**O Problema do caixeiro viajante através de algoritmo genético e do algoritmo do Otimização da Colônia de Formigas.**

**Resumo**: A meta-heurística trata da aplicação de métodos para problemas que não apresentam algoritmo definido com o objetivo de encontrar soluções muito próximas da ótima. Tal abordagem resulta em soluções efetivas com reduzido custo computacional, maior velocidade e de maneira direta. Neste sentido, o algoritmo genético e o algoritmo da otimização da colônia de formigas são exemplos desses métodos. Dessa maneira, este artigo aborda a solução do problema do caixeiro viajante, mostrando como descobrir o menor caminho entre um conjunto de vértices finitos utilizando estes dois algoritmos.

**Abstract:** The meta-heuristic deals with the application of methods for problems that do not have a defined algorithm in order to find solutions very close to the optimum. Such an approach results in effective solutions with reduced computational cost, greater speed and in a direct way. In this sense, the genetic algorithm and the ant colony optimization algorithm are examples of these methods. Thus, this article approaches the solution of the traveling salesman problem, showing how to find the shortest path between a set of finite vertices using these two algorithms.

1. **Definindo o Problema do Caixeiro Viajante**

Em meados da década de 1930, a ciência da computação ainda não era uma disciplina acadêmica bem definida. Na verdade, conceitos fundamentais, como "algoritmo" ou "problema computacional", foram formalizados apenas alguns anos antes.

Nesses anos, o matemático austríaco Karl Menger convidou a comunidade de pesquisa a considerar, de um ponto de vista matemático, o seguinte problema, tirado da vida cotidiana. Um caixeiro viajante deve visitar exatamente uma vez cada uma de uma lista de m cidades e depois retornar à cidade de origem. Ele sabe quanto custa viajar de qualquer cidade i para qualquer outra cidade j. Portanto, qual é o passeio de menor custo possível que o vendedor pode realizar?

O problema do caixeiro viajante (para abreviar, PCV) nasceu.

Mais formalmente, uma instância de PCV é dada por um grafo completo G em um conjunto de nós V = {1,2, ... m}, para algum inteiro m, e por uma função de custo atribuindo um custo cij ao arco (i, j), para qualquer i, j em V.

O PCV é um representante de uma grande classe de problemas conhecidos como problemas de otimização combinatória. Entre eles, o PCV é um dos mais importantes, pois é muito fácil de descrever, mas muito difícil de resolver.

Na verdade, o PCV pertence à classe NP-Completo. Consequentemente, um algoritmo eficiente para PCV (isto é, um algoritmo aplicado através de computação, para qualquer instância de PCV com m nós, o passeio de menor custo possível em tempo polinomial em relação a m) provavelmente não existe. Mais precisamente, tal algoritmo existe se e somente se as duas classes computacionais P e NP coincidirem, uma hipótese muito improvável, de acordo com o desenvolvimento da pesquisa nos últimos anos.

Do ponto de vista prático, isso significa que é quase impossível encontrar um algoritmo exato para qualquer instância de PCV com m nós, para elevados valores de m, que tenha um comportamento consideravelmente melhor do que o algoritmo que calcula qualquer um dos (m-1)! possíveis passeios distintos e, em seguida, retorna o menos caro.

Se estivermos procurando por aplicativos, uma abordagem diferente pode ser usada. Dada uma instância de PCV com m nós, qualquer passeio que passe uma vez por qualquer cidade é uma solução viável, e seu custo leva a um limite superior para o menor custo possível. Algoritmos que constroem em tempo polinomial com respeito a m soluções viáveis ​​e, portanto, limites superiores para o valor ótimo, são chamados de heurísticas. Em geral, esses algoritmos produzem soluções, mas sem nenhuma garantia de qualidade de quanto está seu custo do menor possível. Se puder ser mostrado que o custo da solução retornada é sempre menor que k vezes o menor custo possível, para algum número real k> 1, a heurística é chamada de algoritmo de k-aproximação.

