CEC'2004 Tutorial

Evolutionary Computation in Dynamic and Uncertain Environments

Yaochu Jin

Honda Research Institute Europe
63073 Offenbach/Main, Germany

Outline

- Brief introduction to evolutionary algorithms
- Handling noisy fitness functions
- Using meta-models in evolutionary computation
- Searching for robust solutions
- Tracking moving optimums and dealing with multiple objectives
- A uniform framework for uncertainties in evolutionary computation
- Related additional information

Introduction to Evolutionary Algorithm

Introduction to Evolutionary Algorithms

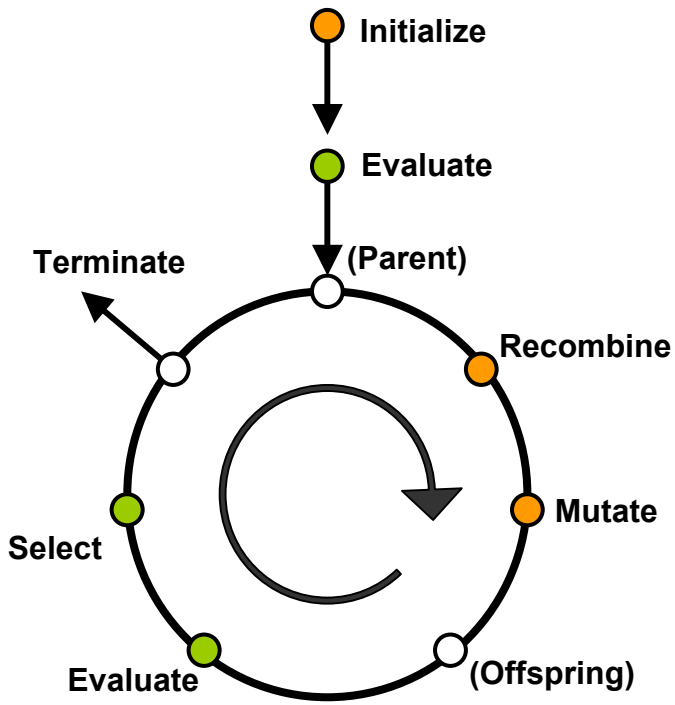
- Generic structure evolutionary algorithms
  - Problem representation (encoding)
  - Recombination, mutation and selection
  - Fitness evaluations
- Evolutionary algorithms
  - Genetic algorithms

    (crossover, mutation, stochastic selection)
  - Evolution strategies

    (mutation, recombination, deterministic selection, self-adaptation)
  - Genetic programming …
- Pros and cons
  - Stochastic, global search
  - No requirement for derivative information
  - Well-tailored for hybrid optimization
  - Population-based search

  o Need large number of fitness evaluations

  o Not well-suitable for on-line optimization



Handling Noisy Fitness Functions

Handling Noisy Fitness Functions (I)

- Basic assumptions
  - Noise is additive
  - Noise is normally or uniformly distributed

$$F(X) = f(X) + z, \quad z \sim N(0, \sigma_N^2), \ \sigma_N^2 \text{ is variance of the noise}$$

- Methods
  - Use a larger population size to reduce the influence of noise
  - Thresholding in selection (Markon et al, 2001)
    - an optimal value for the threshold is derived for (1+1)-ES under certain conditions
  - Use averaging to filter out the noise (re-sampling)
    - sample multiple times (Fitzpatrick and Greffenstette, 1988)
    - sample the neighborhood – averaging over space (Branke98)
  - recombination is recommended in ES (Beyer, 1998)

Handling Noise in Fitness Functions (II)

- Re-sample effectively
  - increase the sample size as evolution proceeds (Aizawa and Wah, 1994)
  - adapt the sample size based on the probability of the individual to be selected (Stagge, 1998)
  - adapt the sample size based on the estimated error probability in tournament selection (Branke, 2003)

- Fitness estimation
  - estimate true fitness using all history data, assuming a Gaussian distribution (Sano and Kita, 2000)
  - estimate true fitness using local regression models (Branke et al, 2001)

Handling Noisy Fitness Functions (III)

- Influence of noise on convergence
  - convergence is slowed down in (deterministic) tournament selection (Miller and Goldberg, 1995)

$$\mu(t+1) - \mu(t) = I\,\sigma_F^2\,(t) \qquad \text{(without noise)}$$

I : selection intensity (effective selection pressure)

$$\mu(t+1) - \mu(t) = I\,\sigma_F^2(t) / [\sigma_F^2(t) + \sigma_N^2\,(t)\,]^{1/2} \qquad \text{(with noise)}$$

$$t_{conv} / t'_{conv} = \sigma_F^2(t) / [\sigma_F^2(t) + \sigma_N^2\,(t)]^{1/2}$$

  - population sizing(Miller and Goldberg, 1996)

$$N = \Gamma(\sigma_F^2 + \sigma_N^2)$$

Γ: population sizing coefficient

Fitness Approximation in Evolutionary Computation

Evolutionary Computation with Approximate Fitness

- Basic assumption: Fitness function is noisy and biased

$$F(X) = f(X) + z, \quad z \sim N(\mu_N, \sigma_N^2)$$

- μ_N and σ_N^2 are mean and variance

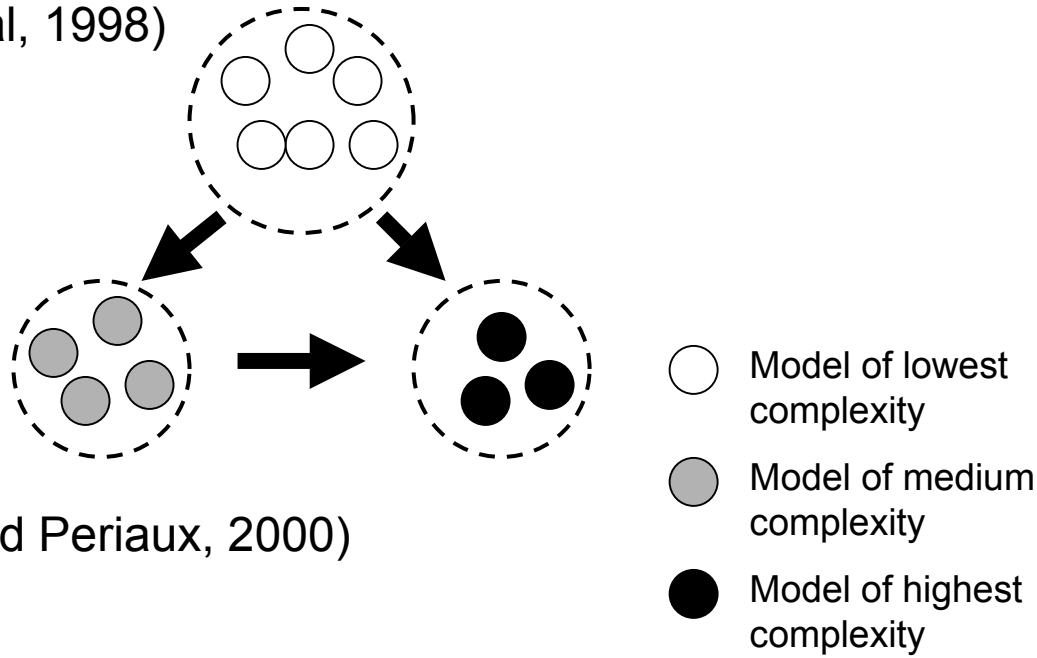
- When fitness approximation is necessary
  - No explicit fitness function exists
  - Fitness evaluation is highly time-consuming
  - Fitness is noisy (filter out noise)
  - Fitness is highly rugged (smooth out the fitness landscape)
  - Search for robust solutions (based on expected fitness or variance etc.)

- How to generate meta-models

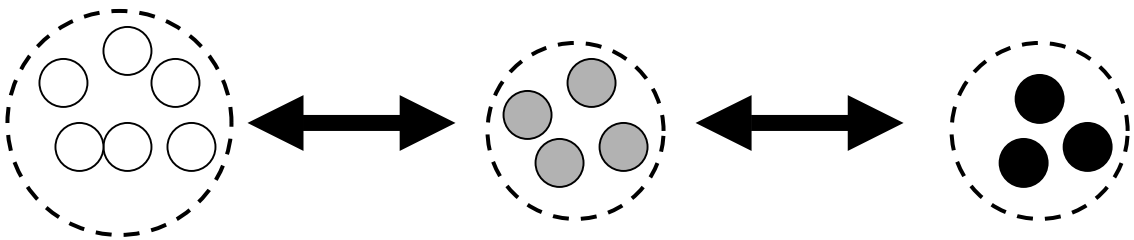
- How to use meta-models
  - Approximate model using multiple populations
  - Approximate model in initialization, crossover and mutation
  - Approximate model in fitness evaluations

Approximate Models Using Multiple Populations

- Injection island model (Eby et al, 1998)



- Hierarchical model (Sefrioui and Periaux, 2000)



Reduction of Randomness in Genetic Operators

- Generate a plurality of individuals randomly and choose the best ones according to the meta-model for the initial population

- Informed crossover (Rasheed and Hirsch, 2000):

  - select two individuals randomly
  - choose a crossover method randomly and generate a potential offspring
  - repeat the above process for a number of times
  - rank the potential offspring individuals and choose the best according to the meta-model as the offspring

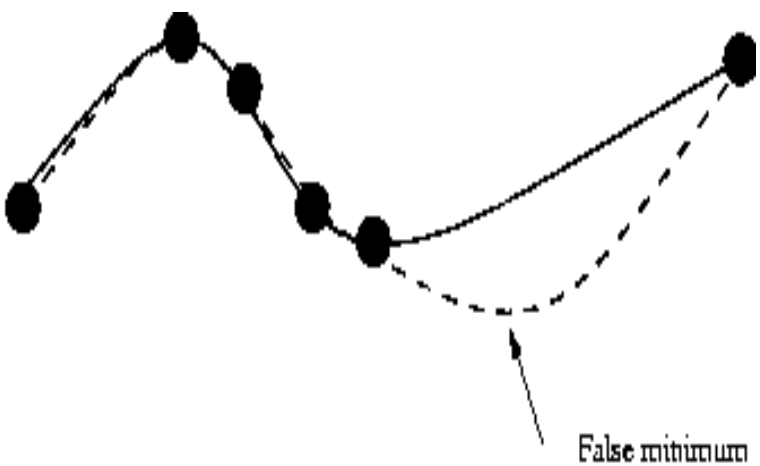
- Informed mutation (Rasheed and Hirsch, 2000; Abboud and Schoenauer, 2002):

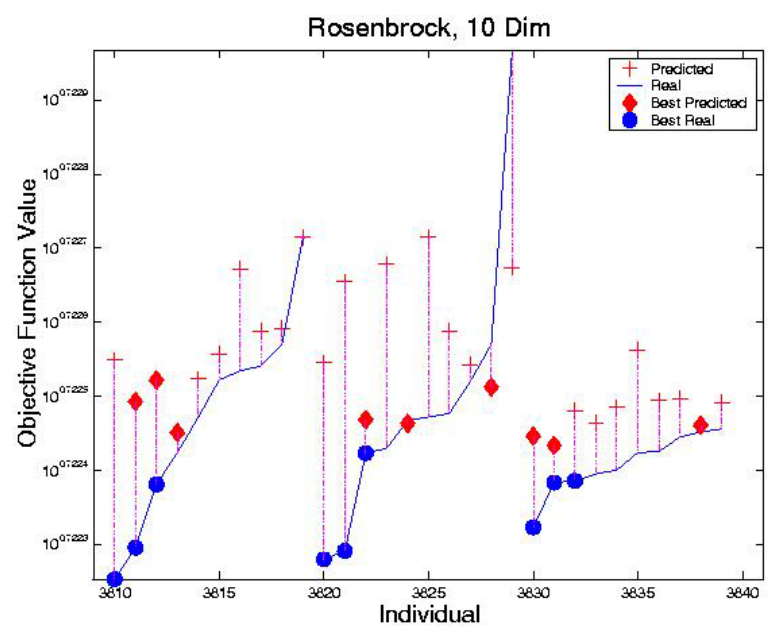
  - Generate multiple individuals randomly
  - Choose the best according to the meta model

Meta-models in Fitness Evaluations

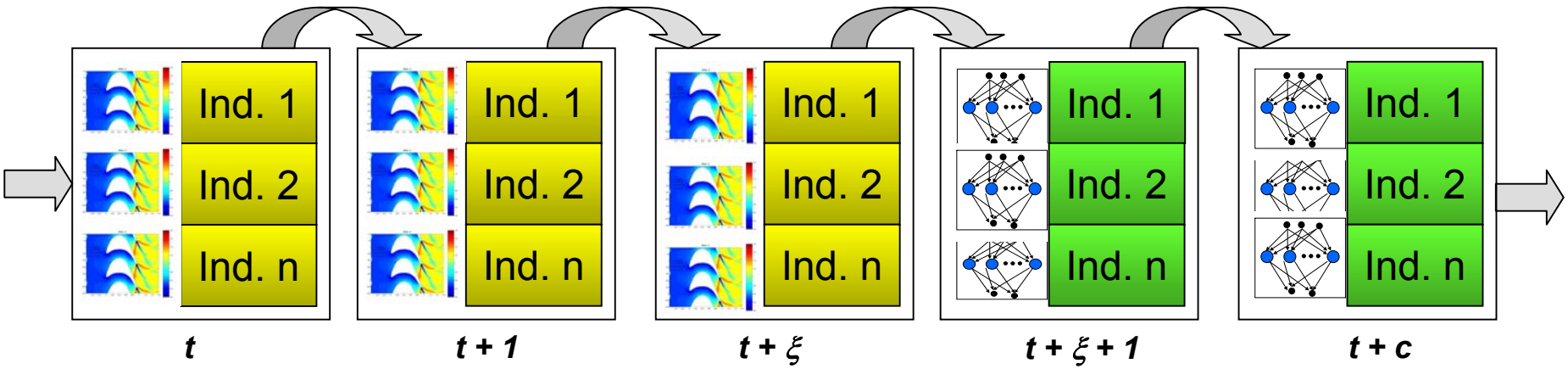
- Due to unavoidable modeling errors, a combination of meta-model with the original fitness function is necessary (evolution control / model management) (Jin et al, 2000)

  - generation-based evolution control
    - attractive for parallel computing
    - less flexibility
  - individual-based evolution control
    - powerful in model management
    - not the best for parallel computing if the number of controlled individual changes



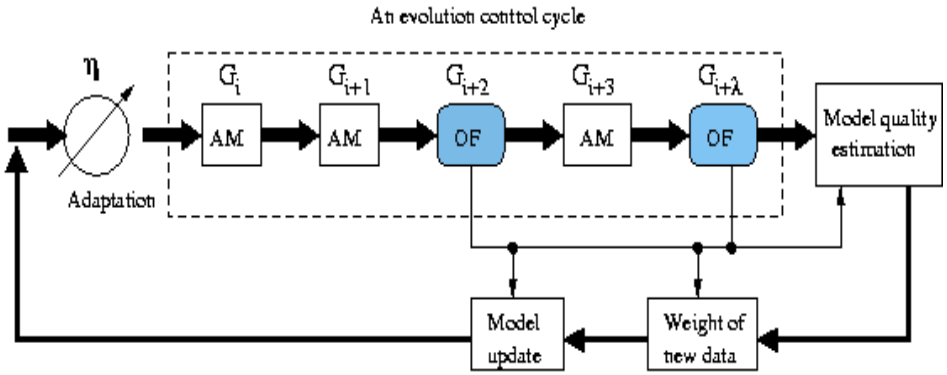


Generation Based Evolution Control (I)

