METAHEURÍSTICA BUSCA TABU PARA O PROBLEMA DE TRANSPORTE COM CARGA FIXA

Márcio Simões de Souza
marcios@dee.feis.unesp.br
marciossouza@uol.com.br

Rubén Romero
ruben@dee.feis.unesp.br
Universidade Estadual Paulista – Faculdade de Engenharia de Ilha Solteira
Departamento de Engenharia Elétrica – Avenida Brasil Norte, 364 – Caixa Postal 31
CEP 15385-000 Ilha Solteira – SP

RESUMO

Este trabalho analisa um algoritmo Busca Tabu para o Problema de Transporte com Carga Fixa. O algoritmo apresenta duas fases: na primeira determina-se uma solução factível para o problema utilizando heurísticas clássicas e, na segunda, é implementado um algoritmo de Busca Tabu para melhorar a qualidade da solução inicial. O algoritmo foi implementado em linguagem Fortran e aplicado à resolução de problemas de testes gerados randomicamente, disponíveis na literatura e com ótimos globais conhecidos, tendo apresentado resultados de boa qualidade com baixo esforço computacional.
Palavras-chave: Problema de transporte com carga fixa, busca tabu, programação linear, fluxo em redes, algoritmos heurísticos.

ABSTRACT

This work presents a Tabu Search Algorithm to the Fixed Charge Transportation Problem. This algorithm consists of two phases: in the first a initial solution was achieved by classics heuristics and in the second the Tabu Search algorithm was implemented for increased the initial solution that one which was achieved previously. This algorithm is applied to solving theoretical problems generated by random processes for which the optimums global were known and the outcome was extremely satisfactory since it resulted in a clear saving of computer effort in relation to solution of medium and small problem.
Keywords: Fixed Charge Transportation Problem; Tabu Search, Linear Programming; Networks Flow; Heuristic Algorithms.

1. – Introdução

Um dos problemas de grande importância para a competitividade das empresas, nos últimos anos, está relacionado com logística. Esses problemas podem estar relacionados com o fornecimento de insumos dos fornecedores para suas plantas industriais ou com a distribuição de mercadorias para pontos de armazenamento ou distribuição. Nesse contexto aparece o problema de transporte com carga fixa (ou custo fixo). O PTCF pode ser formulado da seguinte forma: uma empresa tem m plantas (fabricas) e n depósitos, sendo que, geralmente, o número n de depósitos é muito maior que o número m de fábricas ou plantas, e pretende-se transportar um único produto das plantas para os pontos de armazenamento, devendo-se atender a todas as demandas a partir de quaisquer das plantas industriais. Este problema pode ser formulado de modo balanceado, isto é, assumindo-se que a soma das ofertas disponíveis nas plantas seja igual a soma das quantidades requeridas pelos pontos de armazenamento.

O objetivo do problema consiste em se determinar quais rotas origem-destino serão utilizadas e quais as quantidades deverão ser transportadas em cada rota, ao menor custo possível. sendo que existem custos fixos que dependem somente da implantação da rota e custos variáveis que

dependem da quantidade transportada. Tem-se então o custo total da configuração estabelecida como a soma dos custos fixos (implantação das rotas) com os custos variáveis, assumidos como proporcionais às quantidades transportadas em cada rota. Na prática estes problemas relacionados com a logística geralmente assumem uma forma mais complexa e, nestes casos, o PTCF pode se constituir num subproblema do problema mais complexo.

O PTCF é um problema clássico de pesquisa operacional e para sua resolução, ao longo dos anos, têm sido utilizados muitos algoritmos. Os primeiros algoritmos usados na resolução do PTCF foram algoritmos heurísticos, que encontram soluções de qualidade razoável mas com tempos de processamento muito rápidos além de serem robustos e de fácil implementação. Algoritmos heurísticos para o PTCF e para o Problema de Carga Fixa Geral, apresentando desempenho adequado, podem ser encontrados em [1,2,3,14]. Também têm sido utilizados algoritmos exatos para resolver o PTCF e, nesse contexto, o algoritmo mais usado é o algoritmo de *Branch and Bound* sob diferentes formas. O algoritmo de *Branch and Bound* encontra as soluções ótimas de problemas pequenos e de médio porte entretanto para problemas de grande porte a sua utilização se torna proibitiva devido a problemas com o tempo de processamento e com as memórias para armazenamento da informação [10]. Nos últimos anos, apareceram as metaheurísticas tais como simulated annealing, algoritmos genéticos, busca tabu (*tabu search*), GRASP, etc que vem sendo utilizados para resolver diversos problemas do campo da pesquisa operacional, empregando diferentes formas de implementação e com diferentes níveis de competitividade em relação aos algoritmos heurísticos e exatos já existentes para cada tipo de problema [5,12]. Neste artigo apresentamos um algoritmo de busca tabu (BT) para o problema de transporte com carga fixa que é uma versão modificada do algoritmo proposto em [13]. O algoritmo mostrado apresenta um bom desempenho comparado com outros algoritmos existentes na literatura especializada.

As metaheurísticas apresentam a vantagem de encontrarem soluções ótimas ou quase ótimas mesmo para problemas de grande porte e, geralmente, são relativamente simples tanto na formulação do algoritmo quanto na sua implementação computacional, apresentando entretanto uma desvantagem quanto ao esforço de processamento ainda elevado quando comparado às heurísticas clássicas.

