CS 2310 - Multimedia Software Engineering

A Bi-population Particle Swarm Optimizer for Learning Automata based Slow Intelligent System

Mohammad H. Mofrad
University of Pittsburgh
Thursday, December 08, 2016

Preface

- Slow Intelligence System (SIS)
- Particle Swarm Optimization (PSO)
- Learning Automata (LA)
- Adaptive Intelligence Optimizer (AIO)

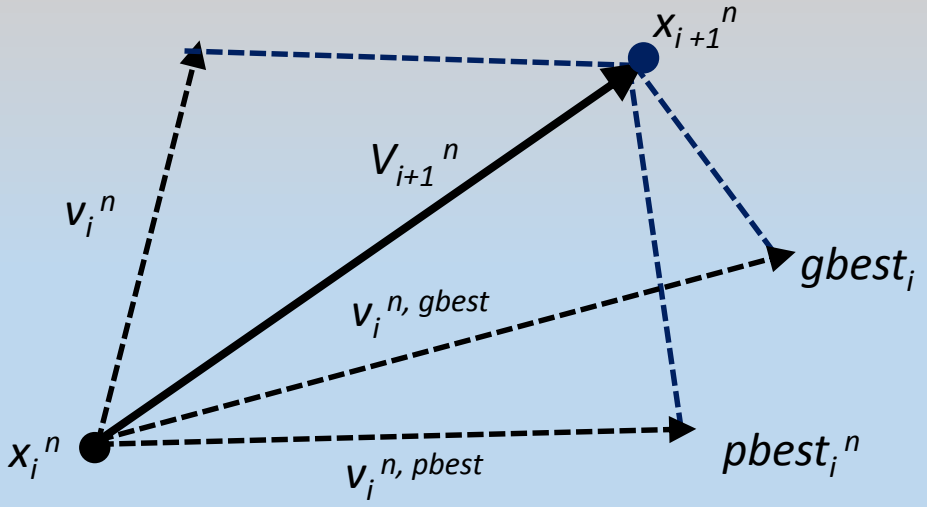
Slow Intelligence System (SIS)

- **Enumeration** (-enum<) of the different available solutions until finding the optimal solution
- **Propagation** (=prop+) of the achieved new information from the new solutions within a body of feasible solutions.
- **Adaptation** (+adap=) of the current solutions using the effective information gained from the elite solutions.
- **Elimination** (>elim-) of the worst solutions that exist in the problem space.
- **Concentration** (>conc=) on the elite solutions to produce new promising solutions.

S.-K. Chang, "Slow intelligence systems," in *International Conference on Multimedia Modeling*, 2010

Particle Swarm Optimization (PSO)

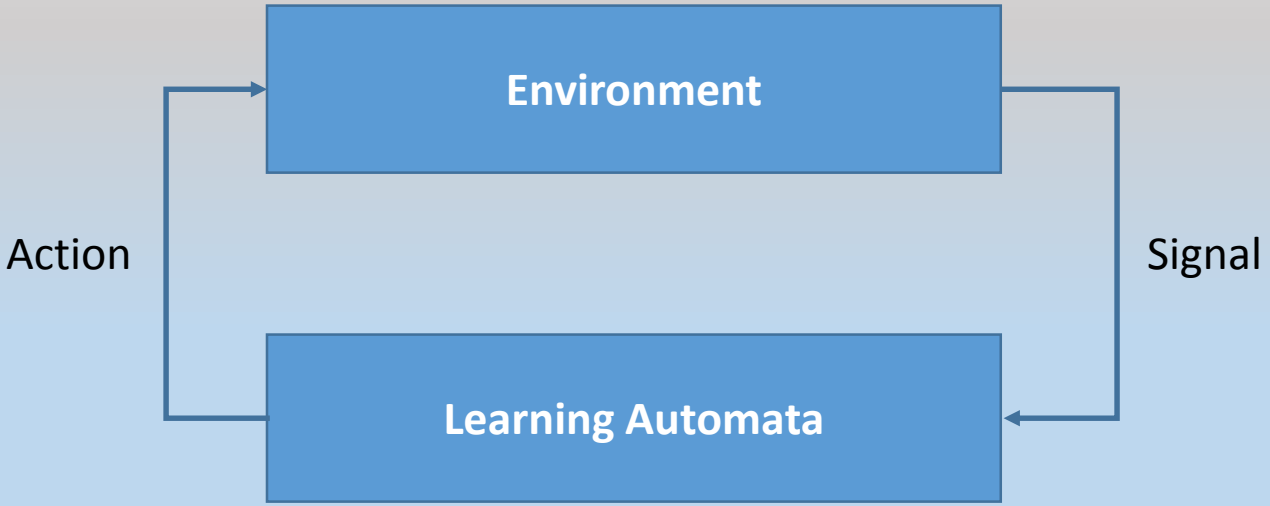
- Inspired from movement of animals
- $V_i = w\,V_i + c_1\,r_1\,(pbest_i - X_i) + c_2\,r_2\,(gbest - X_i)$
- $X_i = X_i + V_i$



