Boletim Técnico da Escola Politécnica da USP
Departamento de Engenharia de Telecomunicações
e Controle

ISSN 1517-3550

BT/PTC/0116

Aplicando a Técnica de Times Assíncronos na Otimização de Problemas de Empacotamento Unidimensional

Reinaldo de Bernardi
Tsen Chung Kang

São Paulo – 2001

O presente trabalho é um resumo da dissertação de mestrado apresentada por Reinaldo de Bernardi sob orientação do Prof. Dr. Tsen Chung Kang.: "Aplicando Técnica de Times Assíncronos na Otimização de Problemas de Empacotamento Unidimensional", defendida em 04/05/01, na Escola Politécnica.

A íntegra da dissertação encontra-se à disposição com o autor e na Biblioteca de Engenharia Elétrica da Escola Politécnica/USP.

FICHA CATALOGRÁFICA

Bernardi, Reinaldo de
Aplicando a técnica de times assíncronos na otimização de problemas de empacotamento unidimensional / R. de Bernardi, T.C. Kang. – São Paulo: EPUSP, 2001.
12 p. – (Boletim Técnico da Escola Politécnica da USP, Departamento de Engenharia de Telecomunicações e Controle, BT/PTC/0116)

1. Empacotamento unidimensional 2. Otimização matemática I. Kang, Tsen Chung II. Universidade de São Paulo. Escola Politécnica. Departamento de Engenharia de Telecomunicações e Controle III. Título IV. Série

ISSN 1517-3550 CDD 621.381046
519.3

Aplicando a Técnica de Times Assíncronos na Otimização de Problemas de Empacotamento Unidimensional

Reinaldo de Bernardi e Tsen Chung Kang
Escola Politécnica da Universidade de São Paulo

1 – Introdução

Otimização é o processo de encontrar a melhor solução (ou solução ótima) de um conjunto de soluções para um problema. Existe um conjunto particular de problemas nos quais é decisiva a aplicação de um procedimento de otimização. Muitos processos podem se beneficiar de uma alocação otimizada de recursos. Esses recursos, que podem incluir capital, equipamentos, tarefas, ou até mesmo tempo, devem ser cuidadosamente alocados nas quantidades corretas, nos tempos corretos, e na seqüência correta para a obtenção do melhor resultado possível. São problemas complexos, muitas vezes de difícil solução, e que envolvem significativas reduções de custos, melhorias de tempos de processos, ou uma melhor alocação de recursos em atividades. Essa classe de problemas que, normalmente, encarregam-se de alocar recursos escassos são conhecidos por Problemas de Otimização Combinatória [PAR88].

Nessa classe de problemas, a otimização trata basicamente do estudo matemático para encontrar um arranjo, agrupamento ou seleção ótima de objetos discretos, não permitindo, portanto, a utilização dos métodos clássicos de Otimização Contínua [RAO78]. Não se pode, por exemplo, empregar os métodos do cálculo diferencial e fazer derivadas iguais a zero, pois estamos em uma situação combinatória e nosso espaço de soluções não é contínuo, mas sim um conjunto de soluções discretas, sendo que o problema envolve decisões lógicas que não podem ser modeladas apropriadamente como contínuas.

Dentre os problemas clássicos da Otimização Combinatória encontra-se o Problema de Empacotamento (do inglês, *Bin-Packing Problem*) que foi extensivamente estudado por ser um problema classificado como difícil [JOH73], onde a possibilidade de existir algoritmos exatos que os resolvam em tempo de execução razoável é muito pequena. Com isso, faz-se necessário o uso de algoritmos aproximados que buscam encontrar soluções muito próximas da ótima em tempos de execução razoáveis (heurísticas).

1.1 - Problema de Empacotamento

A denominação genérica de Problema de Empacotamento consiste, basicamente, na determinação de padrões de divisão (ou combinação) de objetos de maneira a produzir um conjunto de unidades menores, satisfazendo determinadas restrições. Dependendo do tipo de objeto (barras, placas, caixas), temos os denominados Problemas de Empacotamento Unidimensional, Bidimensional e Tridimensional.

1.1.1 - Empacotamento Unidimensional

Num típico Problema de Empacotamento Unidimensional, é desejado particionar uma lista de itens em sublistas de maneira a minimizar o número de partições respeitando a capacidade de cada sublista. Ou como exemplo: deseja-se alocar um conjunto de tarefas (sem restrição de precedência) em um conjunto de máquinas de modo a minimizar o tempo de finalização de todas as tarefas, utilizando o menor número de máquinas possível.

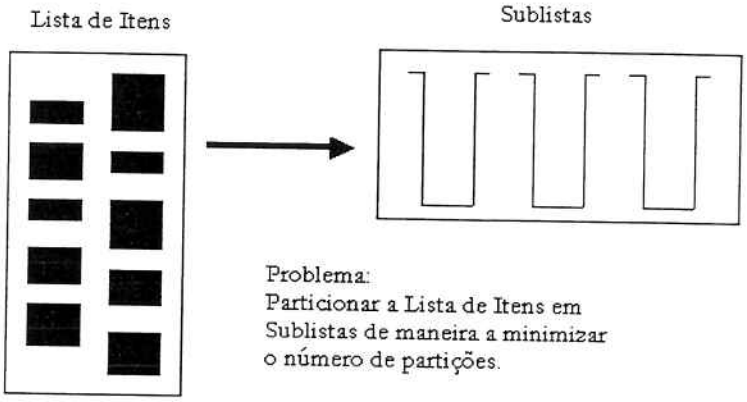

Figura 1. Problema de Empacotamento Unidimensional.