2. - Formulação do Problema

A modelagem matemática para o **PTCF** para m plantas e n centros de consumo assume a seguinte forma:

(PTCF) $\min \quad \sum_{i=1}^{m}\sum_{j=1}^{n} g_{ij}(x_{ij})$ **(1)**

$sa \quad \sum_{i=1}^{m} x_{ij} = d_j$ j=1,2,....,n (demandas) **(1a)**

$\sum_{j=1}^{n} x_{ij} = s_i$ i=1,2,....,m (ofertas) **(1b)**

$x_{ij} \geq 0$ i=1,2,...,m e j=1,2,...,n **(1c)**

sendo

$g_{ij}(x_{ij}) = 0 \quad se \quad x_{ij} = 0$ **(1d)**

ou

$g_{ij}(x_{ij}) = c_{ij}x_{ij} + f_{ij} \quad se \quad x_{ij} > 0$ **(1e)**

Uma propriedade fundamental do PTCF é que a solução ótima desse problema é um ponto extremo do conjunto convexo definido pelas restrições (1 a), (1 b) e (1 c), isto é, é valida a propriedade de problemas de programação linear (PL) de que o ótimo é um ponto extremo de um conjunto convexo poliedral. Entretanto, não é válida a estratégia de solução Simplex de PL porque no PTCF existem muitas soluções ótimas locais (determinadas pela presença dos custos fixos na função objetivo) o que não ocorre em PL em que existe apenas um ótimo local que é também o ótimo global para o problema. Portanto, resolver o PTCF consiste em encontrar um ponto extremo do conjunto convexo antes mencionado que minimize a função objetivo. Em virtude do número de pontos extremos do conjunto convexo ser muito grande, para problemas acima de certo porte, não é possível se utilizar à busca exaustiva.

Os principais algoritmos heurísticos utilizados para se resolver o PTCF utilizam a propriedade fundamental de que o ótimo é um ponto extremo do conjunto convexo definido pelas restrições do problema. Assim, foram usadas propriedades do algoritmo simplex de PL para encontrar um ponto extremo interessante e depois se aplicar uma estratégia de refinamento de busca local para encontrar uma solução de melhor qualidade a partir da solução interessante. Nesse tipo de estratégia se encontram os algoritmos heurísticos apresentados em [1,2,3,12,14]. Obviamente, esses algoritmos param após esgotar a busca local, encontrando um ótimo local. Comentamos brevemente a estratégia fundamental desses algoritmos.

O algoritmo de Balinski [1] transforma os custos fixos e variáveis da função objetivo em um único custo equivalente e resolve o problema resultante como se fosse um problema de PL. Nessa estratégia o custo equivalente é igual ao custo variável mais o custo fixo dividido pelo menor valor entre a oferta da planta i e a demanda do centro de consumo j. Por outro lado, algoritmo de Walker [14] está baseado em uma estratégia de busca através de pontos extremos adjacentes e apresenta uma estratégia dividida em duas fases: (1) fase simplex modificada e (2) fase de melhoramento. A fase 1 encontra um ponto extremo usando o algoritmo simplex modificado. A diferença em relação ao método simplex acontece na escolha das variáveis que devem entrar e sair da base. Esse processo de escolha deve levar em conta os custos fixos das variáveis que participam da troca da base, assim como os problemas de base degenerada em que os custos fixos devem ser adequadamente atualizados. Na fase de melhoramento, o algoritmo de Walker realiza uma busca local em torno do ponto extremo de qualidade encontrada na fase do método simplex modificado. Esse processo de busca local consiste em procurar outros pontos extremos de melhor qualidade na vizinhança do ponto extremo corrente. Walker apresenta três estratégias de busca local que define a profundidade da busca e algumas estratégias específicas. Existem outras propostas heurísticas baseadas em buscas através de pontos extremos como as apresentadas em [2,3,12].

A importância dos algoritmos heurísticos na formulação de algoritmos BT está no fato de que é possível aproveitar as principais características e estratégias desses algoritmos heurísticos para incorporar dentro da estrutura do algoritmo de busca tabu. Assim, por exemplo, pode-se usar as heurísticas de Balinski e/ou Walker para encontrar um conjunto de soluções iniciais de elite para iniciar ou reiniciar o processo de busca do algoritmo de busca tabu. Também, pode-se adotar uma estratégia de busca através de pontos extremos para realizar as transições no algoritmo de busca tabu. A estratégia de busca através de pontos extremos também é usada para caracterizar a vizinhança da solução corrente (que é um ponto extremo) do algoritmo de busca tabu. Assim, podem ser vizinhos da solução corrente, todos ou uma parcela de pontos extremos adjacentes do ponto extremo corrente. O algoritmo de busca tabu apresentado está baseado nas estratégias mencionadas anteriormente.

3. – Fundamentos Básicos de Busca Tabu

A teoria sobre a estratégia de busca tabu (BT) está adequadamente apresentada em [6]. Portanto, apresentamos um resumo das principais características de BT, especialmente aquelas funções que foram usadas no PTCF. BT foi projetada para encontrar configurações ótimas ou quase ótimas de problemas complexos (problemas não convexos, não diferenciáveis, com variáveis mistas,

etc) e com características combinatórias (problemas cujo espaço de soluções cresce de maneira exponencial com o tamanho do problema) como o caso do PTCF. BT realiza um processo de transições através do espaço de soluções do problema a partir de uma solução inicial (topologia ou configuração inicial) de forma similar a uma heurística de busca local. Entretanto, BT tem capacidade de sair de soluções ótimas locais. Assim, dada uma solução, define-se uma vizinhança dessa solução como sendo qualquer solução que pode ser encontrada a partir da solução dada usando um mecanismo de transição. BT é diferente de uma heurística de busca local em dois aspectos fundamentais: (1) a partir da solução atual ou corrente, deve-se passar para a melhor solução vizinha ou a menos pior, o que significa que é permitida uma degradação da função objetivo e, (2) o conjunto das soluções vizinhas da solução corrente não é identificado de forma estática e, portanto, depois de cada transição, deve-se definir uma nova vizinhança que varia dinamicamente em estrutura e tamanho durante todo o processo de otimização. Esta estratégia permite a BT realizar uma busca eficiente e inteligente através do espaço do problema. Para mostrar as estratégias mais importantes de BT, apresentamos as características fundamentais de BT separadas em três partes: (1) algoritmo BT com memória de curto prazo, (2) funções avançadas de TS e (3) estratégia de redução de vizinhos candidatos que devem ser avaliados.

