Relatório de Atividades

ESTUDO E IMPLEMENTAÇÃO DE ALGORITMOS PARA

RESOLUÇÃO DE PROBLEMAS COMBINATÓRIOS EM GRAFOS

Bolsista: Ralphy Antonio Martin Castilho

Orientador: Dr. Horacio Hideki Yanasse

ÍNDICE

RESUMO

O estudo foi limitado a alguns algoritmos para resolução de problemas combinatórios em grafos utilizados na otimização de roteamentos de veículos, coletas de lixo, entrega de correspondências, assim como distribuição, em pontos estratégicos, de redes escolares e de postos de saúde.

Os algoritmos escolhidos foram os que resolvem o menor caminho de um nó à outro (Dijkstra) e entre todos os pares de nós (Floyd), o menor percurso entre todos os arcos de um grafo orientado (Carteiro Chinês para grafos orientados) e localização de facilidades (Problema das Medianas).

Os algoritmos citados estão sendo implementados em linguagem C++ orientado ao objeto no compilador Borland C++ 3.11, 1992.

INTRODUÇÃO

Neste projeto tem-se como objetivo principal o estudo e implementação de alguns algoritmos para resolução de problemas combinatórios em grafos. Os grafos são representações extremamente úteis para aplicações diversas pois podem simbolizar mapas, facilitando visualmente o entendimento e consequentemente a resolução de diversos problemas combinatórios. Os grafos vem sendo aplicados em situações práticas, na otimização do gerenciamento de rotas para serviços de transporte urbano, coleta de lixo, distribuição de correspondência, limpeza pública, vigilância policial, transporte ambulatorial, e de distribuição de pontos de fácil e rápido acesso para redes escolares, postos de saúde, delegacias e bombeiros.

O início das atividades no projeto foi em Fevereiro de 1999, em substituição ao bolsista Rudini Menezes Sampaio. O ex-bolsista implementou uma série de algoritmos para resolução de problemas em grafos durante os períodos anteriores e se desejava incorporá-los em um Sistema de Informações Geográficas (SIG).

ATIVIDADES DO PERÍODO

Nos meses de Fevereiro e Março foram reforçados os conhecimentos em linguagem C e iniciado a pesquisa sobre grafos e suas aplicações.

No mês Abril foi feita uma análise superficial do Trabalho de Iniciação Científica (com duração de quase dois anos), realizado pelo bolsista Rudini Menezes Sampaio.

Logo em seguida, por uma escolha infeliz, foi direcionada a atenção ao software de SIG ARC/INFO, que pela complexidade e falta de especialistas, tomou quase três meses sem grandes conclusões.

Nos meses de Agosto, Setembro e Outubro foram pesquisados e estudados os algoritmos de Dijkstra, Floyd, Caixeiro Viajante e Carteiro Chinês para grafos não orientados.

Em Novembro iniciou-se o estudo da linguagem C++, com enfoque à orientação ao objeto, devido ao que foi proposto na bolsa.

Em Dezembro e Janeiro continuou-se o estudo da linguagem C++.

Nos meses de Fevereiro e Março foram estudados os algoritmos do Carteiro Chinês para grafos orientados e o algoritmo das Medianas.

Em Abril iniciou-se a implementação dos algoritmos de Dijkstra, Floyd, Carteiro Chinês para grafos orientados e o algoritmo das Medianas

DEFINIÇÃO DOS TERMOS E NOTAÇÕES

Um GRAFO é uma entidade abstrata G(N, A) formada por um conjunto finito N de nós e um conjunto finito A de arestas que conectam os nós. Uma aresta que possui uma orientação relacionada a ela é denominada de ARCO. Um arco que conecta um nó i à um nó j, ambos pertencentes à N, é denotado arco (i, j). O grafo com arcos orientados é chamado de grafo orientado (ou direcionado).

O CUSTO de um arco, dependendo dos critérios utilizados, pode representar a distância de percurso, o tempo de viagem, o custo de combustível, a confiabilidade da rota, etc.

Um CAMINHO em um grafo é uma seqüência de arcos e nós adjacentes, respeitando a orientação dos arcos, se houver. Um CICLO é um caminho em que o nó inicial e o nó final são iguais.

O GRAU de um nó em um grafo não orientado é o número de arcos incidentes nele. Em um grafo orientado, o INDEGREE de um nó é o número de arcos adjacentes que são dirigidos à ele e o OUTDEGREE de um nó é o número de arcos adjacentes que partem dele.

Um ciclo EULERIANO é um ciclo que passa exatamente uma vez por cada arco do grafo. Um grafo é CONEXO quando existe um caminho que liga todos os pares de nós do grafo, sendo este caminho independente da orientação dos arcos (para grafo orientado). Um grafo não orientado conexo possui um ciclo Euleriano quando contém exatamente zero nós de grau ímpar. Um grafo orientado conexo possui um ciclo EULERIANO quando o indegree e o outdegree para cada nó possuem o mesmo valor .

PROBLEMAS DE ROTEAMENTO DE VEÍCULOS ESTUDADOS

O problema freqüentemente enfrentado pelos motoristas é a escolha da rota a ser percorrida. Sua escolha normalmente visa a economia e a confiabilidade da rota.