J. Kennedy, “Swarm intelligence,” Handb. Nat.-Inspired Innov. Comput., pp. 187–219, 2006.

Learning Automata (LA)

- Belongs to the reinforcement learning family



Reward signal

$$p_j(n+1) = \begin{cases} p_j(n) + a \times (1 - p_j(n)) & j = i \\ p_j(n) \times (1-a) & \forall j \mid j \neq i \end{cases}$$

Penalty signal

$$p_j(n+1) = \begin{cases} p_j(n) \times (1-b) & j = i \\ b \times (r-1) + (1-b) \times p_j(n) & \forall j \mid j \neq i \end{cases}$$

M. Thathachar and P. S. Sastry, "Varieties of learning automata: an overview," Syst. Man Cybern. Part B Cybern. IEEE Trans. On, vol. 32, no. 6, 2002.

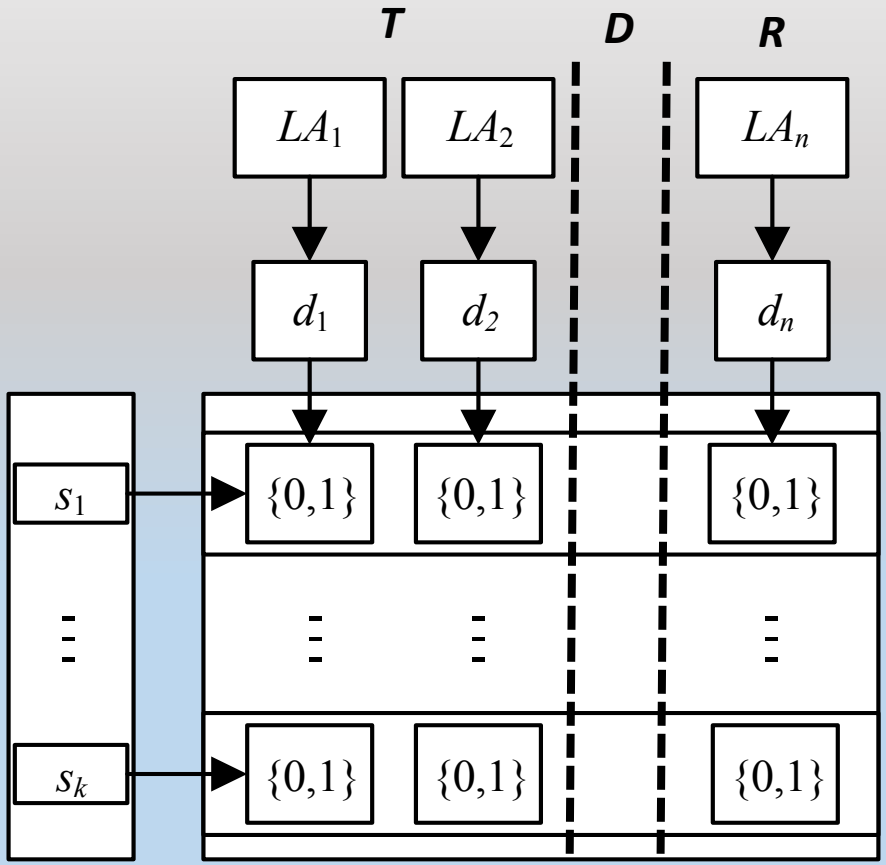
Adaptive Intelligence Optimizer (AIO)

