LUCAS DE OLIVEIRA

ALGORITMOS GENÉTICOS ASSOCIADO À PROGRAMAÇÃO MATEMÁTICA APLICADO AO PROBLEMA DE FABRICAÇÃO DE REFRIGERANTES

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.

Orientador:

Prof. Claudio Fabiano Motta Toledo

LAVRAS
MINAS GERAIS – BRASIL
2009

LUCAS DE OLIVEIRA

ALGORITMOS GENÉTICOS ASSOCIADO À PROGRAMAÇÃO MATEMÁTICA APLICADO AO PROBLEMA DE FABRICAÇÃO DE REFRIGERANTES

Aprovada em ____ de _____ de _____

_____

Prof. Ricardo Silveira Sousa

_____

Prof. Fortunato Silva de Menezes

_____

Prof. Claudio Fabiano Motta Toledo

(Orientador)

LAVRAS
MINAS GERAIS – BRASIL
2009

Dedico esse trabalho aos meus pais, Maria e Wilson, minhas irmãs, Lucimar e Vanderlúcia, e meus cunhados, Áriston e José Aloísio por sempre estarem presentes em todos os momentos da minha vida, e por terem me dado a chance de realizar esse sonho. Amo todos vocês!

AGRADECIMENTOS

A minha família por todo apoio e amor incondicional. Por serem tão solidários e afetuosos, por sempre acreditarem em mim e estarem presentes em cada momento dessa conquista.

Ao meu amigo, professor e orientador Claudio, por sempre estar presente em todos os momentos desta jornada me apoiando.

A todos meus colegas de república que me proporcionaram bons momentos que sempre terei como lembranças.

Agradeço a minha turma da computação, meus amigos, por tudo.

Aos professores e funcionários do Departamento de Ciência da Computação da UFLA.

Enfim, agradeço de coração a cada um que participou desta conquista.

ALGORITMOS GENÉTICOS ASSOCIADO À PROGRAMAÇÃO MATEMÁTICA APLICADO AO PROBLEMA DE FABRICAÇÃO DE REFRIGERANTES

RESUMO

O presente trabalho propõe um algoritmo genético associado à programação matemática. O método é aplicado à resolução de um modelo de otimização inteiro misto para um problema de programação da produção em uma fábrica de refrigerantes. O algoritmo genético determina o seqüenciamento dos lotes para que um modelo simplificado seja utilizado em seguida na determinação do dimensionamento dos lotes. O método proposto é avaliado em um conjunto de instâncias baseadas em dados reais fornecidos por uma indústria de bebidas. Os resultados obtidos são comparados àqueles obtidos por outro método encontrado na literatura e demonstram que a abordagem híbrida proposta superou em muito a abordagem da literatura.

Palavras-chave: Algoritmo genético; programação matemática; dimensionamento de lotes; programação da produção.

GENETIC ALGORITHMS EMBEBED WITH MATHEMATICAL PROGRAMMING APPLIED TO SOLVE A SOFT DRINK PRODUCTION PROBLEM

ABSTRACT

The present report proposes a genetic algorithm embedded with mathematical programming. The method is applied to solve a mixed-integer optimization model for a lot sizing and scheduling problem in a soft drink industry. The genetic algorithm returns the sequencing for lots in such way that a simplified model can be used to define lot sizing. The proposed method is evaluated in a set of instances based on data provided by a soft drink company. The results found are compared with those achieved by other method in the literature showing that hybrid approach outperforms the literature approach.

Keywords: *Genetic algorithm; mathematical programming; lot sizing; scheduling.*

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

1 INTRODUÇÃO

1.1 Contextualização

A produção de refrigerantes no Brasil tem aumentado a cada ano. O volume produzido em 2008 ultrapassou 14 bilhões de litros, representando um aumento de 4,04% em relação ao ano de 2007. Isso levou o setor a um faturamento 9,05% maior em 2008. O Brasil é o terceiro maior mercado de bebidas do mundo, perdendo apenas para EUA e México (ABIR, 2009). Estes dados mostram a importância do setor e a necessidade de novos métodos de resolução integrados a ferramentas computacionais que apóiem a tomada de decisão. Porém, o sistema de planejamento da produção ainda não é automatizado em muitas fábricas de bebidas. Isso motivou o trabalho apresentado nesta monografia, onde um método de resolução para o problema de planejamento da produção em fábricas de refrigerantes será apresentado.

O processo de fabricação de refrigerantes envolve o uso de diferentes matérias primas que misturadas a outros componentes resultam nos xaropes. Os xaropes ficam armazenados em tanques de onde escoam para as linhas de produção. A combinação do tipo de xarope que escoa dos tanques com o tipo de envase realizado nas linhas resulta na bebida ou produto final. Desta forma, a planta industrial apresenta dois níveis de produção: tanques e linhas de engarrafamento. Nos tanques, o responsável pela produção precisa realizar o dimensionamento e a programação (seqüenciamento) dos xaropes, decidindo quanto e quando determinado xarope será armazenado em cada um dos tanques disponíveis. Nas linhas de engarrafamento, o responsável pela produção estabelece o dimensionamento e a programação das bebidas ou produtos finais, decidindo quanto e quando determinado produto será colocado em produção nas linhas existentes.

Diversas restrições e decisões estão presentes em cada nível de produção considerado. As decisões relativas aos tanques e linhas envolvem a programação e o dimensionamento dos xaropes e produtos de forma a atender as demandas e minimizar os custos de produção. Considera-se que as demandas por bebidas são planejadas em um horizonte de tempo mensal e devem ser atendidas ao final de cada semana de trabalho. As decisões e restrições presentes no planejamento da produção são interdependentes, pois as quantidades de xaropes armazenadas nos tanques devem ser suficientes para atender a produção nas linhas. Logo, a programação dos xaropes nos tanques precisa estar sincronizada à programação das linhas já que não pode ocorrer o envase de determinada bebida, se o xarope correspondente não estiver armazenado em um tanque. Esse tipo de problema foi chamado de Problema Integrado de Dimensionamento de Lotes e Programação da Produção (PIDLPP) por Toledo *et al.* (2007).

Um método híbrido será proposto nesta monografia para solucionar o PIDLPP. Métodos híbridos são abordagens que integram o uso de métodos heurísticos com métodos exatos ou outros métodos heurísticos. Nos últimos anos as pesquisas envolvendo métodos híbridos têm ganhado foco. Um levantamento feito por Jourdan *et al.* (2009) mostra que a quantidade de publicações que envolvem esse tipo de abordagem vem em crescente aumento. Isso se deve ao fato de que esses métodos muitas vezes são capazes de explorar as vantagens de cada uma das abordagens integradas. Nas próximas seções são descritos com mais detalhes os objetivos, a metodologia e como o trabalho foi organizado.

1.2 Objetivos do Trabalho

O principal objetivo do presente trabalho é o desenvolvimento de um algoritmo genético combinado com métodos exatos na resolução dos modelos matemáticos propostos para o problema de fabricação de refrigerantes. Os

modelos matemáticos deste tipo de problema costumam ser classificados como inteiro misto, devido à presença de variáveis continuas e inteiras nas restrições dos modelos. O método proposto utiliza um algoritmo genético com população hierarquicamente estruturada para fixar as variáveis inteiras, atendendo as restrições de modelos matemáticos propostos para o PIDLPP. Assim, um método exato solucionará o problema de programação linear (PL) relacionado (modelo relaxado). A meta é conseguir melhorar a qualidade das soluções obtidas pela meta-heurística, deixando para o método exato a determinação das variáveis contínuas do modelo matemático.

Objetivos específicos deste trabalho:

- Estudo do problema de fabricação de refrigerantes e avaliação de instâncias existentes na literatura.
- Estudo e avaliação de abordagens utilizando meta-heurísticas e programação matemática.
- Proposição de um método baseado em meta-heurística e programação matemática para o problema de fabricação de refrigerantes.
- Desenvolvimento e teste do método em instâncias do PIDLPP existentes na literatura.
- Avaliação dos resultados e elaboração de artigos científicos.

1.3 Metodologia

O trabalho foi realizado no Departamento de Ciência da Computação no período de janeiro a novembro de 2009. Ele é classificado como uma pesquisa básica quanto a sua natureza, pois o foco é propor e avaliar uma nova abordagem, deixando a sua efetiva aplicação como trabalhos futuros. Como a pesquisa objetiva avaliar uma nova abordagem para o problema, ela é caracterizada como exploratória. Ao propor e aperfeiçoar um método para

encontrar boas soluções para o problema, a pesquisa também passa a ser caracterizada como operacional quanto aos procedimentos.

1.4 Estrutura do Trabalho

O primeiro capítulo contextualiza o trabalho introduzindo o problema estudado, os objetivos a serem alcançados e a metodologia de pesquisa adotada. O capítulo 2 apresenta uma pequena revisão dos trabalhos existentes na literatura que se relacionam com o problema estudado e com o método a ser desenvolvido. O capítulo 3 estabelece a nova abordagem proposta por esta monografia para solucionar o PIDLPP. O capítulo 4 descreve os testes computacionais realizados e apresenta os resultados obtidos ao aplicar o método proposta em instâncias do PIDLPP existentes na literatura. Por fim, as conclusões e propostas de trabalho futuro são sumarizadas no capítulo 5.

2 REFERENCIAL TEÓRICO

2.1 Produção de Bebidas

2.1.1 Panorama

Em 1772 foi dado o primeiro passo para o surgimento de uma das bebidas mais populares atualmente, o refrigerante. Nesta época o britânico Joseph Priestley, doutor em química, obteve um método capaz de misturar água com dióxido de carbono, resultando em um composto chamado de água carbonatada. A partir dos estudos de Priestley, em 1783, Jacob Schweppe cria a água mineral. Schweppe foi a primeira pessoa a explorar comercialmente e construir uma fortuna através de produção de água carbonatada. Quase cem anos mais tarde, em meados de 1871 nos Estados Unidos, surge a primeira indústria com marca registrada, denominada Lemon's Superior Sparkling Ginger Aleã Ainda em meados deste período, em 1886 nos Estados Unidos, o farmacêutico americano John Stith Pemberton cria a receita de uma das bebidas mais famosas do mundo, a *Coca-Cola*. E poucos anos depois, surge uma nova bebida que vem a se tornar a sua maior concorrente, a *Pepsi Cola*, criada pelo farmacêutico Caled Bradham (ABIR, 2009).

Até meados de 1890 a produção de refrigerantes era feita de forma manual. A produção na indústria de refrigerante só veio a ganhar proporções maiores duas décadas depois com a utilização de máquinas no processo produtivo. O grande marco histórico da indústria de produção de refrigerantes aconteceu em 1892 com a invenção do *crown cork*, também conhecido como *crown cap*, uma tampa que permitiu o armazenamento de gás carbônico em garrafas de vidro. Com essa tampa, as bebidas poderiam ser armazenadas e transportadas pela indústria com mais facilidade e os consumidores poderiam comprar e armazenar as bebidas em casa. Após este marco, o próximo passo

significativo para o avanço da produção só veio acontecer em 1991 com o surgimento de garrafas leves e resistentes que utilizam plástico PET (Polietileno Tereftalato).

Atualmente, o consumo de refrigerantes está presente em todo o mundo. Neste cenário o Brasil se encaixa como o terceiro maior produtor mundial, ficando atrás apenas dos Estados Unidos e México. Segundo a Associação Brasileira de Indústrias de Refrigerantes (ABIR, 2009), o Brasil produziu um volume equivalente a mais de 14 bilhões de litros de refrigerantes somente em 2008. Há mais de 800 fábricas que atuam neste setor no Brasil (Lafis, 2003).

Desde os últimos anos a produção de refrigerantes se encontra em constante crescimento. A Figura 1 mostra a evolução do volume total de refrigerantes produzido anualmente no Brasil. O volume produzido em 2008 representa um aumento de 3,57% em relação ao volume produzido em 2007. Isso levou o setor a um faturamento superior a 20,7 bilhões em 2008, representando uma taxa de 9,05% de aumento em relação ao ano anterior.

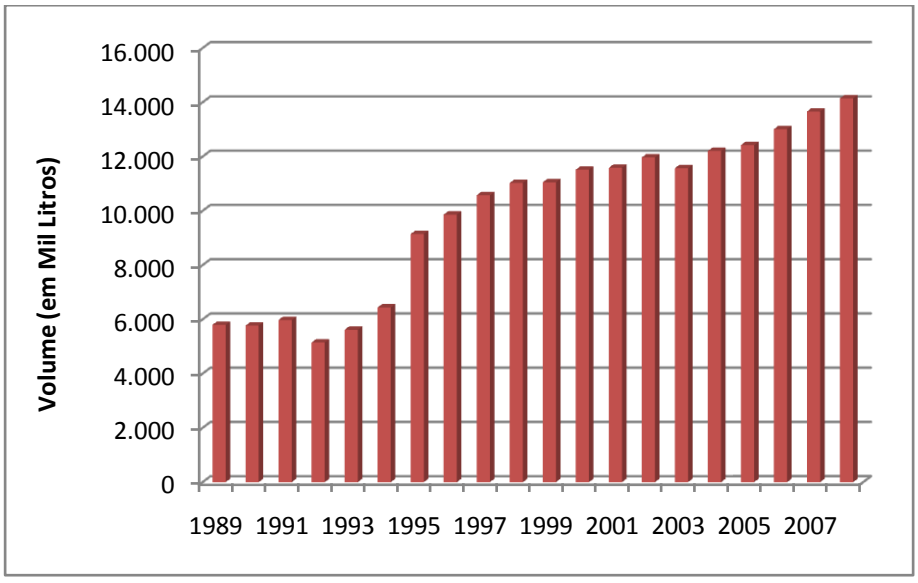

Figura 1 – Volume total de refrigerantes produzido anualmente no Brasil

Estes dados mostram a importância do setor e a necessidade de novos métodos de resolução integrados a ferramentas computacionais que apóiem a tomada de decisão. Porém, o sistema de planejamento da produção deste setor ainda não é automatizado em muitas fábricas.