Infelizmente, o algoritmo de aproximação de k para PCV não é conhecido, para qualquer k> 1. Além disso, em um artigo publicado em 2000, Papadimitriou e Vempala mostraram que um algoritmo de aproximação k para PCV para qualquer 97/96> k> 1 existe se e somente se P = NP. Portanto, também encontrar uma boa heurística para o PCV parece muito difícil.

Melhores resultados são conhecidos para o subproblema NP-Completo do PCV. Por exemplo, um algoritmo de aproximação 3/2 é conhecido para Metric PCV (em uma instância de métrica PCV, a função de custo verifica a desigualdade triangular).

De qualquer forma, a extrema intratabilidade do PCV convidou muitos pesquisadores a testar uma nova técnica heurística neste problema. Quanto mais difícil é o problema que você testa, mais significativo é o resultado obtido.

Uma grande parte deste artigo é dedicada a algumas técnicas heurísticas inspiradas na biologia que foram desenvolvidas nos últimos anos. Essas técnicas se inspiram na natureza. Na verdade, os animais que costumam formar grandes grupos se comportam por instinto tentando satisfazer a necessidade do grupo da melhor maneira possível. Da mesma forma, os sistemas naturais se desenvolvem a fim de (localmente) minimizar seu potencial encontrando um ponto estacionário.

1. **Introdução**

O Problema do Caixeiro Viajante ou PCV é um representante de uma grande classe de problemas conhecidos como problemas de otimização combinatória. Na forma normal do PCV, um mapa das cidades é dado ao vendedor e ele deve visitar todas as cidades apenas uma vez para completar um passeio de forma que a duração do passeio seja o mais curto entre todos os passeios possíveis para este mapa. Os dados consistem em pesos atribuídos às arestas de um grafo completo finito, e o objetivo é encontrar um ciclo hamiltoniano, um ciclo que passa por todos os vértices, do gráfico com o peso total mínimo. No contexto do PCV, os ciclos hamiltonianos são comumente chamados de passeios. Por exemplo, dado o mapa mostrado na figura l, a rota de menor custo seria aquela escrita (A, B, C, E, D, A), com o custo 31.

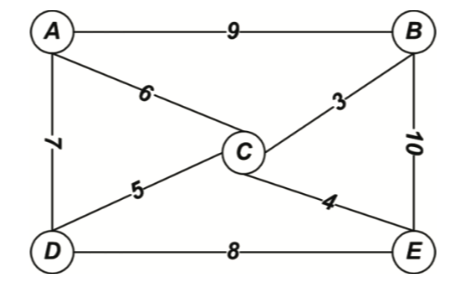


Figura 1: viagem através dos pontos A->B->C->E->D é a viagem ótima.

Em geral, o PCV inclui dois tipos diferentes, o PCV simétrico e o PCV assimétrico. Na forma simétrica conhecida como PCVS, existe apenas um caminho entre duas cidades adjacentes, ou seja, a distância entre as cidades A e B é igual à distância entre as cidades B e A (Fig. 1). Mas no PCVA (PCV assimétrico) não existe essa simetria e é possível ter dois custos ou distâncias diferentes entre duas cidades. Portanto, o número de passeios no PCVA e PCVS em n vértices (cidades) é (n-1)! e (n-1)! / 2, respectivamente. Observe que os gráficos que representam esses PCVs são gráficos completos. Neste capítulo, consideramos principalmente o PCVS. Sabe-se que o PCV é um problema NP-difícil (Garey & Johnson, 1979) e é frequentemente usado para testar algoritmos de otimização. Encontrar ciclos hamiltonianos ou rotas de viagens para o vendedor são possíveis usando um programa dinâmico simples usando tempo e espaço O (2n nO (1)), que encontra caminhos hamiltonianos com pontos finais especificados para cada subgrafo induzido do grafo de entrada (Eppstein, 2007). O PCV tem muitas aplicações em diferentes problemas de engenharia e otimização. O PCV é um problema útil em problemas de roteamento, por exemplo, em um sistema de transporte.

