que guiam de forma inteligente uma heurística.

Assim, BT é uma meta-heurística de busca local dotada de uma estrutura de memória que guarda informações das ultimas soluções visitadas guiando a convergência do método pelo espaço de busca (Hertz *et al.*, 1995).

Uma formulação da BT é apresentada em Glover (1989) e Glover (1990). Glover e Laguna (1993) descrevem a aplicação de BT em problemas combinatoriais.

A estrutura de memória da BT é um fator impactante no caminho por onde a busca é conduzida. Estruturas de memória muito grandes diversificam a busca, pois os movimentos ditos promissores são proibidos durante muitas iterações. Estruturas de

memória muito pequenas intensificam a busca, pois os movimentos promissores são liberados mais rapidamente. Glover e Laguna (1997) estudaram o impacto destas variações. Também avaliaram o uso de técnicas de intensificação em locais promissores e diversificação da solução quando a busca fica presa em mínimos locais. Laguna (1995) descreve o impacto da busca exaustiva em problemas de menor e maior complexidade.

Lü e Hao (2010) utilizam uma versão adaptativa da BT para solucionar um problema de *timetabling*, com penalizações que guiam os operadores e sistema de intensificação e diversificação do espaço de busca. Venditti *et al.* (2010) usam BT como método de resolução de um problema de programação farmacêutico onde a solução é representada como um grafo direcionado. Dois movimentos de busca são definidos: troca de duas tarefas adjacentes de uma mesma máquina e a remoção/inserção de um trabalho de uma máquina para outra.

Outras técnicas heurísticas também se tornaram popular, como os algoritmos evolucionários. Holland (1975) propôs uma nova heurística baseada na evolução das espécies de Darwin, chamada Algoritmo Genético (AG). O AG é um algoritmo evolucionário que, através da geração de uma população de indivíduos (soluções), da troca de genes (recombinações) e de mutações, realiza a pesquisa no espaço de busca. Todavia, o AG se popularizar mais tarde com o trabalho desenvolvido por Goldberg (1989).

A população de um AG pode muitas vezes convergir rapidamente para mínimos locais, tendo em vista que soluções que caracterizam este ponto de mínimo espalham seus genes mais rapidamente devido ao seu valor de aptidão. Paszkowicz (2009) estuda técnicas de penalizações em soluções com estas características como tentativa de diversificação de busca. Toledo *et al.* (2008a) propõem um AG onde a convergência da população se caracteriza pela não inserção de nenhum novo indivíduo na população, após a iteração de recombinações e mutações. Neste caso, reinicializam todos os indivíduos da população com exceção do melhor indivíduo obtido.

Mais tarde, surge outro exemplo de algoritmos evolucionários: o Sistema Imunológico Artificial (SIA) ou também conhecido por Sistema Imune (SI). Trata-se de um algoritmo baseado no paradigma do sistema imune dos vertebrados, que se utiliza de aprendizado e memória para solucionar o problema tratado. Termos como antígeno e anticorpo são usados para designar o problema e uma possível solução respectivamente. Alguns dos modelos propostos para esse tipo de algoritmo são a seleção clonal e a rede imunológica (CASTRO, 2001).

Um SAI é proposto por Hu *et al.* (2009) como solução de um problema de sistema de distribuição de tabaco na China. Trata-se de um problema de rotas, onde objetiva-se minimizar o número de tours em cada distrito, a distância percorrida e o tempo total gasto.

Outros importantes métodos de busca local são: *Greedy Randomized Adaptive Search Procedure* (GRASP) (Nascimento *et al.*, 2010), *Threshold Accepting* (Fleischmann e Meyr, 1997) (Meyr, 2000), entre vários outros.

Além de todos os métodos citados, é importante frisar os algoritmos híbridos, que exploram de alguma forma dois ou mais métodos na tentativa de solucionar um problema.

Meyr (2000) combina *simulated annealing* e *threshold accepting* com reotimização dual para solucionar um problema de programação da produção com custo de trocas e máquinas simples. Zhang e Wu (2010) utilizam um algoritmo de sistema imune com *simulated annealing* em um problema de *job shop*. Wang e Tang (2010) combinam estruturas de *Scatter Search* e Busca Local Variada como método para o problema de planejamento de *prize-collecting*. Nascimento *et al.* (2010) apresenta uma combinação de algoritmo GRASP com *path-relinking* para solucionar o Problema Capacitado Multi-Níveis de Dimensionamento de Lotes. Neste problema, temos um cenário onde algumas combinações de níveis dão origem a alguns itens, e algumas combinações de itens dão origem ao produto final.

Fleischmann e Meyr (1997) desenvolvem uma solução híbrida para o Problema Geral de Dimensionamento de Lotes e Programação da Produção (PGDLPP), onde uma variante de busca local baseada em *threshold accepting* faz o sequenciamento dos itens na máquina e a determinação das quantidades, ou tamanho dos lotes, é feita através de uma heurística gulosa.

Uma revisão mais completa sobre métodos heurísticos em problemas de dimensionamento de lotes e programação da produção pode ser encontrada nos trabalhos de Karimi *et al.* (2003) e Jans (2007).

2.3 Problemas de Programação da Produção

Um Problema de Dimensionamento de Lotes e Programação da Produção (PDLPP) consiste em determinar a quantidade e quando determinados produtos serão produzidos. A demanda de cada produto em geral deve ser atendida dentro de determinado período de tempo, utilizando os recursos disponíveis e respeitando várias restrições. Essas restrições podem estar relacionadas ao tempo de processamento, tempo de troca e capacidade das máquinas. Existem custos envolvidos como custos de estocagem, custos de troca custos de produção, entre outros. O objetivo é planejar a produção de um modo a utilizar da melhor forma possível os recursos, minimizando os custos.

Um bom planejamento da produção pode ser um fator crucial dentro de uma organização. Planejamentos mal elaborados podem se tornar muito caros, impactando diretamente nas finanças de uma empresa. Kuik *et al.* (1994) citado por Jans (2007) descreve os impactos do dimensionamento de lotes e programação da produção dentro das organizações em vários níveis estratégicos.

Existem diferentes modelos matemáticos para o PDLPP. Karimi (2003), citado por Nascimento (2007), diz que as características que classificam um problema de dimensionamento de lotes são: horizonte de planejamento, número de níveis, número de produtos, capacidade de recursos, demanda, custos de configuração, número de

máquinas e número de micro-períodos. Uma breve descrição de cada característica é dada abaixo:

- **Horizonte de planejamento**: Pode ser finito ou infinito, dividido em períodos ou não.
- **Número de níveis**: O PDLPP pode ser multi-nível ou de um único nível. Problemas multi-níveis se caracterizam quando na produção de um produto, este depende de outro, ou seja, um produto pode ser matéria prima para outro produto.
- **Número de produtos**: O problema pode envolver vários produtos ou um único produto apenas.
- **Capacidade de recursos**: O problema pode ser capacitado ou não capacitado, ou seja, o problema pode possuir restrições de capacidade ou capacidade infinita.
- **Demanda**: A demanda do PDLPP pode ser fixa ou dinâmica dentro do horizonte de tempo considerado.
- **Custo de configuração**: Os custos de configuração, ou custos de *setup*, podem ser dependentes dos produtos configurados anteriormente na máquina. Alguns problemas ainda apresentam "famílias" de produtos, onde alguns produtos são muito similares a outros e não há custo de troca entre eles. Já a troca de uma família para a outra geralmente apresenta custos.
- **Número de máquinas**: O problema pode envolver somente uma máquina ou várias máquinas usadas na produção.
- **Número de micro-períodos:** O período de tempo pode ser discretizado em períodos menores, denominados micro-períodos. Os micro-períodos não influenciam no tempo de planejamento em si, mas podem limitar o número máximo de configurações (ou trocas de produtos) dentro de um período de tempo. Diversos problemas consideram que um produto pode ser produzido em um micro-período.

Warner e Whitin (1958) desenvolveram uma solução ótima para o problema de produzir um único item em uma única máquina, sem restrições de capacidade. Bitran e Yanasse (1982) provam que o problema de dimensionamento com múltiplos itens e restrição de capacidade é NP-Difícil.

Jans (2007) classificou os vários modelos matemáticos para o PDLPP em **modelos de dimensionamento de lotes** e **modelos de dimensionamento de lotes dinâmico**. No primeiro caso, a demanda dos produtos é contínua, o tempo não é dividido e o horizonte de planejamento é infinito. No segundo caso, a demanda dos produtos é dinâmica, o tempo é dividido e o horizonte de planejamento é finito. O Problema Econômico de Dimensionamento de Lotes e Programação da Produção (PEDLPP) e o Problema Capacitado de Dimensionamento de Lotes e Programação da Produção (PCDLPP) são exemplos de modelos de dimensionamento de lotes e modelos de dimensionamento de lotes dinâmico, respectivamente.

Se tratando da divisão dos períodos de planejamento, Belvaux e Wolsey (2001) definem dois tipos distintos de terminologias para os problemas: problemas *small bucket* e problemas *large bucket*. Nos problemas *small bucket* um único produto pode ser produzido em uma única máquina no mesmo período de tempo (micro-períodos). Nos problemas *large bucket*, vários produtos são produzidos em uma única máquina por período de tempo.

Segundo Sahlinga *et al*. (2009) os modelos *large bucket* apresentam a vantagem de permitir um flexível re-sequenciamento na produção, dentro do período considerado, enquanto os modelos *small bucket* integram as decisões de dimensionamento e seqüenciamento dos lotes. São exemplos de problemas *small bucket* e *large bucket* o Problema de Dimensionamento de Lotes e Programação da Produção com Setup Contínuo (PDLPPSC) e o Problema Capacitado de Dimensionamento de Lotes e Programação da Produção (PCDLPP), respectivamente (JANS, 2007).

O presente trabalho desenvolve métodos para solucionar o Problema Integrado de Dimensionamento de Lotes e Programação da Produção (PIDLPP). De acordo com os conceitos apresentados, o PIDLPP pode ser classificados como um **problema *small bucket* de programação da produção e dimensionamento de lotes dinâmico**.

3 MATERIAIS E MÉTODOS

Neste capítulo, o problema estudado e os métodos de resolução propostos serão apresentados em detalhe. A seção 3.1 descreve as principais características do problema com ênfase no aspecto da integração de dois cenários produtivos. Esse aspecto diferencia o problema estudado de outros problemas de dimensionamento de lotes e programação da produção. A seção 3.2 apresenta as buscas locais desenvolvidas começando pela representação e inicialização das soluções. Também são descritos os movimentos de busca local e a geração das vizinhanças, terminando com a apresentação dos pseudocódigos dos métodos. A seção 3.3 faz uma breve conclusão sobre os tópicos mostrados neste capítulo.

3.1 Problema Integrado de Dimensionamento de Lotes e Programação da Produção

O Problema Integrado de Dimensionamento de Lotes e Programação da Produção é um problema baseado na situação real de fábricas de refrigerantes onde as decisões de dimensionamento e planejamento da produção devem ser tomadas em dois níveis interdependentes, conforme a Figura 1.

