UMA APLICAÇÃO DA METAHEURÍSTICA ITERATED LOCAL SEARCH À SOLUÇÃO DO PROBLEMA DE FLUXO MULTIPRODUTO INTEIRO

Fábio Pires Mourão (CEFET-MG)
fabiomourao@terra.com.br
Sérgio Ricardo de Souza (CEFET-MG)
sergio@dppg.cefetmg.br
Carlos Alexandre Silva (CEFET-MG)
calex@calex.mat.br
Marcone Jamilson Freitas Souza (UFOP)
marcone.freitas@oi.ufop.br

Este trabalho busca apresentar a aplicação da metaheurística Iterated Local Search (ILS) à solução do Problema de Fluxo Multiproduto Inteiro. Este problema pertence à classe de problemas NP-difíceis, justificando-se, assim, o uso de técnicaas heurísticas para sua solução. O objetivo aqui é determinar o fluxo dos produtos pelos arcos da rede ao menor custo possível, respeitando-se as restrições de conservação de fluxo e de capacidade. O método de busca local utilizado para a geração de solução inicial para o ILS é o Método de Descida Randômica, que apresenta um baixo custo computacional na implementação realizada. Para a validação do desenvolvimento apresentado, são utilizadas as instâncias geradas pelo pacote GenMCF, apropriada para esta classe de problemas. Os resultado mostram que o método proposto é eficiente, obtendo resultados não alcançados através de métodos exatos.

Palavras-chaves: Problema de Fluxo Multiproduto Inteiro, Iterated Local Search, Metaheurística.

1. Introdução

O objetivo deste trabalho é analisar o comportamento de uma metaheurística aplicada ao Problema de Fluxo Multiproduto Inteiro (PFMI). O problema em tela pertence à classe dos problemas NP-difíceis, possuindo grande aplicação de cunho econômico, como nas áreas de telecomunicação e sistemas de transporte. Dentre essas aplicações podem ser citadas:

- Roteamento de tráfego na internet (Buriol (2003));
- Roteamento de mensagens em redes de comunicação (Hu (1969));
- Seqüenciamento de operações em refinarias de petróleo (Milidiu (2001));
- Seqüenciamento de carga (Shan (1985)).

Neste trabalho, é estudada a aplicação da metaheurística *Iterated Local Search* (ILS) ao PFMI, associada ao método da Descida Randômica como método para a geração da solução inicial utilizada pelo ILS.

Problemas de fluxo multiproduto geralmente apresentam um grande número de variáveis. Diante desse fato, a utilização de métodos exatos pode se tornar inviável, motivando a utilização de heurísticas e metaheurísticas na resolução do problema, que podem garantir bons resultados.

Os primeiros relatos na literatura deste tipo de problema são do início da década de 60, em particular com os trabalhos de Hu (1963) e Fulkerson (1962). O problema é modelado por uma rede identificada por um grafo, onde os produtos trafegam pelos arcos capacitados da rede a um determinado custo, o qual pode depender somente do arco ou também estar associado aos produtos, em função dos arcos pelos quais trafegam, i.e., diferentes produtos podem trafegar em um mesmo arco a custos distintos. Os nós representam pontos de oferta e demanda dos produtos. Neste trabalho, para cada produto existe um par de nós origem – destino especificado. O objetivo é, então, o de determinar o fluxo destes produtos na rede ao menor custo possível, de forma a atender os seguintes conjuntos de restrições:

- restrições de conservação de fluxo;
- restrições de capacidade; e
- restrições de integralidade.

As restrições de conservação de fluxo têm a função de gerenciar o fluxo dos produtos na rede. As restrições de capacidade limitam a quantidade de produtos que passam pelo arco, fazendo não trafegar por ele uma quantidade maior que a suportada. A de integralidade garante que as variáveis envolvidas sejam inteiras e não negativas.

Problemas de fluxo multiproduto (PFM) surgem quando vários produtos compartilham os arcos em uma rede e competem pela capacidade dos arcos. Problemas de fluxo multiproduto inteiro (PFMI) surgem quando o fluxo de um determinado produto, identificado pelo par origem-destino, deve utilizar somente um único caminho, sendo indivisíveis as unidades dos produtos. De acordo com Alvelos (2005), *"cada unidade de cada comodidade é indivisível, o fluxo de uma comodidade pode ser dividido por diferentes caminhos, mas o fluxo em cada um deles tem que ser inteiro"*.

