PARALLEL PARTICLE SWARM OPTIMIZATION ON A GRAPHICS PROCESSING UNIT WITH APPLICATION TO TRAJECTORY OPTIMIZATION

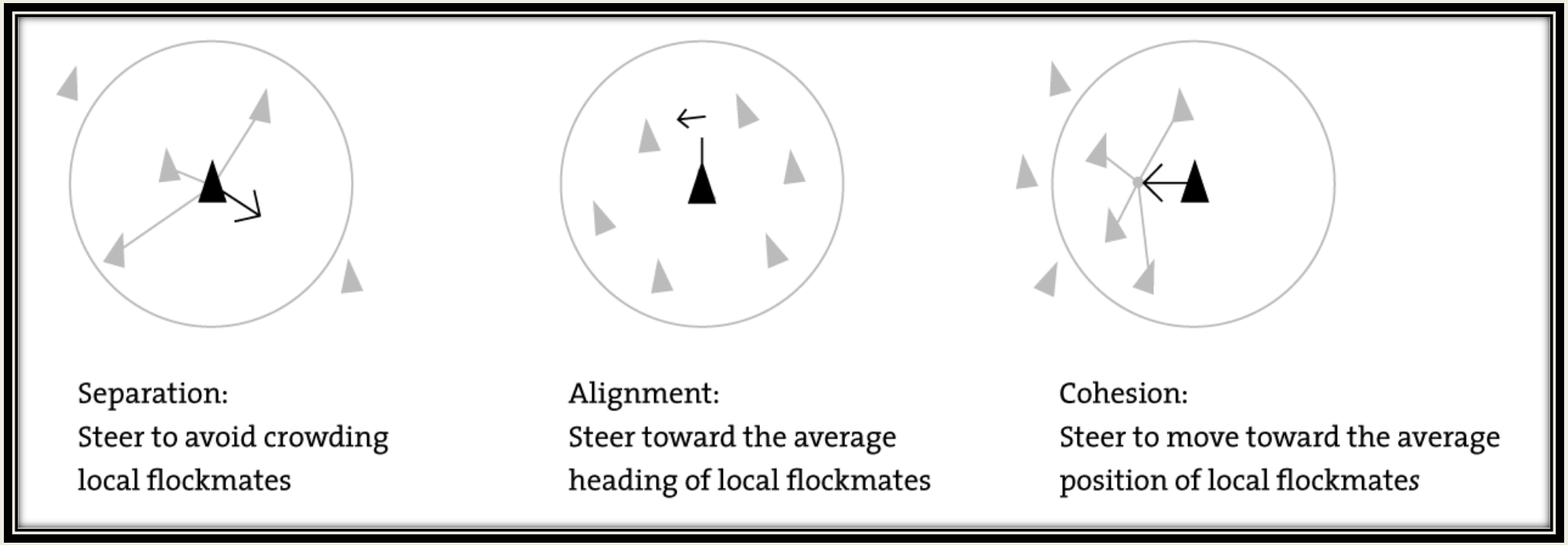
Harri Renney

Academic Goals[1]

1. To reduce computation time of PSO by processing on GPU.
2. Develop 3 implementations to compare.
3. Run tests on 4 generic algorithms + 1 practical example.
4. Compare performance between implementations.
5. Investigate how controllable parameters effect performance.

