**INTRODUÇÃO**

Este trabalho tem como objetivo resolver um problema inspirado no Travelling Salesman Problem (TSP), aplicando técnicas de modelagem em grafos e algoritmos de otimização. O desafio proposto busca minimizar a maior distância entre dois pontos consecutivos de um ciclo hamiltoniano em um grafo completo, um problema relevante em áreas como transporte e logística, onde a eficiência das rotas é crucial.

Para alcançar uma solução eficiente, foi necessário combinar conceitos teóricos com uma implementação prática estruturada. A abordagem envolveu desde a manipulação de dados até o desenvolvimento de algoritmos iterativos, passando por ajustes e refinamentos para garantir melhores resultados. Este trabalho não apenas proporcionou um aprendizado valioso sobre estratégias de otimização e estruturas de dados, mas também destacou a importância da análise crítica e da constante evolução das soluções propostas.

**FORMULAÇÃO**

O problema abordado neste projeto consiste no desenvolvimento de um algoritmo para processar arquivos no formato **“.ins”** do tipo Travelling Salesman Problem (TSP). A partir dos dados de coordenadas **x** e **y** de cada vértice, era necessário calcular a distância entre esse vértice e todos os outros, formando assim um grafo completo. O objetivo final do algoritmo era construir um ciclo hamiltoniano que minimizasse a maior distância entre dois vértices consecutivos do percurso resultante.

Diante desse desafio, a primeira etapa foi a leitura do arquivo de entrada, extraindo informações como o número de vértices, o tipo de cálculo de distância entre arestas (EDGE\_WEIGHT\_TYPE) e as coordenadas dos vértices. Para essa tarefa, utilizamos as bibliotecas fstream e sstream, garantindo uma manipulação eficiente dos dados. Em seguida, implementamos métodos específicos para calcular automaticamente as distâncias entre os vértices, considerando o tipo de aresta especificado no arquivo de entrada. Essas distâncias foram então armazenadas em uma lista de adjacência, juntamente com um valor booleano inicializado como false, indicando se o vértice havia sido acessado durante a execução do algoritmo.

Com o grafo devidamente estruturado, partimos para a formulação da solução. Inicialmente, tomamos como referência o algoritmo do vizinho mais próximo, que utiliza um ponto inicial fixo – vértice 0. O princípio básico desse método é identificar a menor e a segunda menor aresta do vértice inicial: a menor define a saída do percurso e a segunda menor representa o retorno à origem. Dessa forma, o ciclo começa com uma ida e volta otimizadas em termos de distância.

A partir dessa definição inicial, o algoritmo segue um processo iterativo. Ao sair de um vértice, todas as suas arestas são desabilitadas para futuras comparações, impedindo que ele seja revisitado. Além disso, a aresta de retorno ao vértice de origem só se torna disponível quando todos os outros vértices já foram percorridos, garantindo o fechamento adequado do ciclo. O critério de escolha para o próximo vértice do percurso é sempre a aresta de menor distância disponível no momento.

Após implementar essa abordagem, realizamos testes comparativos entre o maior valor de aresta presente no percurso e um valor de referência. Os resultados indicaram que ainda havia espaço significativo para melhorias. Assim, buscamos aperfeiçoar o algoritmo e tivemos a ideia de introduzir uma segunda saída no percurso. Nessa nova abordagem, ao invés de utilizar apenas um único caminho a partir do vértice inicial, consideramos os dois menores vértices adjacentes à origem. Para isso, criamos dois marcadores distintos que avançavam de forma independente, seguindo a mesma lógica do algoritmo original. Sempre que um marcador ficava sem vértices adjacentes disponíveis, o outro assumia a responsabilidade de continuar o percurso até que o ciclo fosse concluído. Essa modificação resultou em uma melhora perceptível nos resultados, mas ainda distante do valor de referência.