3.1 - BT com Memória de Curto Prazo

O algoritmo BT mais elementar é chamado de algoritmo BT com memória de curto prazo. Neste algoritmo, o processo é iniciado com uma solução inicial e se realiza um número determinado de transições até satisfazer um critério de parada. Em cada passo, deve-se analisar um conjunto de vizinhos (todos os vizinhos ou um conjunto reduzido) e passar para o melhor vizinho, desde que não se encontre proibido, que se transforma na nova solução corrente. O caráter agressivo da estratégia BT de passar sempre para a melhor solução vizinha (ou para a menos pior) permite a BT sair de soluções ótimas locais mas também exige manter uma lista que evite retornar a soluções já visitadas. A lista tabu geralmente armazena atributos das soluções visitadas. Uma lista tabu que armazena atributos proibidos reduz bastante as necessidades de memória mas incorpora um outro problema porque o atributo proibido, que corresponde a uma solução já visitada, é compartilhado por outras soluções da região de busca, algumas das quais podem ser muito atrativas mas que não podem ser visitadas porque tem um atributo proibido.

O problema de armazenamento de atributos proibidos é contornado usando uma função de BT chamada de critério de aspiração. Assim, pode-se eliminar a proibição de uma solução vizinha se a função objetivo dessa solução vizinha satisfaz um critério de aspiração especificado. Por exemplo, se a função objetivo da solução vizinha proibida é a melhor das última k soluções visitadas ou se é melhor de todas as soluções já visitadas (incumbente) então, pode-se eliminar a proibição e passar para essa solução vizinha. Uma estratégia BT que realiza um conjunto de transições usando uma lista tabu de atributos proibidos para evitar visitar soluções já visitadas e usa um critério de aspiração para eliminar a proibição de soluções vizinhas de qualidade é conhecido como algoritmo BT com memória de curto prazo e constitui o algoritmo BT mais elementar Algoritmos BT mais sofisticados devem incorporar estratégias adicionais como são as funções avançadas de BT e, nesse caso, o algoritmo BT com memória de curto prazo opera como uma subrotina dentro da estrutura BT mais sofisticada.

3 2 - Funções Avançadas de Busca Tabu

As funções (operadores) avançadas de Busca Tabu incorpora outras estratégias de busca e de vizinhança para melhorar a qualidade da solução encontrada. As principais são as seguintes: (1) intensificação, (2) diversificação, (3) *path relinking* e (4) soluções de elite.

Intensificação é uma função avançada de BT, ou seja, pode ser implementada em adição com a BT com memória de curto prazo. Em outras palavras, uma vez usado o algoritmo BT básico, em vez de parar o processo, pode-se usar intensificação para encontrar soluções melhores. Existem várias formas de implementar a intensificação. Assim, por exemplo, pode-se realizar a intensificação a partir

de configurações de elite armazenadas durante o processo de BT com memória de curto prazo, permitindo a possibilidade de encontrar soluções de melhor qualidade. Também é possível implementar intensificação modificando, de tempo em tempo, a estrutura de vizinhança e os parâmetros de controle do algoritmo BT com memória de curto prazo, restringindo a busca em torno de uma solução de alta qualidade.

Diversificação também pode ser incorporada no processo de BT. Neste caso, tenta-se levar o processo de busca a regiões distantes das regiões já analisados pelo algoritmo BT básico. Diversificação também pode ser implementada de várias formas e duas delas são as seguintes: (1) usando memória de longo prazo baseado em freqüência, onde os atributos pouco usados são incentivados durante um intervalo do processo, encontrando novas configurações e novas regiões de busca com a presença desses atributos, e (2) usando *path relinking* onde se encontram novas soluções de partida, para reiniciar o processo de busca, trocando parcelas de atributos de soluções de elite.

Path relinking é uma função de BT que é usada para encontrar soluções potencialmente atrativas para reiniciar o processo de busca através de intensificação ou diversificação. Durante o processo BT são armazenadas as melhores soluções, chamadas de soluções de elite. Duas ou mais soluções de elite podem ser usadas para direcionar a busca durante um número determinado de transições ou para gerar uma nova solução. No primeiro caso, a partir de uma solução de elite, pode-se realizar algumas transições e, durante esse processo, deve-se usar apenas os atributos de outras soluções de elite. No segundo caso, gera-se uma nova solução usando os atributos de soluções de elite e este tipo de *path relinking* é semelhante com a recombinação genética usada no algoritmo genético. Assim, esse tipo de *path relinking* é um caso particular de recombinação inteligente.

Existem várias estratégias para a redução de vizinhos candidatos, isto é, para determinar o número máximo de soluções vizinhas que devem ser avaliadas. Neste trabalho, as soluções vizinhas avaliadas são um subconjunto de pontos extremos adjacentes do ponto extremo corrente.

