Análise de Desempenho de Diferentes Implementações do Algoritmo Simulated Annealing

Arthur H. da Silva[1], **Gabriel A. Pereira**[2]

[1]Instituto de Informática – Universidade Federal do Rio Grande do Sul (UFRGS)
Caixa Postal 15.064 – 91.501-970 – Porto Alegre – RS – Brasil

{ahsilva,gapereira}@inf.ufrgs.br

Resumo. *Este trabalho apresenta uma análise de desempenho de três versões do algoritmo Simulated Annealing aplicadas a problemas de otimização combinatória. As versões testadas incluem: (i) implementação serial tradicional, (ii) cálculo da função de energia paralelizado utilizando OpenMP, e (iii) execução do algoritmo serial para múltiplas seeds em paralelo. O objetivo é comparar os tempos de execução e avaliar os ganhos de paralelismo em diferentes níveis de granularidade.*

1. Introdução

O Simulated Annealing (SA) é um algoritmo metaheurístico inspirado no processo de recozimento metálico, utilizado para resolver problemas de otimização combinatória e contínua. A técnica é baseada em explorar o espaço de soluções através de movimentos aleatórios, aceitando soluções piores com uma probabilidade que decresce ao longo do tempo, de modo a escapar de mínimos locais.

Embora o SA seja simples de implementar, problemas de grande escala podem exigir longos tempos de execução. Para contornar essa limitação, técnicas de paralelização podem ser aplicadas em diferentes níveis. Este trabalho compara três abordagens:

1. **Simulated Annealing Serial**: execução sequencial do algoritmo para uma seed específica.
2. **Energia Paralelizada**: execução do algoritmo serial, mas com cálculo da função de energia distribuído entre múltiplas threads via OpenMP.
3. **Seeds em Paralelo**: execução do algoritmo serial múltiplas vezes em paralelo, cada instância com uma seed distinta, permitindo avaliar diferentes soluções simultaneamente.

A comparação entre essas versões permite avaliar o impacto do paralelismo em diferentes granularidades, bem como a escalabilidade do algoritmo em ambientes multicore.

2. Simulated Annealing

O algoritmo *Simulated Annealing* (SA) é uma metaheurística probabilística utilizada para encontrar aproximações de soluções ótimas em problemas de otimização combinatória. Seu usa o processo de recozimento, em que um material é aquecido e resfriado lentamente para minimizar sua energia interna. O SA busca minimizar uma função de custo

ou energia por meio de perturbações sucessivas em uma solução candidata, aceitando transições que pioram o custo com uma certa probabilidade que diminui gradualmente ao longo do tempo.

A implementação utilizada neste trabalho baseia-se no algoritmo apresentado em *CP-Algorithms* [e-maxx-eng contributors 2024], disponível em: `https://cp-algorithms.com/num_methods/simulated_annealing.html`.

3. Metodologia

A metodologia adotada neste trabalho consistiu em desenvolver, executar e comparar três versões do algoritmo *Simulated Annealing*. As implementações foram realizadas em C++ utilizando a biblioteca OpenMP para paralelização e seguem a estrutura geral mostrada nos arquivos disponíveis em `https://github.com/arthurhen00/INF01008-programacao-distribuida-paralela/tree/main/OpenMP`.

1. **Versão Serial**: Nesta versão, o algoritmo é executado de forma totalmente sequencial. Uma única instância da classe `state` é criada com base em uma seed específica, e o laço principal do `Simulated Annealing` é executado até o resfriamento completo. Essa versão serve como linha de base para comparação de desempenho, produzindo tempos e valores de energia de referência. `(Utililiza main.cpp e state.cpp)`
2. **Energia Paralelizada**: Nesta abordagem, o foco da paralelização está dentro da função de energia da solução. Utilizando *OpenMP*, o cálculo da função de custo é distribuído entre múltiplas threads, permitindo reduzir o tempo gasto na avaliação de cada solução candidata. Essa estratégia explora o paralelismo de dados, mantendo o fluxo do algoritmo inalterado. `(Utililiza main.cpp e state_parallel.cpp)`
3. **Seeds em Paralelo**: A terceira abordagem executa múltiplas instâncias independentes do algoritmo *Simulated Annealing* em paralelo, cada uma iniciada com uma seed diferente. Essa variação permite que várias execuções concorram por soluções potencialmente melhores. `(Utililiza main_seed_parallel.cpp e state.cpp)`

Todas as execuções foram realizadas em uma mesma máquina, variando o número de threads (`1, 3, 5, 10, 20`), o tamanho do conjunto de entrada (`100, 1000, 10000, 100000`), temperatura inicial 9000000, taxa de caimento 0.999 e seed 919.

