Prática Investigativa – Resenha sobre artigo:

The Traveling Salesman Problem: Na overview of exact and approximate algorithms
(Gilbert Laporte, 1991)

Aluno : Ian Rodrigues dos Reis Paixão
Professora: Raquel Aparecida de Freitas Mini
Disciplina: Projeto e análise de algoritmos
Data: 19/05/2019

1 - Resumo do artigo

1.1 – Introdução e contextualização

O artigo de Laporte faz um *survey* (levantamento bibliográfico) dos principais algoritmos e heurísticas propostos para a resolução do problema do caixeiro-viajante (PCV) até o ano de sua publicação (1991).

O problema em questão é: dado um conjunto de cidades *V* e um conjunto de rotas entre as cidades *E*, devemos encontrar um circuito de tal forma que todas as cidades tenham sido visitadas exatamente uma vez e o custo para tal (distância percorrida através das rotas) seja o menor possível naquele contexto. Podemos interpretar esse problema utilizando a teoria de Grafos, no qual *V* seria o conjunto de vértices, *E* o conjunto de arestas e desejássemos encontrar um caminho Hamiltoniano nesse grafo. Assim como foi feito no artigo, chamaremos esse ciclo de *tour*.

Utilizando essa definição clássica do problema, podemos ver claramente uma aplicação prática do mesmo: calcular o menor custo possível para uma empresa de transporte/entrega que precisa passar por *V* locais. Entretanto, além dessa aplicação, podemos enxergar o PCV em outros casos, como por exemplo:

I. Filamento de computador: utilizar a menor quantidade de fios possível na construção de um computador, sendo que todos os módulos devem estar conectados.
II. Corte de papel de parede: determinar de que forma um papel de parede deve ser cortado de tal forma que se mantenha o padrão desejado e tenha o menor desperdício possível.
III. Fazendo buracos: em um contexto de manufatura, devemos produzir determinado número de buracos em uma peça de metal, por exemplo. Devemos movimentar a

máquina que fará esses buracos a menor quantidade de vezes possível.

IV. Sequenciamento de rotinas: dado que uma máquina deve realizar determinado número de rotinas, sendo que existe um tempo de "troca de contexto" entre duas rotinas (quando finaliza uma e vai começar outra), devemos minimizar esse tempo ocioso (de certa forma, o item III é uma aplicação prática desse).

V. Design de tabuleiro de dardos: criar um tabuleiro de dardos (o alvo em si) de tal forma que maximize o risco para os jogadores.

VI. Cristalografia: Em alguns experimentos de cristalografia, é necessário posicionar os cristais em diferentes aparelhos, cada um fazendo uma medida. A ordem em que serão feitas essas medidas pode ser visto como um PCV, para diminuir o tempo dos testes.

O PCV pode ser considerado como da classe NP-difícil, mesmo tendo sido encontradas soluções polinomiais para algumas instâncias.

1.2 – Algoritmos exatos

Muitos dos algoritmos propostos podem ser melhor entendidos e explicados do contexto de programação linear inteira (PLI). É feita uma análise de alguns desses algoritmos, bem como derivações desses, que serão explicados a seguir.

1. *Formulação PLI:*

   Uma das primeiras formulações desse tipo de problema foi feita por Dantzig, Fulterson e Jonson (DFJ, 1954). Nessa formulação o objetivo é descrever o custo de uma *tour* ótima. Para isso, ela possui insere algumas restrições:

entramos e saímos de cada vértice exatamente uma vez; proibição da criação de *subtours* (*tours* compostas de menos vértices que o total *V*). Entretanto, o custo para essa última é exponencial (2^n). Dessa forma, mesmo para valores moderados de n, é irreal resolvê-lo utilizando PLI. Miller, Tucker e Zemlin (MLZ, 1960) propuseram restrições que diminuem o custo para eliminar as *subtours*, ao custo da utilização de mais variáveis (maior utilização de memória). Mesmo o MLZ sendo mais compacto, o DFJ ainda se mostra superior. Mesmo tendo sido feitas várias derivações e comparações, o DJF ainda se mostrou superior na resolução do problema.

2. Atribuição de limite inferiores e algoritmos de *branch and bound* relacionados:

    Os algoritmos de *branch and bound* (BB) conseguem checar se um determinado caminho para encontrar a solução já deixou de ser válido antes de terminar sua execução. Por exemplo, no contexto de PCV, se definirmos um limite de peso 10, e a rota atual está com peso 11, não precisamos executá-la até o final para identificar que é inválida.