A estratégia BT foi usada para implementar um algoritmo BT para o PTCF. A estratégia implementada é basicamente um módulo de BT básico que interage com processos de busca que usa memória intermediária e memória de longo prazo. O algoritmo apresentado é uma modificação da estratégia apresentada em [13]. A estrutura de vizinhança está determinada pelos vizinhos existentes a partir do ponto extremo que corresponde à solução corrente, isto é, os pontos extremos adjacentes de melhor qualidade ou todos os pontos extremos adjacentes.

4. – Algoritmo de Busca Tabu para o PTCF

A seguir é descrito o procedimento TS para o PTCF, passo a passo, que é uma versão modificada da proposta originalmente apresentada por SUN et al (1998). Denomina-se: z_{min} = melhor valor encontrado para a função objetivo durante o processo de memória atual; k_{min} = número da iteração para a qual foi encontrado z_{min}, desde que o processo atual se iniciou; $x_o = x_{best}$ melhor solução desde o início da busca; $z_o = z_{best}$ = melhor valor encontrado para a função objetivo, desde que se iniciou a busca.

Inicialização

Passo 0

A partir da solução básica factível inicial encontrada usando um algoritmo heurístico (Balinski, Cooper, etc), computa-se o valor de Δ_{ij} para cada $(i,j) \notin \tau$ e introduz-se o arco (i,j) na árvore geradora se a variação líquida da função objetivo for $\Delta_{ij} < 0$. Se $\Delta_{ij} \geq 0$ para todo $(i,j) \notin \tau$ tem-se a **primeira solução ótima local**. Para essa solução ótima local com valor da função objetivo igual a z, compute o parâmetro α.

Faça o contador de iteração k = 0;

Faça $l = 0$ e $l_1 = 0$.

Para todo i e j

faça $h_{ij} = 0$, $z_{ij} = z$ e $t_{ij} = -\infty$.

Fim Para.

Faça i =0

Procedimento memória de curto prazo

Passo 1: Faça i = i + 1. Se i > m então faça i =1. Fim Se. Compute Δ_{iq} para todo q=1 até n e (i,q) $\notin \tau$

Passo 2: Selecione (i,j) como um arco entrando tal que: $\Delta_{ij} = \min\{\Delta_{iq} \mid (i,q) \notin \tau, 1 \le q \le n\}$, com um arco saindo (i_s,j_s). Se $\Delta_{ij} = \infty$, vá para o **Passo 1.**

Passo 3: Se $(k - t_{ij}) \le \xi_{in}$ ou $(k-t_{rs}) \le \xi_{out}$ vá para o **Passo 4.** Caso contrário vá para o **Passo 5**. Fim Se

Passo 4: Se $z + \Delta_{ij} > z_{ij}$ ou $z + \Delta_{ij} > z_{rs}$ faça $\Delta_{ij} = \infty$ e vá para o **Passo 2.** Caso contrário vá para o **Passo 5.** Fim Se.

Passo 5: Atualize a árvore geradora. Faça $h_{rs} = h_{rs} + (k - t_{rs})$; $t_{ij} = k$; $t_{rs} = k$; $z_{ij} = z$; $z_{rs} = z$. Fim Faça

Faça k = k +1 e $z = z + \Delta_{ij}$. Fim Faça.

Passo 6: Se $z < z_{min}$ faça $z_{min} = z$ e $k_{min} = k$. Se $z < z_o$ faça $z_o = z$ e salve $\mathbf{x_o}$ (solução corrente). Fim Se Fim Se.

Passo 7: Se $(k - k_{min}) < \beta N$ vá para o passo 1 (continuar processo de memória de curto prazo). Fim Se.

Se $\beta N \le (k - k_{min}) < (\beta + \gamma)N$ vá para o **Passo 8 (iniciar processo 1 de memória intermediária).** Caso contrário vá para o **Passo 12 (iniciar processo 2 de memória intermediária).** Fim Se

Processo 1 de memória intermediária

Passo 8: Faça i = i + 1. Se i > m faça i = 1. Fim Se.

Para todo q=1 até n e com (i,q) $\notin \tau$ compute $\Delta'_{iq} = \Delta_{iq} - \alpha \frac{h_{iq}}{k}$. Fim Para

Passo 9: Selecione (i,j) como um arco entrando tal que $\Delta'_{ij} = \min\{\Delta'_{iq} \mid (i,q) \notin \tau, 1 \le q \le n\}$, com um arco saindo (i_s,j_s). Se $\Delta'_{ij} = \infty$, vá para o **Passo 8.** Fim Se.

Passo 10: Se $(k - t_{ij}) \le \xi_{in}$ ou $(k-t_{rs}) \le \xi_{out}$ vá para o **Passo 11.** Caso contrário vá para o **Passo 5**. Fim Se.

Passo 11: Se $z + \Delta_{ij} > z_{ij}$ ou $z + \Delta_{ij} > z_{rs}$. faça $\Delta'_{ij} = \infty$ e vá para o **Passo 9.** Caso contrário vá para o **Passo 5.** Fim Se.