- 2 PSO populations
- SIS framework
- LA framework

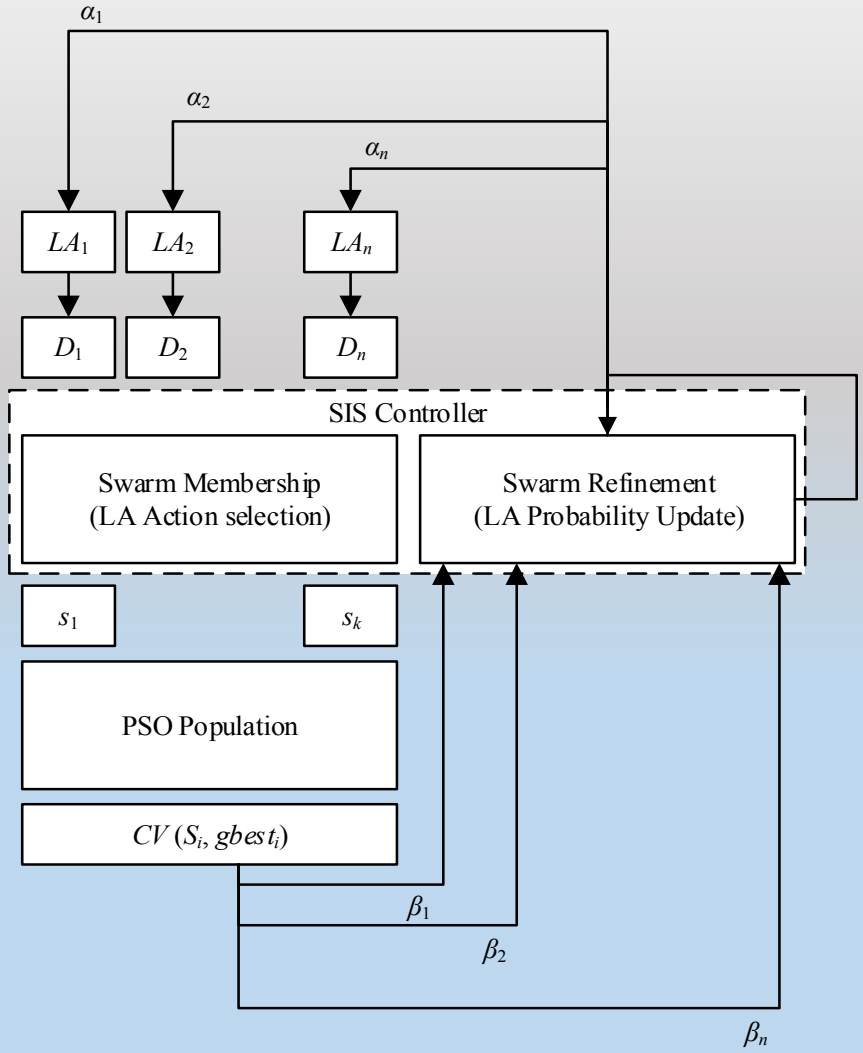
SIS + PSO

- Mapping SIS's operators to PSO formula
  - $cycle_1$: [$guard_{1,2}$] P_0 –enum< P_1 =prop+ P_2 >elim- P_3 >conc= P_4
  - $cycle_2$: [$guard_{2,1}$] P_0 –enum< P_1 =prop+ P_2 +adap= P_3 >elim- P_4 >conc= P_5
- *Simulating SIS's slow & quick decision cycles*
  - $w = w_{max} – (0.75i\ (w_{max} - w_{min})/i_{max})$
  - $w = w_{max} – (i\ (w_{max} - w_{min})/i_{max})$

