MS877

Problema do Carteiro Chinês: Estudo da modelagem e software XNÊS

Orientadora: Profa. Dr. Maria Aparecida Diniz Ehrhardt
Aluna: Fernanda Bia Peteam

DMA - IMECC - UNICAMP

20 de novembro de 2013

1 Resumo

Na segunda parte do projeto, de acordo com o cronograma, foi lido e estudado o artigo de Negreiros Gomes et al. [3] sobre o Problema do Carteiro Chinês (PCC), com enfoque em duas subdivisões do problema, o caso simétrico (não orientado) e o caso dirigido.

Os estudos se referiram à modelagem matemática usada para a resolução dos dois casos, assim como a seus respectivos pseudo-algoritmos. Como o artigo está ligado ao desenvolvimento do software XNÊS e suas aplicações, seu estudo requereu uma familiaridade e aprendizado do funcionamento do programa para aplicação de exemplos. O próprio pacote já possui alguns exemplos numéricos, tanto para grafos gerados quanto para grafos apoiados em mapas de cidades reais.

2 Produção Científica

Esta seção está baseada na leitura e estudo do artigo "O PROBLEMA DO CARTEIRO CHINÊS, ALGORITMOS EXATOS E UM AMBIENTE MVI PARA ANÁLISE DE SUAS INSTÂNCIAS: SISTEMA XNÊS"[3]

2.1 Sobre o XNÊS

Neste artigo, seus idealizadores explicam os algoritmos exatos para a resolução do PCC a partir dos quais o desenvolvimento do software XNÊS foi baseado e, então, usaram esse software para testes computacionais com exemplos, incluindo problemas em cidades reais.

O software XNÊS foi idealizado para gerar grafos (simétricos, orientados e mistos) e soluções exatas para o PCC, usando Modelagem Visual Interativa (MVI), isto é, efeitos visuais nos vértices e ligações. O ambiente é gráfico e possui, como principal componente, um gerador de instâncias de grafos, que podem ser descritas conforme a necessidade do usuário (tipos de grafos). O XNÊS permite manipular grafos dinamicamente e de diversas formas, incluindo/excluindo vértices, mexendo com pesos nas arestas/arcos e permite que os efeitos das mudanças sejam vistas através de uma representação visual do grafo. O sistema verifica automaticamente problemas como grafos que não sejam fortemente conexos, retornando mensagem de erro. Possui um link para uma tabela com informações sobre os pesos e custos das arestas que podem ser alterados, assim como uma janela com informações sobre o grafo: número de vértices, arestas orientadas e não orientadas, custo total do grafo (soma dos custos das arestas). Após a execução dos algoritmos, o sistema abre uma caixa com o custo total da rota e a possibilidade de visualizar o percurso gerado em uma animação dos movimentos do carteiro sobre o grafo em uso e um diagrama de fluxo, onde e possível visualizar todo o percurso.

Ao usar o comando para gerar a instância pode-se determinar o tempo máximo (em segundos) que o programa permanecerá procurando a solução, pois, como é um algoritmo exato, ele tende a procurar a solução ótima exata, podendo demorar muito tempo. Assim, limitando o tempo para um tempo menor que o necessário para encontrar a solução exata, pode-se acabar obtendo não um ótimo, mas um caminho muito próximo, com pouca dispersão. A equipe de desenvolvimento do software (disponível no link Ajuda → Sobre, na interface do programa) está a seguir:

UECE
- Augusto Wagner de Castro Palhano;
- Wiler Rodrigues Coelho Junior;
- Emanuel Ferreira Coutinho;
- Gerson Alves de Castro.

UFES
- Bruno Fernandes Rezende;
- Giafrancesca Cutini Barcelos;
- Lucio Wagner Lessa Pereira.

Orientação
- Marcos José Negreiros Gomes - DSc UECE;
- Francisco José Negreiros Gomes - DSc UFES.

Apoio GRAPHVS Cons. Com. & Rep. Ltda.

O software está disponível no site www.graphvs.com.br.

