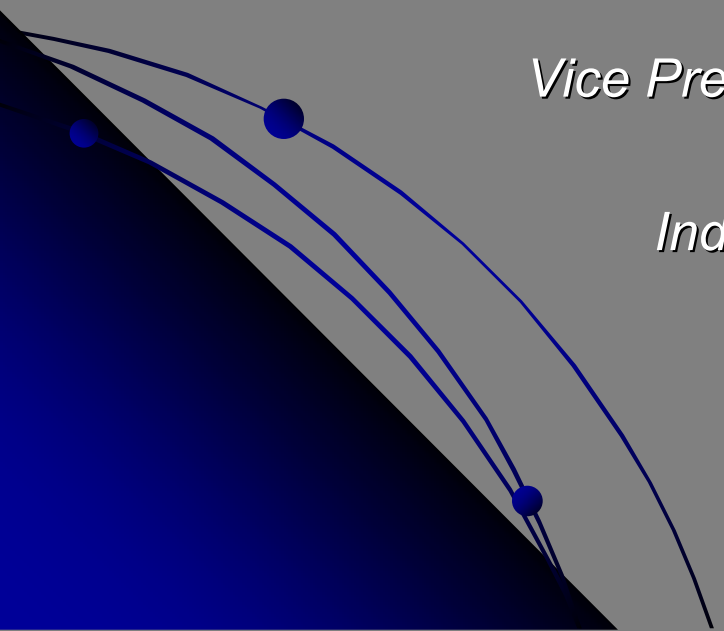
Particle Swarm Optimization

Russell C. Eberhart

Chairman, Department of Electrical and Computer Engineering
Purdue School of Engineering and Technology at IUPUI

Vice President, Computelligence LLC

Indianapolis, Indiana, USA
reberhar@iupui.edu

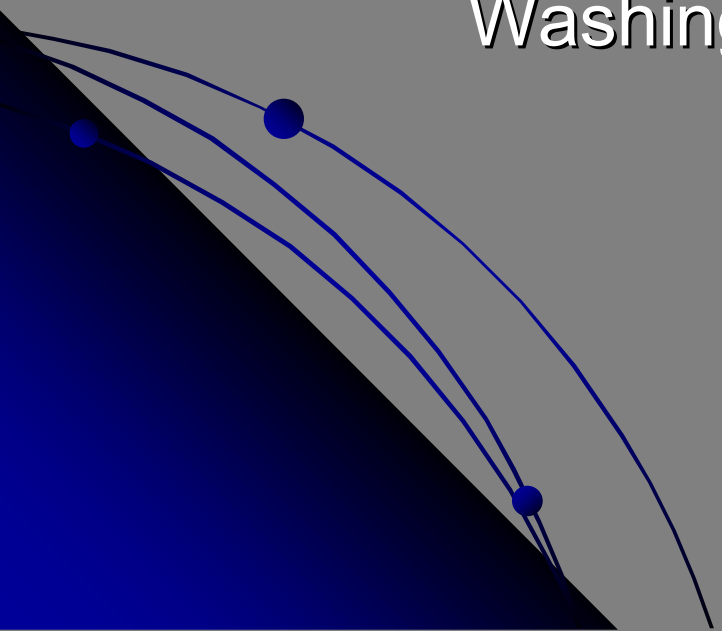


Special thanks to:

Jim Kennedy

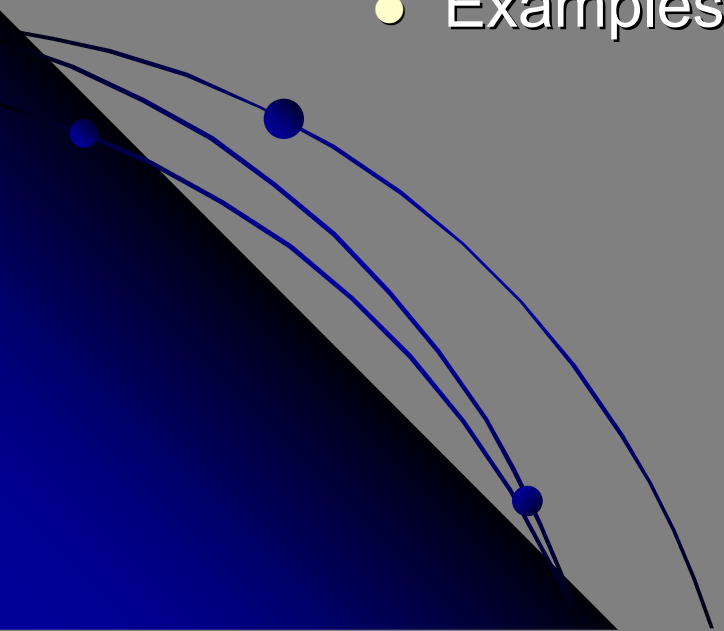
Bureau of Labor Statistics

Washington, DC



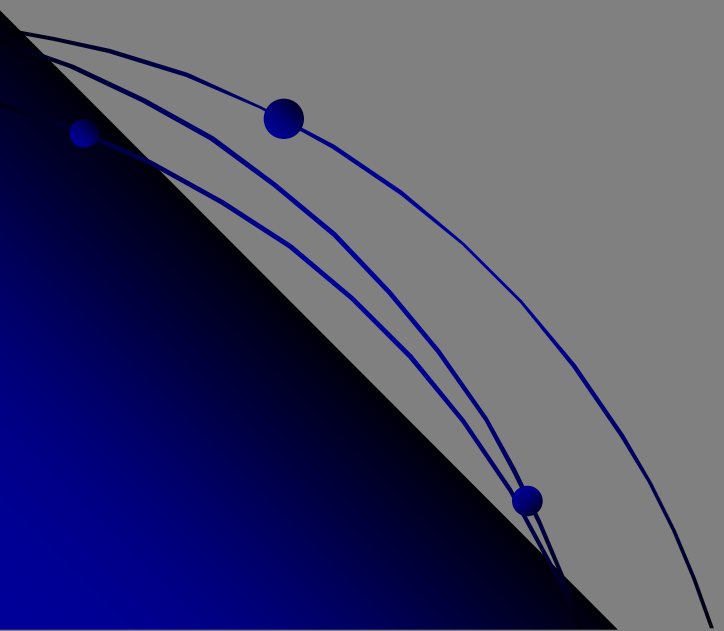
Outline of Presentation

- A Social Society Paradigm Tour
- A Brief Tour of Evolutionary Computation
- Introduction to Particle Swarm Optimization
- Evolving Fuzzy Systems
- Evolving Artificial Neural Networks
- Examples of Recent Applications



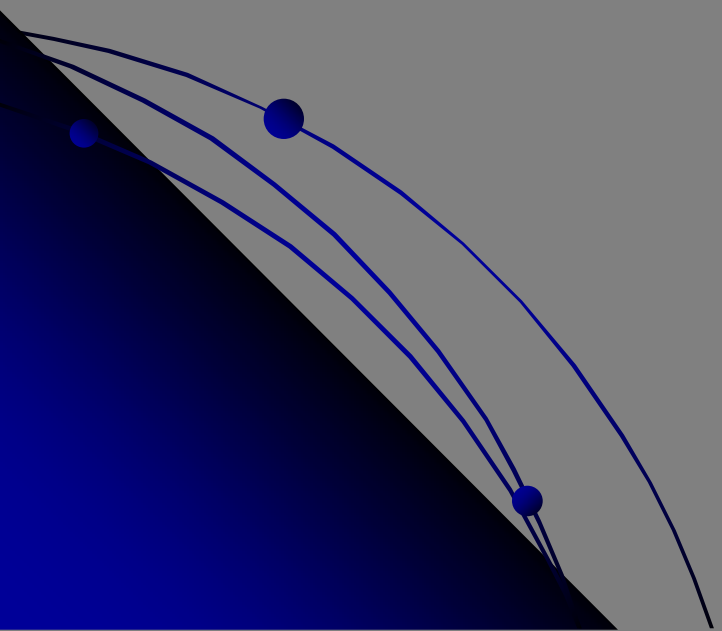
A Social Psychology Paradigm Tour

- Latané's dynamic social impact theory
- Axelrod's culture model
- Kennedy's adaptive culture model



Latané's Dynamic Social Impact Theory

- Behaviors of individuals can be explained in terms of the self-organizing properties of their social system
- Clusters of individuals develop similar beliefs
- Subpopulations diverge from one another (polarization]



Dynamic Social Impact Theory Characteristics

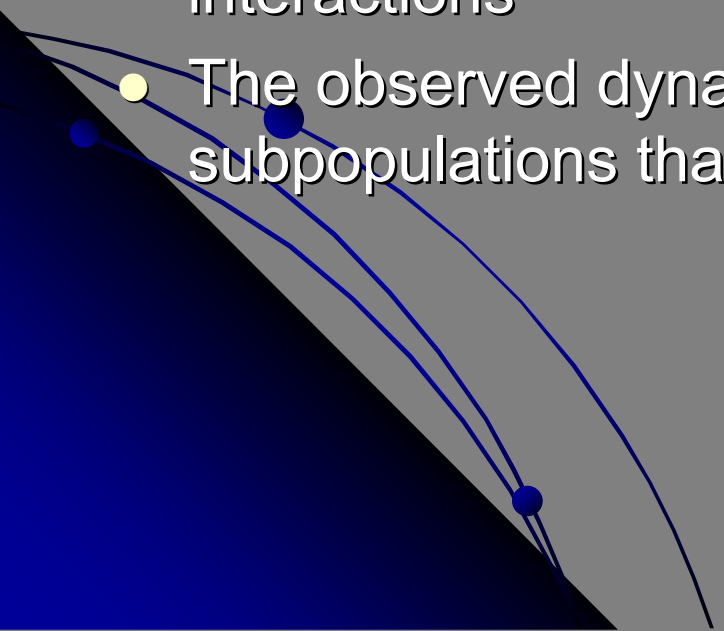
- Consolidation: *Opinion diversity is reduced as individuals are exposed to majority arguments*
- Clustering: *Individuals become more like their neighbors in social space*
- Correlation: *Attitudes that were originally independent tend to become associated*
- Continuing diversity: *Clustering prevents minority views from complete consolidation*

Dynamic Social Impact Theory: Summary

- Individuals influence one another, and in doing so become more similar
- Patterns of belief held by individuals tend to correlate within regions of a population
- This model is consistent with findings in the fields of social psychology, sociology, economics, and anthropology.

