Aplicações heurísticas para um caso real do problema de carregamento de container

Jeanne Dobgenski*
Mestre em Engenharia de Sistemas - UNICAMP
Profesora das Faculdades de Valinhos
Coordenadora do Instituto de Pesquisas Aplicadas e Desenvolvimento Educacional - IPADE
e-mail: jeanne.dob@unianhanguera.edu.br

Douglas Fugita de Oliveira Cezar
Especialista em Tecnologia na Gestão da Informação
Aluno egresso dos cursos de bacharelado em Ciência da Computação
e pós-graguação em Tecnologia na Gestão da Informação
Faculdades de Valinhos
e-mail: douglas_cezar@hotmail.com

Resumo

Este trabalho estuda o Problema de Carregamento de Container verificado em um caso real, numa indústria de fitas adesivas. Tal problema consiste em alocar uma série de caixas dentro de um container com objetivo de otimizar a sua ocupação. O passo inicial é a criação de torres de caixas, utilizando uma heurística construtiva que minimize o espaço não utilizado em cada torre. Após, as torres serão alocadas visando maximizar o espaço utilizado do container. Para isto, utilizam-se duas estratégias: uma baseada em Algoritmos Genéticos e outra em Simulated Annealing. A função objetivo considera todos os espaços não preenchidos do container. Os algoritmos desenvolvidos são superiores à técnica que a empresa emprega, para atender ao pedido do cliente, de forma que os resultados obtidos fornecem uma melhoria média de 17 % a mais de caixas alocadas. Foram realizados testes genéricos que mostraram uma melhoria média de 3% do Simulated Annealing com relação a implementação do Algoritmo Genético.

Palavras-chave: Algoritmos Genéticos, Simulated Annealing, Problema de Carregamento de Container.

Abstract

This work deals with a real case of Container Loading Problem, which happens in a adhesive tape industry. This problem consists in put many boxes inside of a container with maximal space occupation objective. The first step to solve that problem is built boxes´s towers, using a constructive heuristic to minimize the free space in which tower. Then, these towers are allocated in a way to maximize the used space. To do this task are used two strategies, one Genetic Algorithm and other Simulated Annealing. The defined objective function considers all the container non used spaces, The computational tests show the proposed algorithms performance is better than the own industry technique. The Simulated Annealing average results was 17% better than the original ones and 3% better than the Algorithm Genetic results.

Key-words: Genetic Algorithm, Simulated Annealing, Container Loading Problem.

**Bolsistas FUNADESP* - Fundação Nacional de Desenvolvimento do Ensino Superior Particular

Introdução

O Problema de Carregamento de Container (*Container Loading Problem*) é um problema de otimização combinatória permutacional classificado como NP-Difícil, devido a não existência de um algoritmo polinomial que o resolva em um tempo computacional aceitável.

A solução original deste problema consiste em organizar o carregamento de caixas retangulares em um container, com dimensões fixas, sendo o objetivo minimizar o volume não preenchido do container. Além disso, podem ser adicionadas algumas restrições necessárias ao carregamento como o limite de peso do container, empilhamento máximo de cada caixa, rotação, estabilidade da carga e estabilidade das caixas. Estes fatores influenciam diretamente na complexidade do algoritmo e na factibilidade da solução.

Este trabalho tem como objetivo o desenvolvimento de uma solução para um caso específico do Problema de Carregamento de Container, com a utilização das metaheurísticas Algoritmos Genéticos (GOLDBERG, 1989) e *Simulated Annealing* (KIRKPATRIK et al, 1983).

O problema abordado visa organizar uma série de caixas em um único container retangular de dimensões fixas e com um único compartimento. Não há a necessidade de controles mais apurados como o peso máximo a ser carregado, ordem para descarregamento, altura máxima de empilhamento ou estabilidade da carga.

Os algoritmos genéticos são algoritmos baseados na teoria da evolução, de Darwin, na qual várias soluções são tratadas como indivíduos. A cada geração os melhores indivíduos (soluções) são mantidos, passando por cruzamentos e mutações. Com isso as melhores soluções tendem a aparecerem com o passar das gerações. No caso de problemas permutacionais, como é o caso do problema abordado, os algoritmos genéticos possuem operadores específicos, devido às dificuldades de se manter uma solução válida (GOLDBERG, 1989).

O *Simulated Annealing* é uma metaheurística que simula o resfriamento de um conjunto de átomos aquecidos, baseando-se em um fundamento da termodinâmica conhecido como recozimento, no qual um material é aquecido até o ponto de fusão e passa em seguida por uma diminuição lenta e gradual de temperatura se tornando uma estrutura cristalina. Com base neste conceito é feita a melhoria de uma solução que com o passar de cada iteração, simulando a diminuição da temperatura, tende a uma solução de boa qualidade (KIRKPATRIK et al, 1983).

O problema a ser estudado pode ser dividido em fases distintas. A primeira fase consiste em encontrar a quantidade máxima de caixas a serem colocadas no container, que é baseada em porcentagens de produtos a serem armazenadas. A segunda fase se inicia após ser definida uma quantidade aproximada de caixas a serem acomodadas, criando torres de caixas a serem acomodadas no container. Utilizando uma heurística construtiva é gerada uma solução, visando à maximização da quantidade de torres de caixas em um único container. O próximo passo é a utilização da metaheurística Algoritmo Genético e, em outra implementação, o *Simulated Annealing* para que a solução obtida seja melhorada.