Particle Swarm Optimization

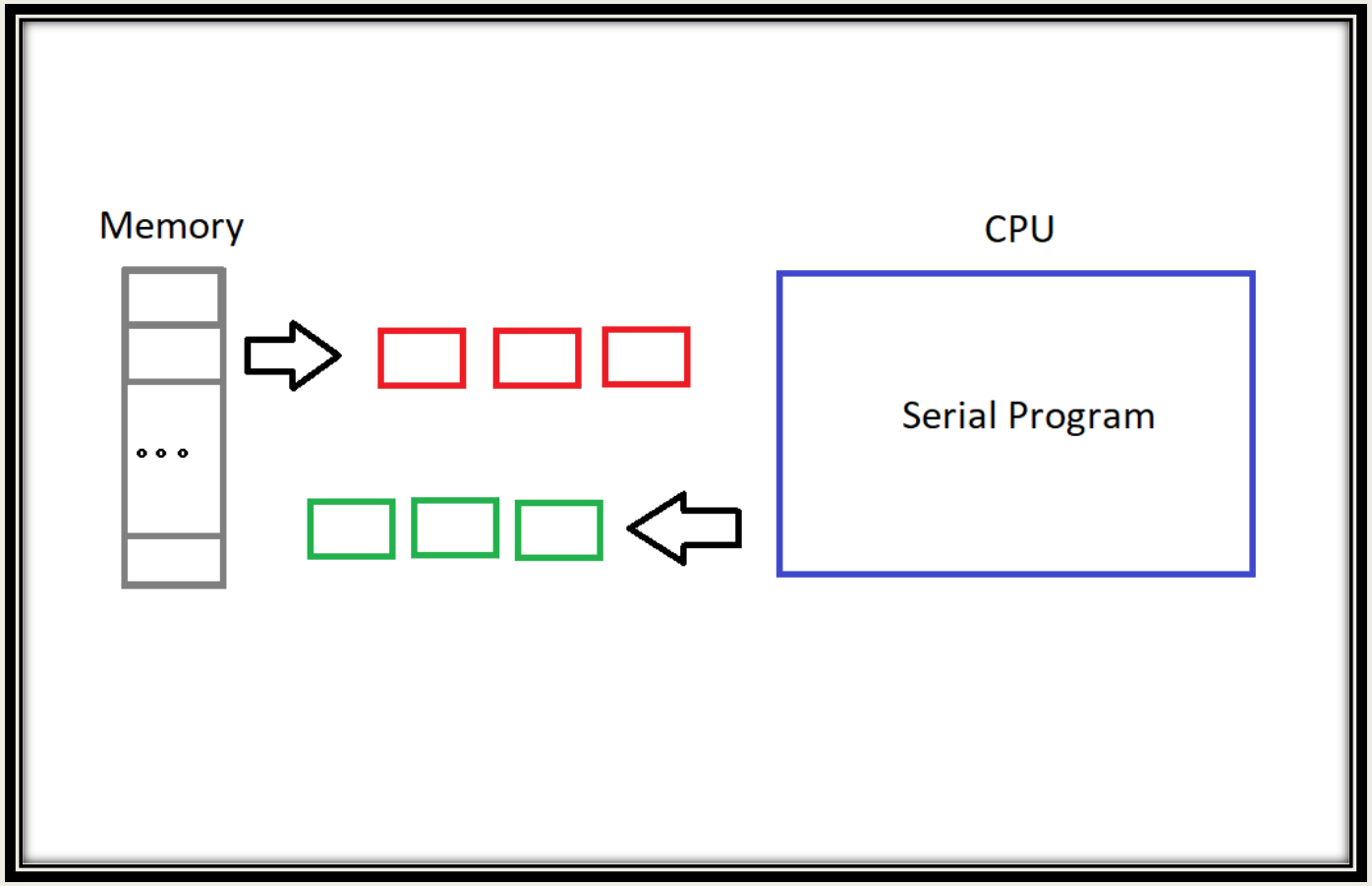
- Particle Swarm Optimization (PSO) is an AI algorithm inspired by bird flocking.
- Optimization algorithm. Concerned with finding a sufficiently optimal value.
- Each particle acts like individual agent.
- First implementation in "Boids" [2] application.



http://coffeepoweredmachine.com/wp-content/uploads/2013/08/boidsThreeRules.png

CPU Implementation – Serial Approach

- Considers each item of data at a time.
- Works well for many problems.
- Avoids memory sharing issues.