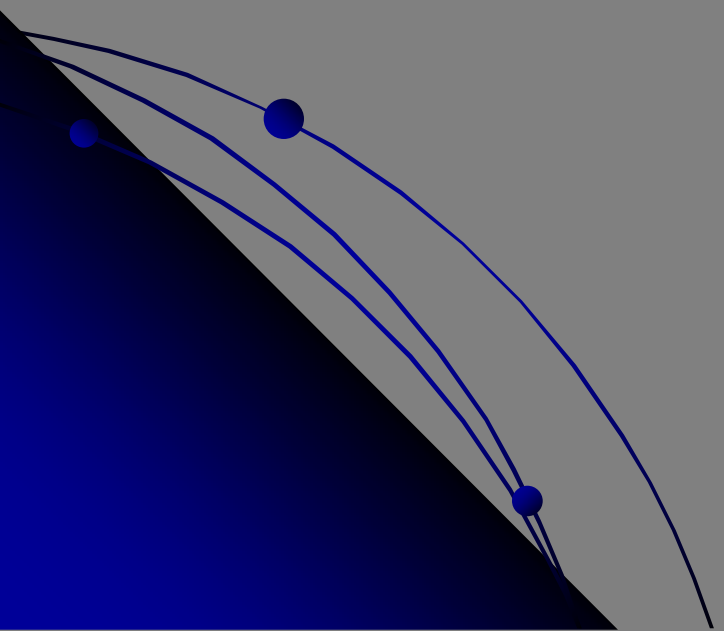
Axelrod's Culture Model

- Populations of individuals are pictured as strings of symbols, or "features"
- Probability of interaction between two individuals is a function of their similarity
- Individuals become more similar as a result of interactions
- The observed dynamic is *polarization* , homogeneous subpopulations that differ from one another



Kennedy's Adaptive Culture Model

- No effect of similarity on probability of interaction
- The effect of similarity is negative, in that it is *dis*similarity that creates boundaries between cultural regions
- Interaction occurs if *fitnesses* are different

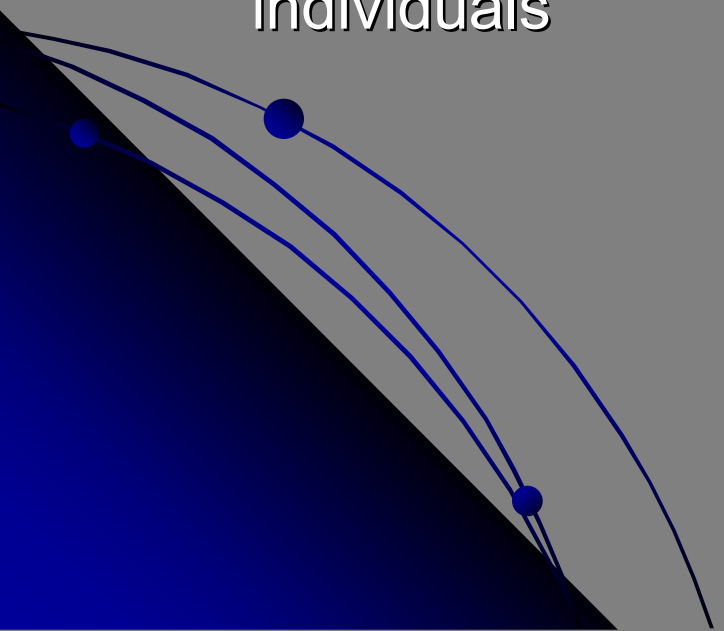


Culture and Cognition Summary

- Individuals searching for solutions learn from the experiences of others (individuals learn from their neighbors)
- An observer of the population perceives phenomena of which the individuals are the parts (individuals that interact frequently become similar)
- Culture affects the performance of individuals that comprise it (individuals gain benefit by imitating their neighbors)

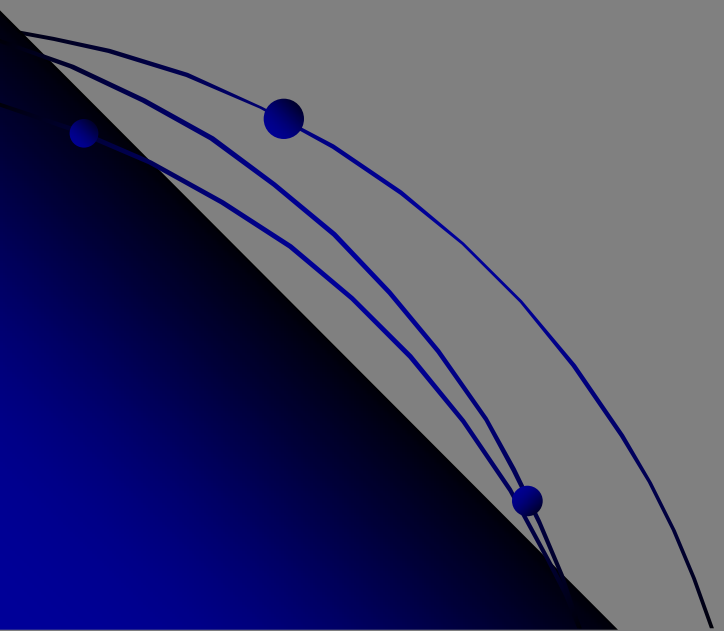
So, what about intelligence?

- Social behavior increases the ability of an individual to adapt
- There is a relationship between adaptability and intelligence
- Intelligence arises from interactions among individuals



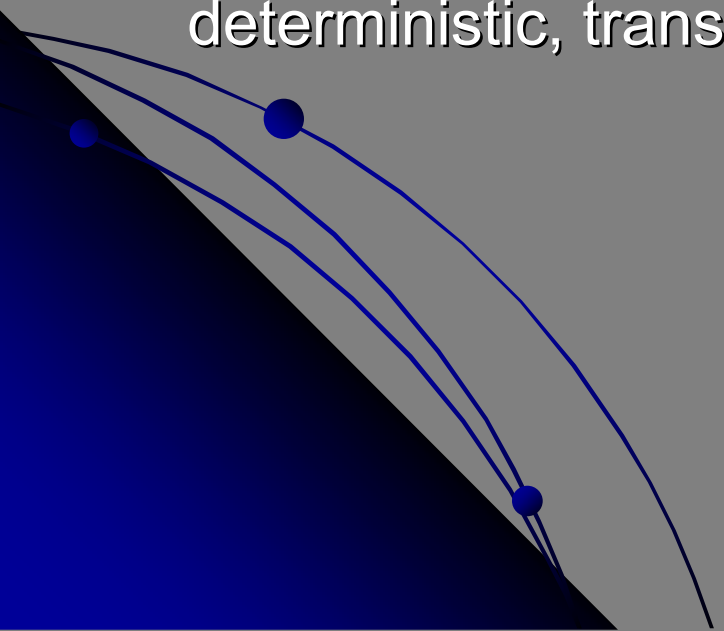
A Brief Tour of Evolutionary Computation

- *Evolutionary computation:* Machine learning optimization and classification paradigms roughly based on mechanisms of evolution such as biological genetics and natural selection



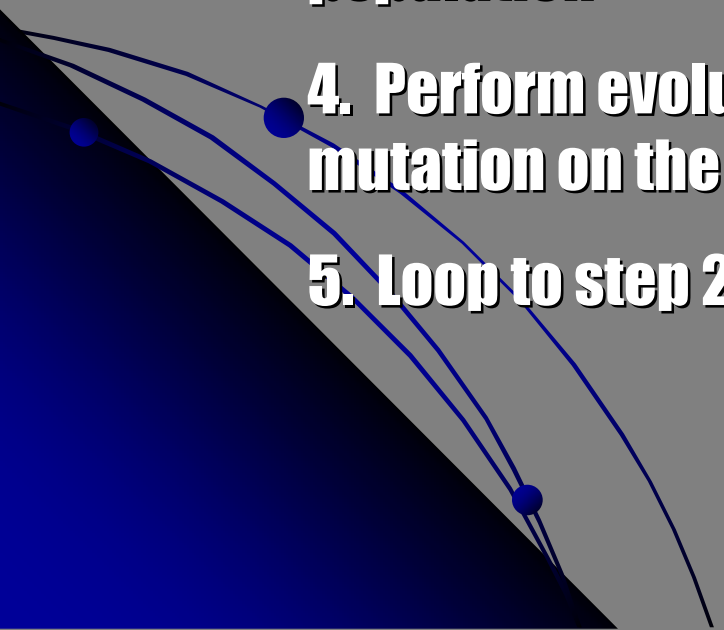
Features of Evolutionary Computation (EC) Paradigms

- EC paradigms utilize a population of points (potential solutions) in their search
- EC paradigms use direct “fitness” information instead of function derivatives or other related knowledge
- EC paradigms use probabilistic, rather than deterministic, transition rules



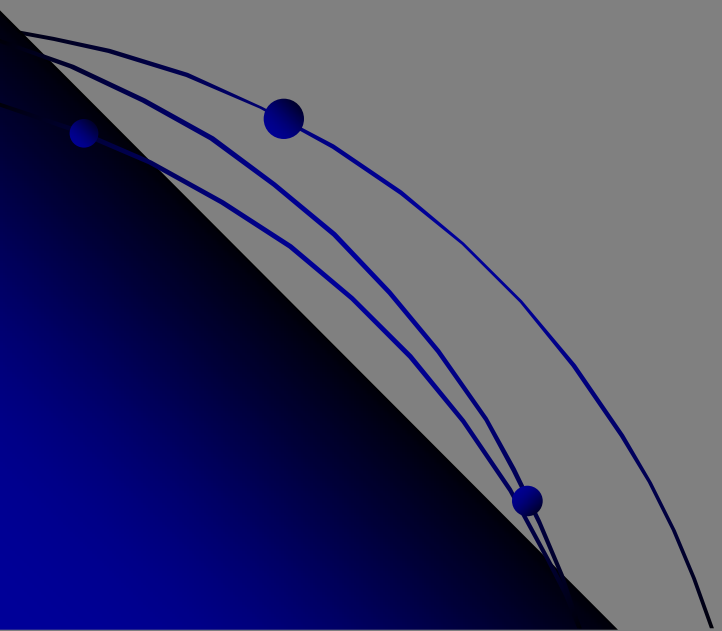
Evolutionary Computation Algorithms

1. Initialize the population
2. Calculate the fitness of each individual in the population
3. Reproduce selected individuals to form a new population
4. Perform evolutionary operations such as crossover and mutation on the population
5. Loop to step 2 until some condition is met