Neste trabalho é proposta uma resolução de PFMI através da Metaheurística *Iterated*

Local Search. O método de busca local utilizado no ILS foi o Método da Descida Randômica, por apresentar um tempo computacional menor quando comparado ao método clássico da descida, já que este verifica todo o espaço de busca. Além disso, como trata-se de um problema de otimização combinatória, testar todo o espaço de busca é um processo inviável.

2. Revisão Bibliográfica

Os primeiro trabalhos relacionados a PFM datam do início da década de 60, com contribuições iniciais de Hu (1963) e Fulkerson (1962). Existem trabalhos mais recentes, sendo que nesta seção alguns destes trabalhos são discutidos.

Em Ahuja (1993), são utilizados os métodos simplex e técnicas de particionamento primal para resolver o PFM. Goffin (1996) apresenta um algoritmo que aborda técnicas de decomposição usando relaxação lagrangeana. São aplicados os métodos de plano de corte e centro analítico para resolver o problema de maximização de uma função dual não-diferencial. Goldberg (1998) utiliza um algoritmo heurístico para solucionar o problema de fluxo multiproduto; quando comparado a métodos exatos, em particular usando o CPLEX 6.5, o algoritmo proposto perde em eficiência.

Park (2002) utiliza a técnica de geração de colunas para resolver duas classes de problemas de fluxo multiproduto inteiro. O objetivo do primeiro problema é selecionar um subconjunto de produtos para serem roteados e associar, a cada um deles, uma única rota ligando o nó de origem ao nó destino, tendo, como finalidade, maximizar o benefício da distribuição. Para avaliar o benefício, é utilizado um valor constante para cada produto em cada rota. O segundo problema consiste na seleção de todos os produtos necessários, sendo o objetivo o de determinar o custo mínimo.

Em Alvelos (2005), a abordagem é por meio de geração de colunas para programação linear e inteira. Os testes computacionais são feitos em instâncias geradas pelo Generator Multicommodity Flow (GenMCF), desenvolvido por Alvelos (2005).

Bocanegra (2005) propõem um algoritmo para resolver o problema não linear de fluxo multiproduto, utilizando planos de corte e centros analíticos. O problema é relaxado utilizando a função lagrangeana parcial, construída a partir de hiperplanos suporte. Foi também utilizado o método de pontos interiores. Segundo Bocanegra (2005), o algoritmo proposto se destaca por gerar uma seqüência monotônica estritamente crescente de cotas para alcançar a solução, e conseqüentemente, segue trajetórias centrais associadas ao máximo da função modelo.

Wille (2005) utilizam as metaheurísticas Busca Tabu e AG para gerar soluções para um probema multiproduto topológico de redes IP. O objetivo é minimizar o custo da distribuição de pacotes pela rede, atendendo a um conjunto de restrições, entre elas a qualidade de serviços. Os resultados computacionais obtidos mostram uma melhor eficiência da abordagem feita pelo AG, gerando boas soluções para redes de tamanho médio, em comparação com o método Busca Tabu.

3. Modelagem Matemática

Nesta seção é apresentada uma modelagem do PFMI, considerando uma formulação baseada em arcos, tendo cada produto uma associação com um par de nós origem – destino. Logo após a modelagem genérica, é discutido um exemplo de um PFMI, a fim de esclarecer melhor a modelagem proposta.

Seja $G = (N,A)$ um grafo orientado representando uma rede com p produtos, a arcos e n nós, sendo as cardinalidades dos conjuntos N e A iguais a n e a, respectivamente. Dessa forma temos que n e a são definidos como:

$\#N = n;$

$\#A = a.$

A formulação matemática do problema é posta da seguinte maneira:

$$\begin{aligned} \min \quad & z = cx && (1)\\ suj.\ a \quad & Nx^i = b^i, \ \ i = 1,...,p && (2)\\ & I.x \le u && (3)\\ & x \in \mathrm{Z}_+ && (4) \end{aligned}$$

sendo que (1) representa a função a ser minimizada; (2) representa o conjunto de restrições de conservação de fluxo; (3) representa o conjunto de restrições de capacidade; (4) representa a restrição de integralidade, para x matriz solução e cada posição da mesma representa o fluxo do produto em um determinado arco. Cada uma dessas variáveis deve ser inteira e não negativa.