Existem diferentes abordagens para resolver o PCV. Resolver o PCV foi um problema interessante durante as últimas décadas. Quase todas as novas abordagens para resolver problemas de engenharia e otimização foram testadas no PCV como uma bancada de teste geral. Os primeiros passos para resolver o PCV foram métodos clássicos. Esses métodos consistem em métodos heurísticos e exatos. Métodos heurísticos, como planos de corte e branch and bound (Padherg & Rinaldi, 1987), só podem resolver pequenos problemas de forma otimizada, enquanto os métodos heurísticos, como 2-opt (Lin & Kernighan, 1973), 3-opt, cadeia de Markov (Martin et al., 1991), simulated annealing (Kirkpatrick et al., 1983) e tabu search são bons para grandes problemas. Além disso, alguns algoritmos baseados em princípios gananciosos, como o vizinho mais próximo e a árvore geradora, podem ser introduzidos como métodos de solução eficientes. No entanto, os métodos clássicos para resolver o PCV geralmente resultam em complexidades computacionais exponenciais. Portanto, novos métodos são necessários para superar essa lacuna. Esses métodos incluem diferentes tipos de técnicas de otimização, algoritmos de otimização baseados na natureza, algoritmos de otimização baseados em população e etc. neste artigo, discutiremos algumas dessas técnicas que são algoritmos baseados em população.

Os algoritmos de otimização baseados na população são as técnicas que fazem parte do conjunto dos algoritmos de otimização baseados na natureza. As criaturas e sistemas naturais que estão trabalhando e se desenvolvendo na natureza são uma das fontes interessantes e valiosas de inspiração para projetar e inventar novos sistemas e algoritmos em diferentes campos da ciência e da tecnologia. Evolutionary Computation (Eiben & Smith, 2003), Neural Networks (Haykin, 99), Time Adaptive Self-Organizing Maps (Shah-Hosseini, 2006), Ant Systems (Dorigo & Stutzle, 2004), Particle Swarm Optimization (Eberhart & Kennedy, 1995), Simulated Annealing (Kirkpatrik, 1984), Bee Colony Optimization (Teodorovic et al., 2006) e DNA Computing (Adleman, 1994) estão entre as técnicas de resolução de problemas inspiradas na observação da natureza.

Neste artigo algoritmos de otimização baseados em população foram introduzidos. Alguns desses algoritmos foram mencionados acima. Outros algoritmos são o algoritmo Intelligent Water Drops (IWD) (Shah-Hosseini, 2007), Artificial Immune Systems (AIS) (Dasgupta, 1999) e Electromagnetism-like Mechanisms (EM) (Birbil & Fang, 2003). Neste artigo, cada seção apresenta brevemente um desses algoritmos de otimização baseados em população e os aplica para resolver o PCV. Além disso, tentamos observar os pontos importantes de cada algoritmo e cada ponto que contribuímos para esses algoritmos foi declarado.

1. **Algoritmos evolutivos**

**3.1 Introdução**

Algoritmos evolutivos (AEs) imita o processo de evolução biológica na natureza. Esses são métodos de busca que se inspiram na seleção natural e na sobrevivência dos mais aptos que existem no mundo biológico. O AE realiza uma pesquisa usando uma população de soluções. Cada iteração de um AE envolve uma seleção competitiva entre todas as soluções na população que resulta na sobrevivência dos mais aptos e na eliminação das soluções pobres da população. Ao trocar partes de uma solução por outra, a recombinação é realizada e forma a nova solução que pode ser melhor do que as anteriores. Além disso, uma solução pode sofrer mutação ao manipular uma parte dela. Recombinação e mutação são usadas para evoluir a população para regiões do espaço onde podem residir boas soluções.

Quatro grandes paradigmas de algoritmo evolutivo foram introduzidos durante os últimos 50 anos: o algoritmo genético é um método computacional, proposto principalmente por Holland (Holland, 1975). Estratégias evolutivas desenvolvidas por Rechenberg (Rechenberg, 1965) e Schwefel (Schwefel, 1981). Programação evolutiva introduzida por Fogel (Fogel et al., 1966) e, finalmente, podemos citar a programação genética proposta por Koza (Koza, 1992). Aqui, apresentamos o AG (Algoritmo Genético) para resolver o PCV. No primeiro, preparamos um breve histórico sobre o AG.

