Universidade Estadual de Campinas

INSTITUTO DE MATEMÁTICA, ESTATÍSTICA E COMPUTAÇÃO CIENTÍFICA

Departamento de Matemática Aplicada

Dissertação de Mestrado

Uma aplicação do algoritmo Colônia de Formigas no problema de Corte Ordenado

por

Fernanda Ledo Marciniuk †

Mestrado em Matemática Aplicada - Campinas - SP

Orientador: Prof. Dr. Antônio Carlos Moretti - IMECC/UNICAMP

Co-orientador: Prof. Dr. Luís Leduíno de Salles Neto - UNIFESP

†Este trabalho contou com apoio financeiro do CNPq.

UMA APLICAÇÃO DO ALGORITMO COLÔNIA DE FORMIGAS NO PROBLEMA DE CORTE ORDENADO

Este exemplar corresponde à redação final da dissertação devidamente corrigida e defendida por Fernanda Ledo Marciniuk e aprovada pela comissão julgadora.

Campinas, 23 de abril de 2010

Prof. Dr:Antônio Carlos Moretti
IMECC/Unicamp
Orientador

Prof. Dr: Luís Leduíno de Salles Neto
UNIFESP
Co-orientador

Banca Examinadora:

1. Antônio Carlos Moretti – IMECC/UNICAMP
2. Prof. Dr. Romis Ribeiro de Faissol Attux – FEEC/UNICAMP
3. Profa. Dra. Márcia Aparecida Gomes Ruggiero – IMECC/UNICAMP

Dissertação apresentada ao Instituto de Matemática, Estatística e Computação Científica, UNICAMP, como requisito parcial para obtenção do Título de MESTRE em Matemática Aplicada

FICHA CATALOGRÁFICA ELABORADA PELA BIBLIOTECA DO IMECC DA UNICAMP
Bibliotecária: Crisllene Queiroz Custódio – CRB8 / 7966

Marciniuk, Fernanda Ledo
M332a Uma aplicação do algoritmo colônia de formigas no problema de corte ordenado / Fernanda Ledo Marciniuk -- Campinas, [S.P. : s.n.], 2010.

Orientadores: Antônio Carlos Moretti ; Luis Leduíno de Salles Neto
Dissertação (Mestrado) - Universidade Estadual de Campinas, Instituto de Matemática, Estatística e Computação Científica.

1. Otimização matemática. 2. Formiga - Comportamento - Modelos matemáticos. 3. Problema do corte de estoque. 4. Algoritmo da formiga. 5. Metaheurística. I. Moretti, Antônio Carlos. II. Salles Neto, Luiz Leduíno de. III. Universidade Estadual de Campinas. Instituto de Matemática, Estatística e Computação Científica. IV. Título.

Título em inglês: Ant colony optimization for the ordered cutting stock problem.

Palavras-chave em inglês (Keywords): 1. Mathematical optimization. 2. Ants - behavior - Mathematical models. 3. Cutting stock problem. 4. Ant algorithms. 5. Metaheuristics.

Área de concentração: Pesquisa operacional

Titulação: Mestre em Matemática Aplicada

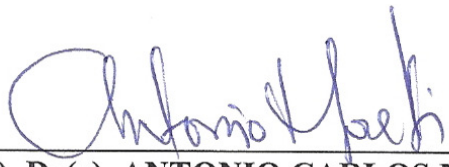
Banca examinadora: Prof. Dr. Antônio Carlos Moretti (IMECC-Unicamp)
Prof. Dr. Romis Ribeiro de Faissol Attux (FEEC-Unicamp)
Profa. Dra. Márcia Aparecida Gomes Ruggiero (IMECC-Unicamp)

Data da defesa: 08/03/2010

Programa de Pós-Graduação: Mestrado em Matemática Aplicada

Dissertação de Mestrado defendida em 08 de março de 2010 e aprovada

Pela Banca Examinadora composta pelos Profs. Drs.



Prof.(a). Dr(a). ANTONIO CARLOS MORETTI

Prof. (a). Dr (a). MARCIA APARECIDA GOMES RUGGIERO

Prof. (a). Dr (a). ROMIS RIBEIRO DE FAISSOL ATTUX

Agradecimentos

Quero agradecer a todos que, direta ou indiretamente, participaram da construção deste trabalho.

A minha família pela paciência, amor e a ajuda financeira.

Aos meus orientadores, Moretti e Leduíno, pela oportunidade, paciência e motivação, os quais foram de extrema importância para a realização deste trabalho.

Aos professores Romis e Márcia participantes da banca de defesa, pelas importantes contribuições.

Ao Rodrigo Golfeto pela paciência e ajuda na construção dos testes para a conclusão deste trabalho.

Aos meus colegas do IMECC, e um agradecimento especial à Juliana pela amizade e ajuda durante os estudos e ao Emerson pelo apoio e companheirismo durante toda essa fase.

Aos meus colegas de república: Cabelo, Cris, Karina, Lettícia, Aninha e Miguel, pelas risadas, conselhos, e apoio nas horas mais difíceis.

Quero agradecer ao Gustavo por todo carinho, pelos puxões de orelha e principalmente pelo apoio moral para a finalização deste trabalho.

Um agradecimento muito especial à Letícia, minha melhor amiga e irmã, pelos conselhos, pelas festas, pelas confidências e pelas broncas de "irmã mais velha".

A Cidinha, Ednaldo, Tânia e Lívia por estarem sempre dispostos a solucionar os meus vários problemas burocráticos.

Ao CNPq pela ajuda financeira.

Resumo

O problema de corte de estoque ordenado, um problema relativamente novo na literatura, é uma adaptação do problema de corte de estoque tradicional onde algumas restrições quanto a limitação do número de ordens de produção em processamento são adicionadas.

Esta dissertação tem como objetivo estudar uma nova abordagem deste problema utilizando uma aplicação da metaheurística colônia de formigas. Esta metaheurística utiliza os princípios de auto-organização de uma população de formigas visando à resolução de problemas de otimização combinatorial.

Palavras-chave: 1. Otimização Matemática. 2. Formiga - Comportamento - Modelos Matemáticos. 3. Problema de corte de estoque. 4. Algoritmo da formiga. 5. Metaheurística.

Abstract

The Ordered Cutting Stock Problem (OCSP), a relatively recent problem in technical literarture, is a variant of the more well-known Cutting Stock Problem (CSP). This variant includes some new constraints in the mathematical formulation, regarding the number of production orders being processed simultaneously.

This work studies a new approach to solve the OCSP, applying the Ant Colony Optimization (ACO) metaheurisitic. This metaheuristic is based in the self-organizing principles that govern ant population's behaviour, solving combinatorial optimization problems.

Sumário

Introdução

Dada a competitividade crescente imposta pela atual economia global, as indústrias manufatureiras, assim como as indústrias em geral, vêm sofrendo profundas mudanças, a fim de tornar seus processos de produção cada vez mais eficientes [30]. Neste contexto, a redução de custos é um fator essencial para uma diferenciação competitiva.

O problema de corte de estoque (CSP, do inglês *Cutting Stock Problem*), um problema clássico de otimização combinatorial e bastante aplicado em indústrias manufatureiras, visa à redução de custos por meio da minimização das perdas de matérias-primas em retalhos. Consiste basicamente em determinar uma sequência ótima para o corte de matérias-primas em itens menores, atendendo a uma demanda preestabelecida.

Durante as últimas décadas, muitas ferramentas e metodologias de gestão e estratégia tiveram muito destaque em todo o mundo, tendo em vista a grande necessidade de melhorar o rendimento das linhas de produção e consequentemente minimizar os seus custos. Uma metodologia bastante difundida é a JIT (*Just in Time*) [30], que tem como objetivo principal a busca contínua pela melhoria do processo produtivo - obtida e desenvolvida principalmente através da redução dos estoques.

O *Just in Time* e outros modelos de gestão como o MRP (do inglês, M*aterial Requirements Planning*), o MRP II (do inglês, *Manufacturing Resources Planning*) e o CRP (do inglês, *Capacity Resources Planning*) têm o estoque como uma de suas principais restrições. O CSP tradicional, portanto, não é um modelo apropriado para tais processos [27], pois ele processa todas as ordens de produção ao mesmo tempo sem levar em consideração a triagem, armazenamento e a separação dos itens nos respectivos lotes, gerando estoques desnecessários.

Para reduzir essa geração de estoques, uma adaptação ao CSP que já vem sendo

utilizada por algumas empresas, é processar todos os itens de cada ordem de produção em sequência. O problema de corte ordenado (OCSP, do inglês *Ordered Cutting Stock Problem*) apresentado por Ragsdale e Zobel [27] modela matematicamente esta solução.

De acordo com Valério de Carvalho [1], o OCSP combina aspectos do problema de corte de estoque tradicional (CSP) e do problema do caixeiro viajante (TSP, do inglês *Travelling Salesman Problem*), um problema de minimização de rotas, que em geral se aplicam heurísticas para sua solução. Para maiores detalhes sobre tais heurísticas diversos autores podem ser indicados: Renaud *et al.*([28]), Johnson e McGeoch ([18]), dentre outros.

Objetivos

Este trabalho tem por objetivo estudar uma aplicação do algoritmo Colônia de Formigas no problema de corte de estoque ordenado (OCSP).

Estrutura do trabalho

Este trabalho está estruturado da seguinte forma:

- No Capítulo 1 são apresentados os conceitos sobre o problema de corte tradicional e o problema de corte ordenado. Neste capítulo é feita uma revisão bibliográfica dos trabalhos já existentes na literatura.
- No Capítulo 2 são apresentados os conceitos básicos sobre a metaheurística Colônia de Formigas e uma aplicação, já conhecida na literatura, do algoritmo em um problema de corte tradicional.
- O Capitulo 3 apresenta como a metaheurística Colônia de Formigas foi adaptada para a aplicação do problema de corte ordenado.
- No Capítulo 4 são apresentados os resultados e análises dos testes computacionais realizados.

- No Capítulo 5 são feitas as considerações finais deste trabalho e algumas propostas para trabalhos futuros.

Capítulo 1

O Problema de Corte Ordenado Unidimensional

Dentro de um processo de produção, devido à necessidade de redução de custos, a aplicação de problemas de corte é crescente. Com o mesmo objetivo de minimizar custos, o problema vem sendo aprimorado e ganhando novas restrições, de acordo com a necessidade de cada tipo de processo, por exemplo, a limitação quanto ao número de lotes (ordens de produção) em produção em um dado momento. Atualmente, muitas pesquisas sobre este problema de corte de estoque com restrições adicionais referentes ao número de lotes em processo vêm sendo feitas. Para maiores detalhes sobre tais pesquisas, diversos autores podem ser indicados: Alves e Carvalho ([1]),Yanasse ([33]), Linhares e Yanasse ([25]), Rasgsdale e Zobel ([27]).

Neste capítulo apresentaremos o problema de corte de estoque ordenado, o qual restringe o número de lotes em processo a um. Este problema foi introduzido em 2004 por Ragsdale e Zobel [27] com o intuito de resolver um problema prático encontrado em uma indústria manufatureira produtora de portas e janelas.

1.1 Problema de Corte Unidimensional

O problema de corte de estoque unidimensional, segundo a tipologia de Dyckhoff [6] é classificado como 1/V/I/R, que representa: Dimensão/ Forma de Alocação das Unidades/

Sortimento das Unidades Grandes/ Sortimento das Unidades Pequenas.

- 1 - Unidimensional.
- V - Seleção de Unidades Grandes.
- I - Unidades de tamanhos iguais.
- R - Muitas unidades de poucos tamanhos diferentes.

O problema pode ser definido como será descrito a seguir:

Dada uma quantidade suficiente de objetos em estoque de comprimento W e um conjunto de m diferentes itens com tamanhos w_i tal que $w_i < W$ para $i = 1, ..., m$, atendendo a uma demanda d_i, para $i = 1, ..., m$, o problema consiste em determinar a frequência de cada padrão de corte, atendendo toda a demanda e minimizando perdas. Um padrão de corte é a maneira como um objeto de tamanho W é cortado para a produção dos itens demandados.

O modelo para o problema de corte unidimensional pode ser apresentado da seguinte forma:

$$\begin{aligned} \text{Minimizar} \quad & c_1 \sum_{j=1}^{n} T_j x_j \\ \text{sujeito a:} \quad & \textstyle\sum_{j=1}^{n} a_{ij} x_j = d_i \quad i = 1, ..., m \\ & x_j \in \mathbb{N} \qquad\qquad j = 1, ..., n. \end{aligned} \tag{1.1}$$

onde:

n - quantidade de padrões de corte (possíveis);

c_1 - custo por metro desperdiçado;

x_j - quantidade de bobinas processadas com o padrão de corte j;

$T_j = W - \sum_{i=1}^{m} a_{ij} w_i$ - perda em metros em cada padrão j;

a_{ij} - número de itens do tipo i no padrão j;

d_i - número de itens do tipo i demandados.