Processo 2 de memória intermediária

Passo 12: Faça $(i_o,j_o) = (i,n)$

Passo 13: Faça i = i + 1. Se i > m faça i =1. Fim Se. Fim Faça

Passo 14: Para j = 1 até n e (i,j) $\notin \tau$ repita Passos 14 (a) até 14 (d): (a) Compute Δ_{ij}; (b) Se $\Delta_{ij} \ge 0$ e Se (i,j) = (i_o,j_o) vá para o **Passo 16**; (c) se $\Delta_{ij} < 0$ faça $(i_o,j_o) = (i,j)$ e faça (i,j) entrar para a árvore geradora com arco saindo (i_s,j_s). Fim Se; (d) atualize a árvore geradora: Faça $h_{rs} = h_{rs} + (k - t_{rs})$; $t_{ij} = k$; $t_{rs} = k$; $z_{ij} = z$; $z_{rs} = z$ Fim Faça. Faça k = k + 1. Faça $z = z + \Delta_{ij}$

Passo 15: Vá para o **Passo 13.**

Conclusão dos processos de memória intermediária e regra de parada

Passo 16: Se $z < z_o$ faça $z_o = z$ e salve $\mathbf{x_o}$ (solução corrente). Se $l \geq L$ então **STOP**. Fim Se. Faça $i_o = i$. Se $l_1 = 0$ faça $l_1 = 1$ e vá para o **Passo 17**. Caso Contrário faça $l_1 = 0$ e vá para o **Passo 23.** Fim Se.

Processo 1 de memória de longo prazo

Passo 17: Faça i = i + 1. Se i > m faça i =1. Fim Se. Compute Δ''_{iq} para todo q=1 até n e $(i,q) \notin \tau$

Passo 18: Selecione (i_e,j_e) como um arco entrante tal que: $\Delta''_{ij} = \min\{\Delta''_{iq} \mid (i,q) \notin \tau, 1 \leq q \leq n\}$, com um arco saindo (i_s,j_s). Se $\Delta''_{ij} = \infty$, vá para o **Passo 17.**

Passo 19: Se $(k - t_{ij}) \leq \xi_{in}$ ou $(k-t_{rs}) \leq \xi_{out}$ vá para o **Passo 20.** Caso contrário vá para o **Passo 21**. Fim Se.

Passo 20: Se $z + \Delta_{ij} > z_{ij}$ ou $z + \Delta_{ij} > z_{rs}$ faça $\Delta''_{ij} = \infty$ e vá para o **Passo 18.** Caso contrário vá para o **Passo 21.** Fim Se.

Passo 21: Atualize a árvore geradora.. Faça $h_{rs} = h_{rs} + (k - t_{rs})$; $t_{ij} = k$; $t_{rs} = k$; $z_{ij} = z$; $z_{rs} = z$. Fim Faça. Faça $k = k + 1$ e $z = z + \Delta_{ij}$. Fim Faça.

Passo 22: Se $i = i_o$ faça $z_{min} = z$ e $k_{min} = k$. Reinicialize os valores de ξ_{in} e ξ_{out} e vá para o **Passo 1** Caso Contrário vá para o **Passo 17**. Fim Se.

Processo 2 de memória de longo prazo

Passo 23: Faça i = i + 1. Se i > m faça i =1. Fim Se. Selecione (i_e,j_e) como um arco entrando tal que $t_{ij} = \min\{t_{iq} \mid (i,q) \notin \tau, 1 \leq q \leq n\}$, com um arco saindo (i_s,j_s). Compute Δ_{ij}

Passo 24: Atualize a árvore geradora. Faça $h_{rs} = h_{rs} + (k - t_{rs})$; $t_{ij} = k$; $t_{rs} = k$; $z_{ij} = z$; $z_{rs} = z$. Fim Faça. Faça $k = k + 1$ e $z = z + \Delta_{ij}$. Fim Faça

Passo 25: Se $i = i_o$ faça $l = l + 1$; $z_{min} = z$; $k_{min} = k$;. Reinicialize os valores de ξ_{in} e ξ_{out} e vá para o **Passo 1.** Caso Contrário vá para o **Passo 23.**

FIM

Símbolos utilizados:

α - coeficiente para penalizar ou incentivar configurações a entrar na base, no processo memória intermediária 1 e memória de longo prazo 1, no BT para o PTCF.

β - coeficiente para determinação do número de iterações nos processos memória de curto prazo no BT para o PTCF.

βN - número de iterações no processo memória de curto prazo e para início do processo memória intermediária 1 no BT para o PTCF.

$(\beta+\gamma)N$ – número limite de iterações no processo memória intermediária 1 e para início do processo memória intermediária 2 no TS para o PTCF.

γ - coeficiente para determinação do número limite de iterações no processo memória intermediária 1 e início do processo memória intermediária 2 no BT para o PTCF.

Δ - variação líquida da função objetivo, correspondente a entrada de (i_e,j_e) na base e a saída de (i_s,j_s) da base, no BT para o PTCF.

Δ' - valor alternativo à variação líquida da função objetivo, correspondente a entrada de (i_e,j_e) na base e a saída de (i_s,j_s) da base, no processo memória intermediária 1 no BT para o PTCF.

Δ'' - valor alternativo à variação líquida da função objetivo, correspondente a entrada de (i_e,j_e) na base e a saída de (i_s,j_s) da base, no processo memória de longo prazo 1 no BT para o PTCF.

τ - árvore geradora mínima ou base para o PTCF.

ξ_{in} – parâmetro correspondente ao tamanho da lista tabu para elementos a entrar na base no BT para o PTCF.

ξ_{out} – parâmetro correspondente ao tamanho da lista tabu para elementos a sair da base do BT para o PTCF.

c_{ij} – custo unitário de transporte da fábrica i para o consumidor j.

d_j – demanda do consumidor j.

f_{ij} – custo fixo ou carga fixa de abertura da rota (i,j), da fábrica i para o consumidor j.