Segue que:

- $N \in Z^{nxa}$ é a matriz de incidência nó-arco;
- $x^i \in Z^a$ é o vetor de fluxo do produto i;
- $b^i \in Z^n$ é o vetor oferta/demanda para o produto i;
- $I \in Z^{axpa}$ é um vetor composto de matrizes identidades $(a\ x\ a)$;
- $x \in Z^{ap}$ é o vetor composto pelos vetores de fluxo de cada produto;
- $u \in Z^a$ é o vetor capacidade dos arcos.

Vejamos um exemplo de um PFMI, com uma rede pequena e bem simples de ser resolvido:

Seja $G = (N,A)$ o grafo que identifica a rede representada pela Figura 1:

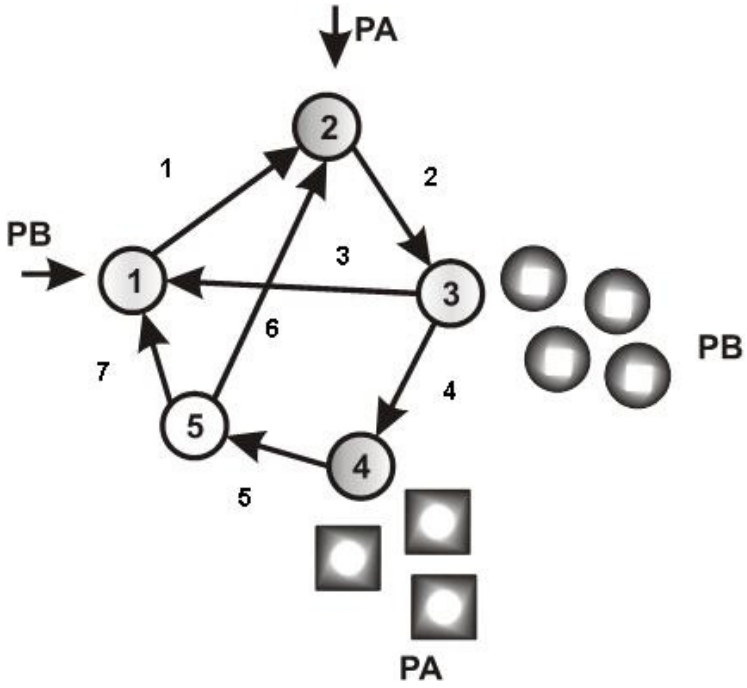

Figura 1 – Exemplo de um PFMI

No exemplo apresentado na Figura 1, o produto A possui como origem o nó 4 e como destino o nó 2. Já o produto B tem como origem o nó 3 e como destino o nó 1, i.e., como mencionado anteriormente, para cada produto existe um par origem-destino, ou seja, um nó de oferta e um de demanda para o produto. Seguem as matrizes e vetores descritos na formulação genérica do PFMI:

$$N = \begin{pmatrix} 1 & 0 & -1 & 0 & 0 & 0 & -1 \\ -1 & 1 & 0 & 0 & 0 & -1 & 0 \\ 0 & -1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 1 & 1 \end{pmatrix}$$

$$b = \begin{pmatrix} 0 & -4 \\ -3 & 0 \\ 0 & 4 \\ 3 & 0 \\ 0 & 0 \end{pmatrix}$$

$$c = \begin{pmatrix} 2 & 3 & 4 & 1 & 1 & 4 & 1 \\ 2 & 3 & 4 & 1 & 1 & 4 & 1 \end{pmatrix}$$

$$u = \begin{pmatrix} 5 & 5 & 5 & 5 & 5 & 5 & 5 \end{pmatrix}$$

São válidas algumas considerações acerca do exemplo apresentado na Figura 1:

N é uma matriz de dimensão *n x a*, sendo que:

$N_{i,j} = 1$, se o arco j possui como origem o nó i;

$N_{i,j} = -1$, se o arco j possui como destino o arco i;

$N_{i,j} = 0$, se o nó i não é caracterizado como origem ou destino associado ao arco j.

b é uma matriz de dimensão $n \times p$, onde os elementos negativos representam que o nó é um ponto de demanda para o produto correspondente à coluna e os elementos positivos representam pontos de oferta para o produto associado. Cada coluna de b pode também ser interpretada como um vetor oferta/demanda para o produto associado à coluna, como foi descrito na formulação matemática, i.e., a coluna i pode ser interpretado como o vetor de oferta/demanda para o produto i.

c é uma matriz de dimensão $p \times a$, onde cada linha representa o custo de um produto ao trafegar por um arco, representado pelas colunas. No exemplo apresentado pela Figura 1, os custos são dados apenas em função do arco, porém em muitas situações o custo pode também ser dado em função do produto, além do arco. Neste trabalho foram consideradas instâncias de ambos os casos.

u é um vetor de magnitude igual a a, onde o valor contido em cada posição se refere à capacidade do arco correspondente. Obviamente em casos mais gerais as capacidades são diferentes, porém nesse simples exemplo as capacidade é a mesma para cada arco.