Evolutionary Computation Paradigms

- Genetic algorithms (GAs) - John Holland
- Evolutionary programming (EP) - Larry Fogel
- Evolution strategies (ES) - I. Rechenberg
- Genetic programming (GP) - John Koza
- Particle swarm optimization (PSO) - Kennedy & Eberhart

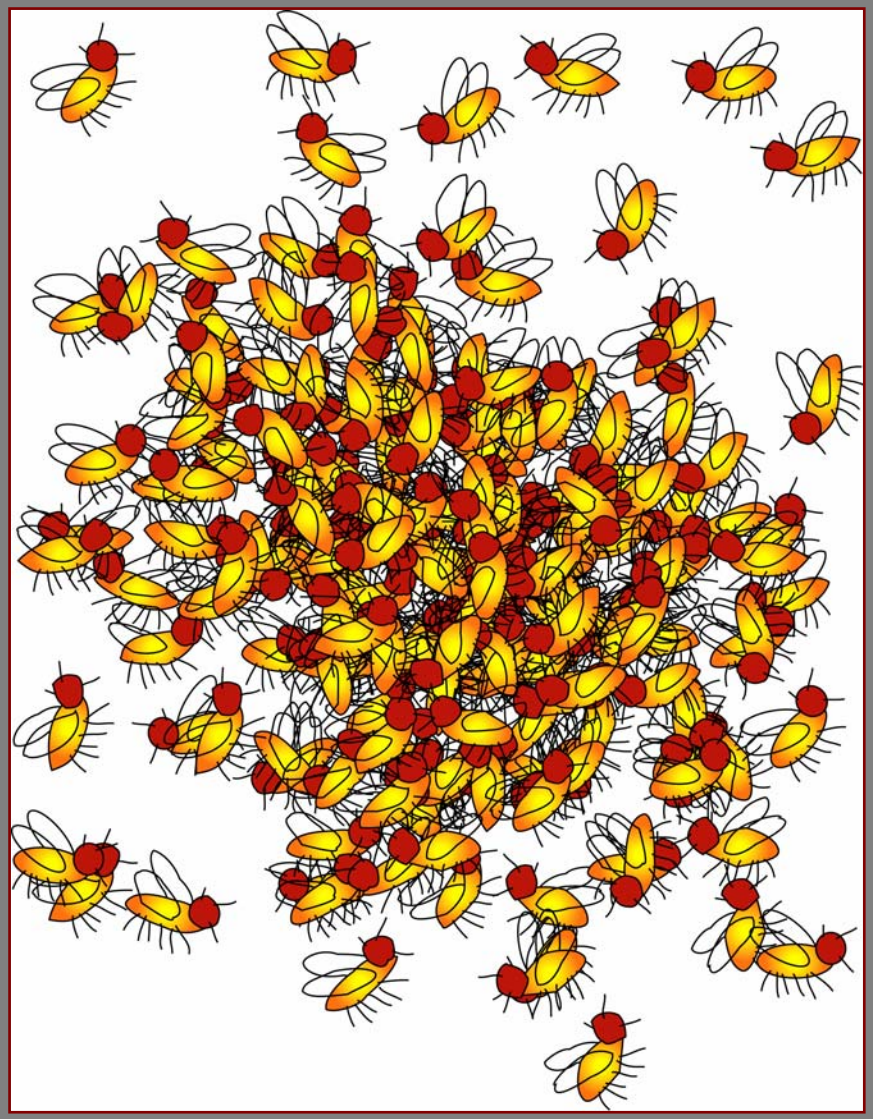


SWARMS

Coherence without choreography

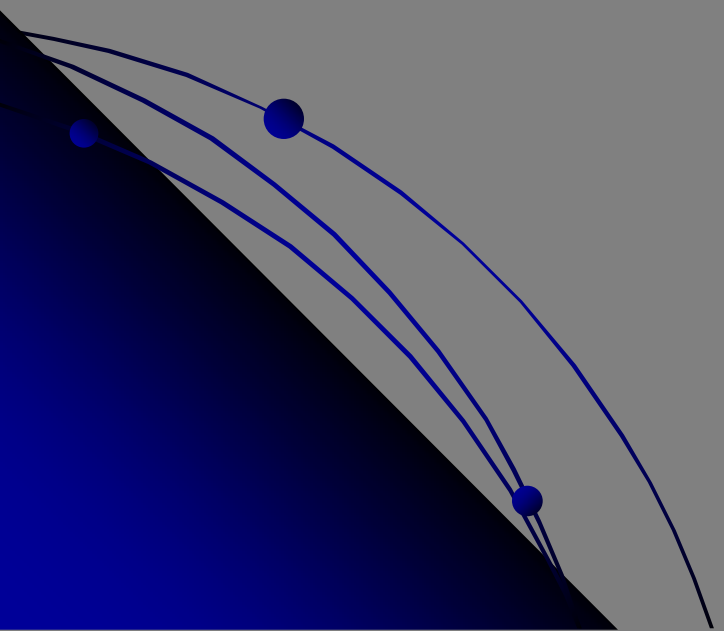
Bonabeau, Millonas, J.-L. Deneubourg, Langton, etc.

- Particle swarms (physical position not a factor)



Intelligent Swarm

- A population of interacting individuals that optimizes a function or goal by collectively adapting to the local and/or global environment
- Swarm intelligence ≅ collective adaptation

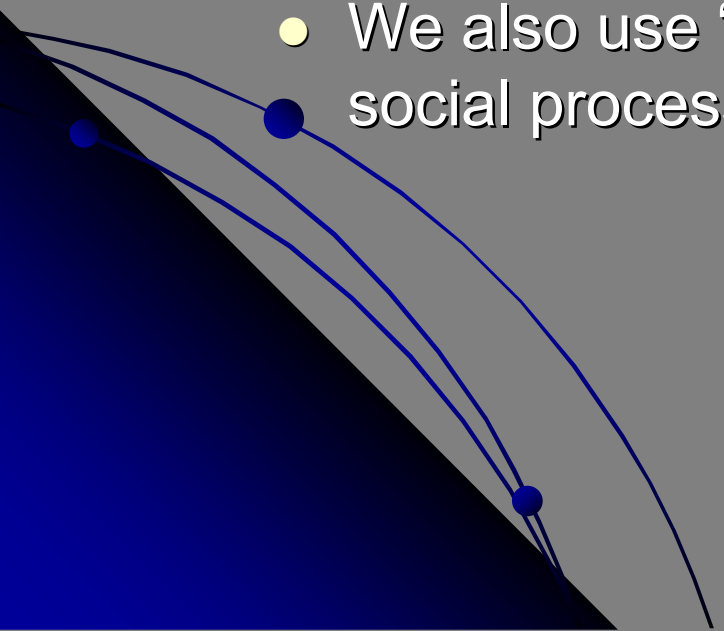


Basic Principles of Swarm Intelligence
(Mark Millonas, Santa Fe Institute)

- Proximity principle: the population should be able to carry out simple space and time computations
- Quality principle: the population should be able to respond to quality factors in the environment
- Diverse response principle: the population should not commit its activities along excessively narrow channels
- Stability principle: the population should not change its mode of behavior every time the environment changes
- Adapability principle: the population must be able to change behavior mode when it's worth the computational price

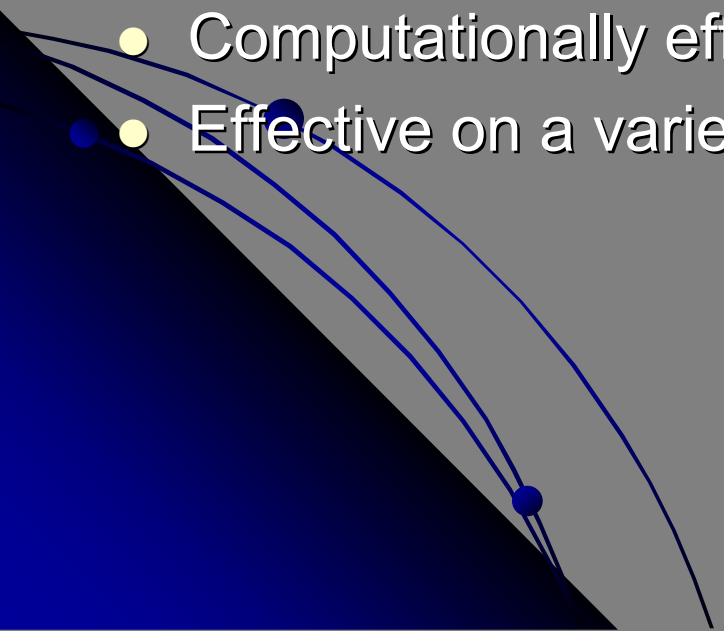
Introduction to Particle Swarm Optimization

- A “swarm” is an apparently disorganized collection (population) of moving individuals that tend to cluster together while each individual seems to be moving in a random direction
- We also use “swarm” to describe a certain family of social processes



Introduction to Particle Swarm Optimization (PSO), Continued

- A concept for optimizing nonlinear functions
- Has roots in artificial life and evolutionary computation
- Developed by Kennedy and Eberhart (1995)
- Simple in concept
- Easy to implement
- Computationally efficient
- Effective on a variety of problems



Evolution of PSO Concept and Paradigm

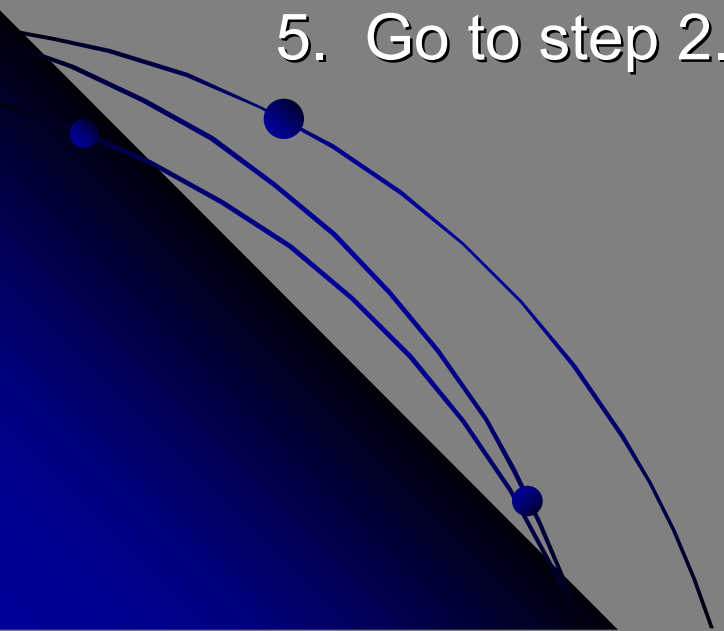
- Discovered through simplified social model simulation
- Related to bird flocking, fish schooling, and swarming theory
- Related to evolutionary computation; some similarities to genetic algorithms and evolution strategies
- Kennedy developed the “cornfield vector” for birds seeking food
- Bird flock became a swarm
- Expanded to multidimensional search
- Incorporated acceleration by distance
- Paradigm simplified

Features of Particle Swarm Optmization

- Population initialized by assigning random positions *and* velocities; potential solutions are then *flown* through hyperspace.
- Each particle keeps track of its “best” (highest fitness) position in hyperspace.
  - This is called “pbest” for an individual particle
  - It is called “gbest” for the best in the population
  - It is called “lbest” for the best in a defined neighborhood
- At each time step, each particle stochastically accelerates toward its pbest and gbest (or lbest).