4. Resultados

Esta seção apresenta os resultados experimentais obtidos com as três versões do algoritmo *Simulated Annealing* (SA): a versão serial, a versão com cálculo de energia paralelizado e a versão que executa múltiplas instâncias em paralelo (uma por seed). O objetivo é avaliar o ganho de desempenho (speedup) e o impacto do número de threads sobre o tempo de execução para diferentes tamanhos de entrada.

Os experimentos foram realizados com entradas contendo 100, 1000, 10000 e 100000 pontos, variando o número de threads entre 1, 3, 5, 10 e 20. Todos os testes utilizaram a seed 919 como base, com variações pseudoaleatórias adicionais nas execuções paralelas para a versão serial com variação de seed.

4.1. Versão Serial

A Tabela 1 apresenta os resultados obtidos pela versão sequencial do algoritmo, servindo como referência para o cálculo de speedup das versões paralelas. Observa-se o aumento quase linear do tempo de execução conforme o tamanho da entrada cresce, o que é esperado, dado que o custo da função de energia cresce proporcionalmente ao número de pontos avaliados.

Tabela 1. Resultados de execução do algoritmo Simulated Annealing serial e speedup (baseado na serial).

Input size	Best Energy	Execution Time (ms)	Speedup
100	4747	66.77	1.0
1000	505636	405.99	1.0
10000	5.19863e+07	4774.71	1.0
100000	5.19617e+09	48099.6	1.0

4.2. Versão Paralela com Cálculo de Energia Distribuído

Na segunda versão, o cálculo da função de energia foi paralelizado utilizando a biblioteca OpenMP. Os resultados estão apresentados na Tabela 2, que também inclui o speedup em relação à versão serial. Observa-se que, para tamanhos de entrada maiores, o speedup aumenta significativamente, atingindo até cerca de $9.6\times$ com 20 threads e entrada de 100.000 pontos. Para entradas pequenas, o overhead da paralelização supera o ganho, resultando em desempenho inferior ao da versão sequencial.

4.3. Versão com Execuções Paralelas por Seed

A terceira versão executa múltiplas instâncias independentes do algoritmo, cada uma com uma seed distinta. Essa abordagem explora o paralelismo em nível de tarefa, mantendo a execução de cada instância em uma thread diferente. Os resultados estão dispostos na Tabela 3. Como esperado, o desempenho geral é semelhante ao da versão serial, já que cada instância roda isoladamente; no entanto, o ganho surge quando múltiplas soluções são exploradas simultaneamente, aumentando a chance de encontrar energias mínimas globais.

A Tabela 4 lista as seeds utilizadas em cada configuração de threads. Observa-se que, em todas as execuções, a seed 919 produziu as menores energias, o que sugere que ela é mais favorável às condições iniciais do problema de otimização proposto.

4.4. Análise Gráfica do Speedup

A Figura 1 apresenta o gráfico de speedup obtido para diferentes tamanhos de entrada e quantidades de threads. O comportamento demonstra que, a partir de um certo tamanho de entrada (acima de 10.000 pontos), o algoritmo paralelo apresenta ganhos expressivos, enquanto entradas menores não compensam o overhead da criação e sincronização de threads.

4.5. Análise Gráfica da eficiência

A Figura 2 apresenta o gráfico de eficiência obtido para diferentes tamanhos de entrada e quantidades de threads. O gráfico mostra o comportamento esperado para uma aplicação paralelizada, ganhos menores para problemas menores, e ganhos maiores à medida que o tamanho do problema aumenta.

Tabela 2. Resultados do algoritmo Simulated Annealing paralelo com cálculo de speedup em relação à versão serial.