2.1.2 Processo de Produção de Bebidas

O processo de produção de vários tipos de bebidas, tais como refrigerantes, sucos e chás, é composto pelas etapas de tratamento de água, preparo dos xaropes, envase e empacotamento. A limpeza da água através da filtragem de impurezas consiste no processo de tratamento da água. A etapa de preparo dos xaropes é dividida em duas partes, primeiramente os ingredientes são misturados em máquinas *premix*. Após isso, a mistura é enviada aos tanques, onde será adicionado açúcar ou adoçante, e a mistura será agitada por hélices para se tornar homogênea. Para que a mistura seja agitada de forma correta é necessária uma quantidade mínima no tanque suficiente para cobrir as hélices (Ferreira, 2006).

Uma vez que os xaropes estejam prontos nos tanques, inicia-se a etapa de envase, que consiste no escoamento dos xaropes para as linhas de produção. Um tanque pode abastecer várias linhas de produção ao mesmo tempo, porém uma linha de produção pode receber xarope de apenas um tanque por vez (Figura 2).

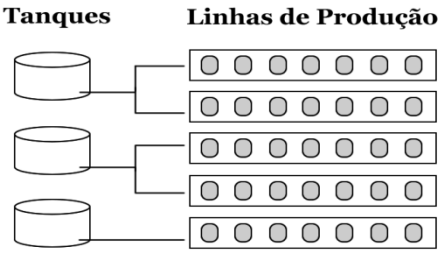

Figura 2 – Distribuição dos tanques por linhas

Nas linhas de produção, os xaropes são misturados com água carbonatada e essa mistura é armazenada em vasilhames. A combinação do tipo de xarope que escoa dos tanques com o tipo de vasilhame utilizado nas linhas resulta em uma bebida ou produto final. Após a etapa de envase, as bebidas são inspecionadas e enviadas para os depósitos.

Desta forma, a planta industrial apresenta dois níveis de produção: tanques e linhas de engarrafamento. Nos tanques, o responsável pela produção precisa realizar o dimensionamento e a programação (seqüenciamento) dos xaropes, onde se deve decidir quanto e quando determinado xarope será armazenado em cada um dos tanques disponíveis. Nas linhas de engarrafamento, o responsável pela produção estabelece o dimensionamento e a programação das bebidas ou produtos finais, decidindo quanto e quando determinado produto será colocado em produção nas linhas existentes.

Segundo Toledo (2005), diversas restrições e decisões estão presentes em cada nível de produção considerado. Todos os tanques envolvidos têm uma capacidade máxima de armazenagem que não pode ser ultrapassada. Os tanques também devem ser abastecidos com uma quantidade mínima de xarope e podem armazenar um único xarope por vez, mas o mesmo xarope pode estar armazenado em diversos tanques. Um tanque é reabastecido apenas quando o xarope nele armazenado for completamente escoado para as linhas de produção. Trocas ou reposições de xarope em um tanque exigem a interrupção da produção nas linhas que o utilizam. Há um custo relacionado à troca de um xarope por outro nos tanques, como também há um gasto de tempo durante esta troca. Isto ocorre porque o armazenamento de um novo xarope no tanque demanda que o xarope seja preparado e o tanque seja higienizado. Esse processo gera um tempo e um custo relacionados à troca de um xarope por outro. Tanto o custo quanto o tempo de troca dependem do xarope previamente armazenado no tanque e do novo xarope que irá ocupar o tanque.

As linhas podem realizar mais de um tipo de envase, como por exemplo, uma mesma linha pode ser ajustada para engarrafar o xarope em um vasilhame de 600*ml*, 1*l* ou 2*l*, dependendo da programação estabelecida. O tempo gasto para produzir uma unidade de uma bebida varia de linha para linha, onde uma mesma linha em geral possui diferentes valores de tempo para diferentes tipos de bebida. De forma semelhante aos tanques, nas linhas também ocorrem custos e tempos entre trocas de bebidas. O tempo e o custo envolvido no preparo de uma linha dependem do tipo de bebida previamente produzida, por exemplo, a seqüência "refrigerante normal - refrigerante diet" resulta em custo e tempo de preparo maior do que a seqüência inversa.

Considera-se que as demandas por bebidas são planejadas em um horizonte de tempo mensal. Ao final de cada semana de trabalho, as demandas devem ser atendidas e todo o excedente produzido deve ser estocado. O armazenamento das bebidas em estoque gera custos que variam de acordo com o tipo de bebida armazenada.

As decisões e as restrições relativas aos tanques e às linhas de engarrafamento são interdependentes, pois as quantidades de xaropes armazenadas nos tanques devem ser suficientes para atender a produção nas linhas. Além disso, a programação dos xaropes nos tanques deve estar sincronizada à programação das linhas, pois não pode ocorrer o envase de determinada bebida, se o xarope correspondente não estiver armazenado em um tanque. A Figura 3 a seguir apresenta uma programação da produção não sincronizada (a) e a mesma programação sincronizada (b).

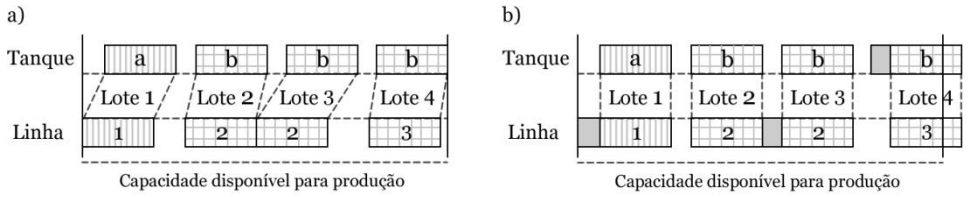

Figura 3 - (a) Programação não sincronizada e (b) Programação sincronizada

Na Figura 3, os retângulos representam o tempo de produção e armazenagem dos lotes de produtos e xaropes, respectivamente. Os espaços entre os lotes representam os tempos de troca. Os tipos de refrigerantes são rotulados pelos números (1, 2 e 3) e os tipos de xaropes pelas letras (a e b). Os retângulos de cor cinza na Figura 3(b) representam os tempos de espera. Observe na Figura 3(a), programação não sincronizada, que a linha de envase iniciou a produção do item 1 usando o xarope a. Isso foi feito sem considerar que o xarope ainda estava sendo preparado no tanque. Na primeira troca (xarope a para b e refrigerante 1 para 2) da Figura 3(a), apesar do tempo de troca ser o mesmo no tanque e na linha, a produção na linha continua adiantada por não ter considerado a espera do preparo do lote anterior. Situações desse tipo devem ser tratadas com uma programação sincronizada entre linhas e tanques, conforme ilustrado na Figura 3(b). Na Figura 3(b), a inserção do tempo de preparação do xarope a é considerada antes do início da produção do item 1.

Quando um lote de bebida necessita de mais de um tanque de xarope, o tempo de preparo deste novo tanque também deve ser levado em conta. Por exemplo, as trocas do refrigerante 2 para 2 e do xarope b para b exigem a inserção de tempo de espera (segundo retângulo de cor cinza da Figura 3(b)). Há também o caso onde o tempo de troca na linha é maior que no tanque, como na troca do refrigerante 2 para 3 que utilizam o mesmo xarope b. Nesta situação o tanque deve esperar o preparo da linha para enviar o xarope b para ser envasado. Observe na Figura 3(b) que, após estabelecer a sincronia, o tempo necessário

para produção aumenta como um todo e o seqüenciamento proposto ultrapassa a capacidade disponível para produção. Isso reforça a necessidade de se considerar a sincronia entre os estágios no momento em que o dimensionamento e o seqüenciamento da produção nas linhas e tanques são estabelecidos.

Considerando todas as restrições apresentadas, o grande objetivo na resolução deste problema é estabelecer uma programação da produção, tanto dos tanques quanto das linhas, de forma que as demandas sejam atendidas e os custos de estoque e troca sejam minimizados.

2.1.3 Modelo Matemático

O problema descrito na seção anterior também é conhecido como Problema Integrado de Dimensionamento de Lotes e Programação da Produção (PIDLPP). Esse problema descreve uma situação bastante geral envolvendo o processo de produção em uma fábrica de refrigerantes. Toledo *et al.* (2007) propuseram um modelo matemático inteiro-misto que introduziu diversas restrições combinadas que até então costumavam ser tratadas separadamente na literatura. A não existência de testes com modelos similares levou os autores a criarem um conjunto de instâncias para avaliar o modelo e as técnicas de solução propostas. Uma primeira abordagem utilizada pelos autores foi a resolução dessas instâncias por meio do pacote computacional GAMS/CPLEX. Todavia, a solução exata se mostrou viável apenas em instâncias de pequena dimensão devido à complexidade do PIDLPP considerado. Toledo *et a.l* (2009) também apresentaram um Algoritmo Genético Multi-Populacional para solucionar o PIDLPP. O algoritmo genético obteve melhores resultados que o GAMS/CPLEX principalmente nas instâncias mais complexas.

A complexidade envolvida na resolução de instâncias do PIDLPP se explica pelo fato deste problema englobar a resolução integrada de dois

Problemas de Programação e Dimensionamento de Lotes Capacitado (PPDLC). O PPDLC aparece como subproblemas do PIDLPP, um nos tanques e outro nas linhas de produção. O PPDLC é um problema de otimização conhecido que Bitran e Yanasse (1982) provaram ser NP-Difícil. O PIDLPP também apresenta custos de troca dependentes da seqüência em cada nível de produção. Logo, há custos de troca dependentes da seqüência em cada PPDLC presente no PIDLPP. Maes *et al.* (1991) demonstraram que determinar uma solução factível para o PPDLC com tempos de troca também é um problema NP-Difícil.

Ferreira *et al.* (2009) também propuseram um modelo matemático para descrever o PIDLPP. Todavia, o modelo proposto foi estabelecido a partir de algumas mudanças nas restrições originais do PIDLPP. O número de linhas foi suposto igual ao número de tanques e cada tanque é dedicado a uma única linha. Isso se justificou pelo fato de que na fábrica um tanque normalmente fica dedicado a uma determinada linha durante a produção. A presente monografia adotará como base o modelo inteiro misto proposto por Ferreira *et al.* (2009) descrito a seguir. Seja {*i*, *j*, *k*, *l*, *m*, *t*, *s*} o conjunto de índices definido como *i*, *j* itens; *t* macro-períodos; *s* micro-períodos; *k*, *l* sabor dos xaropes; *m* máquinas (tanques). Os parâmetros *J*, *M*, *L*, *T* e N representam respectivamente o número de itens, máquinas (tanques), macro-períodos e micro-períodos. Neste modelo, considera-se um horizonte de tempo dividido em *T* macro-períodos, onde cada macro-período está dividido em *N* micro-períodos. Por exemplo, um macro-período pode representar a capacidade em horas disponível em 1 semana de produção. Neste contexto, cada micro-período poderia representar o tempo de produção na semana de determinado produto. Assim, cada micro-período poderá estar associado a um determinado produto. O conjunto S_t é o conjunto de micro-períodos em cada macro-período *t*; ρ_j é o conjunto de máquinas que podem produzir o produto *j*; δ_m é o conjunto de itens que podem ser produzidos

na máquina (linha) *m*; θ_m é o conjunto de xaropes que podem ser armazenados no tanque *m*; ω_{ml} é o conjunto de itens que podem ser produzidos na máquina *m* e utilizam o xarope l; e P_t é o primeiro micro-período do período *t*.

Os custos de estoque e atraso do item *j* são dados pelos parâmetros h_j e g_j respectivamente. A demanda do item *j* no macro-período *t* é dada por d_{jt}. Os índices I e II, são utilizados, respectivamente, nos seguintes parâmetros do modelo para a xaroparia e envase:

s_{kl}^{I} custo de troca do xarope *k* para o xarope *l*;

s_{ij}^{II} custo de troca do produto *i* para *j*;

b_{kl}^{I} tempo de troca do xarope *k* para o xarope *l*;

b_{ij}^{II} tempo de troca do produto *i* para *j*;

a_{mj}^{II} tempo de processamento de uma unidade do item *j* na máquina *m*;

K_{m}^{I} capacidade do tanque *m* em litros;

K_{mt}^{II} capacidade em unidade de tempo da máquina *m* no período *t*;

r_{jl} quantidade do xarope *l* necessária para a produção de uma unidade do produto *j*;

I_{j0}^{+} estoque inicial do produto *j*;

y_{ml0}^{I} = 1, se no tanque *m* o xarope *l* está preparado no início do horizonte de planejamento;

y_{mj0}^{II} = 1, se na máquina *m* o produto *j* está preparado no início do horizonte de planejamento.

As variáveis do modelo são apresentadas a seguir:

I_{jt}^{+} = quantidade em estoque do produto *j* ao final do macro-período *t*;

I_{jt}^{-} = quantidade não produzida de *j* ao final do macro-período *t*;

x_{mjs}^{II} = quantidade do item *j* produzida na máquina *m* no micro-período *s*;

v_{ms}^{II} = tempo de espera da linha *m* pelo preparo de xarope no micro-período *s*;

y_{mls}^{I} = 1, se o tanque *m* armazena o xarope l no micro-período, 0 caso contrário;

y_{mjs}^{II} = 1, se a máquina *m* produz o produto *j* no micro-período *s*, 0 caso contrário;

z_{mkls}^{I} = 1, se há troca no tanque *m* do xarope *k* para *l* no micro-período *s*, 0 caso contrário;

z_{mijs}^{II} = 1, se há troca na máquina *m* do produto *i* para *j* no micro-período *s*; 0 caso contrário.