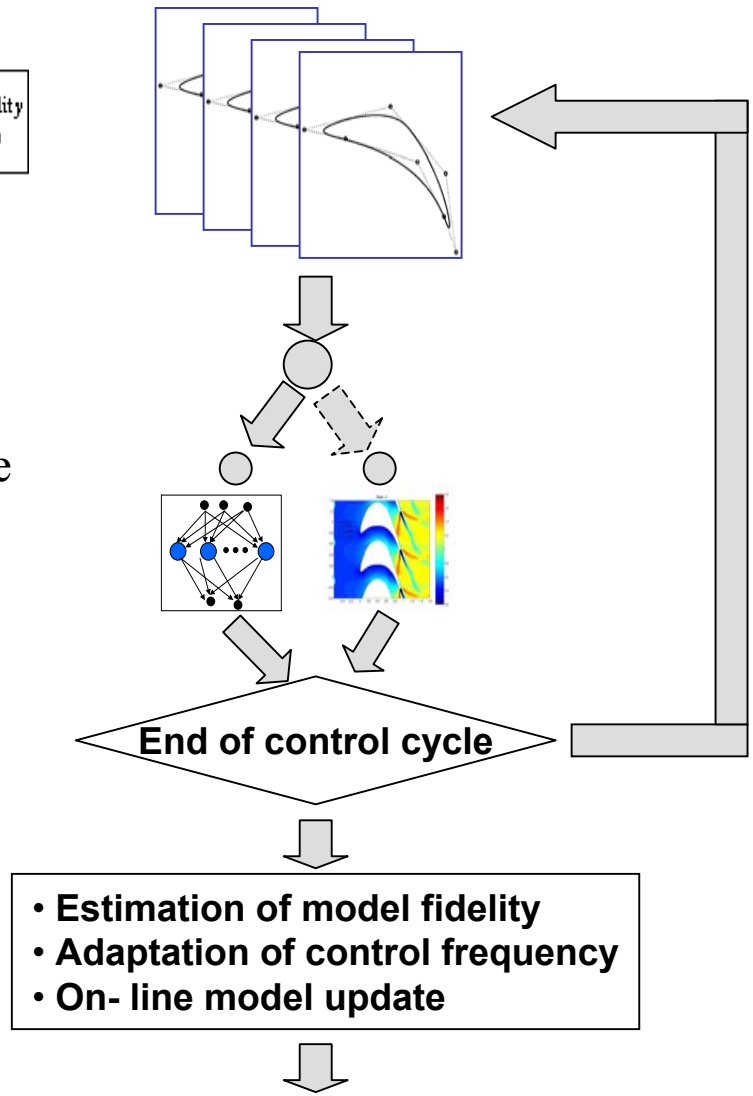


- Determine the control frequency heuristically (Bull, 1999)
- EA evolves on the meta-model until it converges (Ratle, 1998; Ratle 1999)
  - a bias toward the unexplored region is included (Büche et al, 2003)

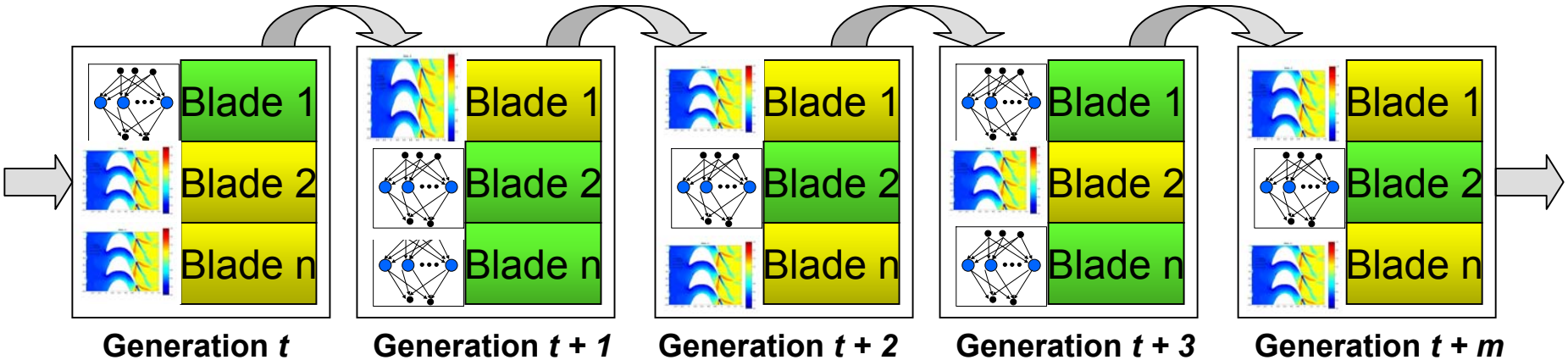
Generation Based Evolution Control (II)



Adaptation of the control frequency based on the model fidelity (Jin et al, 2001; Jin et al, 2002)



Individual-based Evolution Control (I)



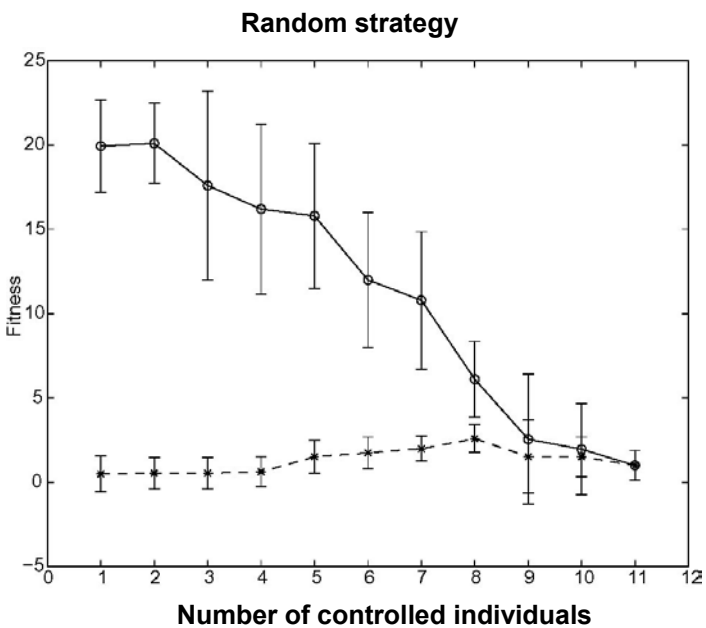
- Choose individuals randomly (Jin et al, 2000)
- Choose the best individuals according to the model (Jin et al, 2000)
  - bias the selection toward individuals with higher uncertainty (Emmerich et al, 2002)

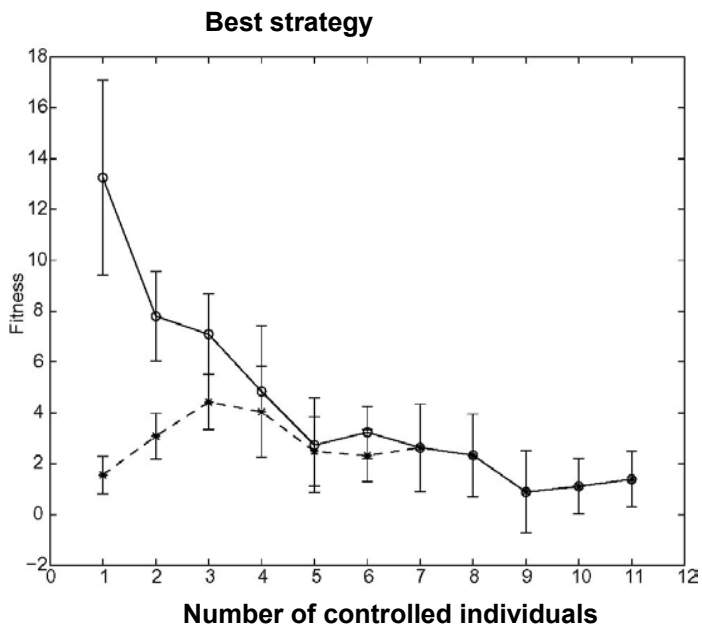
    $f(X) = \underline{f} - \alpha E$ (for minimization)

    - α: a constant
    - E: error bound
  - adapt the number of individuals to be re-evaluated (Hong et al, 2003)
- Choose the individuals with most uncertain individuals (Branke, 2003)
- Pre-selection (Ulmer et al, 2003)

Individual-based Evolution Control (II)

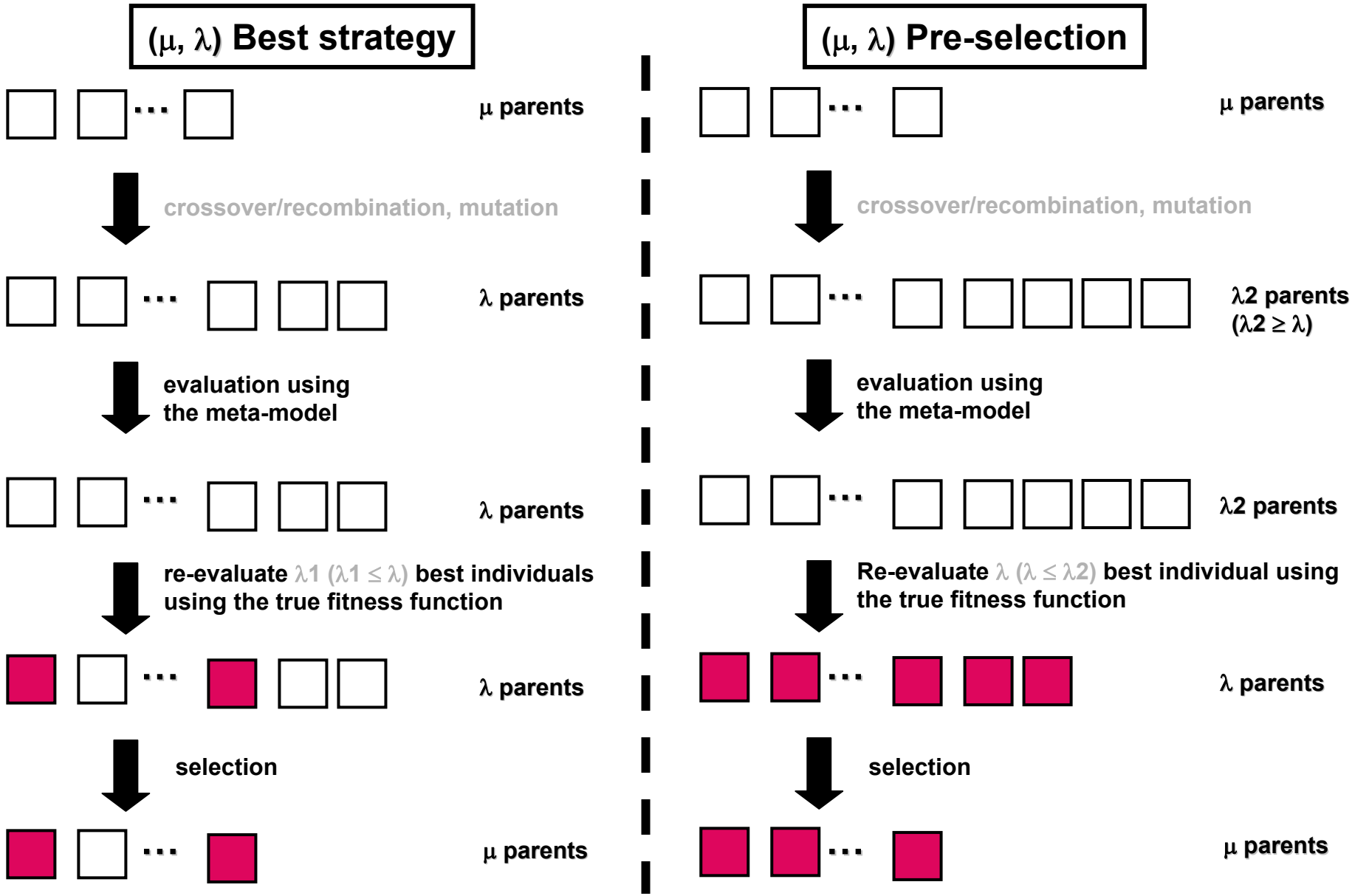
- 12-D Ackley function
- (3,12)-ES
- average over 10 runs





- the best strategy is more efficient than the random strategy
- in the best strategy, about half of the individual should be controlled

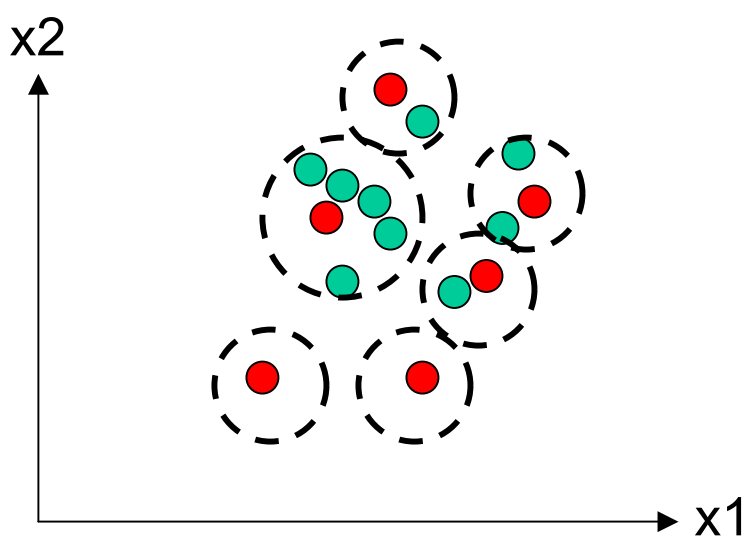
Individual-based Evolution Control (III)

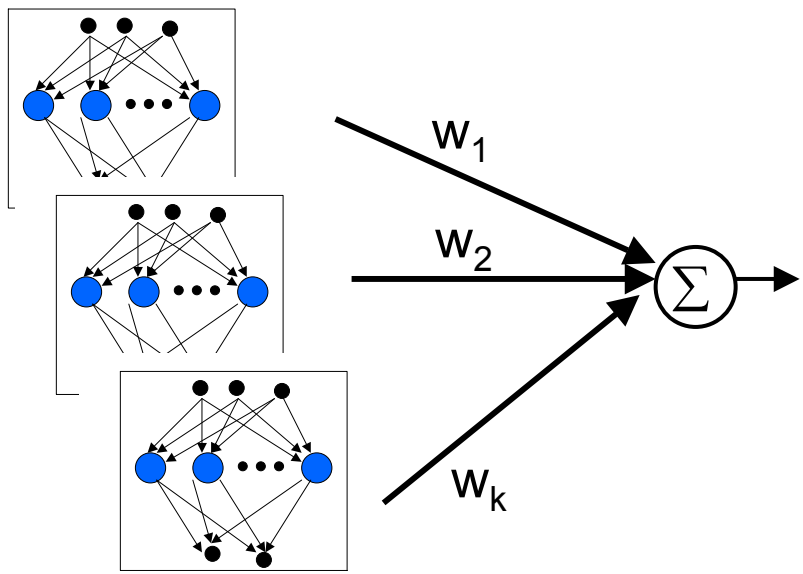


Individual-based Evolution Control (IV)

Clustering-based Individual Choice (Jin et al, 2004)

- Grouping the population into a number of clusters consisting of similar individuals
- Evaluate the individual closest to the cluster center only
- Estimate fitness using neural network ensembles
- estimate the prediction error based on the ensemble





Meta-models - General

- Types of meta-models
  - polynomials (Response surface methodology)
  - neural networks
    - multi-layer perceptrons (MLPs)
    - radial-basis-function networks (RBFNs)
  - Gaussian processes, kriging models (DACE)
  - support vector machines (SVMs)

- Comparison of models
  - no essential difference in model quality
  - Gaussian processes provides an error bound, useful in model management