2.2 Sobre o PCC

O Problema do Carteiro Chinês é dividido de acordo com os tipos de grafos: não orientados (simétricos), orientados e mistos, podendo ter custos fixos e não fixos. A representação matemática de um grafo é da seguinte maneira: $G = (V, L)$ um grafo, formado pelo conjunto V de nós (ou vértices) e um conjunto L de arestas (ou ligações), onde o conjunto de arestas pode ser ampliado para uma dupla de conjuntos $L = (E, A)$, onde E é o conjunto formado por arestas não orientadas (elos) e A é o conjunto de ligações orientadas (arcos) entre vértices de G. Assim, a classificação dos tipos de grafos fica:

1. **Problema do Carteiro Chinês - Caso Simétrico (CPP)**: gerar percurso de custo mínimo sobre um grafo $G = (V, E)$ (E conjunto de arestas - sem orientação), valorado e conexo, a partir de um vértice de origem $v_0 \in V$;

2. **Problema do Carteiro Chinês - Caso Dirigido/Orientado (DCPP)**: gerar percurso de custo mínimo sobre um grafo $G = (V, A)$ (grafo orientado), valorado e fortemente conexo (f-conexo), a partir de um vértice de origem $v_0 \in V$;

3. **Problema do Carteiro Chinês - Caso Misto (MCPP)**: gerar um percurso de custo mínimo sobre um grafo $G = (V, E, A)$, valorado e f-conexo, a partir de um vértice de origem $v_0 \in V$;

4. **Problema do Carteiro Chinês - Caso Íngreme (WPP)**: gerar um percurso de custo mínimo sobre um grafo $G = (V, E)$, valorado e conexo, a partir de um vértice de origem $v_0 \in V$, onde os elos de E podem ter custos distintos de travessia de i para j (c_{ij}) e de j para i (c_{ji}), i.e., $c_{ij} \neq c_{ji}$.

Cabe aqui acrescentar uma definição com relação a grafos, não mencionada no Relatório Parcial:

Um **grafo** é dito ser **fortemente conexo** (f-conexo) se todo par de vértices está ligado por pelo menos um caminho em cada sentido, ou seja, se cada par de vértices participa de

um circuito. Isto significa que cada vértice pode ser alcançável partindo-se de qualquer outro vértice do grafo.

Aprofundaremos o estudo dos dois primeiros casos descritos acima, o caso simétrico (CPP) e o caso dirigido/orientado (DCPP), cujos tempos de resolução são polinomiais (soluções exatas são viáveis).

2.2.1 Versão Simétrica do PCC

No caso simétrico as arestas são valoradas de maneira que o percurso entre quaisquer pares de vértices seja o mesmo tanto na ida quanto na volta.

Como a solução ótima dá-se para um grafo euleriano e, pelo *Teorema de Euler*, um grafo é euleriano se, e somente se, todos os seus vértices possuírem grau par, se o grafo estudado G possuir somente vértices de grau par, basta apenas encontrar um caminho euleriano. Caso contrário, devemos transformar o grafo inicial G, que possui vértices com grau ímpar, em um multigrafo G' onde acrescentam-se arestas ao grafo inicial para que todos os seus vértices possuam grau par.

Um **multigrafo** é um grafo que possui arestas múltiplas entre vértices como, por exemplo, na representação das sete pontes de Königsberg, problema descrito no Relatório Parcial [1].

Essa transformação do grafo G para o grafo G' dá-se pelos algoritmos de 1-*matching* [2],[6].

O *matching* (emparelhamento) é realizado para transformar um grafo não euleriano em um euleriano, atuando nos vértices de grau ímpar, onde são inseridas novas arestas (artificiais), no caso corrente, elos (arestas não orientadas), para deixá-los com grau par. A funcionalidade dos algoritmos de emparelhamento está diretamente ligada ao *Teorema 2* do Relatório Parcial, que diz que o número de vértices de grau ímpar é par. Daí, os elos a serem inseridos seriam encontrados através da busca do caminho mínimo entre cada par de vértices de grau ímpar. Esses elos com caminho mínimo inseridos seriam o equivalente a passar mais de uma vez pelo mesmo elo (distância mínima) entre cada par de vértices de grau ímpar (tornando-o um vértice de grau par), para ter a possibilidade de "chegar e sair"do vértice.

O algoritmo para a transformação de um grafo em um grafo euleriano abordado no artigo estudado é o de 1-*MatchingValorado* proposto por Edmonds & Johnson [2], que busca a melhor combinação de pares de vértices para acrescentar os elos artificiais.

Formulação matemática do problema [2]:

PCC Simétrico:

$$\min \sum_{i,j \in E} c_{ij} x_{ij} \tag{1}$$

sujeita a:

$$\sum_{(v_i,v_j) \in E(S)} x_{ij} \geq 1, \quad S \subset V, \tag{2}$$

$$x_{ij} \in Z_+, \quad \forall (v_i, v_j) \in E(S), \tag{3}$$

onde S é o conjunto dos vértices de grau impar.

Para qualquer conjunto próprio S não vazio de V ($S \subsetneq V$), definimos $E(S) = \{(v_i, v_j) \mid v_i \in S, v_j \in V - S$ ou $v_i \in V - S, v_j \in S\}$ e x_{ij} o número (inteiro) de vezes que um elo de G será repetido em G'.

