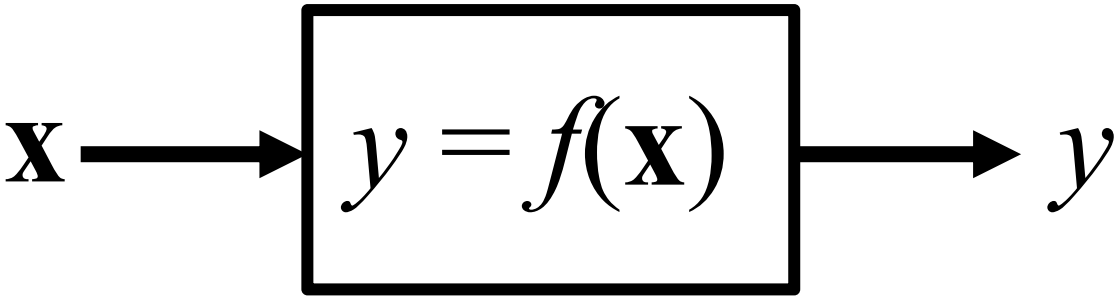
Black Box Optimization



x: an n-dimensional input vector (real number vector)

y: a real number

$f(\mathbf{x})$: Unknown function

Black Box Optimization Problem:

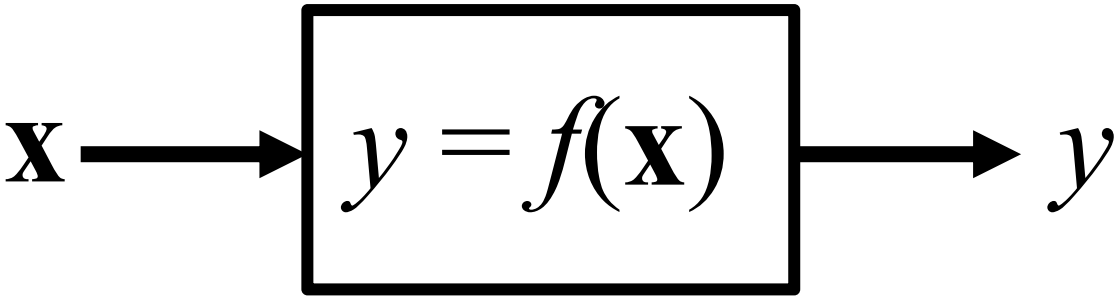
To search for the optimal solution $\mathbf{x}^*$ which minimizes y.

Given computational budget:

You can examine a pre-specified number of solutions.
(You can obtain the value of y for those solutions).

Example: You can examine 500 solutions.

Black Box Optimization



x: an n-dimensional input vector (real number vector)
y: a real number
$f(\mathbf{x})$: Unknown function

Black Box Optimization Problem:

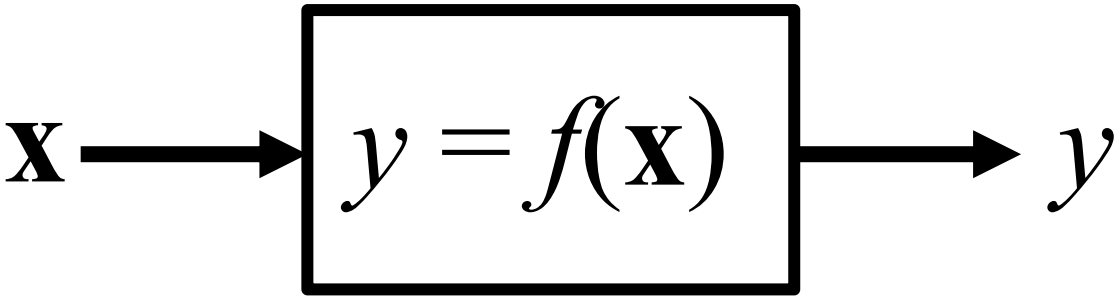
To search for the optimal solution $\mathbf{x}^*$ which minimizes y.

Given computational budget:

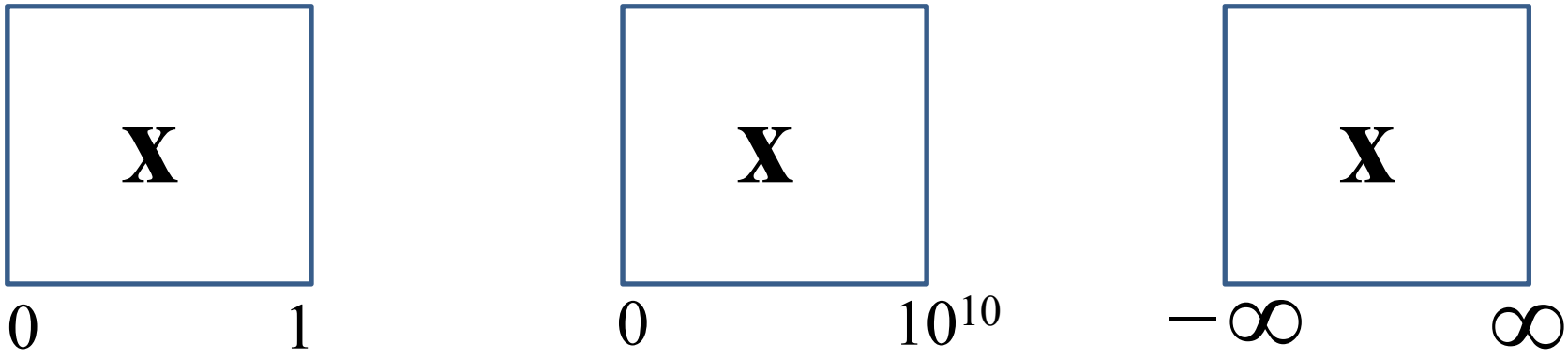
You can examine a pre-specified number of solutions.
(You can obtain the value of y for those solutions).