- Modeling locally is more practical than modeling globally

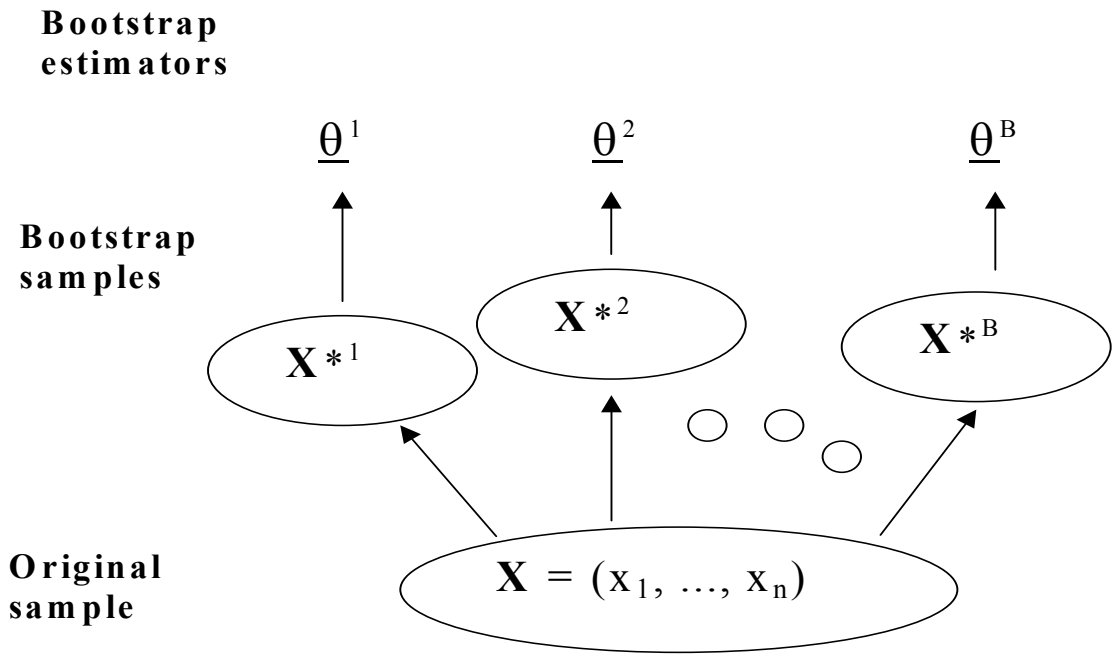
- Small variance is more important than small bias

- Ensemble is a practical and effective technique
  - reduce bias and variance simultaneously
  - provide an estimate of prediction error

Meta-models – Ensemble Techniques

- Bagging – Bootstrap aggregation
- Boosting
- Evolutionary approaches
  - sequentially and independently (different initial structure and parameters)
  - sequentially, negative correlated
  - simultaneously, negatively correlated
  - multi-objective approach (trade-off between accuracy and complexity)

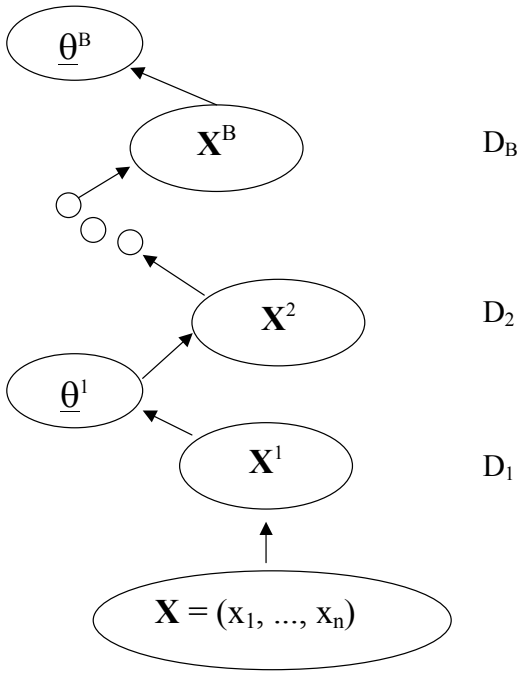
Ensemble Techniques - Bagging



$$\underline{\theta} = (\underline{\theta}^1 + \underline{\theta}^2 + \ldots + \underline{\theta}^B)/B$$

- Reduce variance while bias unchanged
- Degrade the performance of stable procedures

Ensemble Techniques - Boosting



- Initialize weights: $D_1(i) = 1/n$, $i=1,2,\ldots,n$
- For t=1 to B:
  - Fit a model ($\underline{\theta}^t$) to the data using weights D_1
  - Calculate error ε_t of model $\underline{\theta}^t$
  - Compute $\alpha_t = 1/2\ln[(1-\varepsilon_t)/\varepsilon_t]$
  - Update weight:

$$D_t(i) = D_{t-1}(i)\exp[\alpha_t \; I(y_i \neq \underline{\theta}^t(x_i)]$$

- Final model:

$$\underline{\theta} = (\alpha_1\underline{\theta}^1 + \alpha_2\underline{\theta}^2 + \ldots + \alpha_B\underline{\theta}^B)$$

$$\boxed{\underline{\theta} = (\alpha_1\underline{\theta}^1 + \alpha_2\underline{\theta}^2 + \ldots + \alpha_B\underline{\theta}^B)}$$

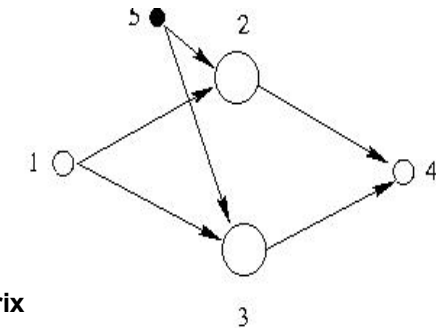
- Reduce both variance and bias
- Sensitive to noise, large number of bootstrap samples needed

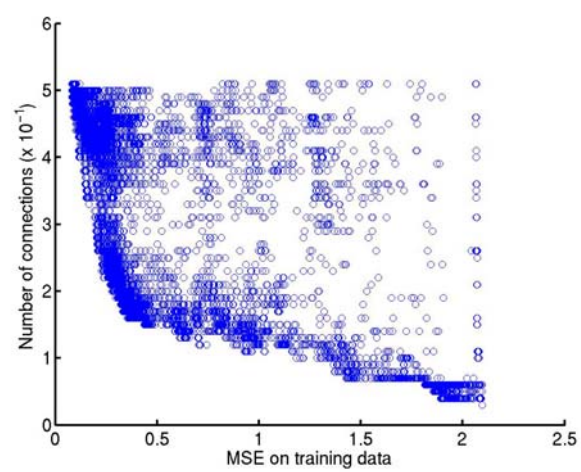
Ensemble Techniques – Multi-objective Network Evolution

- Structure and weight representation
  A connection matrix and a weight matrix
- Genetic operators
  - ➢ node addition/deletion
  - ➢ weight addition/deletion
  - ➢ Gaussian mutation to the weights
- Life-time learning
  - ➢ Rprop$^+$ learning algorithm
  - ➢ encode the change of the weights during life-time learning back to the chromosomes (Lamarckian evolution)
- Objectives
  - ➢ E: mean squared error on training data after life-time learning
  - ➢ Ω: model complexity (count the number of connections)
- Dynamic weighted aggregation

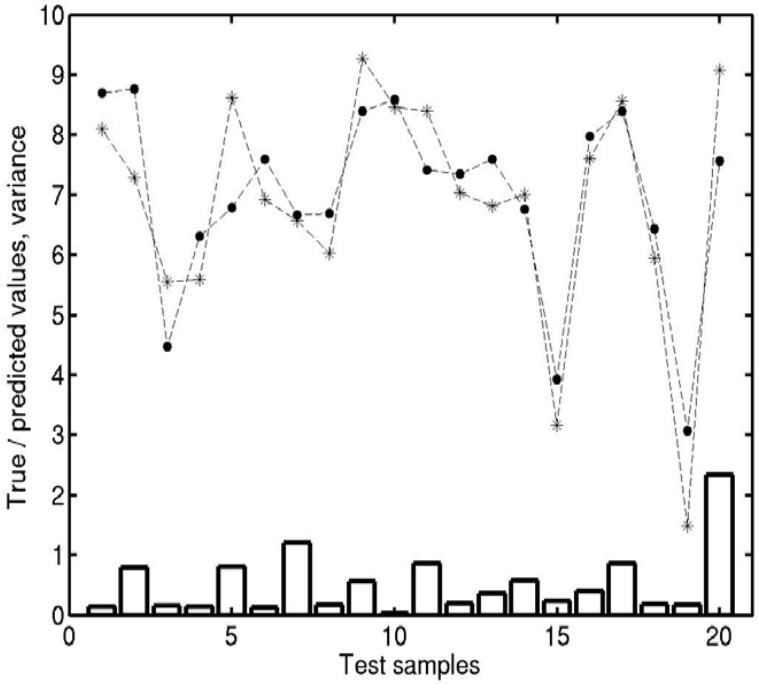
$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 & 1 \end{bmatrix} \Longrightarrow$$

connection matrix

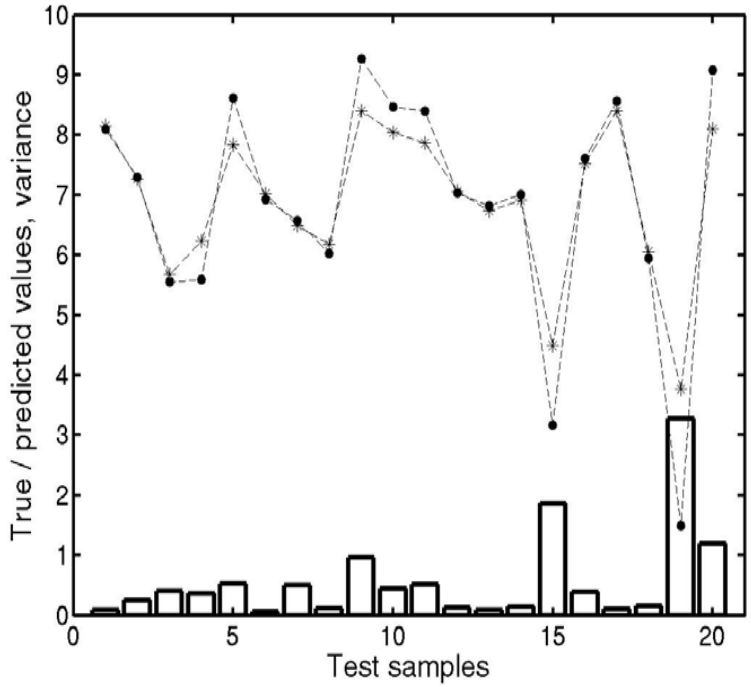




Comparison of Ensemble Techniques



Random initialization



Multi-objective approach

- three inputs
- 80 training data
- 20 test data
- no cross-validation in training

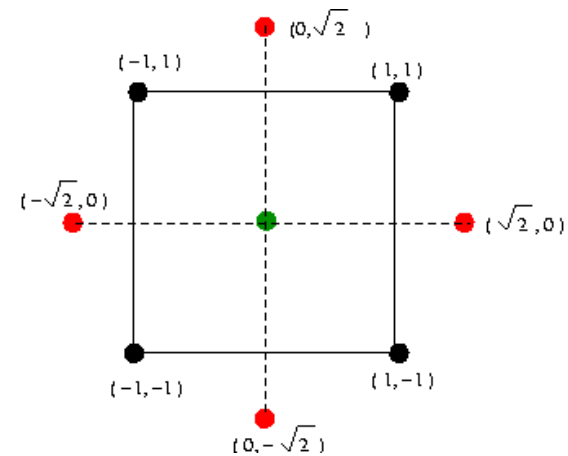
- reduced MSE
- more consistent error prediction

Data Sampling Techniques

- Design of experiments
  - Orthogonal array: X^TX is diagonal (first-order models)
  - Simplex design: require n+1 samples for n variables, so that the angle of any two points make with the origin (θ) satisfies:

    $$\cos(\theta) = -1/n \quad \text{(first-order models)}$$
  - Central composite design (second-order polynomials)
  - D-optimality (to maximize the determination of X^TX equals to minimize the variance of the estimate)

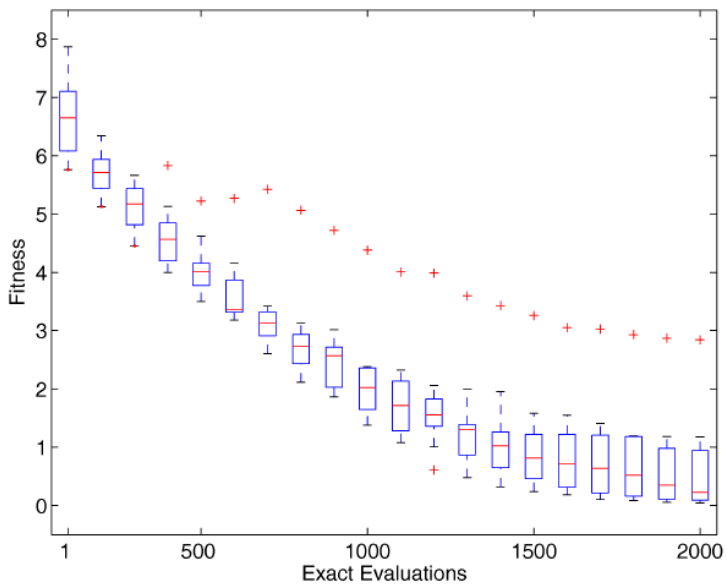
- Active learning
  - Maximize information gain
  - Reduce entropy
  - Reduce generalization error



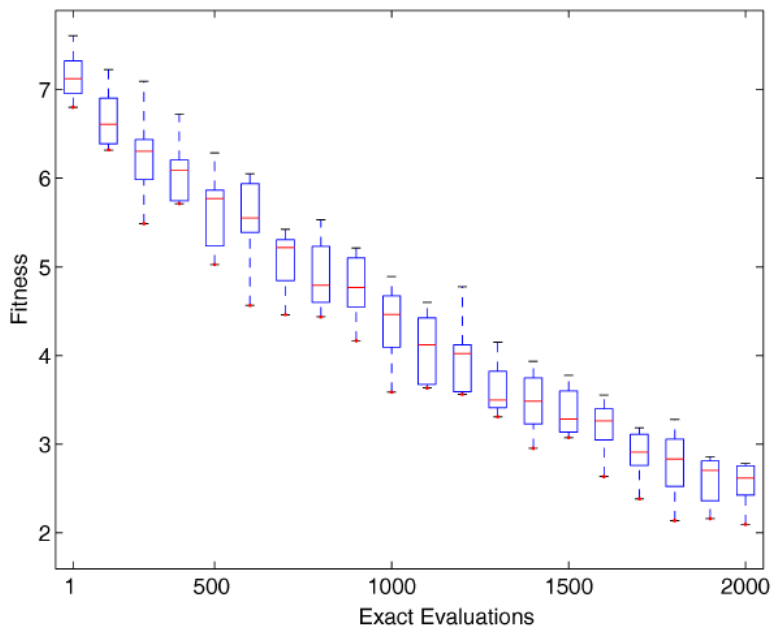
CCD

Optimization Results

- (5,30)-ES Covariance Matrix Adaptation
- 10 individuals are controlled
- Ensemble size = 3
- average over 10 runs

