UNIVERSIDADE DE SÃO PAULO
INSTITUTO DE CIÊNCIAS MATEMÁTICAS E DE COMPUTAÇÃO

Primeiro Trabalho Prático - Resolução em Grupo

Parte II - **Resultados**

SSC0903 - Computação de Alto Desempenho
Prof. Dr. Paulo Sergio Lopes de Souza
2º semestre - 2022 - **Turma A**

Grupo: **TA02**

Nomes dos integrantes deste grupo que resolveram o trabalho

Nome: Antonio Rodrigues Rigolino NUSP: 11795791
Nome: Gustavo Henrique Brunelli NUSP: 11801053
Nome: João Guilherme Jarochinski Marinho NUSP: 10698193
Nome: Matheus Henrique de Cerqueira Pinto NUSP: 11911104

---

1 Sobre os códigos

Ambos os códigos adotam a estratégia de força bruta requisitada, otimizada através da enumeração de todas as permutações possíveis através dos vértices, excluindo-se o vértice inicial. Em comum, os dois códigos contêm as funções que lidam com o grafo gerado e, também, a função de `next_permutation`, responsável por, dado um conjunto inicial, calcular a próxima permutação possível.

1.1 Sequencial

O código sequencial adota uma única função para a resolução do problema do Caixeiro Viajante, tendo fixado o vértice inicial como 0 e, para as possíveis permutações geradas, excluindo o vértice inicial. Nessa estratégia, permutam-se os elementos, calcula-se o custo do caminho da permutação e, então, é somado o custo do vértice inicial até o primeiro elemento da permutação; da mesma forma, é adicionado o custo do vértice final da permutação até o vértice inicial fixado.

1.2 Paralelo

O código paralelo adota uma solução parecida à da solução sequencial para o problema. É fixado o vértice inicial como o 0 e, para todos os casos, ele é excluído para as permutações a serem realizadas. Na estratégia paralela, além disso, são enumerados os vértices acessíveis em processos e fixados também para as permutações. Assim, para cada tarefa diferente, permutam-se os elementos excluindo os fixados e, então, são somados os custos do vértice inicial até o vértice fixado, do vértice fixado até o primeiro vértice da permutação e, por fim, do vértice final da permutação até o vértice inicial.

2 Resultados

Todos os resultados obtidos foram executados em apenas um dos nós do *cluster*, denominado `hal03`. O grupo tentou acesso aos demais, mas não obteve sucesso. Uma possível justificativa para isso pode ser construída sob o fato de que os testes foram realizados próximos à entrega e, consequentemente, vários nós estavam sendo muito usados.

2.1 Validade dos resultados

Foram variados o valor das cidades conforme o conjunto $V_N = \{9{,}10{,}11{,}12{,}13\}$ e, para cada uma das variações, foram executados e armazenados cinco vezes os resultados obtidos. Tanto no código sequencial quanto no paralelo, foram produzidos os mesmos caminhos ideais e, portanto, os mesmos custos.

Importante é notar que as execuções foram realizadas concomitantemente às de outros grupos no *cluster* disponibilizado para a disciplina e, portanto, ao analisar as tabelas construídas, levar esse fator em consideração. Foi possível executar, no entanto, sem problemas, mesmo considerando apenas um nó.

2.2 Tempos de execução, eficiência e *speedup*

A seguir, são apresentadas as medidas de tempo coletadas nas execuções e, em seguida, calculados eficiência e *speedup* para cada uma das variações presentes no conjunto V_N, previamente definido.

