Professor: George Felipe Fernandes Vieira
Algoritmos I
Bacharelado em Tecnologia da Informação
Universidade Federal Rural do Semi-Árido

Discente: Frederico Gregório Emidio Santos Ferreira

Relatório Simples da Solução da Prática 2 de Lab de Algoritmos

Problema Proposto:

Desenvolver um programa para ajudar um caixeiro viajante a encontrar o caminho de menor distância ao visitar todas as cidades de uma rota e retornar à cidade inicial.

Requisitos da Atividade:

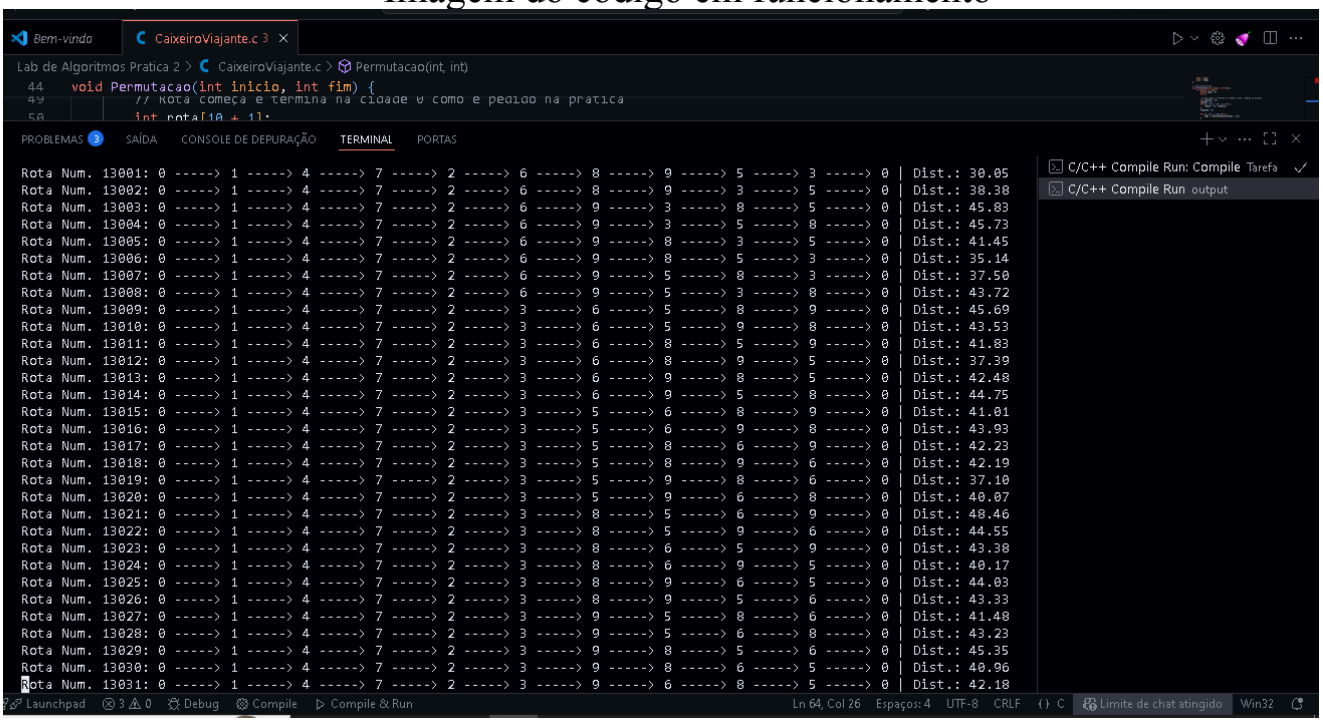
1. Calcular a distância entre todas as cidades usando a fórmula da distância euclidiana.
2. Gerar todas as rotas possíveis que começam e terminam na cidade 0.
3. Avaliar cada rota e determinar qual tem a menor distância total.

Solução Implementada:

1. Definição das coordenadas das cidades.
2. Cálculo da distância entre cada par de cidades com base na fórmula euclidiana.
3. Geração de todas as rotas possíveis (fixando a cidade 0 como início e fim).
4. Avaliação de cada rota e impressão da distância total.
5. Apresentação da melhor rota encontrada.

Todo o Código da solução está no repositório Publico:
https://github.com/FredGregorio/Pratica2LabdeAlgoritmos.git

Imagem do código em funcionamento



Código Finalizado e impressão da melhor rota encontrada e sua distância

Otimização para a solução Utilizando **(Simulated Annealing)**

O Problema do Caixeiro Viajante foi introduzido por K. Menge em 1932. O PCV é um problema combinatório NP difícil comprovado que envolve um caixeiro com n cidades e que precisa visitar cada cidade uma única vez, retornando à primeira cidade que iniciou a rota. O PCV tem tido muitas aplicações reais, como roteamento em engenharia de redes, distribuição crítica para empresas de logística em todo o mundo, roteamento de veículos, gestão da cadeia de suprimentos, etc. Diversos algoritmos de otimização têm sido utilizados para chegar a um ótimo global, encontrando a rota mais curta que satisfaça as restrições mencionadas. São eles: algoritmos genéticos, otimização de colônias de formigas, **Simulated Annealing**, algoritmos de Dijkstra, redes neurais, etc.