Trabalhos Correlatos

Geralmente, as soluções existentes na literatura tratam de classes específicas do problema do container, mas por haver muitas variáveis que podem ser inseridas no problema, há muitas situações para serem pesquisadas. Pode-se identificar algumas grandes classes como a *Strip Packing Problem*, na qual o container possui altura e largura fixa, mas sua profundidade é infinita, tendo como objetivo carregar todas as caixas, organizando-as na menor profundidade possível. Bortfeldt e Gehring (2001) utilizaram a metaheurística Busca Tabu (*Tabu Search Algorithm*) preenchendo os espaços retangulares que ainda não foram preenchidos, que possuem dimensões definidas. Esses autores encontraram soluções para o *Strip Packing* utilizando algoritmos genéticos. Foi utilizado o preenchimento por camadas, não necessitando assim da definição de uma profundidade. O algoritmo é utilizado duas vezes, uma vez unindo os espaços restantes vizinhos quando possível, e na segunda vez este artefato não está disponível, ficando os espaços separados. A que obtiver o melhor resultado é tida como a solução para o problema.

Pureza e Morabito (2003) utilizaram o algoritmo Busca Tabu com heurística de blocos para resolver o problema de paletes do produtor que pode ser considerado um problema de container de duas dimensões. Foram obtidos ótimos resultados utilizando uma estratégia de vizinhança baseada na expansão de blocos, auxiliada por um procedimento de

deslizamento de blocos.

Outra classe de problema é a *Knapsack Loading Problem,* na qual cada caixa tem um peso associado e o objetivo é encontrar um conjunto de caixas que maximize o valor resultante da soma dos pesos das caixas. Se o peso é baseado no tamanho das caixas, o problema a ser resolvido é minimizar o espaço desperdiçado. Pisinger (2000) utilizou um método de construção de paredes verticais, utilizado pela primeira vez por George e Robinson (1980). As paredes foram divididas em camadas que, por sua vez, foram divididas em um número de faixas horizontais. Para achar a solução do problema, foi usada uma heurística baseada em árvore de busca, resultando em 91% de taxa de preenchimento para problemas heterogêneos.

Uma variação do *Bin Packing Problem* é conhecida como *Multi-Container Loading*: no qual se deve carregar as caixas no menor número possível de containeres com dimensões diferentes. Eley (2002) utilizou um algoritmo guloso que constrói blocos homogêneos de caixa que após é otimizado com uma árvore de busca. O algoritmo se mostrou muito bom comparado com outros existentes na literatura, principalmente em casos de problemas fortemente heterogêneos.

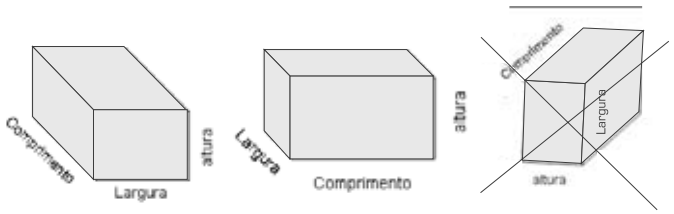
Problema Abordado

O problema a ser solucionado é encontrado na Adelbrás Indústria e Comércio de Adesivos Ltda, indústria de fitas adesivas localizada na cidade de Valinhos - São Paulo - Brasil, na qual o carregamento de containeres é feito através de tentativa e erro. Ou seja, utilizando cálculos matemáticos manuais, sendo a tarefa de organização da disposição das caixas feita com a experiência dos funcionários.

Segundo Vera Hamerski - supervisora de vendas e comércio exterior da Adelbrás, o método utilizado consiste na construção de paredes verticais de caixas, organizando as caixas de diferentes medidas, sempre da maior caixa para a menor. São inseridas caixas, uma ao lado da outra, até que não seja mais possível devido à largura do container, e estas caixas servirão como base para as caixas que serão empilhadas. Após, caixas do mesmo tipo são empilhadas, até que se alcance a altura máxima da torre de caixas. Os espaços vazios são preenchidos com caixas de menor tamanho, quando possível.

Para a heurística proposta, considera-se o carregamento de um único container com dimensões fixas. Não há necessidade de regras para empilhamento máximo de caixas e de peso máximo para o container, pois o peso do produto não representa perigo a estas restrições. As caixas não podem ser transportadas em pé, portanto elas não podem ser rotacionadas no eixo z, apenas nas dimensões exemplificadas na Figura 1.

Figura 1: Rotações aceitáveis para as caixas



Devido à pequena quantidade de caixas a serem utilizadas no problema, o mesmo pode ser definido como levemente heterogêneo. Os tipos diferentes de caixas estão descritos na Tabela 1. As caixas em destaque são as utilizadas na empresa, sendo que a caixa D é a que possui menor utilização, pois só transporta um tipo de produto. As demais não estão em uso.