Tabela 1: Tempos durante as execuções - Sequencial

Execução	**Tamanho de N**				
	9	**10**	**11**	**12**	**13**
#1	0.0008	0.0071	0.1483	1.1751	11.2279
#2	0.0008	0.0071	0.1090	1.1263	12.5234
#3	0.0004	0.0071	0.1158	1.0539	12.5074
#4	0.0008	0.0070	0.1097	1.0724	12.4535
#5	0.0007	0.0070	0.1524	1.2011	9.3405
Médias					
	0.0007	0.0071	0.1270	1.1258	11.6105
Desvio-padrão					
	0.0001	0.00005	0.0192	0.0568	1.2367

Tabela 2: Tempos durante as execuções - Paralelo

Execução	**Tamanho de N**				
	9	**10**	**11**	**12**	**13**
#1	0.0029	0.0012	0.0185	0.2596	2.975
#2	0.0230	0.0018	0.0193	0.2571	3.0864
#3	0.0078	0.0011	0.0122	0.2415	3.074
#4	0.0079	0.0009	0.0245	0.2488	2.5088
#5	0.0069	0.0014	0.0185	0.2609	2.6808
Médias					
	0.0097	0.0013	0.0186	0.2536	2.8650
Desvio-padrão					
	0.0069	0.0003	0.0039	0.00736	0.2305

Tabela 3: Eficiência e *speedup*

Estatística	**Tamanho de N**				
	9	**10**	**11**	**12**	**13**
Eficiência	0.0080	0.5461	0.6207	0.3699	0.3117
Speedup	0.0722	5.4615	6.8279	4.4392	4.0525

3 Gráficos

Figura 1: Tempos de execução dos códigos (em escala logarítmica)