Input size	Threads	Execution Time (ms)	Speedup
100	1	80.931	0.83
100	3	103.710	0.64
100	5	111.728	0.60
100	10	141.280	0.47
100	20	190.267	0.35
1000	1	412.749	0.98
1000	3	216.801	1.87
1000	5	180.848	2.25
1000	10	203.343	2.00
1000	20	254.522	1.59
10000	1	4628.03	1.03
10000	3	1691.78	2.82
10000	5	1159.46	4.12
10000	10	758.297	6.29
10000	20	609.917	7.83
100000	1	46484.6	1.03
100000	3	17129.5	2.81
100000	5	11931.1	3.91
100000	10	7607.41	6.33
100000	20	5000.6	9.62

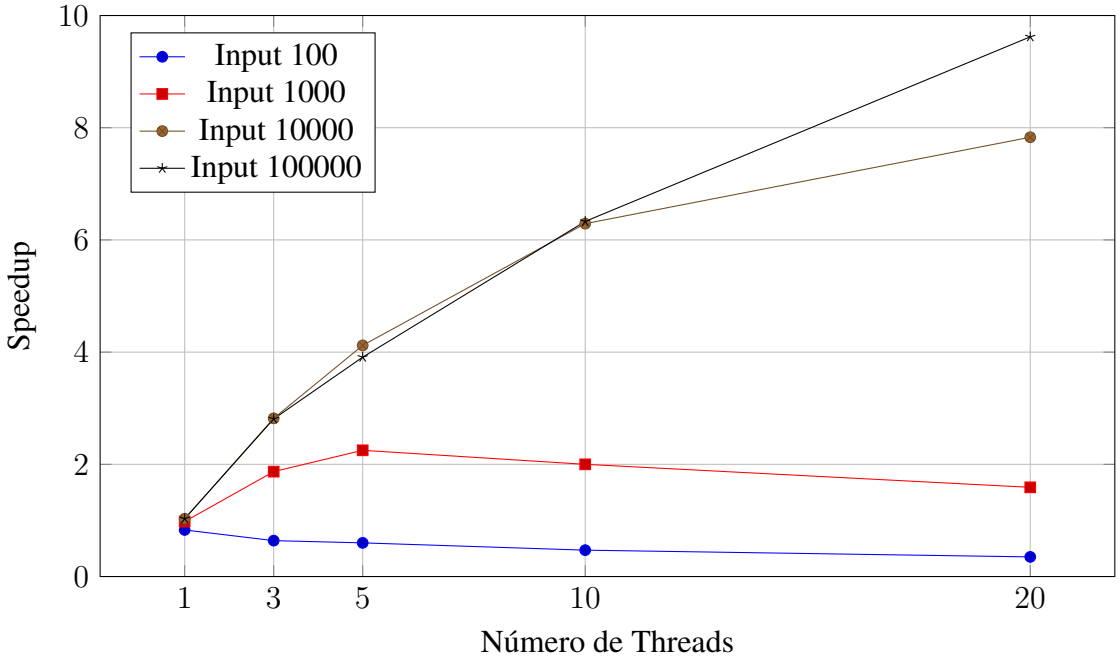

Figura 1. Speedup obtido em função do número de threads para diferentes tamanhos de entrada.

Tabela 3. Performance do algoritmo Simulated Annealing com 1 thread por seed.

Input Size	Threads	Average Execution Time (ms)	Speedup
100	1	67.50	1.00
100	3	67.68	0.997
100	5	70.95	0.951
100	10	77.05	0.876
100	20	77.54	0.871
1000	1	405.15	1.00
1000	3	409.28	0.990
1000	5	419.18	0.966
1000	10	470.28	0.862
1000	20	469.55	0.864
10000	1	4766.44	1.00
10000	3	4794.95	0.995
10000	5	4983.43	0.957
10000	10	5497.83	0.867
10000	20	5495.28	0.867
100000	1	48028.2	1.00
100000	3	48343.7	0.994
100000	5	49836.4	0.964
100000	10	55510.7	0.865
100000	20	55371.9	0.868

Tabela 4. Seeds utilizadas para cada número de threads e seed com melhor energia.

Threads	**Seeds Utilizadas**	**Seed com Best Energy**
1	919	919 (E=4747)
3	680, 919, 408	919 (E=4747)
5	341, 680, 408, 919, 811	919 (E=4747)
10	584, 811, 341, 800, 935, 680, 852, 408, 822, 919	919 (E=4747)
20	800, 919, 519, 178, 852, 842, 408, 235, 935, 680, 811, 535, 822, 584, 304, 797, 290, 385, 562, 341	919 (E=4747)