Essa formulação visa encontrar o grafo G' de forma a minimizar a quantidade de vezes que um elo se repete e repetir elos com menor custo/distância possível. Isso é visto na primeira restrição, que diz que deve haver pelo menos uma repetição do elo de distância mínima/caminho mínimo (função minimizada na formulação) quando há pelo menos um vértice de grau ímpar, enquanto houver vértices de grau ímpar. A segunda restrição diz que o número de vezes que um elo de G será repetido em G' é inteiro e positivo para qualquer elo em $E(s)$ (nulo não entra, pois estaria em contradição com a primeira restrição).

Para encontrar então o multigrafo euleriano mínimo temos o seguinte esquema:

Procedimento *PCC - Simétrico* $(G, G', p(G'))$ [2]

Algoritmo genérico para o PCC - Simétrico
Entrada: G - Grafo (V, E) valorado nas arestas e simétrico
Saída: G' - Grafo Euleriano Mínimo (grafo contendo as arestas repetidas de G)
$p(G')$ - função perímetro do grafo aumentado

Passo 1: Definir $S \subset V$, conjunto dos vértices de grau ímpar;

Passo 2: Encontrar o grafo completo ($K_{|S|}$) valorado com os percursos mínimos entre quaisquer vértices de S;

Passo 3: Aplicar *1-MatchingValorado* $(S, E(S), G')$ - retorna G';

Passo 4: Definir $p(G') = \sum_{i,j \in E'} c_{ij} x_{ij}$ (perímetro do multigrafo aumentado, onde E' é o conjunto dos elos de G'.

Em seguida encontra-se o percurso euleriano sobre G'. Para isso, foi utilizado um algoritmo genérico proposto a partir de [5].

2.2.2 Versão Orientada do PCC

A condição necessária e suficiente para a existência de um circuito euleriano em um grafo orientado é que, além de ser f - conexo, o grafo deve ser simétrico, isto é, para cada nó o grau de entrada (quantidade de arcos que entram no nó) deve ser igual ao grau de saída do nó (quantidade de arcos que saem do nó). Quando isso não acontece é necessário que sejam acrescidas cópias apropriadas de alguns arcos. Para isso, determina-se um multigrafo $G = (V, A)$, com G f-conexo, através de um algoritmo de fluxo de custo mínimo (ou circulação

de custo mínimo) em uma rede ou através de uma transformação de G num grafo para o Problema de Transporte.

O Problema do Fluxo de Custo Mínimo é um problema que tem como objetivo minimizar o custo total de envio de uma oferta disponível através da rede para satisfazer uma demanda dada. Uma das principais aplicações está em planejar uma rede de distribuição de uma certa companhia.

O Problema de Transporte consiste em encontrar a forma mais econômica/eficiente de enviar algum produto/objeto disponível, em quantidades limitadas, de determinados locais (origem, fábrica, centros de distribuição) para outros (clientes, consumidores finais), de forma a esgotar as disponibilidades de cada origem e satisfazer as necessidades em cada destino, onde os custos de transporte de cada origem para cada destino são proporcionais às quantidades transportadas.

Problemas como o de transporte, de designação, de caminho mais curto e de fluxo máximo são casos especiais do problema de fluxo de custo mínimo.

- *Formulação do Problema Clássico de Transporte*

  - m pontos de origem (fábricas, centros de produção), com produção de a_i unidades de determinado produto, $i = 1, 2, \ldots, m$;
  - n pontos de destino (centros de consumo), com demanda de b_j unidade deste produto, $j = 1, 2, \ldots, n$;
  - custos unitários de transporte c_{ij} de cada origem i para cada destino j, com $i = 1, 2, \ldots, m$ e $j = 1, 2, \ldots, n$.

  Temos:

  $m + n$ nós, aos quais são associadas as ofertas a_i e demandas b_j, $a_i > 0$ e $b_j > 0$;

  $m.n$ arcos, aos quais são associados custos unitários de transporte c_{ij}.

  Variável:

  x_{ij}: quantidade transportada da origem i para o destino j, $\forall i, j$, $i = 1, 2, \ldots, m$ e $j = 1, 2, \ldots, n$.

  Hipótese: $\sum_{i=1}^{m} a_i = \sum_{j=1}^{n} b_j$.

Formulação:

$$\min z = \sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}x_{ij} \tag{4}$$

sujeita a

$$\sum_{j=1}^{n} x_{ij} = a_i, \quad i = 1, \ldots, m; \tag{5}$$

$$\sum_{i=1}^{m} x_{ij} = b_j, \quad j = 1, \ldots, n; \tag{6}$$

$$x_{ij} \geq 0, \quad i = 1, \ldots, m, \quad j = 1, \ldots, n. \tag{7}$$

A restrição (5) é a restrição de oferta e a restrição (6) é a de demanda, são as restrições de conservação de fluxo.