A solução ótima para o exemplo representado pela Figura 1 é a seguinte matriz xo.

$$xo = \begin{pmatrix} 3 & 0 \\ 0 & 0 \\ 0 & 4 \\ 0 & 0 \\ 3 & 0 \\ 0 & 0 \\ 3 & 0 \end{pmatrix}$$

A matriz solução x é de dimensão $a \times p$, onde cada coluna representa o fluxo do produto em cada arco. Na solução xo o ***Produto A*** trafega pelos arcos 1, 5 e 7, enquanto o ***Produto B*** somente pelo arco 3. Outra interpretação, posta na formulação, é interpretar essa matriz como um simples conjunto de vetores, onde cada coluna representa o fluxo de um determinado produto, i.e., a coluna i da matriz solução é referente ao vetor x_i comentado na formulação, representando o fluxo do produto i.

Para a rede multiproduto apresentada na Figura 1 o valor ótimo da função objetivo é 28.

4. Descida Randômica

O método da Descida Randômica é uma variação do método clássico de Descida, porém neste método a busca de todo o espaço de vizinhança não precisa ser feita, como é o caso do método clássico da Descida. Na Descida Randômica qualquer solução de melhora é aceita. Caso o vizinho não seja melhor do que a solução ótima corrente, outro vizinho é gerado. O procedimento é interrompido após um determinado número de iterações em melhora,

chamado neste trabalho de iter_max. Em Souza (2007) é apresentado o pseudo-código do procedimento da Descida Randômica, mostrado na Figura 2.

```
procedimento RandomicoDescida(f(.), N(.), IterMax, s);
1  Iter = 0;        {Contador de iterações sem melhora }
2  enquanto (Iter < IterMax) faça
3      Iter = Iter + 1;
4      Selecione aleatoriamente s' ∈ N(s);
5      se (f(s') < f(s)) então
6          Iter = 0;
7          s ← s' ;
8      fim-se;
9  fim-enquanto;
10 Retorne s;
fim RandomicoDescida;
```

Figura 2 – Pseudo-código do método da Descida Randômica

5. Iterated Local Search - ILS

O método ILS é classificado como uma Metaheurística, pois apresenta uma estratégia de fuga de ótimos locais, através da idéia de melhorar um processo de busca local. No ILS a idéia é aplicar perturbações na solução ótima local e em seguida aplica um outro método de busca nesta solução gerada pela perturbação. Tal perturbação não pode ser muito fraca, pois neste caso a solução poderá não sair do ótimo local encontrado, tampouco muito forte, para evitar um reinício aleatório. A figura 3, encontrada em Souza (2007) apresenta o pseudo-código do método ILS.

```
procedimento ILS
    s0 ← SolucaoInicial
    s ← BuscaLocal(s0)
    iter ← 0
    enquanto (iter < iter_max)
      iter ← iter + 1
      s' ← perturbação(s, histórico)
      s'' ← BuscaLocal(s')
      s ← CriterioAceitacao(s, s', s'')
    fim-enquanto
retorne s
```

Figura 3 – Pseudo-código do método ILS

6. Metodologia

6.1 Representação de uma solução

Uma solução é representada por uma matriz x de dimensão $a.p$, onde são representados os fluxos dos produtos em cada arco. Cada elemento dessa matriz é uma variável, onde o elemento $x_{i,j}$ representa o fluxo do produto j no arco i. Cada coluna i da matriz solução representa o fluxo do produto i.

6.2 Vizinhança de uma solução

Para explorar o espaço soluções do problema, é aplicado um movimento que consiste em trocar o fluxo de um produto escolhido aleatoriamente, de acordo com a factibilidade da solução corrente, i.e., se a solução ótima corrente violar alguma restrição de capacidade, então é selecionado um produto que passe por algum arco violado, sendo que primeiro um arco violado é selecionado aleatoriamente e em seguida um produto que passe por este arco é também selecionado aleatoriamente. No caso de solução ótima corrente factível, é selecionado um produto qualquer e traçada uma nova rota para esse produto. Tal método de seleção do produto para o qual o fluxo será trocado implica em maior chance de obter factibilidade.