Particle Swarm Optimization Process

1. Initialize population in hyperspace.
2. Evaluate fitness of individual particles.
3. Modify velocities based on previous best and global (or neighborhood) best.
4. Terminate on some condition.
5. Go to step 2.



PSO Velocity Update Equations

- Global version:

$$v_{id} = w_i v_{id} + c_1 rand()(p_{id} - x_{id}) + c_2 Rand()(p_{gd} - x_{id})$$

$$x_{id} = x_{id} + v_{id}$$

Where ***d*** is the dimension, c_1 and c_2 are positive constants, ***rand*** and ***Rand*** are random functions, and ***w*** is the inertia weight.

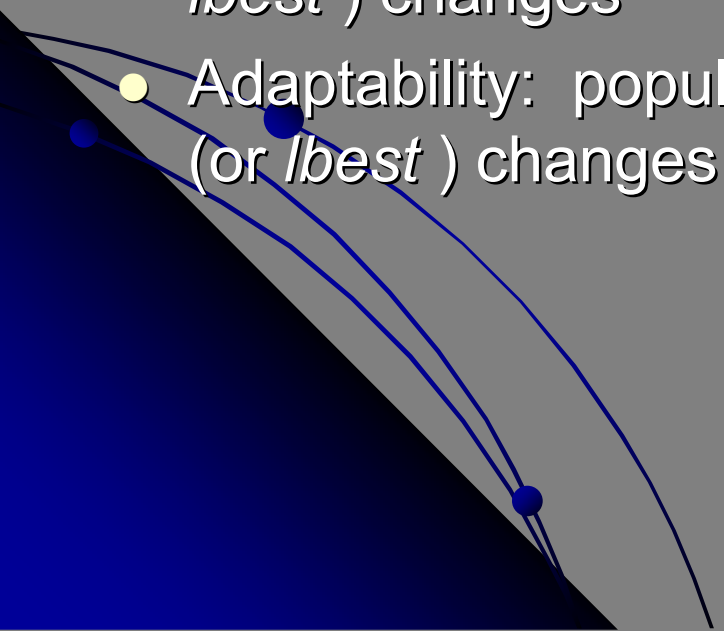
For neighborhood version, change p_{gd} to p_{ld}.

Further Details of PSO

- Performance of each particle measured according to a predefined fitness function.
- Inertia weight influences tradeoff between global and local exploration.
- Good approach is to reduce inertia weight during run (i.e., from 0.9 to 0.4 over 1000 generations)
- Usually set c_1 and c_2 to 2
- Usually set maximum velocity to dynamic range of variable

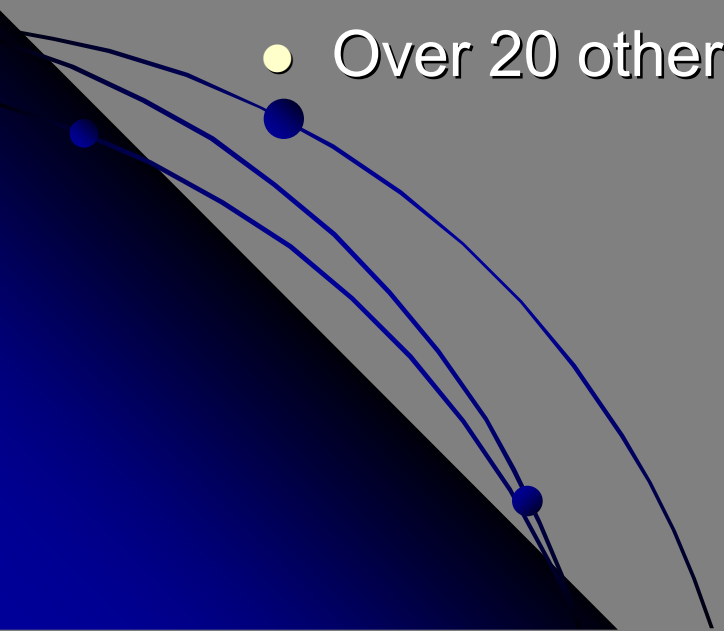
PSO Adherence to Swarm Intelligence Principles

- Proximity: *n*-dimensional space calculations carried out over series of time steps
- Quality: population responds to quality factors *pbest* and *gbest* (or *lbest*)
- Stability: population changes state only when *gbest* (or *lbest*) changes
- Adaptability: population *does* change state when *gbest* (or *lbest*) changes



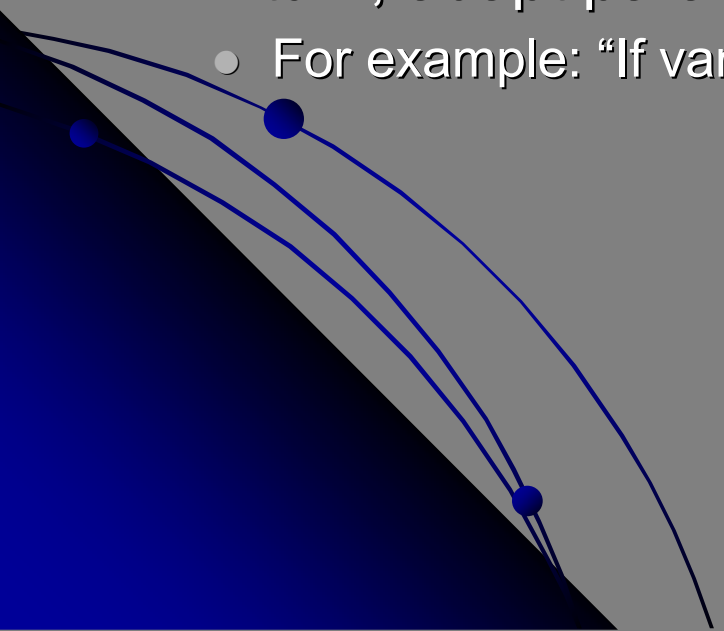
Benchmark Tests

- De Jong's test set
- Schaffer's F6 function
- Evolve neural network weights
  - Iris data set
  - Electric vehicle state of charge system
- Over 20 other benchmark functions tested



Evolving Fuzzy Systems

- Develop (evolve) fuzzy expert systems using evolutionary algorithms such as GA or PSO
  - Evolve rules
  - Evolve membership function types
  - Evolve membership function locations
- In turn, adapt parameters of the EA using fuzzy rules
  - For example: “If variance of fitness is low, set mutation rate high”



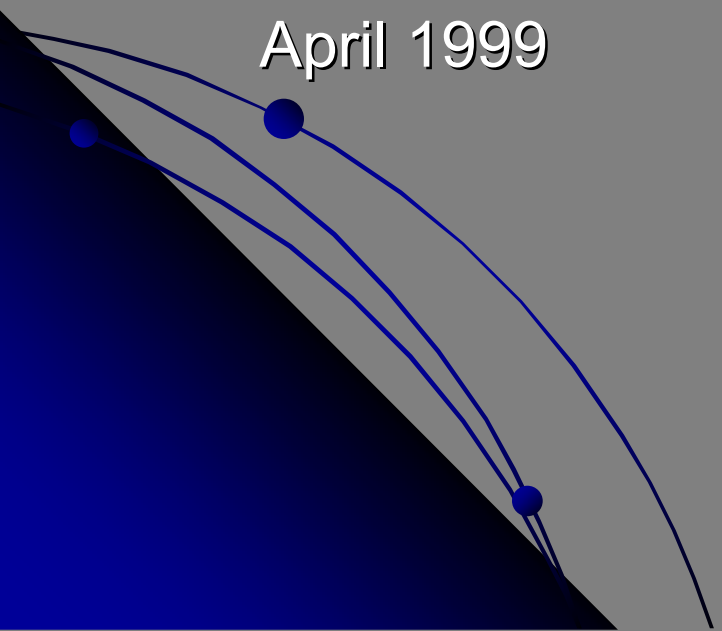
Journal Paper

"Implementation of Evolutionary Fuzzy Systems"

Authors: Shi, Eberhart, Chen

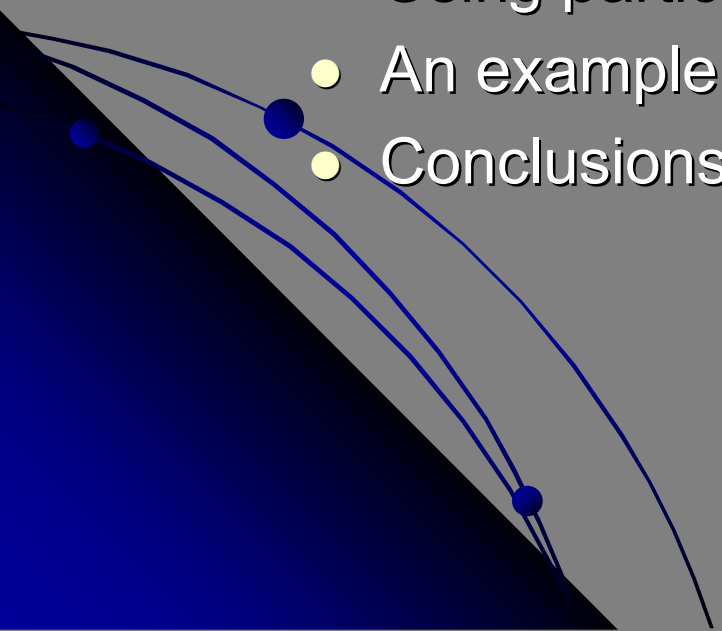
IEEE Transactions on Fuzzy Systems

April 1999



Evolving Artificial Neural Networks: Outline

- Introduction
- Definitions and review of previous work
- Advantages and disadvantages of previous approaches
- Using particle swarm optimization (PSO)
- An example application
- Conclusions



