Relatório - Inteligência Artificial - Projeto Computacional 1

Roberto Sérgio Ribeiro de Meneses - 520403

16/12/2024

Algoritmo Genêtico

O Problema do Caixeiro Viajante (TSP, do inglês *Traveling Salesman Problem*) consiste em encontrar o menor caminho que passa por um conjunto de cidades exatamente uma vez e retorna à cidade inicial. Devido à sua complexidade computacional, métodos exatos não são viáveis para instâncias grandes, tornando os Algoritmos Genéticos (AGs) uma alternativa eficiente para encontrar soluções aproximadas.

1 Objetivos

Este trabalho visa implementar um Algoritmo Genético para resolver o TSP utilizando uma matriz de custos como entrada. As etapas principais incluem:

- Inicialização de uma população de soluções;
- Aplicação dos operadores genéticos de seleção, cruzamento e mutação;
- Avaliação e comparação das soluções ao longo de gerações.

2 Metodologia

2.1 Estrutura do Algoritmo

O Algoritmo Genético foi implementado com as seguintes etapas principais:

Inicialização da População A inicialização da população consiste em criar um conjunto inicial de soluções, onde cada indivíduo é um caminho válido que percorre todas as cidades. Esses caminhos são gerados aleatoriamente, representando diferentes permutações das cidades.

Avaliação do Custo A avaliação do custo é uma etapa fundamental onde o algoritmo calcula a "qualidade"de cada solução gerada. O custo de um caminho é obtido somando as distâncias entre cidades consecutivas, incluindo o custo de retornar à cidade inicial.

Seleção No processo de seleção, o algoritmo escolhe os melhores indivíduos da população para reproduzir e gerar novas soluções. No método de seleção por torneio, um pequeno grupo de indivíduos é selecionado aleatoriamente, e o melhor entre eles, ou seja, aquele com o menor custo, é escolhido para passar para a próxima geração.

Cruzamento (Crossover) No cruzamento, uma parte do caminho de um dos pais é combinada com o outro, garantindo que o resultado seja uma permutação válida.

Mutação A mutação ocorre trocando aleatoriamente duas cidades do caminho com uma certa probabilidade.

2.2 Parâmetros iniciais utilizados

Os seguintes parâmetros foram definidos para a execução do algoritmo:

- Tamanho da população: 100;
- Número de gerações: 100;
- Taxa de mutação: 10%;
- Tamanho do torneio: 10.

A matriz de custos utilizada foi uma matriz 15×15 que representa as distâncias entre as cidades.

3 Resultados

Durante a execução do algoritmo, o custo da melhor solução foi registrado a cada geração. A convergência foi observada, com os custos diminuindo gradativamente até a geração final.

Usando Taxa de Mutação = 0.1, Tamanho da População = 100, Tamanho do torneio = 5,temos que o melhor caminho e o custo final foram os seguintes:

- **Caminho:** [4, 12, 1, 13, 3, 14, 8, 6, 11, 10, 9, 7, 0, 2, 5]
- **Custo:** 230

3.1 Testando outros parâmetros

A seguir são apresentados testes realizados variando os principais parâmetros do algoritmo: tamanho da população, taxa de mutação e tamanho do torneio. Os resultados obtidos permitem analisar o impacto de cada parâmetro no custo final da solução.

3.1.1 Variação do Tamanho da População

Foram testados diferentes valores para o tamanho da população:

- **População = 500:** Menor custo variou entre 190 e 212;
- **População = 1000:** Menor custo variou entre 180 e 205;
- **População = 2000:** Menor custo variou entre 170 e 178.

Conclusão: Aumentar o tamanho da população levou a soluções melhores, pois a diversidade de indivíduos aumenta as chances de encontrar caminhos com menor custo. Contudo, populações muito grandes podem tornar o algoritmo mais lento.