A partir dos parâmetros e variáveis estabelecidos, o modelo matemático proposto por Ferreira *et al.* (2009) é descrito abaixo:

$$Min\ Z=\sum_{j=1}^{J}\sum_{t=1}^{T}(h_j I_{jt}^{+}+g_j I_{jt}^{-})+\sum_{m=1}^{M}\sum_{s=1}^{N}\sum_{k\in\theta_m}\sum_{l\in\theta_m}s_{kl}^{I}z_{mkls}^{I}+\sum_{m=1}^{M}\sum_{s=1}^{N}\sum_{i\in\delta_m}\sum_{j\in\delta_m}s_{ij}^{II}z_{mijs}^{II} \quad (1)$$

Sujeito a:

$$\sum_{j\in\omega_{ml}} r_{jl}x_{mjs}^{II} \geq \frac{K_m^I}{8}\, y_{mls}^{I}, \qquad m=1..M\ ,\ l\in\theta_m, s=1,...,N; \quad (2)$$

$$\sum_{j\in\omega_{ml}} r_{jl}x_{mjs}^{II} \leq K_m^I y_{mls}^{I}, \qquad m=1...M,\ l\in\theta_m, s=1,...,N; \quad (3)$$

$$\sum_{l\in\theta_m} y^I_{ml(s-1)} \geq \sum_{l\in\theta_m} y^I_{mls}, \qquad m=1,...,M;\ t=1,...,T;\ s\in S_t - P_t \ ; (4)$$

$$z^I_{mkls} \geq y^I_{mk(s-1)} + y^I_{mls} - 1, \qquad m=1,...,M;\ k,l\in\theta_m;\ s=1,...,N; (5)$$

$$z^I_{mkls} \geq \sum_{j\in\omega_{mk}} y^{II}_{mj(s-1)} + y^I_{mls} - 1, \qquad m=1,...,M;\ k,l\in\theta_m;\ t=2,...,T;\ s=P_t; (6)$$

$$\sum_{k\in\theta_m}\sum_{l\in\theta_m} z^I_{mkls} \leq 1, \qquad m=1,...,M;\ t=1,...,T;\ s\in S_t; (7)$$

$$I^+_{j(t-1)} + I^-_{jt} + \sum_{m\in\rho_j}\sum_{s\in S_t} x^{II}_{mjs} = I^+_{jt} + I^-_{j(t-1)} + d_{jt}, \qquad j=1,...,J,\ t=1,...,T; (8)$$

$$\sum_{j\in\delta_m}\sum_{s\in S_t} a^{II}_{mj} x^{II}_{mjs} + \sum_{i\in\delta_m}\sum_{j\in\delta_m}\sum_{s\in S_t} b^{II}_{ij} z^{II}_{mijs} + \sum_{s\in S_t} v^{II}_{ms} \leq K^{II}_{mt}, \quad m=1,...,M;\ t=1,...,T; \quad (9)$$

$$v^{II}_{ms} \geq \sum_{k\in\theta_m}\sum_{l\in\theta_m} b^I_{kl} z^I_{mkls} - \sum_{i\in\delta_m}\sum_{j\in\delta_m} b^{II}_{ij} z^{II}_{mijs}, \qquad m=1,...,M;\ \ s=1,...,N; \quad (10)$$

$$x^{II}_{mjs} \leq \frac{K^{II}_{mt}}{a^{II}_{mj}} y^{II}_{mjs}, \qquad m=1,...,M;\ j\in\delta_m;\ t=1,...,T;\ s\in S_t; (11)$$

$$\sum_{j\in\delta_m} y^{II}_{mjs} = 1 \qquad m=1,...,M;\ \ s=1,...,N; (12)$$

$$z^{II}_{mijs} \geq y^{II}_{mi(s-1)} + y^{II}_{mjs} - 1, \qquad m=1,...,M;\ i,j\in\delta_m;\ s=1,...,N; (13)$$

$$\sum_{i\in\delta_m}\sum_{j\in\delta_m} z^{II}_{mijs} \leq 1, \qquad m=1,...,M\ ;\ s=1,...,N\ ; \quad (14)$$

$$I^+_{jt},\ I^-_{jt} \geq 0, \qquad j=1,...,J,\ t=1,...,T;$$

$$x^{II}_{mjs},\ v^{II}_{ms},\ z^{II}_{mijs},\ z^I_{mkls} \geq 0;\ y^{II}_{mjs},\ y^I_{mls} = 0/1,$$

$$m=1,...,M,\ i \text{ e } j\in\delta_m, k \text{ e } l\in\theta_m,\ t=1,...,T,\ s\in S_t. (15)$$

As restrições (2) a (7) são referentes ao estágio I (xaroparia), e as restrições (8) a (15) modelam o estágio II (envase) do processo de produção de bebidas. Os custos de atraso, estoque e troca são minimizados na função objetivo (1). As restrições (2) e (3) determinam respectivamente as capacidades mínima e máxima dos tanques, caso a atribuição de xarope ao tanque efetivamente ocorra (y^{I}_{mls} =1). A restrição (4) garante que os micro-períodos ociosos ocorreram apenas no fim dos macro-períodos. Para considerar as trocas entre xaropes nos tanques e produtos nas linhas são definidas as restrições (5), (6) e (13). A restrição (6) é necessária para controlar as trocas de xaropes no início de cada macro-período, pois o termo $\sum_{j \in \omega_{mk}} y^{II}_{mj(s-1)}$ é que irá determinar o último xarope preparado no tanque no macro-período *t*-1. As restrições (7) e (14) garantem que há no máximo uma troca de xarope no tanque e produto na linha em cada micro período. A restrição (8) estabelece a equação de balanço de estoque para produtos nas linhas e a restrição (9) define uma restrição de capacidade para as linhas em cada período. A variável v^{II}_{ms} é o tempo de espera da linha pelo preparo de xarope no tanque. Essa restrição garante que a programação de tanques e linhas seja sincronizada. Na Figura 3(b) da seção 2.1.2, os retângulos de cor cinza representam esta variável. Então note que o tempo de espera da linha pelo preparo do xarope deve ser considerado nesta restrição de capacidade. Esse tempo de espera é calculado na restrição (10) pela diferença entre o tempo de preparo do xarope e o tempo de preparo do item na linha. A restrição (11) define a possível atribuição do produto à linha (*setup*) naquele micro-período. A restrição (12) determina que a linha sempre esteja preparada para produzir exatamente um item em cada micro-período. O domínio das variáveis é estabelecido na restrição (15).

2.2 Métodos de Resolução

O modelo matemático apresentado na seção anterior, por fazer uso de variáveis inteiras e contínuas, é considerado como um modelo matemático de programação inteira mista. A resolução deste tipo de modelo é feita geralmente por três tipos de métodos: exatos, heurísticos e híbridos. As próximas seções descrevem cada um destes métodos.

2.2.1 Métodos Exatos

Uma das principais características desta classe de métodos é a garantia de obtenção da solução ótima. Todavia, os métodos exatos costumam ser eficientes apenas na resolução de pequenas instâncias de problemas da classe NP-Difícil. Isso se explica pelo consumo de tempo necessário para resolução de instâncias de médio e grande porte.

Dentro desta classe existem inúmeros métodos, tais como o *Branch and Bound*, *Branch and Cut*, Planos de Corte, Programação Dinâmica, Simplex, Pontos Interiores, etc. A seguir é apresentada uma idéia geral de cada um desses métodos. Segundo French *et al.* (2001), o algoritmo *Branch and Bound* foi um dos mais utilizados em problemas de programação inteira desde que foi descrito por Land e Doig em 1960. Este algoritmo se baseia na estratégia de divisão e conquista. Ele trabalha particionando o espaço de busca de forma a dividir o problema original em subproblemas resolvidos individualmente (Jourdan *et al.*, 2009). Inicialmente, o método relaxa o problema original em um problema de programação linear. Se a resolução do problema relaxado não retornar uma solução inteira, o problema é dividido em dois subproblemas. Esses subproblemas apresentam novas restrições que limitam a valores inteiros uma variável que tenha obtido valor fracionário. Assim, os problemas vão se ramificando, tomando o formato de uma árvore de busca. Quando uma solução

inteira ou infactível é encontrada, o problema não é dividido, parando a ramificação daquele galho da árvore. Ao percorrer toda a árvore, a melhor solução inteira encontrada pelo método é a solução ótima para o problema.

A Figura 4 ilustra uma árvore de subproblemas gerada durante a execução de um algoritmo *Branch and Bound*. Neste exemplo, para resolver um problema P, o algoritmo faz a divisão do problema em dois subproblemas denominados de P1 e P2, e para resolver o problema P1, o algoritmo faz a divisão do mesmo em P3 e P4.

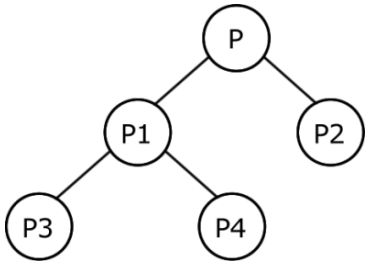

Figura 4 – Árvore de subproblemas gerada pelo método *Branch and Bound*

Outro algoritmo muito utilizado na resolução de problemas de programação inteira é o algoritmo de Planos de Corte. A filosofia deste algoritmo é resolver um problema de programação inteira através da resolução de uma seqüência de problemas de programação linear (Bertsimas e Tsitsiklis, 1997). Ele trabalha relaxando o problema original em um problema de programação linear. O problema relaxado é resolvido e, se a solução ótima retornada por ele for inteira, então ela também é a solução ótima do problema original. Porém, se a solução ótima não for inteira, então é adicionada uma nova restrição ao problema relaxado que satisfaça todas as soluções inteiras do problema e não satisfaça a solução ótima do problema relaxado. Desta forma, restrições vão sendo adicionadas ao problema relaxado até que se encontre uma solução inteira para o problema. Segundo Ferreira (2006), pode-se reduzir o tempo de execução do algoritmo *Branch and Bound* através da redução da

região factível do problema relaxado. Desta forma a aplicação de Planos de Corte nos problemas gerados pelo algoritmo *Branch and Bound* dá origem a um novo método denominado *Branch and Cut*. Uma introdução ao algoritmo *Branch and Cut* pode ser encontrada em Lucena e Beasley (1996). Diversos estudos aplicados ao problema do caixeiro viajante comprovam a eficiência do *Branch and Cut*. Padberg e Rinaldi (1991) utilizaram um algoritmo *Branch and Cut* para obter soluções ótimas de instâncias do caixeiro viajante com 2392 cidades. Applegate *et al.* (2006) conseguiu ir além, desenvolvendo um algoritmo *Branch and Cut* que obteve soluções ótimas para instâncias com 7397 cidades.

O método de Programação Dinâmica, de forma similar ao *Branch and Bound*, também faz uso da estratégia de divisão e conquista. Todavia, duas condições devem ser satisfeitas para se aplicar o método. O problema deve apresentar subestrutura ótima, ou seja, a solução ótima pode ser construída a partir das soluções ótimas de subproblemas do problema original. O problema deve apresentar superposição de subproblemas, ou seja, durante a execução do método, um mesmo subproblema é resolvido mais de uma vez. Desta forma, o algoritmo de Programação Dinâmica trabalha armazenando as soluções encontradas para os subproblemas de forma a evitar a resolução repetida dos mesmos. Para ilustrar o seu funcionamento, vamos examinar o exemplo dado pela Figura 5. Suponha que a árvore é percorrida em profundidade no sentido da esquerda para a direita. Assim a resolução repetida dos subproblemas P3 e P4, nós cinza da árvore, é evitada, pois eles já haviam sido resolvidos anteriormente e as soluções obtidas já estavam armazenadas. Assim, o método de Programação Dinâmica permite a redução do espaço percorrido na busca pela solução ótima e, conseqüentemente, reduz o tempo computacional para encontrá-la.

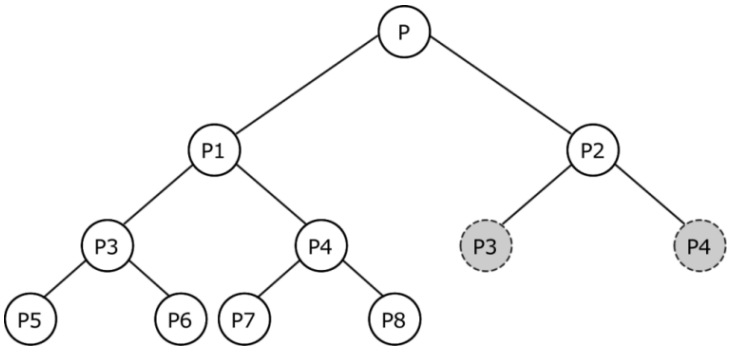

Figura 5 - Árvore de subproblemas gerada pelo método de Programação Dinâmica

Todos os algoritmos apresentados anteriormente são voltados para a resolução de problemas de programação inteira ou programação inteira mista. Todavia, vários destes métodos necessitam da resolução de problemas de programação linear. Dentre os métodos mais conhecidos que resolvem problemas de programação linear, temos o método Simplex e o método de Pontos Interiores.