Após novas análises, identificamos uma limitação fundamental do algoritmo: utilizar um único vértice inicial fixo restringia significativamente a qualidade da solução. Para contornar essa questão, implementamos um procedimento que testava todos os vértices como possíveis pontos iniciais. O ciclo resultante escolhido era aquele cuja maior distância entre dois vértices consecutivos fosse a menor possível. Esse ajuste final trouxe uma melhora expressiva nos resultados, aproximando-os significativamente do valor de referência e permitindo a conclusão da solução final de forma satisfatória.

**RESULTADOS OBTIDOS**

* **Tabela de resultados:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **INSTÂNCIA** | **REFERÊNCIA** | **SI** | **SF** | **DESVIO SI-SF (%)** | **DESVIO SF-REFERÊNCIA (%)** |
| **1** | 3986 | 7556 | 5893 | 22.01 | 32.36 |
| **2** | 1289 | 2400 | 1350 | 43.75 | 4.52 |
| **3** | 1476 | 1476 | 1476 | 0.00 | 0.00 |
| **4** | 1133 | 2108 | 1259 | 40.28 | 10.01 |
| **5** | 546 | 1193 | 991 | 16.93 | 44.90 |
| **6** | 431 | 1270 | 552 | 56.54 | 21.92 |
| **7** | 219 | 1312 | 505 | 61.51 | 56.63 |
| **8** | 266 | 2508 | 654 | 73.92 | 59.33 |
| **9** | 52 | 264 | 111 | 57.95 | 53.15 |
| **10** | 237 | 941 | 590 | 37.30 | 59.83 |

* **Tempo computacional da solução final (SF):**

|  |  |
| --- | --- |
| **INSTÂNCIAS** | **TEMPO COMPUTACIONAL (ms)** |
| **1** | 27067 |
| **2** | 175797 |
| **3** | 475381 |
| **4** | 712810 |
| **5** | 196180 |
| **6** | 231713 |
| **7** | 197747 |
| **8** | 102778 |
| **9** | 35038 |
| **10** | 458047 |

* **Configuração do computador utilizado:**

**Sistema operacional:** Windows 11 Home Single Language, 64 bits, processador baseado em x64  
**Processador:** AMD Ryzen 5 3500U with Radeon Vega Mobile Gfx 2.10 GHz

**RAM instalada:** 8,00 GB

**CONCLUSÃO**

O desenvolvimento deste trabalho proporcionou um aprendizado significativo, tanto na aplicação prática de algoritmos voltados à resolução de problemas complexos quanto na utilização de ferramentas fundamentais da programação. A experiência de implementar uma solução eficiente para o problema do TSP com foco na minimização da maior distância entre vértices em um ciclo hamiltoniano exigiu uma compreensão profunda de conceitos como grafos, estruturas de dados, e estratégias de otimização.

Ao longo do processo, enfrentamos diversos desafios, desde a manipulação de arquivos **.ins** e a construção do grafo completo até a formulação e refinamento de algoritmos capazes de atender aos requisitos do problema. Cada etapa representou uma oportunidade de aprimorar habilidades técnicas e explorar abordagens criativas para solucionar limitações identificadas nas versões iniciais do código. Além disso, a análise e comparação dos resultados com valores de referência nos permitiram desenvolver um pensamento crítico em relação à eficiência e à qualidade das soluções implementadas.

O uso das bibliotecas padrão do C++ (fstream, sstream, utility, queue, vector e cmath) foi essencial para garantir a robustez e a eficácia da solução. Este projeto também destacou a importância de uma boa organização do código e da divisão clara de responsabilidades entre os métodos, aspectos que facilitaram o processo de refatoração e melhoria contínua do algoritmo.

Do ponto de vista acadêmico e profissional, o trabalho nos permitiu consolidar conhecimentos fundamentais em programação, análise de algoritmos e otimização, além de reforçar a importância do estudo contínuo de técnicas computacionais

**BIBLIOGRAFIA**

**Heurística do Vizinho Mais Próximo (Pedro Munari) -** https://www.youtube.com/watch?v=tEryMeECijE