6.3 Função de avaliação

Uma solução *x* é avaliada com uma função *f*, tal que:

$$f(x) = Tr.(c.x) + \alpha.\upsilon$$

onde υ corresponde ao somatório das violações nos arcos e $\alpha > 0$ é um parâmetro de penalização, portanto em soluções factíveis υ é igual a zero, isso leva ao algoritmo minimizar a função objetivo sem a penalização. Não foi utilizado nenhum método para tratar restrições de conservação de fluxo porque a solução inicial gerada aleatoriamente não violada nenhuma restrição desse conjunto, como será mostrado nas próximas seções. As matrizes *c* e *x* já foram definidas anteriormente, sendo que *c.x* possui dimensão *p x p*. O fato de considerar o traço da matriz *c.x* se deve ao fato de que o elemento *(i,i)* dessa matriz, com $1 \leq i \leq p$, representa a soma dos produtos dos elementos da linha *i* da matriz *c* (custos do produto *i*) pela coluna *i* da matriz *x* (fluxo do produto *i* na solução gerada).

7. Geração de uma solução inicial

Foi desenvolvido um algoritmo construtivo aleatório, baseado nas matrizes *N* e *b*. A partir da matriz *b* foi construída uma matriz esparsa de oferta/demanda. O exemplo anterior tem a seguinte matriz esparsa associada:

$$Matriz\_esparsa = \begin{pmatrix} 4 & 1 & 3 \\ 2 & 1 & -3 \\ 3 & 2 & 4 \\ 1 & 2 & -4 \end{pmatrix}$$

Na matriz esparsa, cada elemento na primeira coluna representa um nó de oferta ou demanda, a segunda coluna apresenta os produtos e a terceira coluna representa a oferta/demanda do produto correspondente à linha. A dimensão da matriz esparsa é *2p x 3*.

Seja *M* a matriz esparsa, esta heurística construtiva aleatória primeiramente atribui o valor contido em *M(i,1)* a uma variável auxiliar *aux1*, e em seguida verifica quais são os possíveis arcos "varrendo" a linha *aux1* da matriz *N*, para isso, o algoritmo armazena num vetor *vaux* todos os índices das colunas cujo valor presente na linha *aux1* da matriz *N* seja *1*, pois neste caso o nó armazenado em *aux1* será um nó de oferta para o arco. Em seguida seleciona-se aleatoriamente um elemento do vetor dos índices *vaux* e atribui esse valor a uma variável auxiliar *aux2*. Assim na matriz solução *x*, o elemento $x(p_j, aux1)$ receberá o valor contido em *M(aux1, 3),* onde p_j corresponde ao produto *j* para o qual está sendo traçado o fluxo. Antes de fazer essa atribuição na matriz solução, é verificado se o arco *aux2* não será violado ao receber tal valor, se houver violação da capacidade do arco *aux2*, então um novo elemento do vetor *vaux* será selecionado aleatoriamente, gerando um novo valor para *aux2*. Esta nova seleção aleatória em caso de violação do arco *aux2* é feita até um determinado número de

vezes, chamado neste trabalho de *q_max*, se não houver possibilidade de fazer uma atribuição a um arco tal que ele não ultrapasse sua capacidade, então a violação será aceita. Em seguida o índice da linha onde há um valor correspondente a *-1* na coluna *aux2* da matriz *N* é atribuído a *aux1* e o processo é feito novamente para decidir outro arco *aux2* por onde o produto poderá passar. Este processo é reiniciado desde a origem no caso do nó atribuído a *aux1* ser um nó, tal que não existe um valor igual a *1* na matriz *N* na linha aux1, e ele é diferente do valor armazenado em *M(i+1,1)*, correspondente ao nó de demanda para o produto em questão. No caso de *aux1* receber um valor igual ao armazenado em *M(i+1,1),* então o fluxo do produto em questão foi determinado aleatoriamente e o processo continua para o próximo produto. Obviamente a próxima atribuição de aux1 será o valor armazenado em *M(i+1,1).*

Tal procedimento garante factibilidade quanto às restrições de conservação de fluxo, mas não garante factibilidade quanto às restrições de capacidade.

A geração de uma solução x qualquer para o exemplo ficaria da forma descrita:

Seja inicialmente uma solução x onde todos os elementos são nulos.