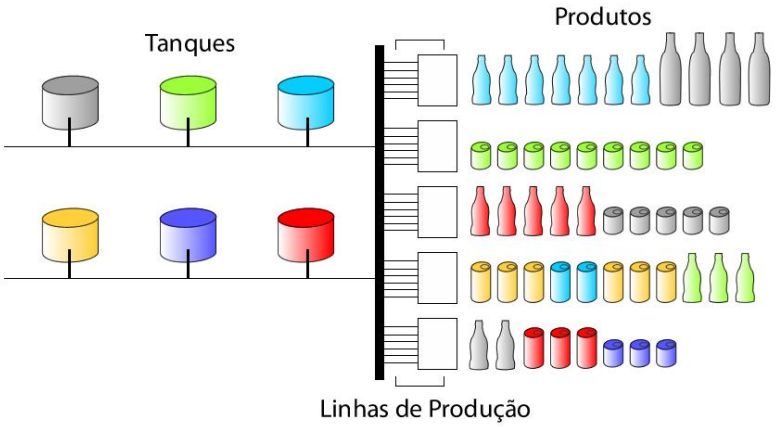

Figura 1 - Níveis de Produção do PIDLPP (ROSA, 2009)

O nível de tanques é responsável pelo armazenamento de xaropes, matéria prima para a produção da bebida, que escoam para o nível de linhas. O tipo de xarope e o tipo de engarrafamento (plástico, vidro, 2*lt*, 1*lt*, 600*ml*, etc) realizado dá origem ao produto final.

A quantidade e o momento de armazenamento dos xaropes devem ser estabelecidos para cada tanque. Cada tanque só pode ser reabastecido quando completamente escoado. Um tanque pode escoar xarope para várias linhas, possui uma capacidade máxima e uma quantidade mínima a ser armazenada. Essa capacidade máxima e mínima do tanque pode variar. Um tanque não pode armazenar dois tipos diferentes de xaropes ao mesmo tempo. Por outro lado, o mesmo xarope pode ser armazenado em diferentes tanques simultaneamente.

As decisões no nível de tanques envolvem custos de troca e custos de estocagem. O custo de troca para um determinado xarope depende do xarope anteriormente armazenado. Isso ocorre pelo fato de que um tanque precisar ser limpo antes de receber outro tipo de xarope. Isso também gera um tempo para que a limpeza seja realizada. O custo de estoque caracteriza-se pela deterioração do xarope que fica armazenado por longos períodos.

A quantidade e o momento de envase dos produtos finais devem ser estabelecidos nas linhas de produção. Uma linha pode ser abastecida por vários tanques e realizar o envase dos xaropes em diferentes tipos de produtos. Um produto pode possuir custo de produção e tempo de processamento diferentes em linhas distintas. Além disso, uma ou mais linhas podem produzir um mesmo tipo de produto. O custo e o tempo de troca para um tipo de produto depende do produto anteriormente configurado na linha. Em outras palavras, a sequência "refrigerante normal - refrigerante *diet*" pode diferir em custo e tempo da sequência "refrigerante *diet* - refrigerante normal". O custo de estoque caracteriza-se pelo custo de armazenamento dos produtos que excederem a demanda em um período.

Um envase não pode ser realizado sem que o xarope esteja armazenado em algum tanque e não faz sentido armazenar um xarope sem que este seja utilizado. Quando um tanque vai ser reabastecido, o mesmo para de escoar xarope para as linhas. Dessa forma, temos que as decisões tomada para tanques e as decisões tomadas para linhas de produção estão integradas e são interdependentes.

Contudo, todas as decisões do PIDLPP, tanto nos tanques quanto nas linhas de produção, devem ser feitas de forma a atender a demanda por produtos, respeitar as restrições do problema e minimizar todos os custos envolvidos.

Toledo *et al.* (2007) descreveram um modelo matemático inteiro-misto para o PIDLPP e propuseram um conjunto de instâncias artificiais divididas em instâncias de menor e média dimensão. Essas instâncias foram resolvidas utilizando um algoritmo de *branch and cut*. Todavia, as soluções ótimas foram encontradas somente para as instâncias de menor dimensão, tendo em vista a alta complexidade do PIDLPP.

Segundo Toledo *et al.* (2009), a complexidade do PIDLPP se deve a integração de dois Problemas Capacitados de Dimensionamento de Lotes e Programação da Produção (PCDLPP), um para o nível de tanques e outro para o nível de linhas. Bitran e Yanasse (1982) provam que o PCDLPP é NP-Difícil.

3.2 Métodos de Resolução

3.2.1 Representação das Soluções

As soluções serão representadas utilizando o mesmo esquema de Toledo *et al* (2009). Neste trabalho, o autor desenvolve uma complexa codificação capaz de representar a quantidade de produto, a linha que o produzirá e o tanque que receberá seu respectivo xarope. As soluções para o PIDLPP são representadas utilizando uma matriz bidimensional, onde as linhas são macro-períodos e as colunas contém informações a serem decodificadas para determinação da solução do problema. Observe a Figura:

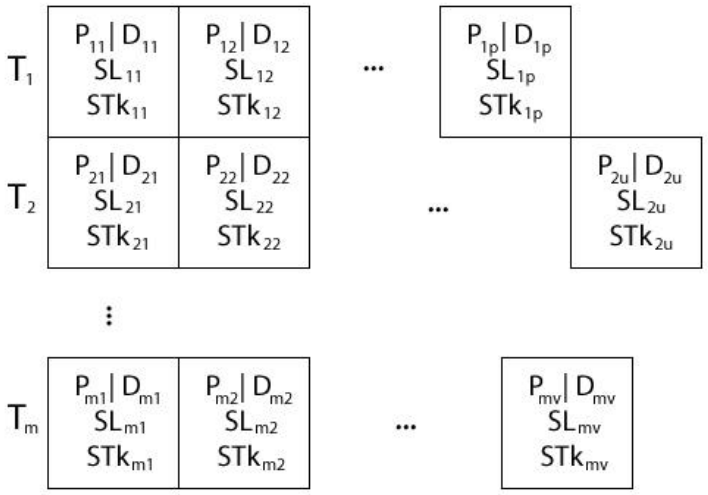

Figura 2 - Representação da solução (TOLEDO, 2005).

Note que o número de colunas é variável para cada linha (macro-período). Isso ocorre porque o número de lotes produzidos em cada macro-período não é fixo, ou seja, a quantidade de trocas de produto dentro de um macro-período pode diferir de outros. É importante lembrar que nos modelos *small bucket* somente é permitida produção de um único produto em cada micro-período. Dessa forma, a representação de Toledo (2005) segue é definida abaixo:

m	Quantidade de macro-períodos.
T_m	Macro-período m.
P_{mv}	Índice do produto do macro-período m e micro-período v.
D_{mv}	Quantidade (ou demanda) a ser produzida no macro-período m e micro-período v.
SL_{mv}	Conjunto de possíveis linhas responsável pela produção da demanda do produto no macro-período m e micro-período v.
STK_{mv}	Conjunto de possíveis tanques responsável por armazenar o xarope para produção da demanda do produto no macro-período m e micro-período v.

O conjunto SL_{mv} representa uma sequência de possíveis linhas para produção do produto. Esse conjunto é dado por $SL_{mv} = (\alpha_1, \alpha_2, ..., \alpha_k)$ onde k é o tamanho máximo

da sequência com $\alpha_i \in \{1, 2, ..., L\}$ e L sendo a quantidade máxima de linhas. O conjunto STK_{mv} representa uma sequência de possíveis tanques para armazenar o xarope do produto P_{mv}. Esse conjunto é dado por $STK_{mv} = (\beta_1, \beta_2, ..., \beta_k)$ em que $\beta_i \in \{1, 2, ..., 2\bar{L}\}$ onde $\bar{L}$ é a quantidade de tanques.

Nesse contexto, o tanque j é obtido a partir da regra β_i dada pela equação (1) descrita abaixo:

$$j = \begin{cases} \beta i, \ 1 \leq \beta i \leq \bar{L} \\ \beta i - \bar{L}, \ \bar{L} < \beta i \leq 2\bar{L} \end{cases} \quad (1)$$

Quando $j = \beta_i$, a ocupação do tanque j será feita após o escoamento do xarope nele contido. Quando $j = \beta_i - \bar{L}$, o tanque será ocupado imediatamente. Dessa forma, a codificação é capaz de representar soluções que preenchem parcialmente ou completamente os tanques.

Por exemplo, considere um problema com dois macro-períodos, duas linhas, dois tanques e dois produtos. As demandas dos produtos está descrita na Tabela 1.

Tabela 1 - Demandas dos Produtos por Macro-Período

	T_1	**T_2**
P_1	150	100
P_2	110	120

Duas possíveis soluções para este exemplo são demonstradas na Figura 3.

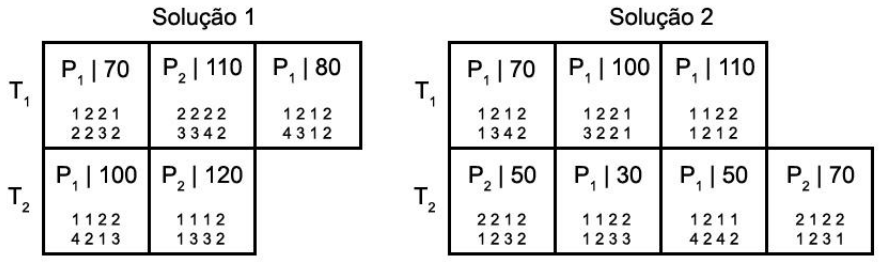

Figura 3 - Duas diferentes soluções

Note que na primeira solução 10 unidades de P_1,referente ao segundo macro-período, são distribuídas no primeiro macro-período. As demais demandas são alocadas nos respectivos períodos. Já na segunda solução, 20 unidades de P_1 e 10 unidades de P_2,referente ao segundo macro-período, são produzidos no primeiro macro-período. Dessa forma temos que é possível adiantar a produção de um macro-período, mas nunca atrasá-la.

O processo de transformar a informação codificada em uma programação da produção (decodificação) no PIDLPP começa na primeira posição do último macro-período (última linha e primeira coluna). Ela irá terminar na última posição do primeiro macro-período (primeira linha e última coluna). A decodificação consiste em selecionar um par de regras (α_i, β_i) dos conjuntos SL_{tn} e STK_{tn} respectivamente e tentar alocar a demanda D_{tn} à linha α_i e seu xarope ao tanque j proveniente da regra β_i. Caso não haja capacidade suficiente na linha ou tanque, a quantidade possível será alocada e o próximo par $(\alpha_{i+1}, \beta_{i+1})$ é selecionado. O processo é repetido até que $D_{tn} = 0$.

Como ilustração do processo de decodificação, observe o estabelecimento da programação da produção abaixo utilizando a solução 1 da Figura 3:

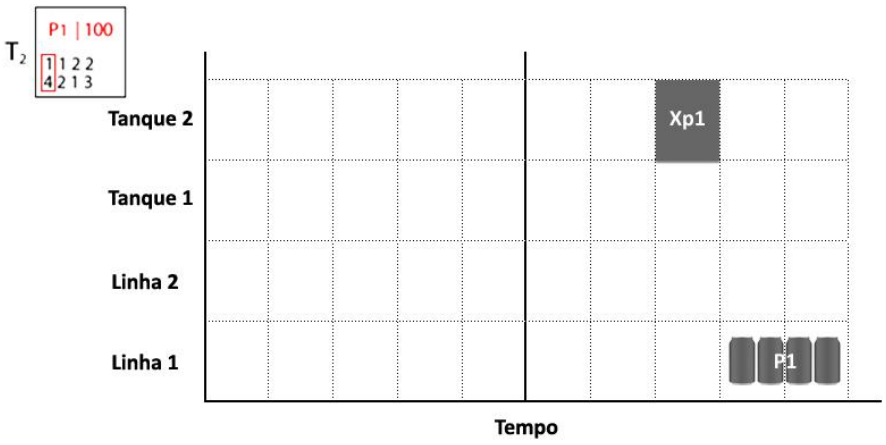

Figura 4 - Decodificação da Primeira Entrada da Solução 1

A Figura 4 ilustra o primeiro passo da decodificação. A primeira entrada é selecionada e então o par de regras *(1, 4)* define onde serão produzidas as 100 unidades de P_1 e onde será armazenado o xarope para produção, respectivamente. Como $\beta = 4$, o *tanque 2* teria que ser ocupado imediatamente (veja equação 1), porém não há nenhum lote de xarope no tanque então um novo lote é criado. Além disso, a preparação do *tanque 2* pelo xarope *Xp1* deve ser feita anteriormente a produção para a sincronização entre linhas e tanques. Desse forma, na Figura 4, na *linha1* está representado o tempo de produção do produto P_1. No *tanque 2*, está representado o tempo de preparação do tanque com a matéria prima para uso na *linha 1*.