3.1.2 Variação da Taxa de Mutação

A taxa de mutação foi testada com valores diferentes:

- **Taxa de Mutação = 0.05:** Menor custo variou entre 198 e 245;
- **Taxa de Mutação = 0.1:** Menor custo variou entre 197 e 239;
- **Taxa de Mutação = 0.15:** Menor custo variou entre 199 e 242;
- **Taxa de Mutação = 0.2:** Menor custo variou entre 198 e 230.

Conclusão: A variação da taxa de mutação não apresentou impacto significativo nos resultados. Uma taxa de mutação muito baixa pode limitar a exploração do espaço de soluções, enquanto taxas mais altas podem dificultar a convergência.

3.1.3 Variação do Tamanho do Torneio

Os valores testados para o tamanho do torneio foram:

- **Tamanho do Torneio = 5:** Menor custo variou entre 188 e 235;
- **Tamanho do Torneio = 10:** Menor custo variou entre 192 e 237;
- **Tamanho do Torneio = 15:** Menor custo variou entre 209 e 252.

Conclusão: Um tamanho de torneio pequeno (como 5) apresentou os melhores resultados, provavelmente devido à maior diversidade na seleção. Aumentar o tamanho do torneio favorece indivíduos com menor custo, mas pode reduzir a diversidade e levar a convergência prematura.

3.2 Resultados do Algoritmo Genético com Variação de Parâmetros

Para analisar o impacto de diferentes configurações no desempenho do algoritmo genético, foram realizados experimentos com os seguintes parâmetros:

- **Tamanho da população:** $[100, 500, 1000, 2000]$;
- **Taxa de mutação:** $[0.05, 0.1, 0.15, 0.2]$;
- **Tamanho do torneio:** $[5, 10, 50, 100]$;
- **Número de gerações:** 50.

Os experimentos resultaram em diferentes custos para cada combinação de parâmetros. O melhor resultado obtido foi um custo total de **168**, utilizando as seguintes configurações:

- **Tamanho da população:** 2000;
- **Taxa de mutação:** 0.1;
- **Tamanho do torneio:** 10.

Este resultado indica que uma população maior combinada com uma taxa de mutação moderada e um tamanho de torneio relativamente pequeno favorece a exploração eficiente do espaço de busca, permitindo ao algoritmo encontrar uma solução de menor custo. A combinação desses fatores sugere um equilíbrio adequado entre diversidade e convergência durante a execução do algoritmo.

4 Conclusão

O Algoritmo Genético demonstrou eficiência na resolução do Problema do Caixeiro Viajante, alcançando boas soluções em poucas gerações. O aumento do tamanho da população aumentou a diversidade genética, enquanto taxas de mutação mais altas (0.2) favoreciam a exploração do espaço de soluções. O tamanho do torneio teve impacto na seleção: torneios maiores (50 e 100) favoreceram indivíduos com melhor desempenho, enquanto torneios menores (5 e 10) mantiveram maior diversidade.

Algoritmo da Colônia de Formigas

O algoritmo executa por um número fixo de iterações, e a cada iteração, as formigas melhoram suas soluções com base na exploração do espaço de soluções e no reforço dos caminhos promissores.

5 Objetivo

O objetivo deste trabalho é aplicar o algoritmo de Colônia de Formigas (ACO) para resolver o Problema do Caixeiro Viajante (PCV), que busca encontrar o caminho mais curto que percorre todas as cidades de um conjunto e retorna à cidade inicial. O algoritmo ACO foi inspirado no comportamento das formigas e tem sido amplamente utilizado para resolver problemas de otimização combinatória como o PCV. A solução visa:

- Minimizar o custo total da viagem.
- Explorar eficientemente o espaço de soluções, utilizando o comportamento das formigas para guiar a busca.
- Melhorar progressivamente a solução através da atualização dos feromônios, refletindo o aprendizado das formigas durante as iterações.

6 Metodologia

A metodologia utilizada para a implementação do algoritmo de Colônia de Formigas é composta por várias etapas, que incluem a construção das soluções, o cálculo do custo das soluções e a atualização dos feromônios. As principais funções do código implementam esses passos.