1.1.2 - Empacotamento Bidimensional

O Problema de Empacotamento Bidimensional pode ser descrito como:
Dispor objetos bidimensionais de diferentes áreas (p_1, p_2, ..., p_n) dentro de um outro objeto também bidimensional de área fixa, de tal forma que não haja sobreposição dos mesmos e que a área total desta disposição seja mínima.

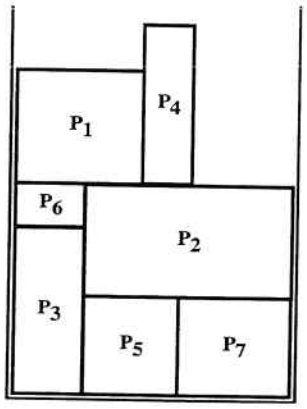

Figura 2. Problema de Empacotamento Bidimensional.

1.1.3 - Empacotamento Tridimensional

No Problema de Empacotamento Tridimensional, deseja-se dispor objetos tridimensionais de diferentes volumes (c_1, c_2, ..., c_n) dentro de um outro objeto também tridimensional de volume fixo (c_F), de tal forma que o volume total desta disposição seja mínimo.

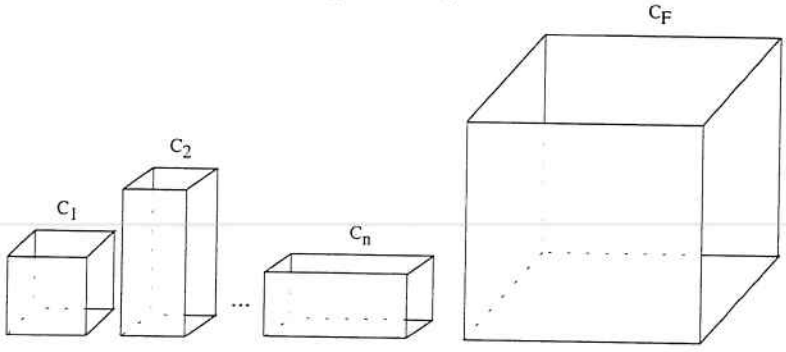

Figura 3. Problema de Empacotamento Tridimensional.

1.2 - O Problema e o Objetivo

Como descrito anteriormente, o Problema de Empacotamento pode ser classificado, dependendo das características dos objetos, como Unidimensional, Bidimensional e Tridimensional.
Neste trabalho estará sendo estudada a versão Unidimensional do Problema de Empacotamento, onde o objetivo da otimização é minimizar o número de sublistas geradas. Este problema é conhecido pela dificuldade de solução devido à explosão fatorial das soluções, sendo caracterizado como problema do tipo $\mathcal{NP}$-difícil [GAR79].
Uma das grandes barreiras que se enfrenta nesse tipo de problema de otimização discreta é a ausência de algoritmos que sejam confiáveis e rápidos. Os mais rigorosos são lentos ou consomem muitos recursos computacionais, enquanto os heurísticos não são muito confiáveis, sendo sensíveis à instância do problema: produzem bons resultados para alguns casos e falham em outros. Entretanto, nem sempre essa identificação é realizável, sendo que em muitos casos essa determinação torna-se um problema de solução tão difícil quanto à própria otimização buscada.

O objetivo deste trabalho não é desenvolver novos algoritmos para a resolução do problema descrito anteriormente, mas combinar alguns algoritmos já existentes com algumas simples heurísticas, na tentativa de que eles cooperem na geração de melhores resultados. A questão é então como combiná-los, de modo que eles interajam e produzam uma solução aceitável.
Foi escolhida, então, uma nova técnica para resolver essa questão, que é a denominada Times Assíncronos (*Asynchronous Teams* – A-Teams) [TAL96]. Nessa técnica, cada algoritmo torna-se um agente e, estes agentes são agrupados em fluxos de dados fortemente cíclicos.
Os agentes podem desempenhar dois papéis diferentes: ou podem gerar novas soluções viáveis, ou podem eliminar soluções menos viáveis (ou que não atendam a algum critério). Aos primeiros dá-se o nome de construtores, e aos últimos dá-se o nome de destruidores. A experiência com essa técnica tem mostrado que algoritmos trabalhando conjuntamente são capazes de produzir resultados muito melhores daqueles que seriam obtidos caso estivessem trabalhando sozinhos [KAN97].

1.3 - A-Teams

1.3.1 - Definição

Uma organização, como definido por Talukdar [TAL92], é uma estrutura na qual agentes tais como: pássaros, peixes, humanos, livros e programas de computadores (softwares) são combinados para formar superagentes, tais como: bandos de pássaros, cardumes de peixes, corporações, bibliotecas e sistemas distribuídos de computador.
Utilizando o contexto do modelo organizacional apresentado acima, adotou-se a definição de A-Teams apresentada por Talukdar [TAL96], onde, se diz que um A-Team é qualquer superagente no qual os agentes são autônomos, os quais comunicam-se assincronamente e o fluxo de dados é cíclico.

1.3.2 - Características

As mais importantes características que distinguem A-Teams são [TAL92]:

- **agentes autônomos**: se um agente possui controle de sua seleção de entradas, escalonamento e alocação de recursos, então este é denominado como um agente autônomo. Como agentes autônomos são completamente independentes de qualquer outro agente, novos agentes podem ser adicionados ou agentes existentes podem ser eliminados sem qualquer notificação para os outros agentes existentes ou para o sistema de supervisão.
- **comunicações assíncronas**: agentes podem ler e escrever informações em memórias compartilhadas sem qualquer sincronismo entre eles. Não existe dependência lógica para acessar as memórias compartilhadas, exceto a integridade dos dados. A autonomia de agentes adicionados em comunicações assíncronas permite que os agentes trabalhem em paralelo todo o tempo.
- **fluxo de dados cíclico**: agentes recuperam, modificam e armazenam informações continuamente nas memórias compartilhadas. Um fluxo de dados cíclico contribui para um fluxo contínuo de

modificações que são transportadas pelos agentes, de forma que interação e realimentação entre agentes sejam possíveis.

