CSE/ECE 848 Introduction to Evolutionary Computation

Module 3, Lecture 13, Part 2

Metaheuristics and Automatically Tuned EC Methods

Kalyanmoy Deb, ECE

Koenig Endowed Chair Professor

Self-Adaptive and Automatically Tuned EC Methods

- Part 1:
  - Popular Metaheuristics methods
  - Algorithmic equivalence among EC methods
  - Automated method for choosing hyperparameters or an algorithm!
- Part 2 (This Part):
  - Self-adaptive EC Methods
  - Automatic Parameter Tuning in EC Methods
- Part 3:
  - **irace**: Automatic Algorithm Configuration Method

Self-Adaptive EC Methods

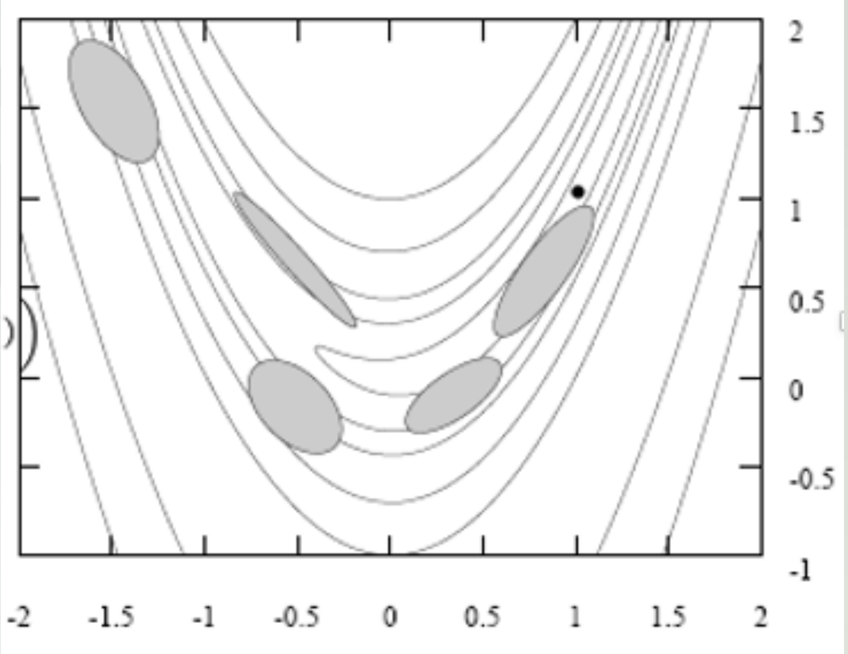
- Static EC Methods
  - All EC parameters are predefined and kept fixed during a run
- Adaptive EC Methods
  - Pre-planned adaptation: Popsize is a function of generation $N(t)$, penalty parameter increased with generation $R(t)$, mutation prob. reduces with generation $p_m(t)$
  - Offline study to check what exact form of variation is good!
- Self-Adaptive EC Methods
  - EC parameters change with current or trend in performance
  - Not known a priori, find on the fly
  - Popsize is a function of diversity of variable vectors, penalty parameter is a function of #infeasible solutions in the population, mutation prob. is dependent on the diversity in variables

Self-Adaptive Evolution Strategy

- Selection-Mutation strategy
  - Performance depends on mutation strength (σ)
- Mutation strength is updated automatically

$$
\begin{aligned}
\sigma_i^{(t+1)} &= \sigma_i^{(t)} \exp\left(\tau' N(0,1) + \tau N_i(0,1)\right), \\
\alpha_j^{(t+1)} &= \alpha_j^{(t)} + \beta_\alpha N_j(0,1), \\
\vec{x}^{(t+1)} &= \vec{x}^{(t)} + \vec{N}\left(\vec{0}, C(\vec{\sigma}^{(t+1)}, \vec{\alpha}^{(t+1)})\right),
\end{aligned}
$$

$\tau' \propto (2n)^{-1/2}$ $\tau \propto (2n^{1/2})^{-1/2}$ $\beta_\alpha = 0.0873$