Outros critérios, tais como, minimizar o número de objetos processados, setup, número máximo de ordens de produção abertas durante o processo ([26],[25],[33],[27]), podem ser definidos, sem perda de generalidade. Considere, por exemplo, o modelo para o problema de corte cuja função objetivo é minimizar o número de objetos processados:

$$\begin{aligned} &\text{Minimizar} \quad c\sum_{j=1}^{n} x_j \\ &\text{sujeito a:} \quad \sum_{j=1}^{n} a_{ij}x_j = d_i, \quad i = 1, ..., m. \\ &\qquad\qquad x_j \in \mathbb{N}, \qquad\qquad j = 1, ..., n. \end{aligned} \tag{1.2}$$

Se existir uma exigência sobre o atendimento exato da demanda, isto é, se o excesso de produção for considerado desperdício, minimizar o número de objetos processados é equivalente à minimizar as perdas [29].

Os estudos relacionados aos modelos matemáticos para o problema de corte tiveram início na década de 40, pelo matemático e economista russo Kantorovich [19]. Em 1961, Gilmore e Gomory apresentaram um dos principais métodos de resolução para o problema de corte, o método com técnicas de geração de colunas [11]. A partir de então, este problema vem sendo bastante estudado, em grande parte graças à sua enorme aplicabilidade e também à sua dificuldade de resolução.

1.2 Métodos para a Resolução do Problema de corte unidimensional

O problema de corte pertence à classe de problemas NP-completos e, dessa forma, não é viável produzir soluções exatas em tempos computacionais razoáveis. Em razão disso, o uso de heurísticas para sua resolução tornou-se uma opção interessante. Existe na literatura um grande número de trabalhos referentes à aplicação de heurísticas no problema

de corte unidimensional, tais como: algoritmos genéticos [7],[22], algoritmo Colônia de Formigas [21], dentre outros.

Na próxima seção, apresentaremos um método tradicional de resolução para o CSP. Na Seção 2.4, também apresentaremos uma aplicação ao CSP, conhecida na literatura, utilizando o algoritmo Colônia de Formigas [21].

1.2.1 A Heurística FFD

A heurística FFD (do inglês, *First-Fit-Decreasing*) é uma heurística construtiva. Tais heurísticas são compostas por repetições exaustivas [16], ou seja, é construído um bom padrão de corte e este é utilizado o máximo de vezes possível. São bem conhecidos na literatura os trabalhos dos autores: Hinxman [16] , Stadtler [31], Wäscher e Gau [10], dentre outros.

A heurística FFD consiste, de forma geral, em alocar os itens demandados em um padrão de corte por ordem decrescente de tamanho. Primeiramente, o item de maior tamanho é alocado no padrão de corte. Verifica-se o tamanho ainda disponível do objeto, se não for grande o suficiente para alocar outro item de igual tamanho, ele tenta alocar o segundo maior item, e assim até que o padrão esteja completo, isto é, nenhum outro item possa ser alocado. Estes passos deverão ser seguidos até que toda a demanda seja atendida.

1.3 O Problema de Corte de Estoque Ordenado

O problema de corte ordenado consiste em determinar a melhor sequência no atendimento a ordens de produção cujo objetivo é encontrar um plano de corte que minimize as perdas.

Cada ordem de produção $j \in \{1 \ . \ . \ .M\}$ possui n_j itens que deverão ser cortados de objetos maiores de comprimento W (bobinas, barras, etc.). Para cada ordem de produção j, um problema de corte de estoque tradicional é aplicado. Desse modo, de acordo com Rasgadale e Zobel [27], o problema de corte ordenado é uma variante do CSP tradicional, no qual um plano de corte buscando minimizar perdas deverá ser determinado. No entanto, esta variação do CSP, além de compartilhar das restrições presentes no CSP

padrão, possui algumas restrições particulares [27]:

- Todos os n_j itens pertencentes à ordem de produção j deverão ser cortados antes de qualquer item pertencente à próxima ordem de produção. Esta propriedade restringe a apenas um o número de lotes abertos ou em produção a cada momento, o que facilita o fluxo destas ordens de produção durante o processo e, portanto, minimiza a quantidade de itens deixados em estoque.

- Ao finalizar o corte de todos os itens de uma dada ordem j, todo comprimento restante do objeto em questão deverá ser utilizado (se possível) no corte dos itens da próxima ordem. Porém, se este comprimento não for totalmente reutilizado, ele deverá ser descartado, isto é, se o comprimento não é grande o suficiente para cortar os itens da ordem atual, e ainda assim é grande o suficiente para cortar algum item de uma ordem subsequente, este comprimento ainda assim será descartado. Apesar de aparentemente esta restrição ser um desperdício, ela evita os gastos com o estoque e transportes destas partes de objetos não utilizadas.

Visto que o problema de corte ordenado, como já dito, consiste em determinar a melhor sequência no atendimento a ordens de produção objetivando encontrar um plano de corte que minimize as perdas, a idéia central do problema é, com o retalho da ordem de produção j previamente processada, escolher a próxima ordem de produção $j+1$ a ser processada de tal forma que esta reutilize o retalho já existente.

Considere o seguinte exemplo:

Exemplo 1- Uma indústria manufatureira produz 5 tipos de produtos de tamanhos distintos, dados na tabela abaixo.

item(y)	1	2	3	4	5
tamanho	30	20	15	10	5

Tabela 1.1: Tamanhos dos itens.

Tais produtos são processados a partir de objetos brutos de tamanho 50.

A empresa, em uma certa data, precisa atender aos pedidos de 4 clientes, cujas demandas d deverão ser inteiramentes atendidas. A tabela abaixo apresenta as demandas por produtos de cada cliente.

Lotes(j)	(d(y,j))
1	{1,0,2,0,2}
2	{0,2,0,2,0}
3	{0,1,1,1,1}
4	{0,0,2,1,0}

Tabela 1.2: Demandas por produto de cada cliente.

O objetivo é determinar um plano de corte cuja perda seja mínima, sendo que todos os clientes devem ser processados em sequência. A figura abaixo apresenta uma solução ao problema: os clientes serão processados na ordem 1,2,3,4. Os itens do cliente 2, por exemplo, serão cortados antes de qualquer item do cliente 3, e depois de qualquer item do cliente 1. Nota-se que esta solução obteve uma perda de $30m$.

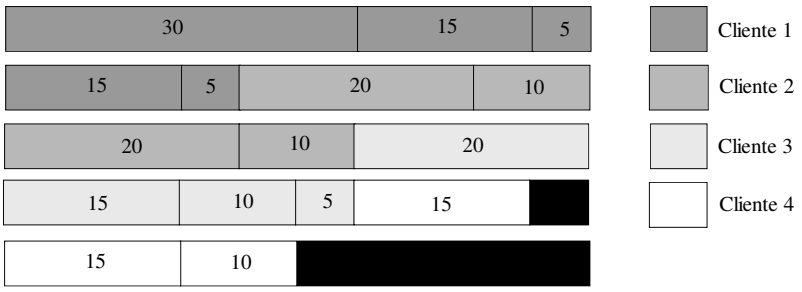


Figura 1.1: Exemplo de uma solução do OCSP

Pode ser feita uma analogia entre o OCSP e o TSP, o qual diz que o caixeiro viajante deve passar por um número de cidades pré determinadas, sem as repetir, com o objetivo de minimizar a distância percorrida. No problema de OCSP queremos processar todos os lotes, sem repeti-los, com o objetivo de minimizar perdas. Dessa forma, podemos dizer que OCSP combina aspectos do Problema de corte de estoque padrão e do Problema do caixeiro viajante [1], dois problemas clássicos NP-completos.

1.4 Modelos para a Resolução do OCSP

1.4.1 Algoritmos Genéticos

Rasgdale e Zobel [27] propuseram um primeiro modelo e uma aplicação ao OCSP usando algoritmos genéticos.

Algoritmos Genéticos (GA, do inglês *Genetic Algorithm*) é uma metaheurística proposta por John Holland [17], inspirada no processo biológico de evolução natural segundo a teoria Neo-Darwinista. GA [13] fundamenta-se na idéia de "sobrevivência do mais adaptado", isto é, dada uma população, os indivíduos com características genéticas melhores têm maiores chances de sobrevivência e de produzirem filhos cada vez mais aptos, enquanto indivíduos menos aptos tendem a desaparecer.

Nesta aplicação, o cromossomo é definido como um vetor contendo informações sobre a permutação dos lotes e a permutação dos itens dentro de cada lote. A função de *fitness* mensura o total da perda associada à solução, isto é, ao cromossomo.

Os cromossomos não guardam as informações sobre os retalhos dos objetos cortados, o que faz com que as operações de crossover e mutação padrão não possam levar esta informação em consideração ao modificarem as soluções, isto é, ao modificarem os cromossomos. Para resolver este problema, Rasgdale e Zobel, desenvolveram a heurística PARTICUT que modifica ou "repara" tais soluções.

Este trabalho foi o precursor do problema de corte ordenado. O método foi testado em problemas reais envolvendo 50 lotes, apresentando resultados bastante favoráveis.

1.4.2 Branch-and-price-and-cut algorithm

Alves e Carvalho [1] propuseram 3 novas formulações inteiras e um algoritmo exato baseado em geração de coluna, *branch and bound* e *cutting planes* para a resolução do problema de corte ordenado.

Diferentemente de Ragsdale e Zobel [27], este trabalho assume diferentes tamanhos de objetos brutos, além de uma classificação dos lotes quanto ao tamanho: pequeno, quando todos os itens podem ser cortados a partir de um único objeto bruto, isto é, composto por itens cujo comprimento total é menor ou igual ao tamanho do objeto, e grande caso contrário.

O algoritmo exibido neste trabalho invoca as formulações inteiras apresentadas no mesmo. Abaixo apresentaremos a formulação do denominado RMP (do inglês, *Restrict Master Problem*) que é o modelo base deste algoritmo.

W_k - Comprimento dos K objetos brutos , $k = 1, ..., K$.

B_k - Quantidade de objetos brutos disponíveis de tamanho W_k, $k = 1, ..., K$.

b_{ij} - Demanda dos itens de tamanho w_{ij} no lote j.

m_j - Quantidade de itens pertencentes ao lote j.

p - Quantidade de lotes.

P_k - Conjunto de padrões de corte associado ao objeto de tamanho W_k.

λ_{dk} - Quantidade de vezes que o padrão de corte $d \in P^k$ é utilizado.

a_{dk}^{ij} - Parâmetro que determina o número de itens de tamanho w_{ij} pertencentes a um lote j cortados em um padrão $d \in P_k$.

c_{dk}^{j} - Parâmetro binário. Este parâmetro recebe 1 quando o padrão j pertence parcialmente ao padrão $d \in P^k$ e divide o padrão com outro lote e recebe 0 caso contrário.

l_{dk}^{e} - Parâmetro binário. Este parâmetro recebe 1 se dois lotes são parcialmente cortados em um mesmo padrão de corte e 0 caso contrário.

$$\min \sum_{k=1}^{K} \sum_{d \in P^k} W_k \lambda_{dk} \tag{1.3}$$

$$\text{s.t.:} \sum_{k=1}^{K} \sum_{d \in P^k} a_{dk}^{ij} \lambda_{dk} = b_{ij} \qquad j = 1, ..., p; i = 1, ..., m_j \tag{1.4}$$

$$\sum_{k=1}^{K} \sum_{d \in P^k} c_{dk}^{j} \lambda_{dk} \leq 2 \qquad j = 1, ..., p \tag{1.5}$$

$$\sum_{d \in P^k} \lambda_{dk} \leq B_k \qquad k = 1, ..., K \tag{1.6}$$

$$\sum_{k=1}^{K} \sum_{d \in P^k} l_{dk}^{e} \lambda_{dk} \leq \mid I_e \mid -1 \qquad \forall I_e \in S \tag{1.7}$$

$$\lambda_{dk} \geq 0 \text{ e inteiro, k=1,...,K, } d \in P_k.$$

A Restrição (1.4) assegura o atendimento de toda a demanda. A Restrição (1.5) garante que um lote será parcialmente cortado com outros lotes em no máximo 2 objetos. A restrição referente à disponibilidade dos lotes é a (1.6). A Restrição (1.7) evita a criação de ciclos. Ciclos acontecem quando itens de um dado lote são cortados entre os itens de uma ordem ainda não finalizada, violando a restrição de sequência do OCSP.

Esta abordagem, além de fornecer um algoritmo exato, foi testada com dados reais utilizando 30 lotes, os quais obtiveram bons resultados.