Em um A-Team, cada algoritmo tem a oportunidade de apresentar melhores soluções que, por sua vez, podem ser utilizadas como entrada por um outro algoritmo existente na organização. Um A-Team permite a cooperação entre os algoritmos, aumentando as chances de geração de melhores soluções do que cada algoritmo por si só. Devido ao fato de que A-Teams não são hierárquicos, não existe a necessidade de processos de gerenciamento para controlar a maneira como os algoritmos cooperam; isto é, membros podem ser facilmente incluídos ou excluídos de um A-Team a qualquer momento, e, enquanto executados, estes estão sob seu próprio controle. Em capítulos seguintes serão explanados diversos aspectos e parâmetros relevantes no desenvolvimento e utilização de A-Teams.

2 - O Problema de Empacotamento Unidimensional

2.1 - Definições

O Problema de Empacotamento Unidimensional (PEU) pode ser representado por uma formulação matemática, conforme apresentado abaixo [MAR90]:

Dados n itens e n sublistas, com:

$w(p_i)$ = valor associado ao item p_i
c = capacidade das sublistas

alocar cada item a uma sublista de maneira que o valor associado total dos itens em cada sublista não exceda a capacidade da mesma e o número de sublistas utilizadas (m) seja mínimo, ou seja:

minimizar: $$m = \sum_{i=1}^{n} y_i \qquad (2.1)$$

sujeito à: $$\sum_{j=1}^{n} w(p_j)x_{ij} \le cy_i, \quad i \in N = \{1, ..., n\}, \qquad (2.2)$$

$$\sum_{i=1}^{n} x_{ij} = 1, \quad j \in N, \qquad (2.3)$$

$$y_i = 0 \text{ ou } 1, \quad i \in N, \qquad (2.4)$$

$$x_{ij} = 0 \text{ ou } 1, \quad i \in N,\ j \in N, \qquad (2.5)$$

onde:

$$y_i = \begin{cases} 1 & \textit{se a sublista } i \textit{ é utilizada}; \\ 0 & \textit{caso contrário}, \end{cases}$$

$$x_{ij} = \begin{cases} 1 & \textit{se o item } j \textit{ está alocado à sublista } i; \\ 0 & \textit{caso contrário}. \end{cases}$$

2.2 - Relaxações baseadas no Limite Inferior (*Lower Bound*)

Para o modelo do PEU apresentado no tópico anterior, a relaxação contínua C(PEU) do problema, dados por (2.1)-(2.3) e

$$0 \le y_i \le 1, \quad i \in N,$$
$$0 \le x_{ij} \le 1, \quad i \in N,\ j \in N,$$

podem ser imediatamente resolvidos [MAR90] pelos valores $x_{ii} = 1, x_{ij} = 0 \ \ (j \neq i)$ e $y_i = w(p_i)/c$ para $i \in N$. Portanto

$$OPT(C(PEU)) = \sum_{i=1}^{n} w(p_i)/c$$

então, um limite inferior para o PEU é: $A_1 = \left\lceil \sum_{j=1}^{n} w(p_j)/c \right\rceil$

Apesar da simplicidade, pode-se esperar que A_1 apresente um comportamento médio aceitável para problemas nos quais, os tamanhos dos itens são suficientemente pequenos em relação à capacidade total das sublistas.

3 - Um A-Team para Otimização do PEU

3.1 - Introdução

Conforme explanado, o objetivo da otimização do PEU é minimizar as sublistas geradas, porém durante a implementação do APEU, optou-se pela análise de uma outra maneira, simplificando assim a codificação.

Uma vez que se tem os dados do problema, número de itens e capacidade de cada sublista, calcula-se o limite inferior, limite este que corresponde ao número mínimo de sublistas a serem utilizadas. Cria-se, então, estas sublistas e aloca-se os itens dividindo-os entre as sublistas. O objetivo da otimização torna-se, então, a minimização da somatória dos excedentes de todas as sublistas e, quando este valor é igual ou menor a zero, tem-se um empacotamento ótimo.

3.2 - As Memórias

O APEU possui uma memória principal e uma memória de descrição do problema que é utilizada pelo agente de inicialização

A memória principal é utilizada como entrada e saída, ou seja, a mesma permite que soluções sejam lidas e outras sejam escritas na mesma.

3.3 - Os Agentes

Quanto ao o desenvolvimento dos agentes, havia duas opções: codificar algoritmos já existentes ou desenvolver novas heurísticas. Neste trabalho foi optado por se desenvolver novas heurísticas, quanto mais simples possível, na tentativa de avaliar o comportamento de A-Teams.

A implementação do APEU é composta de um total de 19 agentes, agentes estes, que estão divididos da seguinte maneira:

- Agente de Inicialização;
- 17 Agentes de Modificação, divididos em dois grupos: Agentes de Alocação e Agentes de Permutação;
- Agente de Destruição.

O objetivo e funcionamento dos agentes são explicados na seqüência.

3.3.1 - Mecanismo de Planejamento

O mecanismo de planejamento foi implementado conforme descrito a seguir:

- O Agente de Inicialização é executado somente uma única vez no início de cada simulação;
- Dentre os Agentes de Modificação (Alocação e Permutação), apenas um é executado a cada iteração.