    A qualidade de um algoritmo BB é definida pela qualidade de seu limite, que pode ser calculado através de PLI ou de um problema de atribuição (PA). Vários algoritmos foram criados usando esses métodos, mas o autor, a princípio, foca no de Carpaneto e Toth (CT, 1980). Esse algoritmo segue a seguinte ideia: pegamos a melhor resolução daquele problema, através do uso de uma heurística. Utilizando cada combinação possível de vértices, calculamos aquele que produzem um resultado menor que o limite definido. Depois, para cada caminho, checamos se eles são válidos (se não possuem *subtours*, por exemplo). Esse algoritmo teve um tempo de execução razoável. Seu limitador é mais espaço de memória do que tempo de execução (devido à grande quantidade de caminhos e filas que ele deve armazenar). Balas e Christofides (BC, 1981) seguem a mesma linha de raciocínio, utilizando um algoritmo mais complexo. O próprio autor do *survey* não se aprofunda muito, devido à essa complexidade. Esse último algoritmo conseguiu ser aproximadamente 33% mais eficaz (utilizando a mesma medida de tempo, conseguiu executar entradas de tamanho 33% maior) do que o de CT. É citado que Miller e Pekny (MP, 1989) descreveram um algoritmo de BB paralelizado que conseguiu ser quase dez vezes mais eficaz do que o de BC.

3. *A menor arborescência geradora limitada e algoritmos relacionados.*

Em um grafo direcionado, uma arborescência geradora mínima é aquela que todos os vértices são conectados tendo apenas grau 1 de entrada e qualquer um pode ser acessado a partir da raiz. O problema de determinar esse custo se reduz a dois subproblemas independentes: determinar o custo mínimo de uma árvore de raiz *r* (resolvido facilmente com complexidade n^2 (Tarjan, 1977)) e encontrar o caminho de menor custo a partir do vértice *r*. Fischetti e Toth (FT, 1991) combinaram um de seus algoritmos com o de CT e conseguiram vários resultados eficazes.

4. *A menor árvore geradora limitada e algoritmos relacionados.*

   Os algoritmos da seção anterior baseados em PA tratam de grafos assimétricos e simétricos. Entretanto, no caso de simétricos, muitas vezes são geradas *subtours* quem precisam de um tempo a mais para serem tratadas. Então é nos dado uma formulação de autoria desconhecida que tratam algumas restrições para esse caso, como especificar que cada vértice tem um grau igual a 2, eliminação de *subtours* e imposição de condições binárias nas variáveis. Uma dessas limitações pode ser aplicada ao algoritmo DFJ. Hald e Karp (HK, 1971) propuseram um algoritmo para encontrar a largura de uma árvore geradora mínima, que seria utilizada como um bom valor de limite para os demais algoritmos (como BB) que tentavam resolver o PCV. Com ele, foram capazes de resolver muitos problemas clássicos com poucas ramificações (*branches*). Várias variações desse algoritmo foram implementadas ao longo do tempo, por vários pesquisadores, inclusive variando a árvore gerado mínima criada.

1.2 – Algoritmos exatos

Como o PCV é da classe NP-difícil, e como foi comentado várias vezes na seção anterior, é normal a criação de Heurísticas para a resolução completa, ou parcial, do problema. Heurísticas são resultados aproximados do melhor valor possível. Podem não ter uma precisão total, mas muitas vezes conseguem ser muito mais velozes que a resolução completa.

1. *Heurísticas com desempenho de pior caso garantido*

   Considere um PCV em um grafo simétrico, onde *C* satisfaz a desigualdade triangular. Como vimos na seção anterior, a complexidade para encontrarmos um limite para o melhor resultado possível por ser feito em $O(n^2)$. Uma possível estratégia para visitar todos os vértices é atravessar a árvore geradora por suas arestas, da seguinte maneira: considerar uma folha (vértice de grau 1) da árvore geradora. Se tiver uma aresta não atravessada nesse vértice, pegamos agora o outro vértice em que essa aresta se liga e repetimos esse passo. Quando não for mais possível, vamos de volta ao vértice k onde i foi primeiramente alcançado. Fazemos isso até k for igual ao primeiro vértice que usamos no primeiro passo. Vários autores foram incrementando alterações nessa heurística, de forma a deixá-la mais eficiente, como por exemplo a utilização de "atalhos" com as arestas. O autor menciona que nenhuma heurística dessa categoria garante desempenho de pior caso para grafos assimétricos de um PCV.