- **Determinação do multigrafo euleriano mínimo através do Problema do Transporte:**

  Seguindo a notação usada em [3], a formulação adotada foi a formulação do Problema Clássico de Transportes, onde os vértices com excesso de entrada são considerados como *suprimento (fonte)* e os vértices com excesso de saída como *demanda (sumidouro)*:

  (i) $d^+(v_i)$ representa o número de arcos saindo do vértice v_i e $d^-(v_i)$ o número de arcos entrando em v_i;

  (ii) I é o conjundo dos vértices v_i, onde $d^+(v_i) > d^-(v_i)$ e daí ligados a um novo vértice f denominado fonte;

  (iii) J é o conjunto dos vértices v_i onde $d^+(v_i) < d^-(v_i)$ e, daí, ligados a um novo vértice s denominado sumidouro.

Quando $d^+(v_i) = d^-(v_i)$, a soma dos arcos que saem e entram no vértice é par e o grafo já seria, então, euleriano e seria necessário apenas encontrar um caminho euleriano.

(**PCC Orientado)**

$$\min \sum_{v_i \in I}\sum_{v_j \in J} c_{ij}x_{ij}. \tag{8}$$

sujeita a:

$$\sum_{v_j \in J} x_{ij} = d^+(v_i) - d^-(v_i), \quad \forall v_i \in I; \tag{9}$$

$$\sum_{v_i \in I} x_{ij} = d^-(v_j) - d^+(v_j), \quad \forall v_j \in J; \tag{10}$$

$$x_{ij} \geq 0, \quad \forall v_i \in I, \quad \forall v_j \in J. \tag{11}$$

A função objetivo visa minimizar o custo da repetição de arcos de acordo com o peso (distância) dos arcos. A primeira restrição (9) pode ser associada à restrição de oferta e a restrição (10) à restrição de demanda do Problema de Transporte.

A solução do Problema de Transporte indica qual nó de suprimento deve ser associado em qual demanda.

Para definir um procedimento genérico que pudesse ser aplicado também ao caso misto, os idealizadores do artigo utilizaram a implementação do algoritmo primal-dual (*Out-of-Kilter*), baseado no de Bazaraa *et al.* [4]; para o Problema de Custo de Fluxo Mínimo, cuja modelagem matemática é dada a seguir.

- *Formulação do Problema de Fluxo de Custo Mínimo*

  $G = (V, A)$ f-conexo;

  $V = \{1, \ldots, m\}$, conjunto de m vértices;

  A: conjunto de n arcos (i, j), orientados de i para j.

  Para cada vértice i associamos quantidades b_i tal que:

  (a) $b_i > 0$: vértice i é centro de produção e $|b_i|$ é a quantidade produzida de determinado produto;

  (b) $b_i < 0$: vértice i é centro de consumo e $|b_i|$ é a quantidade demandada do produto;

  (c) $b_i = 0$: vértice i é vértice de transbordo ou intermediário.

  Para cada arco (i, j) temos:

  (a) c_{ij}: custo unitário de transporte do produto da origem i ao destino j;

  (b) l_{ij}: limite inferior de fluxo de i para j;

  (c) u_{ij}: limite superior de fluxo de i para j.

  Variável: x_{ij}: quantidade de fluxo que vai da origem i para o destino j.

  Hipótese: oferta total = demanda total, isto é, $\sum_{i=1}^{m} b_i = 0$.

  Caso contrário:

  (i) $\sum_{i=1}^{m} b_i > 0$: mais oferta. Cria-se, então, um vértice fictício $m + 1$ com $b_{m+1} = -\sum_{i=1}^{m} b_i$ e arcos fictícios ligando cada centro de produção com o vértice fictício com $c_{i,m+1} = 0$, $l_{i,m+1} = 0$ e $u_{i,m+1} = \infty$. No final, um $x_{i,m+1} > 0$ indica o excesso de produção da fonte i;

(ii) $\sum_{i=1}^{m} b_i < 0$: mais demanda. Cria-se, então, um vértice fictício $m+1$ com $b_{m+1} = -\sum_{i=1}^{m} b_i$, e arcos fictícios ligando o vértice m+1 com os centros de consumo, com $c_{m+1,j} = 0$, $l_{m+1,j} = 0$ e $u_{m+1,j} = \infty$. No final, $x_{m+1,j} > 0$ indica a demanda não atendida no centro de consumo j.

Objetivo: minimizar custo total de transporte.