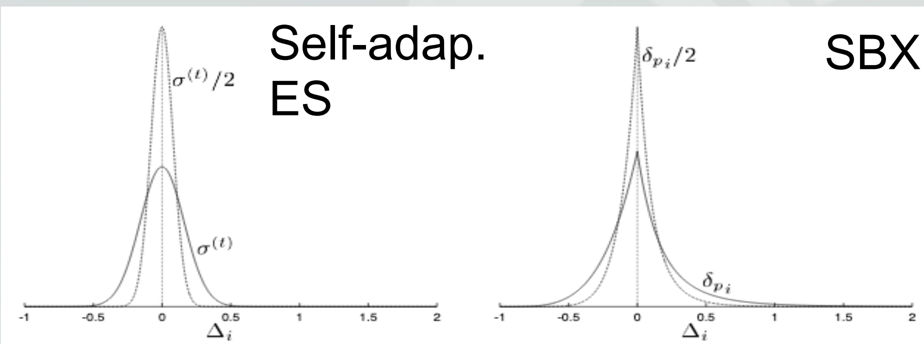
Problem-dependent way of learning how to create new solutions

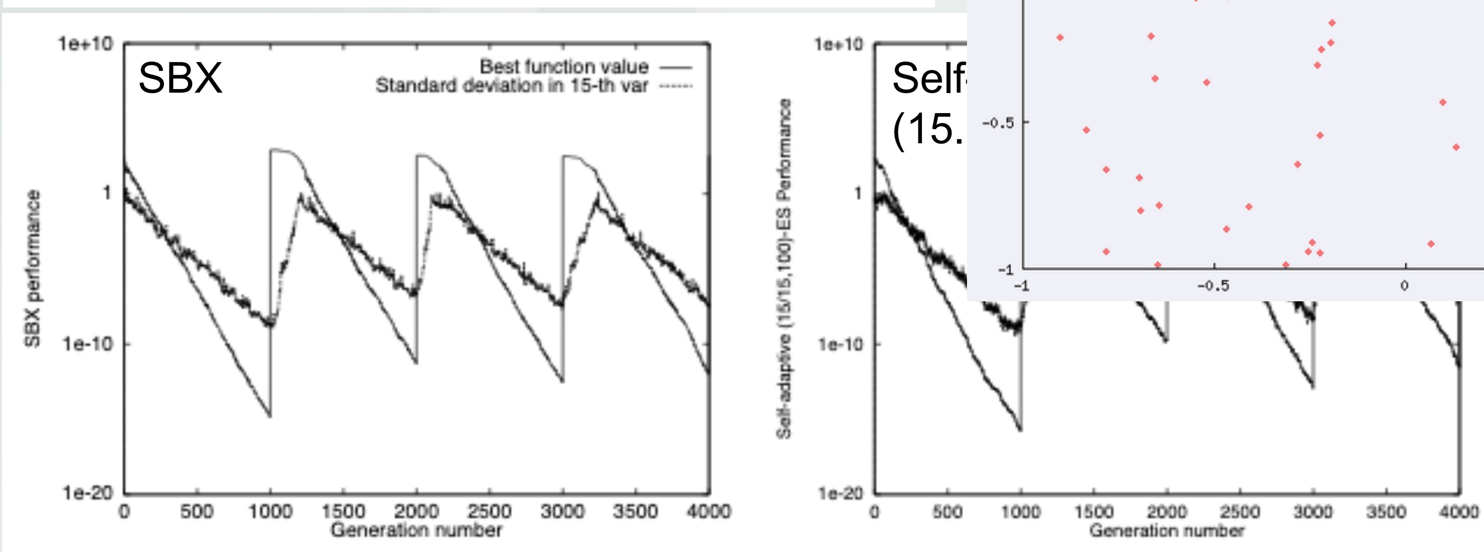
- Directions of improvements can be learnt

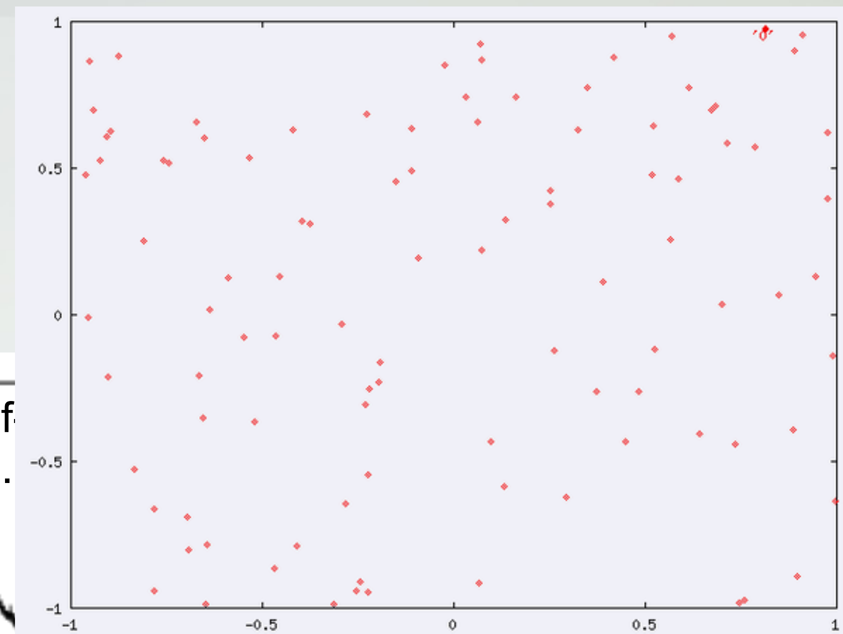
Reference: H.-P. Schwefel (1987). Collective phenomena in evolutionary systems, In P. Checkland and I. Kiss (Eds.) *Problems of Constancy and Change – the Complementarity of Systems Approaches to Complexity.* Budapest: International Society for General System Research, 1025—1033.

