Particle Swarm Optimization

Optimization Methods for Engineering Problems

Brignone Giovanni (303390) Ding Zhipeng (302078) Wu Haoke (302664)

31 May 2022

Politecnico di Torino

Outline

1. Particle Swarm Optimization
2. Application
3. Results
4. Conclusion

Particle Swarm Optimization

Particle Swarm Optimization

Definitions:

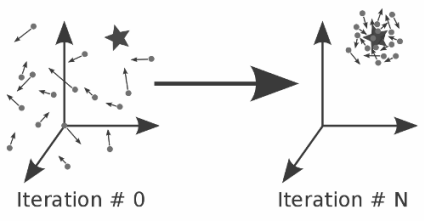
- *Time* (t): iteration step of the optimization loop
- *Particle*: entity associated with a position $x_i(t) \in \mathbb{R}^N$ and a velocity $\vec{V}_i(t)$
- *Swarm*: set of particles
- *Cost function*: $f : \mathbb{R}^N \mapsto \mathbb{R} | f(x_i) = \text{cost}_i$

Particle Swarm Optimization

Optimization loop:

At each time step $\vec{V}_i(t), \forall i$ is updated:

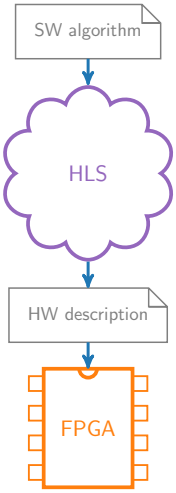
- *Cognitive behavior* (exploration): particle i accelerates toward regions with better $\mathrm{cost_i}$
- *Social behavior* (exploitation): particle i accelerates according to the other particles



Application

High-Level Synthesis

- HLS: convert SW algorithm to HW
- FPGA: implement custom HW
  - reconfigurable logic
  - large but slow off-chip memory
  - fast but small on-chip memory (scratchpad)



Cache module for HLS

- 2-levels array specific cache module
- Fully configurable:
  - # words per line
  - # sets (L1 and L2)
  - # ways (L1 and L2)

Cache module for HLS

- 2-levels array specific cache module
- Fully configurable:
  - # words per line
  - # sets (L1 and L2)
  - # ways (L1 and L2)
- Objective: maximum performance and minimum resource usage

Implementation

Cost function:

- Compress multiple objectives into a single one: weighted sum aggregation

$$c := \sum_i w_i \cdot c_i \qquad \begin{cases} w_{\text{miss}} & = 64 \\ w_{\text{words}} & = 1 \\ w_{\text{L1sets}} & = 4 \\ w_{\text{L2sets}} & = 2 \\ w_{\text{L1ways}} & = 16 \\ w_{\text{L2ways}} & = 8 \end{cases}$$

- Miss ratio evaluated by simulation (one thread per simulation)

Implementation

Degree of Freedom:

- Optimizer space: $\vec{x} \in \mathbb{R}^N$
- Cache parameters space: $\vec{p}, \vec{n} \in \mathbb{N}^5 | \vec{p} = 2^{\circ \vec{n}}$
- Workaround: $\vec{p} = 2^{\circ [\vec{x}]}$
  - Rounding $\Rightarrow$ discontinuity and aliasing
  - Exponential $\Rightarrow$ non-linearity
- Boundaries:

  Words $\in [8, 64]$

  L1 sets $\in [0, 32]$ L1 ways $\in [0, 32]$ $\Rightarrow$ 11,520 combinations

  L2 sets $\in [1, 128]$ L2 ways $\in [1, 128]$

Implementation

Optimizer: `pyswarms` library

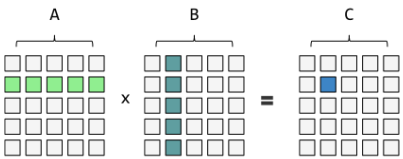
- Cognitive parameter: 0.8
- Social parameter: 0.3
- # particles: 16
- # iterations: 10

Results

Matrix multiplication

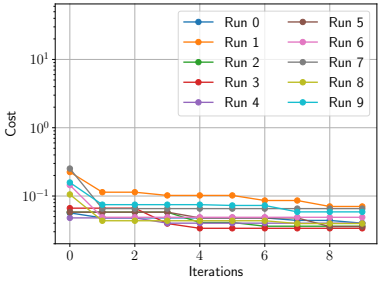
$$C = A \times B, \quad A, B, C \in \mathbb{Z}^{64 \times 64}$$