Tabela 1: Dimensões de caixas utilizadas pela empresa

Dimensão das caixas em mm			
Caixa	**comp.**	**largura**	**Altura**
A	422	305	272
B	305	197	275
C	245	253	227
D	320	320	195
F	330	330	265
G	343	243	137
H	250	245	122
I	395	395	262
J	255	255	595
K	342	238	252

Algoritmo Genético

Tem sido um grande desafio a resolução de problemas que possuem uma complexidade muito elevada, nos quais a solução ótima se limita a um pequeno conjunto de dados, ao invés de utilizar todo seu espaço de busca. Estes problemas possuem representações

matemáticas complexas, sendo difícil sua portabilidade para um algoritmo convencional que comporte todo conjunto de soluções existentes em situações reais. Por exemplo, o problema do caixeiro viajante (*Traveling Salesman Problem* - TSP) que, dado um número finito de "cidades" e o custo da viagem entre cada par de cidades procura encontrar o caminho de menor custo visitando todas as cidades, uma única vez e voltando ao ponto inicial (MICHALEWICZ, 1997). Com 4 cidades, o TSP tem um espaço de busca de 24 soluções possíveis. Com o aumento para 5 ou 6 cidades, tem-se o espaço de busca ampliado para 120 e 720 possíveis soluções, respectivamente, tornando-se impraticável a utilização de algoritmos que buscam uma solução exaustivamente (conhecido como método de força bruta) para resolver este problema com uma grande quantidade de cidades.

Com o objetivo de encontrar soluções para problemas complexos em um tempo computacional aceitável, tem se destacado a metaheurística Algoritmos Genéticos (AGs), criada por John Holland (1975) e disseminada por seu aluno David Goldberg (1989), devido à sua facilidade em resolver problemas relacionados à otimização, tais como o TSP, problemas relacionados a cálculo de rota e problemas relacionados a corte e empacotamento.

Os AGs são algoritmos de busca paralela, baseados em um processo regido por seleção natural, no qual populações competem umas com as outras para se converterem em progenitores, pois "somente os mais fortes sobrevivem", assim descrito por Charles Darwin (1859). Embora à primeira vista este método pareça complexo, devido ao AG utilizar conceitos biológicos, o que se observa é totalmente o contrário, já que esta metaheurística utiliza basicamente operações de manipulação de *strings*.

Esta metaheurística é cercada por termos com origem na genética. O conjunto de soluções potenciais de um problema é tratado como uma população de indivíduos, no qual cada indivíduo é representado por um cromossomo ou genoma, definido por uma estrutura de dados apropriada para cada problema. Cada cromossomo possui vários genes no qual cada gene (ou conjunto deles) contém pedaços da solução. O funcionamento básico de um algoritmo genético é iniciado com a geração de uma população inicial de indivíduos. A partir desta população, são realizados cruzamentos (crossover) e mutações entre os indivíduos. A cada nova geração, os indivíduos mais bem adaptados ao meio sobrevivem e tem maior chance de passar seus genes adiante, assim como na teoria da seleção natural.

Como condição de parada para o AG, podem ser usados:

- número máximo de gerações: é determinado um número máximo de gerações que, ao atingir essa geração, o melhor indivíduo é tido como solução;
- porcentagem de indivíduos iguais na população: se um determinado percentual da população tem os mesmos genes, estes indivíduos são considerados como a melhor solução;
- melhores indivíduos das últimas *n* gerações não se alteraram: durante *n* gerações, os *n* melhores indivíduos da população não sofreram alteração, indicando uma estagnação no processo evolutivo.

Um dos recursos altamente recomendados para a seleção natural é o elitismo, que mantêm os melhores cromossomos da geração anterior na geração atual. Esta pequena alteração pode aumentar muito o desempenho do AG. Além do elitismo, pode ser utilizado o salvacionismo, ou seja, apenas o melhor indivíduo é passado para a próxima geração, ou o método aleatório, no qual os indivíduos que farão parte da população futura são sorteados por um método randômico.

■ *Simulated Annealing*

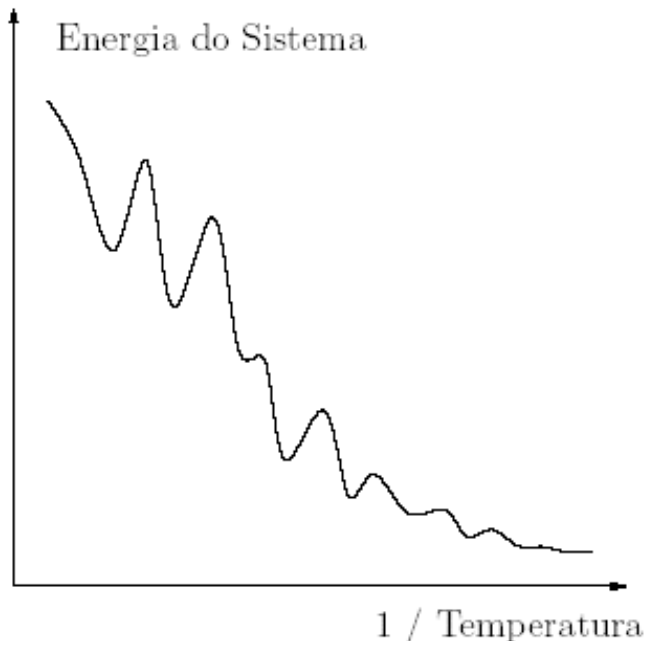
Outra metodologia possível para a resolução de problemas permutacionais, ou seja, os algoritmos "não convencionais" que são específicos para otimização, é o *Simulated Annealing* (SA), que apresenta bons resultados com soluções em que o espaço de busca contém vários ótimos locais.

Annealing, ou Recozimento, é o processo de aquecimento de sólidos (geralmente utilizado em metais) até seu ponto de fusão, seguido de uma lenta e gradativa diminuição da temperatura até o enrijecimento do material. O lento processo de arrefecimento se mostra necessário para que se mantenha um equilíbrio térmico, no qual os átomos encontrarão tempo suficiente para se organizarem em estruturas uniformes com energia mínima, gerando assim um cristal perfeito.