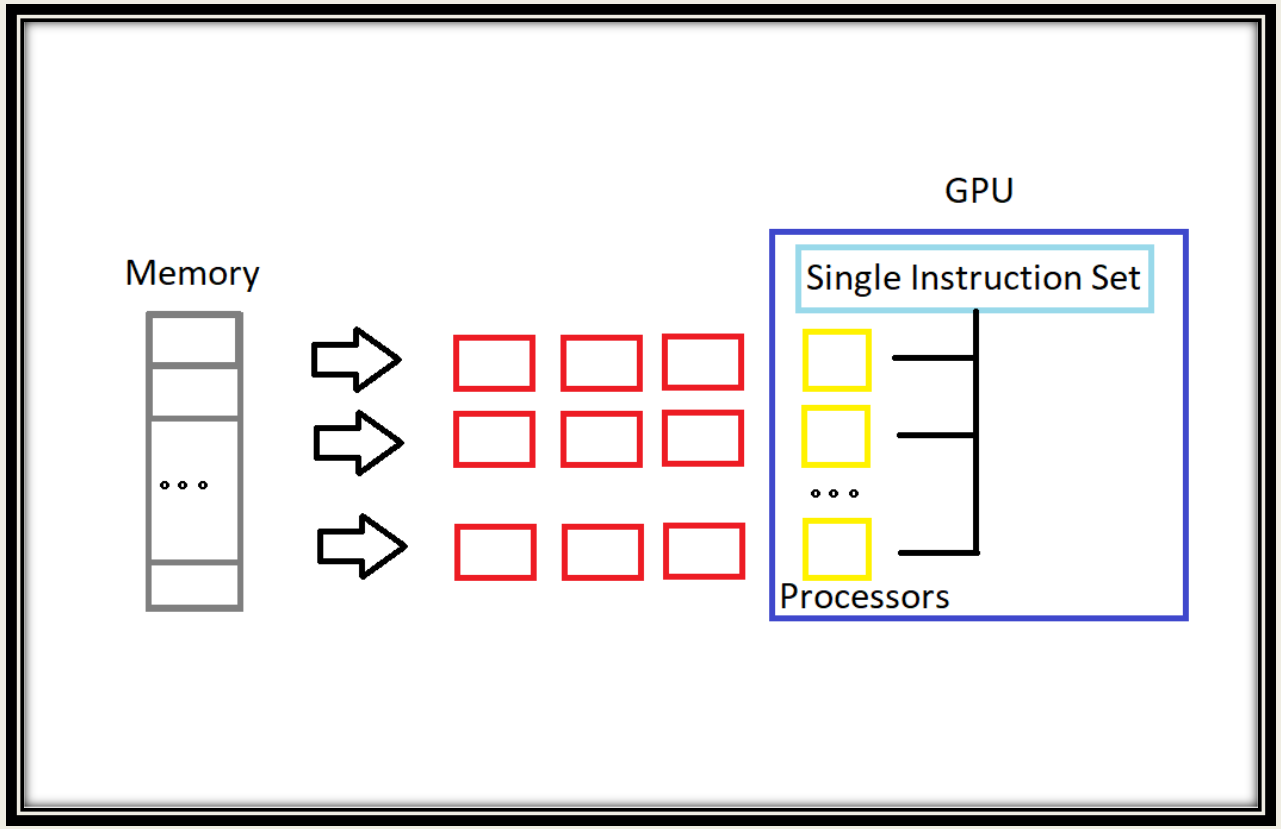
A naïve approach to PSO.

CPU Implementation – Possible Improvements

- Issue with this is paper is comparing serial SISD approach to parallel SIMD.
- Interesting, as the paper still achieves a few things.
- However, it would be really interesting to compare parallel CPU to GPU.
- CPU could use MIMD approach using OpenMP.
- Or could use SIMD using an intrinsic library, like libsimdpp.

SIMD Architecture

- Multiple data with same instructions applied.
- Can be processed parallel across multiple compute units.
- Conventional use in graphics.
- Designed to scale extremely well for large inputs.