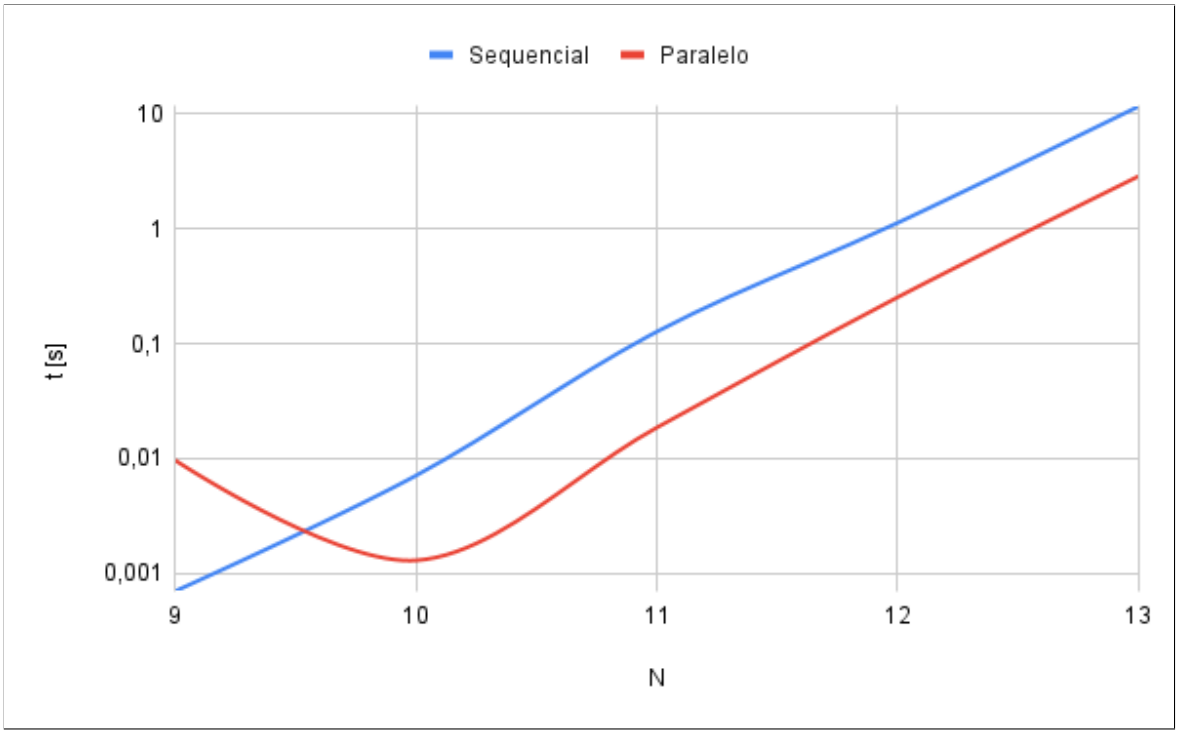


Figura 2: Eficiência nas execuções

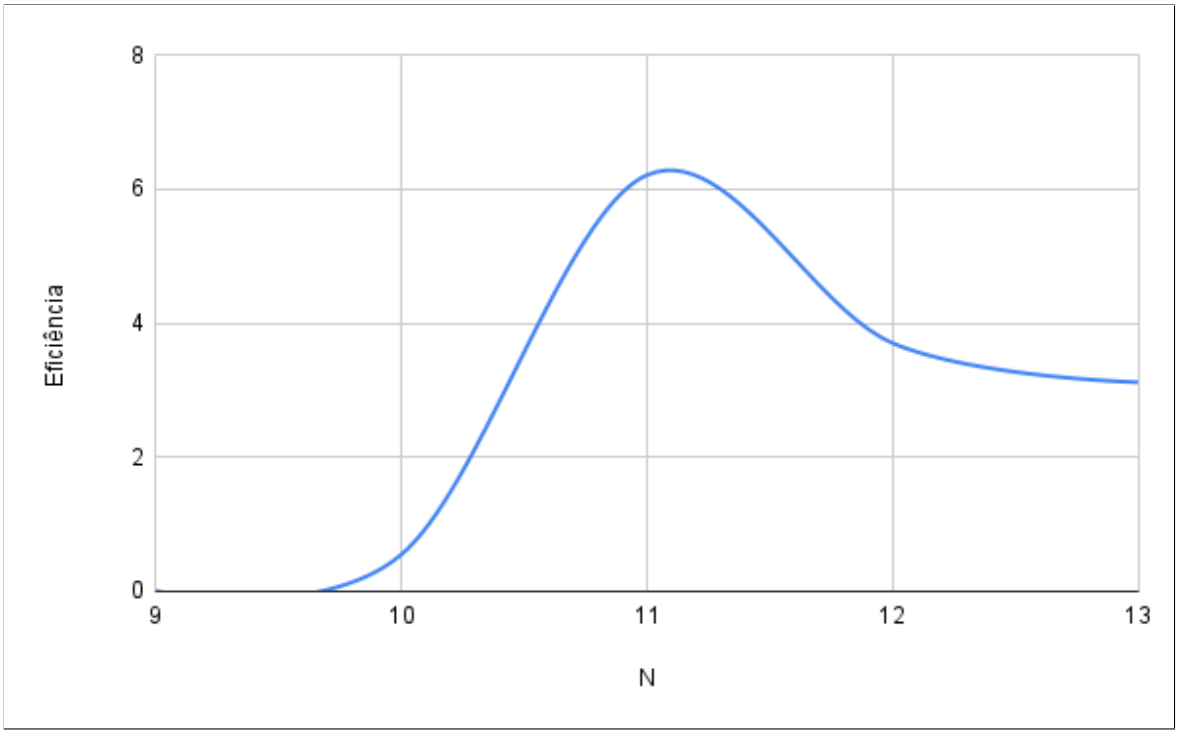
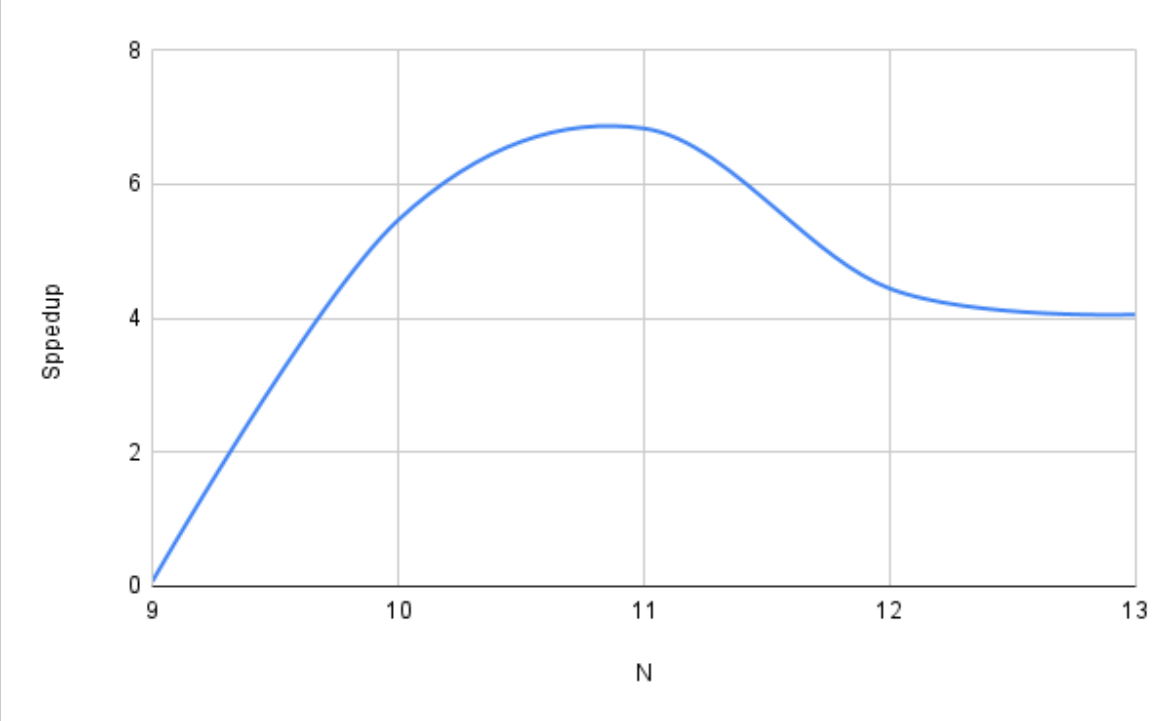


