MS877: Projeto Supervisionado II

O problema do Caixeiro Viajante

Orientador: Prof. Dr. Aurélio Ribeiro Leite de Oliveira

Aluna: Marie Mezher Silva Pereira ra:096900

DMA - IMECC - UNICAMP

23 de junho de 2014

1 Introdução

O Problema do Caixeiro Viajante (PCV) ou TSP (Traveling Salesman Problem) é o problema de otimização combinatória mais conhecido e estudado na literatura, entre suas características estão o fato de ser facilmente explicado, como também o seu crescente número de novas aplicações, tornando intrigante sua resolução pela comunidade científica. Porém há grande dificuldade de obter-se uma solução exata. O PCV é um problema de otimização da classe de complexidade *NP* que corresponde à classe de problemas que não podem ser resolvidos por algoritmo polinomial.

Muitos dos primeiros trabalhos do Problema do Caixeiro Viajante, foram motivados por aplicações diretas em inúmeros problemas práticos. Embora tenham sido desenvolvidos bons algoritmos de aproximação para o PCV, o problema continua a oferecer uma grande atração para a aplicação de novos algoritmos.

2 Problema do caixeiro viajante

Para definir o PCV usaremos o conceito de grafo e ciclo hamiltoniano. Segue a breve explicação destes:

Um grafo é formado por dois conjuntos, um conjunto de vértices (ou nós) e um conjunto de arcos, cada arco está associado a dois vértices, o primeiro é a ponta inicial do arco e o segundo é a ponta final.

Um caminho hamiltoniano num grafo é um ciclo que contém todos os vértices do grafo exatamente uma vez. Segue-se um exemplo de grafo com caminho hamiltoniano.

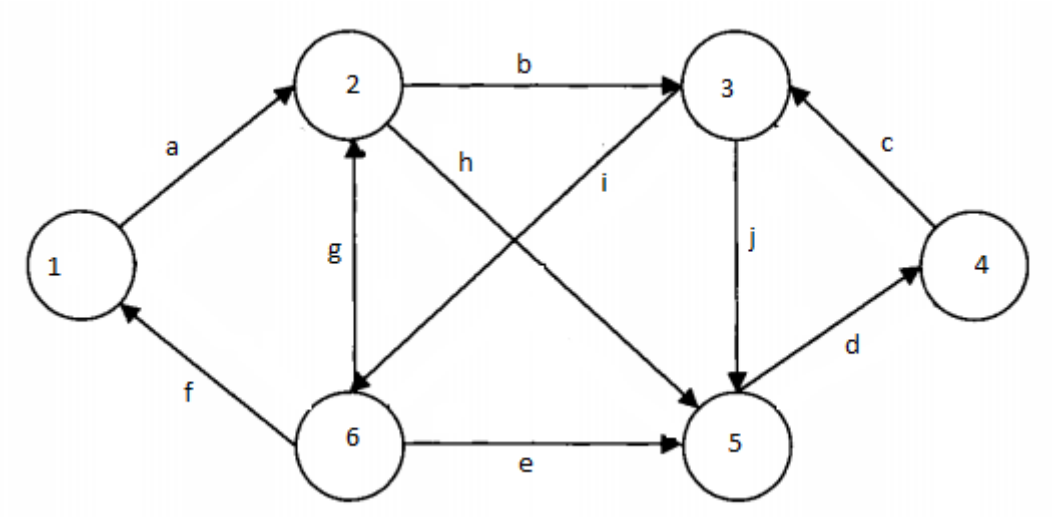

Figura 1: $1, 2, 3, 4$, e 6 representam os nós e, $a, b, c, d, e, f, g, h, i$ e j representam os arcos do ciclo Hamiltoniano.

2.1 O problema

No TSP o caixeiro deve visitar cada cidade exatamente uma vez e retornar ao ponto de partida, ou seja, dado um grafo e um nó inicial, deve-se encontrar um ciclo partindo deste nó que passa uma única vez por cada um dos outros nós voltando ao inicial. Definir a ordem de visita para fazer um *tour* o mais rápido possível e, com custo mínimo, pois os arcos contém custos.

O problema pode ser simétrico, ou seja, em cada arco é possível caminhar nas duas direções com o mesmo custo. No problema assimétrico os custos em direções contrárias podem ser diferentes ou uma das direções não é permitida.

O problema do caixeiro viajante consiste em encontrar o ciclo hamiltoniano de menor custo.