O SA foi pesquisado pela primeira vez por Kirkpatrick et al (1983), o qual encontrou as semelhanças entre o processo da termodinâmica de *annealing* e o processo de otimização de problemas de pesquisa operacional. O algoritmo se baseia na organização de átomos de um sólido, pois, quando a

temperatura está alta, os átomos (trechos da solução) se movem livremente e tem uma alta probabilidade que seu movimento aumente a energia total do sistema. Com a diminuição da temperatura, os átomos se movem gradualmente em direção a uma estrutura regular, diminuindo a probabilidade de alterações que aumente a energia, como ilustra a Figura 2.

Figura 2: Estabilização da energia com a diminuição da temperatura.



Kirkpatrick et al (1983) foi baseado em Metropolis et al (1953), que é um algoritmo simples para simular a evolução de um sólido em uma caldeira na direção do equilíbrio térmico.

Heurísticas Desenvolvidas

Para resolver o problema abordado, foi utilizada uma heurística construtiva baseada na construção de torres[1] que minimizam o espaço não utilizado em cada torre gerada. Pode-se considerar que o espaço de uma torre é dado pelo volume calculado através da área da caixa da base, comprimento e largura, sendo que a altura é a permitida pelo container. Uma vez construídas as torres, essas são ordenadas internamente no container de inúmeras maneiras. Cada ordenação de torres forma uma solução diferente que é otimizada utilizando um algoritmo baseado em Algoritmos Genéticos e outro em *Simulated Annealing*. O algoritmo de definição de quantidade de caixas, a heurística construtiva de torres e aquela que gera a solução inicial, estão descritas nas seções 6.1, 6.2 e 6.3, respectivamente e foram, originalmente, apresentadas em (CEZAR, 2003). As metodologias de busca por solução implementadas, Algoritmos Genéticos (AG) e *Simulated Annealing*, estão em 6.4 e 6.5, respectivamente.

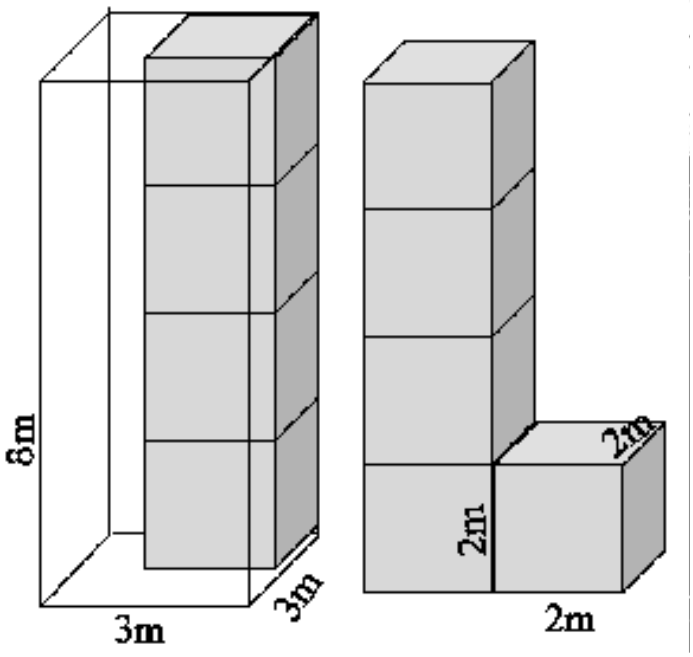
Algoritmo de definição de quantidade de caixas

O problema inicial é a definição da quantidade de caixas a ser carregada no container, pois o pedido é feito baseado na porcentagem de produtos que cabem dentro de cada caixa, e não na quantidade de caixas. O primeiro passo é utilizar uma simples regra de três para que seja estabelecida uma proporção entre os diversos produtos a serem carregados. Por exemplo, em uma caixa **A** cabem 400 itens do tipo 1 e em uma caixa **C** cabem 40 itens do tipo 2. Um pedido divido em cinqüenta por cento do carregamento de cada tipo de produto, terá dez caixas **C** para cada caixa **A**, mantendo a proporção de cinqüenta por cento na quantidade de produtos.

Para definir a quantidade de caixas que serão carregadas dentro do container, é utilizado um cálculo simples utilizando o volume das caixas e o volume total do container. Ao dividir o volume de uma caixa pelo volume do container, obtém-se o volume relativo da caixa ao container. Logo, ao multiplicar o volume relativo de uma caixa por sua relação de proporção e, ao somar o resultado para todas as caixas, obtém-se o volume relativo que um "jogo" de caixas possui com o container. Jogo significa todos os tipos de caixas do pedido, na quantidade para manter a proporção dos itens encomendados. Por fim, basta dividir o volume total do container pelo volume do jogo. Desta forma, sabe-se quantos jogos podem ser inseridos no container. Se ainda houver espaço para alocar mais caixas, insere-se caixas do item que possui a maior porcentagem de encomenda enquanto for possível.

No entanto, este método é falho, devido à quantidade de caixas resultantes não servir fisicamente no container. Por exemplo, supõe-se um container com largura de 3m, comprimento de 3m e altura de 8m e, um conjunto de caixas com largura igual a 2m, comprimento de 2m e altura de 2m. O volume do container é de 72m³ e o volume de cada caixa é de 8m³. Com o cálculo utilizado, seria possível o carregamento de 9 caixas, mas fisicamente é possível a colocação de somente 4 caixas, como ilustrado na Figura 3.