Passo 1: Atribuir a aux1 o valor contido em M(1,1) e atribuir à variável i o índice da primeira linha da matriz. Portanto aux1 = 4 e i = 1;

$$\begin{pmatrix} 4 & 1 & 3 \\ 2 & 1 & -3 \\ 3 & 2 & 4 \\ 1 & 2 & -4 \end{pmatrix}$$

Passo 2: Selecionar aleatoriamente uma arco na linha aux1 na matriz N, de forma que a entrada do arco selecionado na linha aux1 e na coluna selecionada seja 1, no exemplo só existe uma possibilidade e o arco selecionado foi o 5 e este valor atribuído a aux2, se não houver nenhum valor 1 na linha correspondente e ainda se aux1 ≠ M(i+1,1), voltar ao Passo 1;

$$N = \begin{pmatrix} 1 & 0 & -1 & 0 & 0 & 0 & -1 \\ -1 & 1 & 0 & 0 & 0 & -1 & 0 \\ 0 & -1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 1 & 1 \end{pmatrix}$$

Passo 3: O arco selecionado foi o 5 (aux2), este passo consiste em verificar se este arco não está violado quanto à restrição de capacidade, ou seja, verificar se a soma dos produtos que estão passando por ele não superam a sua capacidade. Se violar, voltar ao Passo 2 e se necessário voltar ao Passo 2 até q_max e aceitar o último arco selecionado. Se não violar ir ao Passo 4;

Passo 4: Como o produto para o qual o fluxo está sendo gerado é o produto 1, cujo fluxo estará na primeira coluna da matriz solução, atribuir o valor armazenado em M(i,3) à posição x(aux2,1) na matriz solução;

Passo 5: Atribuir a aux1 o índice da linha que contém o valor -1 na mesma coluna, neste caso aux1 = 5;

$$N = \begin{pmatrix} 1 & 0 & -1 & 0 & 0 & 0 & -1 \\ -1 & 1 & 0 & 0 & 0 & -1 & 0 \\ 0 & -1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 1 & 1 \end{pmatrix}$$

Passo 6: Verificar se aux1 = M(i+1,1). Em caso afirmativo, o fluxo para o produto 1 termina e os passos recomeçam para o produto 2 com a atribuição i = i + 2, em caso negativo, voltar ao Passo 2;

Refazendo o Passo 2, agora existem duas possibilidades de seleção de um arco, seja selecionado o arco 6;

$$N = \begin{pmatrix} 1 & 0 & -1 & 0 & 0 & 0 & -1 \\ -1 & 1 & 0 & 0 & 0 & -1 & 0 \\ 0 & -1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 1 & 1 \end{pmatrix}$$

Os Passos 3, 4 e 5 são feitos normalmente e aux1 = 2, pois:

$$N = \begin{pmatrix} 1 & 0 & -1 & 0 & 0 & 0 & -1 \\ -1 & 1 & 0 & 0 & 0 & -1 & 0 \\ 0 & -1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 1 & 1 \end{pmatrix}$$

Na verificação contida no Passo 6, aux1 = M(i+1,1), o que significa que terminou a geração do fluxo para o produto 1 e recomeçará a criação de uma rota para o produto 2. Além disso, esse algoritmo retornou a rota para o produto 1 passando pelos arcos 5 e 6, respectivamente. Tal rota pode ser observada na Figura 1.

8. Descida Randômica aplicada ao PFMI

Seja *x* uma solução do problema e *x'* pertencente a uma vizinhança de *x*, definida como os elementos do conjunto *N(x)*. De forma que *x'* é gerado a partir de um movimento *n* realizado em *x*.

Um movimento *n* em *x* é definido como gerar um nova rota para um produto escolhido aleatoriamente. A seleção do produto para o qual será gerada uma nova rota é feita da seguinte forma: se a soma das violações no arco não for nula, é selecionado aleatoriamente um dos arcos violados e em seguida é escolhido aleatoriamente um produto que trafega pelo arco selecionado, no caso da violação ser nula, então é escolhido aleatoriamente um produto qualquer. Esta forma de seleção do produto implica em maior chance da solução corrente se tornar factível. A nova rota também é traçada aleatoriamente, de forma que sejam mantidas as restrições de conservação de fluxo, analogamente ao que foi descrito na geração da população inicial.

O critério de parada do método consiste no número máximo de iterações sem melhora. Neste

trabalho este parâmetro foi chamado de ***itermax***, como já mencionado anteriormente.