O algoritmo **Simulated Annealing (Recozimento Simulado)** é um método computacional inspirado no campo da física. Ele simula o processo físico de recozimento de sólidos. Nesse processo, um material como metal ou vidro é elevado a uma temperatura elevada e, em seguida, deixado para esfriar, permitindo que as regiões locais de ordem cresçam para fora, reduzindo assim a tensão e aumentando a resistência dúctil do material. No entanto, foi somente na 5ª Conferência Internacional IEEE sobre Redes de Banda Larga e Tecnologia Multimídia em novembro de 2013 que foi apresentado como um método metaheurístico independente para resolver problemas de otimização discreta e contínua.

Este método tem sido comumente referido como um dos modelos metaheurísticos mais antigos. Ele apresenta excelente desempenho em evitar mínimos locais. Para isso, o algoritmo aceita os piores candidatos com uma probabilidade dependente da temperatura (uma variável de controle) e da diferença de aptidão dada pela fórmula abaixo:

$$p = \begin{cases} 1, & if\ f(y) \geq f(x) \\ e^{-(f(y)-f(x))/T}, & otherwise \end{cases}$$

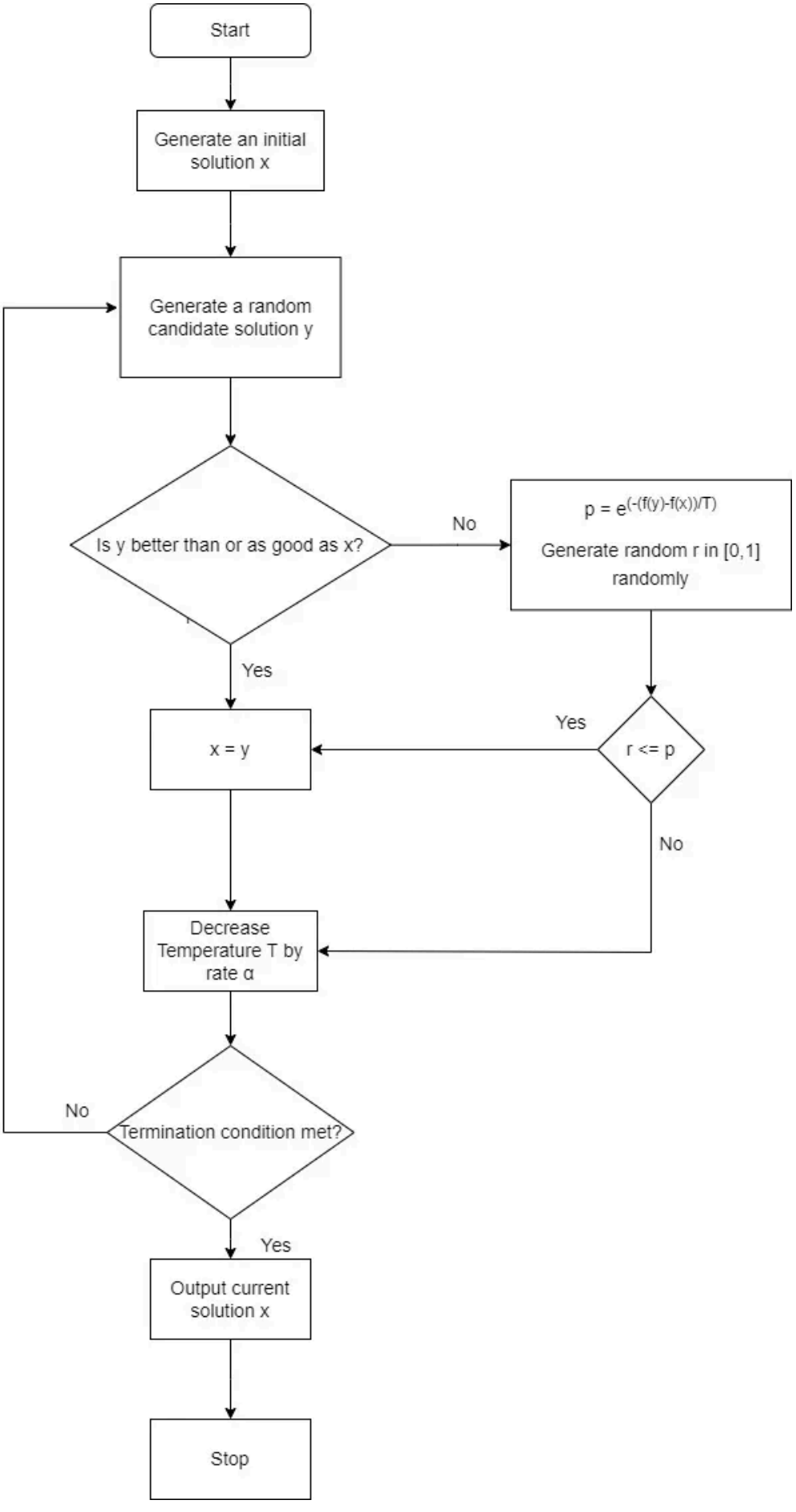
p — a probabilidade de aceitar o novo candidato à solução, y.

x — candidato à solução inicial (neste caso, é uma rota)

y — novo candidato à solução

f(x) — é a função que mede o desempenho do candidato à solução (função de aptidão)

T — a temperatura que é o parâmetro de controle.