6.1 Função _selecionar_proxima_cidade

A função `_selecionar_proxima_cidade` é responsável por selecionar a próxima cidade a ser visitada por uma formiga. Para isso, a função utiliza um modelo probabilístico que leva em consideração dois fatores:

- O valor do feromônio (τ) na aresta entre as cidades atuais e as próximas. O feromônio é elevado a um fator α, que controla a influência do feromônio.
- A visibilidade (η), que é o inverso da distância entre as cidades, elevada a um fator β, que controla a importância da distância.

Esses fatores são combinados para gerar probabilidades de escolha para cada cidade, e a cidade com maior probabilidade é escolhida pela formiga. A probabilidade de uma formiga escolher uma cidade j a partir da cidade i é dada pela fórmula:

$$P_{ij} = \frac{\tau_{ij}^{\alpha} \eta_{ij}^{\beta}}{\sum_{k \in N_i} \tau_{ik}^{\alpha} \eta_{ik}^{\beta}}$$

onde N_i representa o conjunto de cidades não visitadas a partir da cidade i.

6.2 Função _construir_solucao

A função `_construir_solucao` constrói uma solução completa para uma formiga. Começando com uma cidade aleatória, a formiga seleciona as cidades subsequentes uma a uma, utilizando a função `_selecionar_proxima_cidade`. O processo continua até que todas as cidades sejam visitadas. Uma vez que todas as cidades tenham sido visitadas, a formiga retorna à cidade inicial, completando o ciclo. A solução gerada por uma formiga é representada como uma lista de cidades, ou seja, um caminho que percorre todas as cidades.

6.3 Função _calcular_custo_caminho

A função `_calcular_custo_caminho` recebe uma solução (caminho) como entrada e calcula o custo total da viagem. O custo de um caminho é a soma das distâncias entre as cidades consecutivas, incluindo o custo de retornar à cidade inicial. O custo total $C(S)$ para uma solução $S = (s_1, s_2, \ldots, s_n)$ é dado por:

$$C(S) = \sum_{i=1}^{n-1} \text{custo}(s_i, s_{i+1}) + \text{custo}(s_n, s_1)$$

onde $\text{custo}(s_i, s_{i+1})$ é o custo da viagem entre a cidade s_i e a cidade s_{i+1}.

6.4 Função _atualizar_feromonios

A função `_atualizar_feromonios` é responsável pela atualização do feromônio após todas as formigas completarem a construção de suas soluções. O feromônio evapora ao longo do tempo, o que é modelado pela multiplicação do feromônio por $(1 - \text{taxa\_evaporacao})$. Em seguida, o feromônio é depositado nas arestas que foram visitadas pelas formigas. O valor do feromônio depositado é inversamente proporcional ao custo da solução, o que significa que soluções melhores (com menor custo) recebem mais feromônio. O feromônio depositado em cada aresta é dado por:

$$\Delta\tau_{ij} = \frac{Q}{C(S)}$$

onde Q é um valor constante e $C(S)$ é o custo da solução.

6.5 Função executar

A função `executar` orquestra o processo de execução do algoritmo. Ela realiza múltiplas iterações, onde em cada uma as formigas constroem suas soluções, os custos são calculados e os feromônios são atualizados. O objetivo é encontrar a melhor solução, ou seja, o caminho com o menor custo, ao longo das iterações. Para cada iteração, a função imprime o melhor custo encontrado até o momento.