A escolha de qual agente estará ativo a cada iteração é feita randomicamente. Este processo se repete até o fim da otimização;

• O Agente Destruidor é executado apenas uma vez a cada iteração após a execução de um Agente de Modificação.

3.3.2 - Mecanismo de Seleção

O mecanismo de seleção foi implementado conforme descrito a seguir, com exceção do Agente de Inicialização que não possui este mecanismo (pois a seleção de qual problema deverá tratar é passado pelo usuário), assim sendo:

• Todos os Agentes de Modificação do APEU possuem mecanismo de seleção randômica, isto é, os próprios agentes selecionam randomicamente na memória de entrada de dados, a solução que será utilizada (sublista a ser tratada).

• Como o Agente de Destruição é sempre executado após a execução de um Agente de Modificação, este utiliza a mesma memória.

3.3.3 - Agente de Inicialização

O Agente de Inicialização do APEU tem por objetivo gerar uma solução inicial, a partir de um problema entrado pelo usuário. Nesta fase, este agente não se preocupa se a solução gerada é viável ou não, simplesmente aloca os itens seqüencialmente em um número de sublistas calculado pelo limite inferior (*lower bound*).

3.3.4- Agentes de Alocação

Os agentes de alocação do APEU são ao todo nove heurísticas. Um agente de alocação move a cada iteração apenas um item entre duas sublistas, na tentativa de que na primeira sublista seja respeitada a restrição de capacidade.

Os agentes foram nomeados de AA1 à AA9 e uma descrição dos mesmos é apresentada abaixo.

Agente AA1

• Seleciona randomicamente uma sublista com restrição de capacidade não respeitada;

• Gera a partir desta sublista, um conjunto contendo todos os itens cujo tamanho seja maior ou igual ao excedente da mesma;

• Seleciona no conjunto gerado, o item de menor tamanho;

• Seleciona dentre as todas as sublistas, a que possui a menor ocupação;

• Move o item selecionado para a sublista selecionada.

Agente AA2

• Seleciona randomicamente uma sublista com restrição de capacidade não respeitada;

• Gera a partir desta sublista, um conjunto contendo todos os itens cujo tamanho seja maior ou igual ao excedente da mesma;

• Seleciona no conjunto gerado, o item de menor tamanho;

• Seleciona randomicamente, dentre todas as sublistas, uma outra sublista com restrição de capacidade respeitada;

• Move o item selecionado para a sublista selecionada.

Agente AA3

• Seleciona randomicamente uma sublista com restrição de capacidade não respeitada;

• Gera a partir desta sublista, um conjunto contendo todos os itens cujo tamanho seja maior ou igual ao excedente da mesma;

• Seleciona no conjunto gerado, o item de menor tamanho;

• Move o item selecionado para a próxima sublista com restrição de capacidade respeitada.

Agente AA4

- Seleciona randomicamente uma sublista com restrição de capacidade não respeitada;
- Gera a partir desta sublista, um conjunto contendo todos os itens cujo tamanho seja maior ou igual ao excedente da mesma;
- Seleciona no conjunto gerado, o item de menor tamanho;
- Move o item selecionado para a sublista anterior com restrição de capacidade respeitada

Agente AA5
- Seleciona randomicamente uma sublista com restrição de capacidade não respeitada;
- Seleciona randomicamente um item desta sublista;
- Seleciona dentre as todas as sublistas, a que possui a menor ocupação;
- Move o item selecionado para a sublista selecionada.

Agente AA6
- Seleciona randomicamente uma sublista com restrição de capacidade não respeitada;
- Seleciona randomicamente um item desta sublista;
- Seleciona randomicamente, dentre todas as sublistas, uma outra sublista com restrição de capacidade respeitada;
- Move o item selecionado para a sublista selecionada.

Agente AA7
- Seleciona randomicamente uma sublista com restrição de capacidade não respeitada;
- Seleciona randomicamente um item desta sublista;
- Move o item selecionado para a próxima sublista com restrição de capacidade respeitada.

Agente AA8
- Seleciona randomicamente uma sublista com restrição de capacidade não respeitada;
- Seleciona randomicamente um item desta sublista;
- Move o item selecionado para a sublista anterior com restrição de capacidade respeitada

Agente AA9
- Seleciona randomicamente uma sublista com restrição de capacidade não respeitada;
- Seleciona nesta sublista, o item de menor tamanho;
- Seleciona dentre as todas as sublistas, a que possui a menor ocupação;
- Move o item selecionado para a sublista selecionada.

3.3.5 - Agentes de Permutação

Os agentes de permutação do APEU são ao todo oito heurísticas. Um agente de permutação troca a cada iteração dois itens, um de cada sublista, entre si, na tentativa de que na primeira sublista seja respeitada a restrição de capacidade.
Os agentes foram nomeados de AP1 à AP8 e uma descrição dos mesmos é apresentada abaixo.

Agente AP1
- Seleciona randomicamente uma sublista com restrição de capacidade não respeitada;
- Seleciona nesta sublista, o item de menor tamanho;
- Seleciona dentre as todas as sublistas, a que possui a menor ocupação;
- Seleciona nesta sublista, o item de menor tamanho;
- Troca entre as sublistas selecionadas, os itens selecionados.

Agente AP2
- Seleciona randomicamente uma sublista com restrição de capacidade não respeitada;

- Seleciona nesta sublista, o item de maior tamanho;
- Seleciona dentre as todas as sublistas, a que possui a menor ocupação;
- Seleciona nesta sublista, o item de maior tamanho;
- Troca entre as sublistas selecionadas, os itens selecionados.