Figura 3: Exemplo de volumes iguais, mas dimensões diferentes.



Heurística construtiva de torres

Algoritmo 2 Heurística construtiva de torres

i ← 0; **// Contador de torres geradas**
j ← 0; **// Contador de itens num pe dido**
Recebe pedido do usuário
Enquanto houver item no pedido **faça**
j ← j + 1
Define quantidade de caixas C(j)
Ordena C **por área da base**
Enquanto houver caixa em C **faça**
i ← i + 1
Insere caixa maior em C **na base da torre** P(i)

Enquanto houver caixas em C e esta servir em cima da caixa base na torre e não atingir a altura máxima **faça**
Insere caixa na torre P(i)
Fim Enquanto

Ao atingir altura máxima para toda caixa em C
Se couber mais uma caixa na torre e não desrespeitar o espaço restante da altura do container **então**
Insere caixa na torre P(i)
Fim para
Fim Enquanto

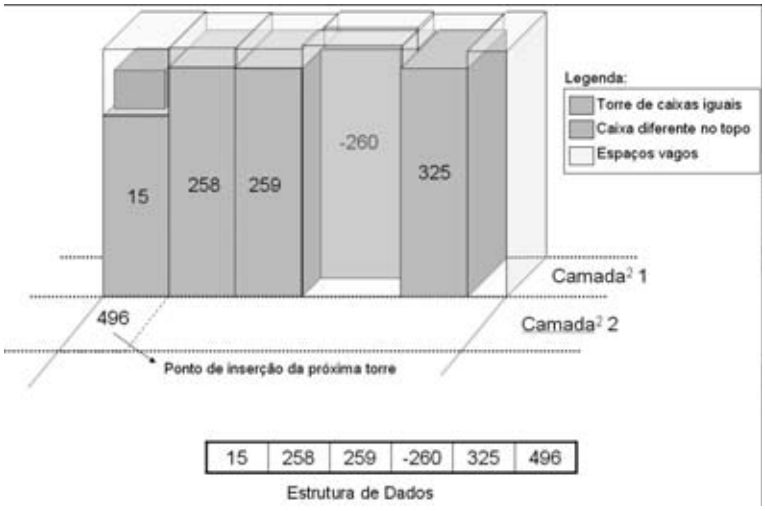
Após o processo que define a quantidade de caixas a serem carregadas, as caixas são organizadas em torres identificadas (com numeração seqüencial) e estáveis. Ou seja, o método para construção das torres seleciona a maior caixa a ser organizada. Caixas do mesmo tipo são empilhadas até a altura máxima que uma torre com o mesmo tipo de caixa pode atingir, e após, são feitas tentativas para que sejam colocadas caixas que sirvam no espaço que seria desperdiçado. Quando não há mais caixas do mesmo tipo para continuar a construção da torre, a próxima caixa a ser inserida deve, necessariamente, possuir uma área da base menor ou igual àquela que a suportará.

Heurística construtiva da solução inicial

Com as torres previamente definidas é utilizada uma heurística para a construção da solução inicial, que é uma representação de um container preenchido.

O processo de preenchimento do container é iniciado através da inserção da torre de maior base, seguida de outras torres tendo a mesma caixa como base. Esta medida visa obter a melhor ocupação dentro de cada camada[2] de torres. A alocação das torres é contínua até que todo o container seja preenchido, respeitando suas dimensões. Caso não haja mais torres de mesma caixa como base, a torre com a caixa base de segunda maior área será a escolhida, e assim sucessivamente, até que sejam alocadas todas as torres de caixas, ou que o container seja totalmente preenchido.

Figura 4: Estrutura de Dados.



A estrutura de dados utilizada para a

representação de uma solução é ilustrada na Figura 4, cujos números identificam as torres utilizadas na solução. O número negativo identifica uma torre que teve a orientação da base invertida, como se a mesma fosse girada em noventa graus.

Algoritmo 3 Heurística construtiva de solução inicial

```
Sendo T o conjunto de torres
Sendo s a solução inicial
Sendo m o número de caixas na solução
Sendo n a posição atual da torre escolhida
m ← 1
Enquanto n > 0 OU couber torres no container faça
   n =  Procura a torre com a caixa de maior base
   s(m) ← T(n)
   m ← m + 1
   Retira T(n) de T
    Enquanto houver torres com a mesma caixa-base em T igual a caixa-base da torre em
      s(m-1) OU a largura da camada extrapolar a largura do container faça
          n = Encontre a torre com a mesma caixa-base que em s(m-1)
          s(m) ← T(n)
          m ← m + 1
          Retira T(n) de T

          Atualiza a largura atual da camada
          Se a largura da camada > largura do container então
                T(n) ← s(m)
                m ← m – 1
                Retira s(m) de s
                Extrapolou a largura
          Fim Se
    Fim Enquanto
    Se Extrapolou a largura então
          Começar nova camada
    Fim Se
Fim Enquanto
```

Algoritmo Genético desenvolvido

Para utilizar um algoritmo genético para resolver esse problema é necessário criar um cromossomo que represente corretamente a solução de um container preenchido. Neste caso, o cromossomo utilizado foi composto da identificação numérica de cada torre e de um sinal positivo ou negativo. Esta alternância de sinais indica a orientação da caixa base da torre; positivo indica que a torre será colocada na posição original (comprimento x largura), e negativo indica que houve uma rotação de 90º no eixo (x,y), alterando a posição da caixa base (largura x altura). A Figura 4 ilustra um exemplo de representação cromossômica utilizada no algoritmo, conforme a estrutura de dados apresentada.