GPU Implementation – SIMD Processing

- General Purpose Graphics Programming framework CUDA used.
- PSO fits SIMD architecture really well.
- Each particle independent, only needs to share local/neighbours information.
- Same process applied to all particles, just different input.

Possible alternatives to CUDA

- Another interesting method would be to use an alternative to CUDA.
- OpenCL is an obvious alternative.
- A true graphics API, like OpenGL. Could be done by mapping PSO into graphical concepts.
- This could work by using colour channels of fragments to hold particle information.

Partial and Full GPU Implementation

- Partial and Full GPU implementations developed.
- Partial only processes fitness calculation on GPU.
- Full processes initialization, calculation and updating on GPU.
- Demonstrates cost of overhead copying between CPU and GPU.

Methods Tested

- 4 different algorithm where run with the implementations and analysed.
- The algorithms are used to verify results. The algorithms have different levels of complexity.
- Sphere, Rosenbrock, Griewank, Ackley.

Table 1. Test functions.

Name	Function
Sphere	$f_1 = \sum_{i=1}^{d} x_i^2, -100 \le x_i \le 100$
Rosenbrock	$f_2 = \sum_{i=1}^{d} [x_i^2 - 10\cos(2\pi x_i + 100)], -10 \le x_i \le 10$
Griewank	$f_3 = \frac{1}{4000}\sum_{i=1}^{d} x_i^2 - \prod_{k=1}^{d} \cos(x_i/\sqrt{i}) + 1, -600 \le x_i \le 600$
Ackley	$f_4 = -20 * \exp(-0.2 * \text{sqrt}((1/d) * (\sum_{i=1}^{d} x_i^2))) - \exp((1/d) * (\sum_{i=1}^{d} \cos(2\pi x_i)))$ $+ \exp(1) + 20, -8 \le x_i \le 8$

Single Factor Analysis

- Single parameter, the number of particles changed.
- Speed-up increases with n.
- Speed-up increase does reduce slightly. N exceeds GPU threads.

- Speed-up not achieved at expense of accuracy.