Introduction

- Neural networks are very good at some problems, such as mapping input vectors to outputs
- Evolutionary algorithms are very good at other problems, such as optimization
- Hybrid tools are possible that are better than either approach by itself
- Review articles on evolving neural networks: Schaffer, Whitley, and Eshelman (1992); Yao (1995); and Fogel (1998)
- Evolutionary algorithms usually used to evolve network weights, but sometimes used to evolve structures and/or learning algorithms

Evolving Neural Networks with Particle Swarm Optimization

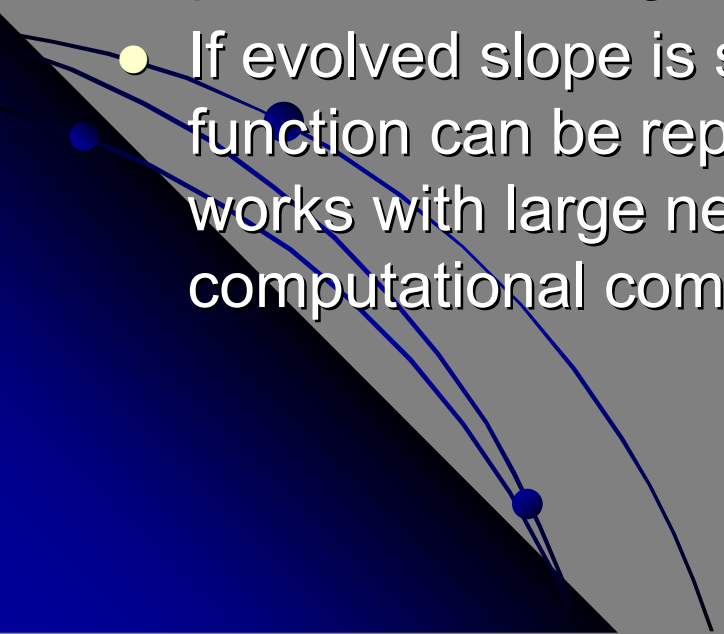
- Evolve neural network capable of being universal approximator, such as backpropagation or radial basis function network.
- In backpropagation, most common PE transfer function is sigmoidal function: $output = 1/(1 + e^{-input})$
- Eberhart, Dobbins, and Simpson (1996) first used PSO to evolve network weights (replaced backpropagation learning algorithm)
- PSO can also be used to indirectly evolve the structure of a network. An added benefit is that the preprocessing of input data is made unnecessary.

Evolving Neural Networks with Particle Swarm Optimization, Continued

- Evolve both the network weights ***and*** the slopes of sigmoidal transfer functions of hidden and output PEs.
- If transfer function now is: $\text{output} = 1/(1 + e^{-k*\text{input}})$ then we are evolving k in addition to evolving the weights.
- The method is general, and can be applied to other topologies and other transfer functions.
- Flexibility is gained by allowing slopes to be positive or negative. A change in sign for the slope is equivalent to a change in signs of all input weights.

Evolving the Network Structure with PSO

- If evolved slope is sufficiently small, sigmoidal output can be clamped to 0.5, and hidden PE can be removed. Weights from bias PE to each PE in next layer are increased by one-half the value of the weight from the PE being removed to the next-layer PE. PEs are thus pruned, reducing network complexity.
- If evolved slope is sufficiently high, sigmoid transfer function can be replaced by step transfer function. This works with large negative or positive slopes. Network computational complexity is thus reduced.

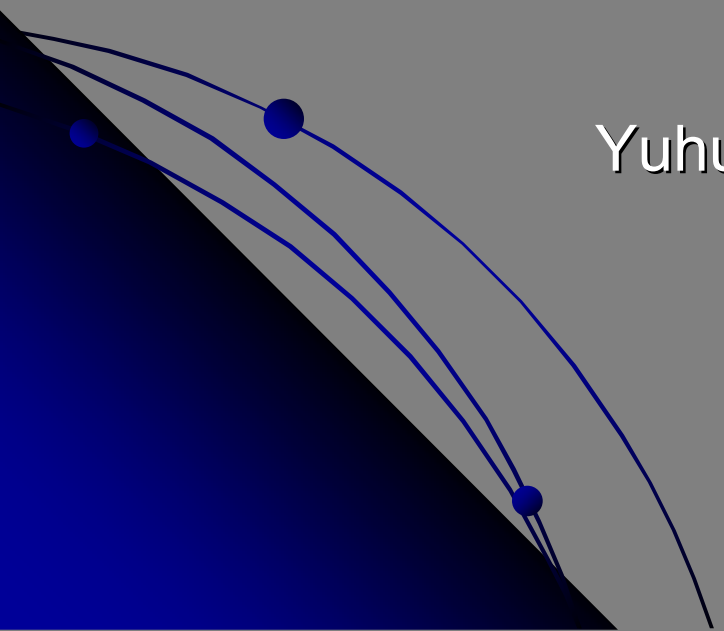


Evolving the Network Structure with PSO, Continued

- Since slopes can evolve to large values, input normalization is generally not needed. This simplifies applications process and shortens development time.
- The PSO process is continuous, so neural network evolution is also continuous. No sudden discontinuities exist such as those that plague other approaches.
- This approach is now protected by a U. S. Patent

Tracking and Optimizing Dynamic Systems with Particle Swarms

Acknowledge:
Yuhui Shi and Xiaohui Hu



computelligence

Outline

- Brief review of particle swarm optimization
- Types of dynamic systems
- Practical application requirements
- Previous work
- Experimental design
- Results
- Conclusions and future effort

computelligence

Original Version with Inertia Weight

$$v_{id} = w_i v_{id} + c_1 rand()\left(p_{id} - x_{id}\right) + c_2 Rand()\left(p_{gd} - x_{id}\right)$$

$$x_{id} = x_{id} + v_{id}$$

Where ***d*** is the dimension, c_1 and c_2 are positive constants, ***rand*** and ***Rand*** are random functions, and ***w*** is the inertia weight. For neighborhood version, change p_{gd} to p_{ld}.

computelligence

Constriction Factor Version

$$v_{id} = K * [v_{id} + c_1 * rand() * (p_{id} - x_{id}) + c_2 * Rand() * (p_{gd} - x_{id})]$$

$$K = \frac{2}{\left| 2 - \phi - \sqrt{\phi^2 - 4\phi} \right|}$$

where $\phi = c_1 + c_2,\ \phi > 4$

(ϕ was set to 4.1, so K = .729)

computelligence

Dynamic System Types

- Location of optimum value can change
- Optimum value can vary
- Number of optima can change
- Combinations of the above can occur

In this project, we varied the location of the optimum.

computelligence

Practical Application Requirements

- Few practical problems are static; most are dynamic
- Most time is spent re-optimizing (re-scheduling, etc.)
- Many systems involve machines and people
  - These systems have inertia
  - 10-100 seconds often available for re-optimization
- Eberhart's Law of Sufficiency applies: If the solution is good enough, fast enough, and cheap enough, then it is sufficient

computelligence

Previous Work

- Testing Parabolic Function

$$error = \sum_{i=1}^{N} (x_i - offset)^2$$

  - Offset = offset + severity
  - Severity 0.01, .1, .5
  - 2000 evaluations per change
  - 3 dimensions, dynamic range –50 to +50

computelligence

Previous Work: References

- Angeline, P.J. (1997) Tracking extrema in dynamic environments. *Proc. Evol. Programming VI*, Indianapolis, IN, Berlin: Springer-Verlag, pp. 335-345
- Bäck, T. (1998). On the behavior of evolutionary algorithms in dynamic environments. *Proc. Int. Conf. on Evol. Computation*, Anchorage, AK. Piscataway, NJ: IEEE Press, pp. 446-451