9. ILS aplicado ao PFMI

Nesta seção são discutidas características específicas do método ILS aplicado ao problema.

O método ILS foi escolhido para a resolução do Problema de Fluxo Multiproduto Inteiro, sendo o Método da Descida Randômica escolhido como busca local para as soluções geradas a partir de perturbações pelo método ILS.

Uma perturbação no método ILS consiste em trocar o fluxo de mais um produto, ou seja, na perturbação de nível 1, são trocados os fluxos de dois produtos, na perturbação de nível 2, são trocados os fluxos de três produtos, e assim por diante. A valor da perturbação é implementado após um número máximo de tentativas dentro de um mesmo nível. Este número máximo de tentativas num mesmo nível foi chamado neste trabalho de ***vezesnível*** e a quantidade máxima de perturbações de ***ilsmax***.

Após alterar os fluxos na solução corrente, o método da Descida Randômica é executado e caso a solução retornada pelo método de busca seja melhor do que a solução ótima corrente, essa solução é trocada.

Para cada nível de perturbação é considerado o critério de parada por nível como sendo um número máximo de iterações sem melhora, chamado neste trabalho de ***vezes_nível***.

O critério de parada do ILS é o número máximo de perturbações feitas sem melhora durante as perturbações, denominado no trabalho de ***ilsmax***.

10. Resultados

Foram testadas as instâncias contidas no pacote carbim, geradas aleatoriamente pelo ***GenMCF***, que é um gerador aleatório de instâncias para o problema. A coluna *f** contém as soluções ótimas encontradas em Alvelos (2005).

A implementação do Algoritmo foi feita utilizando o *Borland C++ Builder 5.0* e a linguagem de programação utilizada foi a linguagem *C*. Os testes foram realizados em um computador Intel Celeron 1.83 GHz, com 1GB de memória RAM DDR 2, sob o sistema operacional Windows XP Professional Edition SP2.

As tabelas apresentadas possuem grupos de instâncias com características diferentes, sendo as mais relevantes a quantidade de nós (coluna *N*), quantidade de arcos (coluna *A*) e quantidade de produtos (coluna *P*), portanto os parâmetros utilizados foram diferentes para cada grupo de instâncias.

A coluna *fo* apresenta a solução encontrada pelo método descrito, enquanto a coluna *f** a solução ótima encontrada por Alvelos (2005). O tempo total de execução do algoritmo também foi apresentado, além da violação total da solução encontrada, i.e., foi exibido o somatório das violações nos arcos. No caso de violação nula, significa que a solução retornada foi factível. Alvelos (2005) não encontrou solução factível para a instância bl10 no tempo máximo de execução de 1 hora. O algoritmo proposto por Alvelos (2005) não retorna nenhuma solução no caso de não encontrar solução factível dentro do tempo máximo de uma hora.

As instâncias foram classificadas em grupos, sendo que da instância *bl01* até *bl04* corresponde ao *Grupo 1*, de *bl05* até *bl08* ao *Grupo 2*, de *bl09* até *bl12*, *Grupo 3* e de *bl13* a *bl16*, *Grupo 4*. A Tabela 1 apresenta os parâmetros utilizados nos grupos de instâncias.

Grupo	qmax	vezesnivel	ilsmax	itermax	alfa
01	70	200	10	400	80000
02	70	50	10	200	80000
03	70	50	10	200	50000
04	70	50	10	100	20000

Tabela 1 – Parâmetros definidos para os grupos de instâncias

Onde *q_max* é o valor máximo de seleção de um arco, no caso de violação, presente no método construtivo aleatório, *vezesnivel* é o número máximo de iterações sem melhora dentro de um nível de perturbação no método ILS, *ilsmax* é o número máximo de perturbações feitas pelo ILS e *alfa* é o parâmetro relacionado ao método de penalização escolhido para o tratamento das restrições de capacidade, i.e., *alfa* armazena o valor que é multiplicado pelo somatório das violações nos arcos da solução encontrada. As Tabelas 2, 3 4 e 5 apresentam os resultados referentes aos grupos mencionados.