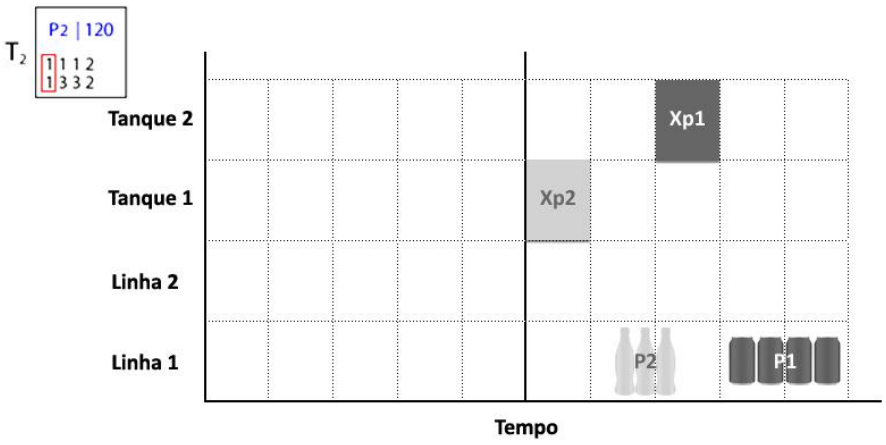

Figura 5 - Decodificação da Segunda Entrada da Solução 1

Para a segunda entrada o par de regras *(1, 1)* é selecionado. Isso significa que as 120 unidades do produto P_2 será produzida pela *linha 1* e o xarope para a produção será armazenado como novo lote no *tanque 1*. Porém, já há produção de P_1 na *linha 1*, o que significa que a produção de P_2 deve ser um pouco atrasada por causa do tempo de troca entre *P2* e *P1*.

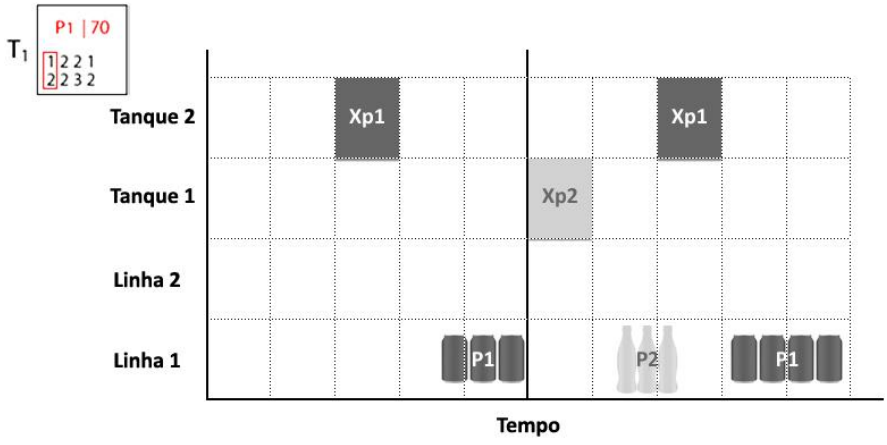

Figura 6 - Decodificação da Terceira Entrada da Solução 1

A terceira entrada é referente a 70 unidades de P_1 a serem produzidas no primeiro macro-período. Dessa forma, o par de regras *(1, 2)* é selecionado e então as 70 unidades são alocadas a *linha 1* e um novo lote do xarope *Xp1* para a produção é armazenado no *tanque 2*.

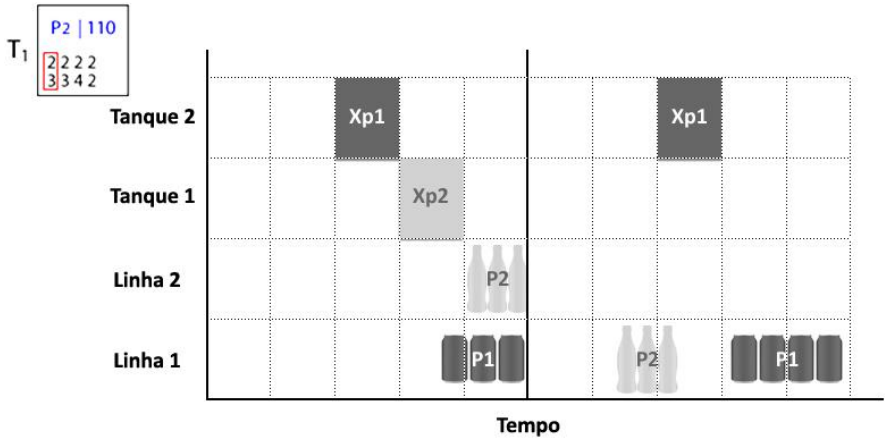

Figura 7 - Decodificação da Quarta Entrada da Solução 1

Para a quarta entrada, o par de regras *(2, 3)* é selecionado. Isso significa que as 110 unidades de P_2 serão produzidos na *linha 2* e que o xarope *Xp2* ocupará imediatamente o *tanque 1*. Dessa forma, o lote anterior é atrasado e o novo lote é adicionado, gerando estoque de *Xp2* entre o macro período 1 e 2. Isso só ocorre quando a restrição de capacidade do tanque não é violada.

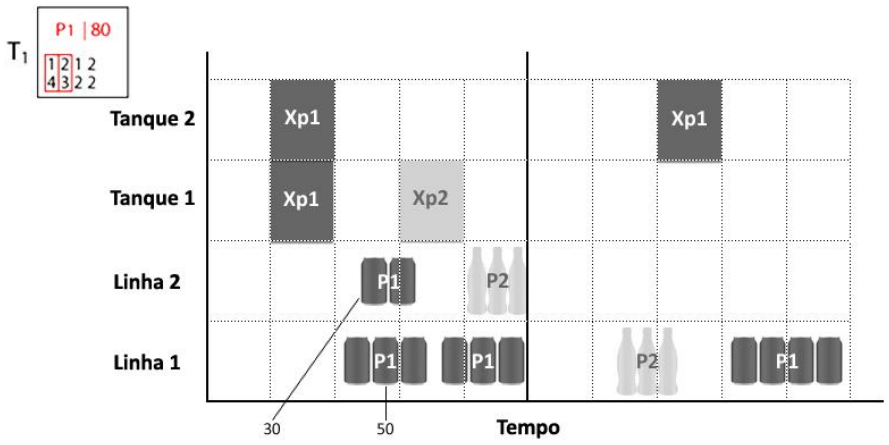

Figura 8 - Decodificação da Quinta Entrada da Solução 1

Para a quinta entrada, inicialmente o par de regras *(1, 4)* é selecionado, o que indica que as 80 unidades de P_1 serão produzidas pela *linha 1* e que o xarope *Xp1* ocupará *o tanque 2* imediatamente. Considerando que o tanque não tenha capacidade suficiente para atrasar o lote de *Xp1*, referente ao macro-período 2, um novo lote é então criado. Supondo que a *linha 1* só tenha capacidade para produzir 50 unidades de P_1, o próximo par de regras *(2, 3)* é selecionado. Isso indica que as 30 unidades restantes de P_1 devem ser produzidas pela *linha 2* e o xarope *Xp1* deve ocupar imediatamente o *tanque 1*. Como o último lote do tanque é referente a um xarope diferente, um novo lote para *Xp1* é criado.

Com a programação da produção definida é possível determinar a qualidade da solução utilizando a função objetivo proposta em Toledo *et al.* (2007):

$$\sum_{i=1}^{J}\sum_{j=1}^{J}\sum_{l=1}^{L}\sum_{s=1}^{T\cdot S} s_{ijl} z_{ijls} + \sum_{j=1}^{J}\sum_{t=1}^{T} h_j I_{jt} + \sum_{j=1}^{J}\sum_{l=1}^{L}\sum_{s=1}^{T\cdot S} v_{jl} q_{jls} +$$
$$+ \sum_{i=1}^{\bar{J}}\sum_{j=1}^{\bar{J}}\sum_{k=1}^{\bar{L}}\sum_{s=1}^{T\cdot\bar{S}} \bar{s}_{ijk} \bar{z}_{ijks} + \sum_{j=1}^{\bar{J}}\sum_{k=1}^{\bar{L}}\sum_{t=1}^{\bar{T}} \bar{h}_j \bar{I}_{jk,t\cdot T^m} + \sum_{j=1}^{\bar{J}}\sum_{k=1}^{\bar{L}}\sum_{s=1}^{T\cdot\bar{S}} \bar{v}_{jk} \bar{q}_{jks} + \quad (2)$$
$$+ M \sum_{j=1}^{J} q_j^0$$

As variáveis q_j^0 e M são variáveis utilizadas para penalizar soluções que não conseguem cumprir toda demanda. As demais variáveis e seu significado específico estão descritos no Anexo A. Em linhas gerais, a equação (2) descreve os custos de troca, custos de estocagem e custos de produção para produtos nas linhas (primeira linha da equação) e xaropes nos tanques (segunda linha da equação). Além disso, a penalização da demanda não atendida também é considerada (terceira linha da equação 2). O objetivo do PIDLPP é obter uma programação da produção que atenda toda demanda, minimizando os custos envolvidos.

3.2.2 Inicialização

Os métodos de busca local abordados neste trabalho necessitam de uma solução inicial para realizar a pesquisa no espaço de busca. Toledo *et al.* (2009) descreveram um processo de inicialização de soluções do PIDLPP usando a codificação anteriormente descrita. O pseudocódigo deste processo é dado na Figura 9.

```
para (t = 1 até T)
   repetir
      selecione aleatoriamente um produto i ∈ {1,...,J} com
      demanda d_it > 0
      L = conjunto de linhas l que processam i
      Tk = conjunto de tanques j que armazenam o xarope de i
      enquanto (d_it > 0)
         determine aleatoriamente D_m ≠ 0 com D_m ∈[0, d_it]
         insira D_m na primeira posição de t
         d_it = d_it - D_m
         para (i = 1 até k)
            gerar aleatoriamente valores para SL_mn e STk_mn com
            α_i ∈ L e β_i ∈ Tk
         fim para
      fim enquanto
   até que toda demanda em t tenha sido distribuída no
indivíduo
fim para
```

Figura 9 - Processo de inicialização (Toledo *et al.*, 2009).

Neste processo, as demandas são distribuídas em lotes de tamanhos aleatórios dentro de seus respectivos macro-períodos, obedecendo ao lote mínimo e máximo. As regras de sequência SL_{mn} e STK_{mn} são criadas de forma que todos os pares (α_i, β_i) representem apenas linhas e tanques que processam o produto do seu respectivo lote, ou seja, linhas que não são capazes de produzir o produto e tanques que não armazenam o xarope do produto não entram nos valores das regras de sequência do lote.

A busca tabu e o *simulated annealing* propostos no presente trabalho utilizam o processo de Toledo *et al*. (2009) descrito acima para obtenção de uma solução inicial.

3.2.3 Movimentos de Busca

Duas versões de movimentos de busca em vizinhança foram criadas: uma versão **aleatória e uma versão exaustiva. A versão aleatória** seleciona entradas da representação da solução de forma aleatória e executa o movimento. A versão **exaustiva procura executar** o movimento para todas as combinações possíveis dessas entradas. Os movimentos podem ser divididos em sete operações distintas:

1. **Operação de Troca**: Duas entradas são selecionadas e suas posições são trocadas (Figura 10).