6.6 Parâmetros do Algoritmo

O algoritmo foi configurado com os seguintes parâmetros:

- **Matriz de Custos:** Define os custos de deslocamento entre cada par de cidades.
- **Número de Formigas:** 1000, que representa a quantidade de formigas que exploram o espaço de soluções.
- **Número de Iterações:** 50, que define o número de vezes que as formigas irão explorar o espaço de soluções e atualizar os feromônios.
- **Alfa:** 1.0, o peso do feromônio na fórmula de escolha da próxima cidade.
- **Beta:** 2.0, o peso da visibilidade (inverso da distância) na fórmula de escolha da próxima cidade.
- **Taxa de Evaporação:** 0.5, que define a taxa com a qual o feromônio se evapora a cada iteração.
- **Feromônio Inicial:** 1.0, valor inicial do feromônio em todas as arestas.
- **Q:** 300, constante utilizada para calcular a quantidade de feromônio a ser depositada ao longo dos caminhos.

O algoritmo tem como objetivo minimizar o custo total do caminho percorrido pelas formigas, resultando em uma solução aproximada para o Problema do Caixeiro Viajante.

7 Resultados do Algoritmo

Durante a execução do algoritmo, o custo da melhor solução foi registrado a cada geração. A convergência foi observada, com os custos diminuindo gradativamente até a geração final.

Usando número de formigas = 100, alfa = 1.0, Beta = 2.0, taxa de evaporação = 0.5, feromonio inicial = 1.0 e q =300, temos:

7.1 Solução Encontrada

O melhor caminho e o custo final foram os seguintes:

- **Caminho:** [2, 14, 13, 11, 10, 4, 5, 8, 3, 12, 1, 9, 6, 7, 0]
- **Custo:** 168.00

7.2 Testando outros parâmetros

7.2.1 Variação do Tamanho da população de formigas

Foram testados diferentes valores para o tamanho da população:

- **População = 500:** Menor custo 168;
- **População = 1000:** Menor custo 168;
- **População = 2000:** Menor custo 168;

7.3 Teste de Parâmetros

7.3.1 Tamanho da População

Ao testar diferentes tamanhos de população, foi possível observar o seguinte comportamento:

- Para **tamanho_população = 500**, o menor custo encontrado variou entre 168 e 175.
- Para **tamanho_população = 1000**, o menor custo encontrado foi consistentemente 168.
- Para **tamanho_população = 2000**, o menor custo também ficou em torno de 168.

Conclusão: Aumentar o tamanho da população não parece ter impacto significativo na melhoria do custo final, já que o menor custo encontrado permanece em torno de 168 a partir de 1000 indivíduos.

7.3.2 Tamanho do Torneio

Quando o número de participantes no torneio foi alterado, os resultados mostraram uma tendência interessante:

- Para **tamanho_torneio = 5**, o menor custo foi 168.
- Para **tamanho_torneio = 10**, o menor custo foi 168.
- Para **tamanho_torneio = 20**, o menor custo foi 168.

Conclusão: Modificar o tamanho do torneio não teve um grande efeito na qualidade da solução, mantendo o menor custo em torno de 168, sugerindo que o algoritmo já é eficiente com um número moderado de competidores.

7.3.3 Taxa de Manutenção (Taxa de Evaporação de Feromônio)

Variações na taxa de evaporação mostraram os seguintes resultados:

- Para **taxa_evaporacao = 0.5**, o menor custo foi 168 e 175.
- Para **taxa_evaporacao = 0.75**, o menor custo foi 168.
- Para **taxa_evaporacao = 0.9**, o menor custo foi 168.

Conclusão: A taxa de evaporação teve um efeito muito pequeno sobre o custo final, com o menor custo permanecendo em 168 mesmo em valores elevados de evaporação.

7.3.4 Valor de Alpha (Peso do Feromônio)

Ao testar diferentes valores para o parâmetro α (que controla a influência do feromônio), os resultados foram:

- Para **alpha = 1.0**, o menor custo variou entre 168 e 175.
- Para **alpha = 2.0**, o menor custo variou entre 180 e 187.
- Para **alpha = 3.0**, o menor custo variou entre 178 e 193.

Conclusão: Aumentar α parece aumentar o custo final, possivelmente devido ao fato de o algoritmo dar maior peso ao feromônio, o que pode levar a uma exploração inadequada do espaço de soluções.