Restrições:

- Satisfazer oferta e demanda dos vértices mais canalização (vértices artificais para quando $\sum_{i=1}^{m} b_i \neq 0$);
- Restrição de conservação de fluxo: fluxo líquido no vértice i deve ser b_i, isto é, (fluxo total que sai do vértice) - (fluxo total que chega no vértice) = b_i.

De modo geral:

$$\min z = \sum_i \sum_j c_{ij} x_{ij}, \quad \forall (i,j) \in A. \tag{12}$$

sujeita a:

$$\sum_j x_{ij} - \sum_k x_{ki} = b_i, \quad i = 1, \ldots, m, \quad \forall (i,j), (k,i) \in A; \tag{13}$$

$$l_{ij} \leq x_{ij} \leq u_{ij}, \quad \forall (i,j) \in A. \tag{14}$$

Notação: ao invés de dois índices, (i,j), usar apenas um índice: a numeração dos arcos de 1 até n.

Exemplo:

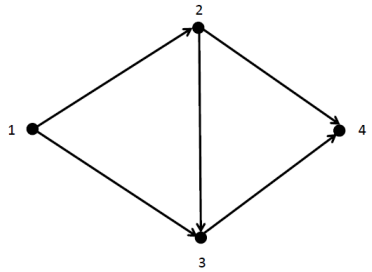

Figura 1: Exemplo grafo orientado.

Neste exemplo, $b_1 = 10$, $b_2 = 10$, $b_3 = -8$, $b_4 = -12$ ($\sum_{i=1}^{4} b_i = 0$), e:

Arco	Vértice Origem	Vértice Destino	Custo	Lim. Inferior (l)	Lim. Superior (u)
1	1	2	2	0	5
2	1	3	3	0	5
3	2	3	6	0	5
4	2	4	7	0	5
5	3	4	1	0	5

Tabela 1: Informações do grafo da Figura 1.

Para encontrar o grafo euleriano a partir do grafo G de forma mais geral, Negreiros Gomes et al. [3], usaram o algoritmo primal-dual (*Out-of-Kilter*) sobre o PCC adaptado ao Problema de Fluxo de Custo Mínimo, isto é, transformando os atributos de distância em atributos de fluxo e custo. Então, o limite inferior para todo arco (i, j) é $l_{ij} = 1$ e o limite superior é $u_{ij} = \infty$ e a distância representará o custo c_{ij} do arco.

Procedimento *PCC - Orientado* (G, G')

Algoritmo genérico para o PCC - Orientado
Entrada: G - Grafo (V, E) valorado nas arestas e simétrico
Saída: Percurso Euleriano

Passo 1: Tomar G, com as informações de distância e testar se o grafo é f-conexo;

Passo 2: Se f-conexo, transformar as informações de distância para informações de fluxo, com os limites de fluxo máximo e mínimo permitidos por arco, e aplicar um algoritmo de fluxo em redes de custo mínimo, *Out-of-Kilter* [4] para obter G';

Passo 3: Após a obtenção de G', onde os resultados de fluxo indicam a quantidade de vezes que cada arco será repetido em G', aplicar o resultado encontrado para formação da rota euleriana, usando o mesmo algoritmo usado para obtenção do caminho euleriano no PCC - Simétrico [5];

2.3 Resultados Numéricos

No software podem-se criar grafos, seus vértices, elos e arcos dinamicamente e direto na interface. Em uma tabela, na interface, inserimos os custos de cada arco/aresta do grafo. O grafo é salvo no programa no formato .grf. O programa verifica se o grafo é fortemente conexo e depois resolve a instância (escolhendo, antes, o vértice de origem). Abre uma janela informando o custo total do percurso e depois gera, dinamicamente, o percurso ótimo. Pode-se, também, criar grafos a partir de mapas e depois procedendo da mesma forma, testando f-conexidade e depois gerando a instância do grafo. No pacote vem alguns exemplos de grafos, com alguma soluções, que auxiliam na melhor compreensão do funcionamento do programa.