1.4.3 Greedy Randomized Adaptive Search Procedure (GRASP)

Golfeto, Moretti e Salles Neto [14] desenvolveram um modelo matemático e um algoritmo para a resolução do problema de corte ordenado. Este algoritmo, que chamaremos de GRASP-OCSP, utiliza a metaheurística GRASP [8] combinada à técnica de Reconexão por Caminhos [12](do inglês, *Path Relinking*).

A metaheurística GRASP [9], que em português significa procedimento de busca adaptativa gulosa e aleatória, consiste em combinar métodos construtivos com uma busca local. GRASP é um processo iterativo ou *Multi-Start* em que cada iteração é composta por duas fases: uma fase de construção, onde as soluções viáveis são construídas a partir de um algoritmo guloso aleatório e uma fase de busca local, que aproveita a solução da

fase de construção e explora a vizinhança ao redor desta, para uma eventual melhora. A melhor solução encontrada ao longo de todas as iterações é retornada como o resultado.

Na primeira fase, frequentemente, cria-se uma lista restrita de candidatos, a RCL. Esta lista é criada mediante uma função gulosa f_g que estima o benefício da seleção de cada um dos elementos.

A técnica de Reconexão por Caminhos ou Religação de Caminhos foi proposta por Glover *et al.* [12] e consiste em gerar e explorar caminhos partindo de uma ou mais soluções elite (soluções de alta qualidade) que, através da geração de caminhos na vizinhança dessas soluções, leva a outras soluções elite. Para tal finalidade, são selecionados movimentos que introduzem atributos das soluções elite iniciais nas soluções correntes. A técnica Reconexão por Caminhos em conexão com a metaheurística GRASP foi inicialmente introduzida por Laguna e Martí [20].

Esta aplicação também utiliza o algoritmo de busca local 2-*opt* [2] em cada uma de suas iterações. Com o objetivo de diminuir perdas, a busca 2-*opt* consiste basicamente em remover duas arestas da solução e inserir novamente tais arestas de forma cruzada; por exemplo, sejam as arestas removidas os pares (i_1, i_2) e (j_1, j_2), as arestas serão novamente inseridas na forma (i_1, j_2) e (i_1, j_2). Se esta nova configuração for melhor que a anterior, é mantida a nova rota. Caso contrário, escolhem-se novamente duas arestas para análise. Um ciclo completo 2-*opt* verificará todas as trocas possíveis e realizará a melhor delas. O algoritmo deverá parar somente quando uma rodada completa não encontrar nenhuma melhora.

O modelo matemático desenvolvido por Golfeto, Moretti e Salles Neto, apresentado logo abaixo, admite objetos brutos de tamanhos idênticos. A notação utilizada será a seguinte:

NC - Quantidade de ordens de produção (número de lotes);
NK - Quantidade de padrões de corte do lote k;
NKL - Quantidade de padrões de corte divididos entre os lotes k e l.
m - Quantidade de itens.
d_{ik} - Parâmetro de define a demanda do item i pertencente ao lote k;
a_{ijk} - Parâmetro que define o número de vezes que o item i aparece no padrão de corte j

para o lote k;

a_{ijkl} - Parâmetro que define o número de vezes que o item i aparece no padrão de corte j para o lote k quando esse é dividido com padrão de corte do lote l;

x_{jk} - Variável de decisão inteira que determina no lote k o número de objetos processados utilizando o padrão j. Esta variável é igual a zero quando o objeto processado possui itens do lote k e l;

z_{kl} - Variável binária. Esta variável recebe 1 quando o objeto processado possui itens do lote k e l e 0 caso contrário;

Este modelo pretende minimizar o número de objetos processados, levando em consideração que, a cada momento, apenas uma ordem de produção deverá estar ativa.

$$\min \sum_{k=1}^{NC} \sum_{j=1}^{NK} x_{jk} + \sum_{k=1}^{NC} \sum_{l=k+1}^{NC} z_{kl} \tag{1.8}$$

$$\text{s.t.:} \sum_{j=1}^{NK} a_{ijk} x_{jk} + \sum_{l=1}^{NC} \sum_{j=1}^{NKL} a_{ijkl} z_{lk} = d_{ik} \qquad i = 1, ..., m; \quad k = 1, ..., NC \tag{1.9}$$

$$\sum_{k=l+1}^{NC} z_{kl} \leq 1 \qquad k = 1, ..., NC \tag{1.10}$$

$$\sum_{k=1}^{l-1} z_{kl} + \sum_{k=l+1}^{NC} z_{lk} \leq 2 \qquad l = 1, ..., NC \tag{1.11}$$

$$x_{jk} \geq 0, x_{jk} \in Z; j = 1, ...NK; k = 1, ..., NC$$

$$x_{kl} \in \{0, 1\}; k = 1, ..., NC; l = k + 1, ..., NC.$$

A demanda total será garantida pela Restrição (1.9). A garantia de que apenas um padrão de corte dividirá o mesmo par de clientes será dado pelas Restrições (1.10) e (1.11).

Cada iteração do GRASP-OCSP combina a heurística construtiva FFD e a busca local 2-*opt*. Primeiramente um cliente é escolhido aleatoriamente para entrar na sequência. Calcula-se o custo de entrada nesta sequência para todos os outros clientes e, com base nesses valores, estimamos então um valor intermediário, denominado BRCL, que será utilizado como limite para construção da Lista Restrita de Candidatos (RCL), de acordo

com a fórmula abaixo:

$$BRCL = o_{min} + 0,6(o_{max} - o_{min})$$

onde, o_{min} significa a melhor solução encontrada, isto é, a solução que contém o menor número de objetos processados utilizando a heurística FFD e o_{max} significa a pior solução encontrada, isto é, o maior número de objetos processados utilizando a heurística FFD.

Todos os elementos que tenham *custo de entrada* $<$ BRCL serão selecionados e formarão a RCL. Dentre todos os elementos que formam a RCL, um elemento será aleatoriamente selecionado para compor a solução.

Esse processo é repetido até que todos os clientes sejam selecionados e uma solução encontrada. Esse procedimento (*Multi-start*) foi utilizado na geração de 80 soluções, nas quais a busca local 2-opt foi executada. Após a execução da busca local, foram selecionadas as 20 melhores soluções (soluções elites) e então a técnica de Reconexão por Caminhos foi executada em 40 pares de soluções selecionados aleatoriamente.

Este trabalho testou casos envolvendo 50, 80 e 100 lotes. Os resultados foram comparados a uma adaptação ao modelo clássico de corte de estoque, apresentando bons resultados.

Capítulo 2

Conceitos dos Algoritmos Colônia de Formiga

Este capítulo apresenta uma introdução aos conceitos do algoritmo colônia de formigas (ACO, do inglês *Ant Colony Optimization*), uma metaheurística baseada no comportamento de formigas reais que faz uso dos princípios de auto-organização destas para coordenar uma população de agentes artificiais, visando à resolução de problemas de otimização combinatória.

2.1 Inspiração Biológica

Os algoritmos colônia de formigas, como já dito anteriormente, são inspirados no comportamento de formigas reais e, mais precisamente, no procedimento de coleta de alimento destas. Este procedimento é baseado em uma comunicação indireta entre as formigas, feita através de feromônios (uma substância produzida pela própria formiga). Enquanto caminham, as formigas depositam esta substância no chão, formando uma trilha de feromônios, que influenciará outras formigas a segui-la também.

Deneubourg *et al.* [3] desenvolveram alguns experimentos com formigas da espécie *I. humilis* chamados *"Double Bridge Experiments"*, cujo objetivo era entender e também quantificar o procedimento de formação da trilha de feromônio. Estes experimentos consistiam em conectar um formigueiro à uma fonte de alimento através de dois caminhos

alternativos, e a porcentagem de formigas que escolhiam cada caminho era medida.

Um dos experimentos consistiu na utilização de caminhos de tamanhos idênticos. Foi observado que cerca de 80% das formigas convergem para um mesmo caminho. Isto acontece porque inicialmente as formigas ficam livres para escolher qualquer um dos dois caminhos, a princípio sem qualquer indício de feromônio. No entanto, devido à flutuação aleatória, um número maior de formigas selecionará um dos caminhos, que com o tempo apresentará um nível de feromônio maior que o outro. Isto porque quanto maior o nível de feromônio, mais formigas serão atraídas para ele, até que a colônia de formigas convirja para o uso de um mesmo caminho. Este procedimento da colônia baseada em autocatálise - isto é, em um processo de *feedback* positivo (processo de realimentação positiva)- é um exemplo de auto-organização das formigas.

Em um segundo experimento [15], os caminhos possuem tamanhos distintos, um dos caminhos é duas vezes o tamanho do outro. O menor caminho é de fato o preferido, pois as formigas que o escolheram são as que chegam primeiro à comida e voltam ao ninho, portanto o nível de feromônio neste é mais alto, o que estimulará outras formigas a escolherem tal caminho, resultando em uma convergência para um mesmo percurso. Apesar de uma grande porcentagem das formigas escolherem o menor caminho, uma pequena porção delas ainda continua utilizando o maior caminho. Isto pode ser interpretado como uma "exploração do ambiente" à procura de novas fontes de alimento.

Um outro fator bastante importante, que deve ser levado em consideração no procedimento de formigas reais, é a evaporação do feromônio. Mesmo sendo lenta, a evaporação do feromônio permite que a colônia "esqueça" os caminhos ruins com o passar do tempo.

De acordo com Dorigo e Stützle [4], a convergência das formigas para um dos caminhos representa um procedimento coletivo macroscópico, que pode ser explicado pela atividade microscópica das formigas, isto é, pela interação local entre os indivíduos da colônia.

2.2 Formigas Artificiais

Dado o problema de minimização $(\mathcal{S}, f, \Omega)$, onde $\mathcal{S}$ representa o conjunto das soluções candidatas, f é a função objetivo que toma valores em cada solução $s \in \mathcal{S}$ e Ω representa o conjunto das restrições, uma formiga artificial neste problema de minimização é um processo estocástico construtivo [4], que iterativamente explora o grafo $G_C = (C, L)$, chamado de grafo de construção, onde C representa o conjunto dos vértices ou nós e L o conjunto das arestas que conectam os vértices do conjunto C, à busca de soluções candidatas factíveis de custo mínimo, isto é, soluções ótimas s^*.

Sendo *i, j* pertencentes a C, os conectores *l(i,j)* $\in L$ estão associados tanto à trilha de feromônio $\tau(i, j)$ quanto à informação heurística $\eta(i, j)$. As trilhas de feromônio são informações numéricas utilizadas por toda a colônia (informações coletivas), que refletem as experiências adquiridas por esta durante o processo. As trilhas de feromônio serão atualizadas ao longo do tempo pela passagem das formigas pelos conectores e pela evaporação do feromônio. Em grande parte das aplicações, esta informação refere-se a uma estimativa sobre o custo adicional ao inserir um dado elemento a uma solução.

A construção das soluções será guiada por uma regra de decisão probabilistica. Esta regra é dada em função das informações locais sobre a trilha de feromônio, das informações locais heurísticas e das restrições do problema.

Para construir soluções factíveis as formigas utilizam uma forma de memória artificial limitada $\mathcal{J}^k$. Esta memória estoca as informações sobre as soluções parciais. Durante ou após a construção de uma solução, as formigas atualizarão as trilhas de feromônios. Os valores associados às trilhas poderão aumentar (devido aos feromônios associados aos conectores presentes na solução) ou diminuir (devido à evaporação do feromônio).

2.3 Algoritmos ACO

Muitos algoritmos ACO foram propostos na literatura. Neste trabalho apresentaremos o original *Ant System* e uma de suas variações de maior sucesso: $\mathcal{MAX} - \mathcal{MIN}$ *Ant System* ($\mathcal{MMAS}$), o qual aplicaremos ao problema OCSP proposto.

Para uma visão mais intuitiva das diferenças existentes nos algoritmos citados acima, os apresentaremos aplicados ao problema do caixeiro viajante.

2.3.1 *ANT SYSTEM*

Desenvolvido por Dorigo *et al.* em 1992 [5] o *Ant System* foi o primeiro dos algoritmos ACO. Inicialmente três versão do *Ant System* foram propostas : *ant-density*, *ant-quantity* e *ant-cycle*. A principal diferença entre tais versões está no procedimento de atualização do feromônio: nas duas primeiras, as formigas atualizam o feromônio imediatamente após se deslocarem da cidade atual à uma cidade adjacente, já na última versão, as formigas apenas atualizarão a trilha de feromônio depois que todas estas tenham construído suas rotas. Neste trabalho apresentaremos a versão *ant-cycle*, a qual apresentou melhores resultados.

Intuitivamente, o problema do caixeiro viajante (TSP, do inglês *Traveling Salesman Problem*) pode ser colocado da seguinte forma: dado um conjunto de cidades e conhecidas as distâncias entre cada uma delas, um caixeiro viajante pretende determinar um menor caminho que passe por todas as cidades, exatamente uma vez, e que volte à cidade de onde partiu.