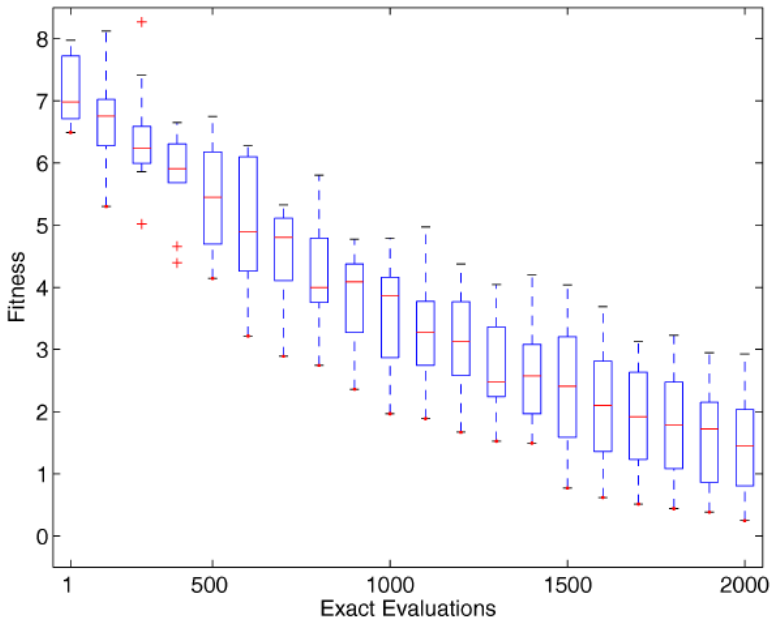


Clustering based based strategy

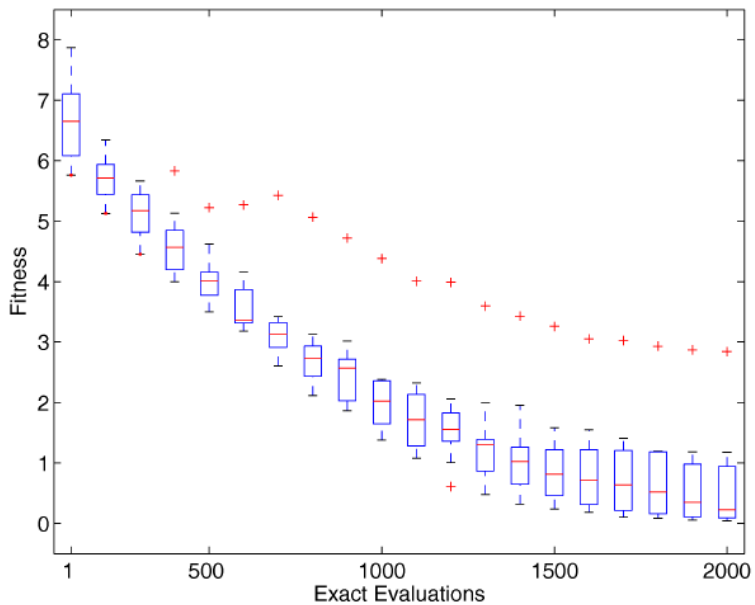


Plain

Single Neural Network versus Ensembles



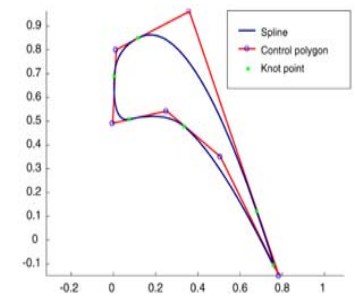
Single NN

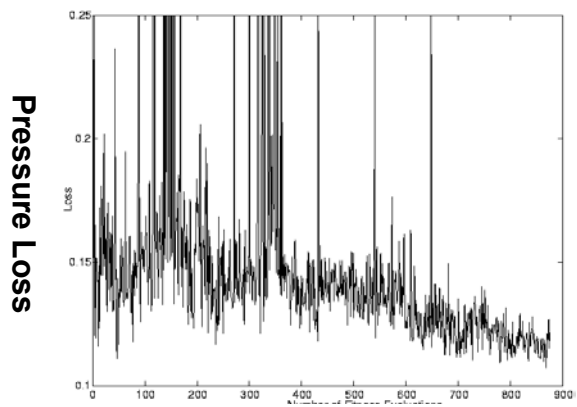


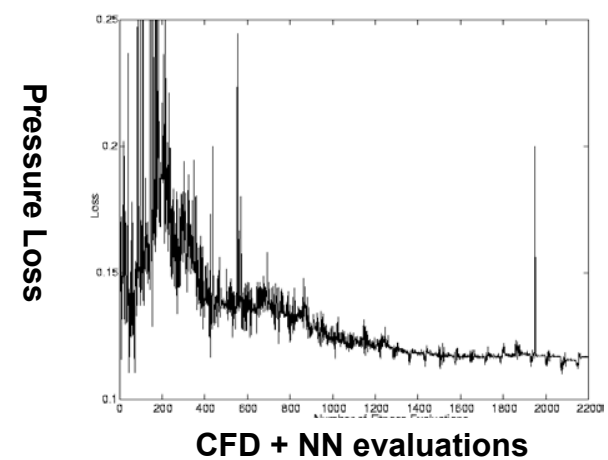
Network ensemble

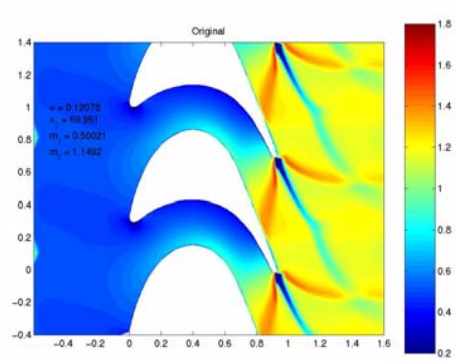
An Example: 2D Blade Design

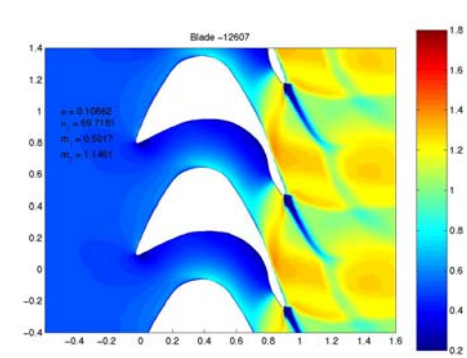
- Non-Uniform Rational B-Spline representation
- Computational fluid dynamics simulation for fitness evaluation
- Minimization of pressure loss +outflow angle deviation
- Mechanical constraints
- Evolution strategy
- Neural network as meta-model







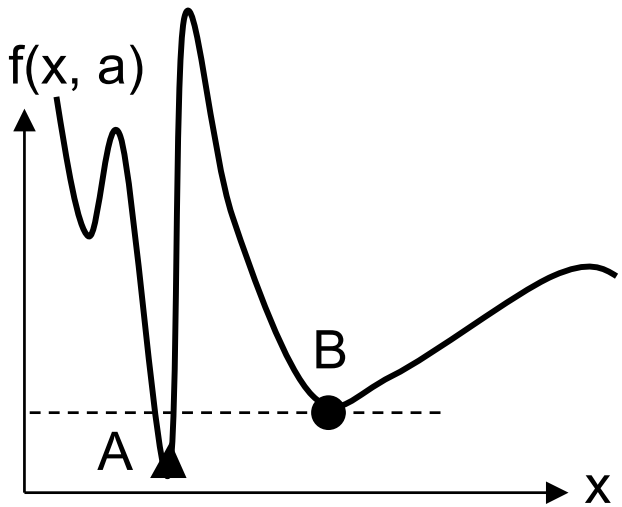


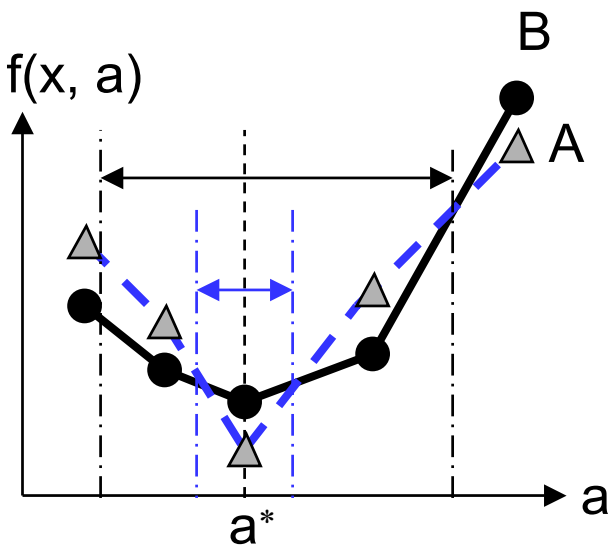


Search for Robust Optimal Solutions

Evolutionary Search for Robust Optimal Solutions

- Robust to variations in design parameters (x)
- Robust to variations in environmental parameters (a)



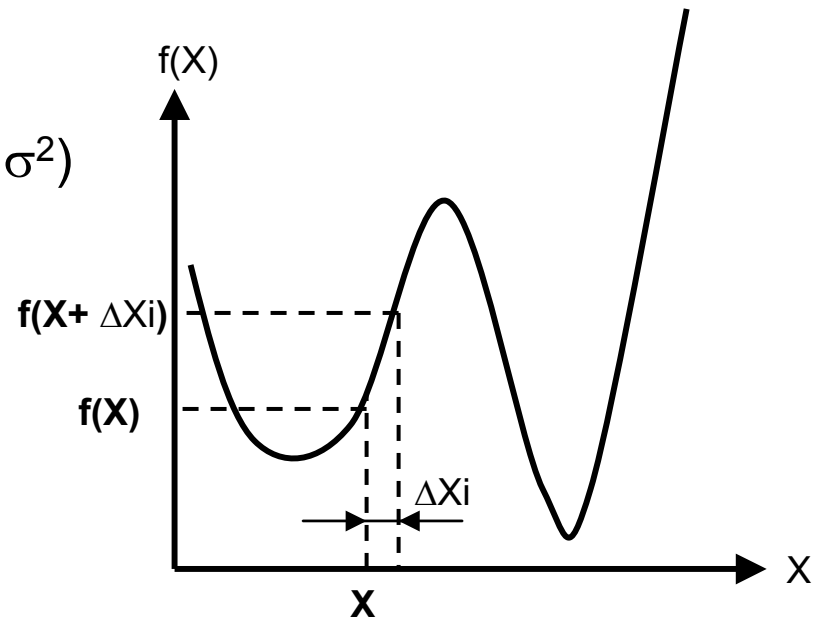


Expectation Based Approach to Robustness

- Averaging based approach:

$$f(x) = \sum_i f(x+\Delta x_i),\ \Delta x_i \sim N(0,\sigma^2)$$

  - Need additional fitness evaluations
  - Use of approximate models could alleviate this difficulty

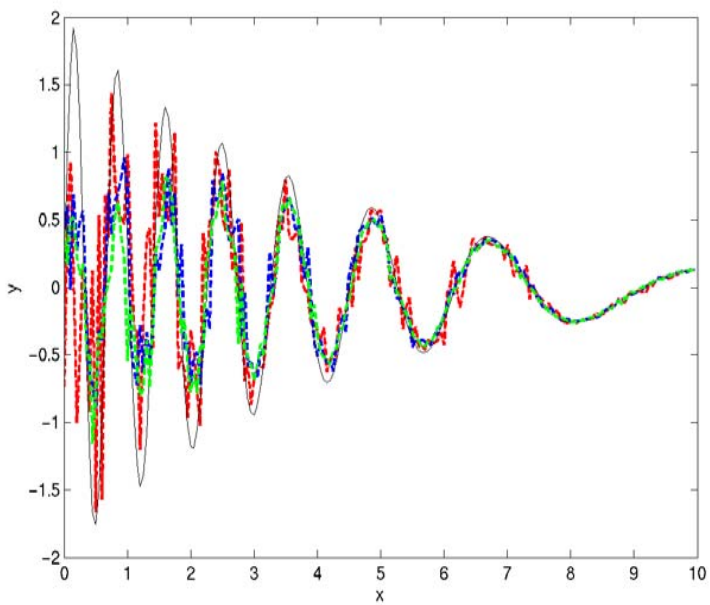


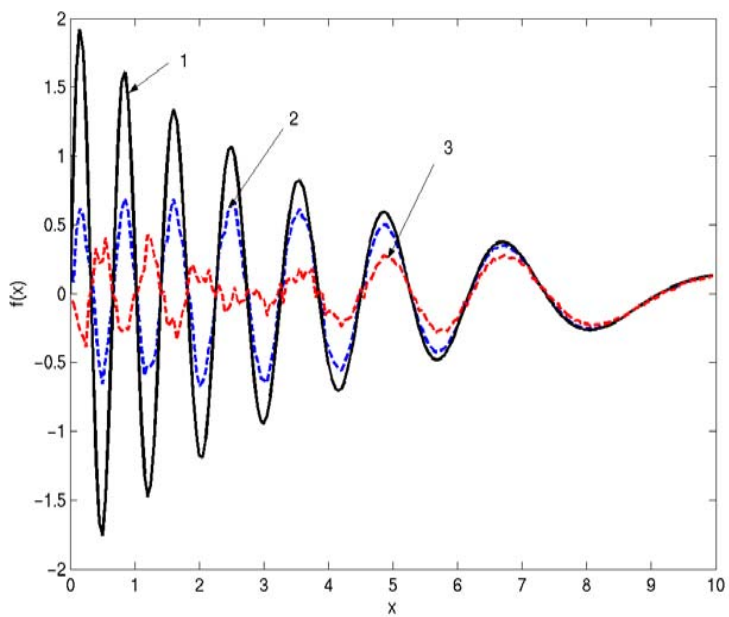
- Perturbation based approach:

$$f(x) = f(x+ \Delta x(t)),\ \Delta x(t) \sim N(0,\sigma^2)$$

  - Approximation can be proved under the assumption of an infinite population size
  - No additional fitness evaluations needed

Weaknesses of Expected Fitness Approach



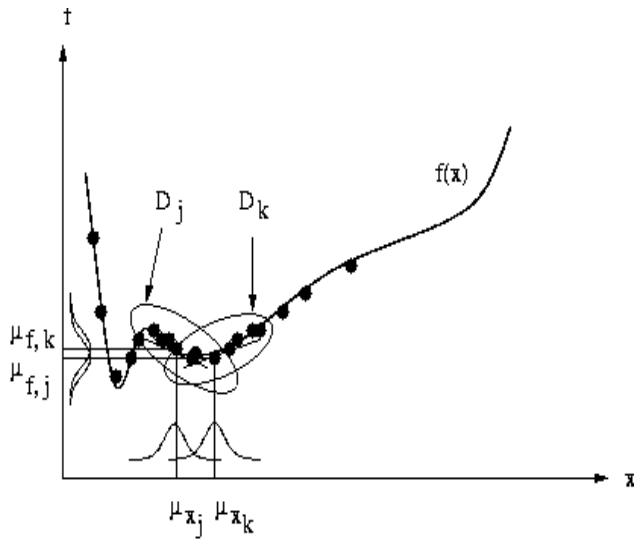


- Search result based on the expected fitness is sensitive to the number of averaging and the variance of averaging
- Search result based on expected fitness may fail to capture the trade-off between performance and robustness

Multi-objective Approach to Robustness

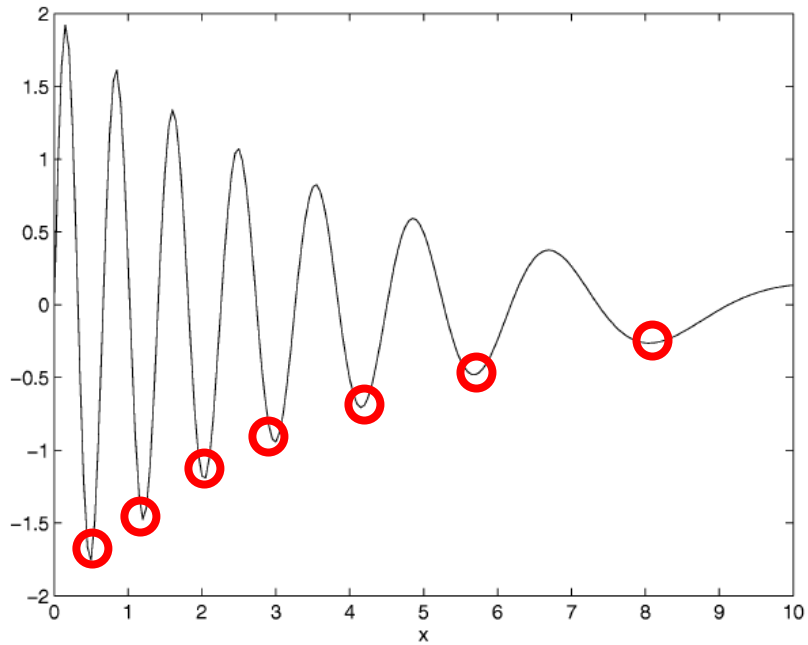
- Expected fitness value (first order moment) and the original fitness
- Higher moment of the original fitness (e.g. variance) and the original fitness function
- Expected fitness and variance of the fitness