Um dos casos mais importantes em problemas de roteamento de veículos (PRV) é a procura do caminho mínimo, ou seja, o de menor custo.

Um outro problema interessante é o do Carteiro Chinês em grafos orientados, o qual consiste em percorrer todos os arcos do grafo respeitando a restrição do sentido (mão única) dos arcos, a um mínimo custo. A restrição do sentido (mão única) dificulta mais este problema de roteamento, mas é o caso mais aplicado na prática, pois nem sempre possuímos ruas com mão dupla.

Um outro problema interessante é o Problema das Medianas, o qual consiste em determinar boas localizações para a construção de facilidades ou de instalação de serviços públicos.

Apresentamos a seguir algoritmos sugeridos na literatura abordada para resolução destes problemas. Observe que nos casos de grafos orientados o custo do arco (i, j) é diferente do custo do arco (j, i).

ALGORITMO DE DIJKSTRA (Larson; Odoni, 1981)

O algoritmo de Dijkstra determina o caminho de menor custo entre um nó e os demais nós do grafo.

O primeiro pré-requisito para o funcionamento correto do algoritmo de Dijkstra é a suposição de que todos os arcos do grafo G(N, A) possuem custo não-negativo.

A resolução consiste em fazer uma busca, a partir de um nó fonte (inicial), do nó mais próximo, do segundo mais próximo e assim por diante, até que se complete a busca em todos os nós. A busca deve respeitar a orientação dos arcos, caso exista.

O método usa dois vetores, D[] e Cam[], em que D[j] é o custo entre o nó fonte e o nó j e Cam[j] é o nó predecessor ao nó j, mais próximo a j. Se D[j]=∞ ou Cam[j]=0 então não existe caminho entre o nó fonte e o nó j.

Seja C[][] a matriz de custos do grafo, min (a, b) uma função que retorna o menor caminho de a à b, *n* o número de nós do grafo, *fonte* o índice do nó fonte, V[] o conjunto dos nós do grafo, S[] o conjunto dos nós visitados e V-S os nós que existem em *V* e não existem em *S*.

1º Passo: Faz-se D[j]=C[*fonte*][j] e Cam[j]=*fonte* para cada nó do grafo;
Cam[*fonte*]=0;

2º Passo:

```
Para i=1 até i=n, faça:
    Se D[i]=∞ e i≠fonte:
        Cam[i]=0;
    Repita n vezes:
        Se D[a] for mínimo:  (a ∈ V-S)
            a é acrescentado em V[];
    Para cada nó b em V-S:
        D[b]=min(D[b], D[a]+C[a,b]);
    Se D[b]=D[a]+C[a,b]):
        Cam[b]=a;
i=i+1;
```

ALGORITMO DE FLOYD (Larson; Odoni, 1981)

O algoritmo de Floyd determina o caminho de custo mínimo entre todos os pares de nós do grafo.

O algoritmo consiste do seguinte: dada a matriz de custos, ela é atualizada *n* vezes, sendo *n* o número de nós do grafo G(N, A), buscando na iteração de número *k* os menores custos entre os pares de nós que passem pelo nó *k*.

O algoritmo usa duas matrizes D[][] e P[][], sendo que ao final do processo serão a matriz de menores custos e a de seqüência dos nós dos caminhos de menor custo.

Seja C[][] a matriz de custos do grafo, min (a, b) uma função que retorna o menor caminho de a à b, *n* o número de nós do grafo G(N, A).

1º Passo: Para i=1 até i=*n*:
- Para j=1 até j=*n*:
  - Se i=j:
    - D[i][j]=0;
    - P[i][j]=NULL;
  - Se não:
    - P[i][j]=i;
    - Se C[i][j] não existe:
      - D[i][j]=∞;
    - Se não:
      - Se D[i][j]= C[i][j];
  - j=j+1;
- i=i+1;

2º Passo: Para k=1 até k=*n-1*:
- Para i=1 até i=*n*:
  - Para j=1 até j=*n*:
    - D_k[i][j]=min(D_{k-1}[i][j], D_{k-1}[i][k]+D_{k-1}[k][j]);
    - Se D_k[i][j]≠D_{k-1}[i][j]:
      - P_k[i][j]=P_{k-1}[k][j];
    - Se não:
      - P_k[i][j]=P_{k-1}[i][j];
    - j=j+1;
  - i=i+1;
- k=k+1;

PROBLEMA DAS MEDIANAS

Considerando um grafo $G(N,A)$ não orientado com n nós, escolhemos k pontos distintos $(k = (1, 2, 3...))$ em G pertencentes ao conjunto $X = \{x_1, x_2, \ldots x_{k-1}, x_k\}$. A distância mínima entre quaisquer pontos $x_i \in X$ e o nó j em G será $d(i, j)$. O conjunto S indica o conjunto de nós das medianas localizadas durante a execução do algoritmo e m indica o número de nós do conjunto S.