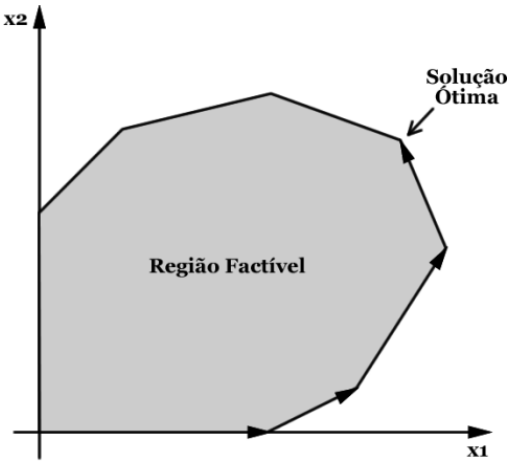

Figura 6 – Execução do método Simplex

O método Simplex trabalha saltando de vértice em vértice da região factível do problema, caminhando sempre na direção que melhora a solução atual (Figura 6). Quando não existe um vértice melhor que o atual, o método termina a execução, pois esta é a solução ótima.

Diferentemente do método Simplex, o método de Pontos Interiores, como o próprio nome sugere, busca a solução ótima explorando pontos no interior da região factível (Figura 7). Segundo Bertsimas e Tsitsiklis (1997), este método representa um grande avanço na teoria da programação linear, superando muitas vezes o método Simplex na resolução de problemas complexos e esparsos.

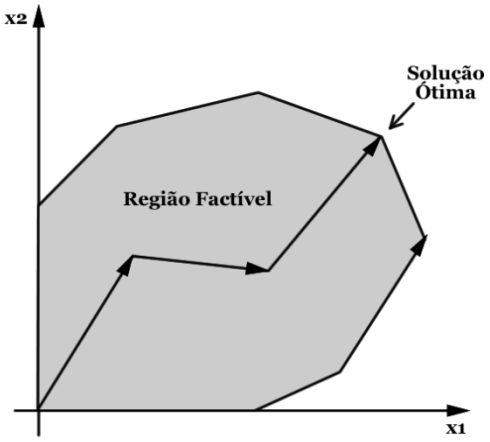

Figura 7 – Execução do método de Pontos Interiores

2.2.2 Métodos Heurísticos

A palavra heurística é originária do grego e significa descobrir ou achar (Reeves, 1993). Métodos heurísticos em otimização objetivam encontrar uma boa solução para um problema dentro de um tempo viável. Esses métodos não garantem a otimalidade da solução encontrada, mas costumam ser capazes de encontrar boas soluções em intervalos de tempo muito menores, quando

comparado a métodos exatos. Desta forma, a utilização de métodos heurísticos se torna bastante interessante quando aplicados à resolução de problemas com elevada complexidade.

Nos últimos anos, uma classe de algoritmos heurísticos denominados meta-heurísticas tem ganhado um destaque especial. Isso se deve ao fato de que vários estudos realizados comprovaram a eficiência destes algoritmos em achar boas soluções. O termo meta-heurística foi introduzido por Fred Glover em 1986 (Glover, 1986). Segundo Osman e Kelly (1996), meta-heurística é um processo de geração iterativo que guia uma heurística secundária, combinando de forma inteligente diferentes conceitos de exploração global e local do espaço de busca. Esses métodos utilizam estratégias de aprendizagem para estruturar as informações, visando encontrar de forma eficiente soluções próximas à solução ótima. Alguns dos principais algoritmos que compõem essa classe são: Busca Tabu, Algoritmos Genéticos, Colônia de Formigas, Busca Local Guiada, GRASP, *Simulated Annealing*, *Scatter Search*, entre outros. Uma revisão sobre estes algoritmos pode ser encontrada em Blum e Roli (2003).

A presente monografia utiliza a meta-heurística Algoritmos Genéticos na resolução do PIDLPP. Segundo Linden (2006), Algoritmos Genéticos (AGs) são técnicas heurísticas inspiradas em processos biológicos de evolução natural e que fazem parte da classe dos algoritmos evolucionários. O primeiro trabalho conhecido envolvendo AGs foi apresentado por John Holland no ano de 1975 (Holland, 1975). Todavia, esse trabalho só veio a ser popularizado mais tarde por um de seus alunos, David Goldberg (1989). Segundo Vose (1998), até hoje a maior parte dos algoritmos genéticos criados são variações do algoritmo descrito por Holland, conhecido como algoritmo genético canônico. Uma introdução aos AGs e suas aplicações pode ser encontrada em Houpt e Houpt (1998).

De acordo com Gen e Cheng (1997), o grande diferencial dos AGs, se comparados a outras técnicas, é a utilização de um conjunto inicial de soluções,

gerado de forma aleatória, chamado de população. Na população são feitas todas as simulações dos processos naturais da evolução de uma espécie, aplicando-se métodos de seleção, reprodução e mutação em seus indivíduos. A Figura 8 apresenta a idéia geral de um algoritmo genético.

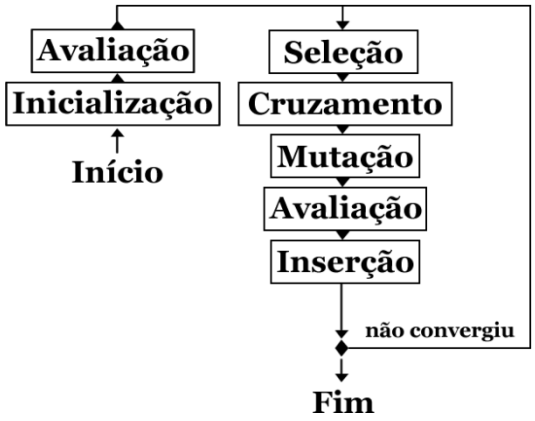

Figura 8 – Algoritmo Genético básico

O primeiro passo na execução de um algoritmo genético é a inicialização da população. Normalmente as soluções iniciais são geradas aleatoriamente. Após ser gerada, a população é avaliada, onde cada indivíduo recebe um valor de aptidão que mede a qualidade da solução que ele representa. A partir desse ponto inicia-se o processo de evolução, onde os operadores genéticos são executados. O operador de seleção é o responsável por determinar quais indivíduos serão escolhidos para o cruzamento. Assim, o operador de cruzamento recebe esses indivíduos escolhidos, chamados de pais, e combina informações entre eles gerando novos indivíduos, os filhos. Para manter a variabilidade entre os indivíduos, é aplicado o operador de mutação. Ele percorre os novos indivíduos gerados decidindo se vai haver mutação e, caso ocorra, pequenas modificações são efetuadas no indivíduo. Após isso, os indivíduos são avaliados e inseridos na população de acordo com algum critério.

Esse critério geralmente é a substituição dos indivíduos menos aptos na população pelos indivíduos mais aptos. Todo esse processo de evolução é repetido até que a população atinja um critério de convergência, terminando a execução do algoritmo.

Desde o seu surgimento, os algoritmos genéticos vêm sendo aplicados com sucesso em diferentes problemas nas mais diversas áreas. Nos trabalhos de Chelouah e Siarry (2003) e Kaelo e Ali (2007) são propostas abordagens usando algoritmos genéticos voltados para o problema de otimização global. Purnaprajna *et al.* (2007) aplica algoritmo genético na otimização do problema de particionamento hardware-software e alocação ótima de recursos. Um algoritmo genético foi utilizado por França *et al.* (2001) para solucionar o *Total Tardiness Single Machine Scheduling Problem*. Em Toledo *et al.* (2009), um Algoritmo Genético Multi-Populacional soluciona instâncias de problemas de programação da produção em fábricas de refrigerantes. Outros trabalhos utilizando AGs podem ser encontrados em Reeves (2003).

2.2.3 Métodos Híbridos

A classe dos métodos híbridos, também conhecidos como métodos cooperativos, engloba algoritmos que fazem uso em conjunto de métodos exatos com métodos heurísticos. Em Talbi (2002) é apresentada uma taxonomia que permite classificar a cooperação entre métodos exatos e meta-heurísticas. Em 2009, Jourdan *et al.* (2009) publicaram um levantamento e classificação de várias abordagens encontradas na literatura que fazem a hibridização de métodos exatos com meta-heurísticas. Segundo os dados apresentados nesse trabalho, o número de publicações que envolvem esse tipo de abordagem tem aumentado durante os últimos anos (Figura 9). Isso se deve ao fato de que as técnicas hibridizadas exploraram as vantagens de cada um dos métodos.

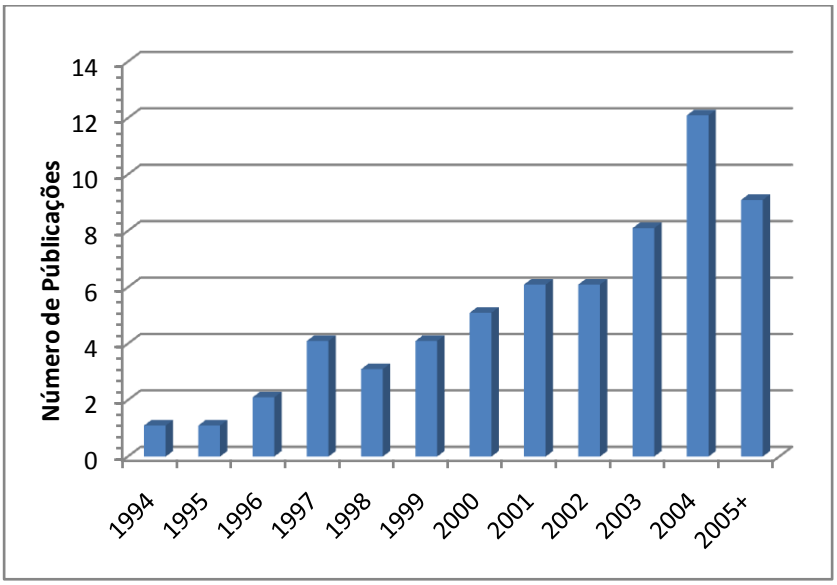

Figura 9 – Evolução da quantidade de publicações de trabalhos que faz hibridização de métodos exatos e meta-heurísticas

A presente monografia propõe uma abordagem que consiste na execução integrada de um AG e um método exato. O AG determina as variáveis inteiras para o modelo matemático proposto por Ferreira *et al.* (2009) para o PIDLPP. Uma vez fixada as variáveis inteiras, torna-se possível estabelecer um modelo que considera apenas as variáveis contínuas. O modelo reduzido é então solucionado por um método exato de programação linear. Considerando a taxonomia proposta em Talbi (2002), essa abordagem pode ser classificada como *LRH* (*Low-level Relay Hybrid*) quanto à forma de cooperação entre os métodos. A classe *LRH* corresponde aos algoritmos que possuem um método incorporado dentro de outro, onde o método incorporado é executado seqüencialmente já que o método global depende dos resultados obtidos para continuar sua execução (Jourdan *et al.*, 2009). No contexto desta classificação, o AG proposto trabalha como método global e o método exato, que resolve o

modelo relaxado, trabalha como método incorporado. Ainda considerando essa taxonomia, a cooperação do espaço de busca é classificada como parcial, pois o AG se concentra na parte inteira deixando para o método exato a parte contínua do espaço de busca.

Atualmente na literatura podem ser encontrados diversos trabalhos que tem aplicado com sucesso abordagens híbridas para problemas de dimensionamento de lotes e programação da produção. Meyr (2000) soluciona um problema de programação e dimensionamento de lotes em máquinas paralelas utilizando uma heurística de busca local para fixar as variáveis binárias do modelo matemático. Assim, um método exato executa a otimização que determina as demais variáveis contínuas do modelo. Defersha e Chen (2008) solucionaram um modelo não-linear com variáveis inteiras utilizando um algoritmo genético associado à programação linear. O algoritmo genético realiza a busca no espaço de soluções que determina diversas variáveis inteiras do modelo. Isso permite uma remodelagem que leva a resolução exata do modelo linear associado, fornecendo a solução final para o problema. Demais trabalhos relacionados podem ser encontrados em Jourdan *et al.* (2009).

3 MÉTODO DE RESOLUÇÃO

Um método híbrido que consiste na integração de um algoritmo genético com um método exato é proposto nesta monografia. A idéia principal desta abordagem é utilizar um algoritmo genético para otimizar o seqüenciamento de lotes, que corresponde a parte discreta do modelo matemático do problema, deixando para um método exato otimizar o dimensionamento dos lotes, que corresponde a parte contínua do modelo matemático do problema. Desta forma, o AG trabalha determinando seqüências de lotes. Para cada seqüência estabelecida pelo AG, é executado um método exato que resolve um modelo matemático, determinando o melhor ajuste no dimensionamento dos lotes para cada seqüência.

Nas próximas seções, é apresentado o modelo matemático utilizado para determinar o dimensionamento de lotes e o algoritmo genético híbrido proposto.

3.1 Modelo Matemático Simplificado para o Dimensionamento de Lotes

Na seção 2.1.3 foi descrito o modelo matemático proposto por Ferreira *et al*. (2009) que formula o problema da produção de bebidas considerando o seqüenciamento e dimensionamento de lotes nos tanques e linhas. Porém, uma vez que a otimização do seqüenciamento de lotes será feita pelo algoritmo genético, este modelo pode ser simplificado em um modelo linear contínuo que trata somente do dimensionamento de lotes.

O primeiro passo nesta simplificação consiste em remover as restrições (4), (5), (6), (7), (12), (13), (14), relativas ao seqüenciamento de lotes, que passam a ser tratadas diretamente pelo algoritmo genético (AG). Assim, a

variável y^{I}_{mls} pode ser completamente removida do modelo. Por outro lado, a variável y^{II}_{mjs} permanece no modelo como um parâmetro fornecido pelo AG. As variáveis responsáveis por controlar as trocas entre lotes, z^{I}_{mkls} e z^{II}_{mijs}, também são removidas e o AG passa a computar diretamente os custos de troca. A função objetivo, eq. (1), é reformulada para considerar apenas os custos de estoque e atraso. Como a sincronização entre linhas e tanques é relacionada ao seqüenciamento de lotes, ela também será tratada pelo AG, permitindo assim a remoção da restrição (10) e da variável v^{II}_{ms} do modelo. O termo $\sum_{i\in\delta_m}\sum_{j\in\delta_m}\sum_{s\in S_t} b^{II}_{ij} z^{II}_{mijs} + \sum_{s\in S_t} v^{II}_{ms}$ presente na restrição (9), que corresponde à soma dos tempos de troca considerando a sincronização, também passa a ser um parâmetro previamente computado pelo AG a partir das seqüencias de produção estabelecidas.