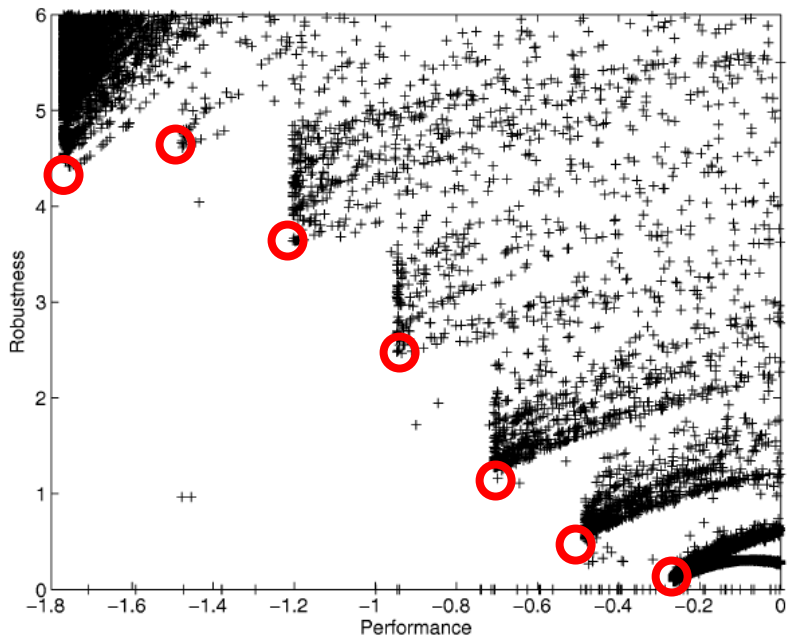
$$\min f^{R} = \frac{\sigma_{f,\,i}}{\sigma_{x,\,j}}$$



Estimation of function variance without additional fitness calculations.

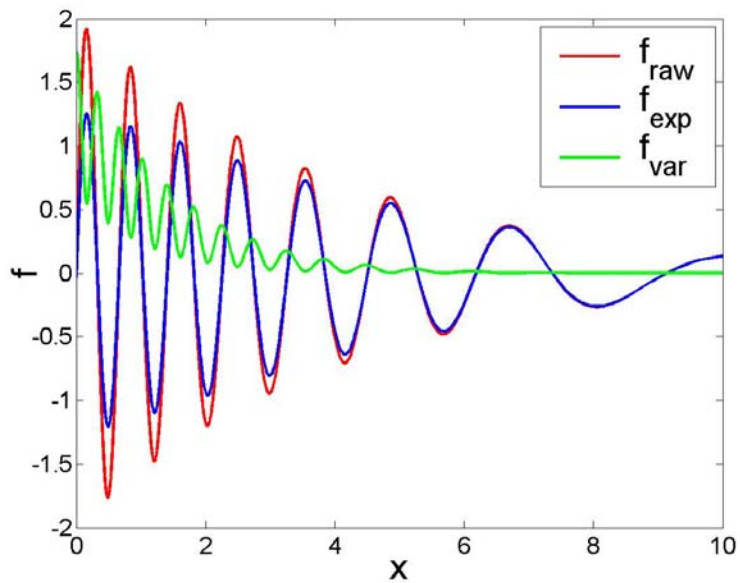
An Example for MOO Approach to Robustness



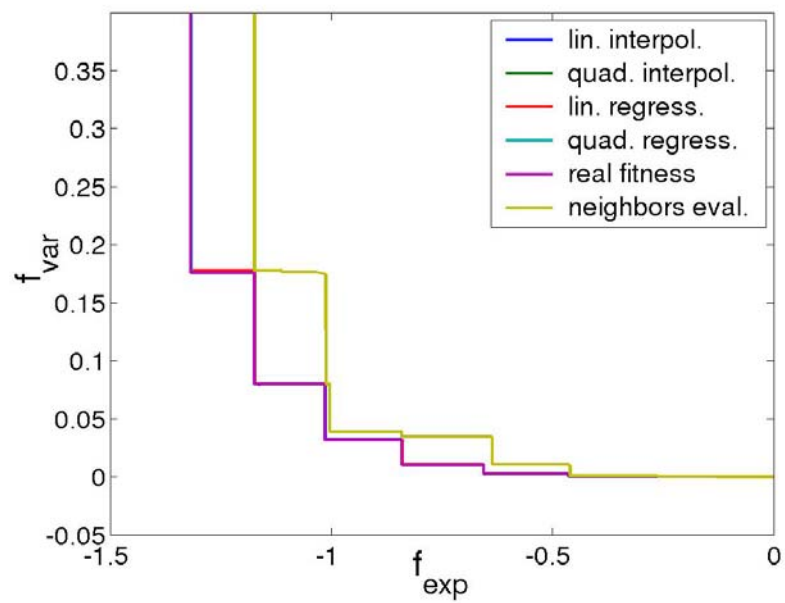


The MOO approach successfully provides a qualitative description of the robustness of different optima.

Using Meta-models in Search for Robust Solutions (I)



Test function

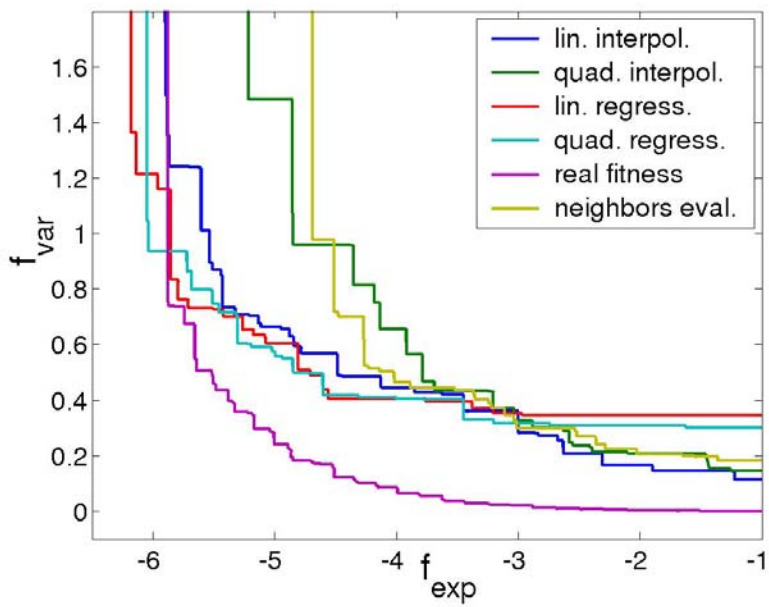


Trade-off between mean and variance

- median Pareto front using meta-models as well as the rough estimation method

- all approximation models perform as as good as when the real fitness was used.

Using Meta-models in Search for Robust Solutions (II)

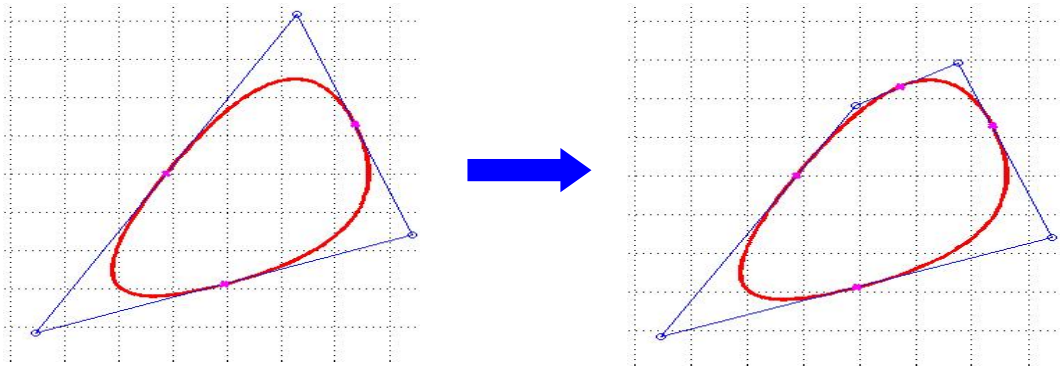


Dimension = 5

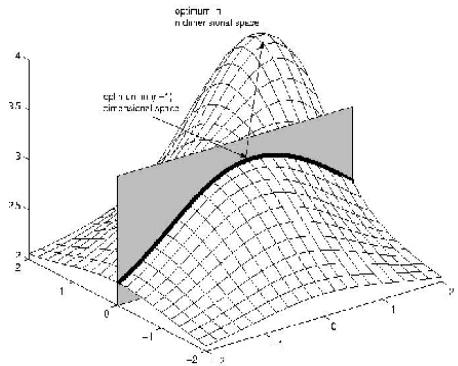
Tracking Moving Optimums and Dealing with Multiple Objectives

Examples of (Implicit) Dynamic Optimization Problems

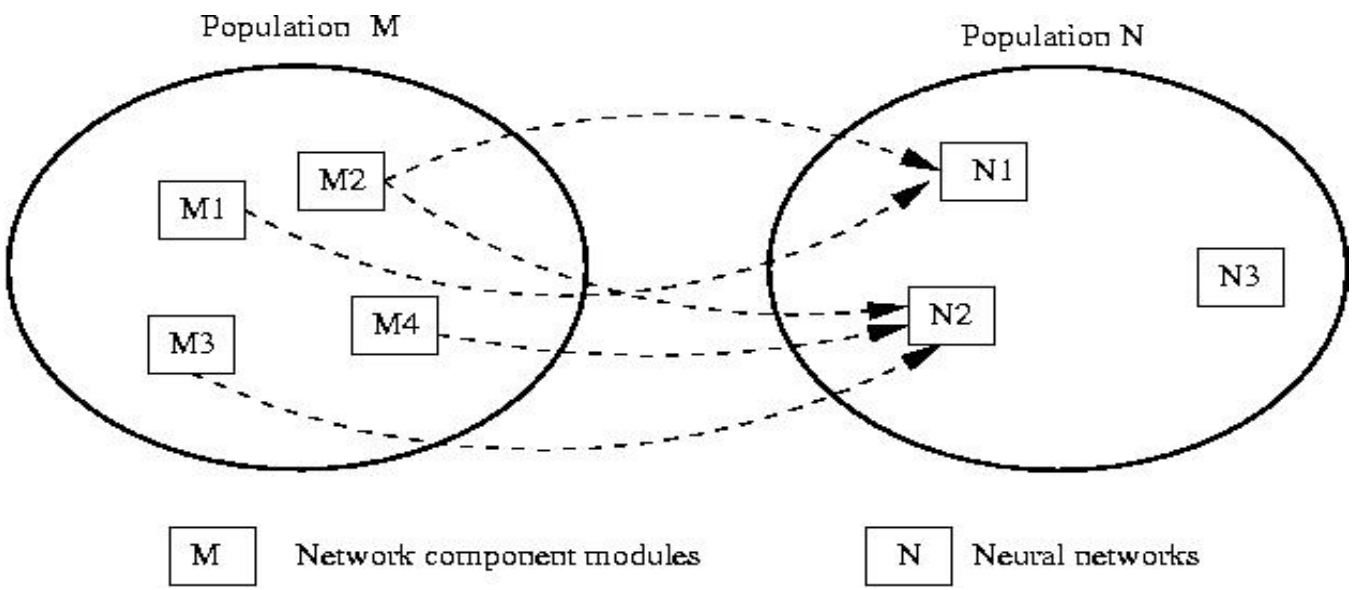
- Adaptive coding in structure optimization



Extension of Search Space



- Co-evolution



Dynamic Optimization Problems (DOPs)

- Optimization problems whose optimal solution changes over time during the optimization, which could result from
  - change of environmental parameters
  - change of constraints
  - change of objectives
  - change of problem settings (representations)

- Types of dynamic optimization problems
  - the optimum moves linearly/nonlinearly in the design space
  - the optimum oscillates periodically among a number of locations
  - the optimum jumps randomly

- What may matter
  - the speed of change
  - the severity of change
  - if the change is periodical, does the optimum move exactly back to the original location?
  - is the change observable/detectable or even predictable?

Methods for Dynamic Optimums (I)

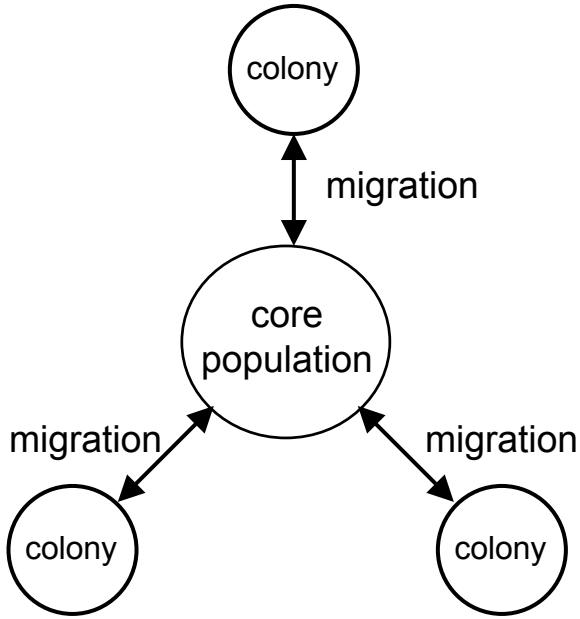
- Maintaining diversity of the population to prevent it from converging
  - add randomly generated individuals in GA (Grefenstette, 1992)
  - fitness sharing (Anderson, 1991)
  - aging of individuals (Ghosh et al, 1998)
  - adaptive chaotic mutation (Nanayakkara et al, 1999)
  - set a lower bound on step-sizes in ES to prevent them from converging to zero (Jin et al, 2004)
- Using explicit memory
  - store the best solutions in history and add them to the population is necessary(Mori et al, 1998; Branke, 1999; Bendtsen and Krink, 2002)
  - store promising genetic materials in a “gene library” for re-use (Tekol and Acan, 2003)

Methods for Dynamic Optimums (II)

- Using implicit memory
  - multiple populations (Oppacher abd Wineberg, 1999; Branke et el, 2000;
  - redundant coding (Smith, 1987; Goldberg and Smith, 1987; Dagsgupta and MacGregor, 1992; Ng and Wong, 1995; Lewis et al, 1998)