$g(x_{ij})$ - função custo de transporte na rota (i,j) no PTCF, quando se considera o custo fixo mais o custo variável.

h - memória de histórico da busca no BT geral ou no BT para o PTCF.

k – iteração corrente no BT para o PTCF.

k_{min} – iteração para a qual se obteve o valor z_{min}, no BT para o PTCF.

l - número corrente de chamadas ao processo memória de longo prazo 1 ou 2.

l_1 - ponteiro para alternância entre estratégias memória de longo prazo 1 ou 2.

L - número máxima de chamadas ao processo memória de longo prazo 1 ou 2.

m – número de fábricas ou número de linhas das matrizes de custos unitários e custos fixos.

N - número de variáveis de decisão, igual a mxn, no PTCF.

n – número de consumidores ou número de colunas das matrizes de custos unitários e custos fixos.

nb – dimensão da base ou árvore geradora mínima, igual a m + n - 1, para o PTguat uni7 5 h[(h)2á1ó2.6(1)c1óri, no BT(prazo 1 ou

k

baseado em SUN et al. (1998), para a melhoria das soluções obtidas pelo primeiro e comparando-se os resultados obtidos, em termos percentuais, com o ótimo global fornecido. Os resultados obtidos são mostrados nas tabelas 1, 2 e 3.

Tabela 1: Problemas Pequenos/Fáceis

Resultados obtidos pelo algoritmo BT										
para os problemas fáceis - exemplos dos livros de Gottlieb										
dados originais		p/balinski+regra noroeste+pl			p/balinski+regra custo min+pl			algoritmo BT		
problema	res.livros/go	ótimo local	número de	% rel.ot.glob	ótimo local	número de	% rel.ot.glob	ótimo local	número de	% rel.ot.glob
FCTP	ótimo global	noro+PL	iterações	noro/ot.glob	cmin+PL	iterações	cmin/ot.glob	TS	iterações	TS/ot.glob
bk4x3	350,00	360,00	3	102,9%	360,00	3	102,9%	350,00	12	100,0%
gr4x6	202,35	234,95	4	116,1%	234,95	4	116,1%	202,35	9	100,0%
puc4x6	112,00	115,00	5	102,7%	112,00	0	100,0%	112,00	4	100,0%
murty5x7	710,00	826,00	7	116,3%	802,00	3	113,0%	710,00	59	100,0%
bal8x12	471,55	504,55	10	107,0%	504,55	11	107,0%	471,55	204	100,0%
kb2x3	65,00	65,00	1	100,0%	65,00	1	100,0%	65,00	7	100,0%
médias	318,48	350,92	5,0	107,5%	346,42	3,7	106,5%	318,48	49,2	100,0%

Resultados comparativos BT e Balinski – fáceis - bk4x3: problema 4x3 de Balinski e Krarup ap. Gottlieb (2001); gr4x6: problema 4x6 de Gray ap. Gottlieb (2001); puc4x6: problema 4x6 de Puccini (1985); murty5x7: problema 5x7 de Murty (1968); bal8x12: problema 8x12 de Balinski ap. Gottlieb (2001); e kb2x3: problema 2x3 de Kuhn e Baumol ap. Salkin (1989).

Tabela 2: Problemas Médios

Resultados obtidos pelo algoritmo BT										
para os problemas de médio porte gerados randomicamente por Gottlieb.										
dados originais		p/balinski+regra noroeste+pl			p/balinski+regra custo mín+pl			algoritmo BT		
problema	res.Gottlieb	ótimo local	n.iter.	% rel.ót.glob.	ótimo local	n.iter.	% rel.ot.glob.	ótimo BT	n.iter.	% rel.ót.glob.
FCTP	ótimo global	noro+PL		noro/ót.glob.	cmin+PL		cmin/ót.glob	alfa=.05/beta=.07		
ran10x10a	1499,0	1616,0	22	107,8%	1736,0	15	115,8%	1499,0	254	100,0%
ran10x10b	3073,0	3337,0	19	108,6%	3337,0	15	108,6%	3092,0	143	100,6%
ran10x10c	13007,0	16636,0	18	127,9%	16636,0	10	127,9%	13115,0	311	100,8%
ran10x12	2714,0	3294,0	19	121,4%	3290,0	9	121,2%	2881,0	409	106,2%
ran10x26	4270,0	4782,0	47	112,0%	4782,0	25	112,0%	4445,0	1009	104,1%
ran12x12	2291,0	2668,0	20	116,5%	2668,0	9	116,5%	2291,0	571	100,0%
ran12x21	3664,0	4158,0	46	113,5%	4098,0	21	111,8%	3726,0	1297	101,7%
ran13x13	3252,0	3598,0	27	110,6%	3486,0	15	107,2%	3284,0	959	101,0%
ran14x18	3712,0	4282,0	45	115,4%	4347,0	19	117,1%	3824,0	1063	103,0%
ran16x16	3823,0	4393,0	30	114,9%	4456,0	15	116,6%	3903,0	2988	102,1%
ran17x17	1373,0	1454,0	48	105,9%	1514,0	26	110,3%	1373,0	2004	100,0%
ran4x64	9711,0	10012,0	59	103,1%	10007,0	41	103,0%	9737,0	83	100,3%
ran6x43	6330,0	6691,0	44	105,7%	6645,0	36	105,0%	6381,0	180	100,8%
ran8x32	5247,0	5837,0	51	111,2%	5837,0	30	111,2%	5402,0	46	103,0%
médias	4569,0	5197,0	35	112,5%	5202,8	20	113,2%	4639,5	808	101,7%

Resultados comparativos BT e Balinski – médios - ran10x10a a ran8x32: problemas fctp gerados randomicamente com dimensões 10x10 a 8x32 de Gottlieb (2001).