Uma alteração na forma como a variável x^{II} determina o dimensionamento de lotes é realizado, objetivando reduzindo a quantidade dessas variáveis no modelo. A alteração consiste em agregar a produção dos lotes de um produto em um macro-período em apenas uma variável, trocando a indexação da variável x^{II} de micro-períodos para macro-períodos. Deste modo, a variável passa a representar a quantidade total produzida por todos os lotes de um mesmo produto em um macro período. Para entender melhor essa alteração, vamos tomar como exemplo um macro-período com dois lotes distintos A e B de um mesmo produto. Considere que o valor do lote mínimo seja 10, o lote máximo seja 40, e a soma das quantidades produzidas pelos lotes A e B seja 50. Respeitando as restrições de lote mínimo e máximo, existem diversas combinações de quantidades para os lotes A e B em que a soma totaliza 50 e que possui o mesmo valor para a função objetivo. Por exemplo, algumas dessas

combinações poderiam ser: lote A igual a 10 e B igual a 40, A igual a 22,5 e B igual a 27,5, A igual a 30 e B igual a 20, A e B iguais a 25, entre outras. A partir deste exemplo, nota-se que é irrelevante para o modelo a quantidade produzida individualmente em cada lote, desde que as restrições de lote mínimo e máximo sejam respeitadas. Torna-se de real importância no modelo somente a quantidade total produzida do produto dentro do macro período. Assim a indexação da variável x^{II} pode ser alterada de micro-períodos para macro-períodos, sem perda de generalidade. Isso reduz consideravelmente a quantidade de variáveis utilizadas no modelo. Por exemplo, em uma instância do modelo com 3 macro-períodos e 25 micro-períodos por macro-período, obtém-se uma redução de 3*25=75 variáveis por produto para 3 variáveis por produto.

Com a alteração da indexação da variável x^{II}, as restrições de lote mínimo e máximo, eqs. (2) e (3), também devem ser alteradas. Considerando que cada lote de um produto deve ter uma quantidade mínima, a soma total das quantidades mínimas de todos os lotes de um mesmo produto em um macro período é igual à quantidade de lotes do produto no macro período multiplicada pelo valor do lote mínimo. Este valor corresponde à quantidade mínima que deve ser produzida do produto no macro período. O parâmetro y^{II}_{mjs} da restrição (2) pode ser convenientemente substituído por um novo parâmetro, q_{mjt}, que corresponde à quantidade de lotes do produto *j* na linha *m* e no macro período *t*, ou seja, como $q_{mjt} = \sum_{s \in S_t} y^{II}_{mjs}$. O mesmo raciocínio se aplica para a restrição de lote máximo. Assim, a substituição do parâmetro y^{II}_{mjs} por q_{mjt} também deve ser feita na restrição (3). As substituições nas eqs. (2) e (3) fazem com que o parâmetro y^{II}_{mjs} seja totalmente substituído pelo parâmetro q_{mjt} no modelo.

Analisando a restrição (11), vemos que a função dela é desativar as variáveis x^{II} que não possuem lotes atribuídos. Porém a nova restrição de lote máximo, descrita logo acima, já cumpre essa função. Desta forma, a restrição (11) pode ser excluída já que não é mais necessária no modelo.

Considerando todas as alterações descritas acima, o novo modelo simplificado que será associado ao AG é definido a seguir.

Conjuntos:

J = Conjunto de todas as bebidas;

M = Conjunto de todas as máquinas;

T = Conjunto de todos os períodos;

α_m = Conjunto de todas as bebidas que podem ser produzidas na máquina *m*;

λ_j = Conjunto de todas as máquinas que podem produzir a bebida *j*;

Dados:

d_{jt} = demanda da bebida *j* no período *t*;

h_j = custo unitário de estocar a bebida *j*;

g_j = custo unitário de atrasar a entrega da bebida *j*;

a_j^{II} = quantidade consumida de tempo para produção de uma unidade da bebida *j* na máquina *m*;

K_m^I = capacidade de tempo disponível do tanque *m*;

K_{mt}^{II} = capacidade de tempo disponível na máquina *m* para produção no período *t*;

r_j = quantidade consumida de xarope para produção de uma unidade da bebida *j*;

q_{mjt} = quantidade de sub-períodos no período *t* preparados para produção da bebida *j* na máquina *m*;

w_{mt} = quantidade de tempo consumida por trocas na produção na máquina *m* durante o período *t*;

Variáveis:

I_{jt}^{+} = quantidade em estoque da bebida *j* no período *t*;

I_{jt}^{-} = quantidade em atraso da bebida *j* no período *t*;

x_{mjt} = produção na máquina *m* da bebida $j \in \alpha_m$ durante o período *t*;

Modelo proposto:

$$Min \sum_{j \in J} \sum_{t \in T} (h_j I_{jt}^{+} + g_j I_{jt}^{-}) \tag{16}$$

Sujeito a:

$$I_{j(t-1)}^{+} + \sum_{m \in \lambda_j} x_{mjt}^{II} + I_{jt}^{-} = I_{jt}^{+} + I_{j(t-1)}^{-} + d_{jt} \qquad j \in J, t \in T \tag{17}$$

$$\sum_{j \in \alpha_m} a_j^{II} x_{mjt}^{II} + w_{mt} \leq K_{mt}^{II} \qquad m \in M, t \in T \tag{18}$$

$$r_j x_{mjt}^{II} \geq \frac{K_m^{I}}{8} q_{mjt} \qquad m \in M,\ j \in \alpha_m, t \in T \tag{19}$$

$$r_j x^{II}_{mjt} \leq K^I_m q_{mjt} \qquad m \in M,\ j \in \alpha_m, t \in T \quad (20)$$

$$\begin{aligned} I^+_{jt}, I^-_{jt} &\geq 0 && j \in J, t \in T \\ x^{II}_{mjt} &\geq 0 && m \in M,\ j \in \alpha_m, t \in T \end{aligned} \quad (21)$$

De fato, comparando o modelo simplificado com o modelo completo, fica bem claro que a simplificação reduz de forma significativa o tamanho do modelo. Este modelo simplificado agora possui apenas cinco restrições, onde a restrição (17) representa o balanço de estoque, a restrição (18) considera a capacidade disponível, as restrições (19) e (20) correspondem, respectivamente, às restrições de lote mínimo e máximo, e a restrição (21) apresenta o domínio das variáveis.

A próxima seção descreve o algoritmo genético híbrido, detalhando como as restrições de seqüenciamento de lotes são tratadas e como o modelo simplificado é associado ao AG.

3.2 Algoritmo Genético Híbrido

O algoritmo genético desenvolvido neste trabalho tem como base o algoritmo genético com população hierarquicamente estruturada apresentado em (Toledo *et al.*, 2009). A seguir é apresentado um pseudocódigo que descreve em linhas gerais o funcionamento do algoritmo genético híbrido proposto neste trabalho (Figura 10). Nas próximas seções será apresentado detalhadamente cada aspecto do AG proposto.

```
Algoritmo Genético Híbrido
início
  carregarModeloSimplificado();
  criar(população);
  para cada indivíduo na população faça
    inicializar(indivíduo);
    avaliarAptidão(indivíduo);
  fim para
  enquanto o tempo máximo de execução não for atingido faça
    enquanto a população não tiver convergido faça
      repetir taxa(cruzamento)*tamanho(população) vezes
        selecionar(pai1, pai2);
        filho = cruzamento(pai1, pai2);
        se probabilidade para ocorrer mutação satisfeita então
          mutar(filho);
        avaliarAptidão(filho);
        se aptidão(filho) > aptidão(pai1) ou aptidão(filho) > aptidão(pai2)
então
          inserir(filho, pai1, pai2);
      fim repetir
      reestruturar(população);
    fim enquanto
    para cada indivíduo na população exceto o melhor faça
      inicializar(indivíduo);
      avaliarAptidão(indivíduo);
    fim para
    se o melhor indivíduo atingiu o limiteMáximoDeVida faça
      inicializar(melhorIndivíduo);
      avaliarAptidão(melhorIndivíduo);
    fim se
  fim enquanto
fim

Avaliar Aptidão (indivíduo)
início
  parâmetros = decodificação&Reparo(indivíduo);
  soluçãoDoModelo = resoluçãoExataDoModeloSimplificado(parâmetros);
  aptidão(indivíduo) = calcularFunçãoAptidão(parâmetros, soluçãoDoModelo);
fim
```

Figura 10 - Pseudocódigo do Algoritmo Genético Híbrido proposto

3.2.1 Indivíduos

Para representar um indivíduo foi utilizada uma matriz indexada por linhas de produção e macro-períodos. Cada posição da matriz possui uma lista com a seqüência dos produtos (bebidas) que ocupam a linha naquele período. A seqüência dos xaropes nos tanques pode ser obtida a partir da seqüência das bebidas nas linhas, considerando que cada bebida corresponde a uma combinação única de um tipo de xarope com um tipo de vasilhame. Desta forma, torna-se desnecessária a representação do seqüenciamento dos xaropes no indivíduo. A Figura 11 abaixo exemplifica um possível indivíduo com seqüências de produtos geradas em cada período.

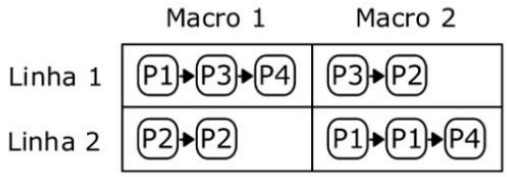

Figura 11 – Codificação para um Indivíduo

Como visto na seção 3.1, as restrições (4), (5), (6), (7), (12), (13) e (14), que correspondem ao seqüenciamento de lotes, foram removidas do modelo simplificado e devem passar a ser tratadas pelo AG. Todavia, várias restrições são satisfeitas pela representação do indivíduo por si só. A restrição (4) é satisfeita uma vez que a alocação dos produtos e xaropes é sempre feita partindo dos primeiros micro-períodos, permitindo que ocorram posições vazias somente no fim dos períodos. As trocas modeladas pelas restrições (5), (6) e (13) são obtidas diretamente das seqüências representadas em cada período do indivíduo. Também não há mais de uma troca para um mesmo xarope, e nem para um mesmo produto, satisfazendo assim as restrições (7) e (14). A representação do

indivíduo permite somente a alocação de um produto por micro-período satisfazendo a restrição (12). Desta forma, todas as restrições que dizem respeito ao seqüenciamento de lotes são respeitadas pela representação adotada.

3.2.2 Decodificação e Reparo

Uma vez que o seqüenciamento de lotes, representado por um individuo, será otimizado pelo modelo simplificado para se obter o melhor dimensionamento, faz-se necessário um processo de decodificação que converta as informações contidas no indivíduo para os parâmetros utilizados pelo modelo.

Os dois parâmetros fundamentais para execução do modelo que estão presentes no indivíduo são q_{mjt} e w_{mt}. A decodificação do parâmetro q_{mjt} pode ser feita de forma simples, basta percorrer cada linha *m* e período *t* do indivíduo contando quantas alocações do produto *j* foram feitas e atribuindo esse valor ao parâmetro. Já a decodificação do parâmetro w_{mt} exige um pouco mais de cuidado, pois devemos tratar a sincronização entre tanques e linhas. Quando vamos contar um tempo de troca entre dois lotes devemos verificar o tempo de troca na linha e no tanque, e considerar sempre o maior tempo, pois o estágio mais rápido deve sempre aguardar o mais lento estar pronto. Considerando isso, a decodificação do parâmetro w_{mt} é feita percorrendo cada linha *m* e período *t* somando todos os tempos de troca entre os lotes alocados, observando o maior entre o tempo de troca de produto e o tempo de troca de xarope para dois lotes consecutivos. Outro parâmetro importante contido na codificação do indivíduo é o valor total dos custos de troca de produtos e xaropes. Esse parâmetro será necessário no cálculo da função de aptidão apresentada posteriormente. A decodificação deste parâmetro é feita percorrendo todas as linhas e períodos

somando os custos de troca entre produtos e entre xaropes de acordo com as seqüências de lotes estabelecidas.