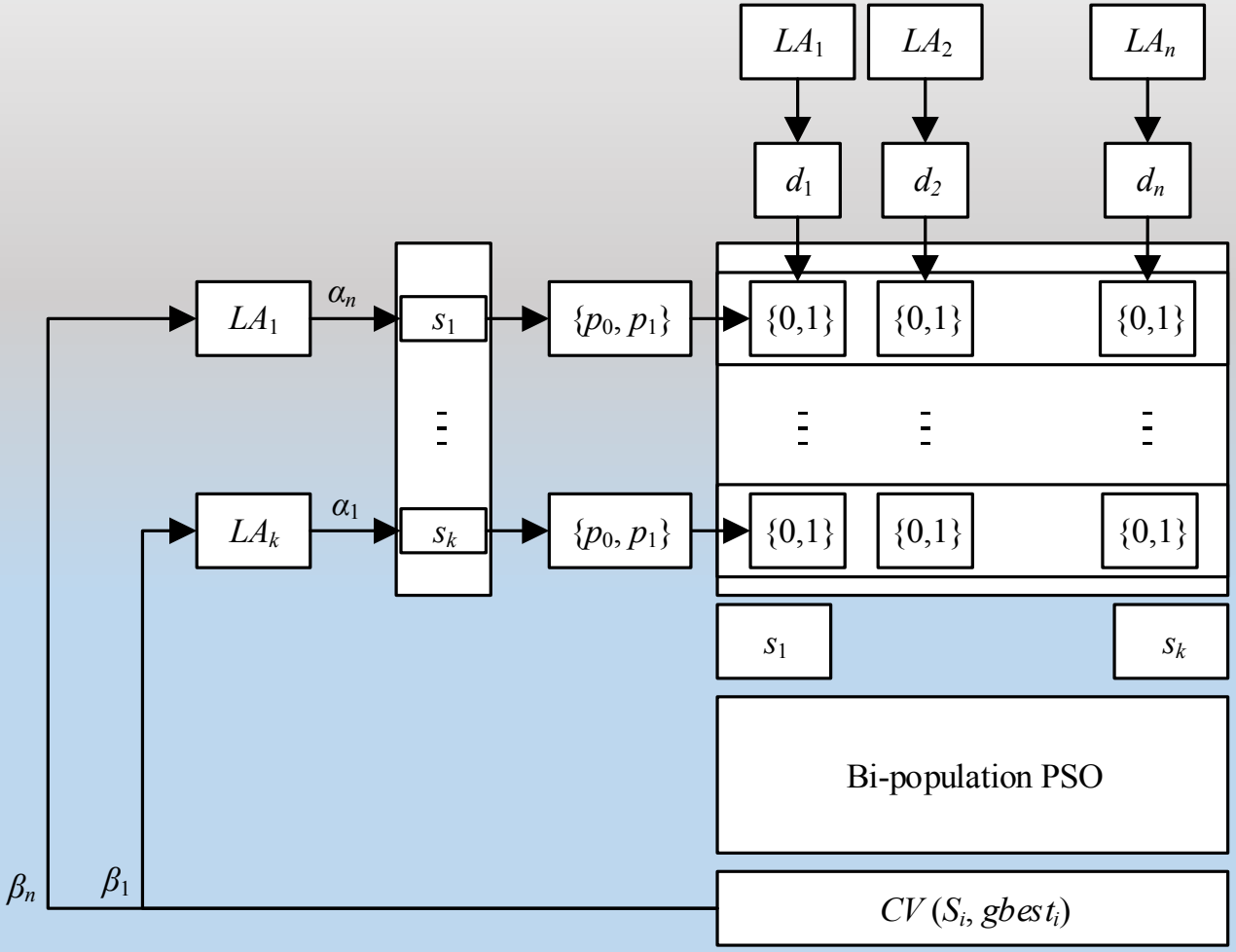
Implementing SIS's TDR system using AIO



Implementing SIS Controller Component



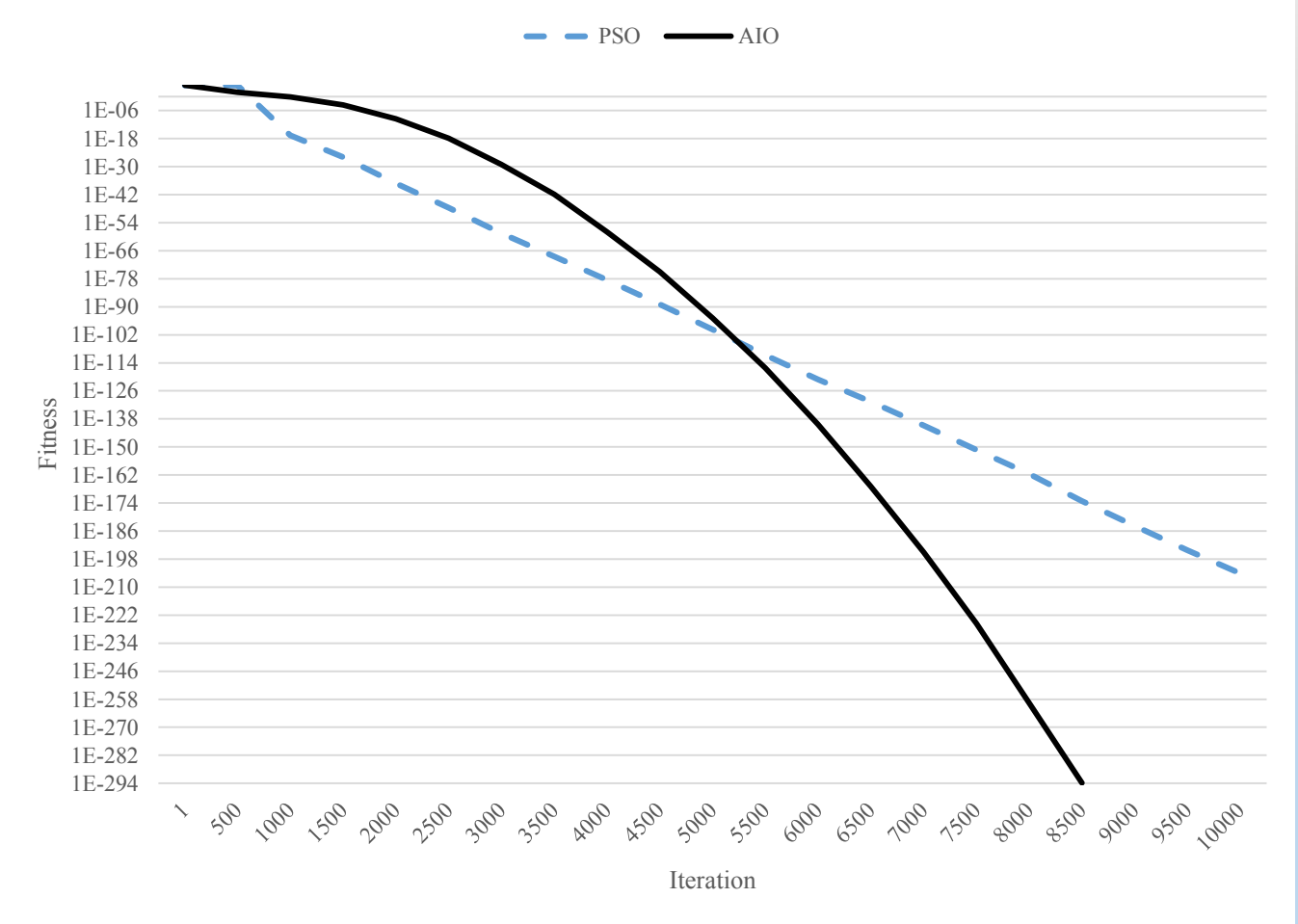
The Isolation of PSO populations