Instância	N	A	P	fo	f*	Tempo	Violação
bl01	32	96	48	1631696	1615947	393,469	0
bl02	32	96	48	1821368	1816947	527,266	0
bl03	32	96	48	17588	17340	581,843	0
bl04	32	96	48	19160	21340	536,141	10

Tabela 2 – Resultados para o Grupo 1

Instância	N	A	P	fo	f*	Tempo	Violação
bl05	32	320	48	504932	474782	579,969	0
bl06	32	320	48	470738	411480	362,594	0
bl07	32	320	48	5906	5751	711,781	0
bl08	32	320	48	6053	5688	175,703	0

Tabela 3 – Resultados para o Grupo 2

Instância	N	A	P	fo	f*	Tempo	Violação
bl09	32	96	192	6663281	6261671	1265,234	6
bl10	32	96	192	6719166		1484,422	0
bl11	32	96	192	69658	69018	1879,984	0
bl12	32	96	192	68221	65902	1393,984	0

Tabela 4 – Resultados para o Grupo 3

Instância	N	A	P	fo	f*	Tempo	Violação
bl13	32	320	192	3453719	3132695	983,453	0
bl14	32	320	192	2951235	2433011	1664,516	0
bl15	32	320	192	37058	34274	1140,734	0
bl16	32	320	192	31828	28074	1578,391	0

Tabela 5 – Resultados para o Grupo 4

11. Conclusões

O método proposto se mostrou eficiente, obtendo resultados próximos aos ótimos e encontrando uma solução factível para uma instância bl10 em um tempo menor que o testado por Alvelos (2005), como mencionado anteriormente, Alvelos (2005) não encontrou solução

factível para essa instância em um tempo de 1 hora de execução do algoritmo proposto em seu trabalho. As instâncias dos Grupos 1 e 2 são pequenas e fáceis de serem resolvidas por métodos exatos, porém as instâncias dos Grupos 3 e 4 são maiores, podendo ser consideradas grandes e o comportamento do método proposto foi bom, com soluções próximas às soluções ótimas.

Referências

Hu, T. C., Multicommodity network flows, Operations Research, 11:344–360, 1963.

Hu, T.C., Integer Programming and Network Flows, Addison-Wesley, 1969.

Buriol, L. S., Roteamento do tráfego na internet: algoritmos para projeto e operação de redes com protocolo ospf, 2003.

R. et al. Milidiu, Um algoritmo grasp para o problema de transporte de derivados de petróleo em oleodutos, Anais do XXXIII Simpósio Brasileiro de Pesquisa Operacional, 237–246, 2001.

Shan, Y.S., A Dynamic Multicommodity Network Flow Model for Real Time Optimal Rail Freight Car Management, PhD thesis, Princeton University, Canada, 1985.

Alvelos, F.P., Branch-and-price and multicommodity flows, PhD thesis, Departamento de Produção e Sistemas da Escola de Engenharia, Universidade do Minho, Portugal, (2005).

Becceneri, J. C., Meta-heuristicas e otimização - curso de verão. Grupo de Pesquisa Operacional, Laboratório Associado de Computação e Matemática Aplicada (LAC), INPE, 2007.

Magnanti, T. L., Orlin e Ahuja, J. B., R. K. Network flows. Prentice Hall, 1993.

Bocanegra S. e Campos F. Santos, M. A. (2000), Uma proposta de soluçãoao para o problema não linear de fluxo multiproduto utilizando pontos interiores, Semana de Ciência da Computação, UFL, Lavras/MG, 69–73.

Gondzio, J., Sarkissian, R., Vial, J. P. Goffin, J. L., Solving nonlinear multicommodity flow problems by the analytic center cutting plane method. Mathematical Programming, 76:131–154, 1996.

Oldham J. D., Plotkin, S. e Stein, C. Goldberg, A. V, An implementation of a combinatorial approximation algorithm for minimum-cost multicommodity flow, em Bixby, R. E., Boyd, E. A. e Rioz-Mercado, R. Z. (Eds), Proceedings of the 6th Internacional Integer Programming and Combinatorial Optimization Conference. Lecture Notes in Computer Sciences, volume 1412, Berlin, German, 1998. Springer-Verlag.

Park, S., Kim, D. e Lee, K., An integer programming approach to the path selection problems. In Proceedings of INOC2005 - International Network Optimization Conference, Faculdade de Ciências, Universidade de Lisboa, Lisboa, Portugal, 2002.

E. C. G. et al. Wille., Topological design of survivable ip networks using metaheuristic approaches, *Third Internation Workshop on QoS in Multiservice IP Networks*, 191–206, 2005.

Souza, M. J. F. *Inteligência Computacional para Otimização, Notas de aula*, Departamento de **Computação, Universidade Federal de Ouro Preto, disponível em** http://www.decom.ufop.br/prof/marcone, 2007.