A figura logo abaixo ilustra como seria a decodificação da solução apresentada na Figura 11 em função dos parâmetros do modelo apresentado na seção 2.1.3. Considere que neste exemplo o produto 1 utilize o xarope 1, os produtos 2 e 3 utilizam o xarope 2, e o produto 4 utiliza o xarope 3.

m = 1 e t = 1	**m = 1 e t = 2**
$q_{111}=1;\ q_{121}=0;\ q_{131}=1;\ q_{141}=1;$ $w_{11}=\max(b_{12}^{I},b_{13}^{II})+\max(b_{23}^{I},b_{34}^{II});$	$q_{112}=0;\ q_{122}=1;\ q_{132}=1;\ q_{142}=0;$ $w_{12}=\max(b_{32}^{I},b_{43}^{II})+\max(b_{22}^{I},b_{32}^{II});$
m = 2 e t = 1	**m = 2 e t = 2**
$q_{211}=0;\ q_{221}=2;\ q_{231}=0;\ q_{241}=0;$ $w_{21}=\max(b_{22}^{I},b_{22}^{II});$	$q_{212}=2;\ q_{222}=0;\ q_{232}=0;\ q_{242}=1;$ $w_{22}=\max(b_{21}^{I},b_{21}^{II})+\max(b_{11}^{I},b_{11}^{II})$ $+\max(b_{13}^{I},b_{14}^{II});$
Custo total de troca	
$s_{12}^{I}+s_{13}^{II}+s_{23}^{I}+s_{34}^{II}+s_{32}^{I}+s_{43}^{II}+s_{22}^{I}+s_{32}^{II}+s_{22}^{I}+s_{22}^{II}+s_{21}^{I}+s_{21}^{II}+s_{11}^{I}+s_{11}^{II}+s_{13}^{I}+s_{14}^{II}$	

Figura 12 - Exemplo de decodificação da solução apresentada na Figura 11

Assim que termina a decodificação, ainda existe um detalhe importante a ser levado em consideração que diz a respeito da restrição de capacidade, eq. (18). Quando um dos tempos totais de troca de um período w_{mt} excede a capacidade da linha K_{mt}^{II}, a solução representada por esse indivíduo já pode ser considerada como infactível. Assim, para evitar o processamento de soluções infactíveis pelo modelo, foi criado um procedimento de reparo da solução. Este procedimento é bem simples, quando um período possuir uma seqüência de lotes com um tempo total de troca que ultrapassa a capacidade do período, lotes

desse período serão aleatoriamente removidos um a um até que o tempo total de troca não ultrapasse a capacidade.

3.2.3 Resolução Exata do Modelo Simplificado

Para realizar a otimização do modelo simplificado associado ao AG, foi feita uma implementação que faz uso do pacote de otimização ILOG CPLEX® (ILOG, 2008). Neste pacote está presente uma biblioteca denominada *Callable Library* que permite o desenvolvimento de modelos matemáticos utilizando as linguagens de programação C e C++. Nesta biblioteca há diversos métodos para resolução de problemas de programação linear e programação inteira.

Através desta biblioteca foi possível utilizar um método para a resolução do modelo simplificado para o dimensionamento de lotes. O primeiro passo desse método é instanciar uma matriz em memória que representa o modelo a ser resolvido. Cada linha dessa matriz representa uma restrição do modelo e cada coluna representa uma variável do modelo. Trata-se de uma matriz com dimensões *m* por *n*, onde *m* é o total de restrições e *n* é o total de variáveis. Cada posição *ij* da matriz armazena o coeficiente que a variável *j* assume na restrição *i*. Após a matriz ser instanciada, o método invoca outro método da biblioteca *Callable Library* que resolve o modelo e retorna o resultado.

No AG, o modelo simplificado é carregado em memória apenas uma vez antes do início da execução. Esse pré-carregamento evita que o modelo seja carregado várias vezes durante a execução do AG, o que consumiria tempo desnecessariamente. Com o modelo carregado em memória, sempre que for necessária a otimização de uma seqüência de lotes, basta re-ajustar as restrições correspondentes aos parâmetros q_{mjt} e w_{mt} e realizar uma nova otimização do modelo.

Considerando que os ajustes nas restrições envolvem somente a alteração dos limitantes das mesmas, a re-otimização dual pode ser utilizada para acelerar o processo de otimização do modelo. A re-otimização dual permite que as informações da solução da seqüência de lotes otimizada anteriormente possam ser usadas para acelerar a otimização da seqüência atual. Meyr (2002) utiliza essa mesma estratégia de re-otimização, porém a resolução do modelo é integrada nas buscas locais *threshold accepting* e *simulated annealing*. A biblioteca *Callable Library* possui essa rotina de re-otimização implementada para o algoritmo dual simplex, desta forma a otimização do modelo simplificado é executada utilizando essa rotina.

Quando a otimização do modelo termina, o algoritmo da biblioteca *Callable Library* retorna para o AG o melhor valor da função objetivo encontrado, ou seja, o menor custo de estocagem e atraso possível para seqüência de lotes otimizada. A partir desse ponto, a função de avaliação descrita na próxima seção pode ser computada.

3.2.4 Avaliação da Aptidão

A avaliação da aptidão de um indivíduo será feita com base no valor total dos custos gerados pela solução que esse indivíduo representa. Sendo assim, a função de avaliação é equivalente a eq. (1) do modelo apresentado na seção 2.1.3 que considera os custos de estocagem, atraso e troca. Porém, o calculo da função é feito de forma separada. O AG é o responsável por fazer o cálculo dos custos de troca, eq. (23), e o método exato é responsável pelo cálculo dos custos de estocagem e atraso, eq. (22).

$$\sum_{j=1}^{J}\sum_{t=1}^{T}(h_j I_{jt}^{+} + g_j I_{jt}^{-}) \qquad (22)$$

$$\sum_{m=1}^{M}\sum_{s=1}^{N}\sum_{k\in\theta_m}\sum_{l\in\theta_m} s_{kl}^{I} z_{mkls}^{I} + \sum_{m=1}^{M}\sum_{s=1}^{N}\sum_{i\in\delta_m}\sum_{j\in\delta_m} s_{ij}^{II} z_{mijs}^{II} \tag{23}$$

O primeiro passo durante o processo de avaliação de um indivíduo no AG (Figura 13) é a decodificação do mesmo. Após a decodificação, o AG fornece os parâmetros necessários para executar a otimização do modelo simplificado e obter o melhor ajuste no dimensionamento de lotes para a seqüência que o indivíduo representa. Neste ponto, o AG já determinou o valor total dos custos de troca que foi calculado durante o processo de decodificação. Após a resolução do modelo matemático a partir da seqüência fornecida pelo AG, os valores dos custos de estocagem e atraso são obtidos. Por último, esses custos são somados fornecendo o valor final da função de aptidão do indivíduo.

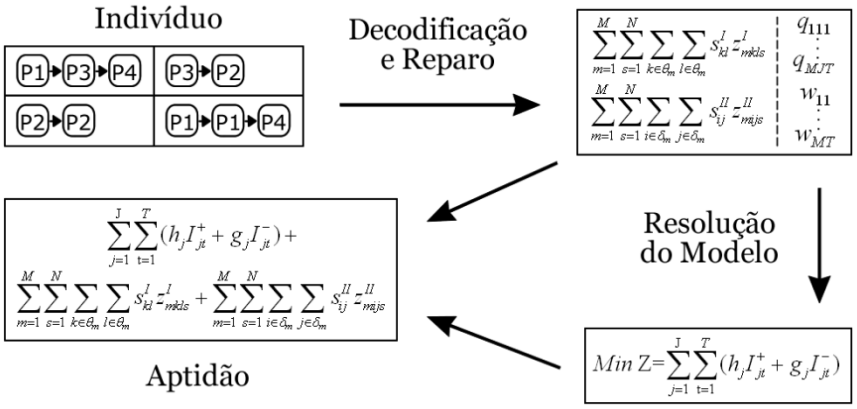

Figura 13 - Processo de avaliação de um indivíduo no AG

Tendo em mente que o objetivo do AG é minimizar os custos de produção, a comparação de aptidão apresentada no pseudocódigo (Figura 13), que avalia se um filho gerado é mais apto que um de seus pais, resulta em verdadeiro se o custo de produção do filho for menor que o custo de produção de um dos pais.

3.2.5 Aspectos Populacionais

Neste AG, as populações organizam seus indivíduos hierarquicamente em árvores binárias que são subdivididas em clusters. Cada cluster possui um nó líder e dois nós seguidores, onde o líder é o indivíduo no cluster que possui a melhor aptidão. A sobreposição de clusters, onde um indivíduo seguidor de um cluster atua como líder em outro, e a condição do líder do cluster ser o indivíduo que tem a melhor aptidão do cluster fazem com que o melhor indivíduo da população fique posicionado no topo da árvore (Figura 14).

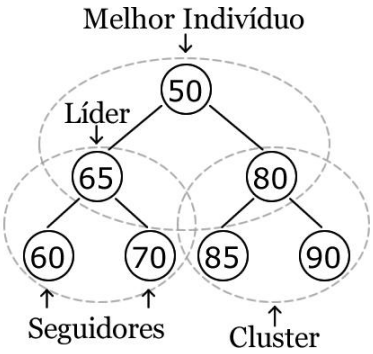

Figura 14 - Organização de uma população estruturada hierarquicamente

Com base nesta estruturação populacional, são utilizados métodos diferentes para seleção e inserção de indivíduos, quando comparados as abordagens convencionais empregadas em AGs. A seleção dos indivíduos para cruzamento é feita sorteando aleatoriamente um cluster e selecionando como pais o nó líder e um seguidor, aleatoriamente selecionado dentro desse mesmo cluster. Um novo indivíduo é inserido na população quando sua aptidão supera a aptidão de um de seus pais. Quando isso acontece o novo indivíduo substitui o pai menos apto.

No pseudocódigo do AG apresentado anteriormente, o laço de repetição que executa *taxa(cruzamento)*tamanho(população)* vezes, criando novos indivíduos, caracteriza um laço geracional, ou seja, cada vez que esse laço é executado por completo significa que uma nova geração foi criada. Sempre que uma nova geração é criada, e novos indivíduos são inseridos na população, o método de reestruturação é executado para reorganizar a população de forma manter a estruturação hierárquica entre os indivíduos (Figura 15). Porém, quando uma nova geração é criada, e nenhum novo indivíduo é inserido na população, ou seja, a população dessa nova geração é idêntica a da geração anterior, considera-se que a população convergiu. Nesse ponto, para se manter a variabilidade populacional, uma re-inicialização dos indivíduos é executada. Todos os indivíduos da população, exceto o melhor, são substituídos por novos indivíduos aleatoriamente inicializados. Se o melhor indivíduo da população não é atualizado de uma geração para outra, um contador é incrementado representando a "idade" do melhor indivíduo da população. Se o valor desse contador atingir a constante *limiteMáximoDeVida*, esse indivíduo também é reinicializado e seu contador é zerado.

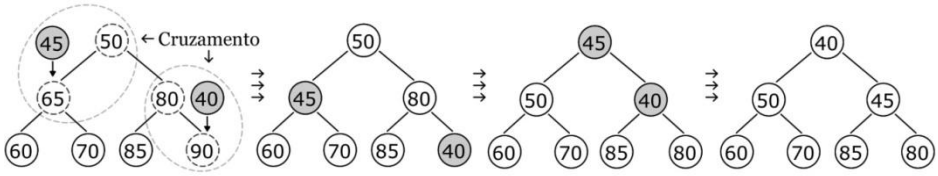

Figura 15 - Inserção de novos indivíduos e reorganização da população

3.2.6 Operadores Genéticos

O cruzamento de indivíduos é feito através do operador de recombinação uniforme apresentado na Figura 16. Esse operador trabalha percorrendo cada linha e macro período, onde os lotes em cada pai são

percorridos paralelamente e um sorteio aleatório é feito para decidir o lote de qual pai será copiado para o filho é executado. O sorteio considera a mesma probabilidade de escolha para cada pai. Nos períodos em que a seqüência de lotes de um pai for maior que a do outro, só há como percorrer os dois pais paralelamente até atingir o final do menor pai. A partir desse ponto, a seqüência restante no maior pai continua sendo percorrida, e o sorteio dos lotes é feito com uma probabilidade de 50% de ser ou não copiado para o filho.

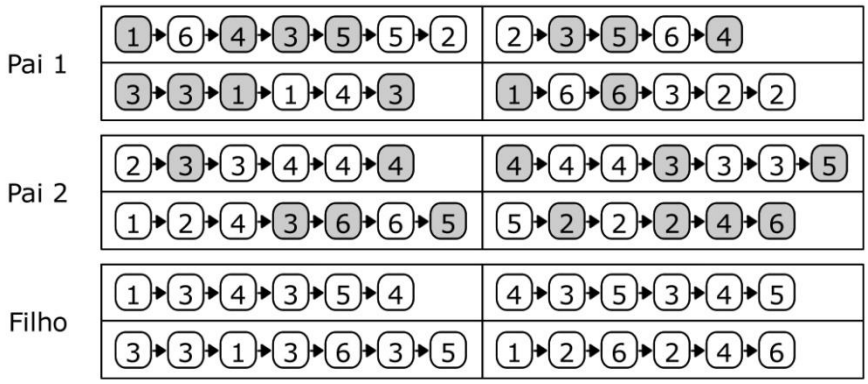

Figura 16 – Exemplo do operador de recombinação uniforme

Após a geração de um novo indivíduo pelo operador de recombinação, o AG faz o sorteio de um número entre zero e um para definir se este novo indivíduo será mutado. Se o número sorteado for menor que a taxa de mutação, que se trata de um valor $\lambda \in (0,1)$ previamente definido pelo usuário, o operador de mutação é aplicado no indivíduo. O operador de mutação deste AG é executado selecionando aleatoriamente um entre os sete tipos de mutação possíveis: O primeiro tipo de mutação escolhe aleatoriamente um produto e insere em uma posição aleatória. O segundo tipo de mutação remove um produto aleatoriamente selecionado. O terceiro tipo seleciona aleatoriamente uma posição e altera o produto alocado na posição para outro determinado aleatoriamente. O quarto tipo troca de posição dois produtos aleatoriamente

selecionados, dentro de uma mesma linha. O quinto tipo de mutação seleciona aleatoriamente uma linha e um período, e reorganiza aleatoriamente essa seqüência gerando uma nova seqüência com uma ordem diferente para os mesmos produtos. O sexto tipo remove um produto aleatoriamente selecionado de uma posição e o insere em outra posição também aleatoriamente determinada. O sétimo tipo sorteia aleatoriamente uma linha e um período. Um novo tamanho para a seqüência de produtos é sorteado entre 1 e o número máximo de micro-períodos. Se esse número for diferente do tamanho atual da seqüência, produtos sorteados aleatoriamente são removidos, caso o tamanho sorteado seja menor, ou inseridos, caso o tamanho sorteado seja maior. As inserções ou remoções são realizadas até que o tamanho do seqüencia alcance o novo tamanho sorteado.