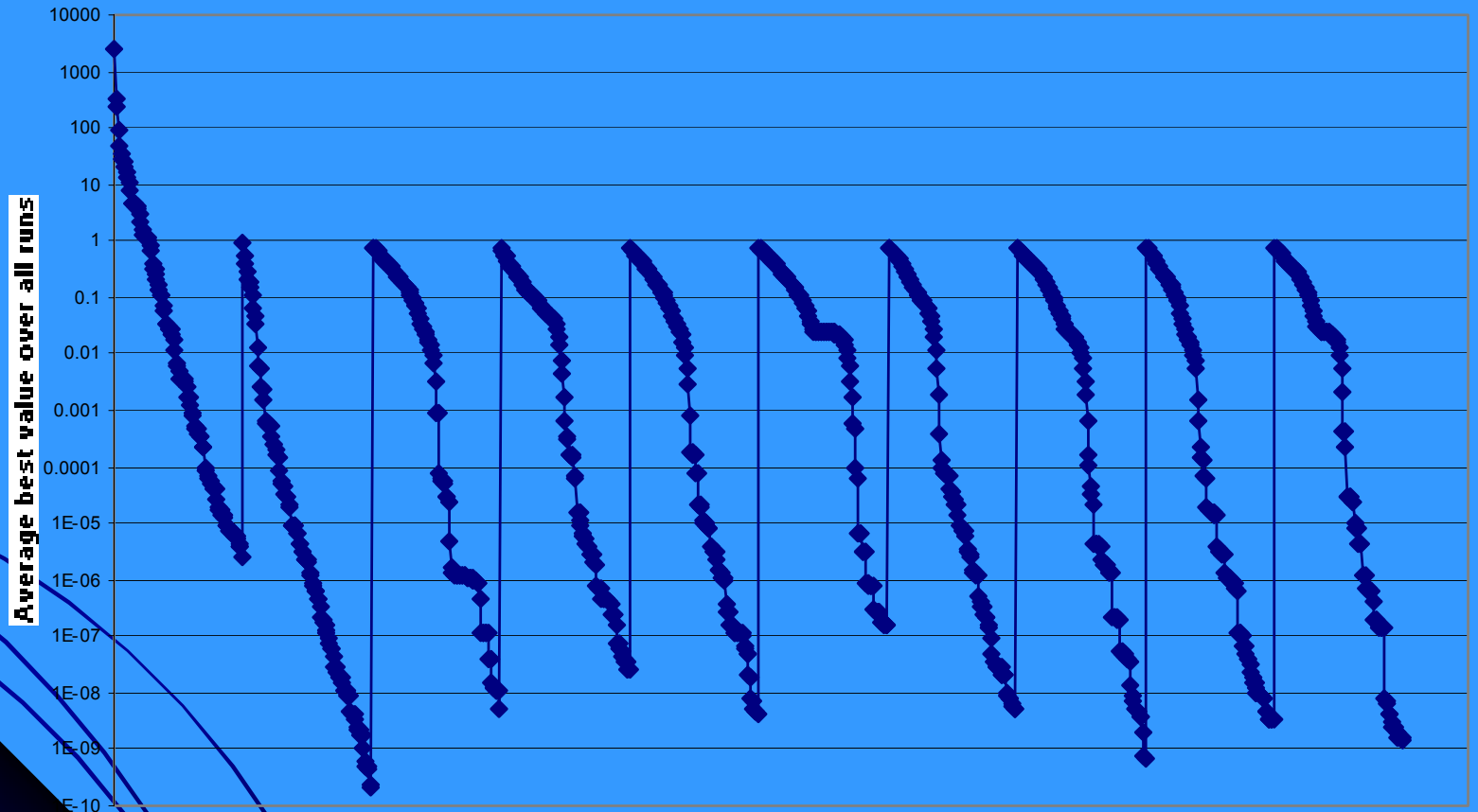
computelligence

Experimental Design

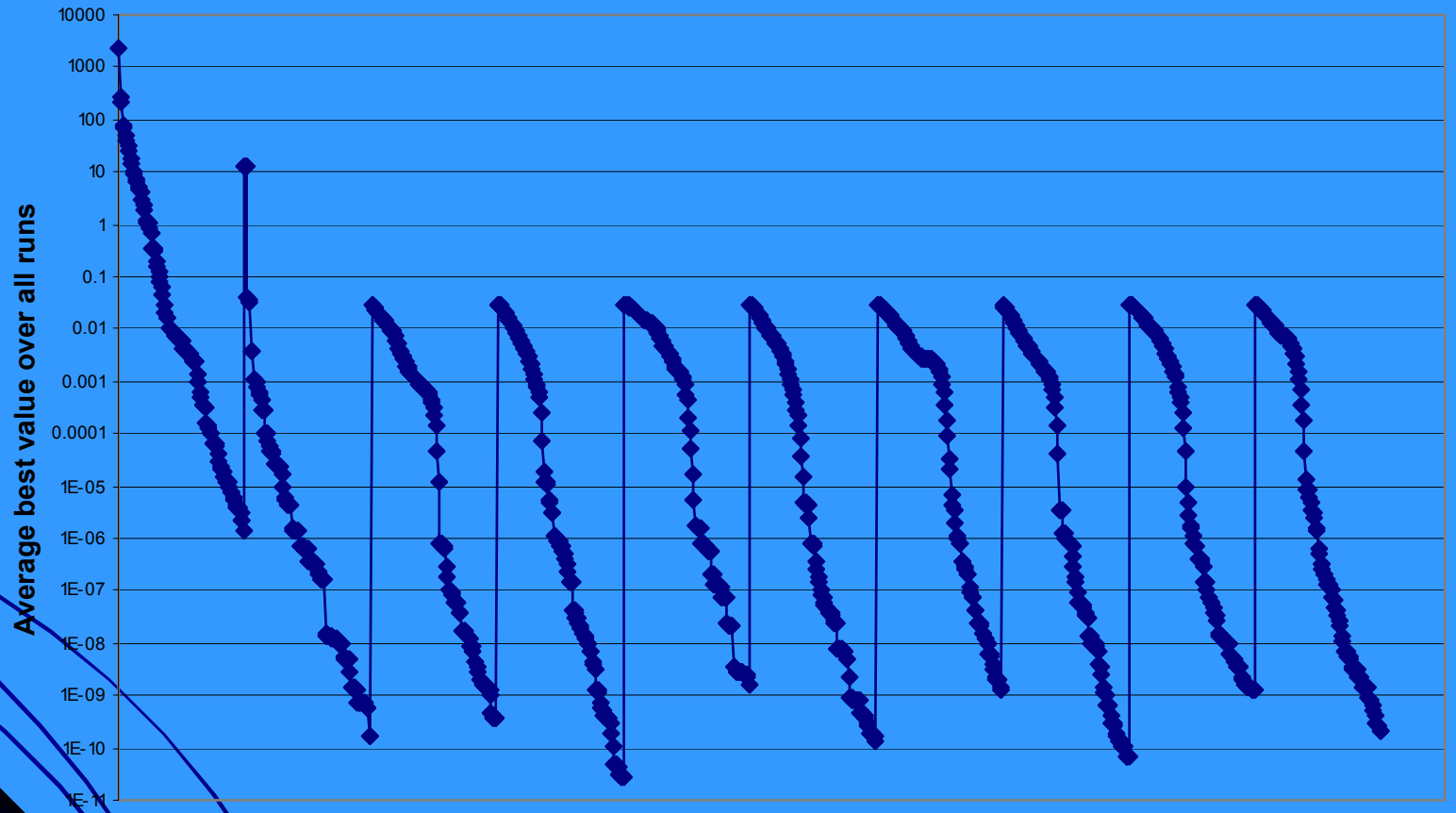
- Two possibilities with swarm
  - Continue on from where we were
  - Re-initialize the swarm
- Inertia weight of [0.5+(Rnd/2.0)] used
- 20 particles; update interval of 100 generations
- When change occurred:
  - Retained the position of each particle
  - Reset values of pbest (also of gbest)

computelligence

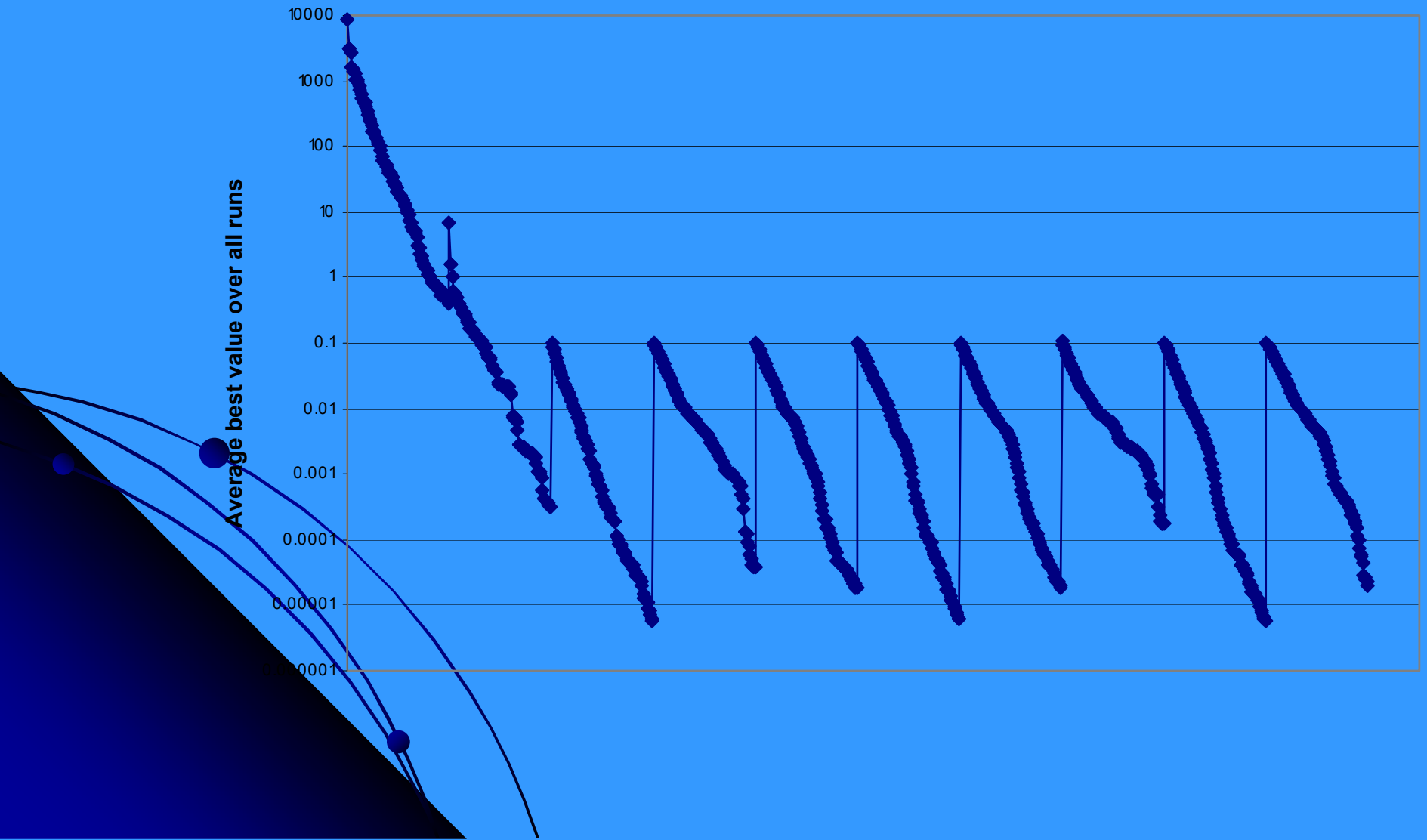
PSO average best over all runs
Severity = 0.5
Three dimensions