Formalmente, o TSP pode ser representado por um grafo, *G(N,A)* onde o conjunto de vértices N representa o conjunto das cidades e o conjunto das arestas A representa o conjunto das conexões entre tais cidades. Para cada $(i,j) \in A$ é assumido um valor $d(i,j)$ que representa a distância entre as cidades i e j. O TSP tem por objetivo encontrar o menor caminho que passe por todas as cidades, exatamente uma vez, enquanto minimiza o custo total das conexões utilizadas.

Consideremos m formigas artificiais. Inicialmente cada uma delas é colocada em uma cidade i escolhida aleatoriamente. Cada formiga irá construir uma rota da seguinte

maneira: a cada passo, cada formiga deverá escolher a próxima cidade observando duas variáveis: os níveis de feromônio e as distâncias. O nível de feromônio entre as cidade é dado por $\tau(i,j)$. Quanto maior for o nível de feromônio, maior o valor de $\tau(i,j)$ e maior a chance de a aresta ser selecionada. A distância entre as cidades determina o valor heurístico $\eta(i,j) = \frac{1}{d_{(i,j)}}$. Portanto, quanto menor for a distância $d(i,j)$, maior o valor de $\eta(i,j)$, e maior a chance da aresta correspondente a i e j ser selecionada. A probabilidade de a formiga k, estando na cidade i, escolher a cidade j, dentre as ainda não visitadas, será dada pela equação abaixo:

$$p_k(i,j) = \begin{cases} \dfrac{[\tau(i,j)]^\alpha \cdot [\eta(i,j)]^\beta}{\sum_{g \in J_k(i)} [\tau(i,g)]^\alpha \cdot [\eta(i,g)]^\beta} & \text{se } j \in J_k(i) \\ 0 & \text{caso contrário} \end{cases} \tag{2.1}$$

Nesta equação, J_k é o conjunto das cidades ainda não visitadas pela formiga k que está na cidade i. Os parâmetros α e β definem a importância relativa das informações heurísticas com oposição às informações do feromônio. Se $\alpha = 0$, então a probabilidade de seleção é proporcional a $\eta(i,j)$, beneficiando, desse modo, as cidades mais próximas. Neste caso, o algoritmo comporta-se como um método guloso estocástico com múltiplos pontos de partida. Se $\beta = 0$, somente a utilização de feromônio tem influência, induzindo a uma estagnação rápida da busca, isto é, todas as formigas seguindo o mesmo caminho e construindo as mesmas soluções, em geral subotimas.

Para construir soluções factíveis, as formigas utilizam a memória $\mathcal{J}^k$, que é utilizada para determinar, em cada passo da construção, quais cidades ainda não foram visitadas e também para que a formiga seja capaz de refazer o caminho de volta.

Uma vez que todas as formigas construíram uma rota, a trilha de feromônio é atualizada. Primeiramente, ocorre a evaporação do feromônio em todos os arcos, seguindo a seguinte equação:

$$\tau(i,j) = (1-\rho) \cdot \tau(i,j) \quad \forall (i,j) \in A, \tag{2.2}$$

onde $0 < \rho \leq 1$ fixado previamente. De acordo com Dorigo e Stützle [4], este parâmetro de evaporação é importante para evitar o acúmulo ilimitado de feromônio nas trilhas e

para que soluções ruins previamente tomadas sejam "esquecidas". De fato, se um arco não é escolhido pela formiga, o valor associado ao feromônio decresce exponencialmente.

Em um segundo momento, todas as formigas deverão depositar o feromônio nos arcos por onde passaram durante a rota, obedecendo a seguinte equação:

$$\tau(i,j) = \tau(i,j) + \sum_{k=1}^{m} \triangle\tau_k(i,j) \quad \forall (i,j) \in A, \tag{2.3}$$

onde,

$$\Delta\tau_k(i,j) = \begin{cases} \dfrac{1}{C_k} & \text{se } (i,j) \in \text{rota da formiga } k \\ 0 & \text{caso contrário.} \end{cases} \tag{2.4}$$

Na equação acima, $\triangle\tau_k(i,j)$ significa a quantidade de feromônio que a k-ésima formiga deposita nos arcos que ela visitou durante a rota. Por esta equação, nota-se que quanto mais os arcos são utilizados nas rotas das formigas, mais feromônio este arco receberá, e maior a chance de ser escolhido em futuras iterações do algoritmo. Também, quanto menor o caminho total C_k, feito pela formiga k, maior o nível de feromônio $\triangle\tau_k(i,j)$ a ser depositado. Desta maneira, o algoritmo dá maior peso às formigas que fizerem os menores percursos.

2.3.2 $\mathcal{MAX} - \mathcal{MIN}$ *Ant System*

Proposto por Stützle e Hoos [32], o $\mathcal{MAX} - \mathcal{MIN}$ *Ant System* ($\mathcal{MMAS}$) é um algoritmo que surgiu de algumas modificações do *Ant System*.

A primeira mudança é que apenas a formiga que obteve a melhor solução durante o processamento do algoritmo deposita feromônio na rota percorrida. Esta melhor solução pode ser adquirida pela formiga que apresenta a melhor solução na iteração atual *ib* (do inglês *iteration best*) ou a formiga que apresenta a melhor solução global *bs* (do inglês *best-so-far*), isto é, a solução da *bs* só será atualizada quando aparecer outra formiga que obtenha uma solução melhor. A regra de atualização do feromônio será modificada para:

$$\tau(i,j) = (1-\rho) \cdot \tau(i,j) + \Delta\tau^{best}(i,j) \tag{2.5}$$

onde,

$$\Delta\tau^{best}(i,j) = \frac{1}{C^{best}}. \tag{2.6}$$

Na equação acima, C^{best} é o caminho total percorrido pela melhor formiga e $\Delta\tau^{best}(i,j)$ significa a quantidade de feromônio que a melhor formiga deposita nos arcos visitados por ela durante a rota. Dessa forma $\Delta\tau^{best}(i,j)$ consegue medir a qualidade das soluções e portanto, equivale à função de *fitness* desta aplicação.

Se a atualização do feromônio for feita via *bs*, a busca irá concentrar-se na solução desta formiga, deste modo, a exploração de soluções melhores será limitada e consequentemente a possibilidade de a busca ficar restrita a soluções pobres aumenta. Se a atualização for feita utilizando *ib*, o número de arcos que receberão feromônio é maior, deixando assim a busca mais aberta.

Independentemente da escolha entre melhor da iteração ou melhor global para a atualização da trilha de feromônio, uma possível estagnação poderá ocorrer. Esta estagnação pode ser causada pelo crescente aumento dos valores das trilhas de feromônios nos arcos pertencentes às soluções das melhores formigas, em geral soluções subotimas.

Para solucionar o problema descrito acima, uma segunda modificação foi introduzida ao $\mathcal{MAX} - \mathcal{MIN}$: imposição de limites aos valores das trilhas de feromônio, $[\tau_{min}, \tau_{max}]$. Em particular, pretende-se limitar a probabilidade *p(i,j)* de uma formiga, estando na cidade i, escolher a cidade j para se mover.

De acordo com Dorigo e Stützle [4], a trilha de feromônio é limitada superiormente pela razão entre a função de *fitness* f aplicada à solução ótima s^* e o parâmetro de evaporação ρ. Para qualquer $\tau(i,j)$ é correto afirmar:

$$\lim_{k\to\infty} \tau(i,j)(k) \leq \tau_{max} = \frac{f(s^*)}{\rho} \tag{2.7}$$

onde k representa o número da iteração.

Baseado na relação acima, $\mathcal{MAX} - \mathcal{MIN}$ aplicado ao TSP estima que

$$\tau_{max} = \frac{1}{\rho \cdot C^*}, \tag{2.8}$$

onde C^* representa a trajetória ótima. De acordo com Stützle [32] e Dorigo e Stützle [4], o algoritmo terá melhores resultados se a trilha de feromônio for também limitada inferiormente.

Uma terceira mudança é inicializar a trilha de feromônio com o valor ou uma estimativa do valor do τ_{max}. Tal inicialização, junto com uma pequena evaporação do feromônio, aumenta a exploração das rotas desde o começo da busca. Uma última modificação é reiniciar as trilhas de feromônio sempre que o sistema chegar em uma estagnação.

2.4 Aplicação de ACO em CSP

Levine e Ducatelle [21] desenvolveram um modelo ACO para aplicar no problema de corte de estoque unidimensional. Este modelo segue as idéias do $\mathcal{MAX} - \mathcal{MIN}$ apresentadas na Seção 2.3.2 como será descrita abaixo:

Trilha de feromônio - $\tau(i, j)$ representa a tendência dos itens i e j se encontrarem num mesmo padrão de corte.

Informação Heurística - Nesta aplicação, a informação heurística é inspirada na heurística construtiva FFD (apresentada na Seção 1.2.1). Em cada padrão de corte os itens serão alocados um a um, seguindo a ordem decrescente de comprimento. A heurística será dada por $\eta_j = w_j$, sendo w_j o tamanho do item j.

Construção dos padrões - Cada formiga inicia uma iteração com uma lista da demanda a ser atendida e um padrão vazio. A cada passo da iteração, a formiga aloca um item no padrão de corte, de acordo com a equação de probabilidade 2.9. Tais itens serão alocados no padrão até que mais nenhum item caiba mais, e então a formiga inicia outro padrão de corte. Este processo se repete até que toda a demanda de itens seja atendida. A probabilidade de uma formiga k alocar um item j de tamanho w_j como próximo item no padrão atual b e solução parcial s é dado por:

$$P_k(s, b, j) = \begin{cases} \dfrac{[\tau_b(j)] \cdot [\eta(j)^\beta]}{\sum_{h \in J_k(s,b)} [\tau_b(h)] \cdot [\eta(h)^\beta]} & \text{se } j \in J_k(s, b) \\ 0 & \text{caso contrário.} \end{cases} \tag{2.9}$$

Nesta equação, o conjunto dos itens factíveis $J_k(s, b)$ de uma formiga k é composto pelos itens que ainda podem ser alocados no padrão, isto é, itens que ainda cabem no padrão. O parâmetro $\tau_b(j)$ será definido como a média dos valores do feromônio entre o item j a ser adicionado e os itens que já pertencem ao padrão b:

$$\tau_b(j) = \begin{cases} \frac{\sum_{i \in b} \tau(i,j)}{\mid b \mid} & \text{se } b \neq \varnothing \\ 1 & \text{caso contrário.} \end{cases} \tag{2.10}$$

Atualização do Feromônio - A trilha de feromônio será atualizada de acordo com a seguinte regra:

$$\tau(i,j) = \rho \cdot \tau(i,j) + m \cdot f(s^{best}), \tag{2.11}$$

onde m indica quantas vezes os itens i e j estão no mesmo padrão de corte e ρ é um valor previamente fixado entre 0 e 1.

$$f(s) = \frac{\sum_{i=1}^{N} (F_i/W)^k}{N}. \tag{2.12}$$

Nesta equação, N significa o total de padrões, F_i significa o comprimento total dos itens alocados no padrão i, e W é o comprimento do objeto a ser processado. O parâmetro k determina o peso dado ao preenchimento do padrão (numerador da equação), em oposição à quantidade de padrões processados (denominador da equação).

Nesta aplicação o limite superior τ_{max} é definido como

$$\tau_{max} = \frac{1}{1-\rho}. \tag{2.13}$$

.

Busca Local - A busca local utilizada nesta aplicação consiste, primeiramente, em excluir os padrões de menor comprimento (o número de padrões a ser excluído é um parâmetro determinado empiricamente), tornando os itens pertencentes a estes livres para serem alocados novamente. Em todos os padrões restantes ocorrerão trocas de um ou dois itens pertencentes a estes padrões por itens livres, sendo que o padrão modificado, se necessário, será completado utilizando a FFD.

Capítulo 3

Aplicações do ACO em OCSP

Este capítulo descreve uma aplicação da metaheurística colônia de formigas no problema de corte ordenado a qual denominaremos ACO-OCSP.

3.1 Construção do Grafo

O primeiro passo para se aplicar ACO em problemas de otimização combinatorial é a definição de um grafo de construção. Em muitas aplicações esta definição não é muito intuitiva, sendo ela feita implicitamente utilizando a matriz de informações heurísticas [4]. Porém em alguns casos, a aplicação é bastante natural, como é o caso do OCSP.

O OCSP será representado por um grafo *G(N,A)*, onde N (conjunto dos vértices) representa os lotes a serem processados e A (conjunto das arestas) representa o conjunto das perdas, isto é, cada aresta $(i,j) \in A$ assume um custo $R(i,j)$ tal que este representará a perda obtida com o corte do lotes j imediatamente após o lote i. A perda $R(i,j)$ nesta aplicação, assim definida, tem semelhança com a distância entre as cidades do problema TSP, o que justifica a aplicação do algoritmo ACO para o problema OCSP.