Por se tratar de um problema permutacional, o algoritmo genético desenvolvido utiliza o operador de *crossover* *Cicle Crossover* - CX, com taxa de crossover de 50%, a mutação é realizada por meio da inversão de elementos entre dois pontos do cromossomo, escolhidos aleatoriamente (taxa de mutação de 20%); a escolha das soluções a serem combinadas é feita por meio da roleta; é inserida diversificação através da geração de soluções aleatórias e, por fim, a escolha das soluções para a nova geração utiliza o elitismo. A população inicial é de 100 elementos (POP).

Algoritmo 4 Pseudocódigo do *Algoritmo Genético*

```
Sendo POP o número de soluções na população
POP ← número de elementos da população

Para  t ← 1 até POP,  // Geração da população inicial
    Gere um indivíduo P(t) através da organização aleatória das torres
        Avalie a função de fitness fit(P(t))
Fim para

t ← 0  // Contador de iterações do AG
Enquanto não atingir condição de parada,
   t ← t + 1
   P(t) ← Crossover CX (P(t-1))
   P(t) ← Mutação (P(t))
   Avalia fitness fit (P(t))
   Aplica seleção natural utilizando o método elitista
Fim Enquanto
```

A função de *fitness* (objetivo) consiste no volume total do container menos a somatória de todos os espaços não preenchidos, ou seja, os espaços vagos entre uma camada e outra, os espaços restantes nas torres e nas laterais do container. Desta forma, a função objetivo utilizada para avaliar uma solução, mostra a quantidade total de ocupação do container, isto é, qual a porcentagem do volume disponível do container que foi utilizado. Neste caso, o objetivo do algoritmo é maximizar a quantidade de espaço ocupado.

Simulated Annealing desenvolvido

O SA desenvolvido possui as características tradicionais de um método de busca do tipo Monte Carlo (METROPOLIS, ULAM, 1949), ou seja, inicia-se com uma dada solução inicial a qual será submetida a um mecanismo de vizinhança, visando encontrar uma solução de melhor desempenho. No entanto, se nenhuma solução vizinha for melhor, aceita-se uma solução com uma probabilidade de piora. Esta estratégia faz com que a busca supere ótimos locais e se dirija a um novo ponto do espaço de busca das soluções.

A estratégia de vizinhança utilizada é variável (VNS - *Variable Neighborhood Search*) (SOUZA, 2004), a qual utiliza a troca sistemática da estrutura de vizinhança para explorar o espaço de busca. Foram utilizados três métodos como estrutura de vizinhança: troca, inserção e rotação. Dentre as estratégias de vizinhança implementadas, o SA escolhe uma aleatoriamente.

As estratégias de vizinhança consideram todas as torres que foram geradas pela heurística construtiva de torres, inclusive aquelas que por falta de espaço no container, não foram utilizadas pela heurística construtiva de solução inicial. Esta possibilidade gera uma perturbação maior, o que possibilita uma melhor exploração do espaço de busca.

Na estratégia que utiliza a troca, são selecionados dois grupos aleatórios de torres e suas posições são trocadas. Grupos de torres que ficaram fora da solução podem ser selecionados, e ter suas posições trocadas com torres que faziam parte da solução, gerando uma maior perturbação.

A estratégia de inserção seleciona um grupo seqüencial de torres e os insere em um local aleatório, alterando a disposição do container.

Já a rotação não move nenhuma torre de lugar. Esta estratégia somente rotaciona um número aleatório de torres (tanto de quantidade de torres como de quais torres), transformando a base da torre de largura x comprimento para comprimento x largura.

A função objetivo consiste no volume total do container menos a somatória de todos os espaços não preenchidos, ou seja, os espaços vagos entre uma camada e outra, os espaços restantes nas torres e nas laterais do container. Desta forma, a função objetivo utilizada para avaliar uma solução, mostra a quantidade total de ocupação do container, isto é, qual a porcentagem do volume disponível do container que foi utilizado. Neste caso, o objetivo do algoritmo é maximizar a quantidade de espaço ocupado.

A temperatura inicial T_0 utilizada no algoritmo foi de 1500, e a temperatura mínima de 0,001. Para o fator de diminuição da temperatura foram utilizados dois parâmetros: 0,8 e 0,4, sendo que os testes realizados utilizando o segundo valor apresentaram uma grande melhora em relação à primeira. O número máximo de iterações foi 100. Os parâmetros utilizados foram obtidos através de métodos empíricos (não estatísticos).

Algoritmo 5 Pseudocódigo do *Simulated Annealing*

```
Sendo s a representação de uma solução
Sendo s* a melhor solução encontrada
Sendo T a temperatura atual
Sendo i a iteração atual da temperatura T
Sendo rand um número randômico entre 0 e 1
Sendo á fator de redução da temperatura T
s ← solução inicial
s* ← s
i ← 0
Se Ä < 0 então
   s ← s'
   Se f(s') < f(s*) então
         s* ← s'
   Fim Se
Senão   Se rand < e^-Ä então
   s ← s'
Fim Se
   Fim Enquanto
   T ← α x T
   i ← 0
Fim Enquanto
```

Resultados

Os resultados computacionais mostrados são referentes a simulações de pedidos que poderiam ter sido efetuados por clientes da empresa. Para efeito de comparação foi executado o programa desenvolvido para alguns problemas testes, sendo que o resultado obtido foi analisado em conjunto com as soluções de ambas as metaheurísticas. Para uma instância, foi possível comparar com o resultado que teria sido colocado em prática pela empresa, o qual foi fornecido por Vera Hamerski.