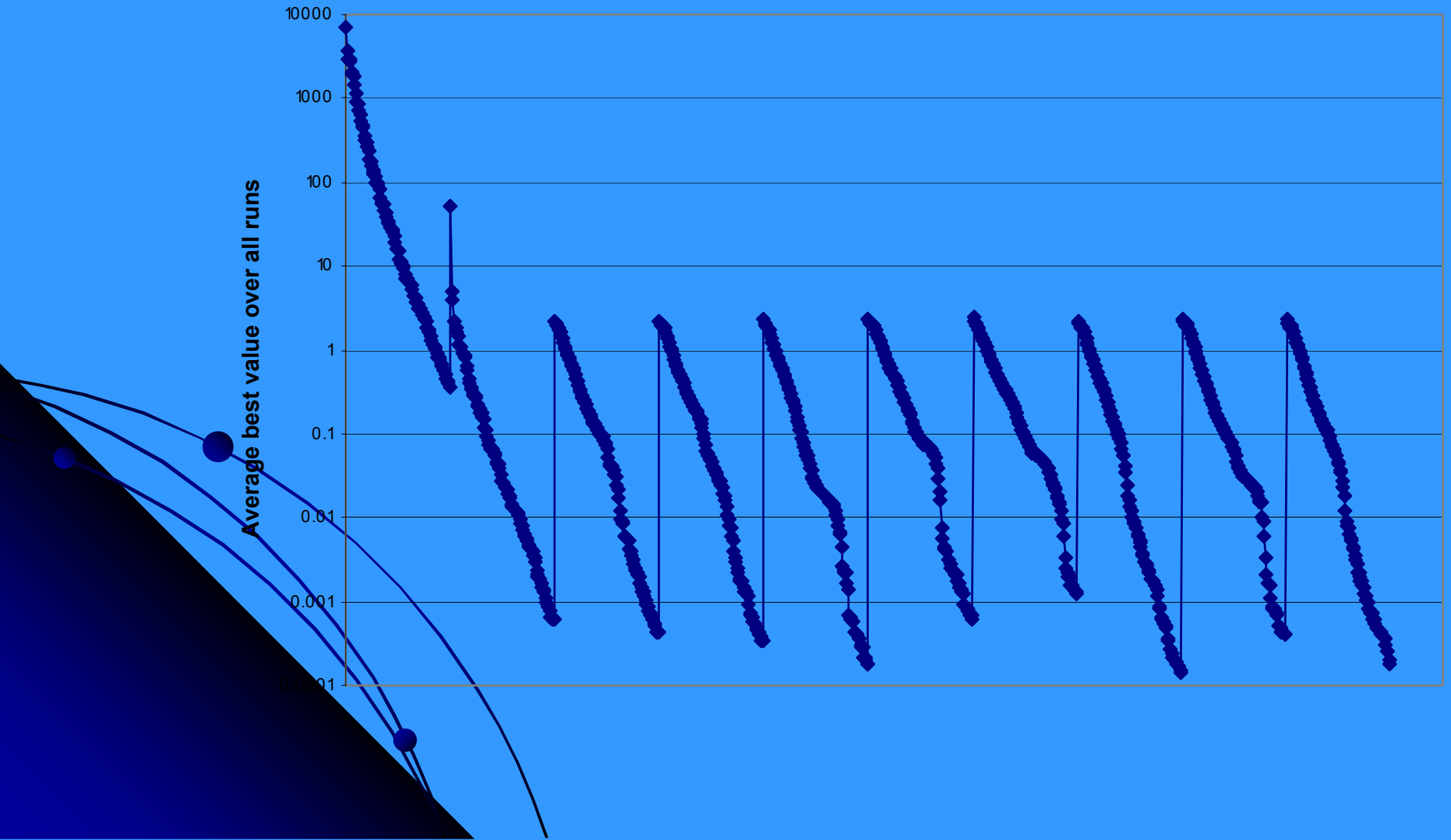
PSO average best over all runs
Severity = 0.1
Three dimensions



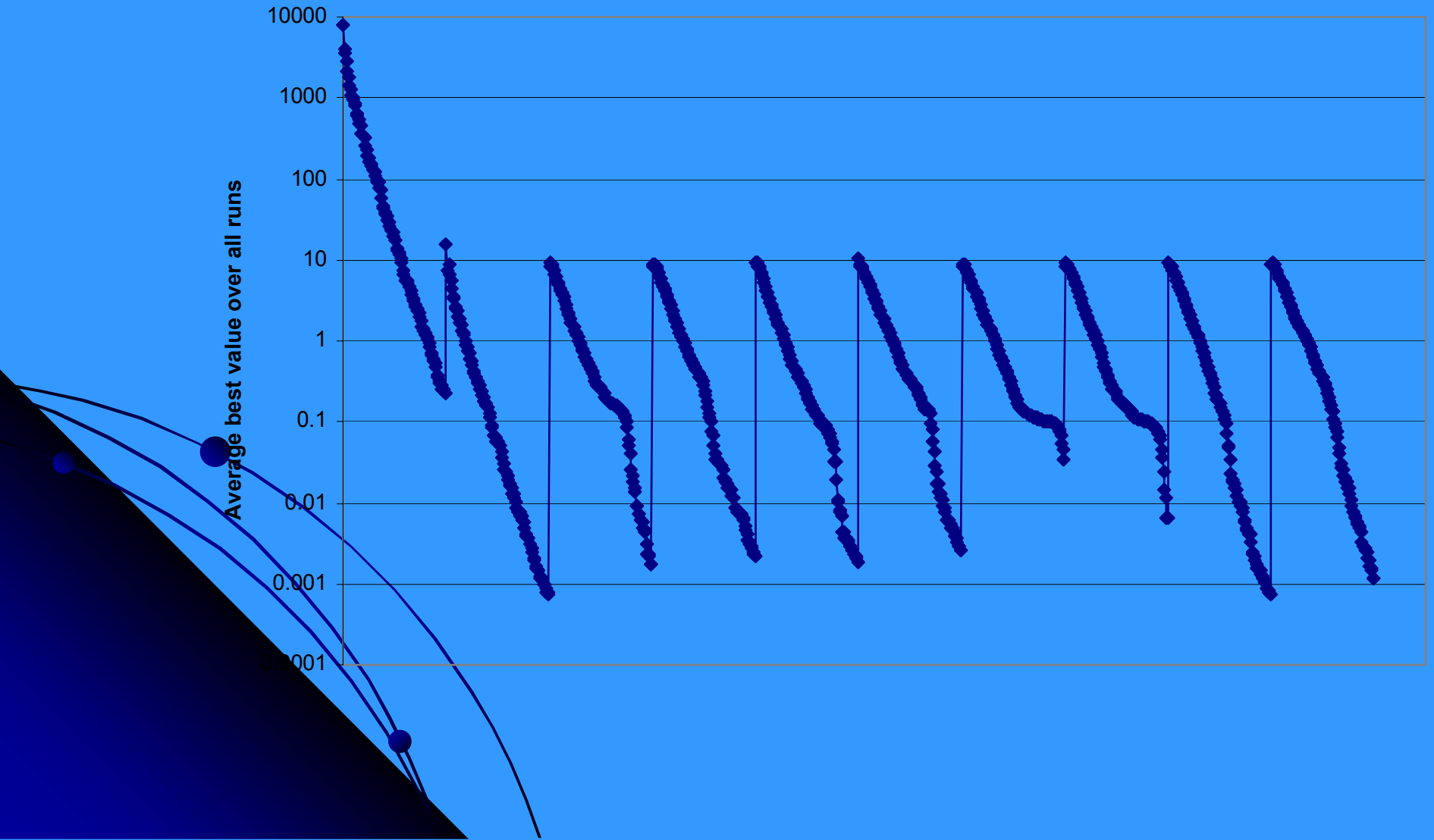
PSO average best over all runs
Severity = 0.1
10 dimensions



PSO average best over all runs
Severity = 0.5
10 dimensions



PSO average best over all runs
Severity = 1.0
10 dimensions

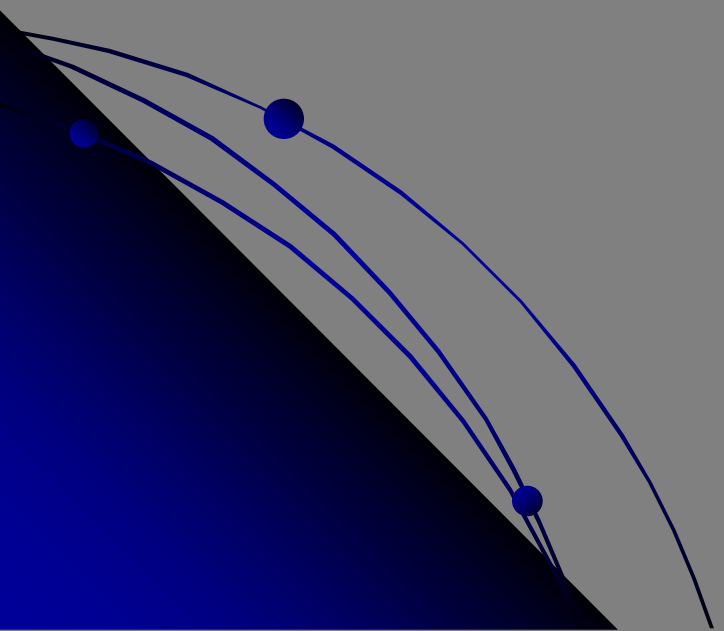


Comparison of Results: Error Values Obtained in 2000 Evaluations

	Severity 0.1	Severity 0.5
Angeline	$5 \times 10^{-4} – 10^{-3}$	0.01-0.10
Bäck	2×10^{-5}	10^{-3}
Eberhart & Shi	$10^{-10} - 10^{-9}$	$10^{-9} – 10^{-8}$

Conclusions and Future Efforts

- Our results, including those in 10 dimensions and with severity = 1, are promising
- We are applying approach to other benchmark functions, and to practical logistics applications