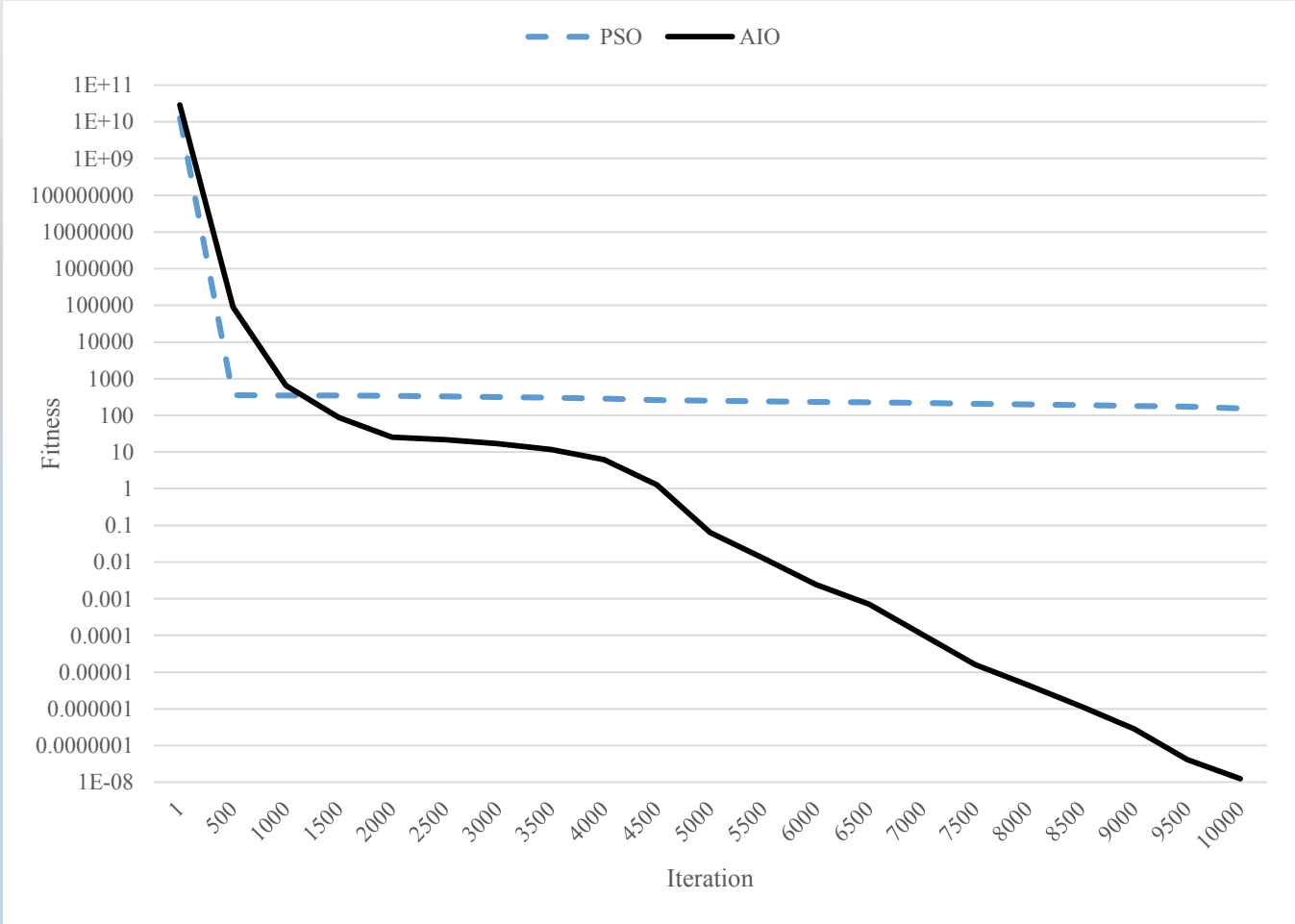
Implementation

- Python 3.4 (we extensively use NumPy package)
- Github repository @ https://goo.gl/V0vTRM
- Code
  - PSO ~200 lines
  - AIO ~400 lines
- XML interface for reading configurations
- 5 benchmark functions
  - Sphere, Rosenbrock, Ackley, Griewanks, Rastirign
  - 30 dimensions
  - 50 particles
  - …

