Travelling Salesman Problem

Tema 3 SM

Dranca Ștefana-Ioana
Verna Dorian-Alexandru

Descrierea Problemei

- Un vânzător ambulant parcurge o serie de orașe
- Toate orașele trebuie să fie parcurse
- Vânzătorul trebuie să se întoarcă în orașul de unde a plecat
- Orice oraș are o legătură directă cu orice alt oraș
- Fiecare drum are un anumit cost
- Trebuie să găsim ordinea în care vor fi vizitate orașele astfel încât costul deplasării să fie minim

Implementarea secvențială

Am realizat două implementări de acest tip:

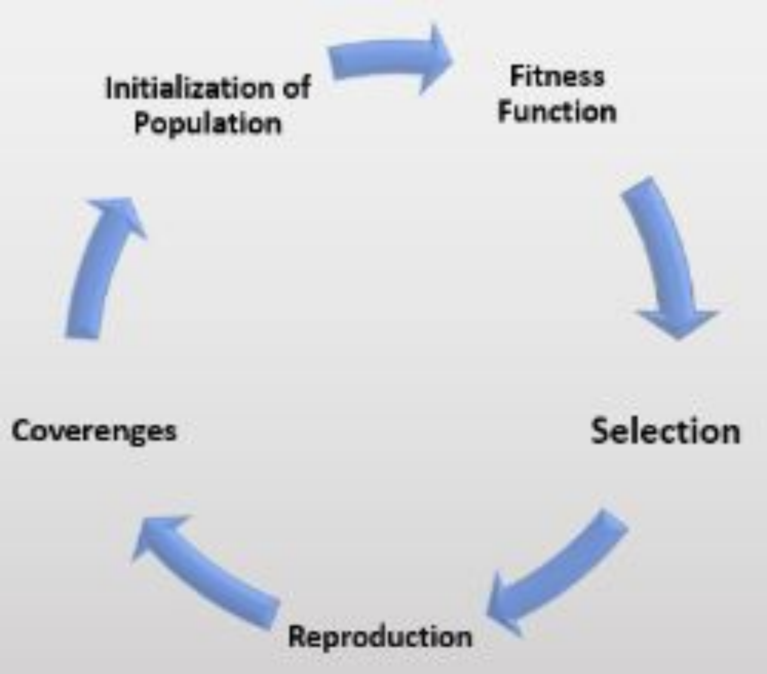
- Implementarea naivă (bazată pe backtracking) - nu este o soluție viabilă pentru problema noastră deoarece timpul de execuție este foarte mare pentru un input mai consistent, complexitatea soluției fiind O(n!)

- Implementarea bazată pe algoritm genetic - algoritmul se bazează pe aproximarea soluției în cadrul mai multor iterații. Scopul este de a obține o soluție din ce în ce mai bună pe măsură ce iterațiile avansează

Algoritm Genetic - Descriere

Noțiuni importante:

- Cromozom - un oraș
- Individ - o rută care include toate orașele (o posibilă soluție)
- Populație - o grupare de mai mulți indivizi sortați în funcție de fitness (din care primul individ reprezintă soluția problemei)
- Fitness - costul total în cazul unui individ



Algoritm Genetic - Varianta Secvențială

- Măsurătorile au fost făcute pe cluster-ul facultății.