Agente AP3
- Seleciona randomicamente uma sublista com restrição de capacidade não respeitada;
- Seleciona nesta sublista, o item de maior tamanho;
- Seleciona dentre as todas as sublistas, a que possui a menor ocupação;
- Seleciona nesta sublista, o item de menor tamanho;
- Troca entre as sublistas selecionadas, os itens selecionados.

Agente AP4
- Seleciona randomicamente uma sublista com restrição de capacidade não respeitada;
- Seleciona nesta sublista, o item de maior tamanho;
- Seleciona dentre as todas as sublistas, a que possui a maior ocupação, porém com a restrição de capacidade respeitada;
- Seleciona nesta sublista, o item de maior tamanho;
- Troca entre as sublistas selecionadas, os itens selecionados.

Agente AP5
- Seleciona randomicamente uma sublista com restrição de capacidade não respeitada;
- Seleciona nesta sublista, o item de maior tamanho;
- Seleciona dentre as todas as sublistas, a que possui a maior ocupação, porém com a restrição de capacidade respeitada;
- Seleciona nesta sublista, o item de menor tamanho;
- Troca entre as sublistas selecionadas, os itens selecionados.

Agente AP6
- Seleciona randomicamente duas sublistas;
- Para cada sublista selecionada, seleciona randomicamente um item;
- Troca entre as sublistas selecionadas, os itens selecionados.

Agente AP7
- Seleciona randomicamente uma sublista com restrição de capacidade não respeitada;
- Gera a partir desta sublista, um conjunto contendo todos os itens cujo tamanho seja maior ou igual ao excedente da mesma;
- Seleciona no conjunto gerado, o item de menor tamanho;
- Seleciona, dentre todos os itens do problema inicial, o item de menor tamanho;
- Seleciona a sublista, cujo item selecionado está alocado;
- Troca entre as sublistas selecionadas, os itens selecionados.

Agente AP8
- Seleciona randomicamente uma sublista com restrição de capacidade não respeitada;
- Seleciona randomicamente um item desta sublista;
- Seleciona randomicamente uma outra sublista;
- Seleciona randomicamente um item desta sublista;
- Troca entre as sublistas selecionadas, os itens selecionados.

3.3.6 - Agente de Destruição

O agente de destruição do APEU tem por objetivo não permitir que uma solução pior que a atual contida na memória, seja armazenada. Para tal, o agente faz a seguinte análise:

- Faz a somatória do excedente da capacidade de cada sublista.
- Compara o valor obtido com o valor existente anteriormente
- Se este número for maior que o existente, elimina a solução.

O agente de destruição elimina soluções consideradas "ruins" mantendo as "boas", fazendo assim, com que o conjunto de soluções se direcionem para um conjunto ótimo.

3.4 - Fluxo de Dados do APEU

Após a apresentação da descrição de cada componente do A-Team desenvolvido, é apresentado agora o Fluxo de Dados do APEU, mostrando as conexões existentes entre cada um deles.

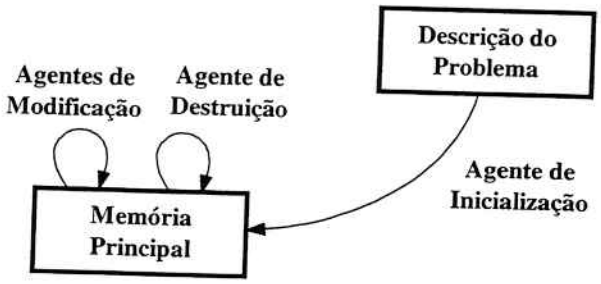

Figura 4. O fluxo de dados do APEU.

3.5 - Critério de Avaliação de Resultados

O critério de avaliação de uma solução, adotado na implementação do APEU, é a descrito a seguir: Um menor valor da somatória dos valores excedentes à capacidade de cada sublista implica em uma melhor solução. Uma solução ótima é reconhecida quando esse valor é igual ou menor a zero.

3.6 - Critério de Parada

Existem implementados no APEU dois critérios de parada. Um deles, que é o desejável, ocorre quando o critério de avaliação passa a ser menor ou igual a zero, indicando uma solução ótima. O outro critério de parada ocorre quando 65530 iterações foram executadas e nenhuma melhora foi observada no critério de avaliação. Neste caso, o APEU finaliza a otimização e indica o número de sublistas adicionais necessárias.

4 - Experimentos e Análise dos Dados

4.1 - Introdução

Para analisar a performance do APEU desenvolvido, foram realizadas simulações com problemas encontrados no *site* pertencente à *The Operational Research Library* – http://mscmga.ms.ic.ac.uk/info.html. Estes problemas, que estão divididos em duas classes, foram gerados e estudados por Falkenauer [FAL96] para testar um algoritmo genético criado por ele.
Uma das classes destes problemas, denominada *uniform*, foi gerada randomicamente sem nenhum ótimo provável, porém, um limite inferior (*lower bound*) pode ser calculado. A outra classe é a denominada *triplet*, onde os problemas foram gerados com soluções pré-determinadas conhecendo-se assim as soluções ótimas.

A justificativa para o desenvolvimento, por Falkenauer, da classe de problemas *triplet*, foi o fato de que não se podia determinar se seu algoritmo era capaz de produzir soluções ótimas globais nos problemas gerados randomicamente (classe *uniform*), garantia que provou com a classe *triplet*.
Na classe *uniform*, existem 4 tamanhos de problemas com listas contendo 120, 250, 500 e 1000 itens, onde, o tamanho dos itens são valores gerados randomicamente entre [20, 100].
A classe *triplet*, contém, também, 4 tamanhos de problemas, porém, com listas de 60, 120, 249 e 501 itens. Cada sublista gerada nesta classe de problema conterá exatamente 3 itens que completará totalmente a sublista. O tamanho do primeiro item é gerado randomicamente e deverá possuir um valor entre ¼ e ½ da capacidade da sublista. O segundo objeto é gerado randomicamente e deverá possuir um valor tal que preencha ½ da capacidade restante da sublista. O último item simplesmente completa a capacidade da sublista.
Para cada tamanho de problema, tanto da classe *uniform* quanto para a classe *triplet*, existem 20 exemplos.