2.2 Formulação Matemática

- Definições das variáveis:

  A variável x_{ij} é binária e representa o arco do nó i até o nó j :

  - $x_{ij} = \begin{cases} 1, & \text{se o carteiro viaja do nó } i \text{ até o nó } j; \\ 0, & \text{caso contrário;} \end{cases}$
  - c_{ij} representa o custo do arco do nó i até o nó j.

- Definição da função objetivo:

  Devemos minimizar o problema, ou seja, minimizar o custo c_{ij} pela variável x_{ij} :

$$Min \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij};$$

- Definição das restrições:

  O caixeiro deixa a cidade i exatamente uma vez:

$$\sum_{j=1, j \neq i}^{n} x_{ij} = 1 \text{ para } i = 1, \ldots, n.$$

  O caixeiro chega na cidade j exatamente uma vez:

$$\sum_{i=1, i \neq j}^{n} x_{ij} = 1 \text{ para } j = 1, \ldots, n.$$

  Logo, o problema é definido como:

$$\begin{aligned} Min \quad & \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij}; \\ sa \quad & \sum_{j=1, i \neq j} x_{ij} = 1 \text{ para } i = 1, \ldots, n. \\ & \sum_{i=1, i \neq j} x_{ij} = 1 \text{ para } j = 1, \ldots, n. \\ & x_{ij} \in (0, 1) \text{ para } i = 1, \ldots, n \; j = 1, \ldots, n. \end{aligned}$$

Essa é a formulação do problema de designação. Esta solução não impede soluções com mais de um ciclo, como no exemplo mostrado na figura abaixo:

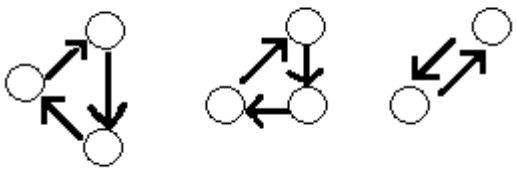

Figura 2: *Subtours.*

Para eliminar soluções com *subtours* devemos garantir conectividade entre os nós, ou seja, impor que o caixeiro deva passar de um conjunto de cidades para o outro. Assim por eliminação de *subtours* temos a seguinte restrição:

$$\sum_{i \in S} \sum_{j \in S} x_{ij} \leq |S| - 1;$$

Escolhendo $|S|$ nós de uma solução, onde $S \subset N$ e $2 \leq |S| \leq n - 1$:

3 Resolução

O TSP é um problema combinatorial no sentido que a solução ótima é um subconjunto de um conjunto de soluções finitas. Então em princípio este problema pode ser resolvido com enumeração. Para obter o tamanho do problema que estamos tentando resolver devemos contar as possíveis soluções.

Logo, começando da cidade 1, o caixeiro tem $n-1$ escolhas para fazer. Na próxima escolha $n-2$ e assim por diante. Então existem $(n-1)!$ prováveis *tours*.

Na tabela 1 a seguir, temos o quão rapidamente uma função cresce. Logo o TSP com $n = 101$ tem aproximadamente 9.33×10^{157} *tours*. Dados obtidos em [2].

n	$\log n$	$n^{0.5}$	n^2	2^n	$n!$
10	3.32	3.16	10^2	1.02×10^3	3.6×10^6
100	6.64	10.00	10^4	1.27×10^{30}	9.33×10^{157}
1000	9.97	31.62	10^6	1.07×10^{301}	4.02×10^{2567}

Tabela 1: Algumas funções típicas

A conclusão que temos é que usando enumeração esperamos conseguir resolver problemas com valores muito pequenos para n. A seguir apresento uma opção mais eficiente para resolver o PCV.

3.1 Heurísticas

Em problemas de programação inteira, Heurísticas são algoritmos aproximados que nos fornecem rapidamente boas soluções, entretanto a solução pode não ser ótima. Heurísticas

podem ser utilizadas para encontrar soluções aproximadas e assim acelerar os métodos exatos (ex. *branch and bound, branch and cut*).

Exemplos de heurísticas no Problema do Caixeiro Viajante:

- Partindo da cidade do nó inicial selecione a aresta de menor custo entre o nó atual e nós ainda não visitados e adicione na rota a ser percorrida. Ao final a rota será uma solução local.

- Ordena as arestas pelo custo e adiciona as arestas nesta ordem sem formar *subtours* Quando formar uma rota temos uma solução.