- A, C accessed by rows: single-line cache
- B accessed by columns: 64-lines cache



Matrix multiplication - A matrix

PSO



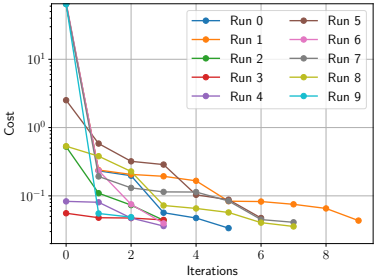
Best cost: 0.0337

Hit: 99.98% Words: 64

L1 sets: 0 L1 ways: 0

L2 sets: 1 L2 ways: 1

Random sampling



Best cost: 0.0337

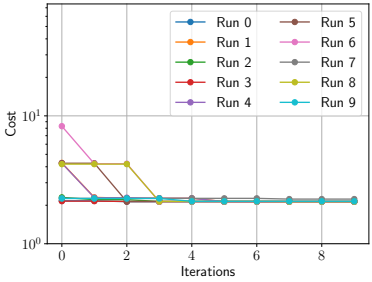
Hit: 99.98% Words: 64

L1 sets: 0 L1 ways: 0

L2 sets: 1 L2 ways: 1

Matrix multiplication - B matrix

PSO



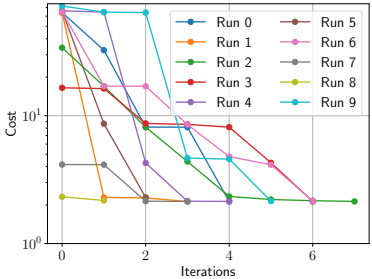
Best cost: 2.1333

Hit: 99.98% Words: 64

L1 sets: 0 L1 ways: 0

L2 sets: 1 L2 ways: 64

Random sampling



Best cost: 2.1338

Hit: 96.88% Words: 32

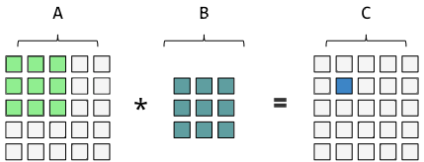
L1 sets: 0 L1 ways: 0

L2 sets: 2 L2 ways: 32

2D convolution

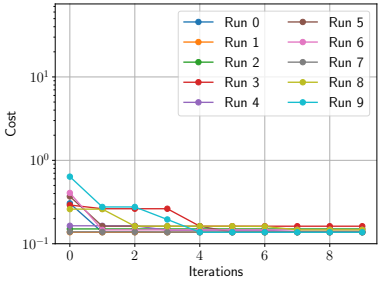
$$C = A * B, \quad A, C \in \mathbb{Z}^{64\times64}, B \in \mathbb{Z}^{3\times3}$$

- A accessed by sliding window: 4-lines cache



2D convolution - A matrix

PSO



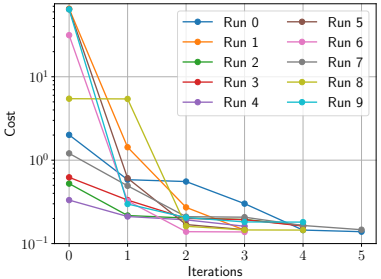
Best cost: 0.1374

Hit: 99.82% Words: 64

L1 sets: 0 L1 ways: 0

L2 sets: 1 L2 ways: 4

Random sampling



Best cost: 0.1379

Hit: 99.82% Words: 64

L1 sets: 0 L1 ways: 0

L2 sets: 2 L2 ways: 2

Conclusion

Conclusion

Achieved results:

- Automatically find good trade-offs between performance and resources

Future work:

- Improve cost definition to better exploit L1 cache
- Apply a more suitable optimization algorithm
  - Multi-objective: find Pareto set instead of single point (dependent on cost definition)
  - Discrete: avoid discontinuity and aliasing, which lead to local minima
- Speed up hit ratio evaluation through surrogate modeling

References

[1] James Kennedy and Russell C. Eberhart. "Particle swarm optimization". In: *Proceedings of ICNN'95 - International Conference on Neural Networks* 4 (1995), 1942–1948 vol.4.