2. *Heurísticas com bom desempenho empírico*

   Nesse tópico, salvo algumas exceções, os algoritmos descritos podem ser usados tanto em grafos simétricos quanto assimétricos. Primeiramente o foco será o procedimento de construção de *tour.*

   a. *Algoritmo de vizinho mais próximo*: Documentado por Rosenkrantz, Stearns e Lewis (RSL, 1977). Escolhemos um vértice aleatório como ponto de partida. Determinamos o vértice mais próximo e o inserimos na *tour*, usando ele de referência agora. Repetimos esse passo até o momento em que não sobre mais vértices de fora. Ligamos o último vértice ao primeiro. A complexidade dessa algoritmo é da ordem de $O(n^2)$.

b. *Algoritmos de inserção*: Documentado por RSL, Stewart (1977), Norback e Love(NL, 1977). Essa categoria inclui uma série de algoritmos que pode ser resumidas nos passos citados a seguir: construímos uma primeira *tour* consistindo de dois vértices. Então consideramos todos os vértices que ainda não estão nela. A partir de um determinado critério (pode ser o vértice mais próximo ou mais distante da *tour*, por exemplo) inserimos os vértices na *tour*. Dependendo do critério, temos um ordem de complexidade $O(n^2)$ ou $O(n \log n)$

c. *Algoritmo remendado para PsCV assimétricos.* Documentado por Karp(1979). Primeiro resolvemos o PA com custo matriz C. Se a solução possuir somente um circuito, paramos. Caso contrário selecionamos os dois circuitos com maior quantidade de vértices. Selecionamos uma aresta de cada um de tal forma que conseguimos reduzir o custo total, unindo as duas (*merge*). Voltamos para o passo de checar se a solução possui apenas um circuito.

3. *Procedimentos de melhoria de tour*

   Utilizados para melhorar uma tour encontrada através de qualquer meio

   a. *Algoritmo r-ótimo.* Documentado por Lin (1965). Pegamos uma *tour* inicial. Removemos *r* arcos da *tour* e tentamos reconectar as correntes restantes de todos os modos possíveis. Se obtivermos uma nova *tour*, a utilizamos como a nova *tour* inicial. Repetimos o preços até não conseguirmos obter mais melhorias. Melhorias em cima desse algoritmo foram propostas, a mais relevante sendo a de Or (1976).

b. *Anelamento Simulado.* Documentado por Kirkpatrick (1983) Esse processo é derivado de uma analogia usada com anelamento de materiais, na área de Mecânica. Para trazer um material a um estado sólido de mínima energia, é necessário aquecê-lo até que suas partículas estejam aleatoriamente distribuídas no estado líquido. Então sua temperatura é reduzida gradativamente em vários passos, até que que o sistema chegue no equilíbrio para um determinado nível de temperatura. Em uma alta temperatura, todos os estados possíveis podem ser alcançados. Entretanto, à medida em que o sistema resfria, o número de possibilidades vai reduzindo e o processo converge para um estado estabilizado.

c. *Busca Tabu.* Documentado por Glover (1977) junto com McMiillan(1986). Nos dois métodos anteriores, sucessivas vizinhanças de uma solução *x* são examinadas. Para a prevenção de ciclos soluções que já foram examinadas e proibidas são inseridas em uma “lista de tabu”, que é constantemente atualizada. O sucesso desse método depende da cuidadosa escolha do número de parâmetros de controle.

4. *Algoritmos compostos.*

a. *Algoritmo CCAO*. Documentado por Golden e Stewart (GS, 1985). Essa heurística foi proposta para P'sCV Euclidianos simétricos. Ela explora uma propriedade bem conhecida desses problemas: e m uma solução ótima, vértices localizados na envoltória convexa de todos os vértices são visitados na ordem em que aparecem na envoltória em questão.
b. *Algoritmo GENIUS.* Documentado por Gendreau, Hertz e Laporte(1992). O algoritmo GENIUS combate um dos principais problemas do CCAO em duas fases: a fase da inserção geral seguida de uma fase de pós-otimização que remove vértices da *tour* e os reinserem sucessivamente, utilizando a regra a inserção geral. O algoritmo se comparou melhor que o CCAO, que a lista de tabu e que o anelamento simulado, apesar da quantidade de comparações ser mais limitada no casos desses últimos dois

2 – Crítica sobre o artigo

2.1 – Pontos positivos:

O artigo segue uma ordem lógica bem coerente: introduz o assunto, explica o problema, contextualiza e dá exemplos de aplicações no mundo real. Possui uma boa quantidade de referências, todas inseridas na bibliografia.

2.2 – Pontos negativos:

Algumas partes do artigo ficaram bem difíceis de entender devida a complexidade do problema em si, pois o autor escreve bem, como dito acima, e falta de imagens dificulta ainda mais o entendimento.