* 1. **Algoritmos genéticos**

Algoritmos Genéticos focam na otimização de problemas combinatórios gerais. Os AGs são estudados há muito tempo como ferramentas de solução de problemas para muitos problemas de busca e otimização, especificamente aqueles que são inerentes aos problemas NP-Completos. Várias soluções candidatas são consideradas durante o procedimento de pesquisa no sistema, e a população evolui até que uma solução candidata satisfaça os critérios predefinidos. Na maioria dos AGs, uma solução candidata, chamada de indivíduo, é representada por uma sequência binária (Goldberg, 1989), ou seja, uma sequência de 0 ou 1 elemento. Cada solução (individual) é representada como uma sequência (cromossomo) de elementos (genes) e é atribuído um valor de aptidão com base no valor fornecido por uma função de avaliação. O valor de aptidão mede o quão perto o indivíduo está da solução ideal. Um conjunto de indivíduos constitui uma população que evolui de uma geração para a seguinte por meio da criação de novos indivíduos e exclusão de alguns antigos. O processo começa com uma população inicial criada de alguma forma, por exemplo, por meio de um processo aleatório. A evolução pode assumir duas formas:

**Crossover:**

Dois cromossomos selecionados podem ser combinados por um operador de crossover, cujo resultado substituirá o cromossomo de menor aptidão da população. A seleção de cada cromossomo é realizada por um algoritmo para garantir que a probabilidade de seleção seja proporcional à adequação do cromossomo. Um novo cromossomo tem a chance de ser melhor do que o substituído. O processo é orientado para as sub-regiões do espaço de busca, onde se supõe que exista uma solução ótima (Goldberg, 1989).

**Mutação:**

No processo de mutação, um gene de um cromossomo selecionado é alterado aleatoriamente. Isso fornece chances adicionais de entrar em sub-regiões inexploradas. Finalmente, a evolução é interrompida quando a meta é atingida ou um tempo máximo de CPU é gasto (Goldberg, 1989).

A seguir, o pseudocódigo da operação AG foi escrito:

1. Comece

2. Inicialização da população

3. Repita até (satisfazendo os critérios de rescisão)

• Seleção

• Cross over

• Mutação

• Criar nova população com as soluções mais adequadas

• Avaliação

• Verificar o critério de finalização.

4. Pegue a melhor solução como saída

5. Fim

* 1. **Resolvendo o PCV usando AG**

Conforme mencionado anteriormente, o PCV é conhecido como um problema NP-completo clássico, que possui espaços de busca extremamente grandes e é muito difícil de resolver (Louis & Gong, 2000). Consequentemente, os métodos clássicos para resolver o PCV geralmente resultam em complexidades computacionais exponenciais.

Esses métodos consistem em métodos heurísticos e exatos. Métodos heurísticos, como planos de corte e branch and bound (Padherg & Rinaldi, 1987), só podem resolver de forma otimizada pequenos problemas enquanto os métodos heurísticos, como 2-opt (Lin & Kernighan, 1973), 3-opt, cadeia de Markov (Martin et al., 1991), simulated annealing (Kirkpatrick et al., 1983) e tabu search são bons para grandes problemas.

Além disso, alguns algoritmos baseados em princípios gananciosos, como o vizinho mais próximo e a árvore geradora, podem ser usados ​​como métodos de solução eficientes. No entanto, devido ao grande número de soluções possíveis e grandes espaços de busca, os AGs parecem ser abordagens sábias para resolver o PCV, especialmente quando são acompanhados por operadores genéticos cuidadosamente projetados (Jiao & Wang, 2000). Os AGs buscam o grande espaço de soluções em busca da melhor resposta e os operadores podem ajudar a tornar o processo de busca mais rápido e também preparar a capacidade de evitar ficar preso em ótimos locais.

Nos últimos anos, resolver o PCV usando algoritmos evolutivos e especialmente AGs tem atraído muita atenção. Muitos estudos têm sido realizados e os pesquisadores procuram contribuir com diferentes partes do processo de resolução.

