Accelerating a local search algorithm for large instances of the independent task scheduling problem with the GPU

Frederic Pinel[1] Johnatan Pecero[1] Pascal Bouvry[1]
Samee U. Khan[2]

[1]Computer Science and Communications
University of Luxembourg

[2]Electrical and Computer Engineering
North Dakota State University

International Research Workshop on Advanced High Performance Computing Systems

Outline

Millicomputing

- ARM cortex-A9 based, Panda boards (OMAP4430)
- Compare with data center server

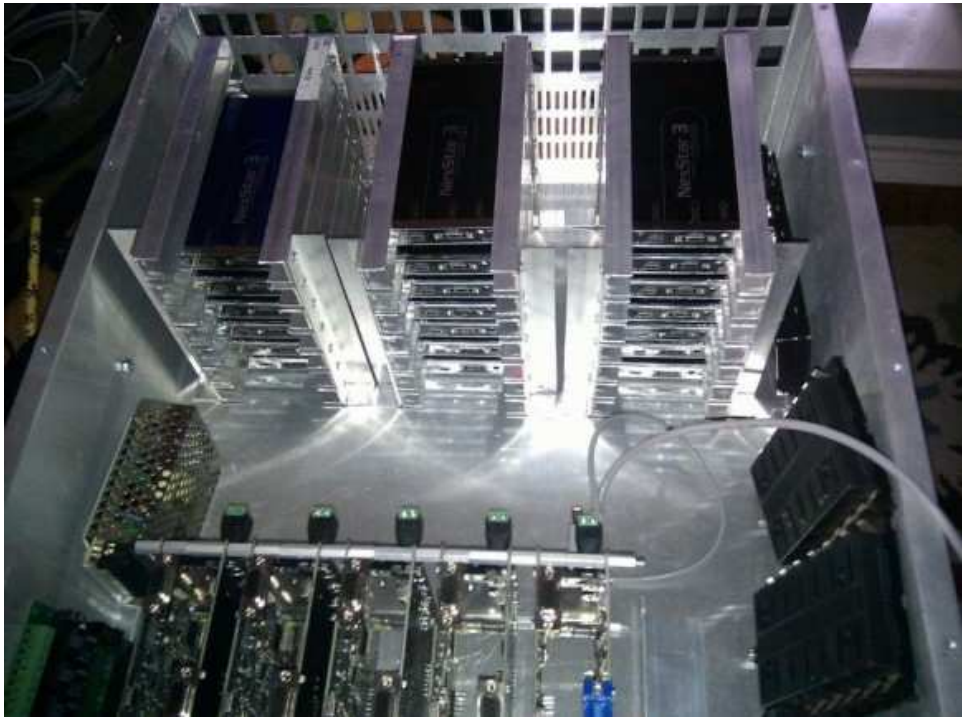


Figure: Canonical ARM build farm

Opportunities

- Low energy (5-20W ?)
- Limited cooling
- Reduced contention
- Better programs
- SLA

Problems

- Programming model
  - Message passing (Erlang)
  - Pipelining (message queues)
  - Independent tasks
- Mapping (spawn multiple task instances)
- Task characterization
- Networking

Problem definition

- Mapping independent tasks on a distributed system
- Hypothesis: estimated time to complete (ETC)
- Makespan, combines several perspectives:
  - User: flowtime
  - Provider: load balance, energy (low machine heterogeneity)
- Fast algorithm: runtime, wallclock

Contribution

- Mapping heuristic
- Parallel version for large instances on the GPU
- Lessons learned