Tabela 3: Problemas Grandes

Resultados obtidos pelo algoritmo BT													
para os problemas de grande porte (50x100) gerados randomicamente por Sun													
dados originais													
problema	res.Sun	ótimo local	n.iter.	% rel.ót.glob.	ótimo local	n.iter.	% rel.ot.glob.	ótimo BT	n.iter.	% rel.ót.glob.	ótimo BT	n.iter.	% rel.ót.glob.
FCTP	ótimo global	noro+PL		noro/ót.glob.	cmin+PL		cmin/ót.glob.	alfa=.05/beta=.15			alfa=.05/beta=.10		
n3700	1254130,0	1342264,0	418	107,0%	1345704,0	209	107,3%	1267873,0	6297	101,1%	1268113,0	2969	101,1%
n3701	1244592,0	1309690,0	403	105,2%	1312769,0	212	105,5%	1254920,0	4090	100,8%	1255938,0	3592	100,9%
n3702	1227393,0	1326705,0	389	108,1%	1319203,0	196	107,5%	1264496,0	3674	103,0%	1251438,0	3245	102,0%
n3703	1216236,0	1316163,0	383	108,2%	1315924,0	208	108,2%	1244701,0	3808	102,3%	1242460,0	3362	102,2%
n3704	1244366,0	1326602,0	421	106,6%	1327781,0	202	106,7%	1274900,0	3998	102,5%	1274669,0	3310	102,4%
n3705	1229634,0	1342738,0	396	109,2%	1342561,0	219	109,2%	1269059,0	3835	103,2%	1262234,0	3975	102,7%
n3706	1241381,0	1322093,0	358	106,5%	1323789,0	204	106,6%	1269987,0	3763	102,3%	1263750,0	3198	101,8%
n3707	1203955,0	1295125,0	398	107,6%	1302363,0	225	108,2%	1235274,0	3961	102,6%	1237320,0	3279	102,8%
n3708	1224734,0	1329354,0	406	108,5%	1329519,0	205	108,6%	1263576,0	3892	103,2%	1270001,0	6316	103,7%
n3709	1216562,0	1324453,0	392	108,9%	1325502,0	192	109,0%	1247976,0	3709	102,6%	1248356,0	3474	102,6%
n370a	1272014,0	1353827,0	377	106,4%	1353578,0	227	106,4%	1279901,0	4513	100,6%	1278813,0	6104	100,5%
n370b	1243314,0	1368880,0	380	110,1%	1370409,0	204	110,2%	1279472,0	3913	102,9%	1277482,0	5433	102,7%
n370c	1228867,0	1343329,0	391	109,3%	1346115,0	197	109,5%	1268640,0	4363	103,2%	1259892,0	4866	102,5%
n370d	1228857,0	1343329,0	391	109,3%	1346115,0	197	109,5%	1268640,0	4363	103,2%	1259892,0	4866	102,5%
n370e	1220967,0	1317883,0	373	107,9%	1319251,0	191	108,0%	1250571,0	4344	102,4%	1248565,0	5412	102,3%
médias	1233133,5	1330829,0	392	107,9%	1332038,9	206	108,0%	1262665,7	4168	102,4%	1259928,2	4227	102,2%

Resultados comparativos BT e Balinski – grandes - ran10x10a a ran8x32: problemas fctp gerados randomicamente com dimensões 10x10 a 8x32 de Gottlieb (2001).

Os problemas pequenos e fáceis foram obtidos da literatura consultada e de GOTTLIEB (2001) sendo os demais diretamente dele. Os problemas médios e grandes (50x100), gerados aleatoriamente são totalmente conectados e densos em termos de custos fixos. A quantidade fornecida/demandada para os problemas 50x100 é de 50.000 As soluções ótimas globais para os problemas médios e grandes também se encontram disponíveis em GOTTLIEB (2001), sendo que os problemas 50x100 foram os mesmos utilizados por SUN et al. (1998).

O procedimento **BT** obteve o ótimo global de modo rápido, com um pequeno número de iterações, para os problemas pequenos e fáceis. Ficou bastante próximo do ótimo global para os problemas médios e ligeiramente acima do ótimo global no caso dos problemas grandes. No experimento limitou-se o número de chamadas às memórias de longo prazo 1 e 2 a apenas 2 de cada, através da fixação do parâmetro L=4.

De modo a obter um compromisso entre a qualidade das soluções e o tempo de resposta, procurou-se ajustar empiricamente, de modo adequado, os valores dos parâmetros β e γ. Estes parâmetros definem o número de iterações que o algoritmo TS vai realizar dentro de cada processo sem que haja variação no valor de zmin. Os parâmetros β e γ, para os problemas pequenos, foram fixados em 0,30 e 0,70, respectivamente, enquanto que para os problemas médios, foram adotados os valores 0,05 e 0,70. Para os problemas grandes (50x100) foram utilizados os pares (0,05 e 0,10) e (0,05 e 0,15), para β e γ respectivamente, correspondendo os limites de 250 iterações (alterações da base), sem que se houvesse melhoria no valor zmin, para a busca local (memória de curto prazo). Têm-se as faixas 250-750 e 250-1000, respectivamente, como número de iterações dentro do processo de intensificação (memória intermediária 1) e, acima dos 750 e 1000, respectivamente, para a chamada da intensificação efetuada pela memória intermediária 2, que por sua vez aciona alternadamente os processos de diversificação chamados de memória de longo prazo 1 e 2.