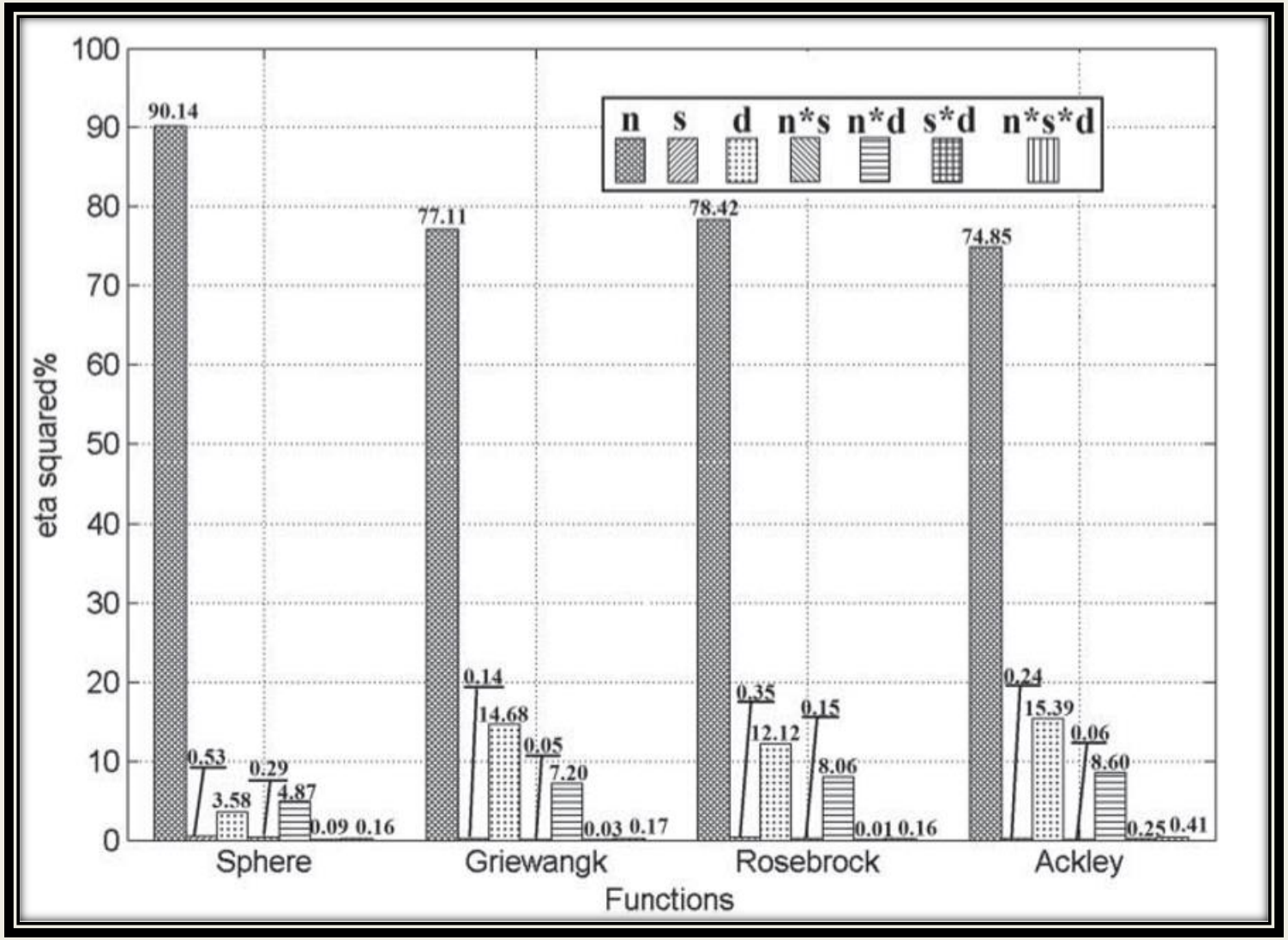
Table 3. Results of the Sphere function ($d = 50$, $s = 256$).

	f^*			t (s)				
n	CPU	pGPU	fGPU	CPU	pGPU	fGPU	sp^p	sp^f
1600	0.775	0.007	0.000	11.342	14.246	1.638	0.796	6.924
2500	0.000	0.000	0.000	18.235	20.399	1.703	0.894	10.708
3600	0.000	0.000	0.000	24.820	34.826	1.841	0.713	13.482
4900	0.000	0.000	0.000	40.404	47.256	1.888	0.855	21.400
6400	0.000	0.000	0.000	59.657	63.446	1.950	0.940	30.593

Multiple Factor Analysis

Control factors are:

- N = Number of particles
- D = Solution dimensions
- S = Thread block-size



Trajectory Analysis

- Trajectory analysis for aircrafts. PSO used as an alternative to direct numerical methods.
- Particle count of 10,000! Demonstrates effectiveness of parallel processing.
- Full GPU in 9s
- Partial GPU in 20s
- CPU in 1464s
- Therefore a speed-up of 161x by GPU.

Conclusion

- PSO fits SIMD architecture naturally.
- Factors that effect performance analysed.
- Comparison between parallel and serial approach to PSO.
- Partial GPU shows copying overhead.
- Reinforces GPGPU programming is powerful and available.
- Future parallel CPU version worth investigating

References

[1]: Wu, Q., Xiong, F., Wang, F. and Xiong, Y., 2016. Parallel particle swarm optimization on a graphics processing unit with application to trajectory optimization. *Engineering Optimization*, *48*(10), pp.1679-1692.

[2]: Reynolds, C.W., 1987, August. Flocks, herds and schools: A distributed behavioral model. In *ACM SIGGRAPH computer graphics* (Vol. 21, No. 4, pp. 25-34). ACM.