Análise de Desempenho da Paralelização do Problema de Caixeiro Viajante

Gabriel Freytag
Guilherme Arruda
Rogério S. M. Martins
Edson L. Padoin

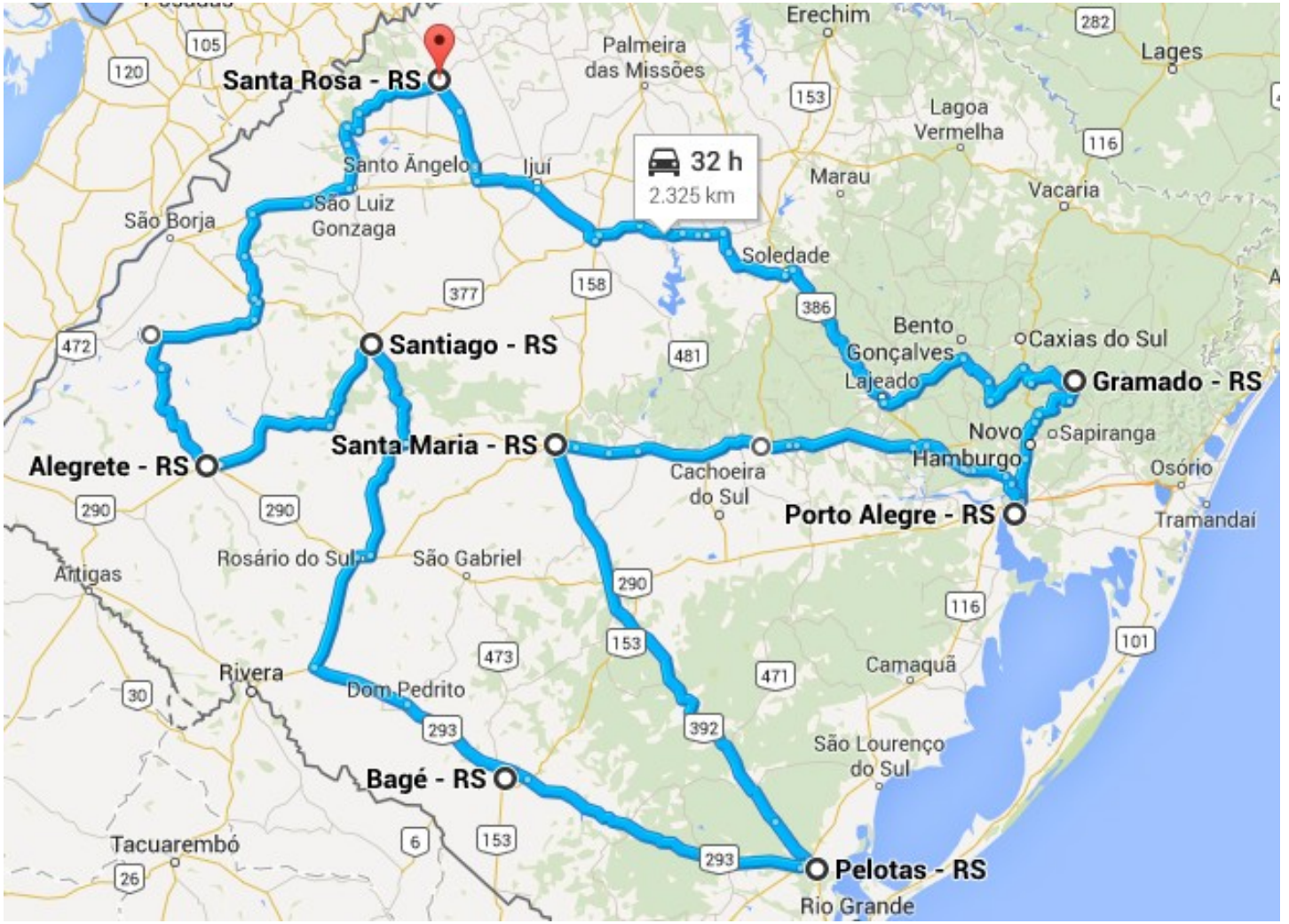
Universidade Regional do Noroeste do Estado do Rio Grande do Sul (UNIJUÍ)