- Am variat numărul de orașe

- Numărul de iterații (generații folosite) este egal cu 1000

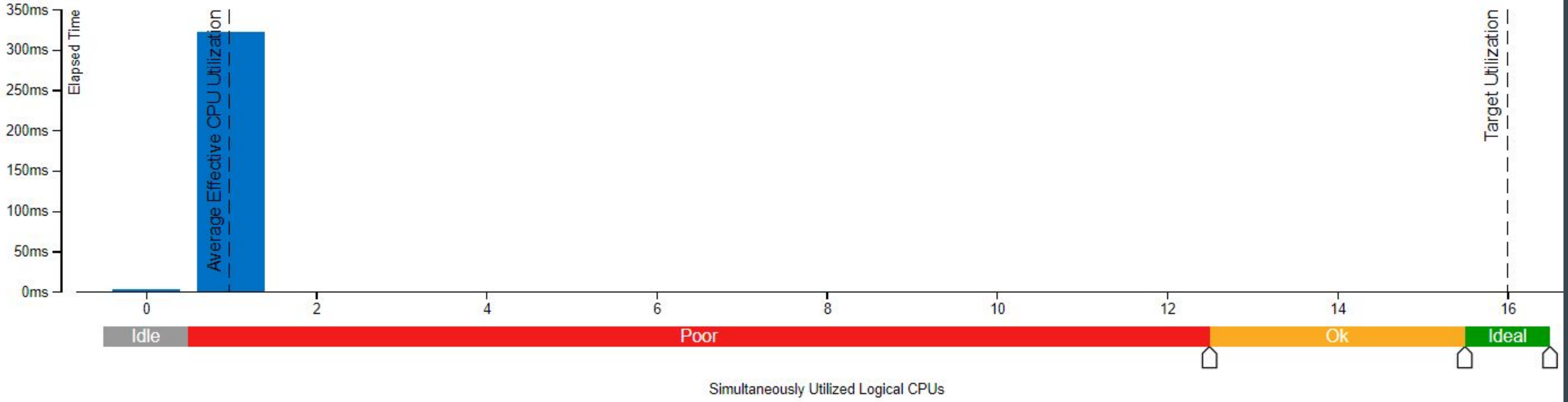
nr orase	secvential
	1 core
4	0.000746
5	0.000739
6	0.000925
10	0.001609
30	0.007023
100	0.05681
200	0.235715
500	2.198291
1000	13.712297
2000	108.096527

Algoritm Genetic - Varianta Secvențială

Effective CPU Utilization: 6.1% (0.983 out of 16 logical CPUs)

Effective CPU Utilization Histogram

This histogram displays a percentage of the wall time the specific number of CPUs were running simultaneously. Spin and Overhead time adds to the Idle CPU utilization value.



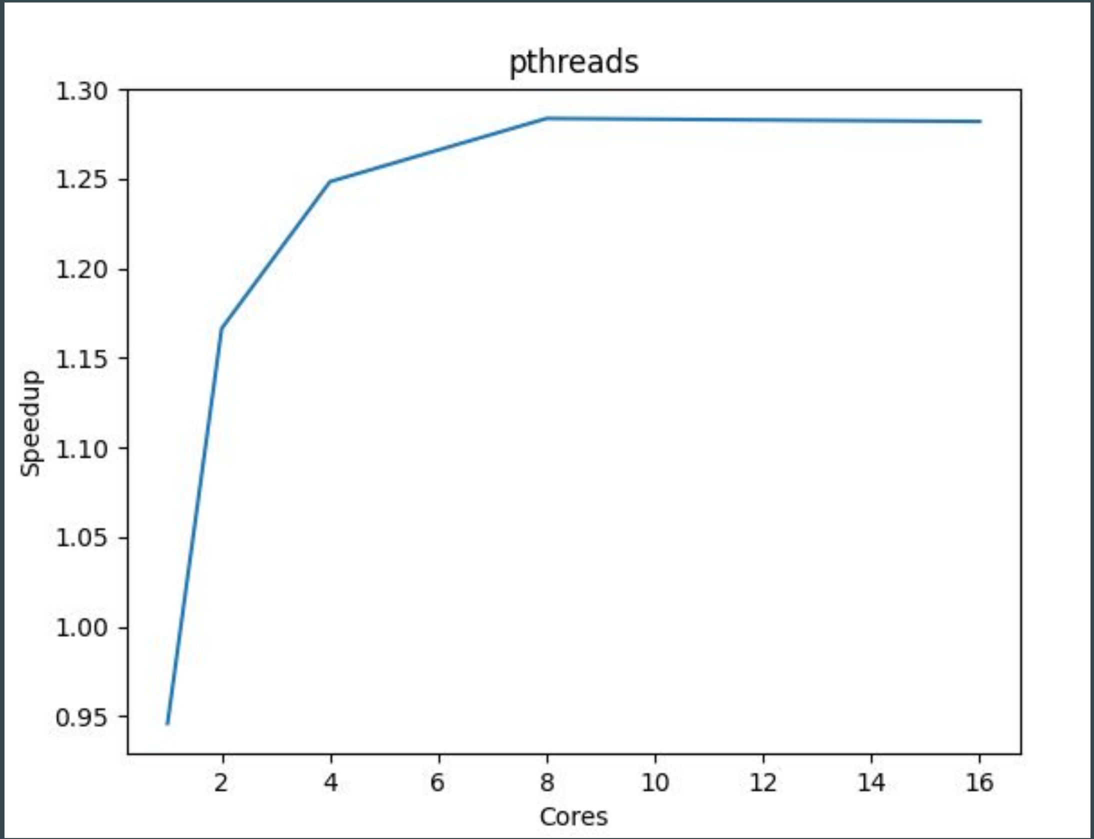
Total Thread Count: 1

Thread Oversubscription: 0s (0.0% of CPU Time)