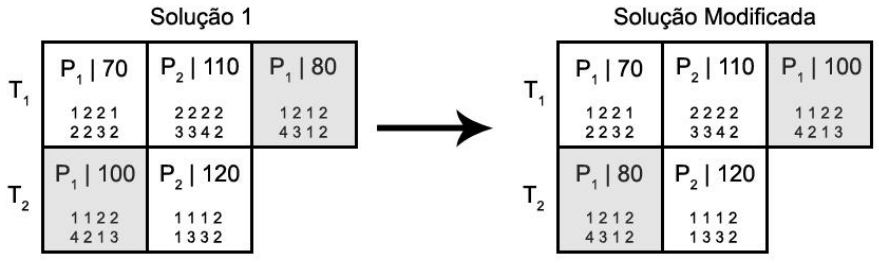

Figura 10 - Operação de Troca

2. **Operação de Inserção**: Uma entrada e uma posição são selecionadas e a entrada é inserida na posição (Figura 11).

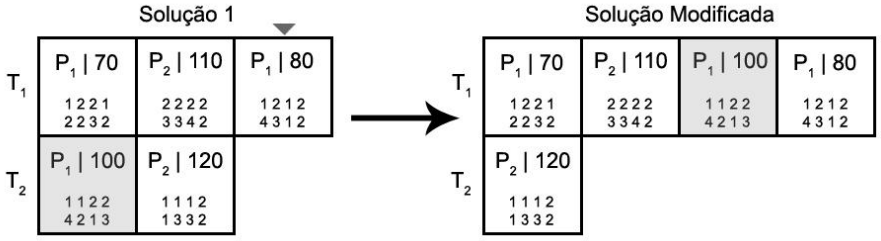

Figura 11 - Operação de Inserção

3. **Operação de Fusão**: Duas entradas com produtos iguais são selecionadas e a demanda da primeira entrada é atribuída à demanda da segunda entrada (Figura 12).

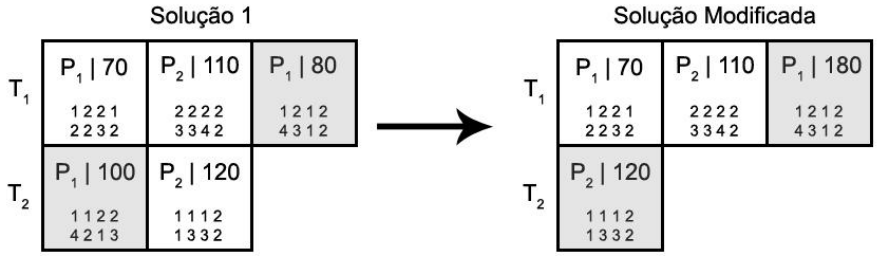

Figura 12 - Operação de Fusão

4. **Operação de Divisão**: Uma entrada e uma posição são selecionadas e parte da demanda da entrada é inserida na posição (Figura 13).

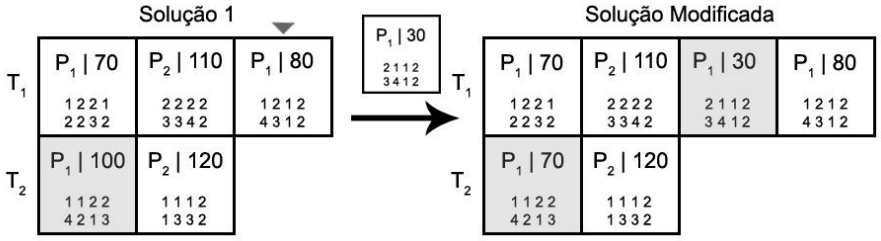

Figura 13 - Operação de Divisão

5. **Operação de Regras**: Uma entrada é selecionada e suas regras de linhas e tanques são reinicializadas (Figura 14).

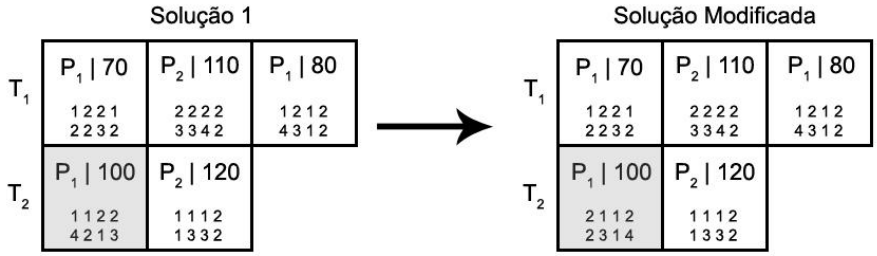

Figura 14 - Operação de Regras

6. **Operação de Embaralho**: Um macro-período é selecionado e as posições das entradas são embaralhadas (Figura 15).

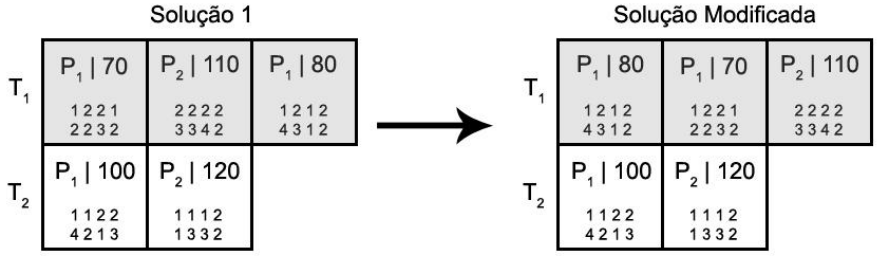

Figura 15 - Operação de Embaralho

7. **Operação de Inversão**: Um macro-período é selecionado e as posições das entradas são invertidas (Figura 16).

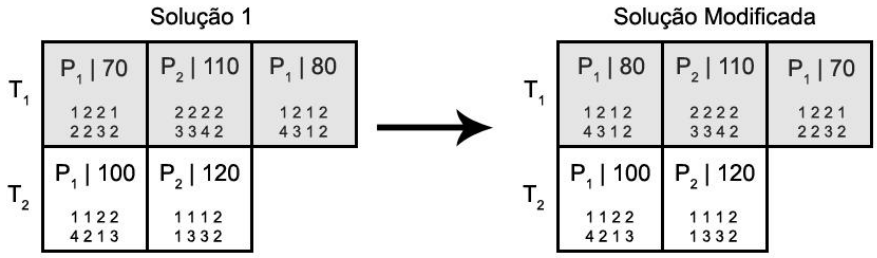

Figura 16 - Operação de Inversão

Dessa forma, temos sete operações definidas gerando um total de quatorze movimentos (7 aleatórios e 7 exaustivos) executados pelas buscas locais. Inicialmente, um estudo sobre quais as melhores versões de movimentos, exaustivos, aleatórios ou uma combinação entre eles, é realizado. Este estudo será descrito na próxima seção.

3.2.4 Geração de Vizinho

A geração de vizinhos define o modo como a procura por melhores soluções será executada através do espaço de busca. Três aspectos serão analisados na geraçãoo de vizinhança: as **versões de movimentos**, a **escolha do movimento** e a forma de executar o movimento.

1. **Versões de Movimentos:** O conjunto de movimentos executados para geração de vizinhos pode considerar somente movimentos aleatórios, somente movimentos exaustivos ou a combinação entre eles.
2. **Escolha do Movimento:** O movimento a ser executado na iteração pode ser escolhido de forma determinística ou de forma aleatória.
3. **Forma de Execução:** O movimento escolhido pode ser executado por várias iterações ou por apenas uma. A forma de execução impacta somente nos movimentos aleatórios. Nos movimentos exaustivos a execução ocorre sempre por várias iterações.

Na **Versões de Movimentos,** num movimento aleatório todas as entradas e posições da representação da solução são selecionadas aleatoriamente. Por exemplo, no

movimento de fusão, uma entrada da matriz de representação da solução é selecionada aleatoriamente. A partir dessa escolha, uma outra entrada com o mesmo produto também é aleatoriamente selecionada para execução do movimento de fusão. No movimento exaustivo, todas as posições e entradas possíveis devem ser testadas para seleção daquela com melhor resultados. Por exemplo, no movimento de fusão, cada entrada seria selecionada e, a partir dela, todas as fusões possíveis com as demais entradas com mesmo produto seriam testadas. A combinação dos movimentos significa que todos os 14 movimentos (exaustivos e aleatórios) podem ser executados de acordo com a escolha do movimento adotada.

Na versão da Escolha do Movimento, um movimento é selecionado de forma determinística utilizando a operação MOD, a partir do número de movimentos disponíveis e do número da iteração em execução (equação 3).

$$m \equiv iteração \ (\mathbf{MOD}\ nMovimentos) \quad (3)$$

A variável *m* utilizada acima representa o número do movimento escolhido. O parâmetro *nMovimentos* define a quantidade de movimentos. A variável *iteração* é um contador que representa a **iteração geral** do método.

A escolha do movimento de forma aleatória é feita através da geração de um número aleatoriamente escolhido entre o número de movimentos possíveis (equação 4).

$$m \in [1, nMovimentos] \qquad (4)$$

Para facilidade de compreensão, utilizaremos as letras **A** e **B** para indicar a primeira e segunda possibilidade de Escolha do Movimento. As duas maneiras de execução do movimento impactam na condução da pesquisa pelo espaço de busca.

No contexto dos métodos propostos neste trabalho, executar um movimento aleatório várias vezes sobre uma mesma solução significa varrer mais persistentemente a vizinhança gerada por aquele movimento. Por outro lado, executar uma única vez significa dar maior possibilidade de um novo movimento executar a busca em outro

tipo de vizinhança. Serão utilizadas as denominações **V1** e **V2** para indicar a execução por várias iterações e a execução por uma única iteração, respectivamente.

O pseudocódigo da maneira de execução **V1** é descrito na Figura 17.

```
para (1 até nMovimentos)
       escolha um movimento m
       enquanto (nMaxIterações ainda não atingido)
              execute o movimento m sobre a solução s
       fim enquanto
fim para
```

Figura 17 - Forma de execução V1

O pseudocódigo da maneira de execução **V2** é descrito na Figura 18.

```
enquanto (nMaxIterações ainda não atingido)
       escolha o movimento m
       execute o movimento m sobre a solução s
fim enquanto
```

Figura 18 - Maneira de execução V2

A variável *nMaxIterações* representa em **V1** a quantidade de iterações que o movimento *m* será executado sobre a solução *s* e em **V2** a quantidade vezes que um movimento *m* será escolhido e executado sobre *s*. O valor dessa variável é definido como 70% da quantidade de entradas que a solução *s* possui.

No discorrer deste trabalho, para se referir a um determinado método de busca local é descrito também a maneira que os vizinhos são gerados, utilizando a seguinte denotação:

[método] [(versões de movimentos)] [versão de escolha] [versão de execução].