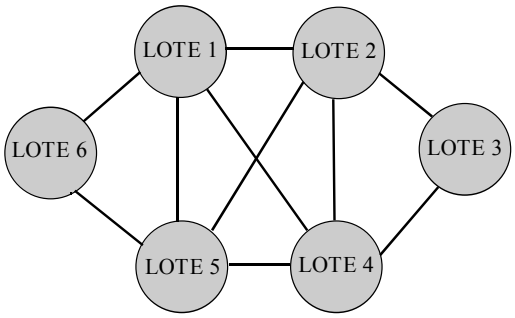


Figura 3.1: Exemplo de um grafo aplicado ao OCSP

3.2 Construção das Soluções

Cada formiga construirá sua solução caminhando sobre o grafo de construção, apresentado na Seção anterior, da seguinte forma: inicialmente, cada uma das m formigas será colocada em um dos n lotes aleatoriamente. Nos passos subsequentes cada formiga escolherá o próximo lote a ser processado, dentre os ainda não escolhidos por esta formiga, de acordo com a regra probabilística apresentada abaixo. Os vértices do grafo serão incorporados um a um na solução, formando as soluções parciais, até que esta inclua todos os vértices.

$$p_k(i,j) = \begin{cases} \dfrac{[\tau(i,j)] \cdot [\eta(i,j)]^{\beta}}{\displaystyle\sum_{g \in J_k(i)} [\tau(i,g)] \cdot [\eta(i,g)]^{\beta}} & \text{se } j \in J_k(i) \\ 0 & \text{caso contrário} \end{cases} \tag{3.1}$$

Esta regra traz a probabilidade de um lote j ser escolhido imediatamente após o lote i já pertencente a solução.

Os valores de i e j das trilhas de feromônio $\tau(i,j)$ e das informações heurísticas $\eta(i,j)$ são apresentados em matrizes $n \times n$ cujas construções serão apresentadas a seguir.

A construção da solução finaliza apenas quando todos os lotes tenham sido processados, isto é, todos os vértices estejam compondo a solução.

3.3 Restrições

O problema de corte ordenado apresenta restrições referentes à ordem e restrições referentes ao atendimento da demanda. São elas:

1. Cada lote n só poderá ser processado uma única vez.

2. Todos os itens y pertencentes ao lote j deverão ser cortados antes de qualquer item pertencente ao próximo lote.

3. Ao finalizar o corte de todos os itens de um dado lote j, todo comprimento restante do objeto em questão deverá ser reutilizado (se possível) no corte dos itens do próximo lote. Mas serão desconsiderados no corte dos itens dos lotes seguintes ao *j+1*.

4. As restrições referentes ao corte tradicional também serão consideradas neste problema, isto é, todos os itens pertencentes a todos n lotes deverão ser cortados, assim como a demanda destes itens não deverá ser excedida.

3.4 Definição das Trilhas de Feromônio e Informações Heurísticas

A escolha de uma definição para a heurística e para a trilha de feromônio que esteja de acordo com a natureza do problema é um fator de extrema importância para uma implementação ACO, pois as informações heurísticas combinadas às informações sobre a trilha de feromônio serão usadas diretamente na construção das soluções. Mais precisamente, elas serão utilizadas na regra probabilística (3.1) que guia as formigas para boas soluções.

A trilha de feromônio representa a informação de atratividade do caminho deixado pelas formigas anteriores. Nota-se que esta definição é análoga à utilizada para a aplicação em TSP [4]. Estas trilhas de feromônio serão apresentadas em uma matriz $n \times n$ que será atualizada a cada iteração. O procedimento de atualização da matriz de informações heurísticas será apresentado na Seção 3.6.

A informação heurística associada a todo conector *(i,j)* $\in A$ será dada por:

$$\eta(i,j) = \frac{C(i,j) - R(i,j)}{C(i,j)} \tag{3.2}$$

sendo que *C(i,j)* significa o comprimento resultante do corte dos lotes i e j respectivamente (incluindo as perdas) e *R(i,j)* é o somatório do comprimento das perdas obtidas com este corte.

Esta equação significa a razão entre o comprimento efetivamente utilizado (sem as perdas) e o comprimento total, o que significa que quanto menor a perda *R(i,j)*, maior será o valor de $\eta(i,j)$ e então, maior a chance deste arco *(i,j)* ser escolhido. Fazendo uma analogia ao problema do TSP apresentado na Seção anterior, a distância entre as cidades equivale a $d(i,j) = \dfrac{C(i,j)}{C(i,j) - R(i,j)}$, ou seja, quanto menor a perda percentual em retalho, menor a distância.

Nesta fase de construção das heurísticas, não será computada como perda o retalho final, pois este retalho ainda será utilizado no corte de outro lote. Tais informações heurísticas, assim como as informações das trilhas de feromônio, serão guardadas em uma matriz $(n \times n)$.

Neste trabalho utilizaremos duas estratégias para a construção da matriz de informações heurísticas: Matriz Fixa de Informações Heurísticas e a Matriz Variável de Informações Heurísticas. Nas próximas seções apresentaremos tais matrizes com maiores detalhes.

3.4.1 Matriz Fixa de Informações Heurísticas

A Matriz Fixa de Informações Heurísticas (η_F), é calculada apenas no início do algoritmo e não sofrerá modificações ao longo de todo o processamento.

Cada célula (i, j) da matriz é calculada utilizando a heurística FFD apresentada na Seção 1.2.1 do Capítulo 1. Inicialmente, o lote i é inteiramente processado utilizando o procedimento da heurística FFD e com o retalho obtido iniciaremos o corte do lote j. Dessa forma obteremos *C(i,j)* e, consequentemente, *R(i,j)*, que é o somatório das perdas obtidas ao longo da construção de *C(i,j)*.

Para a construção dos elementos *(i,j)* pertencentes a esta matriz, primeiramente, fixa-se o primeiro lote i a ser processado. Usando a heurística FFD montam-se os padrões para este lote fixo em composição com todos os outros n-1 lotes ainda não processados. A seguir, apresentaremos um exemplo da construção desta matriz. Utilizando os mesmos parâmetros do exemplo da Seção(1.3), montaremos a primeira coluna da mesma.

- $\eta(1, 1)$ - Será zero, afinal o lote só poderá ser processado uma única vez.
- $\eta(1, 2)$- Processa-se o lote 1 em composição com o lote 2.

  A partir do retalho obtido com o corte do lote 1, processaremos o lote 2.

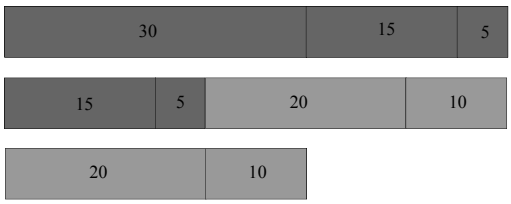


Figura 3.2: Lote 1 e lote 2 processados

Logo,

$C(1, 2) = 130$

$R(1, 2) = 0.$

Portanto, $\eta(1, 2) = 1$

- $\eta(1,3)$ - Lote 1 em composição com o lote 3.

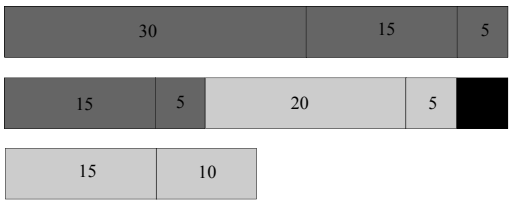


Figura 3.3: Lote 1 e lote 3 processados

Logo,

$C(1,3) = 125$.

$R(1,3) = 5$.

Portanto, $\eta(1,3) = 0.96$.

- $\eta(1,4)$ - Lote 1 em composição com o lote 4.

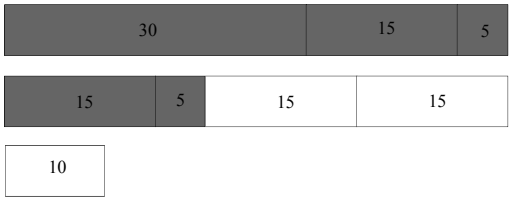


Figura 3.4: Lote 1 e lote 4 processados

Logo,

$C(1,4) = 110$.

$R(1,4) = 0$.

Portanto, $\eta(1,4) = 1$.

Seguindo esta mesma idéia, os lotes 2, 3 e 4 deverão ser fixados e relacionados a todos os outros lotes.

A matriz de perdas, isto é, a matriz de informações heurísticas resultantes será:

$$\eta_F = \begin{pmatrix} 0 & 1 & 1 & 0.96 \\ 1 & 0 & 1 & 1 \\ 0.96 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix} \tag{3.3}$$

Considere agora um outro caso em que uma solução parcial seja dada por $S = \{1, 2, 5\}$.

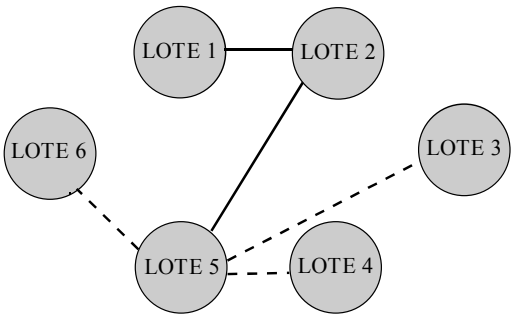


Figura 3.5: Exemplo de um grafo representando a solução parcial $S = \{1, 2, 5\}$

.

A formiga deverá escolher probabilisticamente entre os vértices ainda não escolhidos 3, 4 ou 6. Para isto ela fará uso da regra probabilística já apresentada, a qual irá fazer referência aos valores da matriz fixa de informações heurísticas. De acordo com as características desta, os valores para $\eta(5, i)$ tal que $i \in \{3, 4, 6\}$, não levarão em consideração as ligações anteriores $1 \longrightarrow 2$ e $2 \longrightarrow 5$, e, dessa forma os cortes de $\eta(5, i)$ não utilizarão o resto do objeto do corte anterior para iniciar o seu processamento.

Conceitualmente, portanto, a matriz de perdas η_F deveria ser diferente para cada solução parcial, dado que a sequência escolhida até o momento poderá exercer influência na configuração de cortes futuros. Para contornar este suposto problema foi desenvolvida uma adaptação a esta Matriz Fixa: a Matriz Variável de Informações Heurísticas.

3.4.2 Matriz Variável de Informações Heurísticas

A Matriz Variável de Informações Heurísticas (η_V) sofre modificações durante todo o processo de formação da solução. Após a construção de cada solução parcial a matriz é atualizada, pois o cálculo para os valores de $\eta(i, j)$ levará em consideração todas as ligações anteriores.

Apresentaremos a construção desta heurística através de um exemplo ilustrativo. Considere o exemplo apresentado na Seção (1.3).

Inicialmente a matriz será construída exatamente como a Matriz Fixa de Informações Heurísticas apresentada na seção anterior. A solução parcial S_1 será formada utilizando os valores desta matriz. Suponha que o vértice escolhido a compor a solução seja o lote 2, desse modo a solução parcial será $S_1 = \{1, 2\}$.

Deverá ser feita, então, uma mudança na segunda coluna da matriz. A partir do retalho obtido com a ligação anterior $(1 \rightarrow 2)$ inicia-se o corte do próximo lote.

- $\eta(2, 1)$ e $\eta(2, 2)$ - Recebem zero, pois não são valores factíveis.
- $\eta(2, 3)$ - Lote 2 em composição com o lote 3.

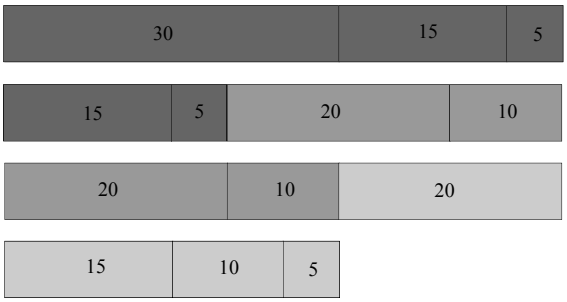


Figura 3.6: Lotes 2 e 3 processados

$C(2, 3) = 190$.

$R(2, 3) = 0$.

Portanto, $\eta(2, 3) = 1$.

- $\eta(2,4)$ - Lote 2 em composição com o lote 4.