Sumário

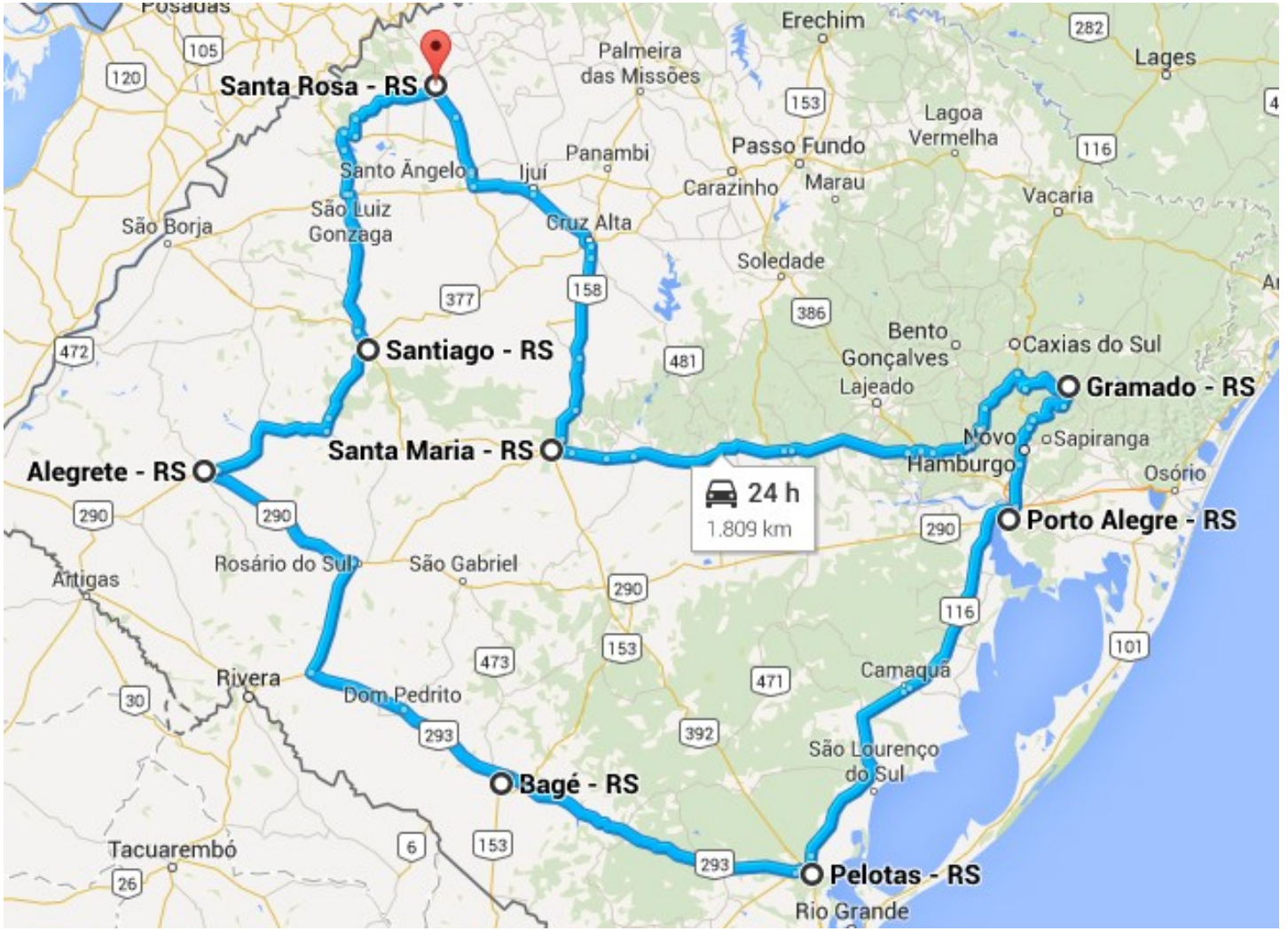
Introdução

- Problema do Caixeiro Viajante:
  - Clássico problema de **otimização combinatória**;
    - Encontrar melhor subconjunto, conforme restrições, sobre um conjunto discreto e finito de dados;
    - Exigem grandes tempos computacionais;
    - Tempo exponencial conforme tamanho das entradas;
    - Problema NP (*Non-Deterministic Polynomial time*).
  - Aplicável a inúmeros problemas práticos;

Introdução


Santa Rosa - RS
Santo Ângelo
Ijuí
São Luiz Gonzaga
São Borja
32 h
2.325 km
Santiago - RS
Alegrete - RS
Santa Maria - RS
Rosário do Sul
São Gabriel
Rivera
Dom Pedrito
Bagé - RS
Tacuarembó
Pelotas - RS
Rio Grande
Cachoeira do Sul
Soledade
Marau
Palmeira das Missões
Erechim
Lagoa Vermelha
Vacaria
Lages
Bento Gonçalves
Caxias do Sul
Lajeado
Gramado - RS
Novo Hamburgo
Sapiranga
Porto Alegre - RS
Osório
Tramandaí
Camaquã
São Lourenço do Sul