Algoritm Genetic - Varianta Pthreads

nr orase	pthreads				
	1 cores	2 cores	4 cores	8 cores	16 cores
4	0.002192	0.022373	0.043098	0.077745	0.135649
5	0.002416	0.019862	0.034567	0.061488	0.112922
6	0.00248	0.020073	0.031598	0.056052	0.112088
10	0.003167	0.019262	0.029584	0.056537	0.111597
30	0.009196	0.023746	0.033403	0.060197	0.114865
100	0.061118	0.061133	0.064449	0.091429	0.142849
200	0.245516	0.194626	0.169031	0.191529	0.233031
500	2.200357	1.903924	1.570423	1.502781	1.447843
1000	13.771847	12.25699	10.82628	10.30033	10.124735
2000	114.271725	92.66697	86.58731	84.20769	84.318942

Algoritm Genetic - Varianta Pthreads



Algoritm Genetic - Varianta Pthreads

Top Hotspots

This section lists the most active functions in your application. Optimizing these hotspot functions typically results in improving overall application performance.

Function	Module	CPU Time	% of CPU Time
compute_generation_fitness_pthreads	main	0.240s	33.8%
__pthread_barrier_wait	libpthread.so.0	0.150s	21.1%
check_chromosome	main	0.112s	15.8%
func@0x18b644	libc.so.6	0.090s	12.7%
mutate_generation_openmp	main	0.040s	5.6%
[Others]	N/A*	0.078s	11.0%

*N/A is applied to non-summable metrics.

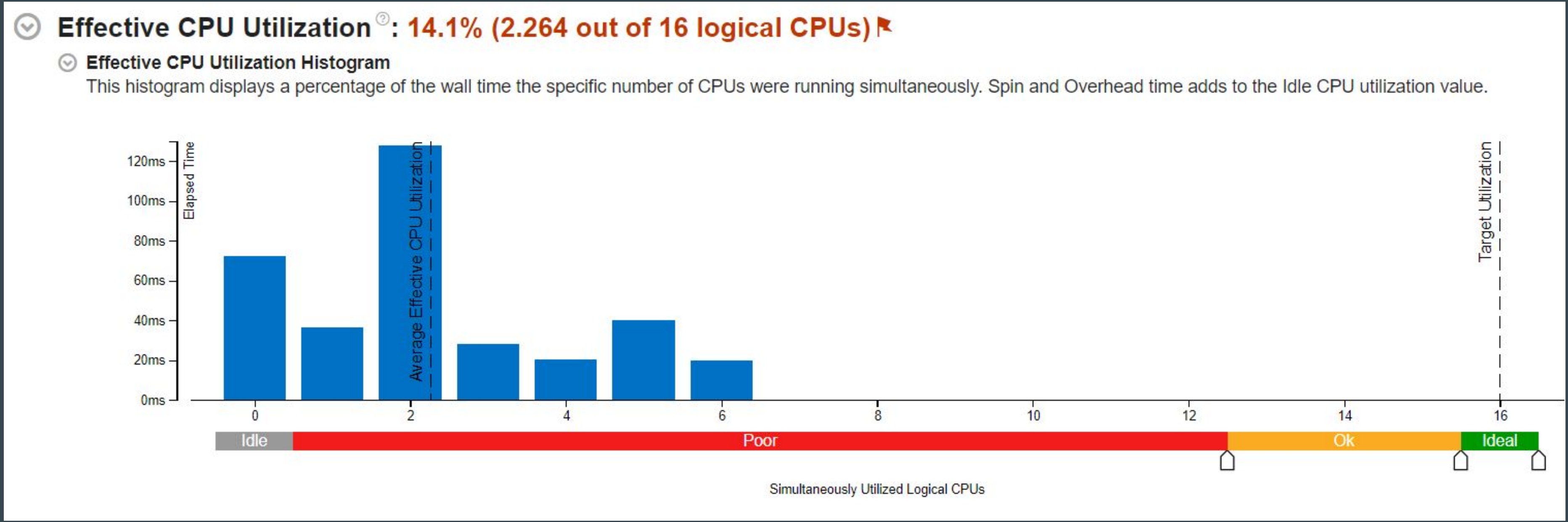
Top Waiting Objects

This section lists the objects that spent the most time waiting in your application. Objects can wait on specific calls, such as sleep() or I/O, or on contended synchronizations
significant amount of Wait time associated with a synchronization object reflects high contention for that object and, thus, reduced parallelism.