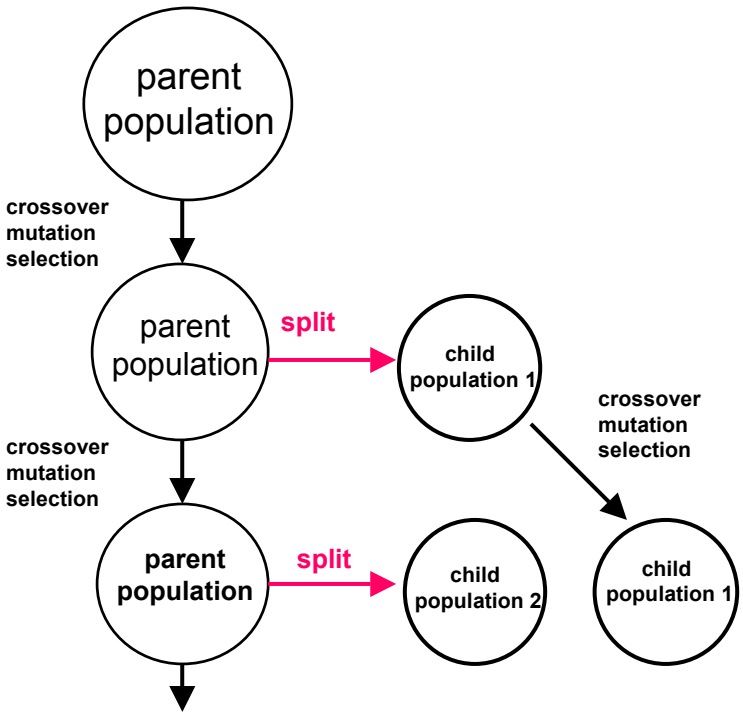
- (Self-)adaptation and learning
  - hyper-mutation: increase mutation when time-averaged best performance worsens (Cobb and Grefenstette, 1993)
  - self-adaptation - a double-side sword (Angeline, 1997; Bäck, 1998; Weicker and Weicker,2000)
  - life-time learning (Sasaki and Tokoro, 1998)

Tracking Moving Optimum Using Multiple Populations (I)



- Oppacher and Wineberg, 1999; 2000
  - the core population is used to exploit the promising area
  - a number of colonies are used to explore the search space
  - diversity measure (distance to the core population) is included in fitness evaluations of the colonies

Tracking Moving Optimum Using Multiple Populations (II)



- Branke et al, 2000
  - the parent population explores the search space
  - a child population is created when certain conditions are met
  - the size of parent and child population is adjusted
  - the child population only searches a limited range of the search space

Which for What

- Explicit memory and redundant coding are well-suited for cases in which the optimum oscillates periodically. Redundant coding approach seems to be effective only if the location of optimums are very limited

- Multi-population approach is good for tracking competing peaks. However, the search ability will decrease if too many child populations are created and the size of population is dramatically reduced

- Diversity of the population is most efficient for tracking continuously moving optimums

- Life-time learning for adaptation to small but very fast change of the optimum

Performance Indices for Dynamic Optimization

- PIs for stationary optimization
  - best-so-far
  - off-line performance: average best-so-far
  - on-line performance: average of all evaluations

- PIs for dynamic optimization
  - adaptation performance (Mori et al, 1997):

$$I = 1/T \sum f_{best}(t)/f_{opt}(t)$$

T:	number of generation
$f_{best}(t)$:	best fitness in the population at time t
$f_{opt}(t)$:	global optimum at time t

  - accuracy (Trojanowski and Michalewicz, 1999)

$$Acc = 1/K \sum err_i$$

err_i :	difference between the current best in the population just before change and the optimum value averaged over the entire run

  - the average distance to the optimum at each generation (Weicker and Weicker, 1999)

  - best-of-generation average (Grefenstette, 1999; Bäck, 1999)

Dynamic Optimization Test Functions

- General requirements for constructing dynamic optimization test problems:
  - computationally efficient
  - different dynamic behaviors are realizable
  - the complexity of the fitness landscape is controllable
  - the trajectory of the optimum can be known
  - the test problem should be somehow relevant to real-world applications

- Main approaches:
  - switching between different objectives (Cobb and Grefenstette, 1993)
  - shifting stationary functions (Cobb and Grefenstette, 1993; Angeline, 1997; Farina et al, 2003)

    $f(\mathbf{x}, t) = f(\mathbf{x}+\mathbf{d}(t))$;

    (applicable to MOO)
  - competing/moving peaks (Branke, 1999; Morrison, 1999)

    $f(\mathbf{x}, t) = \max_{i=1, M}\{P_i(\mathbf{x}, t)\}$;

    $P_i(\mathbf{x}, t)$ is time-varying in terms of peak height, location and width
  - adapting multi-objective test functions to dynamic optimization (Jin et al, 2004)

    $f(\mathbf{x},t) = \sum_{i=1, M} w_i(t)\, f_i(\mathbf{x})$

    $w_i(t)$ time-varying weights for each objective

    M is number of objectives of the MOO test function

Dynamic Single Objective Optimization

$$DOP_F(\mathbf{x},t) = w(t)\, f_1(\mathbf{x}) + (1-w(t))\, f_2(\mathbf{x})$$

- If the weight changes randomly, and if MOP_F(**x**) is convex and continuous, then the optimum of DOP_F(**x**,t) moves randomly

- If the weight changes linearly, and if MOP_F(**x**) is convex, uniform and continuous, the optimum of DOP_F(**x**) moves linearly

  - if MOP_F is uniform with regard to the fitness space, then the optimum moves linearly in the fitness space; if uniform with regard to the parameter space, then the optimum moves linearly in the parameter space

  - If the weights changes linearly and MOP_F(**x**) is convex and non-uniform, or if the weight changes non-linearly and MOP_F(**x**) is convex and uniform, the optimum of DOP_F(**x**) moves nonlinearly

  - If the weight switches among a few given values, the optimum of DOP_F(**x**) oscillates

Remark: When constructing a DOP from an MOP, the weights must not be constrained. In this case, the behavior of the moving optimum becomes more complicated.

Dynamic Multi-Objective Optimization

- A stationary three-objective optimization problem:

$$MOO_F(\mathbf{x}) = \min \{f_1(\mathbf{x}), f_2(\mathbf{x}), f_3(\mathbf{x})\}$$

- Reformulate it as follows:

$$DMOP_F(\mathbf{x},t) = \min\{F_1(\mathbf{x},t), F_2(\mathbf{x},t)\}$$

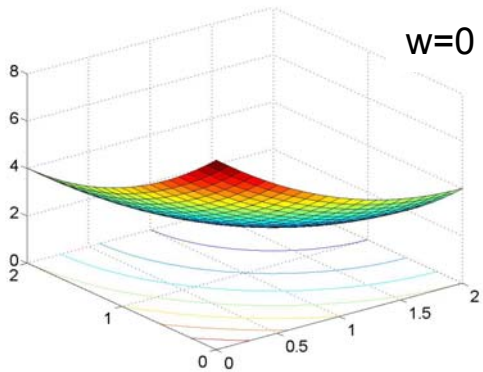
$$F_1(\mathbf{x},t) = w(t)\, f_1(\mathbf{x}) + (1- w(t))\, f_2(\mathbf{x})$$

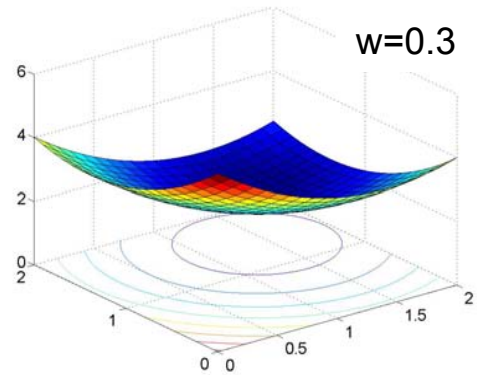
$$F_2(\mathbf{x},t) = w(t)\, f_1(\mathbf{x}) + (1- w(t))\, f_3(\mathbf{x})$$

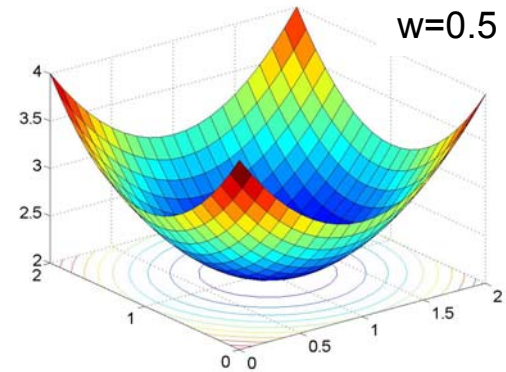
- The Pareto front moves when w changes over time.

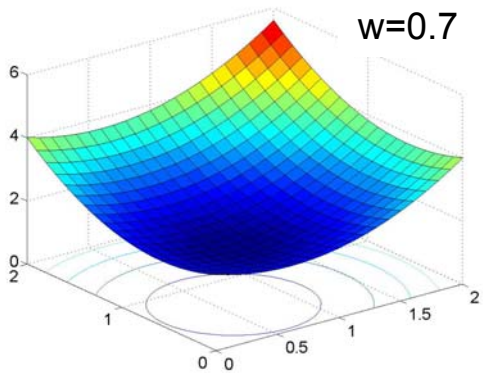
Remark: w(t) should be constrained so that for any given w, the solution of DMOP is a subset of the solution of the original MOO problem.

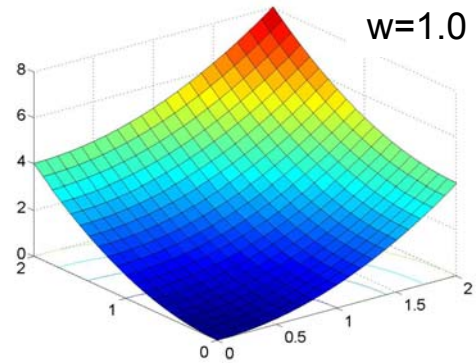
Generating Moving Peaks











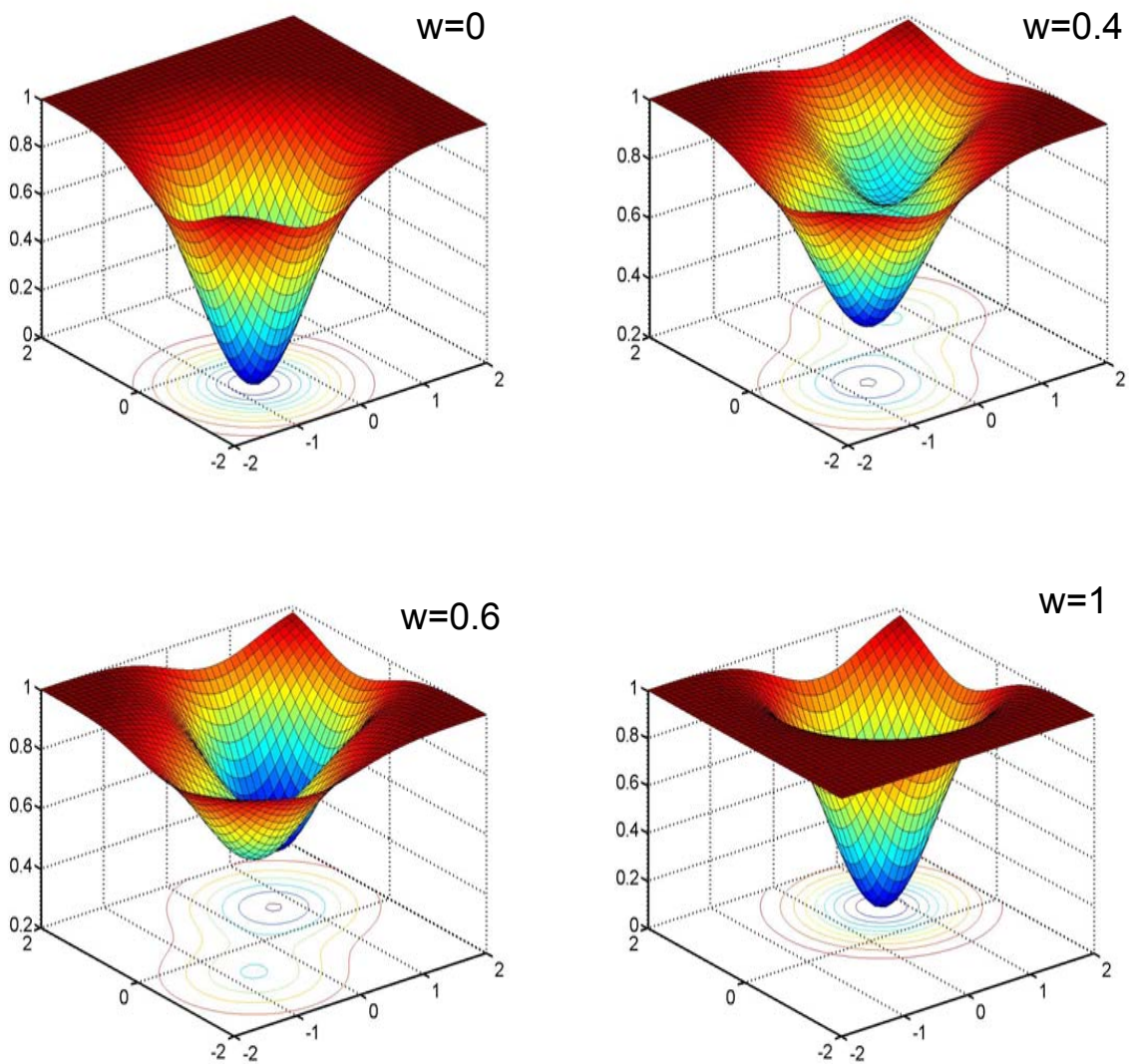
MOP:

- Schaffer' test function
- The Pareto front is convex and uniform
- Trajectory defined by $x_1=x_2$

DOP:

- The optimum moves linearly from (0,0) to (2,2) if the weight change linearly
- If the weight changes randomly, the optimum will be located randomly on the curve defined by $x_1=x_2$ between (0,0) and (2,2)

Generating Competing Peaks



MOP:

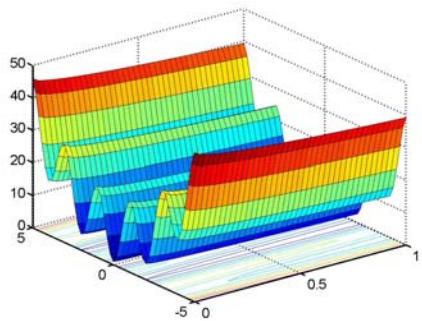
- Fonseca's test function
- The Pareto front is concave
- The two ends of the Pareto-front are at (-0.717, -0.717) and (0.717, 0.717)

DOP

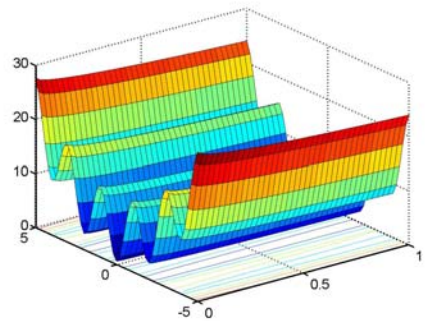
- When weights change, MOPs with a concave Pareto front generate competing peaks
- Both the height and the shape of the peaks change
- The two peaks locate at (-0.717, -0.717) and (0.717, 0.717), and the peak height is 1.

The winning peak switches when w=0.5 (the two peaks are of the height for this weight).