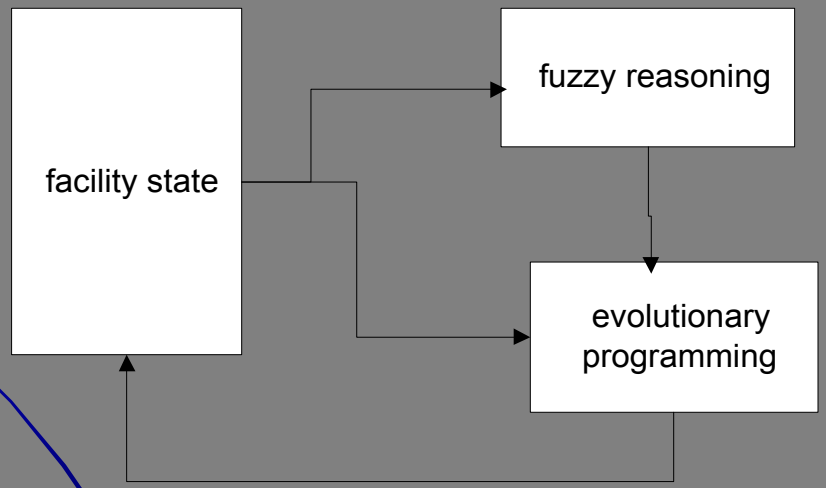


Example Application: Reactive Power and Voltage Control

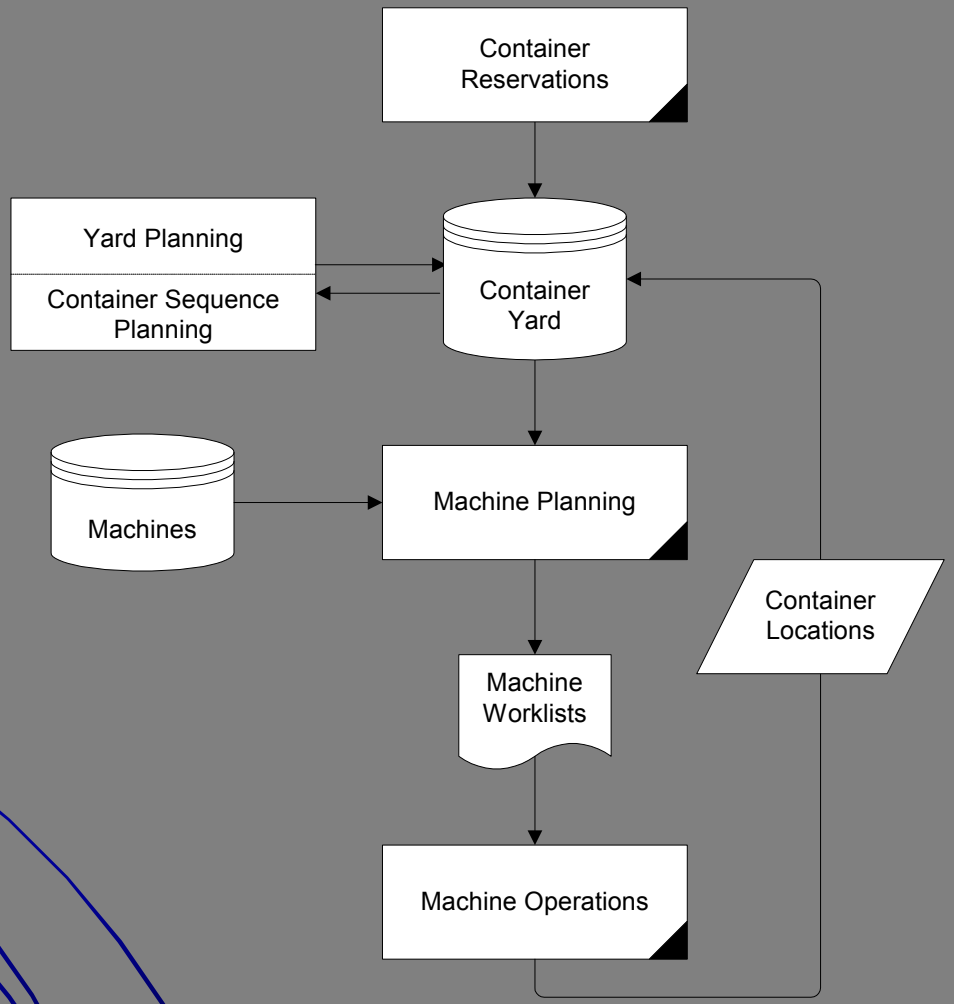
- Japanese electric utility
- PSO used to determine control strategy
  - Continuous and discrete control variables
- Hybrid binary/real-valued version of PSO developed
- System voltage stability achieved using a continuation power flow technique

Scheduling System for Integrated Automated Container Terminal

- Objective - develop planning and scheduling algorithm for fully integrated automated container terminals
- Approach - Fuzzy system and evolutionary programming



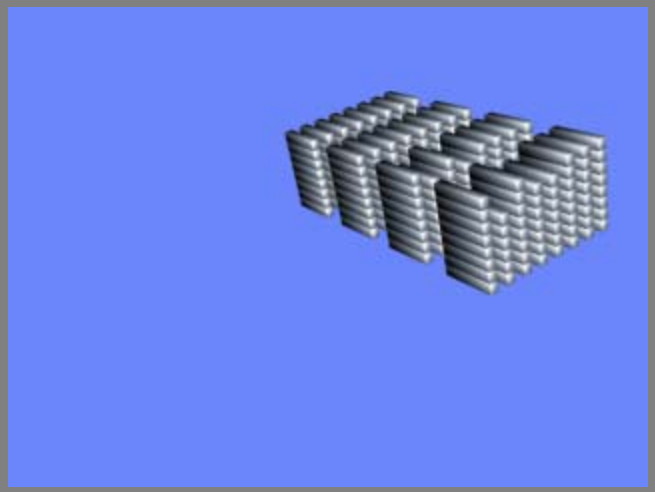
Scheduling System for IACT – Workflow



Container Planning Sequences

- **500** Containers
- Move from yard to staging area along the berth
- Planning results
- Number of movements:

<600

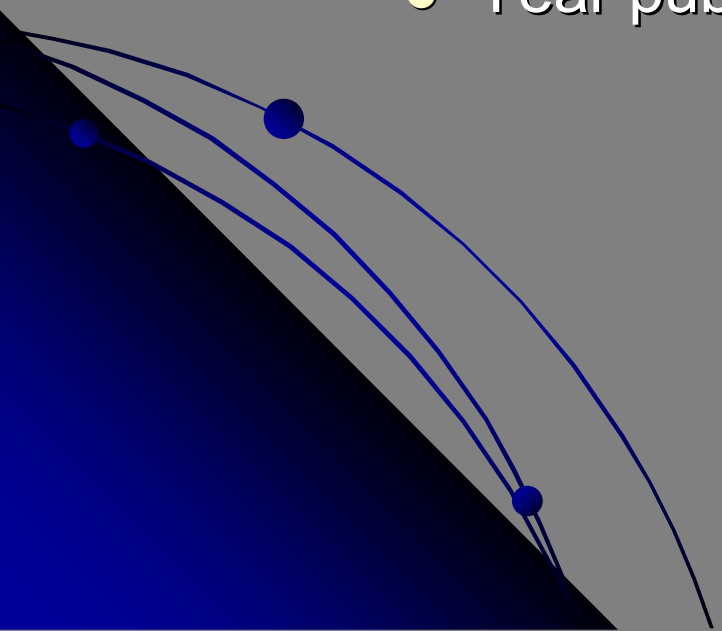


More Examples of Recent Applications

- Scheduling (Marine Corps logistics)
- Manufacturing (Product content combination optimization)
- Figure of merit for electric vehicle battery pack
- Medical analysis/diagnosis (Parkinson's disease and essential tremor)
- Human performance prediction (cognitive and physical)

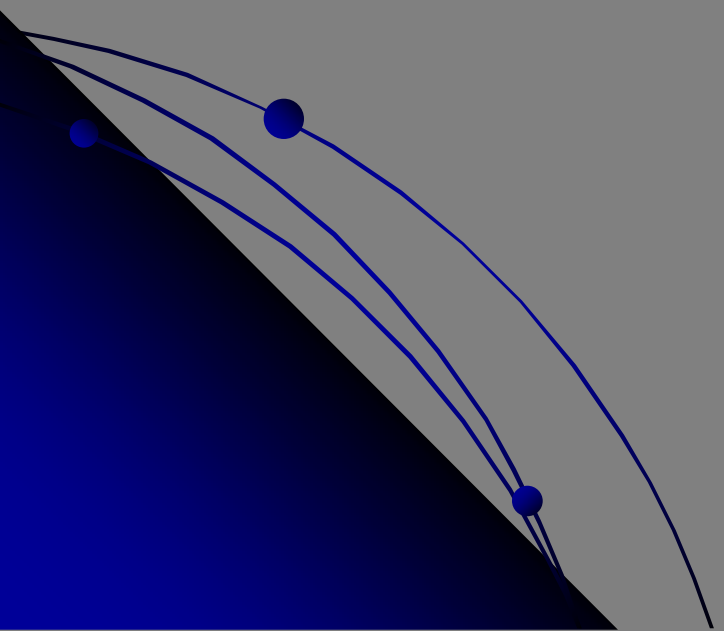
Original Book

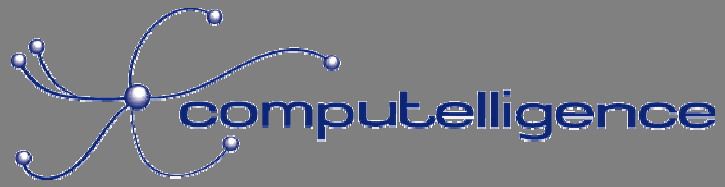
- Title: *Computational Intelligence PC Tools*
- Authors: Eberhart, Dobbins, and Simpson
- Publisher: Academic Press
- Year published: 1996

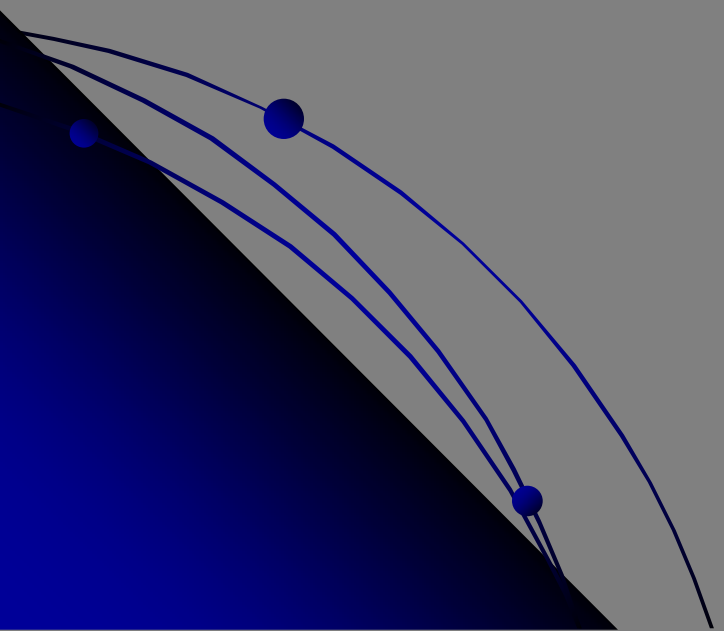


Recent Book

- Title: *Swarm Intelligence*
- Authors: Kennedy, Eberhart and Shi
- Publisher: Morgan Kaufmann division of Academic Press
- Publication date: 2001





New Book

Computational Intelligence: Concepts to Implementations, Eberhart and Shi, Morgan Kauffman, 2005.