Sync Object	Wait Time with poor CPU Utilization	(% from Object Wait Time)	Wait Count
Barrier 0x7039cabd	3.414s	100.0%	64,032
Thread 0x5d58cb2a	0.283s	100.0%	16
Stream input/input8.in 0x26e083ab	0.001s	100.0%	1

*N/A is applied to non-summable metrics.

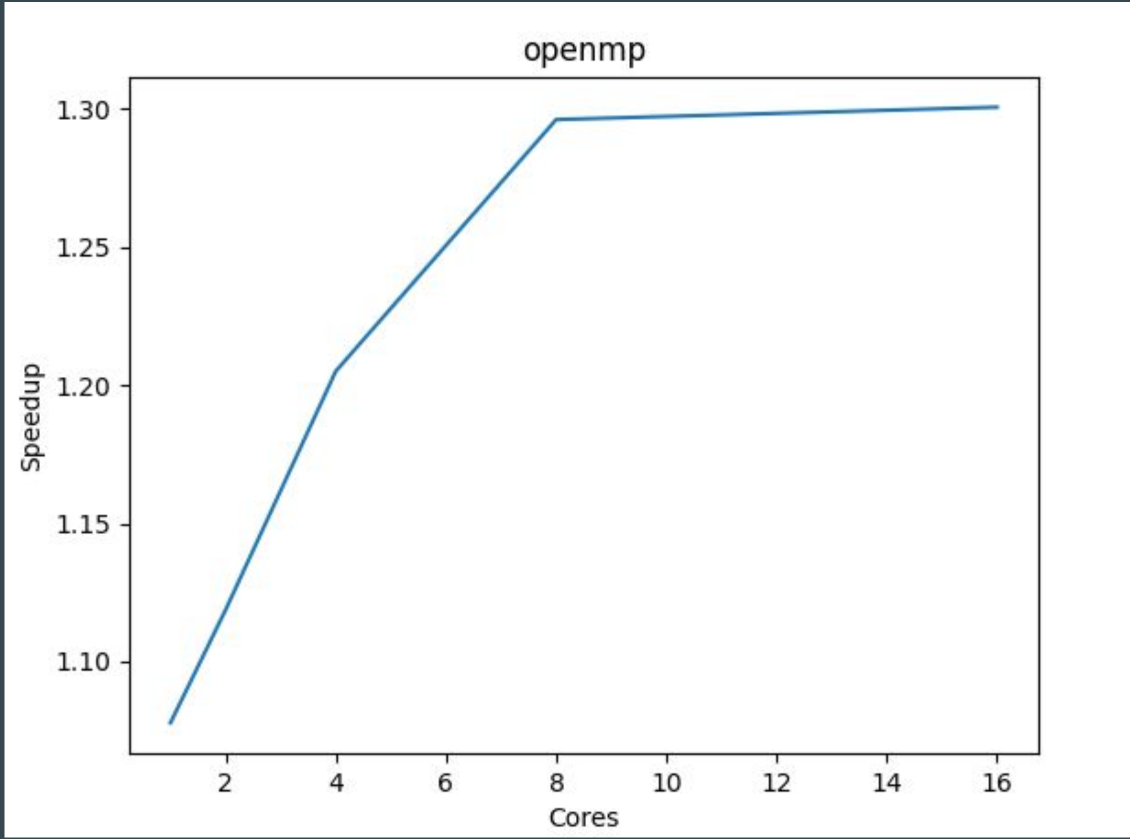
Algoritm Genetic - Varianta Pthreads



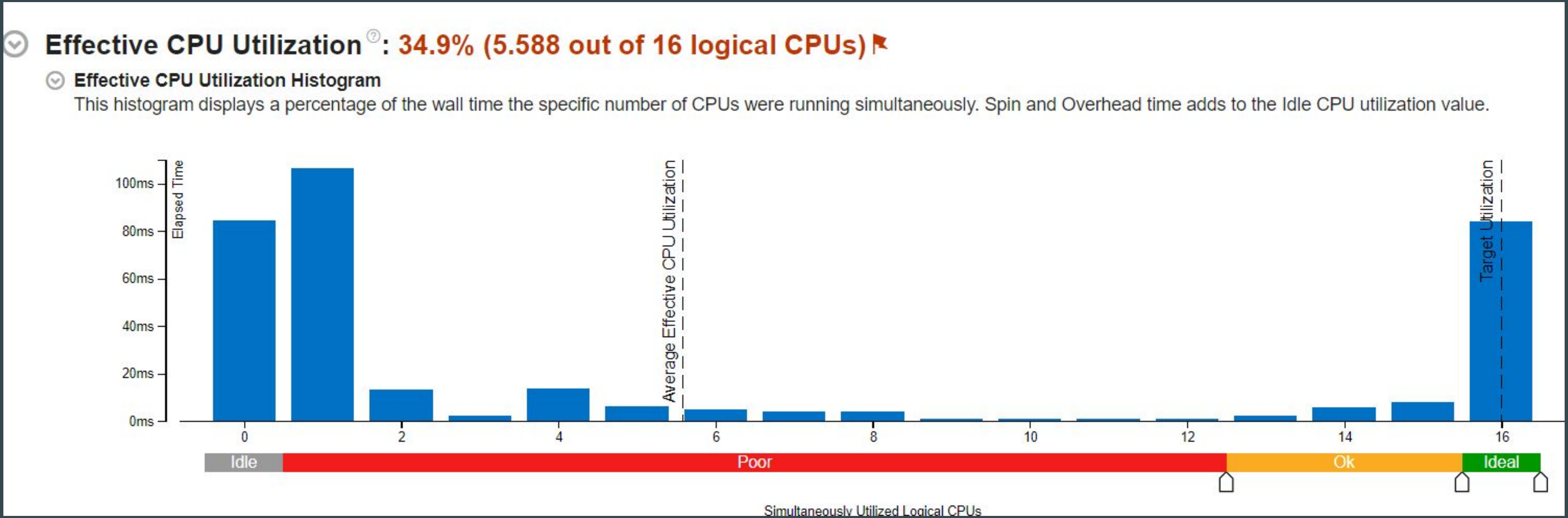
Algoritm Genetic - Varianta OpenMP

nr orase	openmp				
	1 cores	2 cores	4 cores	8 cores	16 cores
4	0.002518	0.00377	0.004738	0.006881	0.012095
5	0.002753	0.003821	0.003977	0.004868	0.007652
6	0.002882	0.0036	0.00373	0.004651	0.013083
10	0.003481	0.004305	0.004556	0.006751	0.008817
30	0.009179	0.009994	0.009396	0.010376	0.017803
100	0.05893	0.048844	0.039836	0.042679	0.043834
200	0.247654	0.177245	0.142324	0.126989	0.140743
500	2.193666	1.853424	1.502721	1.353176	1.347331
1000	13.683745	11.98996	10.72463	10.11278	10.297248
2000	100.287794	96.62906	89.70455	83.40377	83.116151