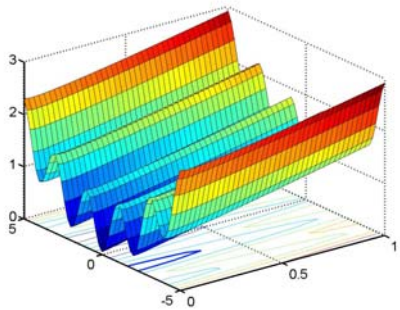
Complexity Control



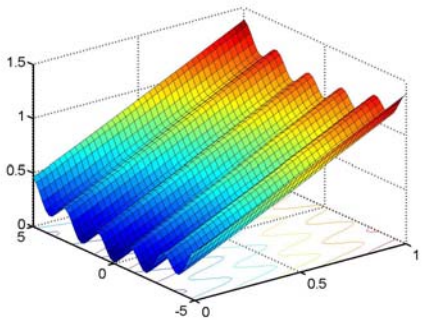
w=0



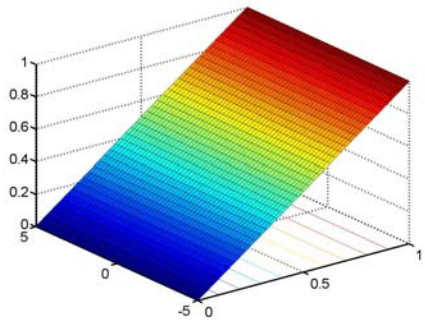
w=0.4



w=0.8



w=0.8

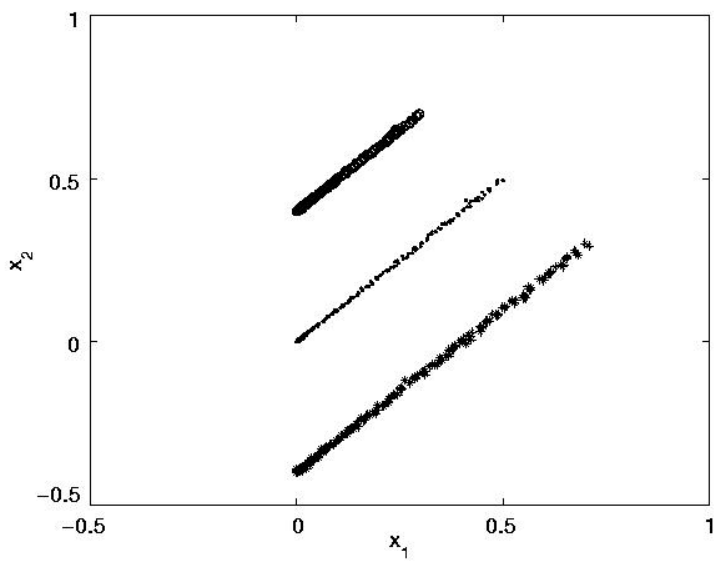


w=1

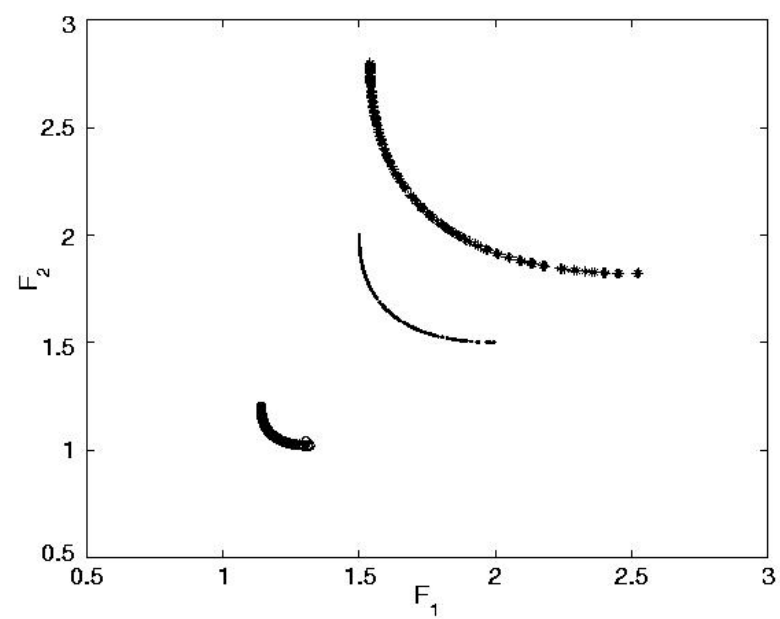
- Complexity, e.g., multi-modality, deceptiveness can be controlled in constructing MOP. In other words, existing ‘hard’ MOP test functions can be used for constructing hard DOP test functions.

Dynamic Multi-objective Optimization

A moving Pareto front constructed from a three-objective problem



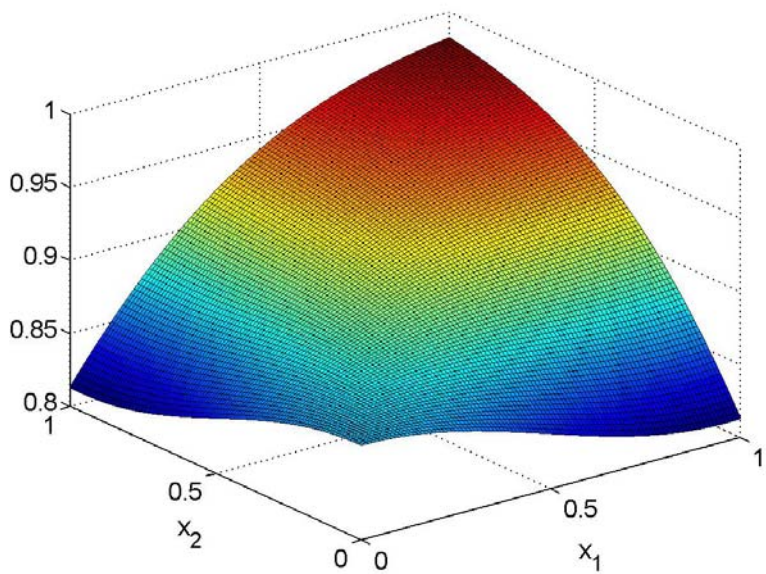
Parameter space



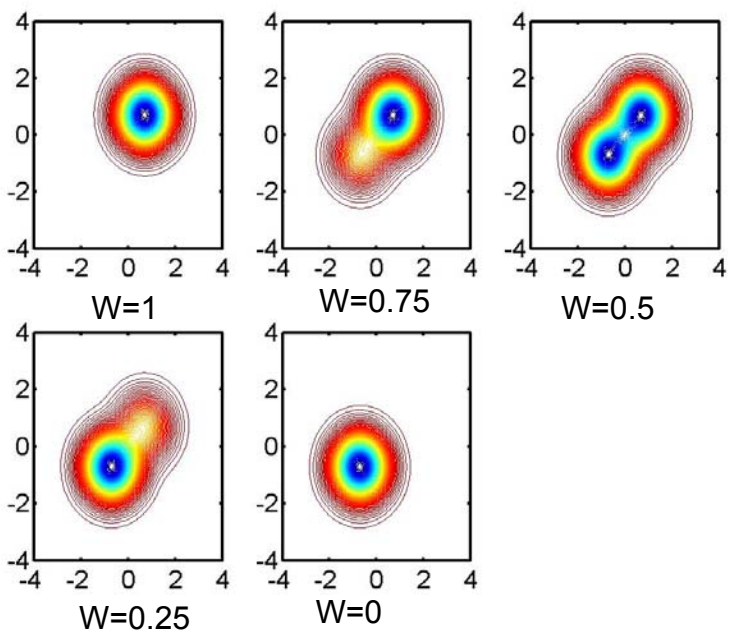
Objective space

Multi-modality Introduced by Concave MOPs

- It is well-known that the Pareto-optimal solutions located in a concave area of the Pareto-front cannot be obtained using the conventional weighted aggregation method
  - Geometric explanation
  - SOO point of view: If an MOO is concave, then there must exist a weight, so that the aggregated function is **multi-modal** (finite or infinite)



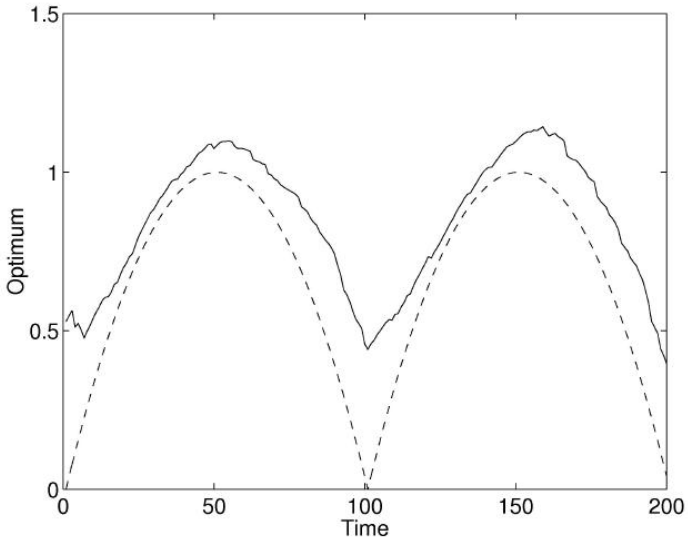
FON1



FON2

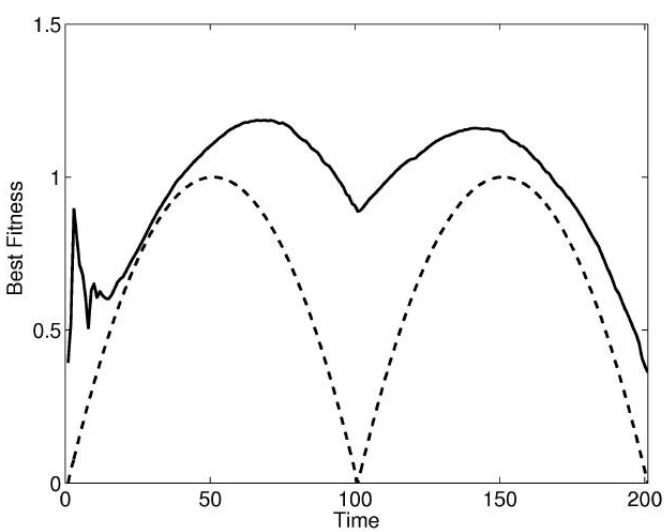
Both FON1and FON2 have a concave Pareto front.

Tracking A Slowly Moving Optimum

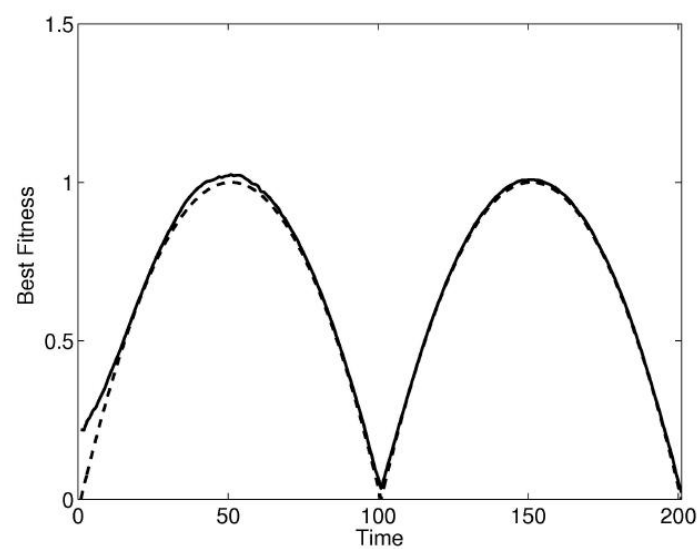


GA: popsize =100, one-point crossover, rate 0.7,
mutation rate = 1/l

- ES-CMA tracks fastest and best
- GA is something better than ES

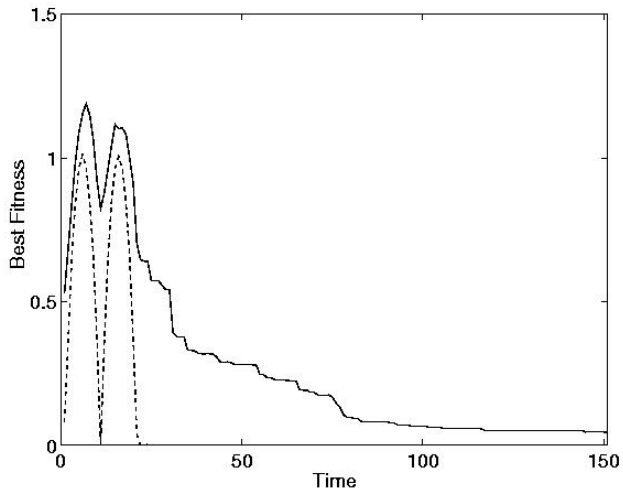


(15,100)-ES



(15,100)-ES-CMA

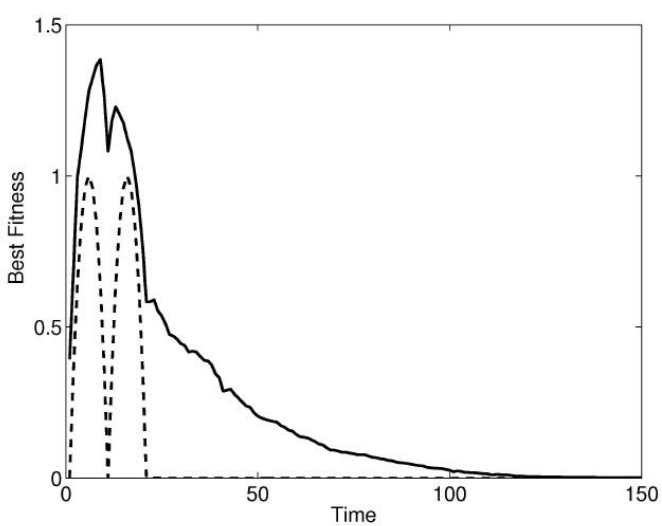
Tracking A Rapidly Moving Optimum



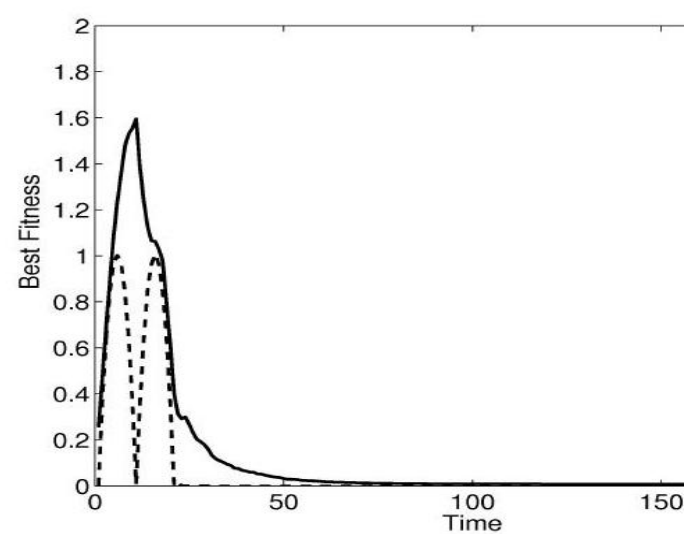
GA

GA: popsize =100, one-point crossover, rate 0.7,
mutation rate = 1/l

- No algorithm can track a rapidly moving minimum
- ES-CMA has the highest “overshoot”