Alguns pesquisadores apresentam diferentes formas de operadores de AG (Yan et al., 2005) em comparação com os anteriores e outros tentam combinar AG com outras abordagens possíveis como OCF (Lee, 2004), ODP e etc. Além disso, alguns autores implementam uma nova ideia evolucionária ao combinar alguns algoritmos e ideias anteriores para criar um novo método (Bonyadi et al., 2007). Aqui, investigamos alguns desses trabalhos e comparamos seus resultados. Devido à disseminação de trabalhos relacionados não podemos citar todos eles aqui. Mas o leitor deve consultar as referências preparadas para obter mais informações.

1. **Otimização da colônia de formigas (OCF)**

**4.1 Introdução**

A heurística OCF (Otimização da Colônia de Formigas) é inspirada no comportamento real das formigas (figura 2) em encontrar o caminho mais curto entre o ninho e o alimento (Beckers et al., 1992). Isso é conseguido por uma substância chamada feromônio, que mostra o traço de uma formiga. Em sua busca, a formiga usa informação heurística que é seu próprio conhecimento de onde vem o cheiro da comida e a decisão das outras formigas do caminho em direção à comida por informações de feromônio (Holldobler & Wilson, 1990).

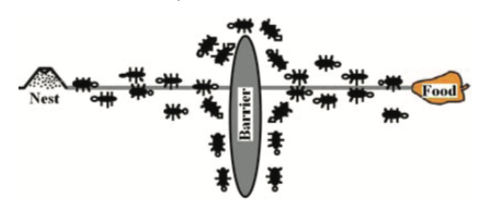


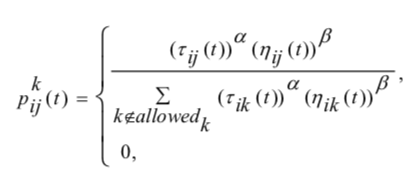
Figura 2: Comportamento real das formigas ao encontrar o menor caminho entre a comida e o ninho.

De fato, o algoritmo utiliza um conjunto de formigas artificiais (indivíduos) que cooperam para a solução de um problema trocando informações via feromônio depositado nas bordas do grafo. O algoritmo OCF é empregado para imitar o comportamento de formigas reais e é o seguinte:

* Inicializar
* Laço
  + Cada formiga é posicionada em um nó inicial
  + Laço
    - Cada formiga aplica uma regra de transição de estado para construir incrementalmente uma solução e uma regra de atualização de feromônio local
  + Até que todas as formigas tenham construído uma solução completa
  + Uma regra de atualização de feromônio global é aplicada
* Até a condição final

**4.2 Transição de estado**

Considere que n é o valor da cidade; m é a quantidade de formigas em um problema ACO; dij é o comprimento do caminho entre as cidades adjacentes i e j; ij (t) é a intensidade da trilha na borda (i, j) no tempo t. No início do algoritmo, um algoritmo de inicialização determina as posições das formigas em diferentes cidades e o valor inicial ij (0), uma pequena constante positiva c para a intensidade da trilha é definida nas bordas. O primeiro elemento da lista de tabu de cada formiga é definido para sua cidade inicial. A transição de estado é dada pela equação 1, que a formiga k da cidade i escolhe se mudar para a cidade j:



Equação 1: Comportamento real das formigas ao encontrar o menor caminho entre a comida e o ninho.

Onde k permitido = {N-tabuk}, que é o conjunto de cidades que ainda precisam ser visitadas por formigas posicionadas na cidade i (para tornar a solução viável), α e β  são parâmetros que determinam a importância relativa da trilha versus visibilidade, e η = 1 / d é a visibilidade da borda (i, j).