Self-Adaptive ES and RGA with SBX

- Difference between child and parent has similar distribution
  - Both have similar self-adaptive behavior
- Problem changes after 1,000 generations





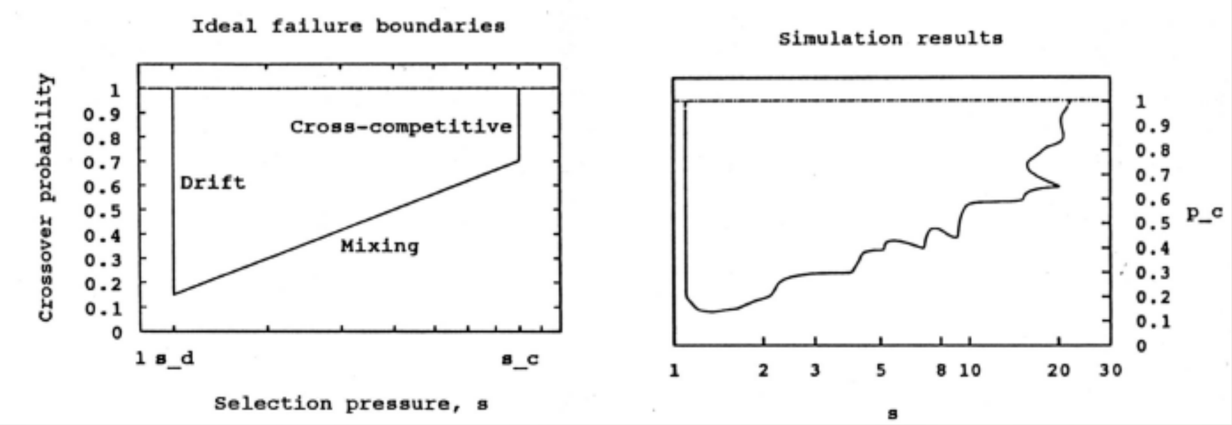


Self-Adaptation from Exploration-Exploitation Trade-off

- Exploitation: Emphasis of current best solution
  - Mating selection, survival selection, steady-state GAs, elite preservation, niching
- Exploration: Extent of search in creating child pop.
  - Crossover, mutation, inversion, local search
- One-max problem: Selection take-over time and mixing time must be of the same order: $p_c \geq A \ln S$

f(01101001101) = 6

Reference: D. E. Goldberg and K. Deb and D. Thierens (1993). Toward a better understanding of mixing in genetic algorithms. *Journal of the Society of Instruments and Control Engineers (SICE), 32*(1),10-16.