Por exemplo: uma busca tabu (BT) que gera os vizinhos utilizando somente o conjunto de movimentos aleatórios (aleat.) com a escolha aleatória de movimento (B) e executando um movimento por várias iterações (V1) é referenciada por **BT (aleat.) B V1.**

3.2.5 Busca Tabu

A busca tabu (BT) proposta neste trabalho como abordagem para o PIDLPP possui tamanho de lista de proibição de movimentos (*tamanhoLista*) determinado de forma dinâmica. Há uma troca entre diversificação e intensificação durante as iterações do método ao utilizar lista dinâmica. A atualização do tamanho da lista pode ser realizada com probabilidade de 50% no fim de cada iteração do método. Os vizinhos são gerados somente com movimentos que não pertençam à lista tabu *T* e utilizam alguma combinação das versões de geração descritas na seção anterior. O pseudocódigo do método é descrito na Figura 19.

```
Inicialize uma solução s

s* = s

T = ∅

tamanhoLista ∈ [1, nMovimentos-1]

enquanto (limite de tempo ainda não atingido)

       s' = melhor vizinho gerado de s utilizando movimentos
       que não pertencem a T

       m = movimento que gerou s'

       se ( f(s') < f(s*) )

              s* = s'

       fim se

       s = s'

       T = T ∪ {m}

       x ∈ [0, 1]

       se (x <= 0.5)

              tamanhoLista ∈ [1, nMovimentos-1]

              atualize (T, tamanhoLista)

       fim se

fim enquanto
```

Figura 19 - Pseudocódigo da Busca Tabu

A função $f()$ calcula a qualidade de uma solução de acordo com o processo de decodificação e com a Equação 2 descritos na seção 3.3.1. O movimento m que gera o melhor vizinho s' da solução s é adicionado à lista tabu T a cada iteração do método. Caso o tamanho da lista seja excedido, tanto pela adição de movimentos ou quanto pela atualização do tamanho máximo, os movimentos mais antigos são liberados.

3.2.6 *Simulated Annealing*

Da mesma forma que na BT, os vizinhos para o *simulated annealing* (SA) também são gerados utilizando alguma combinação das versões de geração descritas na seção 3.3.4. O decréscimo da temperatura T é feito a cada iteração de geração de vizinho e é descrito por $T = T / (1 + (\beta * \sqrt{T}))$ (SOUZA, 2005). O parâmetro β possui valor 0,7. Quando T atinge o valor de 0,01 um reaquecimento é realizado e T assume novamente o valor da temperatura inicial T_0. O pseudocódigo do método é descrito na Figura 20.

```
Inicialize uma solução s; s* = s; T = T0
enquanto (limite de tempo ainda não atingido)
        sTemp = s
        para ( cada s' = vizinho gerado de sTemp )
                Δ = f(s') - f(s)
                se ( Δ < 0 )
                        s = s'
                        se ( f(s') < f(s*) )
                                s* = s'
                        fim se
                se não
                        x ∈ [0, 1]
                        se ( x < e^(-Δ/T) )
                                s = s'
                        fim se
                fim se
                T = T / (1 + (β * √T))
        fim para
        se (T < 0.01)
                T = T0
        fim se
fim enquanto
```

Figura 20 - Pseudocódigo do *Simulated Annealing*

Enquanto a BT considera somente o melhor vizinho gerado para atualizar a solução *s*, o SA considera todos os vizinhos. Isso acontece por causa da verificação probabilística que o SA adota na atualização da solução *s*. Todos os vizinhos gerados a partir de *sTemp* podem substituir *s* desde que possuam qualidade melhor ou uma qualidade pior que poderá ser aceita com probabilidade dada por $e^{-\Delta/T}$. A qualidade das soluções também são calculadas pela função $f()$ e a temperatura inicial T_0 é ajustada como 1000.

3.3 Conclusão

Neste capítulo foi visto uma breve descrição sobre o cenário do PIDLPP. Além disso, toda a metodologia empregada neste trabalho também é descrita.

A codificação proposta em Toledo (2005) e o cálculo da qualidade da solução de Toledo *et al.* (2007) serão utilizados como ferramentas para a BT e o SA. Sete diferentes operadores de busca são definidos e a geração de vizinhança é feita utilizando estes operadores em conjunto com os tipos de movimentos, a escolha do movimento e a maneira de execução. Os pseudocódigos e os parâmetros da BT e do SA também foi estabelecido.

4 RESULTADOS COMPUTACIONAIS

4.1 Introdução

O presente capítulo apresenta os resultados computacionais obtidos para os métodos propostos. Inicialmente, os conjuntos de instâncias e as métricas adotadas na avaliação dos resultados são apresentados. Em seguida, um estudo envolvendo o uso dos movimentos e as estratégias de exploração em vizinhança são descritos. Esse estudo é conduzindo sobre instâncias com variados níveis de complexidade. Os resultados obtidos ajudam a definir as melhores abordagens para a busca tabu e o *simulated annealing* descritos anteriormente. Essas melhores abordagens são então comparadas aos resultados obtidos considerando todas as instâncias dos conjuntos propostos.

4.2 Instâncias e Medidas de Avaliação

As metas-heurísticas do presente trabalho foram avaliados utilizando quatro conjuntos de instâncias obtidos de Toledo *et al.* (2009). Os três primeiros conjuntos foram gerados artificialmente e variam entre instâncias de pequena e média complexidade. O quarto conjunto trata-se de instâncias criada a partir da programação real de uma fábrica de refrigerantes.

A Tabela 2 apresenta os parâmetros que definem os três primeiros conjuntos.

Tabela 2 - Parâmetros para os três primeiros conjuntos de instâncias

Conjunto	$L / \overline{L} / J / \overline{J} / T$	Qtd. Variáveis Binárias no Modelo	Qtd. Variáveis Contínuas no Modelo	Qtd. Restrições no Modelo
S1	2/2/2/1/2	210	533	575
S2	3/2/3/2/3	642	2662	2071
S3	4/2/4/2/4	1077	5590	5735

Os conjuntos S1 e S2 possuem 10 instâncias cada. O conjunto S3 possui 9. Cada instância foi gerada por Toledo *et al.* (2009) seguindo os parâmetros da Tabela 3.

Tabela 3 - Parâmetros de geração de S1, S2 e S3 (Toledo *et al*, 2009)

Parâmetro	Valor	Significado
h_j	1($/u)	Custo de estoque do produto *j* (preço por unidade)
$\bar{h}_j$	1($/u)	Custo de estoque do xarope *j* (preço por unidade)
v_{jl}	1($/u)	Custo de produção do produto *j* na linha *l* (preço por unidade)
v_{jk}	1($/u)	Custo de produção do xarope *j* no tanque *k* (preço por unidade)
Q_k	5000*l*	Capacidade máxima do tanque *k*
Q^k	1000*l*	Capacidade mínima do tanque *k*
C	5	Capacidade disponível em cada período
st_{ijl}	[0.5; 1]	Tempo de troca do produto *i* para o produto *j* na linha *l*
sst_{ijk}	[1; 2]	Tempo de troca do xarope *i* para o xarope *j* no tanque *k*
sc_{ijl}	1000*st_{ijl}	Custo de troca do produto *i* para o produto *j* na linha *l*
ssc_{ijk}	1000*sst_{ijk}	Custo de troca do xarope *i* para o xarope *j* no tanque *k*
P_{il}	[1000; 2000]	Tempo de processamento do produto *i* na linha *l*
d_{it}	[500; 10000]	Demanda do produto *i* no macro período *t*
r_{ij}	[0.3; 3]	Fator de conversão da quantidade de xarope *j* necessária para produzir o produto *i*

O quarto e último conjunto foi estabelecido utilizando parâmetros de uma programação real de uma fábrica de refrigerantes. Um total de 6 instâncias foi criado. Os parâmetros de cada instância são dados pela Tabela.

Tabela 4 - Parâmetros das Instâncias do Conjunto S4

Instância	$L / \bar{L} / J / \bar{J} / T$
A1	5/9/33/11/1
A2	6/9/49/14/2

Tabela 5 - Parâmetros das Instâncias do Conjunto S4

A3	6/9/58/15/3
B1	6/10/52/19/1
B2	6/10/56/19/2
B3	6/10/65/21/3

O custo de estoque e de produção, tanto de produtos quanto de xaropes é de 1 por unidade. A capacidade mínima dos tanques é de 1000l e a capacidade máxima é de 24000l. O tempo de troca de produtos é o mesmo para todas as linhas, e leva 0.5h. O tempo de troca de xaropes nos tanques também é o mesmo, porém leva 1h para xaropes iguais ou 2h para xaropes diferentes. O tempo de processamento dos produtos varia entre 50 e 2000 unidades por hora e a demanda também varia entre 47 unidades a 180000 unidades no mesmo período.

Para medida de comparação, todas as instâncias dos conjuntos S1, S2 e S3 foram resolvidas utilizando um algoritmo *Branch and Cut* do pacote computacional GAMS/Cplex versão 2.0.10.0 e 7.0 respectivamente. Esse algoritmo foi executado por uma hora, retornando a solução ótima para o conjunto S1 e a melhor solução encontrada para os conjuntos S2 e S3. Nesses dois conjuntos o método exato não foi capaz de retornar a solução ótima dentro do tempo de execução. A comparação do conjunto S4 é feita utilizando o custo estimado pela fábrica de refrigerantes na programação de cada instância.

As buscas locais propostas neste trabalho foram executadas por 30 minutos para as instâncias dos conjuntos S1, S2 e S3 e por uma hora para as instâncias do conjunto S4. Um total de 10 execuções foram realizadas sobre cada instância. Os computadores utilizados nestes testes possuem processador Core 2 Duo, 2.0 GHz; com 2 GB de memória.

Dessa forma, é possível calcular o desvio dos resultados obtidos pelas buscas locais em relação aos encontrados pelo algoritmo *Branch and Cut* ou pela estimação da fábrica utilizando a Equação 5.

$$\text{Desvio}(\%) = 100\frac{(Z - Z^*)}{Z^*} \qquad (5)$$

Z é a média das 10 execuções realizadas e *Z** é o resultado retornado pelo algoritmo *Branch and Cut* ou a estimação da fábrica. Dessa forma, um desvio negativo indica que a média *Z* obteve melhor resultado do que o parâmetro de comparação *Z**.

4.3 Testes Preliminares e Estudo de Vizinhança

Inicialmente, foram selecionadas as instâncias S1-I2, S3-I5, B1 e B2 como conjunto de instâncias testes **T1** para as avaliações preliminares de estudo de vizinhança. Esses testes preliminares foram feitos em quatro etapas e tem como objetivo refinar as características de geração de vizinhança descritas na seção 3.3.4.

A primeira etapa consiste no refinamento das melhores metodologias de **tipos de movimentos** a serem utilizados. A segunda etapa avalia a forma de **escolha do movimento**. A terceira etapa estuda duas **formas de execução dos movimentos**. A quarta e última etapa define a quantidade dos movimentos utilizados. Nessa etapa, estatísticas da quantidade de melhorias de cada movimento sobre a melhor solução encontrada são utilizadas. As Tabelas 5 e 6 abaixo mostram os resultados da primeira etapa de testes.

Tabela 6 - Avaliando Tipos de Movimentos para BT

	BT (aleat.) B V1		**BT (exaust.) B V1**		**BT (comb.) B V1**	
Inst.	**Infact.**	**Desvio**	**Infact.**	**Desvio**	**Infact.**	**Desvio**
S1-I2	0	**0,15**	0	2,78	0	2,58
S3-I5	0	**-4,79**	0	-3,25	0	1,38

Tabela 7 - Avaliando Tipos de Movimentos para BT

B1	0	**-4,51**	1	-0,71	0	-2,50
B3	0	**-3,53**	7	4,81	2	6,10

A busca tabu utilizando somente movimentos aleatórios obteve melhores desvios em todas as instâncias em relação às outras versões. Além disso, a versão aleatória não apresentou nenhuma execução com infactibilidade. A versão com movimentos exaustivos apresentou os piores desvios em S1-I2 e em B1. Além disso, a versão exaustiva apresenta desvio positivo e 7 infactibilidades para B3, demonstrando ineficiência quando a complexidade do problema aumenta. Uma infactibilidade também foi obtida em B1. A versão com os movimentos combinados obteve os piores desvios nas instâncias S3-I5 e B3. Além disso, a versão combinada ainda apresentou 2 infactibilidades para a instância B3.