4.2 - Resultados Recentes

No trabalho de Falkenauer [FAL96] é apresentado o desempenho de um algoritmo genético, denominado *hybrid grouping genetic algorithm* – GGA, para os problemas descritos no tópico anterior. O GGA é um algoritmo genético modificado que incorpora algoritmos de busca local e o algoritmo *First Fit Decreasing* [JOH73].
Para ambas as classes, *uniform* e *triplet*, Falkenauer relata que o GGA apresenta um melhor desempenho em termos de qualidade da solução (número de sublistas utilizadas) e tempo de execução quando comparado com o algoritmo MTP de Martello e Toth (algoritmo baseado nas estratégias *First Fit Decreasing* e *Best Fit Decreasing*). Falkenauer explana, também, que a eficácia do GGA é acentuada quando o número de itens aumenta. Seus resultados mostram que o GGA gera um menor número de sublistas e foi capaz de produzir uma solução em um tempo menor que um procedimento *branch and bound*. Por outro lado, em dois problemas de 120 itens e em três problemas de 250 itens da classe *uniform*, ele reporta que tanto o GGA quanto o procedimento *branch and bound* não foram capazes de gerar uma solução igual ao limite inferior (*lower bound*).
Falkenauer comparou o desempenho de seu algoritmo com um procedimento *branch and bound* em termos de qualidade da solução (número de sublistas utilizadas) e tempo de execução. O propósito desta comparação foi mostrar que o GGA apresenta-se como um dos melhores algoritmos para o problema de empacotamento unidimensional.

4.3 - Simulações e Desempenho do APEU

Cada um dos 20 problemas dos quatro tamanhos da classe *uniform* e dos quatro tamanhos da classe *triplet* foram avaliados através de 50 execuções de cada problema.
A verificação do desempenho do APEU, bem como uma comparação com o algoritmo de melhor desempenho existente (GGA) são apresentados no próximo tópico.

4.4 - Análise dos Dados

Neste tópico é apresentada uma análise dos dados obtidos através das simulações. As comparações são feitas em relação ao desempenho (quanto ao número de sublistas utilizadas) do algoritmo GGA.
Para os problemas de tamanho de listas com 120 itens, o APEU apresentou soluções com resultados iguais aos apresentados pelo GGA em 16 dos 20 problemas analisados, resultados estes que representam os melhores limites inferiores conhecidos. Outras 3 soluções necessitam 1 sublista adicional e 1 solução apresentou um melhoria, utilizando 1 sublista a menos. Nos problemas de tamanho de listas com 250 itens, o APEU apresentou soluções com resultados iguais aos do GGA em 15 dos 20 problemas analisados, para os problemas restantes, 1 sublista adicional foi necessária. Nos problemas de tamanho 500, os resultados apresentados se igualaram em 16 problemas, os 4 restantes necessitaram 1 sublista adicional. Finalmente, para a classe *uniform*, para os problemas de tamanho 1000, o APEU apresentou resultados iguais em 16 problemas e os 4 restantes necessitaram 1 sublista adicional.

Para todos os problemas da classe *triplet* simulados, verificou-se que o APEU apresentou resultado igual ao GGA, resultado este considerado ótimo global.
A tabela 1 apresenta o resumo das comparações de desempenho de todos os oito problemas. O número de vezes que o APEU melhorou, igualou ou necessitou de sublistas adicionais comparados à melhor solução já apresentada (Falkenauer) é mostrado.

Tabela 1. Resumo das comparações de desempenho APEU x GGA.

Problema	Menor	Igual	Maior
U120	1	16	3
U250		15	5
U500		16	4
U1000		16	4
T60		20	
T120		20	
T249		20	
T501		20	

5 - Conclusão

Este trabalho demonstrou que A-Teams apresentam-se como uma ferramenta de grande valor quando aplicados na solução de problemas de otimização combinatorial como o Problema de Empacotamento Unidimensional, fato este, sendo a principal contribuição deste trabalho, pois na literatura pesquisada não se encontrou nenhuma referência à utilização desta técnica em Problemas de Empacotamento.
Também foi demonstrado que A-Teams permitem, com sucesso, a combinação de diversos algoritmos interagindo assincronamente e interativamente para aplicações em problemas considerados complexos.
Um ponto de grande relevância é o que diz respeito aos resultados obtidos e apresentados no tópico 4. Analisando estes resultados verifica-se que mesmo com uma implementação de agentes muito simples, a combinação dos mesmos produz resultados, que na maioria das vezes iguala-se aos melhores resultados até então conhecidos. Com esta análise espera-se que com a implementação de agentes mais poderosos, teríamos um grande melhora no desempenho da implementação.
Não podemos deixar de citar, também, o fato de que se os agentes são simples, a implementação dos mesmos torna-se rápida, implicando assim na obtenção de bons resultados sem um maior dispêndio de tempo desenvolvendo-se algoritmos complicados.

6 – Bibliografia

[COF84] Coffman, E. G; Garey, M. R.; Johnson, D. S., "Approximation algorithms for bin-packing - an update survey". Algorithm Design for Computer System Design, Springer, Vienna.