Algoritm Genetic - Varianta OpenMP



Algoritm Genetic - Varianta OpenMP



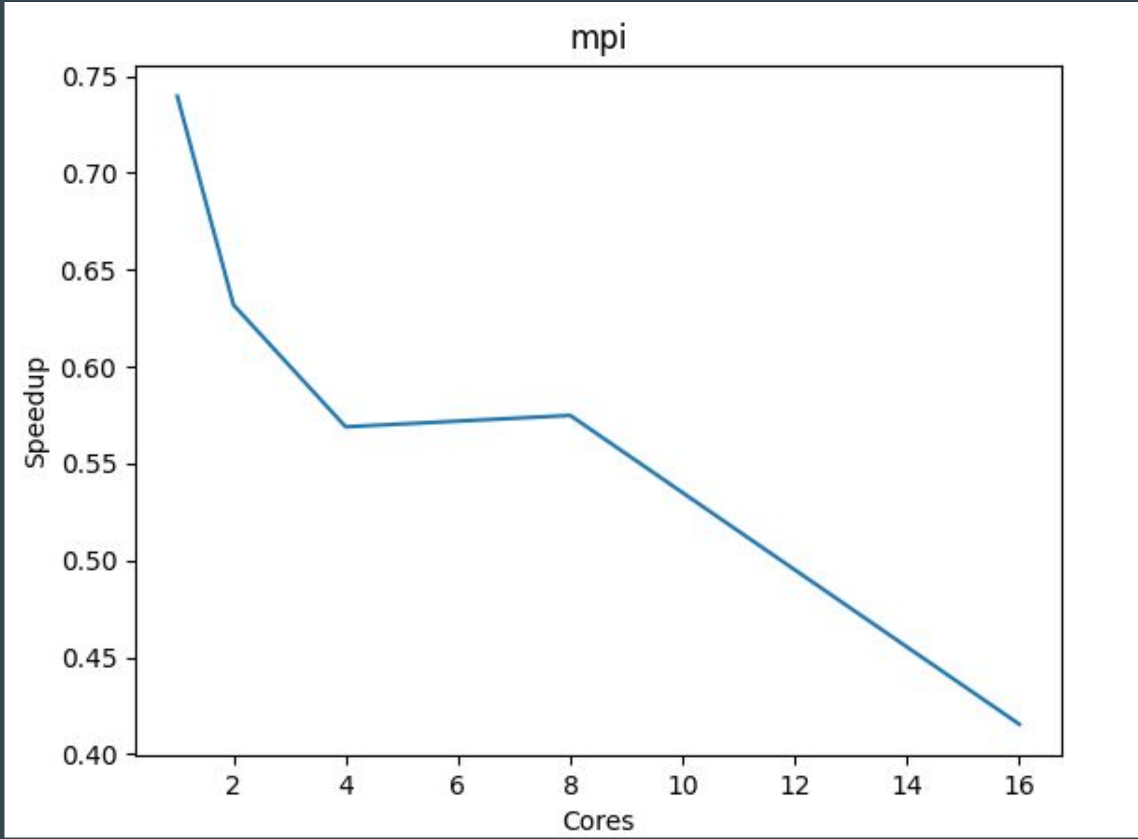
Algoritm Genetic - Varianta OpenMP

Function	CPU Time: Total			CPU
	Effective Time ▼	Spin Time	Overhead Time	Effective Time
clone	88.4%	0.0%	0.0%	0s
start_thread	88.4%	0.0%	0.0%	0s
func@0x1a654	88.4%	0.0%	0.0%	0s
TSP_parallel_openmp._omp_fn.0	84.0%	0.0%	0.0%	0s
func@0x1d294	45.2%	0.0%	0.0%	0.858s
GOMP_parallel	24.9%	0.0%	0.0%	0.041s
compute_generation_fitness_open	18.2%	0.0%	0.0%	0.337s
mutate_generation_openmp	16.6%	0.0%	0.0%	0.036s
func@0x18b644	14.7%	0.0%	0.0%	0.280s
_start	11.6%	0.0%	0.0%	0s
__libc_start_main	11.6%	0.0%	0.0%	0s
TSP_parallel_openmp	11.6%	0.0%	0.0%	0s
main	11.6%	0.0%	0.0%	0s
func@0x1d0f4	9.7%	0.0%	0.0%	0.184s
memcpy	5.5%	0.0%	0.0%	0s
generate_random_chromosomes	5.3%	0.0%	0.0%	0s
memcpy	4.2%	0.0%	0.0%	0s
check_chromosome	3.8%	0.0%	0.0%	0s
check_chromosome	3.8%	0.0%	0.0%	0.072s
memcpy	3.4%	0.0%	0.0%	0s
rand	2.6%	0.0%	0.0%	0.050s
memcpy	1.7%	0.0%	0.0%	0s
generate_random_numbers	1.6%	0.0%	0.0%	0s
__libc_malloc	0.5%	0.0%	0.0%	0.010s
qsort_r	0.4%	0.0%	0.0%	0.008s

Algoritm Genetic - Varianta MPI

nr orase	mpi				
	1 cores	2 cores	4 cores	8 cores	16 cores
4	0.00393	0.080493	0.171593	0.226174	0.295138
5	0.004271	0.091973	0.198002	0.273049	0.368081
6	0.004825	0.103704	0.222666	0.326806	0.433796
10	0.007126	0.171691	0.390838	0.506275	0.6391
30	0.021465	0.463044	1.010709	1.458132	1.82956
100	0.118927	1.403667	3.525586	4.61524	6.398606
200	0.415511	3.372182	6.779008	9.419304	12.865109
500	3.313864	9.491189	16.14473	22.88896	34.124191
1000	19.193256	28.79443	41.83028	52.6319	80.315373
2000	146.17503	171.0729	190.0014	188.0525	260.157001

Algoritm Genetic - Varianta MPI



Algoritm Genetic - Varianta MPI

Top Hotspots

This section lists the most active functions in your application. Optimizing these hotspot functions typically results in improving overall application performance.