Background

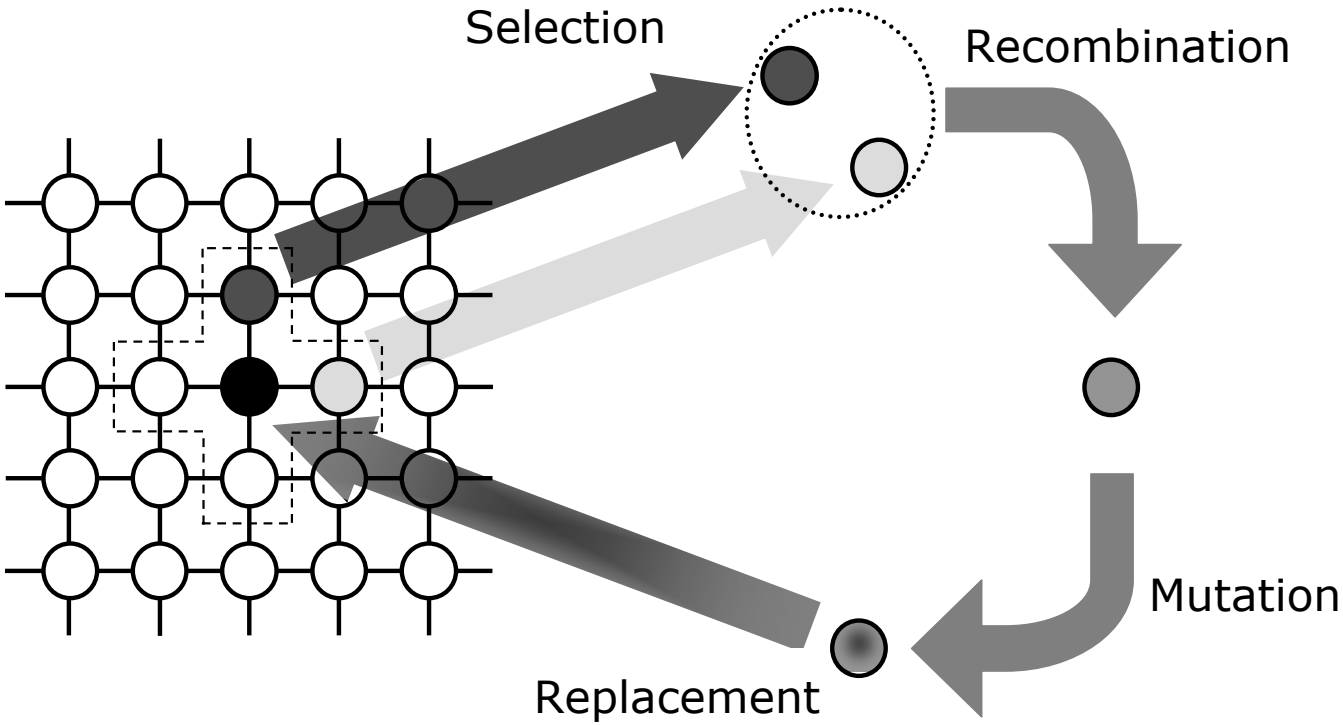


Figure: Cellular genetic algorithm (CGA)

Parallel CGA

- SIMD
- Clusters of sequential machines