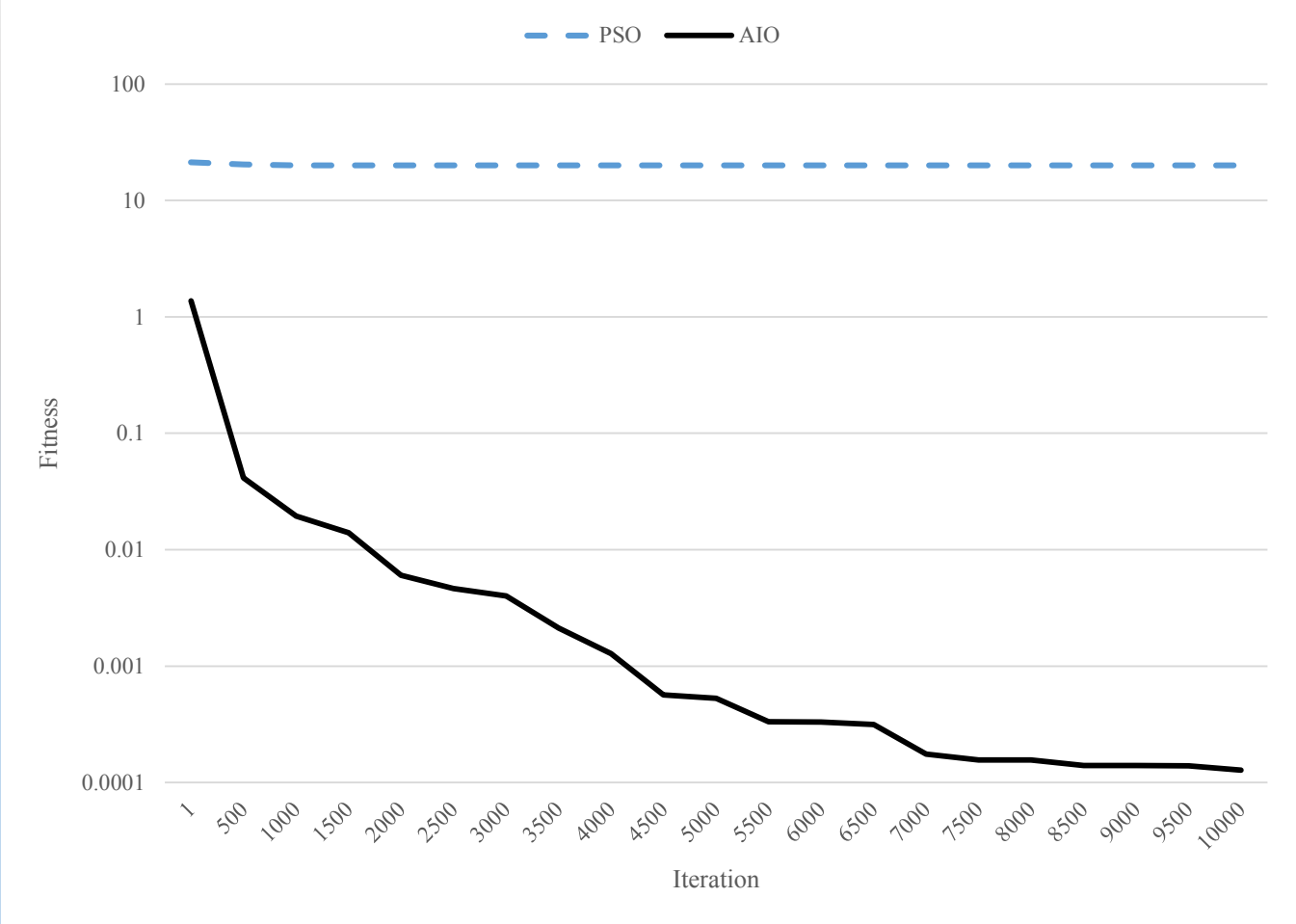
Results – Sphere Benchmark



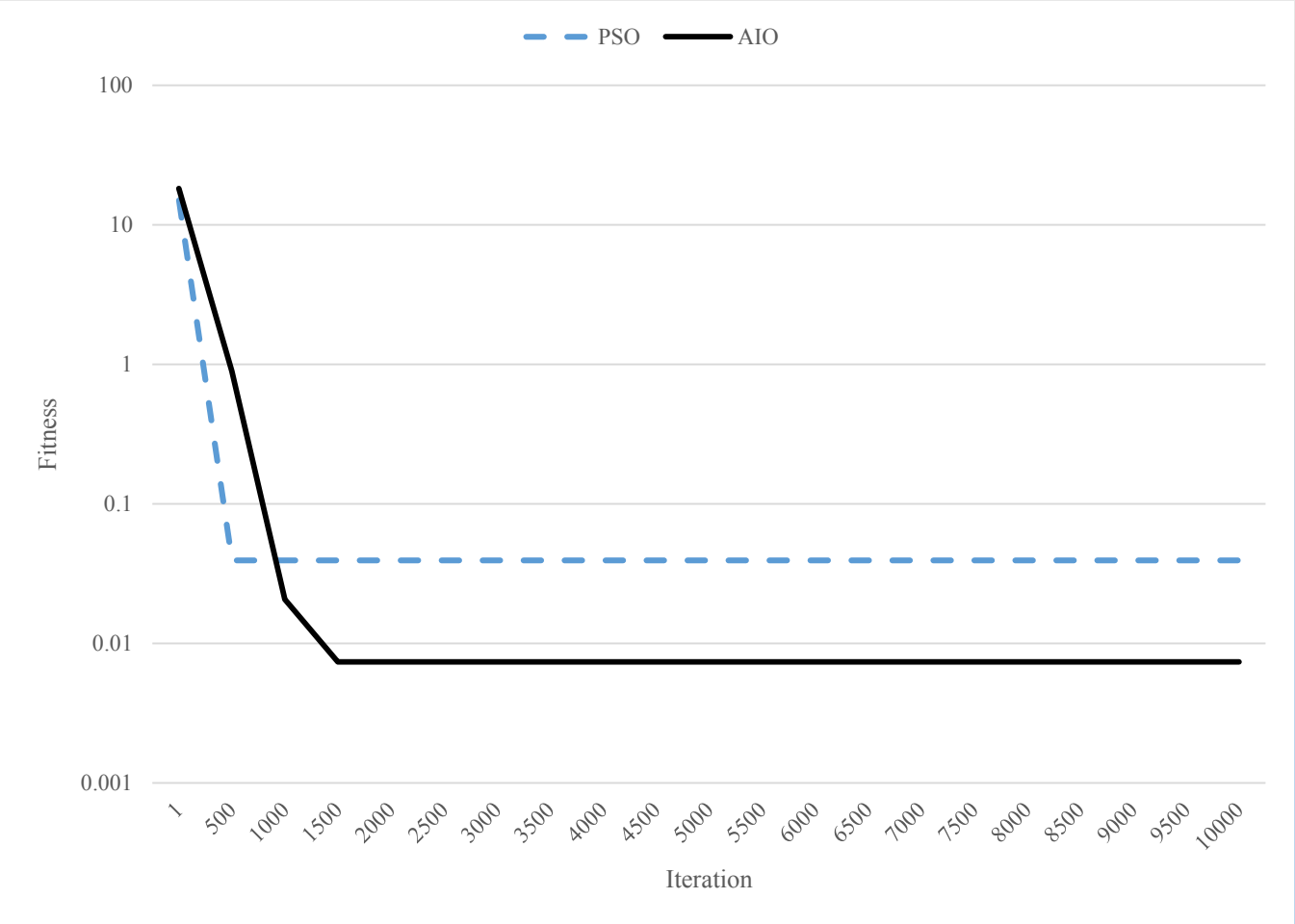
Results – Rosenbrock Benchmark



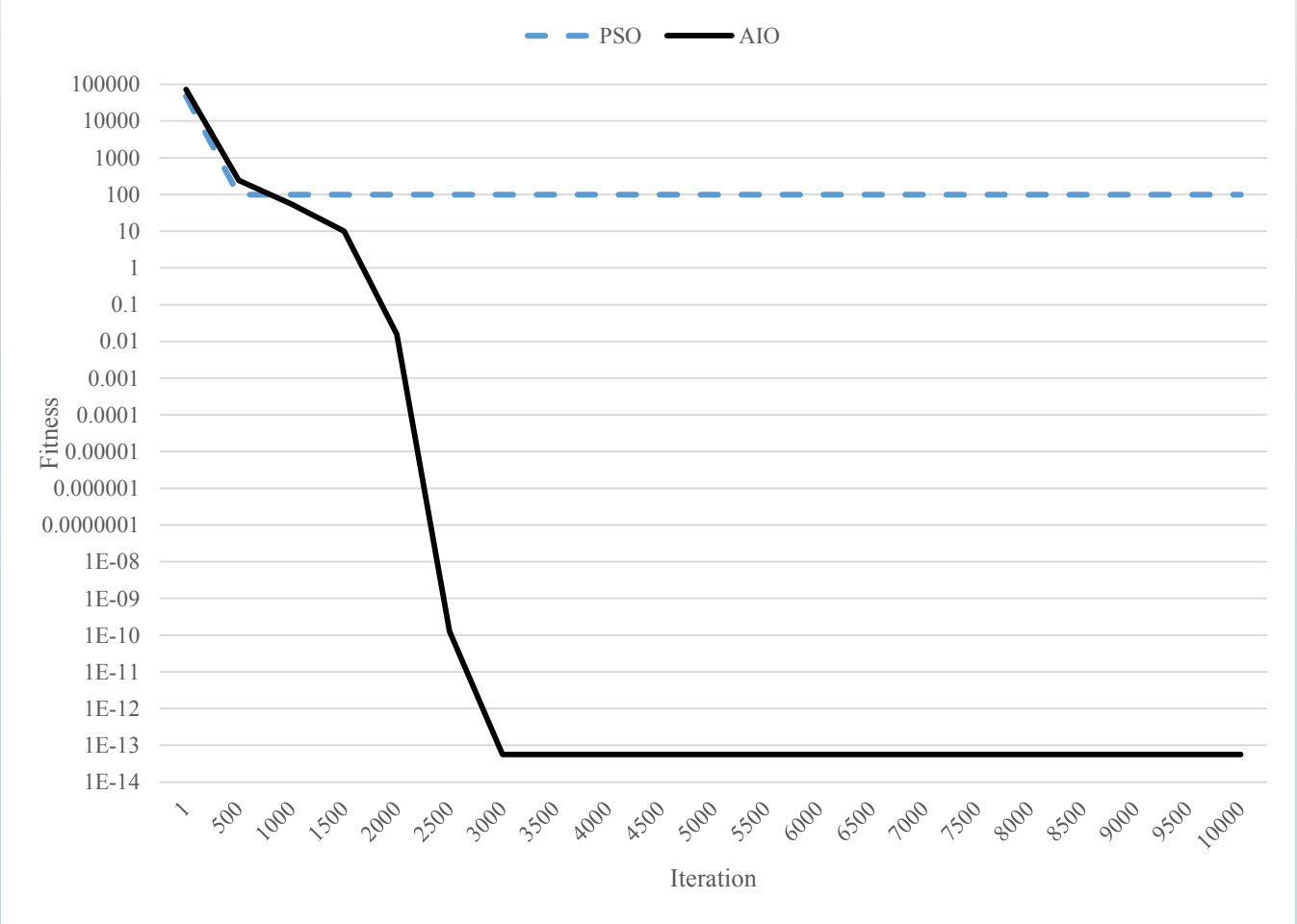
Results – Ackley Benchmark



Results – Griewanks Benchmark



Results – Rastrigin Benchmark



Future Work

- Clustering
- Dimensionality reduction
- Feature selection
- Parameter estimation

Questions?