Tabela 8 - Resultados da Primeira Etapa para SA

	SA (aleat.) B V1		**SA (exaust.) B V1**		**SA (comb.) B V1**	
Inst.	**Infact.**	**Desvio**	**Infact.**	**Desvio**	**Infact.**	**Desvio**
S1-I2	0	**2,82**	0	9,13	0	5,66
S3-I5	2	**-1,30**	1	7,30	0	5,95
B1	0	**-5,45**	1	-0,31	0	-1,76
B3	2	**-4,25**	10	-	10	-

Nos resultados obtidos para o *simulated annealing*, a versão aleatória também se sobre sai perante as demais versões em todas as instâncias em relação aos desvios e apresentou um total de 4 infactibilidade. A versão exaustiva mais uma vez demonstra ineficiência para a instância B3, não retornando nenhuma solução factível nas 10 execuções e, dessa forma, apresentando um total de 12 infactibilidades no conjunto de teste. Além disso, a versão exaustiva obteve os piores desvios em todas as instâncias. A

versão com os movimentos combinados também não retornam nenhuma solução factível para B3 e apresenta um total de 10 infactibilidades.

Considerando o desempenho dos dois métodos, a versão com movimentos aleatórios foi escolhida para a próxima etapa de avaliações.

A Tabela 7 e a Tabela 8 apresentam os resultados da segunda etapa de testes, ou seja, a etapa de avaliação da escolha do movimento.

Tabela 9 - Resultados da Segunda Etapa para BT

	BT (aleat.) A V1		**BT (aleat.) B V1**	
Inst.	**Infact.**	**Desvio**	**Infact.**	**Desvio**
S1-I2	0	0,39	0	**0,15**
S3-I5	0	**-4,94**	0	-4,79
B1	0	**-4,80**	0	-4,51
B3	3	**-3,64**	0	-3,53

Para a busca tabu, a versão de escolhas de movimentos do tipo A obteve melhores desvios em 3 das 4 instâncias do conjunto em relação a versão de escolhas de movimentos B, porém com pequena diferença nos resultados. Entretanto, a versão A apresenta 3 execuções infactíveis para a instância B3 sendo que a versão B não apresenta infactibilidade.

Tabela 10 - Resultados da Segunda Etapa para SA

	SA (aleat.) A V1		**SA (aleat) B V1**	
Inst.	**Infact.**	**Desvio**	**Infact.**	**Desvio**
S1-I2	0	6,55	0	**2,82**
S3-I5	0	**-2,84**	2	-1,30
B1	0	-4,53	0	**-5,45**
B3	1	4,23	2	**-4,25**

Para o *simulated annealing* a versão A alcançou melhor desvio somente na instância S3-I5. Além disso, a versão A obtém desvio positivo na instância B3, onde ainda apresenta 1 infactibilidade. A versão B apresenta um total de 4 infactibilidades e melhores desvios em 3 das 4 instâncias do conjunto. Assim, apesar da versão B apresentar um total de infactibilidades maior, o ganho nos desvios, principalmente na instância B3, é considerável. Além disso, a versão B apresenta desvios melhores do que a versão A nas instâncias S1-I2 e B1 onde ambas as versões não apresentam nenhuma infactibilidade. Dessa forma, a versão com escolha aleatória de movimentos (versão B) também foi escolhida para a próxima etapa.

A Tabela 9 e a Tabela 10 mostram os resultados da terceira etapa de testes, onde a forma de movimento é avaliada.

Tabela 11 - Resultados da Terceira Etapa para BT

	BT (aleat.) B V1		**BT (aleat.) B V2**	
Inst.	**Infact.**	**Desvio**	**Infact.**	**Desvio**
S1-I2	0	**0,15**	0	2,75
S3-I5	0	**-4,79**	0	-2,72
B1	0	**-4,51**	0	-3,64
B3	0	**-3,53**	0	-3,42

Tabela 12 - Resultados da Terceira Etapa para SA

	SA (aleat.) B V1		**SA (aleat.) B V2**	
Inst.	**Infact.**	**Desvio**	**Infact.**	**Desvio**
S1-I2	0	2,82	0	**2,79**
S3-I5	2	**-1,30**	0	-1,27
B1	0	**-5,45**	0	-5,35
B3	2	**-4,25**	1	-1,35

Na busca tabu, a versão V1 da forma de execução dos movimentos supera todos os desvios em relação à versão V2. Ambos não apresentam infactibilidades. Já para o *simulated annealing* a versão V1 obtém melhores desvios em 3 das 4 instâncias do conjunto, mas apresenta um total de 4 infactibilidades contra apenas 1 de V2. A versão V2 supera o desvio de V1 somente na instância S1-I2. Todavia, a diferença dos desvios é muito pequena, com exceção da instância B3 onde V1 apresenta 2 infactibilidades, sendo 1 a mais em relação a V2.

Dessa forma, há uma diferença de desempenho dos métodos relacionada à forma de execução dos movimentos. Assim, a BT com a versão V1 e SA com versão V2 são selecionados para a próxima etapa de testes.

A quarta e última etapa de testes consiste em retirar os movimentos com menos impacto no desempenho do método. Isso é feito tomando como base a quantidade de melhorias provocadas por um movimento em relação à melhor solução encontrada e o tempo gasto para a execução do movimento. A média das melhorias e do tempo gasto de cada movimento considerando 10 execuções em cada instância, tanto para a BT quanto para o SA, é apresentada na Tabela 11.

Tabela 13 - Estatísticas dos Movimentos

Movimentos	**Média de Melhorias**	**Média do Tempo Gasto (ms)**
Troca	45,87	34,79
Inserção	51,72	34,24
Fusão	41,83	34,02
Divisão	22,03	34,23
Regras	40,95	34,61
Embaralho	6,05	40,83
Inversão	0,53	38,68

Os movimentos de **Inversão** e de **Embaralho** apresentam menor média de melhorias e maior média de tempo gasto. Dessa forma, os testes da quarta etapa

consistem em retirar primeiramente o movimento de inversão e posteriormente o movimento de embaralho, verificando o impacto disso nos resultados. A Tabela 12 e a Tabela 13 apresentam os resultados da quarta etapa tanto para a **BT (aleat.) B V1** e para o **SA (aleat.) B V2** respectivamente.

Tabela 14 - Resultados da Quarta Etapa para BT

	BT 7 Mov.		**BT 6 Mov.**		**BT 5 Mov.**	
Inst.	**Infact.**	**Desvio**	**Infact.**	**Desvio**	**Infact.**	**Desvio**
S1-I2	0	**0,15**	0	**0,15**	0	0,18
S3-I5	0	**-4,79**	0	-3,90	0	-1,01
B1	0	-4,51	0	-4,89	0	**-5,47**
B3	0	-3,53	0	-4,11	2	**-4,12**

Tabela 15 - Resultados da Quarta Etapa para SA

	SA 7 Mov.		**SA 6 Mov.**		**SA 5 Mov.**	
Inst.	**Infact.**	**Desvio**	**Infact.**	**Desvio**	**Infact.**	**Desvio**
S1-I2	0	2,79	0	2,77	0	**2,73**
S3-I5	0	-1,27	0	**-1,49**	0	-1,26
B1	0	-5,35	0	**-5,66**	0	-5,44
B3	1	-1,35	1	**-1,61**	3	-1,42

A busca tabu utilizando 5 movimentos apresenta 2 infactibilidades e melhores desvios em B1 e B3. As demais versões não apresentam infactibilidades. A versão com 6 movimentos obtêm o melhor desvio somente em S1-I2 e melhores resultados em B1 e B3 em relação a versão com 7 movimentos. Além disso, a diferença no desvio para a instância B3 é de apenas 0,01% em relação à versão com 5 movimentos, que obteve infactibilidades nesta instância. O *simulated annealing* com 5 movimentos apresenta melhor desvio apenas em S1-I2 e um total de 3 infactibilidades contra apenas 1 das demais versões. A versão do SA com 6 movimentos apresenta melhores desvios em 3

das 4 instâncias e desvio muito próximo para a instância S1-I2 em relação a versão com 5 movimentos. A versão com 7 movimentos obtém melhor resultado em relação a versão com 5 movimentos apenas na instância S3-I5, porém a diferença é de apenas 0,01%. Dessa maneira, em ambos os métodos, busca tabu e *simulated annealing*, a versão com 6 movimentos possui melhor performance em relação às demais versões.

Assim, com os testes preliminares e com o estudo de geração de vizinhança, as melhores versões dos métodos de busca local são a **BT (aleat.) B V1** e o **SA (aleat.) B V2** ambos utilizando 6 dos 7 movimentos inicialmente propostos.

4.4 Resultados Finais e Comparação com Outro Método

As melhores versões das buscas locais obtidas com os testes preliminares foram executadas para todas as instâncias dos conjuntos S1, S2, S3 e S4. Além disso, o Algoritmo Genético (AG) de Toledo *et al.* (2008a) foi executado novamente nas mesmas máquinas utilizadas neste trabalho para efeito de comparação. A seção 4.4.1 apresenta os resultados para as instâncias artificiais (conjuntos S1, S2 e S3), a seção 4.4.2 apresenta os resultados para as instâncias industriais (conjunto S4).

4.4.1 Resultados Para os Conjuntos S1, S2 e S3

Os resultados para o conjunto S1 são apresentados na Tabela 14.

Tabela 16 - Resultados Finais do Conjunto S1

	AG		**BT**		**SA**	
Inst.	**Infact.**	**Desvio**	**Infact.**	**Desvio**	**Infact.**	**Desvio**
I1	0	0,06	0	**0,00**	0	0,30
I2	0	2,51	0	**0,17**	0	2,78
I3	0	**0,01**	0	**0,01**	0	**0,01**
I4	0	0,25	0	**-0,01**	0	**-0,01**
I5	0	1,25	0	**0,07**	0	2,18
I6	0	**0,02**	0	**0,02**	0	**0,02**
I7	0	**-0,01**	0	**-0,01**	0	**-0,01**

Tabela 17 - Resultados Finais do Conjunto S1

I8	0	**0,01**	0	**0,01**	0	**0,01**
I9	0	**0,00**	0	**0,00**	0	**0,00**
I10	0	**0,01**	0	0,22	0	**0,01**
Média		0,41		**0,05**		0,53

Este conjunto foi o único em que método *branch and cut* obteve o ótimo para todas as instâncias considerando a execução de 1 hora. Em geral a média das 10 execuções de cada meta-heurística foi igual ou muito próxima do ótimo mesmo sendo executadas por 30 minutos, sendo o maior desvio de 2,78% obtido pelo SA na instância I2. A BT obteve um total de 9 vitórias, contra 7 do SA e apenas 6 do AG. A soma do número de vitórias ultrapassa o número de instâncias no conjunto porque em várias instâncias as médias das 10 execuções dos métodos foram iguais. A média geral, mostrada na última linha da tabela, sumariza todos os desvios, onde a busca tabu aparece com a melhor média de desvio. O SA apresenta o pior desvio médio, porém com diferença de apenas 0,12% em relação ao GA. Nenhum método apresentou infactibilidade para este conjunto.

Os resultados para o conjunto S2 é apresentado na Tabela 15.