- Multicore machines
- Communication models

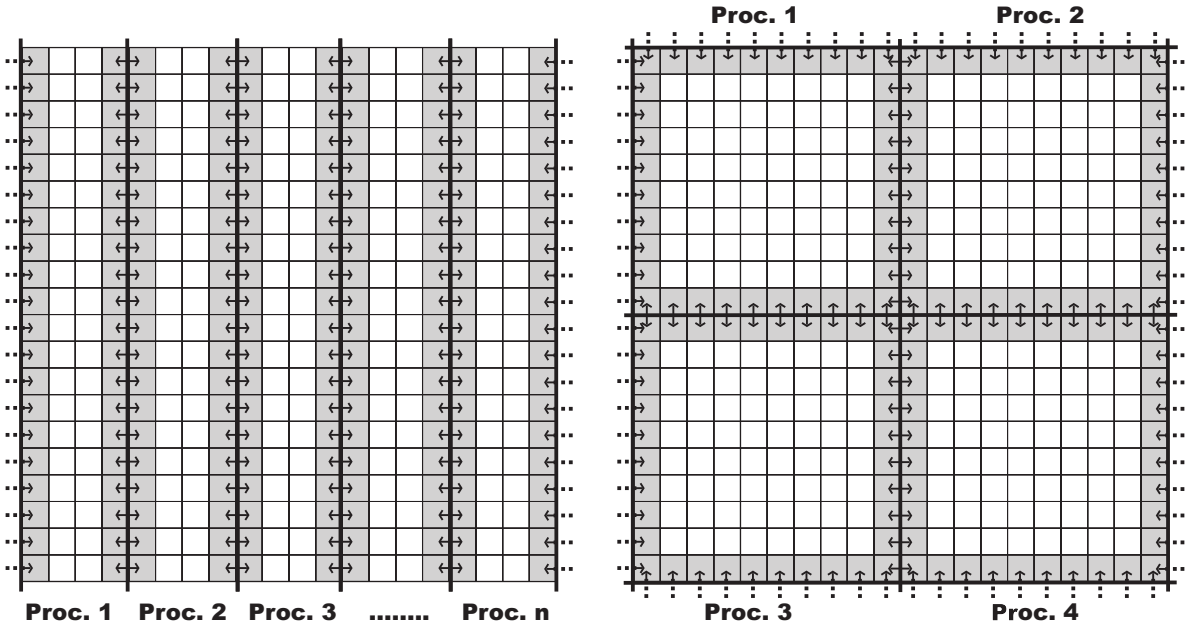
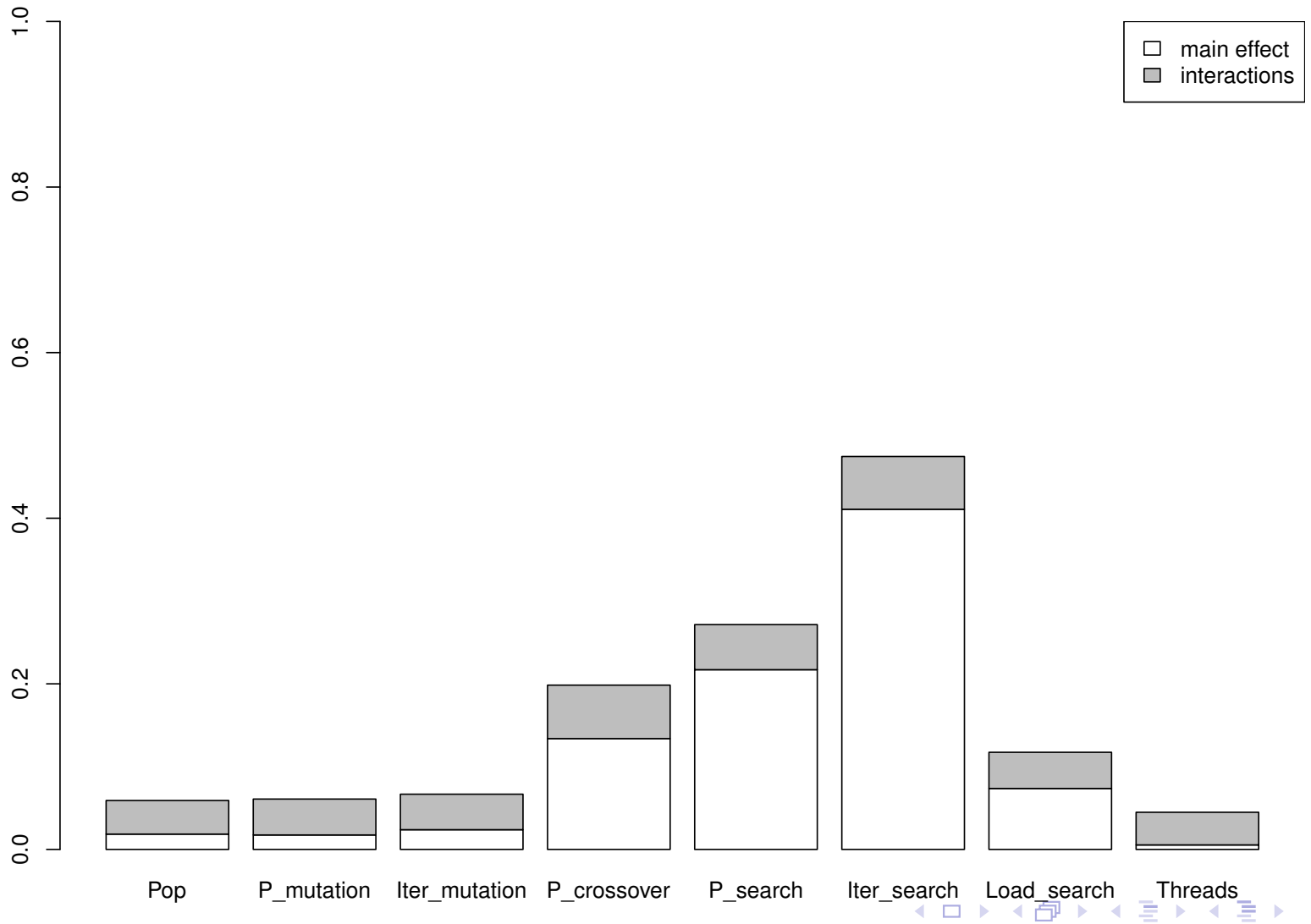