7.3.5 Valor de Beta (Peso da Visibilidade)

Variações no valor de β (que controla a influência da visibilidade) resultaram nos seguintes custos:

- Para **beta = 1.0**, o menor custo variou entre 168 e 178.
- Para **beta = 2.0**, o menor custo foi 168.
- Para **beta = 3.0**, o menor custo variou entre 168 e 175.

Conclusão: A variação de β teve um impacto menor do que o esperado, com o custo final se mantendo em torno de 168 para a maioria dos valores testados.

7.3.6 Valor de Feromônio Inicial

Alterando o valor inicial do feromônio, os resultados foram:

- Para **feromonio_inicial = 1.0**, o menor custo variou entre 168 e 175.
- Para **feromonio_inicial = 1.5**, o menor custo variou entre 168 e 178.
- Para **feromonio_inicial = 2.0**, o menor custo variou entre 168 e 176.

Conclusão: A escolha do valor inicial do feromônio não causou grandes variações no custo final, sugerindo que valores iniciais mais elevados não resultam necessariamente em soluções mais eficientes.

7.3.7 Valor de Q (Depósito de Feromônio)

Variações no valor de Q, que controla o depósito de feromônio em cada solução, mostraram os seguintes resultados:

- Para **q = 300**, o menor custo variou entre 168 e 175.
- Para **q = 150**, o menor custo foi entre 168 e 175.
- Para **q = 450**, o menor custo foi entre 168 e 178.

Conclusão: O valor de Q não teve um grande impacto sobre a qualidade da solução final, com o custo permanecendo estável em torno de 168 independentemente do valor do parâmetro.

7.4 Resultados do Algoritmo Genético com Variação de Parâmetros

Neste experimento, o algoritmo de Colônia de Formigas foi executado com diferentes combinações de parâmetros, com o objetivo de avaliar o impacto de cada variável no desempenho do algoritmo. Os parâmetros variáveis utilizados foram:

- **Tamanho da População**: [100, 500, 1000]
- **Número de Gerações**: 50 (fixo)
- α: [0.5, 1.0, 2.0]
- β: [1.0, 2.0, 3.0]
- **Taxa de Evaporação**: [0.5, 0.75, 0.9]
- **Feromônio Inicial**: [1.0, 1.5, 2.0]
- **Q**: [150, 300, 450]

Esses parâmetros foram combinados de diferentes formas para realizar múltiplas execuções do algoritmo. A seguir, apresentamos os resultados obtidos para cada configuração.

População	α	β	**Taxa de Evaporação**	**Feromônio Inicial**	**Q**	**Menor Custo**
100	0.5	2.0	0.5	1.0	300	168
100	0.5	2.0	0.5	1.5	450	168
100	1.0	1.0	0.75	1.0	300	168
100	2.0	2.0	0.75	2.0	300	168
500	2.0	2.0	0.5	1.0	300	168
500	2.0	2.0	0.75	2.0	450	168
1000	0.5	1.0	0.75	1.5	450	168
1000	0.5	2.0	0.5	1.0	150	168
1000	0.5	2.0	0.5	1.0	300	168
1000	0.5	2.0	0.5	1.0	450	168

Tabela 1: Resultados obtidos com diferentes combinações de parâmetros

8 Conclusão

O algoritmo de Colônia de Formigas alcançou sempre o melhor custo de 168, independentemente das variações nos parâmetros. O tamanho da população afetou o tempo de execução, com populações maiores demandando mais iterações para convergir. O parâmetro α (peso do feromônio) impactou negativamente quando foi maior que β, pois o algoritmo passou a dar demasiada importância ao feromônio. A taxa de evaporação, variando entre 0.5, 0.75 e 0.9, influenciou a rapidez da exploração, com taxas menores favorecendo mais exploração. O feromônio inicial e Q não alteraram o custo, mas afetaram a velocidade de convergência. Em resumo, o principal fator para o desempenho foi o equilíbrio entre α e β.