Tabela 18 - Resultados Finais do Conjunto S2

	AG		**BT**		**SA**	
Inst.	**Infact.**	**Desvio**	**Infact.**	**Desvio**	**Infact.**	**Desvio**
I1	0	-0,17	0	-0,26	0	**-0,29**
I2	0	0,70	1	**-0,33**	1	2,14
I3	0	3,87	0	**0,06**	0	0,19
I4	0	**0,89**	0	**0,89**	0	**0,89**
I5	0	**-1,32**	0	**-1,32**	0	**-1,32**
I6	0	-3,80	0	**-3,94**	0	**-3,94**

Tabela 19 - Resultados Finais do Conjunto S2

I7	0	1,19	0	**-1,06**	0	0,81
I8	0	-9,87	0	**-10,70**	0	-10,67
I9	1	12,09	0	**5,16**	3	8,86
I10	0	-0,46	0	**-1,37**	0	0,76
Média		0,31	0	**-1,29**		-0,26

No conjunto S2, a BT apresentou a melhor média geral com 9 vitórias em 10 possíveis. Além disso, a BT superou os desvios dos demais métodos em 6 instâncias. O SA é superior aos demais métodos apenas na instância I1 e obteve um total de 4 vitórias. Porém, o SA é superior em relação ao AG em 6 instâncias e apresentou melhor média geral. Contudo, o SA apresentou um total de 4 infactibilidades contra apenas 1 dos demais métodos.

Os resultados para o conjunto S3 é apresentado na Tabela 16.

Tabela 20 - Resultados Finais do Conjunto S3

	AG		**BT**		**SA**	
Inst.	**Infact.**	**Desvio**	**Infact.**	**Desvio**	**Infact.**	**Desvio**
I1	0	1,16	0	**-0,66**	0	1,58
I2	0	4,70	0	**1,22**	0	4,08
I3	0	-8,56	0	-9,90	0	**-10,59**
I4	0	2,92	0	**0,93**	0	2,46
I5	0	-1,12	0	**-3,90**	0	-1,49
I6	0	-2,51	0	**-2,88**	0	-2,19
I7	0	-10,07	0	**-11,92**	0	-10,11
I8	0	0,10	0	-3,02	0	**-3,98**
I9	0	15,34	0	**-4,62**	0	6,03
Média		0,22		**-3,86**		-1,58

No conjunto S3, a BT obteve 7 vitórias em 9 possíveis, além disso, ela também foi superior em 7 instâncias em relação aos demais métodos. A BT ainda apresentou a melhor média de desvios, sendo a diferença de 2,28% em relação ao SA que foi o segundo colocado. O SA possui 2 vitórias e possui desempenho superior ao AG em 7 instâncias. O SA também foi superior ao AG em termos de desvio médio, sendo essa diferença de 1,36%.

O resumo do número de vitórias dos métodos é demonstrado no gráfico da Figura 21.

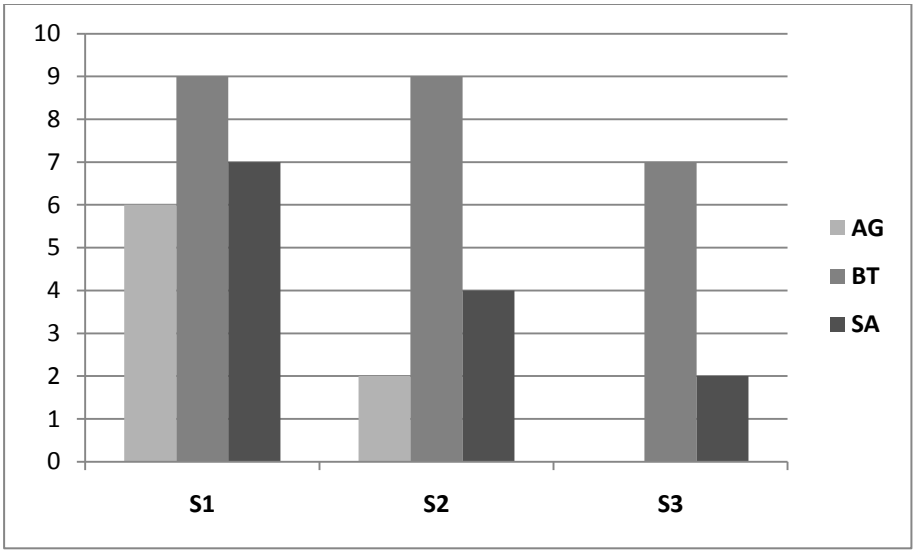

Figura 21 - Número de Vitórias para S1, S2 e S3

Em linhas gerais, a BT e o SA obtiveram maior número de vitórias em todos os conjuntos de instâncias, sendo o número de vitórias da BT superior ao do SA. Além disso, é visível que à medida que a complexidade das instâncias aumenta, o AG obtêm cada vez menos vitórias.

A Figura 22 mostra o número de vitórias desconsiderando as instâncias em que os melhores resultados foram iguais para dois ou mais métodos.

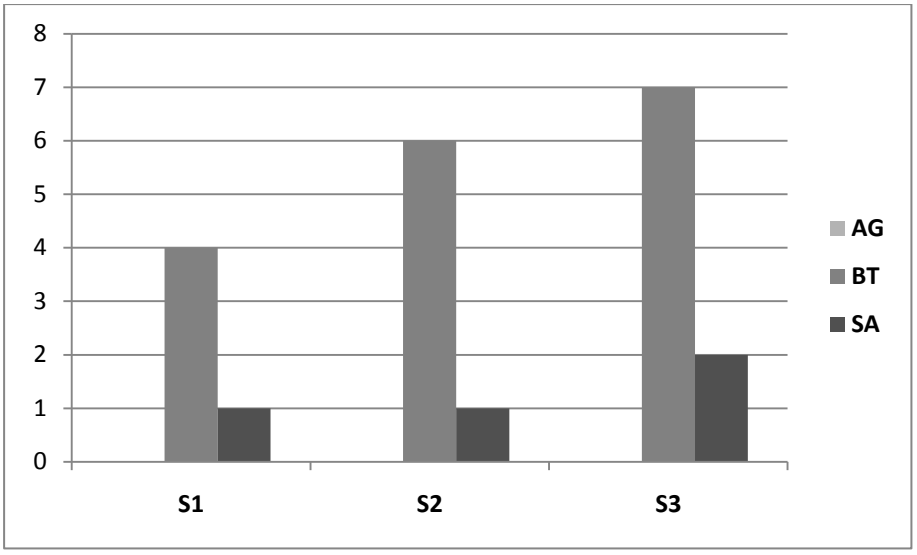

Figura 22 - Quantidade de Melhores Resultados para S1, S2 e S3

Em linhas gerais, o AG não foi superior em nenhuma instância dos conjuntos S1, S2 e S3. A BT obteve os melhores resultados em 17 instâncias e o SA em 4 instâncias. As demais instâncias obtiveram resultados iguais para um ou mais métodos. Além disso, a BT e o SA destacam-se a medida que os problemas tornam-se mais difíceis e mostram serem superiores ao AG.

4.4.2 Resultados Para o Conjunto Industrial S4

Os resultados para o conjunto industrial S4 são apresentados na tabela 17 abaixo:

Tabela 21 - Resultados Finais do Conjunto Industrial S4

	AG		**BT**		**SA**	
Inst.	**Infact.**	**Desvio**	**Infact.**	**Desvio**	**Infact.**	**Desvio**
A1	0	-2,02	0	**-2,67**	0	-2,22
A2	0	-4,25	0	**-4,94**	0	-4,03
A3	1	1,00	1	**-1,43**	1	-1,07

Tabela 22 - Resultados Finais do Conjunto Industrial S4

B1	0	-2,33	0	-4,89	0	**-5,66**
B2	0	-2,84	0	-4,27	0	**-5,53**
B3	0	-2,08	0	**-4,11**	1	-1,61
Média		-2,09		**-3,72**		-3,35

O AG apresenta a pior média geral de desvios. A BT obteve melhores desvios em 4 das 6 instâncias e apresenta a melhor média geral. Além disso, a BT é superior ao AG nas outras 2 instâncias, B1 e B2. O SA foi superior aos demais métodos em apenas 2 instâncias, porém obteve melhores desvios em 4 instâncias em relação ao AG. O número de infactibilidades da BT foi de 1, igual ao do AG. O SA obteve 2 infactibilidades no total. Em linhas gerais, o mesmo ocorre com as instâncias do conjunto S4 em relação aos demais conjuntos, onde à medida que a complexidade das instâncias aumenta, com exceção do SA em B3, a distância dos desvios entre a abordagem genética e a abordagem de busca local fica visível, sendo as buscas locais superiores. Em outras palavras, mesmo em instâncias industriais, que possuem alta complexidade, a busca tabu e o *simulated annealing* foram superiores às estimativas da fábrica e ao algoritmo genético.

5 CONCLUSÕES

O presente trabalho apresentou duas abordagens de buscas locais, busca tabu e *simulated annealing*, aplicadas ao Problema Integrado de Dimensionamento de Lotes e Programação da Produção (PIDLPP). Os métodos são comparados utilizando quatro conjuntos de instâncias, sendo três conjuntos constituídos de instâncias artificiais, que variam de baixo a médio nível de complexidade, e uma conjunto baseado na programação real de uma fábrica de refrigerantes com alto nível de complexidade. Para o conjunto de instâncias artificiais, a comparação é feita em relação ao método *Branch and Cut* implementado utilizando o pacote computacional GAMS/Cplex. Para o conjunto de instâncias industriais, a comparação é feita com as estimativas de custo da fábrica. Além disso, o algoritmo genético de Toledo *et al.* (2008a) também é utilizado como meio de comparação em todos os conjuntos de instâncias.

O presente trabalho também fez um estudo de geração de vizinhança para a codificação proposta em Toledo (2005) que foi utilizada como representação das soluções nos métodos propostos.

Os testes preliminares mostraram que o uso de movimentos exaustivos consome muito tempo computacional e não retorna bons resultados quando a complexidade das instâncias do PIDLPP aumenta. Já o uso de movimentos aleatórios se mostrara bastante eficientes retornando bons resultados. A combinação entre eles também não é uma boa estratégia. Isso pode ser explicado pela complexidade da representação da solução. Movimentos exaustivos precisam executar um número elevado de possibilidades para o tipo de codificação utilizada. Isso torna movimentos aleatórios mais eficientes já que tendem a encontrar melhores vizinhos mais rapidamente.

Os testes preliminares mostraram ainda que escolher aleatoriamente um movimento a ser executado é uma estratégia melhor do que escolher o movimento de maneira determinística. Além disso, escolher um movimento e executar por várias iterações, ou seja, explorar por várias iterações sua vizinhança, é uma boa estratégia

para a busca tabu. Por outro lado, escolher o movimento e executá-lo por uma única iteração é uma melhor estratégia para o *simulated annealing*.

Em linhas gerais, a busca tabu quando comparado com todos os outros métodos, retorna melhores desvios em todos os conjuntos de instâncias utilizados nos testes desse trabalho. Além disso, a busca tabu apresenta o maior número de vitórias por conjunto. O *simulated annealing* apresenta a segunda melhor média em todos os conjuntos de instâncias, exceto no conjunto S1 onde o AG obtém a segunda melhor média. SA também supera o AG em todas as instâncias em número de vitórias. Entretanto, o *simulated annealing* apresenta três vezes mais infactibilidades do que os demais métodos.

Dessa forma, a busca tabu utilizando somente movimentos aleatórios, com escolha aleatória dos movimentos, executando o movimento escolhido por várias iterações e utilizando seis dos sete movimentos propostos neste trabalho obteve melhores resultados considerando a média dos desvios, número de infactibilidades e número de vitórias. O *simulated annealing* utilizando movimentos aleatórios, escolha aleatória dos movimentos, executando o movimento escolhido por uma única iteração e também utilizando seis dos sete movimentos propostos neste trabalho, obteve melhores resultados em desvio médio e em número de vitórias do que o AG de Toledo *et al.* (2008a). Porém se revela uma abordagem mais instável já que apresenta maior número de infactibilidades. Assim, a busca tabu se mostrou a melhor abordagem proposta pelo presente trabalho e o *simulated annealing* se mostrou melhor do que o AG considerando qualidade de solução e inferior quando se considerada a estabilidade do método.