Figure: Population decomposition

Starting point

- Parallel asynchronous cellular genetic algorithm
- Initialized with heuristic (Min-Min)
- With local search (5 iter)
- Problems: 512 tasks $\times$ 16 cores

Feedback


main effect
interactions
1.0
0.8
0.6
0.4
0.2
0.0
Pop
P_mutation
Iter_mutation
P_crossover
P_search
Iter_search
Load_search
Threads

Adaptation

- Simplified algorithm
- Min-Min, incremental formulation
- Local search (30 iter), complete-state formulation

Results

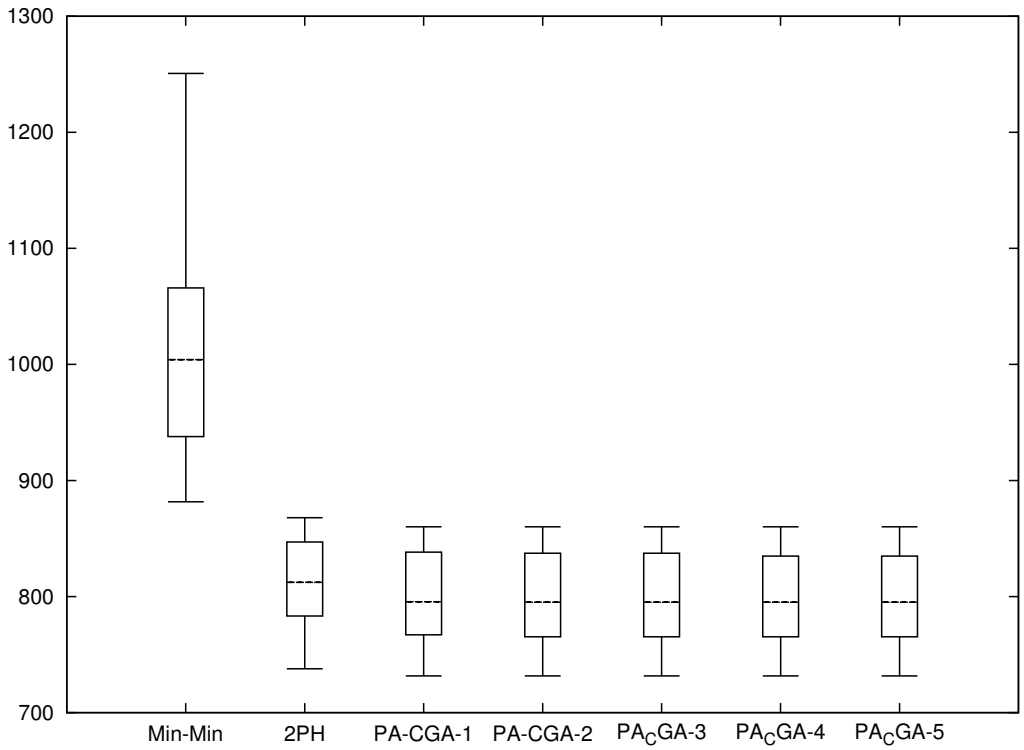


Figure: Consistent, high-h. tasks, low-h. machines

Results

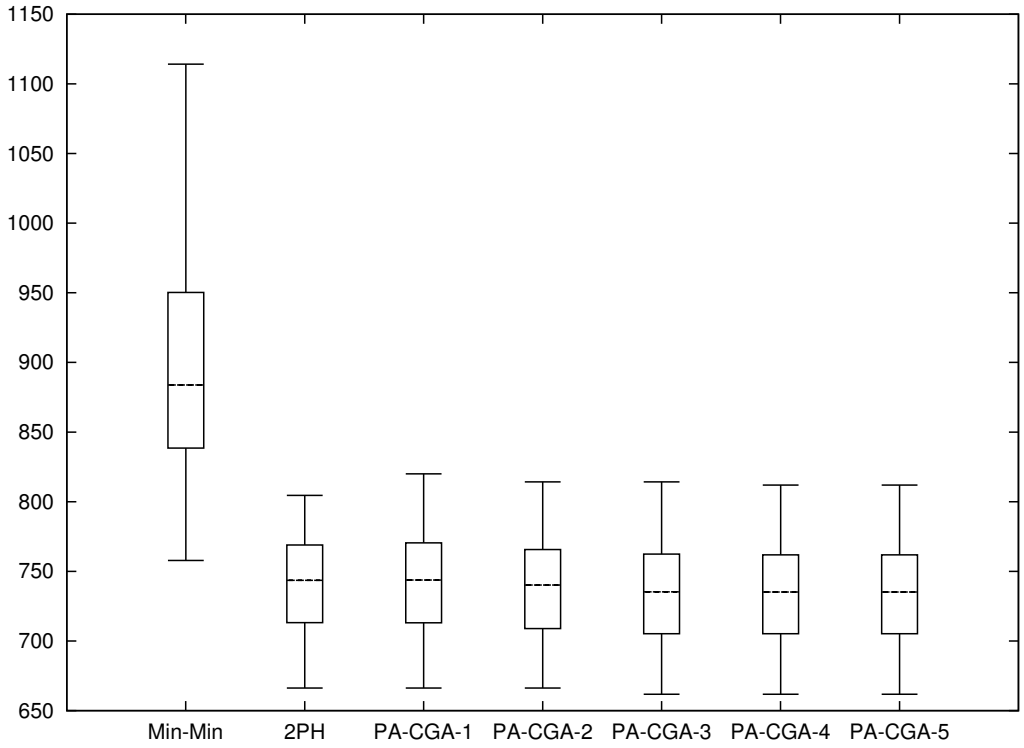


Figure: Semi-consistent, high-h. tasks, low-h. machines