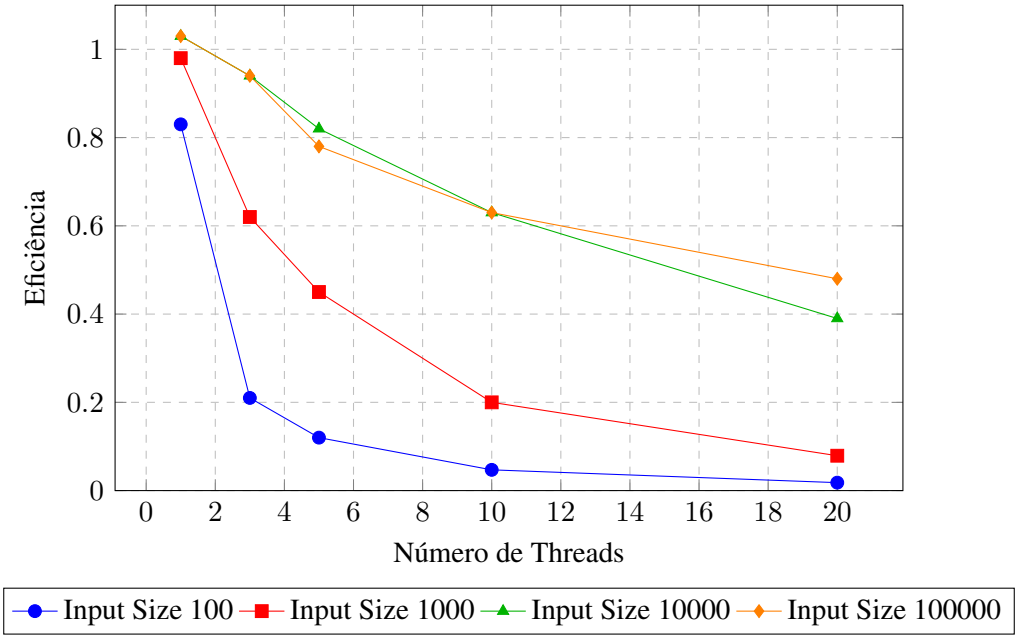

Figura 2. Eficiência do algoritmo Simulated Annealing paralelo em relação à versão serial.

5. Análise vtune

No geral, quanto maior o N, mais cache bound o algoritmo ficou (valores entre 20% e 30%), e quanto mais threads, mais memory bound (valores entre 20% e 30%).

Os números grandes de N implicarem em um algoritmo cache bound se dá, em partes, devido ao modo com que o próximo estado era decidido, já que um par de valores aleatórios tinha sua posição trocadas. Com N alto, os vetores são grandes e por isso é improvável que os dois estejam na cache ao mesmo tempo.

A quantidade de threads implicar em um algoritmo memory bound se dá, em partes, devido à troca de contexto nos escalonamentos e ao pouco trabalho que cada thread recebia.

As execuções com várias seeds não sofreram muito no quesito competição por memória RAM, em geral os valores de pipeline slots memory bound se mantiveram abaixo de 15%.

6. Conclusão

Este trabalho apresentou uma análise detalhada de desempenho de três abordagens distintas para o algoritmo *Simulated Annealing*, com foco na exploração de diferentes níveis de paralelismo: serial, paralelização do cálculo da energia e execução de múltiplas instâncias com seeds distintas.

Os resultados demonstraram que a paralelização do cálculo da função de energia proporcionou ganhos significativos de desempenho para entradas de maior dimensão. O speedup alcançou até $9{,}6\times$ com 20 threads e 100.000 pontos, evidenciando boa escalabilidade e aproveitamento eficiente dos recursos de hardware. No entanto, para entradas pequenas, o overhead de criação e sincronização de threads superou o benefício do paralelismo, resultando em desempenho inferior à versão serial.

A versão baseada em execuções paralelas por seed, mostrou-se útil em cenários de exploração de múltiplas soluções, mas não apresentou ganhos diretos de tempo, uma vez que cada instância opera de forma independente. Essa abordagem, amplia a diversidade das soluções encontradas e pode ser vantajosa em contextos onde a qualidade da solução é mais relevante que o tempo total de execução.

Referências

da Silva, A. H. and Pereira, G. (2025). Implementação do algoritmo simulated annealing com openmp. https://github.com/arthurhen00/INF01008-programacao-distribuida-paralela/tree/main/OpenMP. Repositório GitHub. Acesso em: out. 2025.

e-maxx-eng contributors (2024). Simulated annealing — cp-algorithms. https://cp-algorithms.com/num_methods/simulated_annealing.html. Acesso em: out. 2025.