Os testes foram realizados em um microcomputador AMD Athlon XP 2800+ (2600 Mhz) e 1,0 Gb memória RAM. A implementação computacional foi feita em Visual Studio.net utilizando a

linguagem Visual Basic.net, e a interface gráfica foi feita utilizando MatLab 5.2.

Para os testes efetuados, foi considerado um único container de vinte pés (5660mm de comprimento devido a uma margem de segurança exigida pela empresa, 2300mm de largura e 2395mm de altura).

As instâncias utilizadas para o teste dos algoritmos são mostradas na Tabela 2. Para cada instância foram testadas cinco diferentes combinações de porcentagem de itens a serem pedidos. Foram utilizados somente três tipos diferentes de caixas, pois este é o número máximo de caixas diferentes num pedido da empresa considerada neste trabalho. Além disso, a empresa aceita pequenas variações na porcentagem dos produtos solicitados. Desta forma, na implementação do AG e do SA não houve a preocupação em alocar caixas na porcentagem exata, mas sim numa muito próxima a solicitada, pois a quantidade de caixas que não foram utilizadas é insignificante.

Os testes realizados mostraram que o desempenho de ambos os algoritmos desenvolvidos é superior à técnica que a empresa emprega, para determinar a quantidade de produtos que serão produzidos para atender ao pedido do cliente. O resultado obtido por intermédio do algoritmo baseado em *Simulated Annealing*, para a instância cuja solução usada pela empresa é conhecida, forneceu uma melhoria média de 17% de caixas a mais alocadas. O Algoritmo Genético obteve uma melhora de 10% em relação à quantidade de caixas alocadas, para essa mesma instância. Os demais testes mostram a ocupação máxima obtida pela solução.

As informações referentes aos resultados obtidos pelos algoritmos em cada instância são mostradas na Tabela 3. É importante observar que o ganho calculado é com relação ao resultado obtido com o AG, ou seja, verifica-se a melhoria que o SA obteve para as mesmas instâncias se comparado com o AG.

Foram efetuados cinco testes para cada instância e para cada porcentagem de produto sugerida, sendo que na Tabela 3 são apresentadas as médias dos testes realizados. O ganho apresentado é sobre a média apresentada.

No resultado apresentado na Figura 5 foram adicionadas 199 caixas a mais daquele que seria proposto pela empresa, através do algoritmo genético desenvolvido. Na Figura 6, observa-se o resultado obtido, para a mesma instância, pelo algoritmo S*imulated Annealing* proposto. Pelos cálculos efetuados através do sistema atual adotado pela empresa, seriam alocadas 2153 caixas, enquanto que o algoritmo genético alocou 2352 caixas. Já o S*imulated Annealing* se mostrou mais eficaz, alocando 2657 caixas para o mesmo exemplo.

O tempo computacional não foi considerado, pois ambos os algoritmos apresentaram tempos semelhantes de execução.

Tabela 2: Instâncias utilizadas para teste

Instância 1			Instância 2			Instância 3			Instância 4		
Caixa	Produtos por caixa	%	Caixa	Produtos por caixa	%	Caixa	Produtos por caixa	%	Caixa	Produtos por caixa	%
A	400	30	A	240	50	F	16	25	G	36	30
C	36	20	B	400	20	G	42	25	J	5	30
H	60	50	C	400	30	I	60	50	K	48	40

Tabela 3: Resultados obtidos após os testes com cada instância

	Instâncias											
	1			*2*			*3*			*4*		
%	***AG***	***SA***	***Ganho***	***AG***	***SA***	***Ganho***	***AG***	***SA***	***Ganho***	***AG***	***SA***	***Ganho***
1	90,44	94,33	4,30	82,87	90,13	8,76	92,18	94,99	3,05	90,27	94,01	4,15
2	91,37	94,41	3,32	83,16	89,79	7,96	92,19	94,82	2,85	89,68	91,77	2,33
3	88,47	93,55	5,75	85,55	90,42	5,69	92,61	94,28	1,81	90,94	94,50	3,92
4	92,05	94,66	2,84	83,45	89,10	6,77	92,41	94,45	2,21	89,68	91,94	2,51
5	88,87	91,75	3,24	85,28	88,81	4,15	91,87	92,82	1,03	90,62	91,57	90,28
Média	90,24	**93,74**	**3,89**	84,06	**89,65**	**6,67**	92,25	**94,27**	**2,19**	90,24	**92,76**	**20,64**