4 RESULTADOS COMPUTACIONAIS

O método proposto foi avaliado através de testes com instâncias encontradas na literatura que possuem resultados obtidos por outros métodos. As instâncias adotadas para os testes foram obtidas de Ferreira *et al.* (2009) e foram criadas a partir de dados reais fornecidos por uma fábrica situada no Interior de São Paulo. A primeira instância (I1) foi gerada a partir dos dados associados a duas linhas de envase existentes na fábrica. Foram geradas mais quatro instâncias (I2, I3, I4 e I5) baseadas na instância I1. A Tabela 1 resume as características dessas instâncias. Estas instâncias consideram duas linhas de envase, 23 itens sendo produzidos na linha 1e 13 itens produzidos na linha 2, 18 xaropes, 3 macro-períodos e 25 micro-períodos por macro-período. Além dos dados das instâncias I1-I5, também foram coletados na fábrica dados relativos à demanda de um período de 30 semanas. Estas demandas permitiram a geração de outras 10 instâncias (I6-I15). Cada uma dessas 10 instâncias está associada à demanda de três semanas consecutivas de produção na fábrica. As instâncias I6, I7, I14 e I15 correspondem a períodos de demanda mais alta do que as instâncias I8-I13, aumentando o nível de dificuldade nessas instâncias. Maiores detalhes são descritos em (Ferreira *et al.*, 2009).

Tabela 1 - Características dos exemplares I1-I5 (Ferreira *et al.*, 2009)

Instância	**Modificações**
I1	Dados originais fornecidos pela fábrica.
I2	Custos de estoque do I1 dobrados.
I3	Custos de atraso do I1 dobrados.
I4	Demanda total de cada item do I1 redistribuída aleatoriamente nos três macro-períodos.
I5	Capacidade das máquinas reduzida em 25%.

Ferreira *et al.* (2009) avaliou abordagens com diversas combinações de parâmetros do CPLEX e uso de diferentes estratégias *Relax and Fix*. Cada uma

das execuções dessas abordagens foi feita dentro de um limite máximo de 4 horas. Esse limite foi baseado no fato de que na fábrica é gasto esse mesmo tempo para estabelecer a programação da produção. Os resultados dos testes realizados levaram os autores à conclusão que a abordagem *Relaxation Approach* (RA) se destacou entre as demais. Desta forma os melhores resultados da abordagem RA foram escolhidos para serem comparados com os resultados obtidos pelo AG.

Considerando que o AG é um método aleatório, execuções diferentes para uma mesma instância podem retornar resultados diferentes. Assim, os resultados do AG foram obtidos a partir da média de 10 execuções em cada uma das instâncias, com um limite de tempo de 3 minutos para cada execução. O limite de tempo foi escolhido baseado no tempo necessário para que o AG alcance convergência estabilizando em uma solução, sem apresentar melhorias significativas após esse limite. O AG foi ajustado para executar com uma população estruturada em árvore binária com 15 indivíduos, taxa de cruzamento de 1,7, taxa de mutação de 0,7 e *limiteMáximoDeVida* definido como 20. O desenvolvimento do AG foi feito na linguagem de programação C++ utilizando o ambiente de programação Microsoft Visual C++ 2008 Express Edition. Também foi utilizada a biblioteca CPLEX® Callable Library do pacote ILOG CPLEX® 11.1. A execução do AG utilizou uma máquina com 1.0 GB de RAM e processador Pentium Intel 4 com freqüência de 3.2 GHz, compatível com a configuração da máquina usada nas execuções em Ferreira *et al.* (2009).

A tabela a seguir compara os resultados que o AG obteve com os resultados obtidos pela abordagem RA. Os resultados são apresentados em termos dos custos de produção determinados pela eq. (1) descrita na seção 2.1.3. A tabela apresenta para cada instância os melhores resultados obtidos por RA, e também a média, o melhor e o pior resultado obtido pelas execuções do AG. Além disso, a tabela também apresenta os desvios dos valores obtidos pelo AG

em comparação com os valores obtidos por RA. A fórmula utilizada para determinar os desvios segue logo abaixo.

$$desvio_{AG,RA} = \frac{resultado_{AG} - resultado_{RA}}{resultado_{RA}} \quad (24)$$

Tabela 2 - Comparação dos resultados obtidos pelos métodos em I1-I15

	RA	AG Híbrido					
Instâncias	**Melhor**	**Média**	**Desvio**	**Melhor**	**Desvio**	**Pior**	**Desvio**
I1	306.834	211.716	-31.0%	200.392	-34.7%	220.702	-28.1%
I2	276.185	227.992	-17.4%	201.432	-27.1%	243.651	-11.8%
I3	290.841	217.501	-25.2%	198.334	-31.8%	235.812	-18.9%
I4	314.402	214.132	-31.9%	202.623	-35.6%	220.595	-29.8%
I5	348.554	247.719	-28.9%	233.591	-33.0%	260.059	-25.4%
I6	526.473	404.145	-23.2%	386.884	-26.5%	418.739	-20.5%
I7	509.464	300.872	-40.9%	277.923	-45.4%	315.906	-38.0%
I8	509.668	267.329	-47.5%	235.106	-53.9%	282.938	-44.5%
I9	412.237	234.074	-43.2%	213.094	-48.3%	246.715	-40.2%
I10	429.868	211.397	-50.8%	198.486	-53.8%	222.639	-48.2%
I11	289.170	226.220	-21.8%	215.442	-25.5%	240.229	-16.9%
I12	491.725	245.512	-50.1%	221.571	-54.9%	267.634	-45.6%
I13	369.540	229.540	-37.9%	218.128	-41.0%	243.812	-34.0%
I14	449.511	272.063	-39.5%	243.764	-45.8%	286.869	-36.2%
I15	446.194	338.965	-24.0%	324.228	-27.3%	363.536	-18.5%

Analisando os resultados na tabela acima, é visível que mesmo o AG tendo sido executado em um limite máximo de 3 minutos, um intervalo de tempo muito menor comparado às 4 horas de execução da abordagem RA, ele supera os resultados obtidos por RA em todas as instâncias. Mesmo as piores execuções do AG obtiveram resultados que superam RA com desvios expressivos, chegando a taxas de até 48,2% de melhoria. Os resultados médios do AG apresentam grandes desvios em relação aos resultados de RA, variando de 17,4% a 50,8%. Até as melhores execuções, que apresentam melhorias

significativas nos desvios comparadas às médias, devem ser consideradas, pois o tempo total das 10 execuções do AG para cada instância é 30 minutos, intervalo de tempo ainda relativamente bem inferior às 4 horas de execução da abordagem RA. Logo, o responsável na indústria poderia executar 10 vezes o método e escolher efetivamente a melhor solução para ser utilizada na programação da produção. Esse processo levaria 30 minutos, tempo bastante inferior às 4 horas atualmente gastas no planejamento da produção.

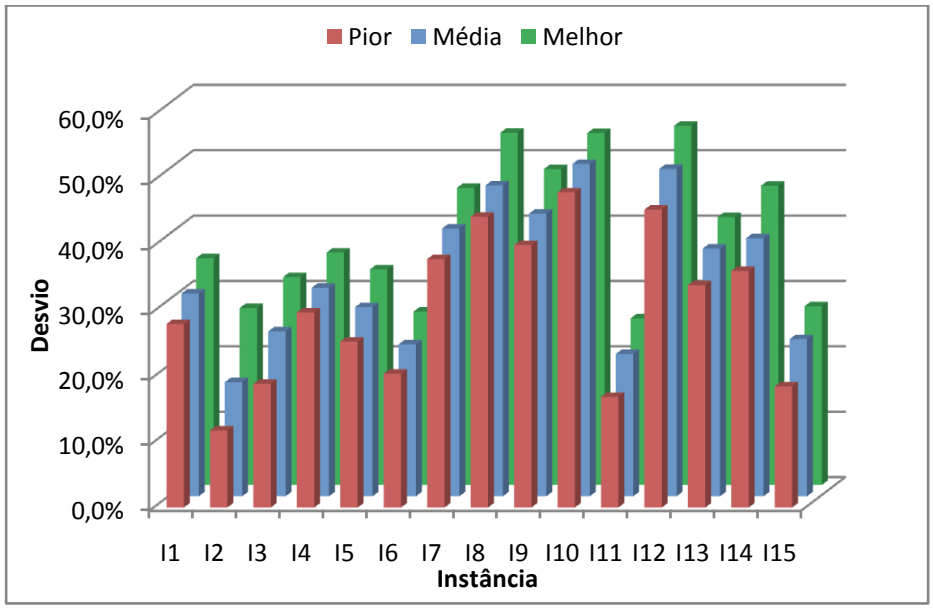

Figura 17 - Gráfico dos desvios apresentados na Tabela 2

No gráfico acima são apresentados os desvios dos resultados vistos na Tabela 2, porém com o sinal invertido. Assim, os maiores valores passam a ser os melhores. Nas instâncias I1-I5 o AG obteve os menores desvios nas instâncias I2 e I3 que correspondem, respectivamente, às instâncias com custos de estoque e troca dobrados. Os melhores desvios aparecem para I1, I4 e I5 que correspondem, respectivamente, às instâncias que tem os dados originais da fábrica, demanda redistribuída, e capacidade reduzida. A redução da capacidade

em I5 adiciona um grau de dificuldade maior ao problema, mas não foi obstáculo ao bom desempenho do método nesta instância. No conjunto de instâncias I6-I15, relativo a um período de demanda de 30 semanas da fábrica com três semanas consecutivas de demanda pra cada instância, o AG conseguiu obter os desvios ainda mais expressivos. Dentre as dez instâncias, sete obtiveram os maiores desvios, sendo que três delas apresentaram desvios da média entre 35% e 40% e quatro com desvios entre 40% e 50%. As três instâncias que obtiveram desvios menores ficaram com valores em torno de 20%. Nesse conjunto de instâncias, I6, I7, I14 e I15 correspondem a períodos de demanda mais alta do que as demais, aumentando o nível de dificuldade. Das quatro instâncias, duas (I7 e I14) ficaram entre as que obtiveram os maiores desvios e outras duas (I6 e I15) ficaram entre as que obtiveram menores desvios.

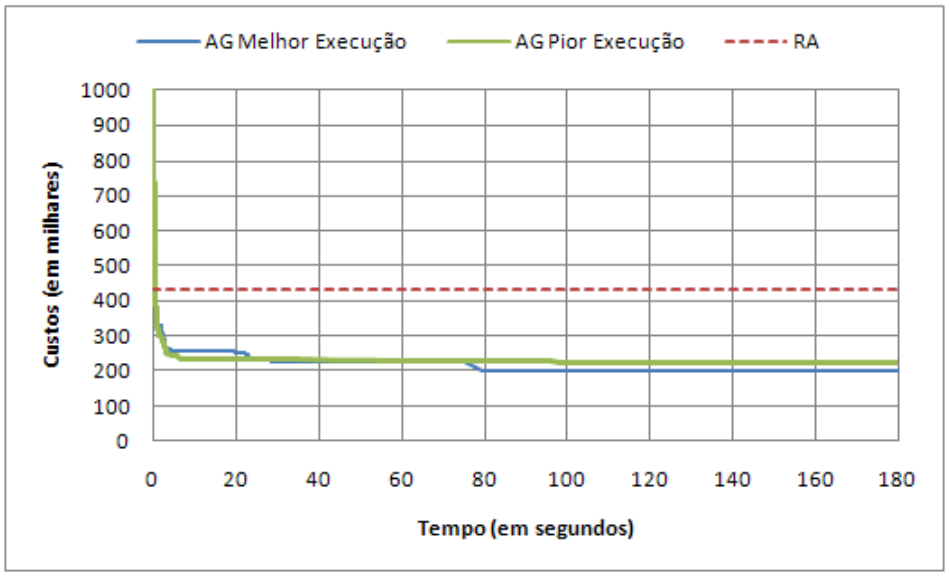

Figura 18 - Melhor e pior execução do AG na instância I10

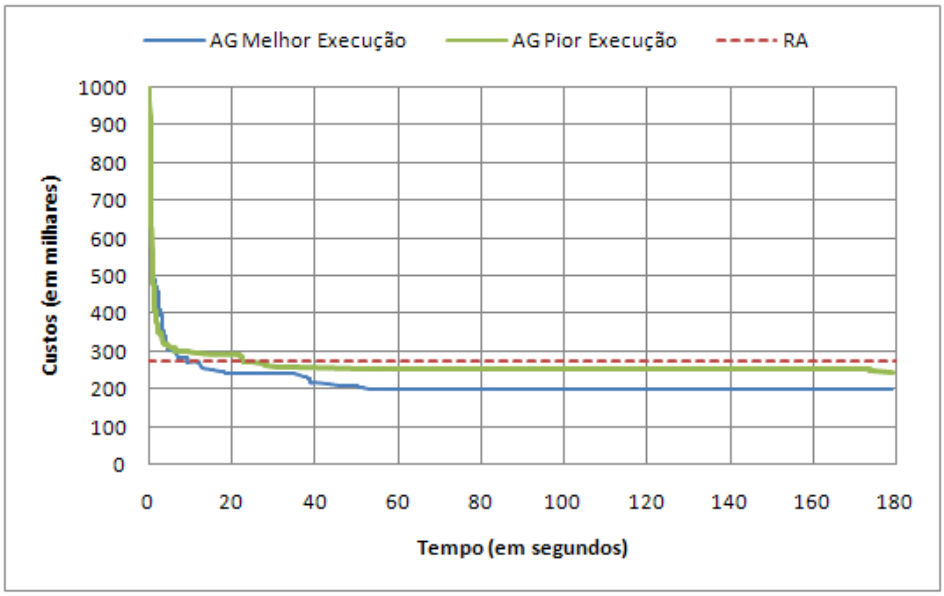

Figura 19 - Melhor e pior execução do AG na instância I2

O AG obteve o maior e o menor desvio médio nas instâncias I10 e I2, respectivamente. Por isso, essas instâncias foram selecionadas para avaliar a convergência do AG. Essa avaliação é feita através de um gráfico que apresenta o valor da melhor solução encontrada em função do tempo (Figura 18 e 19). A linha tracejada no gráfico representa o valor da solução obtida por RA em 4 horas.