Function	Module	CPU Time	% of CPU Time
sendmsg	libpthread.so.0	0.920s	21.2%
getdelim	libc.so.6	0.862s	19.9%
send	libpthread.so.0	0.636s	14.7%
ofi_bsock_recv	libtcp-fi.so	0.224s	5.2%
compute_generation_fitness_mpi	main	0.180s	4.1%
[Others]	N/A*	1.518s	35.0%

*N/A is applied to non-summable metrics.

Algoritm Genetic - Varianta MPI

Top Waiting Objects

This section lists the objects that spent the most time waiting in your application. Objects can wait on specific calls, such as sleep() or I/O, or on contended synchronizations. A significant amount of Wait time associated with a synchronization object reflects high contention for that object and, thus, reduced parallelism.

Sync Object	Wait Time with poor CPU Utilization	(% from Object Wait Time)	Wait Count
Condition Variable 0x0c5f0240	5.138s	100.0%	1
Sleep	0.446s	100.0%	1,093
Socket 0xfae1572b	0.040s	100.0%	1,027
Socket 0xac436205	0.001s	100.0%	4
Socket 0x6d4b6176	0.001s	100.0%	3
[Others]	0.003s	100.0%	80

**N/A is applied to non-summable metrics.*

Algoritm Genetic - Variantele Hibride MPI + OpenMP/Pthreads

- Pentru a efectua măsurătorile pentru variantele hibride, am folosit 2 procese MPI și, pentru fiecare dintre acestea, cate 4 thread-uri openmp, respectiv pthreads.

	mpi_pthreads	mpi_openmp
	8 cores	8 cores
4	0.154001	0.089801
5	0.166165	0.086649
6	0.209577	0.097199
10	0.177904	0.106566
30	0.133604	0.120222
100	0.447274	0.406023
200	0.78166	0.922855
500	3.461432	3.349992
1000	20.954332	20.15937
2000	257.3336	218.552851

Algoritm Genetic - Variantele Hibride MPI + OpenMP/Pthreads

Top Hotspots

This section lists the most active functions in your application. Optimizing these hotspot functions typically results in improving overall application performance.

Function	Module	CPU Time	% of CPU Time
func@0x1d0f4	libgomp.so.1	3.180s	40.0%
sendmsg	libpthread.so.0	1.576s	19.8%
getdelim	libc.so.6	0.868s	10.9%
ofi_bsock_recv	libtcp-fi.so	0.504s	6.3%
PMPI_Isend	libmpi.so.12	0.136s	1.7%
[Others]	N/A*	1.686s	21.2%

*N/A is applied to non-summable metrics.

Top Hotspots

This section lists the most active functions in your application. Optimizing these hotspot functions typically results in improving overall application performance.

Function	Module	CPU Time	% of CPU Time
sendmsg	libpthread.so.0	1.610s	31.6%
getdelim	libc.so.6	0.828s	16.3%
ofi_bsock_recv	libtcp-fi.so	0.510s	10.0%
compute_generation_fitness_mpi_pthreads	main	0.140s	2.8%
func@0xacbd4	libc.so.6	0.140s	2.8%
[Others]	N/A*	1.862s	36.6%

*N/A is applied to non-summable metrics.

Concluzii

- Cele mai bune variante de a paraleliza implementarea algoritmului genetic sunt cele de OpenMP și Pthreads

- Cu cât crește numărul orașelor luate în calcul, cu atât crește speedup-ul

- Variantele paralelizate sunt mai eficiente atunci când numărul de orașe este mai mare decât ~100

Concluzii

- În general, speedup-ul ajunge la un plafon în momentul în care trecem de 8 core-uri

- Varianta mpi și variantele hibride nu reprezintă niște soluții viabile, deoarece implică comunicarea multor informații între nodurile angajate în rularea programului, implicit existând și un overhead considerabil.

- Din rezultatele de la partea de profiling se pot observa părțile unde execuția este afectată de acest overhead