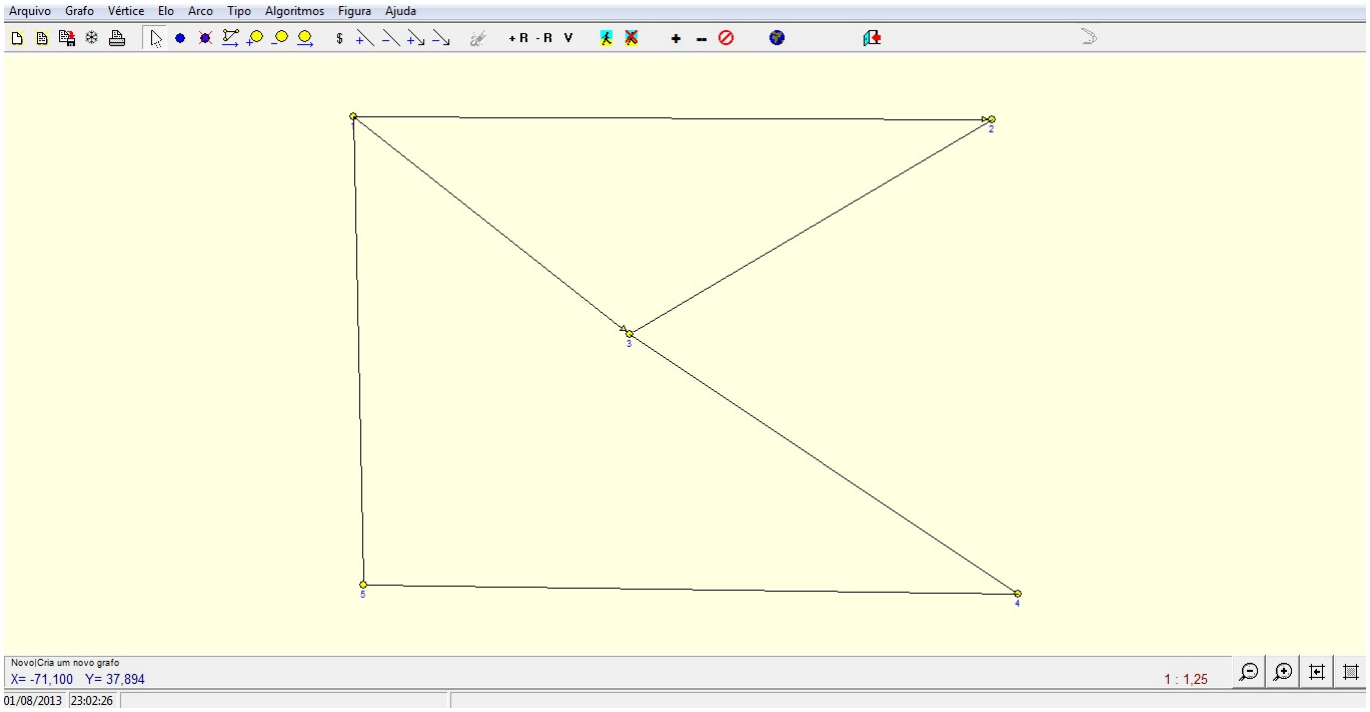

Figura 2: Grafo do Exemplo 1.

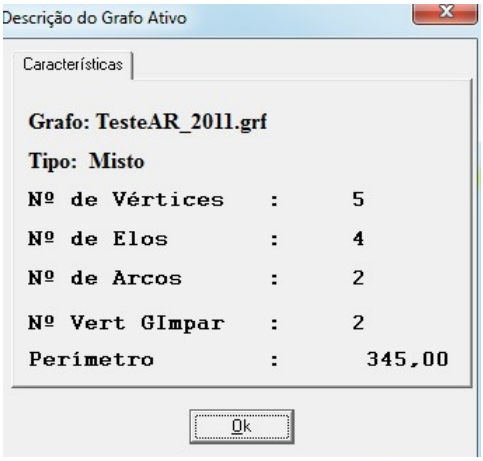

Figura 3: Descrição do grafo do Exemplo 1.

Na Figura 2 é possível ver a interface do programa. Na segunda barra superior temos as ferramentas para criar e analisar grafos dinamicamente, como criar vértices, elos, arcos que saem ou que entram; alterar os custos dos mesmos; inserir um plano de fundo no grafo como, por exemplo, o mapa do local que está sendo representado pelo grafo; assim como ferramentas para mexer com o arquivo como imprimir a tela atual; salvar o grafo atual; abrir algum grafo já existente e abrir uma nova página. Na primeira barra superior temos as alterações no arquivo: abrir grafos ou soluções já existentes, já salvas; abrir multigrafo solução; fornecer solução no formato .txt; e mais algumas alterações no grafo: marcar origem; exibir os rótulos dos vértices; descrição do grafo (quantos vértices, quantos elos, quantas arestas, perímetro do grafo); algumas funções já ditas antes, como inserir e remover vértices e arestas, determinar o

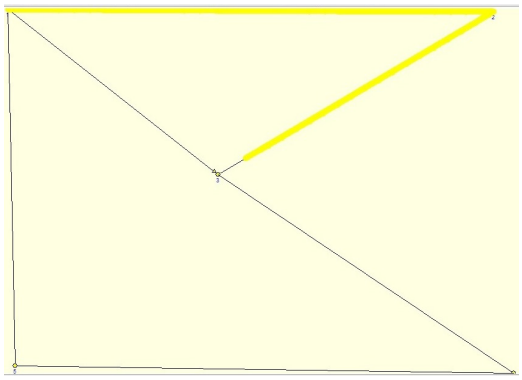

Figura 4: Percurso gerado dinamicamente.