Results

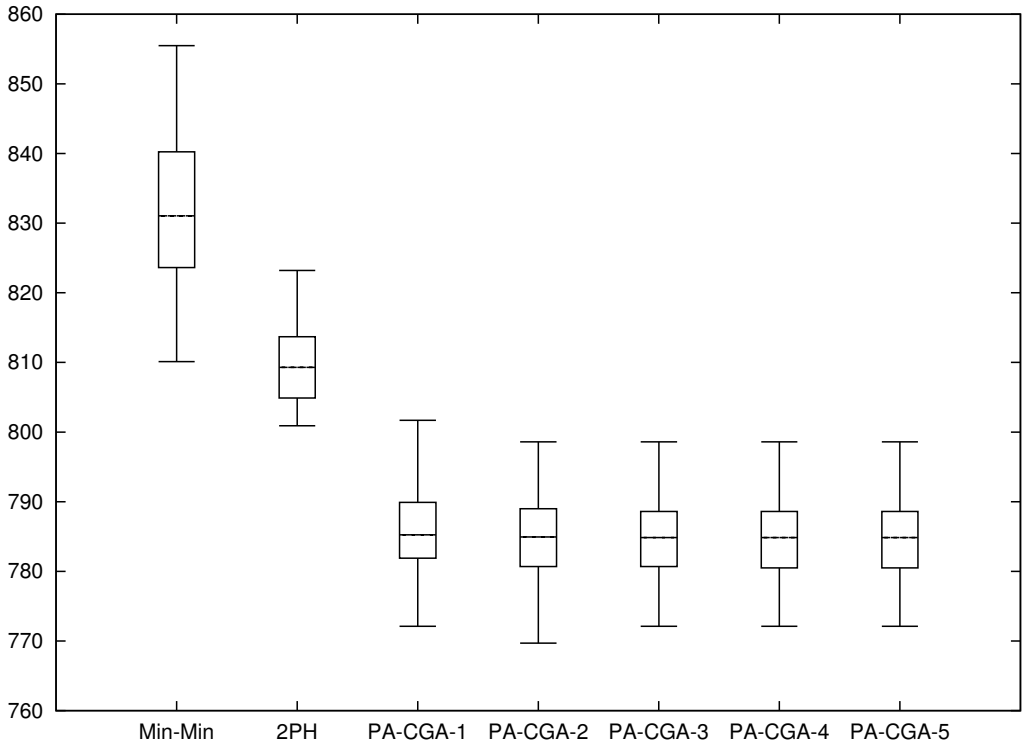


Figure: Consistent, low-h. tasks, low-h. machines

Results

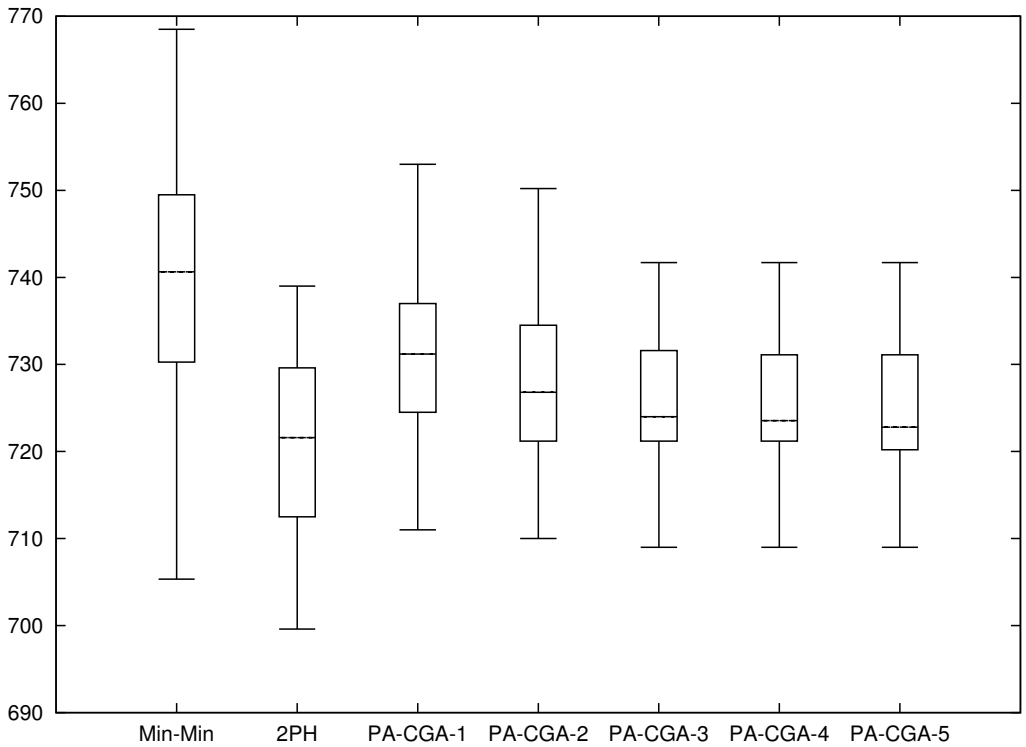


Figure: Semi-consistent, low-h. tasks, low-h. machines

Min-Min runtime

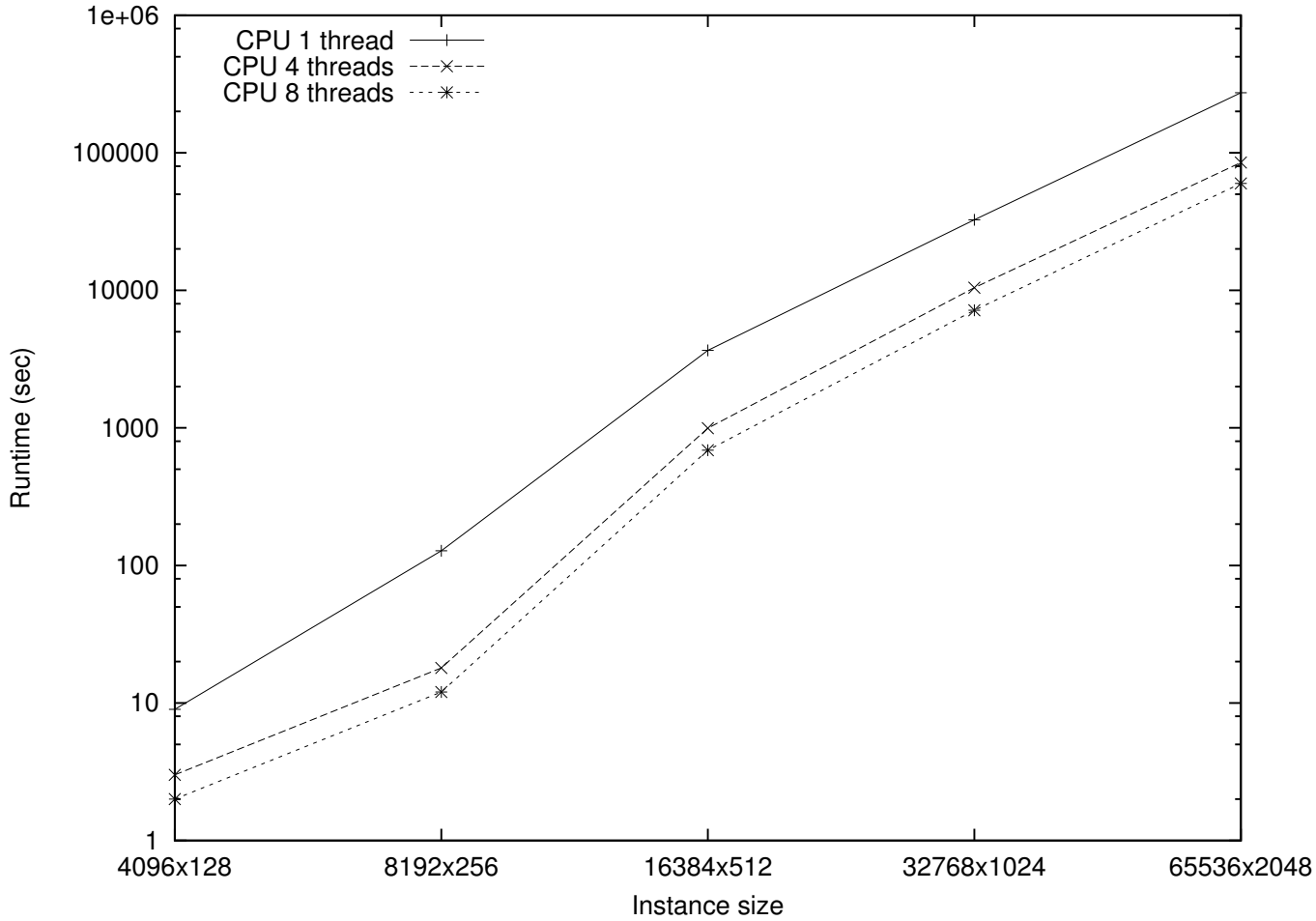


Figure: Runtime Min-Min (minimum in array)