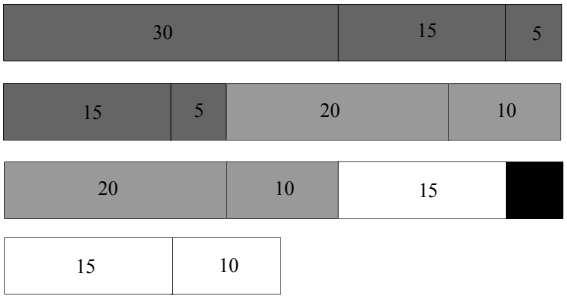


Figura 3.7: Lotes 2 e 3 processados

$C(2,4) = 175$.

$R(2,4) = 5$.

Portanto, $\eta(2,4) = 0.97$

3.5 Função de *Fitness*

Para que o algoritmo consiga medir a qualidade e convergir para boas soluções, ele utiliza uma função de *fitness*. Nesta aplicação utilizaremos a função de *fitness* definida abaixo:

Seja um conjunto de soluções Ω. Fixada uma solução $D = \{\lambda_1, \lambda_2, ..., \lambda_n\}$ teremos para n par:

$$f(s) = \frac{\sum_{i=1}^{n/2}(C(\lambda_{2i-1}, \lambda_{2i})) - \sum_{i=1}^{n/2}(R(\lambda_{2i-1}, \lambda_{2i}))}{\sum_{i=1}^{n/2}(C(\lambda_{2i-1}, \lambda_{2i}))}. \tag{3.4}$$

Para n ímpar teremos:

$$f(s) = \frac{(\sum_{i=1}^{n/2}(C(\lambda_{2i-1}, \lambda_{2i}))) + \overline{C_{\lambda_n}}) - (\sum_{i=1}^{n/2}(R(\lambda_{2i-1}, \lambda_{2i})) + \overline{R_{\lambda_n}})}{\sum_{i=1}^{n/2}(C(\lambda_{2i-1}, \lambda_{2i})) + \overline{C_{\lambda_n}}}. \tag{3.5}$$

Onde, n significa a quantidade de lotes processados. O parâmetro $\overline{C_{\lambda_n}}$ representa o comprimento obtido com o corte do último lote (λ_n) e $\overline{R_{\lambda_n}}$ o somatório das perdas obtidas com o corte do último lote (λ_n).

De acordo com a Restrição 3 apresentada na Seção 3, o retalho obtido ao se processar o último lote deverá ser descartado, deste modo, $\overline{R_{\lambda_n}}$ também deverá levar em consideração o último retalho.

Baseado na mesma Restrição 3 do problema, podemos dizer que o objetivo do problema aqui discutido é minimizar o número de objetos processados. Dessa forma, poderemos definir $(\sum_{i=1}^{n/2}(C(\lambda_{2i-1}, \lambda_{2i}))) + \overline{C_{\lambda_n}})$ como sendo o produto entre o número total de objetos processados (n) e o tamanho do objeto bruto W.

$$f(s) = \frac{(n \cdot W) - ((\sum_{i=1}^{n/2}(R(\lambda_{2i-1}, \lambda_{2i})) + \overline{R_{\lambda_n}}))}{(n \cdot W)}. \tag{3.6}$$

Esta função é a razão entre o comprimento total do objeto efetivamente utilizado (comprimento total sem as perdas) e o comprimento total do objeto utilizado incluindo as perdas.

3.6 Atualização da Trilha de Feromônio

A atualização da Trilha de Feromônio seguirá a idéia do $\mathcal{MAX} - \mathcal{MIN}$. Desse modo, apenas a melhor formiga atualizará a trilha de feromônio ao final de cada iteração. A atualização da trilha utilizará a equação abaixo:

$$\tau(i,j) = (1-\rho).\tau(i,j) + f(s^{best}). \tag{3.7}$$

O τ_{max} será definido da seguite forma:

$$\tau_{max} = \frac{C_{inf} - R_{inf}}{\rho \cdot C_{inf}}. \tag{3.8}$$

C_{inf} e R_{inf} correspondem ao limite inferior do comprimento total dos objetos processados e ao limite inferior das perdas obtidas ao longo do processo, respectivamente. O limite inferior das perdas, neste caso, será apenas o retalho obtido com o corte do último lote, isto é, como se pudéssemos, a partir do objeto W, processar todos os itens de tal forma que não existisse perda alguma entre os itens, apenas o retalho do último lote. Dessa forma, C_{inf} é definido como produto de todos os itens demandados (de todos os n lotes) pelos seus respectivos tamanhos somado ao retalho R_{inf}:

$$C_{inf} = \lceil (\sum_{j=1}^{N} (D(y,j) \cdot T(y,j))) \cdot \frac{1}{W} \rceil \cdot W \tag{3.9}$$

logo,

$$C_{inf} - R_{inf} = \sum_{j=1}^{N} (D(y,j) \cdot T(y,j)) \quad i = 1, ..., m. \tag{3.10}$$

Onde, m é o número de itens pertencentes aos lotes, $D(y,j)$ corresponde à demanda dos itens y pertencentes ao lote j e $T(y,j)$ correspondem aos comprimentos dos itens y pertencentes ao lote j.

Definimos $\tau_{min} = 0,002$ através de testes computacionais.

3.7 Busca Local

Assim como a grande maioria das metaheurísticas, ACO pode obter um melhor desempenho ao se acoplar ao algoritmo um método de busca local [4]. Dependendo de cada tipo de problema, diferentes buscas locais podem ser aplicadas, dentre elas: 2-*opt* [2], 3-*opt* [23] e *Lin-Kernighan*[24].

Neste trabalho, aplicam-se duas heurísticas de busca local: 2-*opt*, apresentada na seção 1.4.3, que será aplicada após a construção das soluções, e uma busca local acoplada à FFD, durante a construção dos padrões de corte.

A heurística 2-*opt* será aplicada após cada iteração, apenas na solução apresentada pela melhor formiga, devido à utilização do algoritmo $\mathcal{MAX} - \mathcal{MIN}$.

Durante a construção dos padrões de corte inseridos nos métodos de construção da matriz heurística e função de *fitness*, uma busca local é acoplada à FFD. Sabendo que os padrões serão construídos um a um, a busca será aplicada após a construção de cada um destes padrões da seguinte forma: retira-se o último item alocado (desde que este item não seja o menor item demandado), e este deverá ser substituido por um item de comprimento menor mais próximo e então o padrão será completado utilizando a FFD. Assim, chegamos à solução 1 modificada. De forma análoga, deveremos aplicar este procedimento a todos os itens pertencentes à solução. Destas, tomaremos a de menor perda e substituiremos a solução inicial. Exemplificando, aplicaremos no padrão abaixo a busca local acoplada à FFD. Esta busca local só será aplicada a padrões de corte cuja perda seja diferente de zero.

Considere na tabela abaixo os tamanhos e as demandas para cada item de um lote qualquer. Considere $W = 20$.

Item	1	2	3	4
Tamanho	7	5	4	2
Demanda	2	2	2	2

Tabela 3.1: Tamanhos e demandas para cada item do lote.

Utilizando a FFD tradicional, o primeiro padrão será montado da seguinte forma:

Figura 3.8: Exemplo de Padrão formado utilizando a FFD.

Aplicando a busca a este padrão de corte cujo resto é igual a 1, obteremos 3 novos padrões.

Modificação 1: Substitui-se o último item pertencente ao padrão pelo item mais próximo de menor comprimento.

Figura 3.9: Exemplo de Padrão modificado pela busca local.

Modificação 2: Substitui-se o penúltimo item pertencente ao padrão pelo item mais próximo de menor comprimento.

Figura 3.10: Exemplo de Padrão modificado pela busca local.

Modificação 3: Substitui-se o primeiro item pertencente ao padrão pelo item mais próximo de menor comprimento.

Figura 3.11: Exemplo de Padrão modificado pela busca local.

3.8 Exemplo de uma aplicação ACO-OCSP

Nesta seção apresentaremos um exemplo de uma aplicação ACO-OCSP. Efetuaremos apenas uma iteração do algoritmo proposto.

Uma indústria manufatureira produz 10 tipos de produtos de tamanhos distintos, dados na tabela abaixo. Tais produtos são processados a partir de objetos brutos de tamanho 1000.

item	1	2	3	4	5	6	7	8	9	10
tamanho	195	174	162	137	98	58	40	34	15	13

Tabela 3.2: Tamanhos dos itens dos lotes.

A empresa, em uma certa data, recebe pedidos de 5 clientes, cujas demandas deverão ser inteiramentes atendidas. A tabela abaixo apresenta as demandas por produtos de cada lote de cliente.

Lotes	(d(item,lotes))
1	{0 , 9 , 1 , 0 , 4 , 6 , 6 , 0 , 8 , 21}
2	{32 , 0 , 0 , 9 , 5 , 0 , 2 , 9 , 0 , 0}
3	{4 , 0 , 12 , 0 , 0 , 5 , 0 , 7 , 0 , 13}
4	{16 , 6 , 0 , 0 , 10 , 0 , 8 , 7 , 10 , 1}
5	{0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 1 , 1}

Tabela 3.3: Demandas por produto de cada cliente.

- Passo 1- Inicialização dos dados

Consideremos os valores:

$\alpha = 1.$

$\beta = 2.$

$\rho = 0.9.$

número de formigas = 5.

Inicialização da matriz de Informações Heurísticas

Utilizaremos a estratégia da matriz Fixa de Informações Heurísticas apresentada na Seção 3.4.1.

$$\begin{pmatrix} 0 & 0.986 & 0.9970 & 0.9933 & 0.9965 \\ 0.9905 & 0 & 0.9895 & 0.9853 & 0.9980 \\ 0.999 & 0.9877 & 0 & 0.9948 & 0.9979 \\ 0.996 & 0.988 & 0.9959 & 0 & 0.9952 \\ 0.999 & 0.984 & 0.9987 & 0.9933 & 0 \end{pmatrix}$$

Inicialização da Matriz de Feromônios

De acordo com a estratégia $\mathcal{MAX} - \mathcal{MIN}$ utilizada no algoritmo, a matriz deverá ser inicializada com o valor de τ_{max} obtido de acordo com (3.8).

$$\tau_{max} = 2122.96$$

Desse modo,

$$\begin{pmatrix} 2122.963 & 2122.96 & 2122.96 & 2122.96 \\ 2122.96 & 2122.96 & 2122.96 & 2122.96 \\ 2122.96 & 2122.96 & 2122.96 & 2122.96 \\ 2122.96 & 2122.96 & 2122.96 & 2122.96 \end{pmatrix}$$

- Passo 2- Construção das soluções

Uma formiga escolhe aleatoriamente o primeiro lote a fazer parte de sua solução. Suponha que o lote escolhido seja o Lote 4, então teremos a solução parcial $S_1 = \{4\}$.

Aplicando a regra probabilística da Seção 3.1 em todos os lotes ainda não pertencentes à solução, teremos:

$$p_k(4,1) = 0.2508.$$
$$p_k(4,2) = 0.2468.$$
$$p_k(4,3) = 0.2516.$$
$$p_k(4,5) = 0.2508.$$

Para obtermos o próximo lote a fazer parte da solução usaremos um método similar ao algoritmo da roleta dos algoritimos genéticos, onde cada lote candidato será representado em uma roleta proporcionalmente à sua probabilidade. Assim, os lotes com uma

maior probabilidade terão uma maior fatia da roleta e uma maior chance de serem sorteados. Finalmente, a roleta é girada um determinado número de vezes, e são escolhidos, aqueles sorteados na roleta.

Suponha que o lote sorteado seja o Lote 2, logo $S_2 = \{4, 2\}$.

$$p_k(2, 1) = 0.3342.$$
$$p_k(2, 3) = 0.3341.$$
$$p_k(2, 5) = 0.3317.$$

Mais uma vez aplicando o algoritmo da roleta e seguindo a mesma idéia acima, todas as soluções parciais serão construídas até que a formiga tenha processado todos os lotes, obtendo a solução final $S = \{4, 2, 1, 5, 3\}$.

Calcularemos a função de *fitness* dada pela Fórmula (3.6).

$$f(s) = 0.9438.$$

Após todas as formigas terem construído sua solução paralelamente, a melhor formiga será escolhida, isto é, a formiga que apresentou o menor número de lotes processados. A tabela abaixo apresenta as soluções e os valores da função de *fitness* para as 5 formigas.

Formiga	Solução	fitness
1	S={2,3,1,5,4}	0.9887.
2	S={4,5,2,1,3}	0.9438.
3	S={1,3,4,5,2}	0.9887.
4	S={4,2,1,5,3}	0.9438.
5	S={3,4,1,5,2}	0.9887.

Tabela 3.4: Valores das soluções e funções de *fitness* para as formigas do exemplo.

Neste Exemplo, 3 formigas apresentaram o mesmo valor para a função de *fitness*. Isto aconteceu devido à simplicidade deste exemplo, e também por este exemplo trabalhar apenas a primeira iteração, na qual todas as trilhas de feromônio são inicializadas com o mesmo valor. Em aplicações práticas a chance disso acontecer é reduzida.

