Resultados da tabela 1 foram obtidos pela execução em um processador de 6 núcleos e 12 threads com o compilador gcc com a otimização -O2 e executado no WSL 2.

Tabela 1

Algoritmo	Nº Threads	Dimensões	Menor Tempo	Desempenho
Sequencial	1	500	0,1195	
Concorrente	1	500	0,5044	0,2370
Sequencial	2	500	0,1193	
Concorrente	2	500	0,2595	0,4599
Sequencial	1	1000	0,9677	
Concorrente	1	1000	4,1538	0,2330
Sequencial	2	1000	0,9656	
Concorrente	2	1000	2,0769	0,4649
Sequencial	1	2000	9,0922	
Concorrente	1	2000	33,7374	0,2695
Sequencial	2	2000	8,8648	
Concorrente	2	2000	17,2646	0,5135

Tabela 1: Dados das execuções do algoritmo implementado comparando em três execuções e registrando a menor tempo.

Em todas as execuções a implementação concorrente resultou em perda de desempenho.

Na implementação com 1 thread o aumento no tempo de execução mostra o gasto de tempo que criar, executar e retomar uma thread pode causar. Uma ressalva é que esse gasto pode estar exagerado devido à coleta de dados ter sido em WSL 2 e não num linux nativo.

Na implementação com 2 threads o aumento de desempenho mostra que o aumento de tempo na execução da parte sequencial pode ser compensado pelo processamento paralelo.

A tabela também revela a tendência do aumento de desempenho conforme a quantidade de threads permitidas. Com 6 threads o desempenho foi de 2,7; com 12 threads o desempenho foi de 4,2; e com 18 threads o desempenho foi de 3,9. O aumento se confirma até 12 threads.