Figura 3: *Speedup* nas execuções



4 Conclusões

Conforme explicitado na seção de validade de resultados, é provável que a concorrência do *cluster* tenha afetado os resultados negativamente ao longo das cinco variações de N e cinco execuções. Isso, no entanto, não impediu a execução do código paralelo, demonstrando a versatilidade trazida pela biblioteca MPI. Importante é notar, também, a simplicidade e clareza nas instruções da biblioteca, o que facilitou a transposição do PCAM numa solução em código.

Ao longo das execuções, é notório que o código sequencial apresenta resultados melhores para valores de N baixos, o que é esperado, haja vista que a paralelização para valores pequenos de tarefas, considerando o que foi proposto no projeto, tende a ser mais custosa. Em contrapartida, a medida em que N aumenta, o código paralelo se torna melhor sob o ponto de vista do tempo de execução. Por fim, caso houvesse testes utilizando mais nós, era esperado que essa diferença se tornasse ainda maior.

As métricas analisadas, eficiência e *speedup* corroboram a afirmação anterior. Analisando os respectivos gráficos, traçados nas Figuras 2 e 3, é notório que ambos permanecem sempre acima de 0, mas que, a partir da execução considerando $N = 12$, ambos apresentam uma tendência de queda. Apesar dessa constatação, é notório, através das análises das tabelas, que houve melhoria, mesmo usando apenas um nó, o que atesta a utilidade da biblioteca e, também, das estratégias de paralelização.

Analisando o projeto como um todo, desde a construção do código sequencial, passando pela análise do problema e a ideia de paralelização com o PCAM, para,

enfim, construir o código paralelizado com MPI, é indiscutível sua importância para formação dos autores. Foram considerados conceitos discutidos desde o início da disciplina e, assim, colocados em prática num problema que exigiu a interpretação e até mesmo a busca por novos conceitos em prol da melhoria dos códigos.