Usando heurísticas a rota encontrada como solução do PCV pode não ser ótima ou até estar bem longe do ponto ótimo, mas, podemos enunciar três principais motivos, segundo [1](em livre tradução), da utilização deste método. São eles:

1. Boas heurísticas estão disponíveis para muitos problemas inteiros e de otimização combinatória devido à sua estrutura.

2. Resolver problemas reais de programação inteira por outras abordagens geram soluções muito lentamente e na realidade são necessários soluções em segundos ou minutos, não horas ou mesmo dias.

3. A formulação de problemas de programação inteira por outros métodos tais como *branch-and-bound*, *branch-and-cut* entre outros são muito difíceis.

Os métodos heurísticos são procedimentos bastante utilizados na realidade e tendem a encontrar rapidamente uma solução local.

4 Aplicações reais

O problema do Caixeiro Viajante tem aplicações em diversas áreas tais como: problemas de transporte (entrega e coleta de cargas, transporte de funcionários, ônibus escolar, roteamento de veículos), linha de produção, problemas de manufatura (programação de operações em máquinas, programação de transporte entre células de manufatura, otimização do movimento de ferramentas de corte). Novas aplicações poder ser formuladas utilizando-se múltiplos caixeiros viajantes. Segue-se uma explicação da aplicação em sequeciamento de genoma.

4.1 Estudos genéticos em sequenciamento de genoma

Sendo os estudos de sequenciamento de genoma, tema de grandes proporções, como por exemplo, um ser humano tem 23 cromossomos e cada um deles deve ser mapeado. Leonard Adleman [3] em 1994, percebeu a similaridade entre o DNA e os computadores e como o armazenamento de informações no DNA (bases: Adenina, Timina, Guanina, Citosina) é semelhante aos computadores (base binária 0 e 1). E desenvolveu um protótipo de computador genético para resolver problemas matemáticos como o PCV.

Em estudos genéticos o PCV prevê uma maneira de integrar mapas locais do genoma em um único mapa. Assim de maneira geral, as cidades são os mapas locais e o custo de viajar de um nó a outro é o peso que um mapa local segue imediatamente outro mapa local.

Para representar cada cidade do PCV Adleman [3], associou bijetoramente as cidades envolvidas à sequências com número fixo de nucleotídeos (blocos construtores do DNA).

Para o caminho entre cada cidade, Adleman usou técnicas laboratoriais para fazer o acoplamento da segunda metade da sequência de nucleotídeos do nó (ou cidade) 1 com a primeira metade da sequência do nó 2 e então fabricar a sequência complementar desse acoplamento, ou seja, o arco entre o nó 1 e o nó 2.

Conseguida a representação genética dos dados do PCV, os procedimentos genéticos utilizados de modo a obter ciclos passando por todos os nós foram a ligação das sequências hibridizantes e para que o ciclo comece no nó inicial e terminasse no nó final como também passasse obrigatoriamente uma vez em cada nó, foi utilizado a separação das rotas legítimas. Adleman precisou usar dezenas de procedimentos laboratoriais que levaram 1 semana, para fazer com que a informação associada ao problema fosse processada geneticamente e assim poder determinar o ciclo hamiltoniano de menor custo.

5 Conclusão

Neste projeto foi apresentado o problema do caixeiro viajante, sua definição e complexidade tal como uma abordagem a sua resolução por um método não exato. Foi detalhado um exemplo do PCV em aplicação real, sendo este o estudo de sequenciamento de genoma.

É fascinante o quão fácil o problema é descrito e quantas generalizações deste estão disponíveis na literatura. Além da grande dificuldade de solução exata, é importante lembrar que no PCV quanto maior a quantidade de vértices e arestas o grafo possuir, o grau de complexidade e tempo de resolução do problema aumentam. Tendo em vista as inúmeras aplicações na realidade e a quantidade de problemas que podem ser modelados de forma similar ao PCV, a obtenção de melhores resultados no seu algoritmo ocasiona uma melhoria em processos executados diariamente em casos reais.

Enfim, o problema do caixeiro viajante ainda está em aberto, e há muito campo à ser explorado na obtenção de soluções ótimas, tanto em métodos exatos, no melhoramento de soluções heurísticos e na combinação com outros métodos.

Referências

[1] *Applied Integer Programming* D.-S. Chen, R. G. Batson and D. Yu, John Wiley and Sons, 2010.

[2] *Integer Programming* L. A. Wolsey, John Wiley, 1998.

[3] *Molecular computation of solutions to combinatorial problems* Leonard Adleman. Science 266, 1021-1024 (1994).