6 REFERÊNCIAS BIBLIOGRÁFICAS

ABIR, **Dados de Mercado-Refrigerantes**, Associação Brasileira da Indústria de Refrigerantes e Bebidas Não Alcoólicas. Disponível em: http://www.abir.org.br, acessado em: 7 de Setembro de 2009.

BARBAROSOĞLU, G., ÖZDAMAR, L. **Analysis of Solution Space-Dependent Performance of Simulated Annealing: the Case of the Multi-Level Capacitated Lot Sizing Problem**. Computers & Operations Research 27, p. 895-903, 2000.

BERTSIMAS, D., TSITSIKLIS, J. N., **Introduction to Linear Optimization**, Athena Scientific, 1997.

BELVAUX, G., WOLSEY, L.A., **Modelling practical lot-sizing problems as mixed-integer programs**, Management Science 47 (7), 993–1007, 2001.

BITRAN, G. R.; YANASSE, H. H. **Computational Complexity of Capacitated Lot Size Problem.** Management Science, n. 28, p. 1178-1186, 1982.

BOZYEL, M. A.; ÖZDAMAR, L. **The capacited lot sizing problem with overtime decisions and setup times**. IIE Transactions, p. 1043-1057, 2000.

CASTRO, L. N. **Engenharia Imunológica: Desenvolvimento e Aplicação de Ferramentas Computacionais Inspiradas em Sistemas Imunológicos Artificiais.** Dissertação de Mestrado, Campinas, SP, UNICAMP, 2001.

FLEISCHMANN, B.; MEYER, H. **The general lotsizing and scheduling problem.** Department for Production and Logistics, University of Augsburg, Universiffitsstr. 16, Augsburg, Germany, 1997.

GENDREAU, Michel., **An Introduction to Tabu Search**. In: Handbook of Metaheuristics, F. Glover e G.A. Kochenberger, Kluwer, Boston, 55-82, 2003.

GLOVER, F. **Future patchs for Integer Programming and links to Artificial Intelligence**. Computer and Operations Research, 5:553-549, 1986.

GLOVER, F. **Tabu Search, Part I** .ORSA Journal on Computing 1, pp. 190-206, 1989.

GLOVER, F. **Tabu Search, Part II**. ORSA Journal on Computing 2, pp. 4-32, 1990.

GLOVER, F; LAGUNA M. **Tabu Search**. In C.R. Reeves, editor, Modern Heuristic Techniques for Combinatorial Problems, Advanced Topics in Computer Science Series, chapter 3, pages 70-150. Blackwell Scientific Publications, London, 1993.

GLOVER, F.; LAGUNA, M. **Tabu Search**, Kluwer, Norwell, MA, 1997.

GOLDBERG, D. E., **Genetic Algorithms in search, optimization, and machine learning**, Addison Wesley, 1989.

HERTZ, A.; TAILLARD, E,; WERRA, D. **A tutorial on Tabu search. Proceedings of**
Giornate di Lavoro. AIRO-95, 13–24, 1995.

HOLLAND, J.H., **Adaptation in natural and artificial systems**, The University of Michigan Press, 1975.

HUA, Z.; DING, Y.; SHAO, Q., **Immune co-evolutionary algorithm based partition balancing optimization for tobacco distribution system**. Expert Systems with Applications 36, p. 5248–5255, 2009.

JANS, R.; DEGRAEVE, Z. **Meta-heuristics for dynamic lot sizing: A review and comparison of solution approaches**. European Journal of Operational Research 177, p. 1855–1875, 2007.

JIN, F.; SONG. S.; WU, C. **A simulated annealing algorithm for single machine scheduling problems with family setups**. Computers & Operations Research 36, p. 2133-2138, 2009.

KARIMI, B.; GHOMI, S. M. T. F.; WILSON, J. M., **The capacitated lot sizing problem: a review of models and algorithms**. OMEGA, v. 31, p. 365–378, 2003.

KIRKPATRICK, S., GELATT, C. D., VECCHI, M.P., **Optimization by Simulated Annealing**, Science, 220 (4598), 671-680, 1983.

KUIK, R.; SALOMON, M.; VAN WASSENHOVE, L., N. **Batching decisions: Structure and models**. European Journal of Operational Research 75, p. 243–263, 1994.

LAGUNA, M. **Tabu Search Tutorial**. II Escuela de Verano Latino-Americana de Investigación. Operativa, Rio de Janeiro, Brazil, 1995.

LÜ, Z.; HAO, J.; K. **Adaptive Tabu Search for course timetabling**. European Journal of Operational Research 200, p. 235–244, 2010.

MACFARLANE, A.; TUSON, A. **Local search: A guide for the information retrieval practitioner**. Information Processing and Management 45, p. 159–174, 2009.

MEYR, H., **Simultaneous lot sizing and scheduling by combining local search with dual reoptimization**, European Journal of Operational Research 120, 311–326, 2000.

NASCIMENTO, M. C. V.; RESENDE, M. G. C.; TOLEDO, F. M. B. **GRASP heuristic with path-relinking for the multi-plant capacitated lot sizing problem**. European Journal of Operational Research 200, p. 747–754, 2010.

NASCIMENTO, M. C. V. **Uma heurística GRASP para o problema de dimensionamento de lotes com múltiplas plantas**. 2007. 66 f. Dissertação (Mestrado em Ciências da Computação e Matemática Computacional) - Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Paulo.

ÖZDAMAR, L.; GÜLAY, B. **An integrated Lagrangean Relaxation-Simulated Annealing Approach to the Multi-Level Capacited Lot Sizing Problem**. International Journal Production Economics 68, p.319-331, 2000

PASZKOWICZ, W. **Properties of a genetic algorithm equipped with a dynamic penalty function.** Computational Materials Science 45, p. 77-83, 2009.

ROSA, K. A. **Avaliando Algoritmos Genéticos com Diferentes Estruturas Populacionais em um Problema Integrado de Dimensionamento de Lotes e Programação da Produção**. 2009. Monografia (Bacharel em Ciências da Computação) - Departamento de Ciência da Computação, Universidade Federal de Lavras, Minas Gerais.

SAHLINGA, F.; BUSCHKÜHL, L.; TEMPELMEIER, H.; HELBERA, S. **Solving a multi-level capacitated lot sizing problem with multi-period setup carry-over via a fix-and-optimize heuristic**. Computers & Operations Research 36, p. 2546-2553, 2009.

SOUZA, M. J. F. **Inteligência Computacional para Otimização**. (Desenvolvimento de material didático ou instrucional - Material didático), 2005.

TOLEDO, C. F. M. **Problema Conjunto de Dimensionamento de Lotes e Programação da Produção.** Tese de Doutorado, Campinas, SP, UNICAMP, 2005.

TOLEDO, C. F. M.; FRANÇA, P. M.; MORABITO R. E KIMMS, A. **Um modelo de otimização para o problema integrado de dimensionamento de lotes e programação da produção em fábricas de refrigerantes**. Pesquisa Operacional, 27 (1), 155-186, 2007.

TOLEDO, C. F. M.; FRANÇA, P. M.; MORABITO, R.; KIMMS, A. **Multi-population genetic algorithm to solve the synchronized and integrated two-level lot sizing and scheduling problem**. International Journal of Production Research, p. 1-23, 2008a.

TOLEDO, C. F. M.; FRANÇA, P. M.; FERREIRA, J. E. **Meta-Heuristic Approaches for a Soft Drink Industry Problem**. In: 13th IEEE International Conference on Emerging Technologies and Factory Automation, Hamburg, Alemanha, p. 1384-1391, 2008b.

TOLEDO, C. F. M., FERREIRA, J. E., SIMEONE, F., ROSA, G. P. **Metaheurísticas Aplicadas ao Problema Geral de Dimensionamento de Lotes e Programação da Produção**. In: XLI Simpósio Brasileiro de Pesquisa Operacional, Bahia, Brasil, p. 3277-3288, 2009.

VENDITTI, L.; PACCIARELLI, D.; MELONI, C. **A tabu search algorithm for scheduling pharmaceutical packaging operations**. European Journal of Operational Research 202, p. 538–546, 2010.

WANG, X.; TANG, L. **A hybrid metaheuristic for the prize-collecting single machine scheduling problem with sequence-dependent setup times.** Computers & Operations Research 37, p.1624–1640, 2010 .

WARNER, H. M.; WHITIN, T. M. **Dynamic version of the economic lot size model**. Management Science 5, 89-96. 1958.

ZHANG, R.; WU, C. **A hybrid immune simulated annealing algorithm for the job shop scheduling problem.** Applied Soft Computing 10, p. 79–89, 2010.

ANEXOS

ANEXO A - Variáveis da Função Objetivo de Toledo *et al.* (2007)

Função objetivo:

$$\sum_{i=1}^{J}\sum_{j=1}^{J}\sum_{l=1}^{L}\sum_{s=1}^{T\cdot S} s_{ijl} z_{ijls} + \sum_{j=1}^{J}\sum_{t=1}^{T} h_j I_{jt} + \sum_{j=1}^{J}\sum_{l=1}^{L}\sum_{s=1}^{T\cdot S} v_{jl} q_{jls} +$$
$$+ \sum_{i=1}^{\bar{J}}\sum_{j=1}^{\bar{J}}\sum_{k=1}^{\bar{L}}\sum_{s=1}^{T\cdot \bar{S}} \bar{s}_{ijk} \bar{z}_{ijks} + \sum_{j=1}^{\bar{J}}\sum_{k=1}^{\bar{L}}\sum_{t=1}^{\bar{T}} \bar{h}_j \bar{I}_{jk,t\cdot T^m} + \sum_{j=1}^{\bar{J}}\sum_{k=1}^{\bar{L}}\sum_{s=1}^{T\cdot \bar{S}} \bar{v}_{jk} \bar{q}_{jks} + \quad (2)$$
$$+ M \sum_{j=1}^{J} q_j^0$$

Onde:

Variável	Significado
T	Número de macro-períodos.
S	Número de micro-períodos nas linhas.
$\bar{S}$	Número de micro-períodos nos tanques.
J	Número de produtos.
$\bar{J}$	Número de xaropes.
L	Número de linhas.
$\bar{L}$	Número de tanques.
s_{ijl}	Custo de troca do produto *i* para o produto *j* na linha *l*.
$\bar{s}_{ijk}$	Custo de troca do xarope *i* para o xarope *j* no tanque *k*.
z_{ijls}	Igual a 1 se houve troca do produto *i* para o produto *j* na linha *l* no micro-período *s*. Igual a 0 em outros casos.
$\bar{z}_{ijks}$	Igual a 1 se houve troca do xarope *i* para o xarope *j* no tanque *l* no micro-período *s*. Igual a 0 em outros casos.

h_j	Custo de estoque do produto *j*.
$\bar{h}_j$	Custo de estoque do xarope *j*.
I_{jt}	Quantidade estocada do produto *j* no macro-período *t*.
$\bar{I}_{jkt}$	Custo do xarope *j* no tanque *k* no final do macro-período *t*.
v_{jl}	Custo de produção do produto *i* na linha *l*.
$\bar{v}_{jk}$	Custo de produção do xarope *j* no tanque *k*.
q_{jls}	Quantidade produzida do produto *j* na linha *l* e micro-período *s*.
$\bar{q}_{jks}$	Quantidade do xarope *j* armazenada no tanque *k* no micro-período *s*.
M	Número real positivo muito grande.
q_j^0	Quantidade da demanda do produto *j* não atendida.
T^m	Número de micro-períodos em um macro-período.