[FAL96] Falkenauer, E.
"A Hybrid Grouping Genetic Algorithm for Bin Packing"
Journal of Heuristics, 2(2), pp. 5-30

[GAR79] Garey, M. R.; Johnson, D. S., "Computers and Intractability: A Guide to the Theory of NP-Completeness". W. H. Freeman and Company, 1979

[JOH73] Johnson, D. S., "Near-Optimal Bin Packing Algorithms". Technical Report MAC TR-109, Project MAC. Massachusetts Institute of Technology, Cambridge, MA, 1973

[KAN97] Kang, T. C.; Talukdar, S., "Overview on Designing Autonomous Agents Teams (A-Teams)" .Apresentado na Conference on Management and Control of Production and Logistics MCPL'97, Campinas, 1997

[MAR90] Martello, S.; Toth, P., "Knapsack Problems: Algorithms and Computer Implementations". John Wiley & Sons, 1990

[PAR88] Parker, R. G.; Rardin, R. L., "Discrete Optimization". Computer Science and Scientific Computing. Academic Press, Inc., 1988

[RAO78] Rao, S. S., "Optimization - Theory and Applications" - Second Edition. Wiley Eastern Limited, 1978

[TAL92] Talukdar, S.; Souza, P. S. de, "Scale Efficient Organizations". Proceedings of the 1992 IEEE International Conference on Systems, Man and Cybernetics, Chicago, Illinois, Oct. 18-21, 1992.

[TAL96] Talukdar, S.; Baerentzen,L; Gove, A; Souza, P. S. de, "Asynchronous Teams: Cooperation Schemes For Autonomous Agents". Engineering Design Center, Carnegie Mellon University, 1996.

BOLETINS TÉCNICOS - TEXTOS PUBLICADOS

BT/PTC/9901 – Avaliação de Ergoespirômetros Segundo a Norma NBR IEC 601-1- MARIA RUTH C. R. LEITE, JOSÉ CARLOS TEIXEIRA DE B. MORAES

BT/PTC/9902 – Sistemas de Criptofonia de Voz com Mapas Caóticos e Redes Neurais Artificiais – MIGUEL ANTONIO FERNANDES SOLER, EUVALDO FERREIRA CABRAL JR.

BT/PTC/9903 – Regulação Sincronizada de Distúrbios Senodais – VAIDYA INÉS CARRILLO SEGURA, PAULO SÉRGIO PEREIRA DA SILVA

BT/PTC/9904 – Desenvolvimento e Implementação de Algoritmo Computacional para Garantir um Determinado Nível de Letalidade Acumulada para Microorganismos Presentes em Alimentos Industrializados – RUBENS GEDRAITE, CLÁUDIO GARCIA

BT/PTC/9905 - Modelo Operacional de Gestão de Qualidade em Laboratórios de Ensaio e Calibração de Equipamentos Eletromédicos – MANUEL ANTONIO TAPIA LÓPEZ, JOSÉ CARLOS TEIXEIRA DE BARROS MORAES

BT/PTC/9906 – Extração de Componentes Principais de Sinais Cerebrais Usando Karhunen – Loève Neural Network – EDUARDO AKIRA KINTO, EUVALDO F. CABRAL JR.

BT/PTC/9907 – Observador Pseudo-Derivativo de Kalman Numa Coluna de Destilação Binária – JOSÉ HERNANDEZ LÓPEZ, JOSÉ JAIME DA CRUZ, CLAUDIO GARCIA

BT/PTC/9908 – Reconhecimento Automático do Locutor com Coeficientes Mel-Cepstrais e Redes Neurais Artificiais – ANDRÉ BORDIN MAGNI, EUVALDO F. CABRAL JÚNIOR

BT/PTC/9909 – Análise de Estabilidade e Síntese de Sistemas Híbridos – DIEGO COLÓN, FELIPE MIGUEL PAIT

BT/PTC/0001 – Alguns Aspectos de Visão Multiescalas e Multiresolução – JOÃO E. KOGLER JR., MARCIO RILLO

BT/PTC/0002 – Placa de Sinalização E1: Sinalização de Linha R2 Digital Sinalização entre Registradores MFC- PHILLIP MARK SEYMOUR BURT, FERNANDA CARDOSO DA SILVA

BT/PTC/0003 – Estudo da Técnica de Comunicação FO-CDMA em Redes de Fibra Óptica de Alta Velocidade – TULIPA PERSO, JOSÉ ROBERTO DE A. AMAZONAS

BT/PTC/0004 – Avaliação de Modelos Matemáticos para Motoneurônios – DANIEL GUSTAVO GOROSO, ANDRÉ FÁBIO KOHN

BT/PTC/0005 – Extração e Avaliação de Atributos do Eletrocardiograma para Classificação de Batimentos Cardíacos – ELDER VIEIRA COSTA, JOSÉ CARLOS T. DE BARROS MORAES

BT/PTC/0006 – Uma Técnica de Imposição de Zeros para Auxílio em Projeto de Sistemas de Controle – PAULO SÉRGIO PIERRI, ROBERTO MOURA SALES

BT/PTC/0007 – A Connected Multireticulated Diagram Viewer – PAULO EDUARDO PILON, EUVALDO F. CABRAL JÚNIOR

BT/PTC/0008 – Some Geometric Properties of the Dynamic Extension Algorithm – PAULO SÉRGIO PEREIRA DA SILVA

BT/PTC/0009 – Comparison of Alternatives for Capacity Increase in Multiple-Rate Dual-Class DS/CDMA Systems – CYRO SACARANO HESI, PAUL ETIENNE JESZENSKY

BT/PTC/0010 – Reconhecimento Automático de Ações Faciais usando FACS e Redes Neurais Artificiais – ALEXANDRE TORNICE, EUVALDO F. CABRAL JÚNIOR