Min-Min on GPU

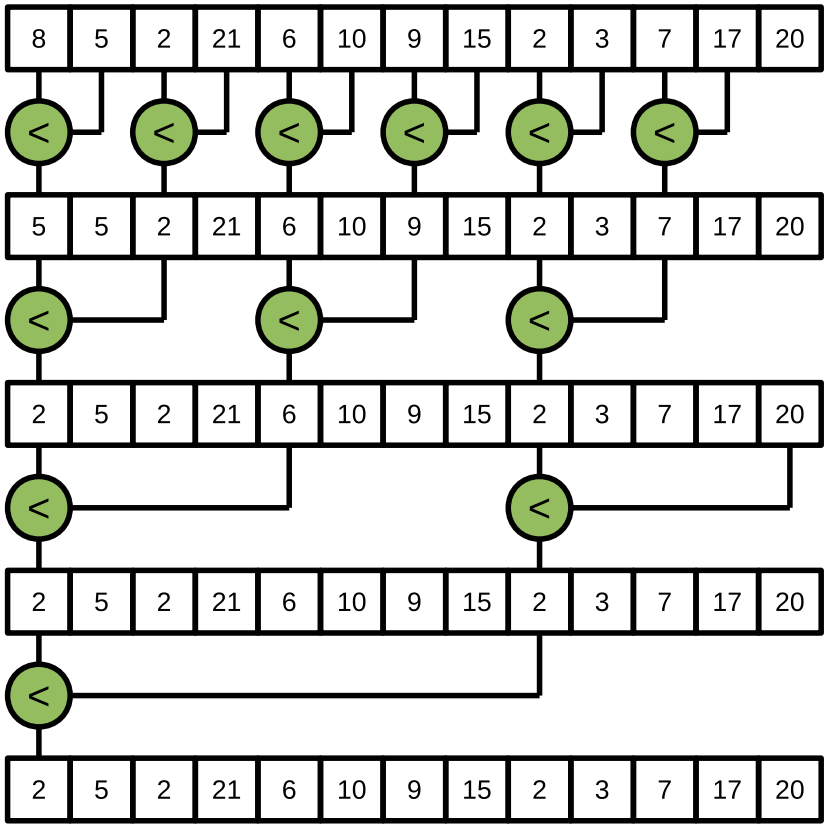


Figure: Parallel reduction in Min-Min

Min-Min runtime

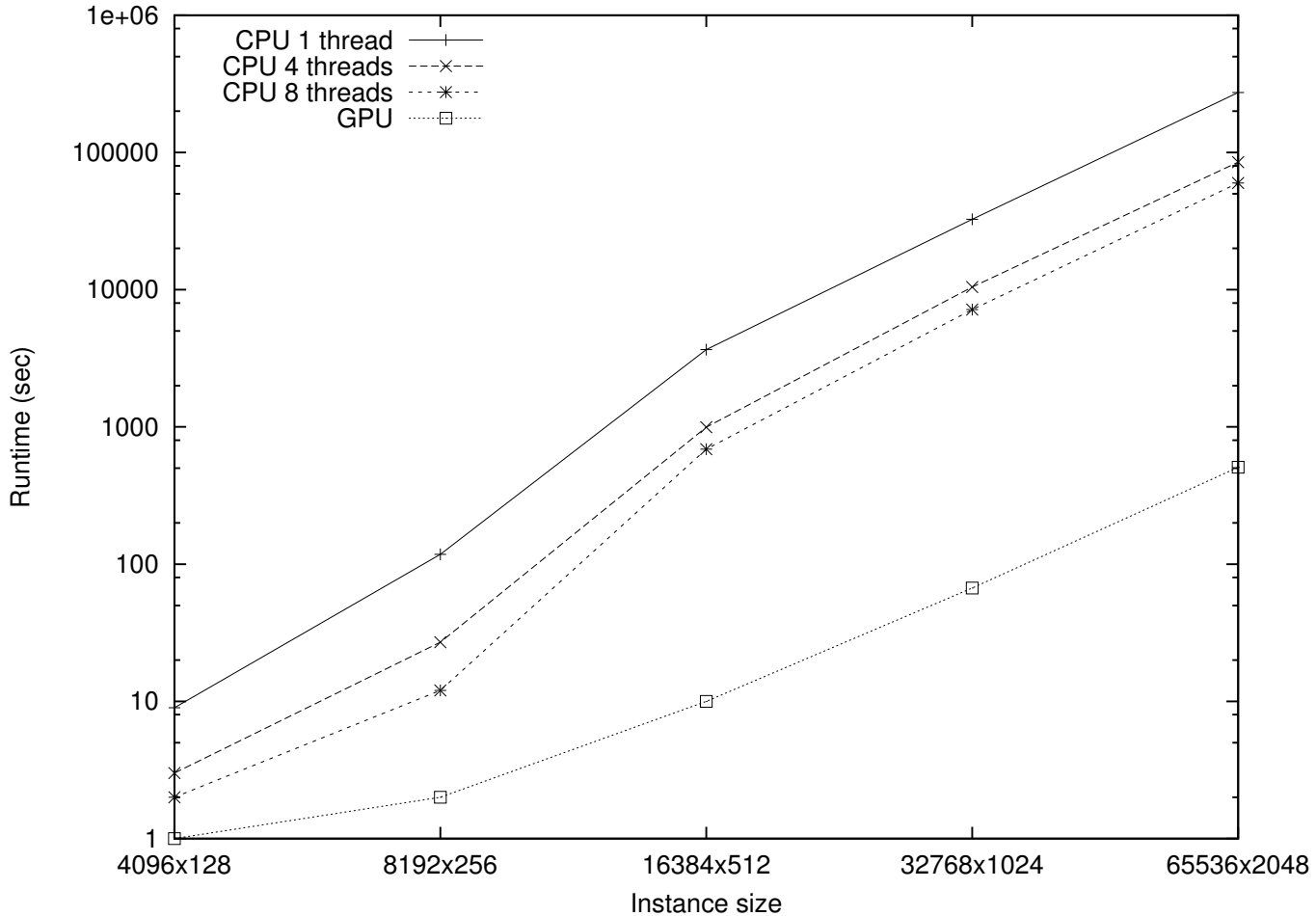


Figure: Runtime Min-Min

Quality of solution

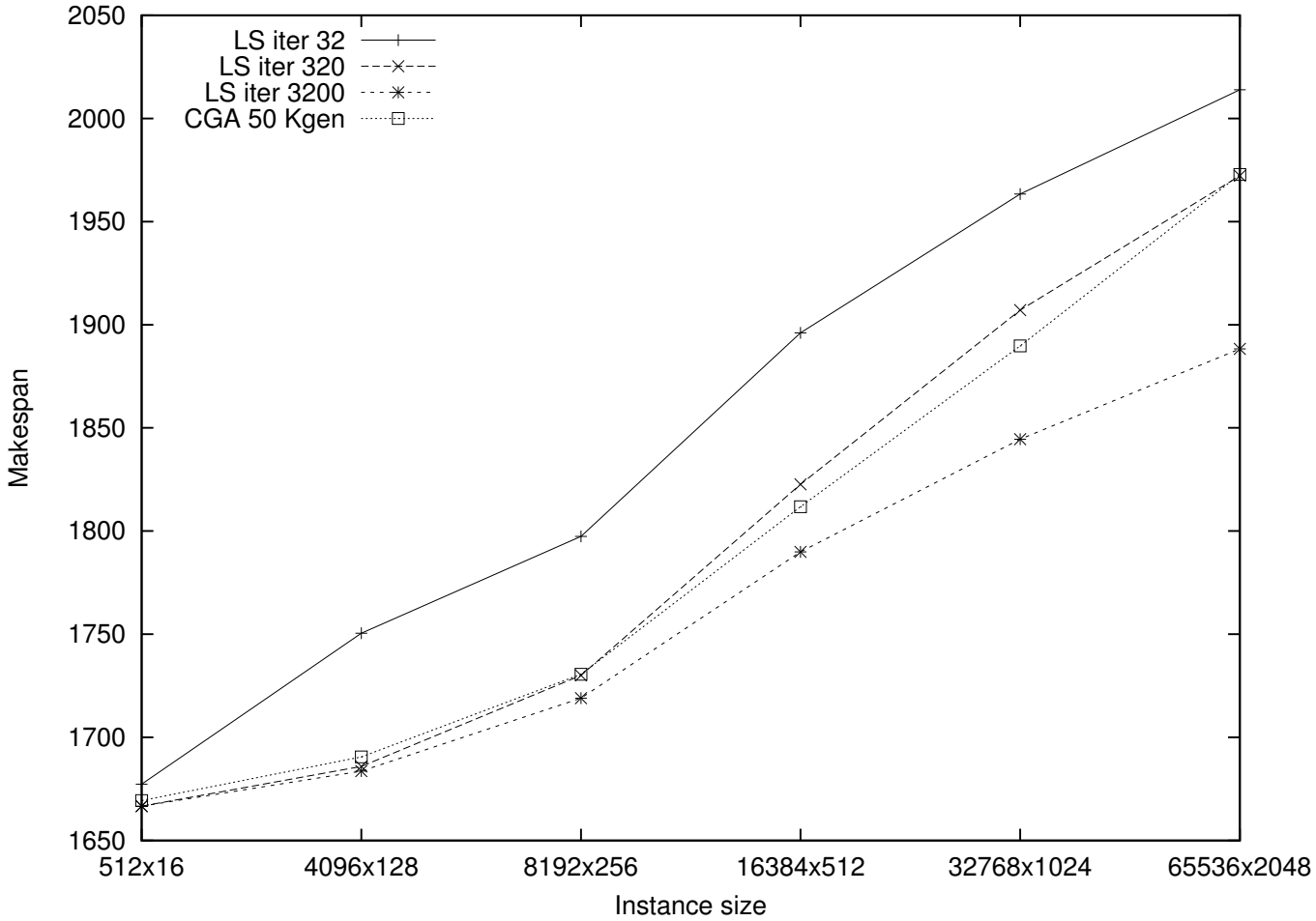


Figure: Makespan

Performance

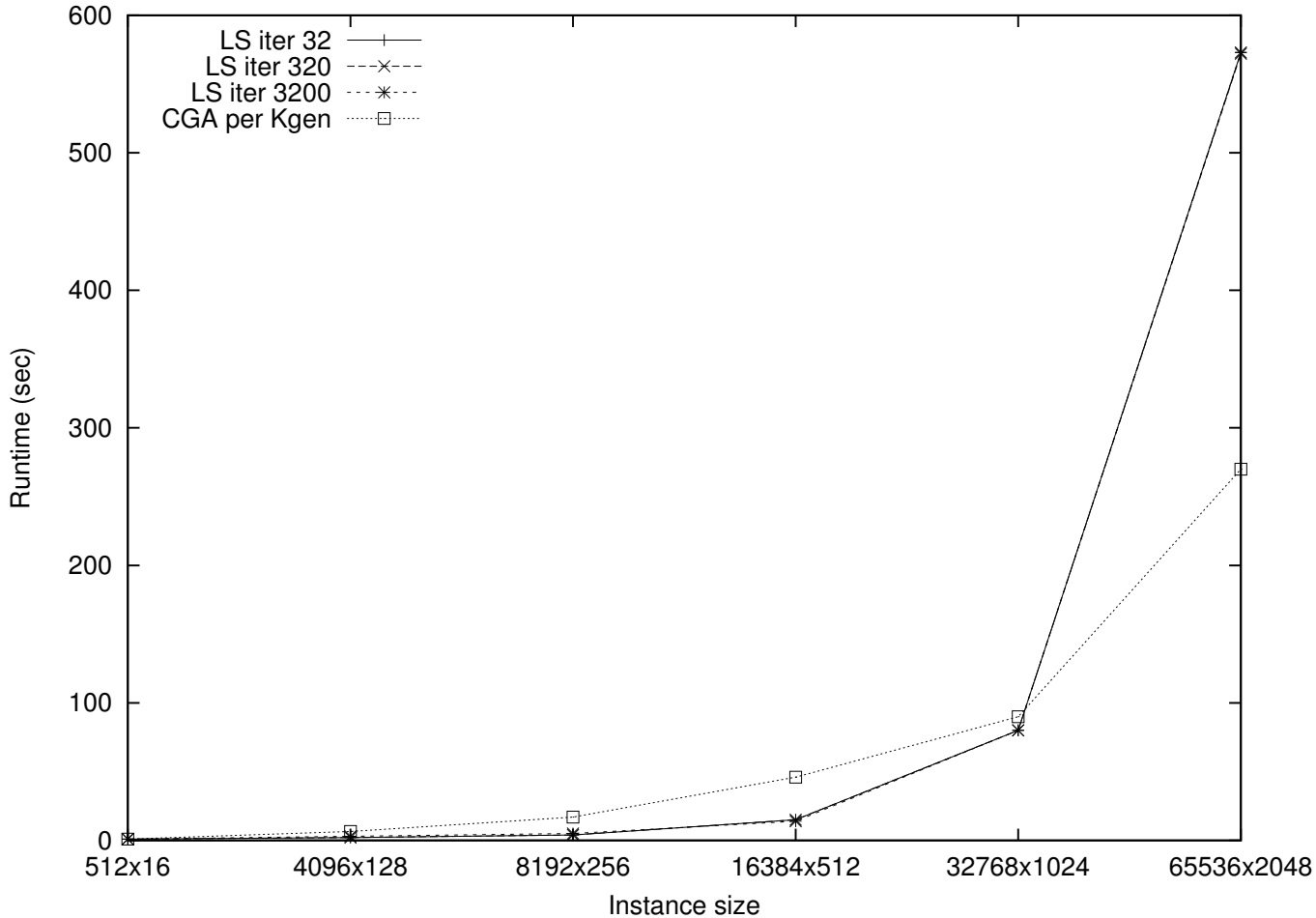


Figure: Runtime

Perspectives

- Failure
- Solution $\rightarrow$ Feedback $\rightarrow$ Loop
- Learning process

Machine learning opportunities

- Learn from problem instance
  - Classify problem instance
  - Task characterization
- Learn mapping rules
  - Adapt (parameters, heuristics)
  - Algorithm (oracle: solved instances)
  - Data placement
  - Networking

Questions

Thank you.