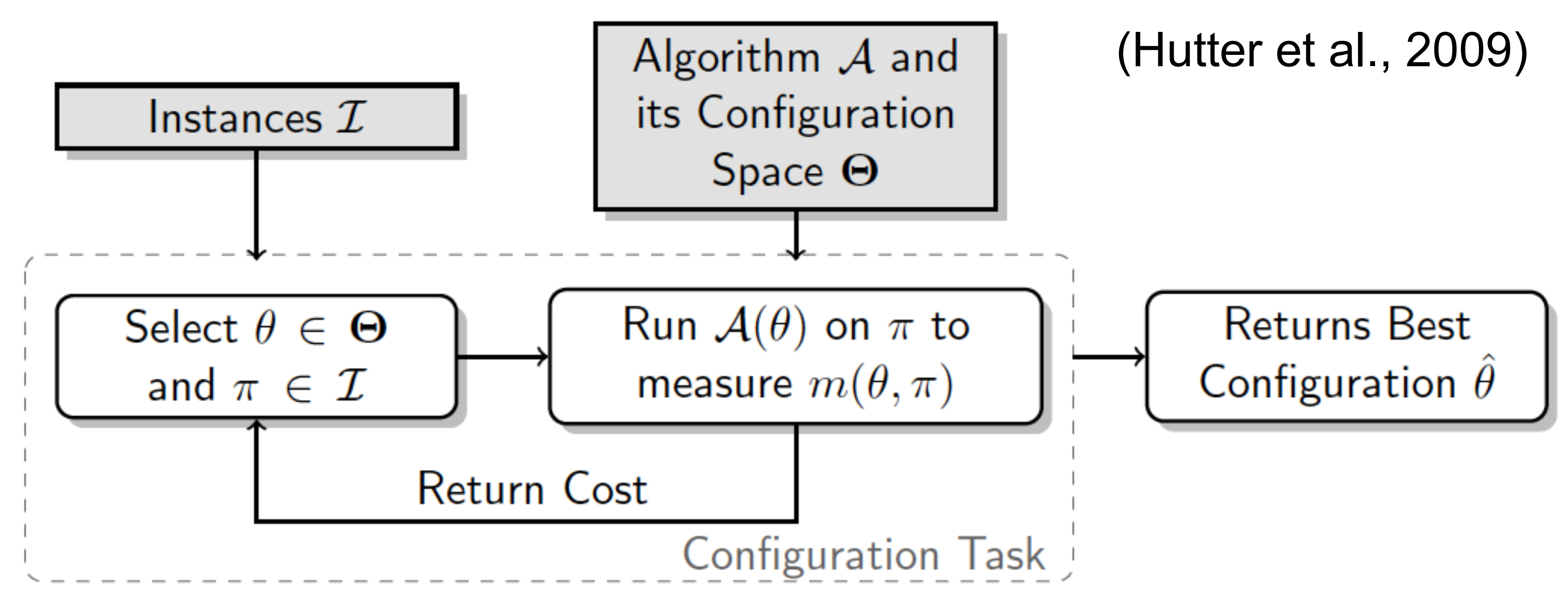
Automated Parameter Tuning

- How to tune GA parameters
  - n_1 popsize, n_2 crossover prob., n_3 mutation prob., etc.
  - $(n_1 \times n_2 \times n_3 \times \ldots)$ experiments, then their replicates
  - Fix n_2, n_3, …, vary n_1 and find best $\overline{n_1}$; Then fix $\overline{n_1}$, n_3, … and find best $\overline{n_2}$, and so on
- Meta-GA (Grefenstette, 1986)
  - A GA to find optimal GA parameters
- Ensemble-based EC methods
  - Many operators in play
  - Initially all have equal probability, survived children counted
  - Prob. of each operator is updated based on survival prob.
  - Algorithm gets updated based on its performance on problem

Reference: Grefenstette J. (1986). Optimization of control parameters for genetic algorithms, *IEEE Transactions on Systems, Man and Cybernetics, 16* (1), 122-128.

Algorithm Configuration Procedure

- Meta-GA procedure formalized



- Algorithm $\mathcal{A}$ is parameterized with θ
- Problem instance set $\mathfrak{I}$ (predefined)
- $\hat{\theta} \in \mathrm{argmin}_{\theta \in \Theta} \mathbb{E}_{\pi \sim \mathfrak{I}}(m(\theta, \pi))$, $\hat{\theta} \in \mathrm{argmin}_{\theta \in \Theta} \frac{1}{N} \sum_{i=1}^{N} m(\theta, \pi_i)$

Reference: Hutter, F., Hoos, H. H., Leyton-Brown, K., & Stützle, T. (2009). ParamILS: An automatic algorithm configuration framework. *Journal of Artificial Intelligence Research*, *36*, 267-306.

Cost Function $m(\theta, \pi)$

- Cost function $m(\theta, \pi)$ for an algorithm parameter setting θ and a problem class π
- Speed, accuracy, memory, energy consumption, latency, …
- Fairness: similar performance on different problem classes
- Reproducibility (replications), how many runs?
- Parameter space Θ usually mixed, real, discrete, categorical
- Configuration optimization needs to be efficient
- AutoML procedure: Taking human out of the loop

End of Module 3, Lecture 13, Part 2

- Self-adaptation is the key for metaheuristics
- Search and optimization algorithm need to be self-adaptive
- Exploitation-exploration thinking is helpful in algorithm design and modification
- Automatic algorithm configuration is a viable way to find best parameter setting for an algorithm

- Part 3:
  - **irace** method