Introdução


Santa Rosa - RS
Santiago - RS
Santa Maria - RS
Alegrete - RS
Gramado - RS
Porto Alegre - RS
Bagé - RS
Pelotas - RS
24 h
1.809 km
Erechim
Palmeira das Missões
Lages
Lagoa Vermelha
Passo Fundo
Panambi
Santo Ângelo
Ijuí
Carazinho
Marau
Vacaria
São Luiz Gonzaga
São Borja
Cruz Alta
Soledade
Bento Gonçalves
Caxias do Sul
Lajeado
Novo Hamburgo
Sapiranga
Osório
Rosário do Sul
São Gabriel
Antigas
Rivera
Dom Pedrito
Camaquã
São Lourenço do Sul
Tacuarembó
Rio Grande

O Problema do Caixeiro Viajante

- Consiste num vendedor que necessita visitar cada uma das *n* cidades uma única vez, iniciando em uma cidade e retornando ao local de partida. A questão é encontrar a rota que minimize a distância total a ser percorrida pelo vendedor. [Dantzig et al. 1954, Lin 1965].

O Problema do Caixeiro Viajante

- Dada uma matriz de custo $D = (d_{ij})$, onde $d_{i,j}$ = custo de ir da cidade i para a cidade j, encontrar uma permutação $P = (i_1, i_2, i_3, \cdots, i_n)$ dos inteiros de 1 até n que minimize a quantidade $d_{i1i2} + d_{i2i3} + \cdots + d_{ini1}$. [Lin 1965]

Descrição dos Algoritmos

- Método de busca:
  - Baseado na média aritmética do custo das ligações entre cidades não visitadas;
  - Calculada a média a partir de cada cidade em que o vendedor se encontra;
  - Cidades não visitadas são separadas em dois vetores: cidades custo maior que média; cidades custo menor que média;
  - Limite de interações;
  - Número de interações reiniciado a cada nova melhor rota encontrada.

Descrição dos Algoritmos

- Algoritmo sequencial:
  - Cria uma solução inicial aleatória;
  - Preenche vetor de cidades não visitadas e define aleatoriamente cidade origem;
  - Calcula a média aritmética e separa as cidades não visitadas nos dois vetores;
  - A cada interação, seleciona aleatoriamente uma cidade contida no vetor de ligações com custo abaixo da média;
  - Caso esteja vazio, seleciona do vetor de custo acima da média, priorizando as ligações com menor custo.

Descrição dos Algoritmos

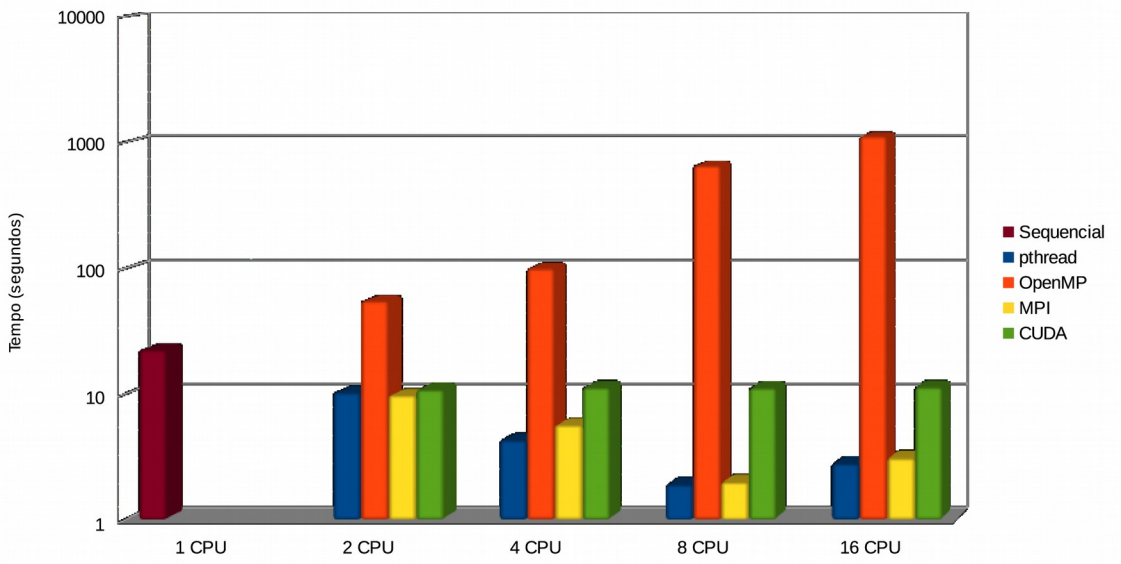
- Algoritmos paralelos:
  - Com Pthread, cada *thread* executa os mesmo passos do algoritmo sequencial, porém, em determinado intervalo de cidades;
    - Ao final, as rotas parciais de cada *thread* são unidas;
  - OpenMP utiliza a mesma estrutura do algoritmo sequencial, mas os laços *for* são executados em paralelo;
  - Com MPI, cada *threads* executa os mesmo passos do algoritmo sequencial, assim como com Pthread;
    - Também retornam rotas parciais que são unidas;
  - Com CUDA, cada *thread* executa os mesmos passos do algoritmo sequencial.