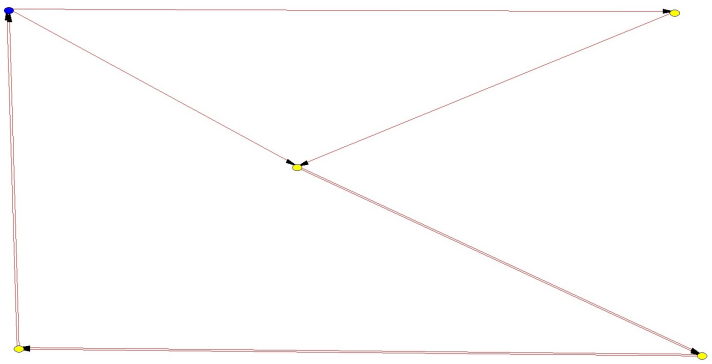

Figura 5: Multigrafo solução.

tipo de grafo em que está trabalhando (não orientado, orientado ou misto); o comando para verificar a f-conexidade e para resolver a instância atual (o grafo atual). É possível construir grafos orientados, não orientados e mistos, alterando os custos de seus elos e/ou arcos através de uma tabela que contém informações dos elos e arcos, podendo alterar, também, a escala com a qual será trabalhada. Isso deve-se ao fato de serem utilizados, principalmente, mapas reais, onde os percursos devem ser definidos.

2.3.1 Testes realizados

-Exemplo 1

Na Figura 2 vemos também um exemplo de um grafo misto com 5 vértices, 4 elos, 2 arcos e com 2 vértices possuindo grau ímpar e cujo perímetro é 345,00 e é fortemente conexo. Na escala manual temos os seguintes custos para os elos e arcos:

Origem	Destino	Ligação	Custo
1	2	Arco	40
1	3	Arco	50
1	5	Elo	110
2	3	Elo	30
3	4	Elo	55
4	5	Elo	60

Tabela 2: Informações grafo simples.

Na Figura 5 podemos notar que o vértice intitulado como 1 está em azul, isso indica que ele é o vértice origem do percurso, é da onde o carteiro parte inicialmente. E podemos calcular, para esse caso, o custo à mão e vemos que o custo do percurso que o carteiro deve percorrer possui custo de 570,00 o que é coerente com o resultado de custo do percurso fornecido pelo programa logo antes de gerar o caminho dinamicamente. Na Figura 4 nota-se um caminho em amarelo enquanto o resto do grafo está com linhas pretas. Essa linha amarela é um instantâneo do software gerando o caminho dinamicamente. A cor amarela segue o grafo todo, mostrando passo a passo o percurso ótimo. A Figura 5 é o multigrafo solução do problema e mostra, estaticamente, o percurso percorrido através das setas e quantas vezes o carteiro passou pelo

mesmo elo ou arco, partindo do vértice 1 (em azul).

-Exemplo 2

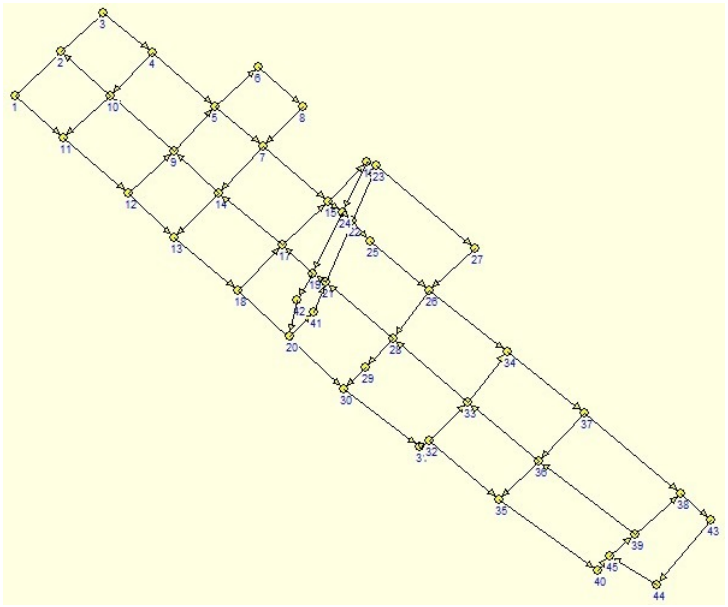

Figura 6: Grafo do Exemplo 2

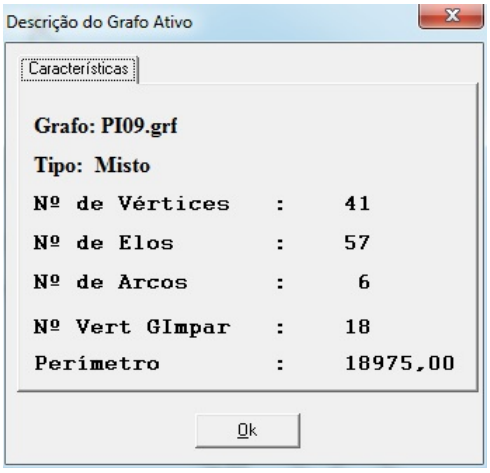

Figura 7: Descrição do grafo do Exemplo 2.