Os gráficos acima demonstram que o tempo que o AG leva para obter uma solução equivalente a solução obtida por RA é bem menor que os 3 minutos definidos como limite máximo de execução. Por isso, a fim de comparar o desempenho entre os dois métodos, também são apresentadas na tabela abaixo as médias do tempo em que o AG leva para chegar a uma solução equivalente ou melhor que a solução obtida por RA. A média é calculada para um total de 10 execuções. O *speedup* também é calculado (eq. (25)) e apresentado na tabela a seguir.

$$speedup_{AG,RA} = \frac{tempo_{RA}}{tempo\ médio_{AG}} \quad (25)$$

Tabela 3 - Desempenho do AG em relação à RA considerando o tempo

	AG Híbrido	
Instância	**Tempo Médio (segundos)**	***Speedup***
I1	4.29	3360
I2	12.67	1136
I3	6.56	2194
I4	4.21	3420
I5	4.22	3416
I6	6.71	2147
I7	2.46	5860
I8	1.24	11577
I9	1.30	11060
I10	0.88	16399
I11	6.42	2243
I12	1.14	12642
I13	1.89	7629
I14	2.39	6031
I15	9.56	1506

De acordo com os valores da Tabela 3, o AG supera em muito a abordagem RA em termos de tempo computacional. As mesmas soluções que RA leva horas para encontrar, o AG leva poucos segundos. O AG gastou em média 12,67 segundos para obter a mesma solução que RA. O ganho de desempenho (*speedup*) do AG comparado a RA fica na casa de milhares, sendo que o AG é de 1136 a 16399 vezes mais rápido que RA nas instâncias testadas.

Dessa forma, o AG híbrido associado ao método exato foi capaz de superar a abordagem RA proposta por Ferreira *et al.* (2009), tanto em tempo de execução quanto em qualidade das soluções finais obtidas.

5 CONCLUSÕES

A presente monografia objetivou desenvolver e estudar um método híbrido que consiste na integração de um algoritmo genético com um método exato, voltado para a resolução do problema da produção de refrigerantes. A idéia principal desta abordagem é utilizar um algoritmo genético para otimizar o seqüenciamento de lotes, que corresponde a parte discreta do problema, integrando a resolução exata de um modelo matemático para otimizar o dimensionamento de lotes, correspondente a parte contínua do problema.

Existem na literatura diversos modelos matemáticos que descrevem diferentes variações do problema da produção de refrigerante. Neste trabalho foi adotado o modelo matemático proposto por Ferreira *et al.* (2009), apresentado na seção 2.1.3, que descreve o problema da produção de bebidas considerando o seqüenciamento e dimensionamento de lotes nos tanques e linhas de produção, onde cada linha possui um tanque dedicado.

Uma vez que a otimização do seqüenciamento de lotes foi realizada pelo algoritmo genético, o modelo matemático pode ser simplificado para tratar somente o dimensionamento de lotes. Desta forma diversas restrições e variáveis que tratam o seqüenciamento de lotes foram removidas do modelo. A função objetivo passou a considerar apenas os custos relativos ao dimensionamento de lotes. Além disso, também foi alterada a variável que determina o dimensionamento de lotes, reduzindo assim a quantidade de variáveis contínuas no novo modelo proposto. Todas as alterações propostas são apresentadas na seção 3.1 levaram a um modelo simplificado capaz de ser resolvido por um método exato de programação linear.

O algoritmo genético desenvolvido neste trabalho tem como base o algoritmo genético com população hierarquicamente estruturada apresentado em (Toledo *et al.*, 2009). Neste AG, as populações organizam seus indivíduos

hierarquicamente em árvores binárias que são subdivididas em clusters, onde os indivíduos mais aptos ocupam o topo da árvore. Para representar um indivíduo, foi utilizada uma matriz indexada por linhas de produção e macro períodos, onde cada posição da matriz possui uma lista com a seqüência dos produtos (bebidas) que ocupam a linha naquele período. Adotando essa codificação, foi necessário criar um processo de decodificação que convertesse as informações contidas na codificação do indivíduo para os parâmetros utilizados pelo modelo. A resolução do modelo simplificado foi feita com uma implementação usando a biblioteca *Callable Library* do pacote de otimização ILOG CPLEX®. Assim, antes da execução do AG, o modelo simplificado é carregado em memória. Com o modelo em memória, sempre quando for necessária a otimização de uma seqüência de lotes, basta ajustar os parâmetros e realizar a re-otimização do modelo para os novos parâmetros. A avaliação da aptidão de um indivíduo é feita com base no valor total dos custos gerados pela solução que esse indivíduo representa. O cruzamento de indivíduos é feito através do operador de recombinação uniforme, e a mutação através de um operador composto por quatro diferentes tipos de mutação.

A avaliação da abordagem proposta foi feita através de testes com instâncias encontradas na literatura que possuem resultados obtidos por outros métodos, permitindo que a abordagem possa ser comparada com tais métodos. Um conjunto de 15 instâncias propostas por Ferreira *et al.* (2009) foram avaliadas. Estas instâncias foram criadas a partir de dados reais fornecidos por uma fábrica situada no Interior de São Paulo. Ferreira *et al.* (2009) apontaram a abordagem *Relaxation Approach* (RA) como a de melhor desempenho na resolução dessas instâncias. Por isso, os melhores resultados da abordagem RA foram comparados aos resultados obtidos pelo AG.

Os resultados do AG foram obtidos a partir da média de 10 execuções em cada uma das instâncias, com um limite de tempo de 3 minutos para cada

execução. O limite de tempo foi escolhido baseado no tempo necessário para que o AG estabilize em uma solução.

Apesar do tempo reduzido fornecido ao AG, a meta-heurística superou os resultados encontrados por RA em todas as instâncias. Mesmo as piores execuções do AG obtiveram resultados que superam RA com desvios expressivos, chegando a taxas de até 48,2%. Os resultados médios do AG apresentam desvios expressivos em relação aos resultados de RA, variando de 17,4% a 50,8%. Até mesmo as melhores execuções, que apresentam melhorias significativas nos desvios comparadas às médias, devem ser consideradas, pois o tempo total das 10 execuções do AG para cada instância é 30 minutos, intervalo de tempo ainda relativamente bem inferior às 4 horas de execução da abordagem RA.

Foi feito um estudo da convergência da melhor e pior execução nas instâncias I10 e I2, instâncias que obtiveram o maior e o menor desvio nos resultados médios, respectivamente. Neste estudo, foi possível concluir que o tempo que o AG leva para obter uma solução equivalente a solução obtida por RA é bem menor que os 3 minutos definidos como limite máximo de execução. O AG gastou na média entre 0,88 e 12,67 segundos para obter a mesma solução que RA no conjunto de instâncias avaliadas. A partir dos tempos médios do AG, e do tempo de execução de 4 horas da abordagem RA, também foi calculado o ganho de desempenho (*speedup*) do AG em relação à RA. O *speedup* do AG comparado a RA fica na casa de milhares, sendo que o AG é de 1136 a 16399 vezes mais rápido que RA nas instâncias testadas.

O AG híbrido com um método exato proposto neste trabalho se demonstrou capaz de superar a abordagem RA proposta por Ferreira *et al.* (2009), tanto em tempo de execução do método, quanto em qualidade das soluções, obtendo valores melhores em intervalos de tempo consideravelmente reduzidos.

Uma perspectiva futura para esse trabalho seria testar essa abordagem em um conjunto de instâncias mais complexo. Isso se justifica pelo fato que o AG foi capaz de encontrar boas soluções em intervalos de tempo relativamente pequenos pra instâncias de médio porte, sendo interessante agora avaliar como a abordagem se comporta com instâncias de grande porte.

No estudo de convergência do AG, ficou claro que o método converge rápido pra boas soluções. Porém uma vez achando uma solução boa ele estabiliza e fica preso naquela solução. Desta forma, novas abordagens de algoritmos genéticos podem ser propostas para tentar evitar que essa convergência prematura aconteça.

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: 20 de Março de 2009.

APPLEGATE, D., BIXBY, R., CHVÁTAL, V. e COOK, W., **The Traveling Salesman Problem: A Computational Study**, Princeton University Press, 2006.

FLEISCHMANN, B. e MEYR, H., **The general lotsizing and scheduling problem**, Operations Research Spektrum, Volume 19, p. 11-21, 1997.

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

BITRAN, G. R. e YANASSE, H. H., **Computational complexity of the capacited lot size problem**, Management Science, Volume 28, p. 1174–1186, 1982.

BLUM, C. e ROLI, A., **Metaheuristics in combinatorial optimization: Overview and conceptual comparison**, ACM Computing Surveys, 35(3), 268-308, 2003.

CHELOUAH, R., SIARRY, P., **Genetic and Nelder-Mead algorithms hybridized for a more accurate global optimization of continuous**

multiminima functions, European Journal of Operational Research, 148(2), 335-348, 2003.

DEFERSHA, F.M., CHEN, M., **A linear programming embedded genetic algorithm for an integrated cell formation and lot sizing considering product quality**, European Journal of Operational Research, Volume 187, p. 46-49, 2008.

FERREIRA, D., **Abordagens para o Problema Integrado de Dimensionamento e Sequenciamento de Lotes da Produção de Bebidas**, Tese de Doutorado, Universidade Federal de São Carlos, Departamento de Engenharia de Produção, Dezembro, 2006.

FERREIRA, D., MORABITO, R., RANGEL, S., **Um modelo de otimização inteira mista e heurísticas relax and fix para a programação da produção de fábricas de refrigerantes de pequeno porte**, Produção, Volume 18 (1), p. 76-88, 2008.

FERREIRA, D., MORABITO, R., RANGEL, S., **Solution approaches for the soft drink integrated production lot sizing and scheduling problem**, European Journal of Operational Research, Volume 196 (2), p. 697-706, 2009.

FRANÇA, P.M., MENDES, A.S. E MOSCATO, P., **A memetic algorithm for the total tardiness single machine scheduling problem**, European Journal of Operational Research, 1(132), 224-242, 2001.

FRENCH, A.P., ROBINSON, A.C., WILSON, J.M., **Using a hybrid genetic-algorithm/branch and bound approach to solve feasibility and optimization integer programming problems**, Journal of Heuristics 7, 551–564, 2001.

GEN, M., CHENG, R., **Genetic algorithms & engineering design**, John Wiley & Sons New York, 1997.

GLOVER, F., **Future paths for integer programming and links to artificial intelligence**, Computers & Operations Research, 13, 533–549, 1986.

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

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

HOUPT, RL, HOUPT, SE, **Practical genetic algorithms,** New York, NY: Wiley, 1998.

ILOG INC., **ILOG CPLEX Callable Library C API 11.1 Reference Manual**, Copyright, ILOG, 2008.

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.

JOURDAN, L., BASSEUR, M. E TALBI, E-G., **Hybridizing Exact Method and Metaheuristics: A Taxonomy**, EJOR, 199 (3), 620-629, 2009.

KAELO, P., ALI, M.M., **Integrated crossover rules in real coded genetic algorithms,** European Journal of Operational Research, 176(1), 60-76, 2007.

LAFIS empresa de consultoria, **relatório requisitado pela Indústria de Bebidas Ipiranga**, Ribeirão Preto, 2003.

LINDEN, R., **Algoritmos Genéticos: uma importante ferramenta da Inteligência Computacional**, Rio de Janeiro: Braspor, 2006.

LUCENA, A. e BEASLEY, J. E., **A branch and cut algorithms**, Advances In Linear And Integer Programming, Oxford University Press, Inc., New York, NY, 1996.

MAES, J. O. E VAN WASSENHOVE, L. N., **Multilevel capacitated lotsizing complexity and lp-based heuristics,** European Journal of Operational Research, 53:131–148, 1991.

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

OSMAN, I.H. e KELLY, J.P., **Meta-heuristics: Theory and applications**, Kluwer, Boston, 1996.

PADBERG, M. e RINALDI, G., **A branch-and-cut algorithm for the resolution of large-scale symmetric traveling salesman problems**, SIAM Review 33, p. 60-100, 1991.

PURNAPRAJNA, M., REFORMAT M., PEDRYCZ, W., **Genetic algorithms for hardware-software partitioning and optimal resource allocation**, Journal of Systems Architecture, 53(7), 339-354, 2007.

REEVES, C. R., **Genetic Algorithms**. In: Handbook of Metaheuristics, F. Glover e G.A. Kochenberger, Kluwer, Boston, 55-82, 2003.

REEVES C., **Modern Heuristics Techniques for Combinatorial Problems**, John Wiley & Sons, 1993.

TALBI, E.G., **A taxonomy of hybrid metaheuristics**, Journal of Heuristics, 8(2), 541–564, 2002.

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, Volume 47, p. 3097-3119, 2009.

VOSE, M. D., **The Simple Genetic Algorithm: Foundations and Theory**, Cambridge, MA: MIT Press, 1998.