Ambiente de Testes

- Plataformas de Experimentação:
  - Desenvolvidos na liguagem C++;
  - Testes realizados numa máquina com:
    - Processador Intel Core i7 Ivy Bridge 3630QM, octa-core;
    - 8 GB memória RAM;
    - Placa gráfica Nvidia GeForce GT 640M com 1 GB de memória;
    - Sistema operacional Linux Elementary OS kernel 3.13.5.
  - Número de interações: 100.000;
  - Número de *Threads*: 2, 4, 8 e 16;
  - Testes realizados 10 vezes com cada algoritmo em cada configuração de *threads,* com entrada de 58 cidades.

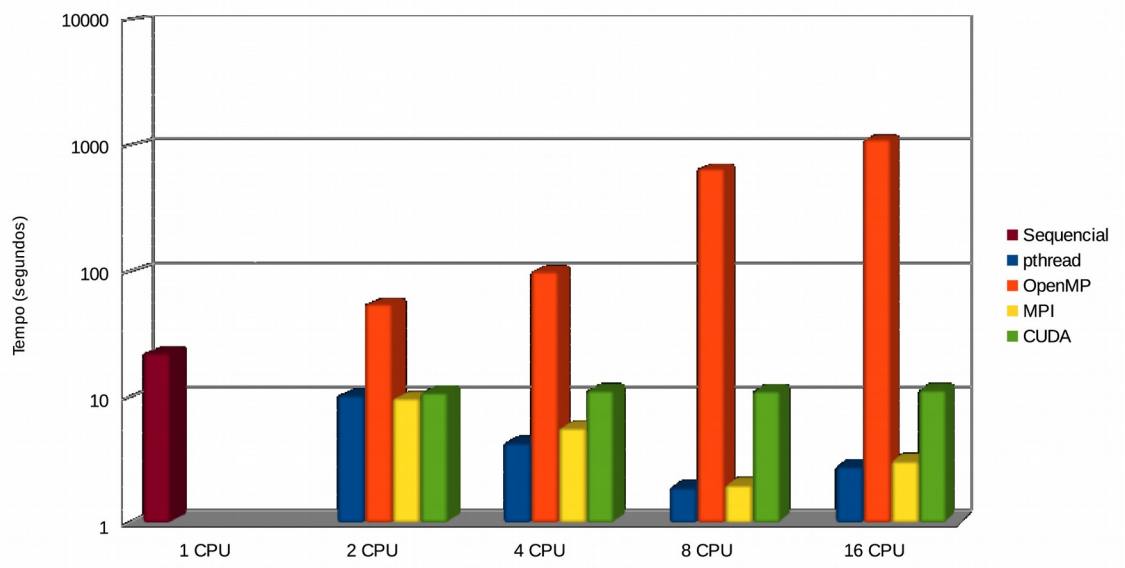
Resultados

- Pthread: com a divisão do problema em partes menores, os resultados foram satisfatórios;
- OpenMP: possui variáveis compartilhadas, necessita controle de concorrência entre as *threads*, reduzindo o desempenho;
  - A paralelização apenas dos laços *for* e a falta de otimização são causas do baixo desempenho.



Resultados

- MPI: resultados próximos aos obtidos com Pthread, pela divisão do problema e comunicação por mensagens;
- CUDA: resultados permaneceram idênticos, motivado pelo tamanho reduzido do problema e poucas *threads*.



Conclusão e Trabalhos Futuros

- Problema da minimização do tempo computacional;
- Utilizamos a paralelização para reduzir o tempo;
- Pthread e MPI obtiveram melhor desempenho;
- Tamanho reduzido do problema e poucas *threads*, além da falta de otimização, comprometeram o desempenho com CUDA;
- Com OpenMP, pela falta de otimização e dificuldades enfrentadas, obteve pior resultado;
- Um trabalho futuro, seria a otimização do algoritmo OpenMP;
- Outro trabalho seria realizar testes com entradas de diferentes tamanhos, e número maior de threads, onde CUDA possivelmente teria um desempenho melhor.

Bibliografia

- Dantzig, G., Fulkerson, R., and Johnson, S. (1954). Solution of a large-scale traveling-salesman problem. *Journal of the operations research society of America,* 2(4):393–410.
- Lin, S. (1965). Computer solutions of the traveling salesman problem. *Bell System Technical Journal, The,* 44(10):2245–2269.

Obrigado pela atenção!

Análise de Desempenho da Paralelização do Problema de Caixeiro Viajante

Contatos:

{gabriel.freytag
guilherme.arruda
rogerio.martins
padoin}@unijui.edu.br

Universidade Regional do Noroeste do RS (UNIJUÍ)