Atribuíram-se valores para os parâmetros da lista tabu dentro das faixas: $7 \leq \xi_{in} \leq 12$ e $5 \leq \xi_{out} \leq 7$, para os problemas grandes, enquanto que para os problemas pequenos e médios atribuíram-se valores dentro das faixas $7 \leq \xi_{in} \leq 12$ e $2 \leq \xi_{out} \leq 4$

Para os problemas pequenos e simples o BT obteve o ótimo global (100,0%) em todos os casos; para os problemas médios as soluções obtidas pelo BT ficaram na faixa de 100,0% a 106,2% e com média de 101,7% em relação aos ótimos globais e, para os problemas 50x100, os ótimos locais ficaram na faixa de 100,9% a 103,7% e com média de 102,2%, em relação aos ótimos globais, podendo se considerar estes resultados como bastante satisfatórios.

6. – Conclusões

Os problemas de transporte com carga fixa são problemas difíceis de se resolver principalmente quando se têm portes médios ou grandes sendo que os algoritmos para eles desenvolvidos são geralmente muito pouco eficientes e dependentes do número de variáveis inteiras envolvidas e da estrutura especial do problema que deve ser explorada.

A aplicação de algoritmos heurísticos como o BT é uma abordagem que se torna atrativa, principalmente quando envolvem um grande número de variáveis, em virtude de o BT trabalhar de modo rápido e gerar boas soluções. Estas características são bastante interessantes, principalmente em se tratando de problemas de planejamento, quando os dados envolvidos são apenas hipóteses, com alguma margem de incerteza, a serem comprovados posteriormente durante e após a implantação do projeto, quando deverão ocorrer pequenos ajustes.

Pelos resultados obtidos constata-se que o BT funciona bem na maioria dos problemas de transporte com carga fixa testados embora seja altamente dependente dos parâmetros de controle que devem ser fixados após muitas tentativas, ou seja, de modo empírico, utilizando-se problemas com ótimos conhecidos e com dimensões equivalentes as dos problemas que se deseja resolver.

Constata-se que a Busca Tabu, com a implementação efetuada, consegue trabalhar adequadamente os conceitos de intensificação da busca e de diversificação da busca, conseguindo explorar adequadamente o espaço de soluções e encaminhar a busca para regiões fora dos ótimos locais que são obtidos ao longo dos processos, obtendo desse modo novos ótimos locais, porém altamente dependente dos parâmetros de controle fixados, exigindo muitas tentativas para se poder fixar os parâmetros de forma adequada.

As funções introduzidas para se incentivar certas configurações a entrar na base no processo de intensificação e para penalizar certas configurações em detrimento de outras soluções no processo diversificação, utilizando as informações armazenadas de modo implícito de ocorrências em termos de fatos recentes e de freqüência com que certas configurações entram, saem ou permanecem na base funcionou adequadamente.

Pode-se concluir ainda, que na implementação efetuada, o BT se apresenta, a exemplo das técnicas que utilizam o PPL como base, como um algoritmo determinístico, ou seja, para determinados parâmetros de controle, chega-se às mesmas soluções, caminhando pelos mesmos ótimos locais, embora pudesse se dar um certo caráter aleatório ao BT pela implementação de alterações dos parâmetros de busca ao longo do processo, dentro de uma certa faixa de valores pré-determinados.

Bibliografia

[1] BALINSKI, M.L. *Fixed cost transportation problem*. Naval Research Logistics Quarterly 8, p.41-54, 1961.

[2] COOPER, L., DREBES, C. *Aproximate Solution Method for the Fixed Charge Problem*, Naval Research Logistics Quarterly, Vol. 14, pp 101-113, 1967.

[3] COOPER, L. *The Fixed Charge Problem: A New Heuristic Method,* Computer and Mathematics with Applications, 1(1), Pergamon Press,l. pp 89-95, 1975.

[4] BAZARAA, M.S., JARVIS, J.J., SHERALI, H.D. *Linear Programming and Network Flows.* 2th.ed. New York: John Wiley, 1990, p.419-527.

[5] HIRSCH, W.M., DANTZIG, G. *The fixed charge problem.* Naval Research Logistics Quartely 15, p.413-424, 1968.

[6] GLOVER, F., LAGUNA M. *Tabu Search*, Kluwer, 1997.

[7] GOTTLIEB, J, PAULMANN, L. *Genetic Algorithms for the Fixed Charge Transportation Problem.* Proceeding of the 5th IEEE International Conference on Evolutionary Computation, Anchorage, Alaska, p.330-335, 1998. Disponível em: http://www.in.tu-clausthal.de/~gottlieb/papers/icec98.abstract.html

[8] GOTTLIEB, J. *FCTP instances.* Disponível em: http://www.in.tu-clausthal.de/~gottlieb/benchmark/fctp/

[9] HILLIER, F.S., LIEBERMAN, G.J. *Introduction to Operations Research*. 6th ed. New York: McGraw-Hill, 1995, p.303-352 e p.494-557.

[10] PALEKAR, U.S., KARWAN, M.H., ZIONTS, S. *A Branch-and-Bound Method for the Fixed Charge Transportation Problem.* Management Sciences. 36(9), p.1092-1105, 1990.

[11] SALKIN, H.M.F. *Integer Programming*. Wesley, 1975, p.482-504.

[12] STEINBERG, D.I. *The Fixed Charge Problem*. Nav. Res. Log. Quart. 17, p.217-236, 1970.

[13] SUN, M., ARONSON, J.E., MCKEOWN, P.G., DRINKA, D. *A tabu search heuristic procedure for the fixed charge transportation problem*. European Journal of Operational Research 106, p.441-456, 1998.

[14] WALKER, W.E. *A heuristic adjacent extreme point algorithm for the fixed charge problem*. Management Sciences 22 (3), p.587-596, 1976.