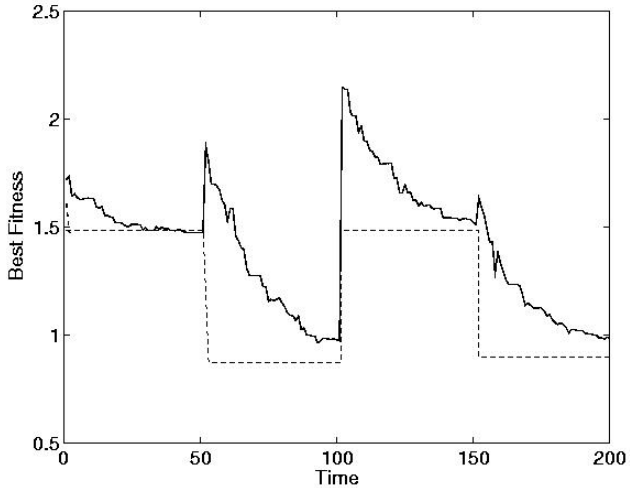


(15,100)-ES



(15,100)-ES-CMA

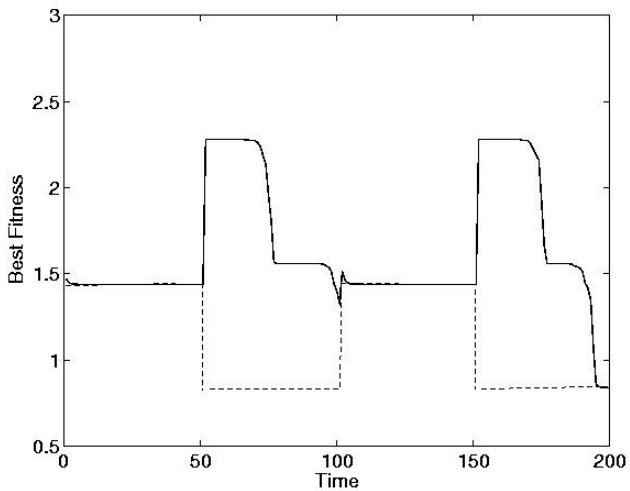
Tracking A Randomly Jumping Optimum



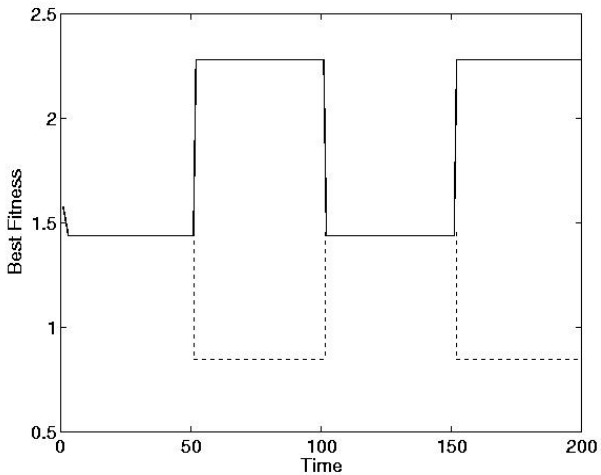
GA

GA: popsize =100, one-point crossover, rate 0.7,
mutation rate = 1/l

- GA tracks but with low speed
- ES tracks in some cases
- ES-CMA fail to track

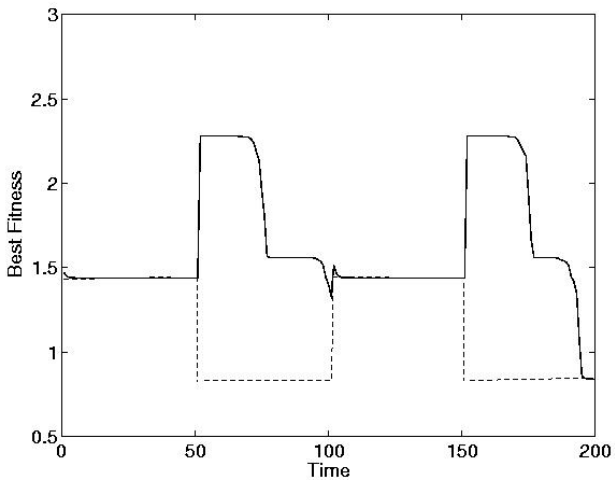


(15,100)-ES

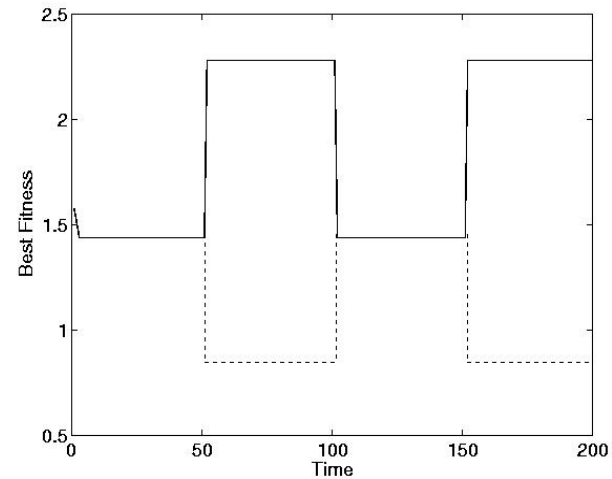


(15,100)-ES-CMA

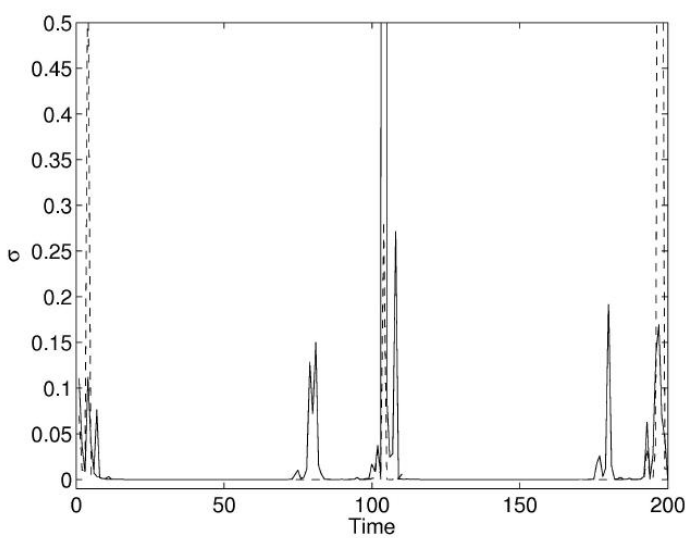
Why Evolution Strategies Fail



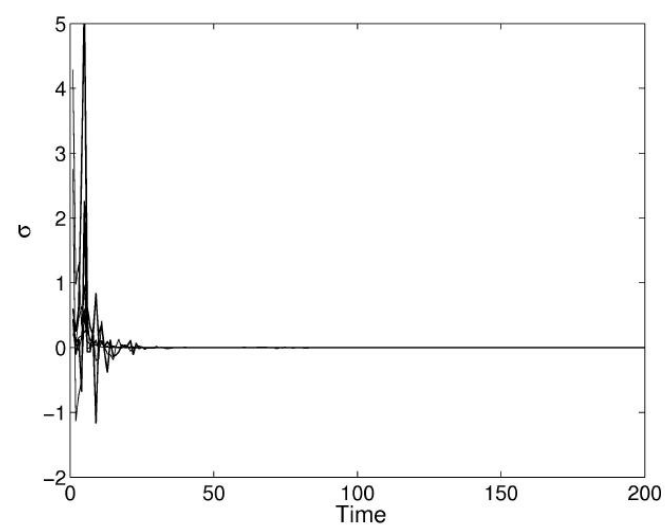
ES: Tracking trajectory



ES-CMA: Tracking trajectory

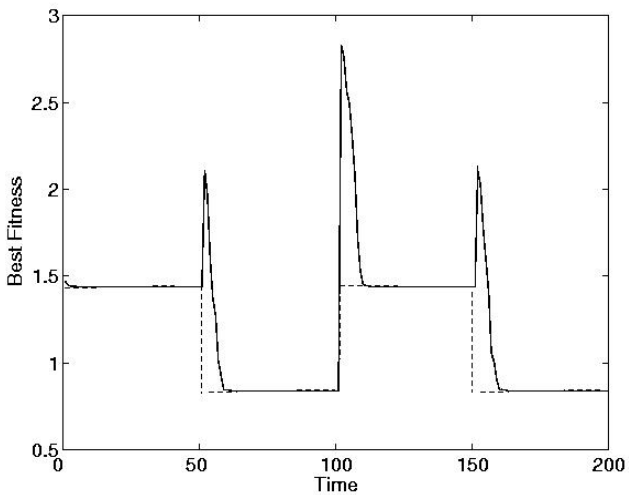


ES: Adaptation of Step-sizes

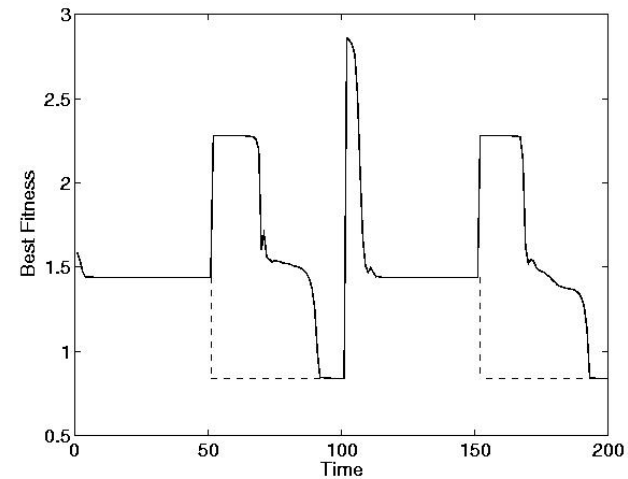


ES-CMA: Adaptation of Step-sizes

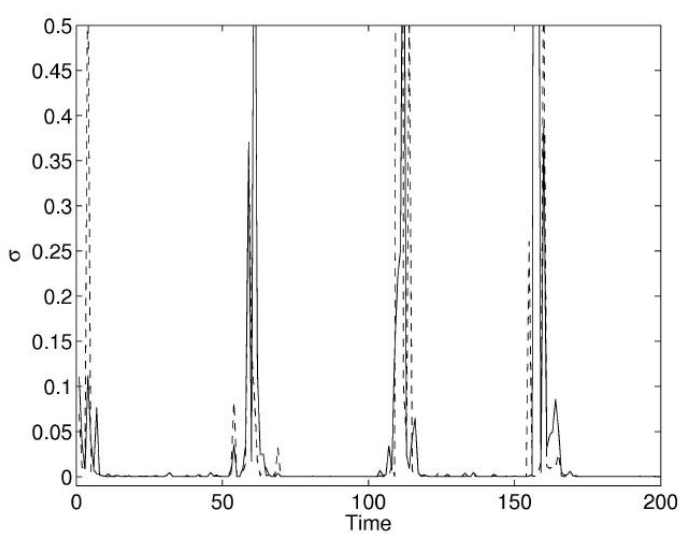
Lower-Bound Checking of Step-size



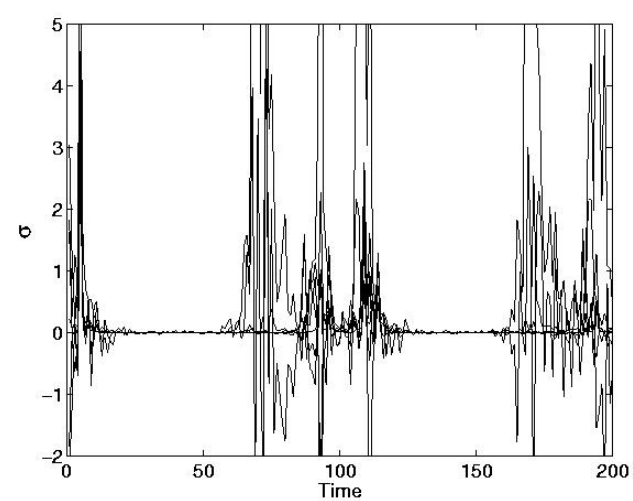
ES: Tracking trajectory



ES-CMA: Tracking trajectory



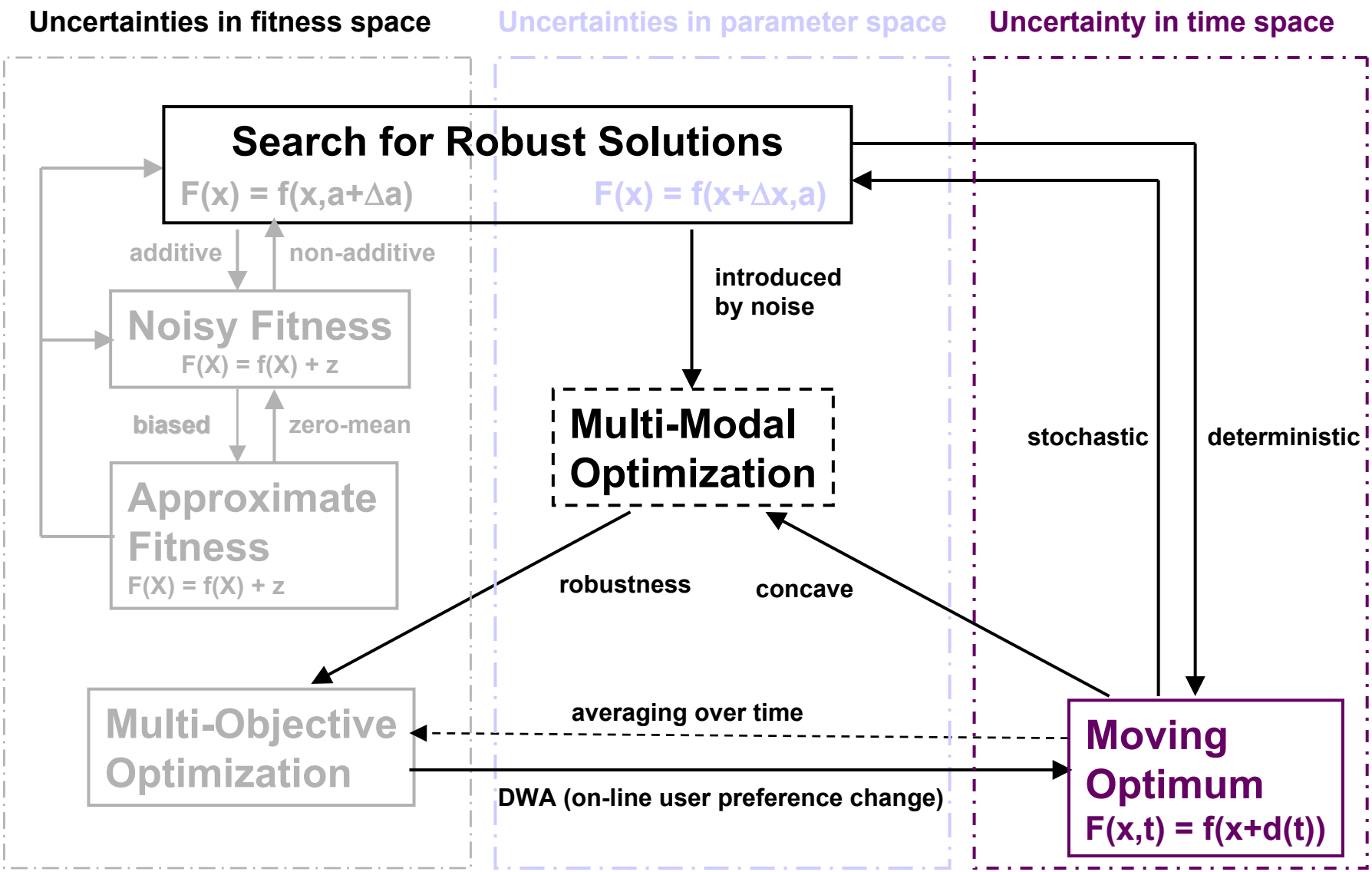
ES: Adaptation of Step-sizes



ES-CMA: Adaptation of Step-sizes

Relationships between Different Dynamic and Uncertain Optimization Problems

Relationships



Additional Information

- IEEE Computational Intelligence Society, Evolutionary Computation Technical Committee, Working Group on "Evolutionary Computation in Dynamic and Uncertain Environments"
  http://www.soft-computing.de/IEEE-ECiDUE.html
  - Bibliography on "Fitness Approximation in Evolutionary Computation" maintained by Yaochu Jin
  - Bibliography on "Evolutionary Optimization in Stochastic and Dynamic Environments" maintained by Jürgen Branke

- The 2nd European Workshop on "Evolutionary Optimization in Stochastic and Dynamic Environments" to be held in Lousanne, May 2005

- IEEE Transactions on Evolutionary Computation, Special Issue on "Evolutionary Optimization in the Presence of Uncertainties", 2005

- Soft Computing, Special Issue on "Approximation and Learning in Evolutionary Computation", 2004

- The 1st European Workshop on "Evolutionary Optimization in Stochastic and Dynamic Environments", Ciombra, Portugal, April 2004. LNCS, Springer, 2004