Cálculo da atualização da trilha de feromônio

As formigas deverão atualizar todas as arestas contidas em suas soluções. Atualizaremos a primeira formiga cuja solução é dada por $S = \{2, 3, 1, 5, 4\}$.

Primeiramente calcula-se a evaporação em todas as arestas, utilizando a fórmula abaixo:

$$\tau(i, j) = (1 - \rho) \cdot \tau(i, j)$$

Agora todos as arestas pertencentes às soluções das melhores formigas deverão ser atualizadas da seguinte forma:

$$\tau(2, 3) = \tau(2, 3) + f(s^{best})$$
$$\tau(2, 3) = 213.285$$

$$\tau(3, 1) = \tau(3, 1) + f(s^{best})$$
$$\tau(2, 3) = 213.285$$

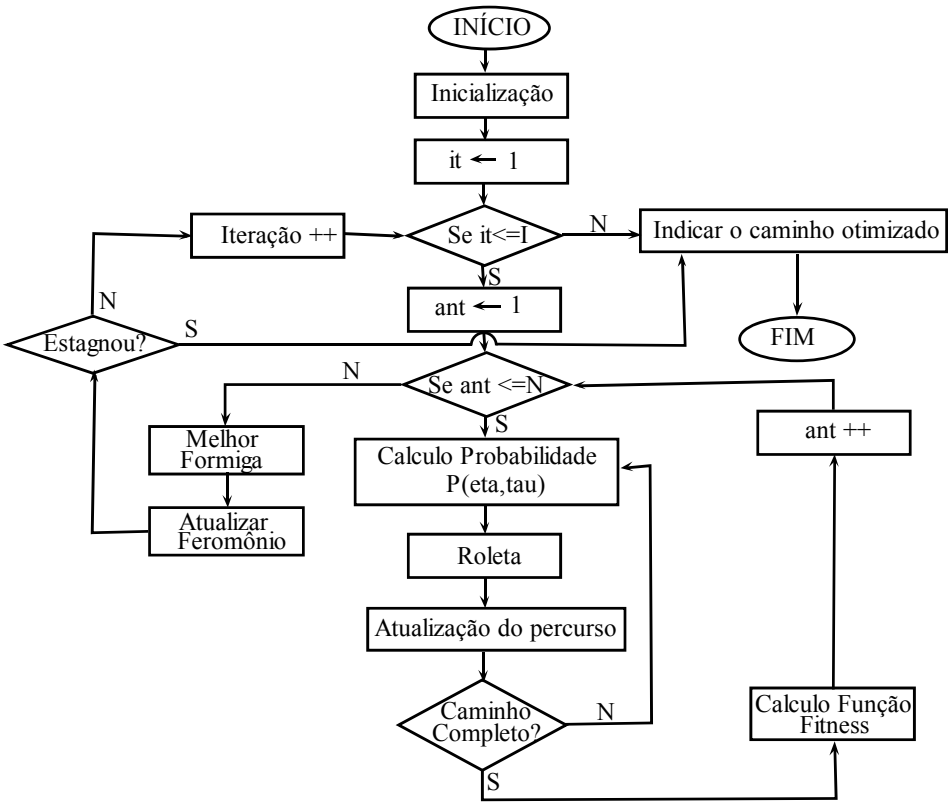
$$\tau(1, 5) = \tau(1, 5) + f(s^{best})$$
$$\tau(1, 5) = 213.285$$

$$\tau(5,4) = \tau(5,4) + f(s^{best})$$

$$\tau(5,4) = 213.285$$

Neste Exemplo, as arestas pertencentes à solução obteram valores iguais, pois estamos apresentando a primeira iteração. Tais valores se tornarão diferentes com o passar das iterações.

Abaixo, apresentaremos um fluxograma da aplicação ACO em OCSP.



A seguir, uma explicação sobre as etapas do algoritmo apresentado será dada. 1- Inicialização dos dados:

- Parâmetros de entrada.
- Cálculo da Matriz de Informações Heurísticas.
- Cálculo do τ_{max}.
- Inicialização da Matriz de Feromônios com o valor do τ_{max}.

2- Construção das Soluções:

- Cada formiga, paralelamente, inicializa a solução parcial com um lote, escolhido aleatoriamente.
- Aplica-se a regra probabilística em todos os lotes restantes.
- A partir das probabilidades calculadas no passo anterior, escolhe-se o próximo lote a fazer parte da solução parcial (através do Algoritmo da Roleta).
- Repete-se os dois passos anteriores até que todos os lotes tenham sido escolhidos, resultando em uma solução completa.

3- Atualização da trilha de Feromônios.

- Cálculo da função de *fitness* para todas as soluções construídas pelas formigas.
- Escolhe-se a formiga que apresentou o maior valor para a função de *fitness* - esta será a melhor formiga.
- A melhor formiga atualiza a trilha de feromônios, respeitando os limites τ_{max} e τ_{min} fixados previamente.

Repete-se as etapas 2 e 3 até o número máximo de iteração, definido previamente, ou até que o sistema estagne. Neste caso, o sistema será reinicializado.

Capítulo 4

Testes Computacionais

Neste capítulo apresentaremos os testes computacionais realizados com o algoritmo prosposto no capítulo anterior, desenvolvido na linguagem C e executado em uma plataforma Intel Pentium(R)4 CPU 2.40GHz e 480 MB de RAM.

4.1 Problemas para teste

Para a realização dos testes aqui apresentados foram geradas 18 classes de problemas através do gerador de problemas de corte unidimensional CUTGEN1 [10].

Os parâmetros utilizados no CUTGEN1 para a geração dos problemas são os mesmos utilizados na aplicação OCSP-GRASP [14]:

m - Representa a quantidade de itens produzidos por lote.

L - Tamanho do objeto bruto a ser cortado.

l_i - Largura do item i=1,...,m.

v_1 - Limite inferior da relação $\frac{l_i}{L}$.

v_2 - Limite superior da relação $\frac{l_i}{L}$.

Dbar - Demanda média para cada item.

ICPRB - Representa a quantidade de lotes processados;

Classe	ICPRB	m	Dbar	v_1	v_2
1	50	10	20	0.01	0.20
2	50	10	20	0.01	0.80
3	50	10	20	0.20	0.80
4	50	10	100	0.01	0.20
5	50	10	100	0.01	0.80
6	50	10	100	0.20	0.80
7	80	10	20	0.01	0.20
8	80	10	20	0.01	0.80
9	80	10	20	0.20	0.80
10	80	10	100	0.01	0.20
11	80	10	100	0.01	0.80
12	80	10	100	0.20	0.80
13	100	10	20	0.01	0.20
14	100	10	20	0.01	0.80
15	100	10	20	0.20	0.80
16	100	10	100	0.01	0.20
17	100	10	100	0.01	0.80
18	100	10	100	0.20	0.80

Tabela 4.1: Parâmetros utilizados no CUTGEN1.

4.1.1 Parâmetros utilizados no algoritmo

Vários são os parâmetros definidos no algoritmo colônia de formigas: o número n de formigas, os parâmetros referentes à importância relativa das informações heurísticas com oposição às informações do feromônio α e β e a evaporação do feromônio ρ. Para a aplicação ACO-OCSP, também é importante definirmos qual o tipo de matriz de informações heurística utilizar e qual o tipo de formiga a ser utilizada para a atualização das trilhas de feromônio. Os valores para tais parâmetros foram definidos através de vários testes computacionais.

O valor obtido para n é equivalente ao número total de itens, que, no caso, foi fixado em 10 para todas as classes testadas. Os melhores valores para α e β estão contidos no intervalo $[1, 9]$, sendo que grande parte das classes ao assumir os valores 1 e 2, respectivamente, encontraram soluções de melhor qualidade (valores encontrados através de testes computacionais). Já o parâmetro ρ foi definido como 0.05 na maioria dos casos, valor este sugerido por Dorigo e Stützle [4] e que, de fato, apresentou melhores resultados.

Apresentamos nas Seções 3.4.1 e 3.4.2 duas formas de construção da Matriz de Informações Heurísticas: Matriz Fixa e Matriz Variável de Informações Heurísticas. A estratégia utilizando a Matriz Variável, apesar de teoricamente mais robusta, visto que leva em consideração todas as perdas apresentadas nos cortes anteriores para a sua construção, apresentou soluções que não foram melhores do que as da Matriz Fixa, e com uma piora em tempo computacional bastante significativa, cerca de 50 vezes o tempo encontrado usando a matriz fixa.

No que se refere ao tipo de formiga a ser utilizada para a atualização das trilhas de feromônio, *best-so-far* ou *best iteration*, observou-se que formigas do tipo *best-so-far* apresentam uma pequena melhora apenas no que se refere a tempo computacional, visto que as soluções encontradas em ambas foram as mesmas. Ao utilizar as duas formigas conjuntamente, observa-se uma piora na qualidade das soluções assim como em tempo computacional.

Além destes parâmetros acima citados, testes foram feitos com o objetivo de comparar o algoritmo puro e o algoritmo somado a uma busca local. Foram aplicadas duas heurísticas de busca local no algoritmo proposto: 2-opt, aplicada após a construção das soluções e a busca local, apresentada na Seção 3.7, acoplada à FFD durante a construção dos padrões de corte.

A aplicação das buscas locais no OCSP-ACO, contrariando o que era esperado, não apresentou nenhuma melhora nos resultados comparados ao algoritmo puro. Encontramos uma piora, não muito significativa, de em média 0.5%. O tempo computacional também sofreu um aumento de quase 50% em média. Isso nos leva a concluir que estas buscas locais não foram muito eficientes para esta aplicação.

4.1.2 ACO comparado ao GRASP

Para medir a qualidade das soluções, foram feitas comparação ao modelo GRASP-OCSP, desenvolvido por Golfeto, Moretti e Salles Neto [14]. Os parâmetros utilizados para os problemas de teste foram apresentados na Seção 4.1.

A Tabela 4.2 apresenta o número de objetos processados, obtidos com a aplicação do métodos ACO (método puro) e GRASP, para cada uma das 18 classes de teste. Para uma comparação aos dois métodos tomamos o valor da variação percentual $\frac{(ACO - GRASP)}{GRASP}$ a qual denominamos GAP.

Classe	ACO	GRASP	GAP
1	330	331	-0.3%
2	1621	1621	0%
3	1726	1724	0.12%
4	763	765	-0,26%
5	2866	2866	0%
6	3914	3914	0%
7	551	553	-0.36%
8	2826	2821	0.177%
9	2792	2789	0.10%
10	1144	1146	-0.17%
11	6181	6177	0.064%
12	7063	7062	0.014%
13	694	696	-0.28%
14	3446	3443	0.08%
15	3857	3852	0.13%
16	1401	1407	-0.43%
17	7199	7199	0%
18	8030	8026	0.05%

Tabela 4.2: Comparativo OCSP-Grasp × OCSP-ACO.

Os valores negativos para o GAP significam que ACO-OCSP apresentou melhor performance quando comparado ao GRASP-OCSP, isto é, processou todos os lotes utilizando um menor número de padrões. Observamos, a partir dos valores do GAP que os resultados são bastante próximo, sendo que a maior variação, em módulo, é 0.43%. Notemos que quatro classes apresentaram valores iguais nas duas estratégias, implicando em um GAP igual a zero.

Os dados da Tabela 4.3 a seguir apresentam os tamanhos dos itens em relação ao tamanho do padrão e a diferença entre o maior item e o menor, todos na forma percentual, para que conjuntos com diferentes escalas possam ser comparados.

Classe	Maior item	Menor item	faixa de variação
1	19,5%	1,3%	18,2%
2	72,7%	3,5%	69,2%
3	73,6%	20,3%	53,3%
4	19,4%	3,1%	16,3%
5	79,9%	10,1%	69,8%
6	65%	31,1%	33,9%
7	17,9%	1,7%	16,2%
8	79,2%	5,3%	73,9%
9	74,6%	21,9%	52,7%
10	19,4%	2%	17,4%
11	77,1%	5%	72,1%
12	79,5%	33,6%	45,9%
13	17%	3,8%	13,2%
14	68,7%	4,2%	64,5%
15	74%	22,7%	51,1%
16	19,6%	2,6%	17%
17	78,8%	10%	68,8%
18	79,2%	29,7%	49,5%

Tabela 4.3: Faixa de variação dos itens.

A partir das Tabelas 4.2 e 4.3 observamos que classes que apresentam um melhor desempenho ao utilizar a estratégia ACO, isto é, com valores de GAP negativo, são compostas por itens que ocupam no máximo 19.6% do padrão. Este fato é evidenciado nos gráficos de dispersão apresentados abaixo.