Start
Generate an initial solution x
Generate a random candidate solution y
Is y better than or as good as x?
No
p = e^(-(f(y)-f(x))/T)
Generate random r in [0,1] randomly
Yes
x = y
Yes
r <= p
No
Decrease Temperature T by rate α
No
Termination condition met?
Yes
Output current solution x
Stop

Como nossa probabilidade de aceitar novos candidatos piores depende da temperatura, à medida que a temperatura esfria, eventualmente aceitaremos apenas candidatos melhores que o atual.

Como o algoritmo de simulated annealing compara diferentes soluções candidatas e decide qual delas será adotada a cada iteração, é necessário gerar essas soluções. No trabalho, foram utilizados quatro métodos.

- **Operador inverso (x):** altera a ordem das rotas entre dois nós i e j selecionados aleatoriamente. Onde $i,j \leq 0 \leq n$ tal que $x^1[i] = x[j]$, $x^1[i+1] = x[j-1]$, etc. Espero que um exemplo esclareça melhor. Digamos que a rota inicial seja '1–2–3–4–5–6'; este operador poderia produzir '1–4–3–2–5–6'. As posições aleatórias selecionadas neste exemplo são a cidade 2 e a cidade 5. O número inteiro mínimo entre i e j será 'i' e o máximo será considerado 'j'.

- **Operador de troca (x):** troca a posição de duas cidades em uma rota. Duas posições, i e j, são selecionadas aleatoriamente e as cidades nessas posições são trocadas entre si. '1–2–3–4–5–6' poderia se tornar '1–5–3–4–2–6'.

- **Operador de inserção(x):** Este operador seleciona uma cidade na posição aleatória 'i' e a move para uma posição aleatória 'j' em outro lugar na rota.

- **Inserir sub-rotas(x):** Isso é semelhante ao operador de troca de rota, mas em vez de inserir uma cidade em uma posição diferente, um intervalo de cidades é selecionado como uma sub-rota e inserido em uma posição aleatória.

Embora vários algoritmos de Simulated Annealing terminem quando a temperatura atinge 0, para ter um espaço de busca muito grande, esse algoritmo só termina quando um candidato é selecionado 1.500 vezes e a mesma pontuação de aptidão ocorre 150.000 vezes. Você pode alterar esses valores conforme desejar. Foi utilizada uma temperatura inicial de 5.000. Normalmente, é uma boa prática executar o algoritmo mais de uma vez para obter uma solução mais geral. No trabalho o algoritmo foi executado 10 vezes e calculada a média dos resultados.

A desvantagem do Simulated Annealing é seu alto tempo de convergência. É por isso que outros métodos de otimização são preferidos, como algoritmos genéticos, métodos de permutação gulosa, otimização de colônias de formigas, escalada de colinas, etc. Houve várias tentativas bem-sucedidas de combinar muitos desses métodos. O recozimento simulado é uma ótima ferramenta para encontrar seus ótimos globais.

Link do Repositório GIT do trabalho do **Simulated Annealing para o caixeiro viajante:** http://github.com/allanah1/Simulated_Annealing

Fonte:
https://medium.com/@francis.allanah/travelling-salesman-problem-using-simulated-annealing-f547a71ab3c6

Referências contidas no trabalho da fonte.

1. G. Ye e X. Rui, 'Um algoritmo genético e de simulação de recozimento aprimorado para TSP', em 2013, 5ª Conferência Internacional IEEE sobre Redes de Banda Larga e Tecnologia Multimídia, novembro de 2013, pp. 6–9. DOI: 10.1109/ICBNMT.2013.6823904.

2. S. Zhan, J. Lin, Z. Zhang e Y. Zhong, 'Algoritmo de Simulação de Recozimento Baseado em Lista para o Problema do Caixeiro Viajante', Comput. Intell. Neurosci., vol. 2016, p. 1712630, mar. 2016, DOI: 10.1155/2016/1712630.

3. J. Liu e W. Li, 'Método de permutação gananciosa para algoritmo genético no problema do caixeiro viajante', em 2018, 8ª Conferência Internacional sobre Informação Eletrônica e Comunicação de Emergência (ICEIEC) , junho de 2018, pp. 47–51. doi: 10.1109/ICEIEC.2018.8473531.