BT/PTC/0011 – Estudo de Caso: Tornando um Projeto Testável Utilizando Ferramentas Synopsys – REINALDO SILVEIRA, JOSÉ ROBERTO A. AMAZONAS

BT/PTC/0012 – Modelos Probabilísticos para Rastreamento em Carteiras de Investimento – HUGO G. V. DE ASSUNÇÃO, OSWALDO L. V. COSTA

BT/PTC/0013 – Influência de um Controle Imperfeito de Potência e Monitoramento da Atividade Vocal na Capacidade de Sistemas DS/CDMA – MÁRCIO WAGNER DUARTE ROLIM, PAUL JEAN ETIENNE JESZENSKY

BT/PTC/0014 – Canceladores de Interferência Sucessivo e Paralelo para DS/CDMA – TAUFIK ABRÃO, PAUL JEAN E. JESZENSKY

BT/PTC/0015 – Transmissão de Serviços de Multimídia num Sistema Móvel Celular CDMA de Banda Larga – EDUARDO MEIRELLES MASSAUD, PAUL JEAN ETIENNE JESZENSKY

BT/PTC/0016 – Disseminação do HIV em uma População Homossexual Heterogênea – MARCOS CASADO CASTÑO, JOSÉ ROBERTO CASTILHO PIQUEIRA

BT/PTC/0017 – Implementação e Avaliação em Laboratório de um Monitor Cardíaco Portátil para Três Derivações – RAISA FERNÁNDEZ NUNEZ, JOSE CARLOS TEIXEIRA DE BAROS MORAES

BT/PTC/0018 – Projeto de Filtros Recursivos de N-ésima Banda – IRINEU ANTUNES JÚNIOR, MAX GERKEN

BT/PTC/0019 – Relative Flatness and Flatness of Implicit Systems – PAULO SÉRGIO PEREIRA DA SILVA, CARLOS CORRÊA FILHO

BT/PTC/0020 – Estimativa de Fluxo Sangüíneo nas Artérias Coronárias Usando Imagens de Cineangiocardiografia – ANA CRISTINA DOS SANTOS, SÉRGIO SHIGUEMI FURUIE

BT/PTC/0021 – Modelos Populacionais para AIDS e Análise do Equilíbrio sem Epidemia – ELIZABETH FERREIRA SANTOS, JOSÉ ROBERTO CASTILHO PIQUEIRA

BT/PTC/0101 – Model-Based Soft-Sensor Design for On-Line Estimation of the Biological Activity in Activated Sludge Wastewater Treatment Plants – OSCAR A. Z. SOTOMAYOR, SONG WON PARK, CLAUDIO GARCIA

BT/PTC/0102 – Reconhecimento Automático do Locutor Utilizando a Rede Neural Artificial *Field Distributed Memory* – FDM – MARCELO BLANCO,EUVALDO F. CABRAL JR.

BT/PTC/0103 – Algoritmos de Filtragem e Previsão em Modelos de Volatilidade – FERNANDO LOVISOTTO, OSWALDO L. V. COSTA

BT/PTC/0104 – Método de Diferenças Temporais Aplicado às Equações de Riccati Acopladas entre Si – OSWALDO L. V. COSTA, JULIO C. C. AYA

BT/PTC/0105 – Método de Diferenças Finitas e de Monte Carlo em Derivativos – ANDRÉ CURY MAIALI, OSWALDO LUIZ DO VALLE COSTA

BT/PTC/0106 – Resolução de um Problema Inverso de Eletromagnetismo por Meio de Redes Neurais Artificiais – ARNALDO MEGRICH, JORGE MIECZYSLAW JANISZEWSKI

BT/PTC/0107 – Projeto de Controlador de Temperatura para Perfusão Peritoneal com Hipertermia e Quimioterapia – GIANCARLO ANTONIO BERZACOLA, FUAD KASSAB JÚNIOR

BT/PTC/0108 – 0 Papel de Diferentes Grupos Populacionais na Transmissão Sexual do HIV – ELIZABETH FERREIRA SANTOS, JOSÉ ROBERTO CASTILHO PIQUEIRA

BT/PTC/0109 – Terapias Ótimas Anti-HIV para a Redução da Transmissão Vertical – RENATO BEVILACQUA, LUIZ HENRIQUE ALVES MONTEIRO

BT/PTC/0110 – Brain Signal Analysis Using Non-Linear ARIMA Models – ERNANE J. X. COSTA, EUVALDO FERREIRA CABRAL JR.

BT/PTC/0111 – Cancelamento de Eco Acústico Estéreo: Análise de Algoritmos Adaptativos e um novo Método de Redução do Desalinhamento – ROBERTO ROSCHEL BELLI, PHILLIP MARK SEYMOUR BURT

BT/PTC/0112 – Natural Gas Flow Computer With Open Architecture Using Intelligent Instrumentation And Field Bus – OSMEL REYES VAILLANT, CLAUDIO GARCIA

BT/PTC/0113 – Aplicação de Métodos de Inteligência Artificial em Inteligência de Negócios –ROGÉRIO GARCIA DUTRA, EUVALDO FERREIRA CABRAL JR.

BT/PTC/0114 – Detectores Multiusuário para DS/CDMA – Canceladores de Interferência –TAUFIK ABRÃO, PAUL JEAN E. JESZENSKY

BT/PTC/0115 – Reconhecimento Automático do Locutor Usando Pré-Processamento em Sons Nasalizados com Diversos Classificadores Neurais – ROBERTO AMILTON BERNARDES SÓRIA, EUVALDO FERREIRA CABRAL JR.