O gráfico abaixo relaciona os valores ACO-GRASP (obtidos com a diferença entre o número de objetos processados em ACO e GRASP) com os valores percentuais do tamanho do menor item do lote em relação ao tamanho do padrão (terceira coluna da Tabela 4.3.

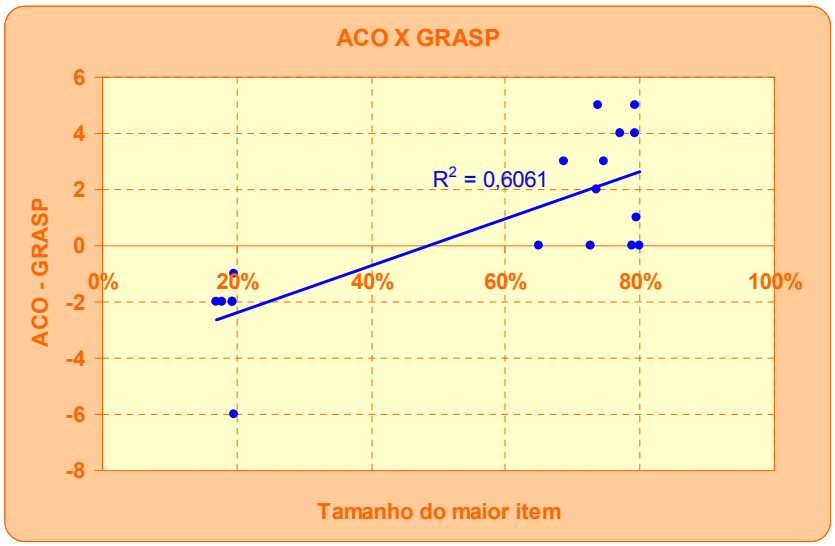


Figura 4.1: Gráfico de dispersão ACO-GRASP × tamanho do maior item.

Os gráficos de dispersão abaixo relacionam os valores de ACO-GRASP e GAP com a faixa de variação entre os tamanhos dos itens.

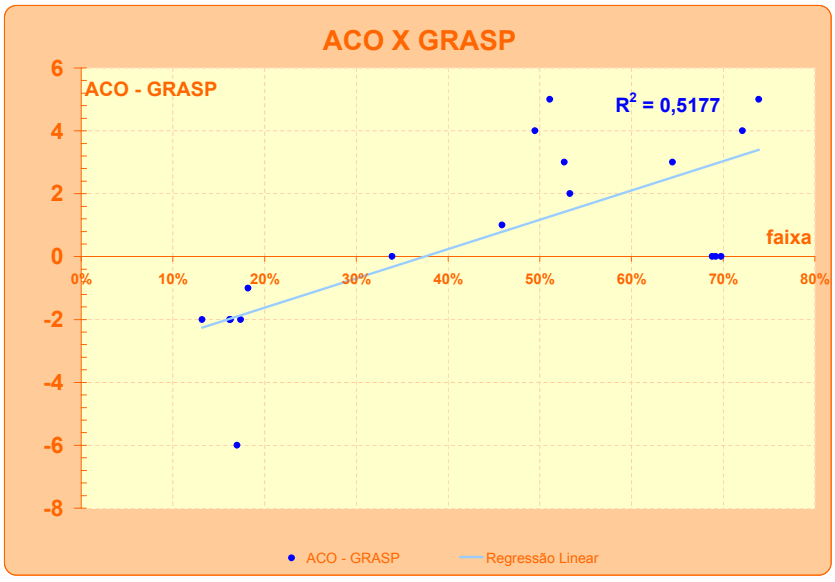


Figura 4.2: Gráfico de dispersão ACO-GRASP × faixa de variação.

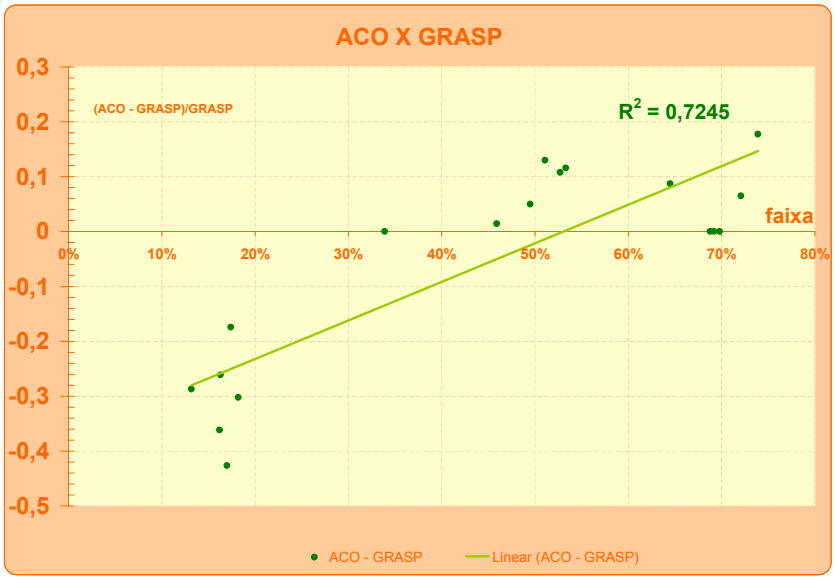


Figura 4.3: Gráfico de dispersão ACO-GRASP/ACO × faixa de variação.

Em suma, de acordo com os valores do GAP apresentados, os resultados entre ACO e GRASP são bastante próximos. No entanto podemos notar, a partir dos gráficos

de dispersão, que o método OCSP-ACO tende a ter um melhor desempenho quando aplicado a lotes com itens de tamanhos menores e também a lotes que apresentem uma faixa de variação entre tamanhos dos itens menor.

Capítulo 5

Conclusão e Perspectivas

A análise dos resultados dos testes computacionais mostrou que o algoritmo proposto obteve bons resultados quando comparado a outras soluções já publicadas na literatura. Seu desempenho tende a ser melhor quando aplicado a lotes cujos itens são menores e também a lotes que apresentam uma faixa de variação dos tamanhos dos itens menor.

Este trabalho apresentou uma nova abordagem para a solução do problema de corte ordenado, usando a aplicação da metaheurística colônia de formigas. Por ser uma abordagem nova, durante a pesquisa realizada, várias possibilidades para futuros trabalhos puderam ser identificadas

- Pode-se aprimorar o desempenho da algoritmo aplicando uma busca local mais agressiva à heurística FFD;
- Novas funções de *fitness* podem ser desenvolvidas e testadas;
- Novas funções relacionando as perdas às informações heurísticas $\eta(i, j)$ podem também ser elaboradas;
- Aplicação de outras variantes do ACO para a resolução do problema;
- Testes de outros limites inferiores e superiores para as trilhas de feromônio $\tau(i, j)$;
- Finalmente, a aplicação e testes do modelo proposto em casos reais.

Bibliografia

[1] Alves, C. e Carvalho, J.M.V. *New programming formulations and an exact algorithm for the ordered cutting stock problem.* Journal of the Operational Research Society, pp. 1-12.2007.

[2] Croes, A. *A Method for Solving Traveling-Salesman Problem.* Opns. Res. 3, 791-812. 1958.

[3] Deneubourg, J.-L., Aron, S., Goss, S., Pasteels, J.-M. *Theself-organizing exploratory pattern of the Argentine ant.* Journal of Insect Behavior, vol.3, pp.159–168, 1990.

[4] Dorigo, M. e Stützle, T. *Ant Colony Optimization.* A Bradford Book, 2004.

[5] Dorigo, M., Maniezzo, V., e Colorni, A. *The ant system: Optimization by a colony of cooperating agents.* IEEE Transactions on Systems, Man, and Cybernetics B, 26(1):29–41, 1996

[6] Dyckhoff, H. *A typology of cutting and packing problems.* European Journal of Operation Research, vol. 444, pp. 145-159, 1990.

[7] Falkenauer, E. e Delchambre, A. A genetic algorithm for bin packing and line balancing. *In Proceedings of the 1992 IEEE International Conference on Robotics and Automation.* May 10–15, 1992, Nice, France. Los Alamitos, CA: IEEE Computer Society Press, 1186–1192. 1992.

[8] Feo T.A. e Resende M.G.C.. A probabilistic heuristic for a computationally difficult set covering problem. *Operations Research Letters.* Vol. 8, pp. 67-71. 1989.

[9] Festa P. e Resende M .G. C.. *GRASP: An annotated bibliography.* In C.C. Ribeiro and P. Hansen, editors, Essays and Survey in Metaheuristics, pp. 325-367, 2002.

[10] Gau T. e Wascher G. *CUTGEN1: A Problem Generator for the Standard One-dimensional Cutting Stock Problem.* European Journal of Operational Research, vol. 84, pp.572-579, 1995.

[11] Gilmore,P.C. e Gomory, R.E. *A Linear Programming Approach to the Cutting Stock Problem.* Operations Research. vol.9, pp. 849-859, 1961.

[12] Glover,F., Laguna, M. e Marti, R. *Scatter Search and Path Relinking: Advances and Aplications Book Series* International Series in Operations Research and Management Science, vol. 57, pp 1-35, 2000.

[13] Goldberg, D.E. *Genetic Algorithms in Search, Optimization and Machine Learning.* Addison-Wesley, Berkeley, 1989.

[14] Golfeto R. R., Moretti A.C. e Salles Neto L. L.. *A GRASP Metaheuristic for the Ordered Cutting Stock Problem* Ingeniare. Revista chilena de ingeniería, vol. 16 Nº 3, pp. 421-427, 2008.

[15] Goss, S., Aron, S., Deneubourg, J. L., e Pasteels, J. M. *Self-organized shortcuts in the Argentine ant.* Naturwissenschaften, vol.76, pp.579–581, 1989

[16] Hinxman, A. *The trim-loss and assortment problems: a survey.* European Journal of Operational Research, vol.5, pp.8-18, 1980.

[17] Holland, J.H. *Genetic Algorithm.* Scientific American, 267(1), 66-72.

[18] Johnson, D. S. e McGeoch, L. A. The travelling salesman problem: A study in local optimization. *Local Search in Combinatorial Optimization.* pp.215-310. Chichester, UK, Jonh Wiley and Sons.

[19] Kantorovich, L.V. *Mathematical Methods of Organizing and Planning Production.* Management Science. vol.6, pp. 366-422, 1960.

[20] Laguna M. e Martí R.. *GRASP and Path Relinking for 2-Layer Straight Line Crossing Minimization* INFORMS Journal on Computing, vol.11, pp. 44-52, 1999.

[21] Levine, J. e Ducatelle F. Ant Colony Optimisation and local search for bin packing and cutting stock problems. *Journal of the Operational Research Society*, vol. 83, 2003

[22] Liang, K.-H., Yao, X., Newton, C., Hoffman, D. A new evolutionary approach to cutting stock problems with and without contiguity. *Computers and Operations Research.*Volume 29 , Issue 12. table of contents Pages: 1641 - 1659. 2001

[23] Lin, S. An Effective Heuristic Algorithm for the Traveling-Salesman Problem. *Bell System Technology Journal, 44:2245–2269, 1965.*

[24] Lin, S. e Kernighan, B. W. An Effective Heuristic Algorithm for the Traveling-Salesman Problem. *Operations Research, 21(2):498516, 1973.*

[25] Linhares, A. e Yanasse H.H. Connections between cutting-pattern sequencing, VLSI design, and flexible machines. *Computers and Operations Research, Volume 29, Number 12, pp. 1759-1772(14),October 2002.*

[26] Moretti A.C., Salles Neto L. L. Nonlinear cutting stock problem model to minimize the number of different patterns and objects. *Computational & Applied Mathematics* vol.27 no.1 Petrópolis 2008.

[27] Rasgsdale, C. e Zobel, C. *Ordered Cutting Stock Problem.* Decision Science, vol. 35, pp. 83-100, 2004.

[28] Renaud J., Boctor F., Ouenniche J. A heuristic for the pickup and delivery traveling salesman problem. *Computers and Operations Research.*27(9), 905–916.

[29] Salles Neto L.L. Modelo Não Linear para Minimizar o Número de Objetos Processados e o Setup num Problema de Corte Unidimensional Tese de doutorado.2005

[30] Slack N., Chambers S.,Johnston R. *Administração da Produção.* Editora Atlas S.A, 2002.

[31] Stadtler, H. *A one-dimensional cutting stock problem in the aluminium industry and its solution.* European Journal of Operational Research, vol.44, pp.209-223, 1990.

[32] Stützle, T. e Hoos, H. H. MAX-MIN Ant System. *Future Generation Computer Systems*, vol.16(8), pp.889–914, 2000.

[33] Yanasse, H.H. On a pattern sequencing problem to minimize the maximum number of open stacks. *European Journal of Operational Research.*.Elsevier, vol. 100(3), pages 454-463.