Na Figura 6 podemos ver que o grafo é misto e, na descrição do grafo (Figura 7) temos as informações de quantos vértices, elos e arcos o grafo possui. Podemos visualizar e também editar os custos dos elos e arcos do grafo, assim como em qualquer outro grafo. Testando a f-conexidade do grafo, temos que o mesmo é fortemente conexo. Nas figuras 9 e 10 temos, respectivamente, o percurso sendo gerado dinamicamente e o multigrafo solução, indicando os caminhos repetidos, partindo do vértice 1 (em azul), tomado como o vértice de origem do problema. O custo fornecido pelo programa ao rodar é de 11125.

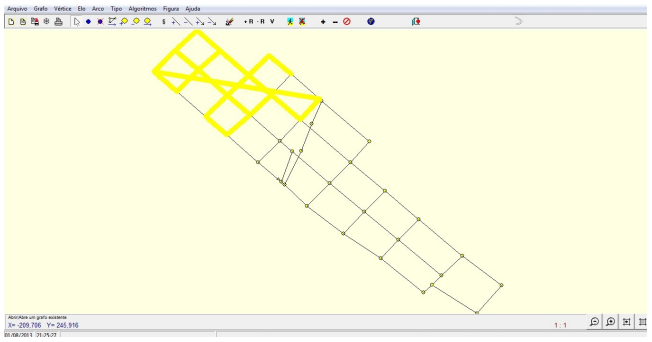

Figura 8: Percurso sendo gerado dinamicamente.

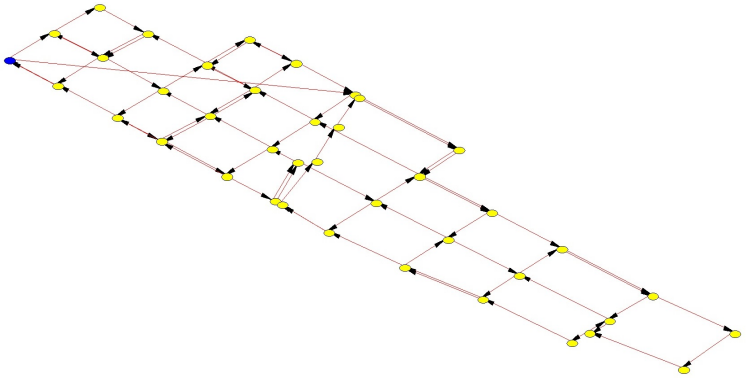

Figura 9: Multigrafo Solução.

3 Referências Bibliográficas

[1] A. Constantino, Departamento de Informática, Universidade Estadual de Maringá, Notas de aula: Grafos Eulerianos e o Problema do Carteiro Chinês.

[2] J. Edmonds & E. Johnson,(1973). Matching, Euler tours and the Chinese postman problem. Mathematical Programming, 5, 88-124.

[3] M.J. Negreiros Gomes, W. Coelho Jr., A. W. de Castro Palhano, E. Ferreira Coutinho, G. Alves de Castro, F.J. Negreiros Gomes, G. Cutini Barcellos, B. Fernandes Rezende, L. W. Lessa Pereira (2009). O problema do carteiro chinês, algoritmos exatos e um ambiente MVI para análise de suas instâncias: sistema XNÊS. Pesquisa Operacional, 29 (2) pp. 323–363.

[4] N.S.,Bazaraa; Jarvis, J.J. & Sherali, H.D. (1990). Linear Programming and Network Flow.New York, John Wiley & Sons, Singapore.

[5] R.E. Burkard & U. Derigs. (1980). Assignment and Matching Problems: Solution Methods with Fortran programs. In: Lecture Notes in Economics and Mathematical Systems, vol. 184, Spring-Verlag, Berlin.

[6] U. Derigs, (1988). Programming in Networks and Graphs – On the combinatorial background and near-equivalence of network flow and matching algorithms. In: Lecture Notes in Economics and Mathematical Systems[edited by M. Beckmann and W. Krelle,], vol. 300, Springer-Verlag.