Algoritmo (Larson; Odoni, 1981):

Passo 1: Achar a distância mínima entre todos os pares de nós do grafo G usando o algoritmo de Floyd e montar uma matriz de distâncias mínimas (com n linhas por n colunas);

Passo 2: Multiplicar a j-ésima coluna da matriz de distâncias mínimas pela demanda correspondente da carga, com $j = (1, 2, 3, \ldots, n)$, para obter uma outra matriz $[h_j * d(i,j)]$;

Passo 3: Para cada linha i da matriz $[h_j * d(i,j)]$, faça a soma de todos os elementos da linha. O nó correspondente a linha cuja soma de seus elementos for o menor valor obtido é a localização da 1-mediana;

Passo 4: Admite-se que $m = 1$ e que a 1-mediana está em i, então $S = \{i\}$;

Passo 5: Inclui-se uma nova facilidade no conjunto S, escolhendo a localização entre os nós de $N - S$, ou seja, os nós que não estão em S, no qual provém uma maior possibilidade de melhorar a função objetivo com o número das medianas incrementadas em 1. Faça-se $m \leftarrow m + 1$;

Passo 6: Na tentativa de melhorar a função objetivo com a substituição de um dos nós em S com um nó que está em $N - S$. A cada vez que uma solução melhorada for obtida, ela deve ser usada como uma nova solução do conjunto S, então repete-se este passo. Quando todas as possibilidades de substituição de nós do conjunto S forem tentadas sem melhorar a função objetiva, execute o Passo 7;

Passo 7: Se $m = k$, pare; caso contrário, retorne ao Passo 5.

CARTEIRO CHINÊS EM GRAFOS ORIENTADOS

O algoritmo para resolução do Problema do Carteiro Chinês em grafos orientados verifica se o número de arcos que entram e o número de arcos que saem de cada um dos nós do grafo são iguais, assim possibilitando a obtenção do Ciclo Euleriano.

Chama-se P_i a polaridade do nó i definido como o número de arcos que entram menos o número de arcos que saem do nó i. Quando P_i for maior que zero, dizemos que i é um nó de oferta, caso P_i for menor que zero, i será um nó de demanda. Os conjuntos do nós de oferta e de demanda são chamados de O e D, respectivamente.

Algoritmo (Larson; Odoni, 1981):

Passo 1: Verificar quais os nós de oferta e os de demanda, computando o P_i de cada um e as distâncias mínimas $d(i,j)$ de todos os nós $i \in O$ para todos os nós $j \in D$.

Passo 2: Faça combinações de nós de oferta com os de demanda, de modo a minimizar $\sum_{i \in O} \sum_{j \in D} d(i,j) x_{ij}$, com $\sum_{j \in D} x_{ij} = P_i$ para todo $i \in O$, $\sum_{i \in O} x_{ij} = -P_i$ para todo $j \in D$ e $x_{ij} \geq 0$.

Passo 3: Para cada $x_{ij} > 0$ encontrado, é adicionado ao grafo G, x_{ij} cópias dos caminhos mínimos de $i \in O$ à $j \in D$, gerando um grafo G' com todos os $P_i = 0$ para todos os nós.

Passo 4: Ache então o Ciclo Euleriano de G'. Este ciclo é a resolução do Problema do Carteiro Chinês para o grafo G.

IMPLEMENTAÇÃO

A implementação dos algoritmos ainda está em andamento, mas não é de toda complexa, com exceção do algoritmo do Matching Problem, algoritmo que obtém o conjunto de pares de nós de grau ímpar cuja soma de custos dos menores caminhos de cada par seja mínimo, que deverá ser adaptado para resolução do Carteiro Chinês em grafos orientados.

O código-fonte utiliza, para a matriz de custos dada, uma matriz unidimensional (vetor) ao invés de uma matriz bidimensional, isto evita possíveis problemas com utilização de ponteiros, além de facilitar na alocação dinâmica de memória e inicialização da mesma. Se exemplo de inicialização:

Para matriz (vetor bidimensional):

```
Para i=1 até i=n:        //n...número de nós
    Para j=1 até j=n:
        matcust[i][j]=custo do nó i ao nó j;   //matcust...matriz de custo
        j=j+1;
    i=i+1;
```

Para vetor unidimensional:

```
Para i=1 até i=n:
    Para j=1 até j=n:
        matcust[j+(i*n)]=custo do nó i ao nó j;
        j=j+1;
    i=i+1;
```

CONCLUSÃO

Os algoritmos estão em desenvolvimento, sendo testados e implementados em linguagem C++ orientado ao objeto no compilador Borland C++ 3.11, 1992.

A maior dificuldade encontrada está sendo na implementação em C++.

BIBLIOGRAFIA

Christofides, N. Graph Theory – An Algorithmic Approach, Editora Academic Press INC., 1975;

Jamsa, K.; Klander, L. Programando em C/C++, Editora Makron Books, 1999;

Larson, R. C.; Odoni, A. R. Urban Operations Research, Editora Prentice-Hall, 1981;

Sampaio, Rudini M. Estudo e Implementação de Algoritmos de Roteamento, Trabalho de Graduação CTA/ITA, 1998.