**ENGENHARIA DE SOFTWARE**

LUIZ HENRIQUE GALHARDO FARIAS

**TRABALHO FINAL**

**SISTEMAS DISTRIBUIDOS**

**DESAFIO DO CAIXEIRO VIAJANTE**

**CORNÉLIO PROCÓPIO**

**2024**

**SUMÁRIO**

1. INTRODUÇÃO 3

2. OBJETIVOS 4

3. MATERIAIS E MÉTODOS 5

4. RESULTADOS ESPERADOS 7

REFERÊNCIAS 10

# 1. INTRODUÇÃO

1. O Problema do Caixeiro Viajante (TSP, do inglês Traveling Salesman Problem) é um dos desafios mais conhecidos e estudados na área de otimização combinatória e ciência da computação. O TSP consiste em determinar a rota mais curta que permite a um "caixeiro viajante" visitar todas as cidades de um conjunto exatamente uma vez, retornando ao ponto de partida. Esse problema é de grande importância prática em áreas como logística, planejamento de rotas, entregas e redes de distribuição.
2. Devido à sua complexidade computacional, o TSP é classificado como NP-difícil, o que implica que, com o aumento do número de cidades, o tempo necessário para encontrar uma solução cresce exponencialmente. Uma abordagem sequencial para resolver o problema realiza cálculos de forma linear, tornando-se inviável para grandes conjuntos de cidades. No entanto, os avanços em programação paralela e distribuída oferecem formas mais eficientes de abordar o TSP, dividindo a carga de trabalho entre múltiplos núcleos de processamento e diferentes máquinas.
3. Este trabalho busca demonstrar a implementação do TSP em Java utilizando três métodos distintos: sequencial, paralela e distribuída. Inicialmente, o problema será resolvido de maneira sequencial, destacando as limitações de desempenho para grandes instâncias. Em seguida, uma implementação paralela será desenvolvida, explorando o uso de múltiplos núcleos de CPU para reduzir o tempo de execução. Finalmente, a abordagem distribuída será aplicada, utilizando várias máquinas para processar subproblemas em paralelo, proporcionando um ganho significativo de desempenho em cenários de alta complexidade.
4. Essas abordagens permitirão uma análise comparativa de desempenho e eficiência, ressaltando os benefícios e desafios de cada técnica no contexto do TSP. Além disso, o trabalho evidenciará como a computação paralela e distribuída pode contribuir para a solução de problemas complexos na ciência da computação.

# 2. OBJETIVOS

Este trabalho tem como objetivo principal demonstrar a resolução do Problema do Caixeiro Viajante (TSP) em Java utilizando três abordagens diferentes: sequencial, paralela e distribuída. Além de implementar essas metodologias, o trabalho visa analisar detalhadamente o tempo de execução em cada abordagem, evidenciando os ganhos de desempenho proporcionados pela paralelização e distribuição da carga de trabalho.

Para alcançar esse objetivo, foram definidos os seguintes objetivos específicos:

1. **Implementar uma Solução Sequencial:** Desenvolver uma solução em Java para o TSP de forma sequencial, utilizando um único núcleo de processamento. Esta implementação servirá como referência para comparação com as outras abordagens.
2. **Implementar uma Solução Paralela:** Desenvolver uma versão paralela do TSP em Java, explorando a capacidade de execução simultânea em múltiplos núcleos de CPU. Esta abordagem visa dividir o problema em partes menores e processá-las em paralelo, reduzindo o tempo de execução em comparação com a versão sequencial.
3. **Implementar uma Solução Distribuída:** Criar uma solução distribuída para o TSP em Java, utilizando múltiplas máquinas para processar subproblemas de forma coordenada. A abordagem distribuída busca dividir a carga de trabalho entre diferentes unidades de processamento em uma rede, proporcionando ganhos de desempenho em cenários de alta complexidade.
4. **Analisar e Comparar o Desempenho:** Medir e comparar o tempo de execução das abordagens sequencial, paralela e distribuída em diferentes instâncias do problema. Esta análise permitirá quantificar os ganhos obtidos por meio da paralelização e distribuição, destacando as limitações e benefícios de cada abordagem no contexto da resolução do TSP.

Este trabalho busca fornecer uma compreensão clara de como diferentes estratégias computacionais podem impactar o desempenho na solução de problemas complexos e como a utilização de técnicas paralelas e distribuídas pode reduzir significativamente o tempo de execução em problemas que envolvem grandes volumes de dados e alta complexidade computacional.

# 3. MATERIAIS E MÉTODOS

* NETBEANS;
* WORD;
* CHATGPT AI;
* GEMINI;

# 4. RESULTADOS ESPERADOS

Espera-se que as abordagens de resolução do Problema do Caixeiro Viajante (TSP) em Java – sequencial, paralela e distribuída – apresentem variações significativas em termos de tempo de execução, eficiência e escalabilidade. Abaixo estão os resultados esperados para cada abordagem:

### **Abordagem Sequencial:**

* **Tempo de Execução:** Deve crescer exponencialmente com o aumento do número de cidades, evidenciando suas limitações para grandes instâncias.
* **Escalabilidade:** Baixa, pois utiliza apenas um núcleo de processamento e não permite divisão eficiente do trabalho.
* **Referência Base:** Os tempos obtidos servirão como base para comparação com as outras abordagens.

### **Abordagem Paralela:**

* **Redução no Tempo de Execução:** Espera-se uma redução substancial em relação à abordagem sequencial, especialmente em processadores multicore, mais evidente com um maior número de cidades.
* **Melhor Utilização de Recursos:** Utiliza múltiplos núcleos de CPU, distribuindo a carga de trabalho e demonstrando a eficiência do uso de threads.
* **Limitações de Escalabilidade:** Apesar de melhorias, ainda haverá limitações quando o número de cidades crescer significativamente.

### **Abordagem Distribuída:**

* **Significativa Redução do Tempo de Execução:** Deverá apresentar a maior redução no tempo de execução, dividindo o problema entre diferentes máquinas ou instâncias.
* **Alta Escalabilidade:** Permite adicionar mais máquinas conforme a complexidade aumenta, melhorando o desempenho.
* **Eficiência da Comunicação entre Nós:** Dependerá dos tempos e custos de comunicação entre máquinas, beneficiando mais problemas com grande volume de dados, mas introduzindo sobrecarga em instâncias menores.

Esses resultados esperados evidenciam os benefícios da paralelização e distribuição na resolução de problemas complexos, mostrando como essas abordagens reduzem o tempo de execução e permitem escalabilidade em sistemas com recursos limitados.

# REFERÊNCIAS