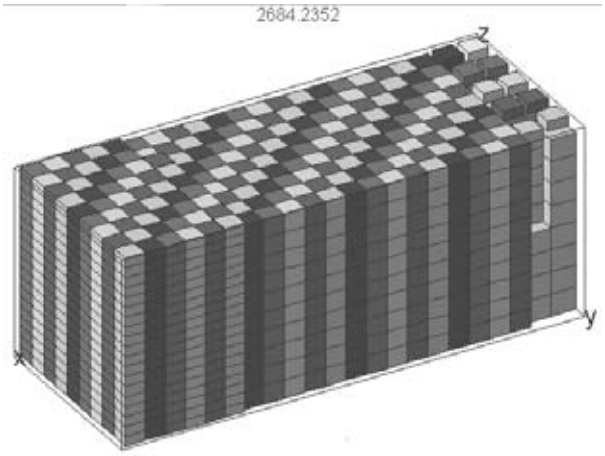


Figura 5: Solução obtida com AG

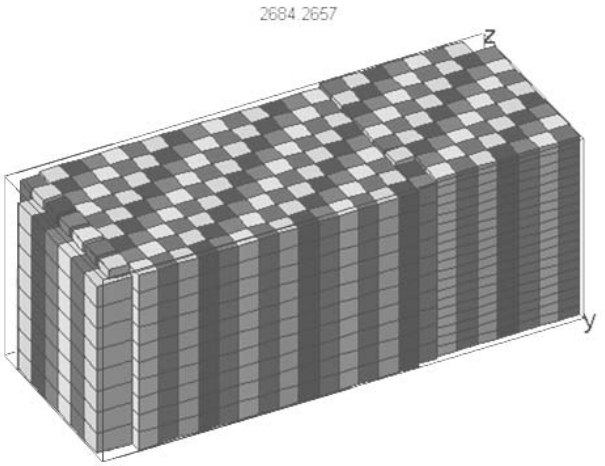


Figura 6: Solução obtida com *SA*

Conclusão

O trabalho desenvolvido procura uma alternativa para otimizar o processo de embarque da produção em container, conseqüentemente, da ordem de produção dos pedidos que vem do exterior para a empresa que inspirou essa pesquisa.

Este trabalho teve como objetivo utilizar um novo método para resolver o problema específico apresentado. A metaheurística S*imulated Annealing* se mostrou eficaz para a resolução de problemas combinatórios tal qual o problema do container, cuja melhoria da solução deste caso específico foi superior ao Algoritmo Genético implementado.

A estratégia de construir torres para alocar as caixas e então trabalhar com o S*imulated Annealing* ou com o Algoritmo Genético para encontrar a melhor disposição das torres no container, são duas possíveis estratégias para resolver esse problema. Ambas se mostraram eficazes, por não haver uma grande discrepância entre as diferentes caixas, o que evitou a ocorrência de grandes espaços não preenchidos entre as camadas, assim como dentro de cada torre. No entanto, percebe-se que minimizar os espaços vazios que as torres deixam ao serem construídas influi diretamente na qualidade da solução. Desta forma, novas heurísticas a serem desenvolvidas podem considerar este aspecto.

Para a continuação deste trabalho, fica a proposta de utilizar parâmetros baseados em métodos estatísticos e novas estruturas de vizinhanças, que possibilitem diferentes perturbações na solução. Pode-se também mudar a estratégia de alocação que ao invés de ser baseada em torres de caixas, seja baseada em blocos de caixas.

Referências Bibliográficas

BORTFELDT, A., GEHRING, H. A hybrid genetic algorithm for the container loading problem. *European Journal of Operational Research*, 131, pp 143-161, 2001.

CEZAR, D. F. O. *Otimização de espaço em container*. Monografia (Bacharelado em Ciência da Computação) - Faculdade de Ciências Administrativas das Faculdades de Valinhos, Valinhos, 2003.

DARWIN, C. On the Origin of Species, John Murray, 1859.

ELEY, M. Solving loading problems by block arrangement. *European Journal of Operational Research*,141, pp 393-409, 2002.

GEORGE J. A., ROBINSON D. F. A heuristic for packing boxes into a container, Computer and Operations Research, 7, 147-156, 1980.

GOLDBERG, D. E. Genetic algorithms in search, optimization and machine learning. Addison-Wesley, 1989.

HOLLAND, J.H. Adaptation in natural and artificial systems. MIT Press, 1975.

INGBER, L. Simulated annealing: Practice versus theory. *Mathl. Comput. Modelling,* v.18, n.11,29-57, 1993.

KIRKPATRICK, C; GELATT, C. D.; VECCHI, M. P. Optimization by simulated annealing. *Science*, v. 220, n. 4598:671-690, 1983.

METOPOLIS,N. *et al.* Equation of State Calculations by Fast Computing Machines, *J. Chem. Phys.*, v. 21, n. 6:1087-1092, 1953.

METROPOLIS, N., ULAM, S. The Monte Carlo Method, *Journal of the American Statistical Association*, v44, p. 335, 1949.

MICHALEWICZ, Z. Genetic Algorithms + Data Structure = Evolution Programs. Springer Verlag, 1997.

PISINGER, D. Heuristic for the container loading problem. *European journal of operational research*,141, pp. 382-392, 2000.
PUREZA, V; MORABITO, R. Uma heurística de busca tabu simples para o problema de carregamento de paletes do produtos. *Pesquisa Operacional,* v.23, n.2:359-378, 2003.
SOUZA, M.J.F., et al. Metaheurísticas aplicadas ao problema de programação de tripulações no sistema de transporte público. *TEMA Tend. Mat. Apl. Comput*., 5, n.2: 357-368, 2004.

Notas

[1] Uma torre é uma pilha de caixas iguais ou não. A base da pilha contém a caixa com maior dimensão em comprimento e largura, no caso de caixas heterogêneas, buscando equilíbrio para a torre.
[2] Uma camada é uma fila de torres, lado a lado, até atingir a largura do container.