**4.3. Resolvendo o PCV usando OCF**

Como mencionado, o algoritmo OCF tem um bom potencial para resolução de problemas e recentemente atraiu muitas atenções especificamente para resolver conjuntos de problemas NP-Completo. Um dos melhores trabalhos para resolver o PCV usa o SCF (Sistema de Colônia de Formigas) é apresentado em (Dorigo & Gambardella, 1997). Eles usam o algoritmo SCF para resolver o PCV e afirmam que o SCF supera outros algoritmos inspirados na natureza, como a computação evolutiva. Além disso, eles compararam o SCF-3-opt, uma versão do SCF aprimorada com um procedimento de busca local, a alguns dos algoritmos de melhor desempenho para PCVs simétricos e assimétricos.

Uma das outras abordagens recentes para resolver o PCV é proposta em (Song et al., 2006). Em particular, a opção que uma formiga caça para a próxima etapa, o uso de uma combinação de dois tipos de modelos de avaliação de feromônio, a mudança do tamanho da população na colônia de formigas durante a execução do algoritmo e a mutação do feromônio foi estudado. Uma das atitudes mais poderosas em seu artigo foi escolher o modelo OCF apropriado proposto por M. Dorigo, que foram chamados de modelos de ciclo, quantidade e densidade de formigas. Esses três modelos diferem na forma como a trilha de feromônios é atualizada. No algoritmo do ciclo das formigas, a trilha é atualizada depois que todas as formigas terminam seus passeios. Em contrapartida, nos dois últimos modelos, cada formiga deposita seu feromônio a cada passo sem esperar o final do passeio (Song et al., 2006). Além disso, eles afirmam que no estágio inicial das iterações, a velocidade de convergência é mais rápida usando o modelo de densidade de formigas em comparação com os outros dois modelos. Assim, no início, o modelo de densidade de formigas é aplicado. O sistema tem a vantagem de utilizar as informações globais, além de ser ele ser usado nas demais ocasiões.

Um mecanismo de mutação igual ao do algoritmo genético foi adicionado ao algoritmo OCF melhorado para ajudar o algoritmo a saltar dos ótimos locais. Em sua proposta de OCF melhorada, um método de dimensionamento populacional é usado que muda o número de indivíduos (formigas).

1. **Conclusão**

Este artigo conceituou o problema do caixeiro viajante, mostrando como a aplicação das soluções é de grande valia para diferentes setores, como por exemplo, o setor logístico.

Isto porque encontrar o caminho de menor distância tem grande relevância nos custos envolvidos para o deslocamento.

Este problema é de fácil conceituação e, quando se considera poucas cidades (vértices), de fácil solução. No entanto, a medida que a quantidade de vértices (cidades) aumenta, as possibilidades que devem ser analisadas aumentam exponencialmente. Ao fazer o uso da “força bruta”, mesmo com a elevada capacidade computacional, poderíamos levar anos para encontrar a solução ótima.

Por isto, as duas abordagens mostradas neste artigo buscam elucidar maneiras de encontrar soluções próximas da ótima, contudo levando um tempo muito menor para isso.

1. **Referências**

Garey, M. R. & Johnson, D. S. (1979). Computers and Intractability: A Guide to the Theory of NP- Completeness. W. H. Freeman.

Eppstein, D. (2007). TSP for Cubic Graphs. Journal of Graph Algorithms and Applications (JGAA), Vol. 11, No. 1, pp. 61–81.

Padherg M. & Rinaldi R., (1987). Optimization of a 532-city symmetric travelling salesman problem by branch and cut. Operations Research Letters, vol. 6, no.1, pp. 1-7.

Lin, S. & Kernighan B., (1973). An effective heuristic algorithm for the traveling-salesman problem. Operations Research, vol. 21, no. 2, pp. 498-516.

Kirkpatrick, S.; Gelatt, C. D. & Vechi, M. P. (1983). Optimization by simulated annealing. Science, vol. 220, no.4598, pp. 671-680.

Teodorovic, D.; Lucic, P.; Markovic, G. & Dell’Orco, M. (2006). Bee Colony Optimization: Principles and Applications. 8th Seminar on Neural Network Applications in Electrical Engineering, NEUREL-2006.

Eiben A. E. & Smith, J. E. (2003). Introduction to Evolutionary Computing. Springer-Verlag.

Haykin, S. (1999). Neural Networks, Prentice-Hall, second edition.