Example: You can examine 500 solutions.

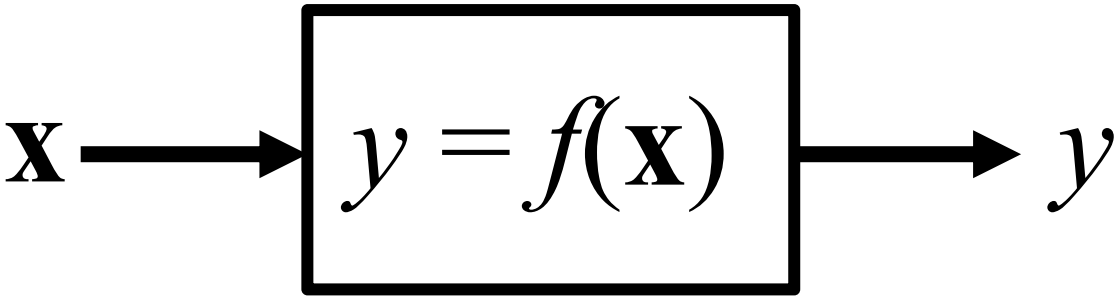
Black Box Optimization



When we have no constraint condition about x, this problem is extremely difficult and unrealistic whereas it is very interesting.

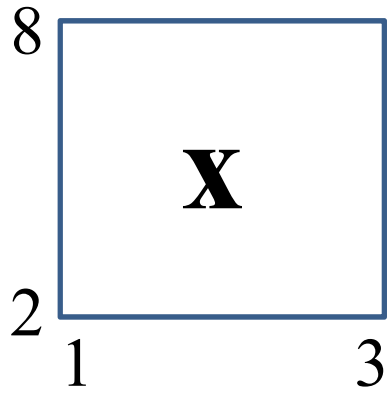


Black Box Optimization

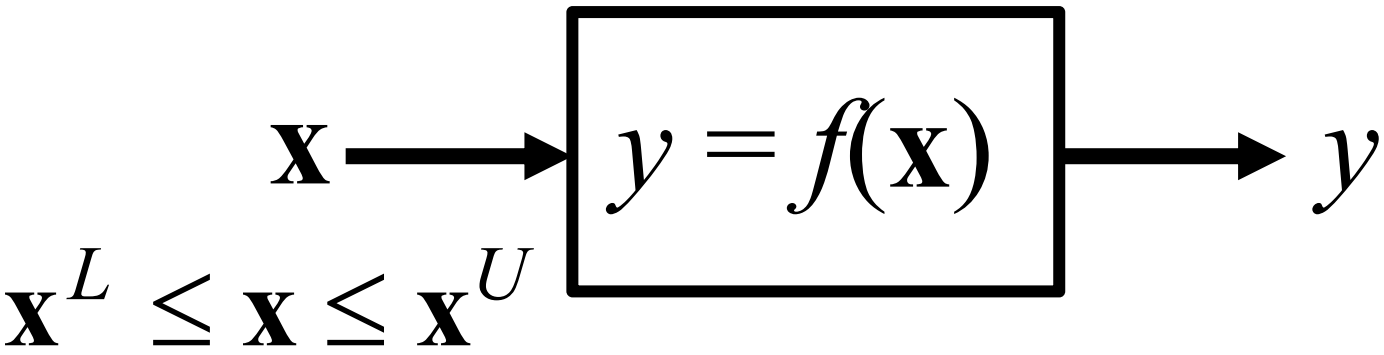


When we have no constraint condition about x, this problem is extremely difficult and unrealistic whereas it is very interesting. Usually, we have the given lower and upper bound of **x**.

$$\mathbf{x}^L \leq \mathbf{x} \leq \mathbf{x}^U$$



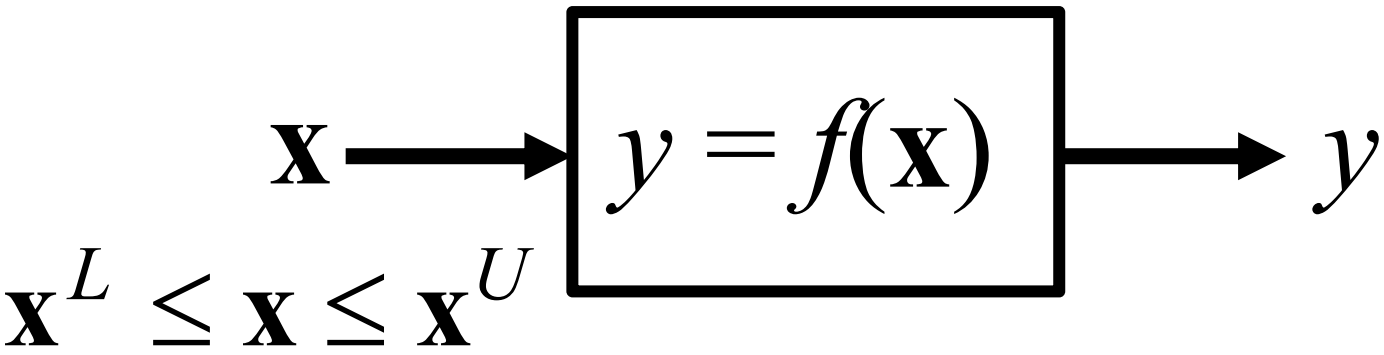
Black Box Optimization



How to solve this problem?

Your idea: _____.

Black Box Optimization



How to solve this problem?

1. Meta-heuristic algorithms such as evolutionary computation.
2. Gradient-based algorithms with gradient estimation by solution sampling in the neighborhood of the current solution.
3. Surrogate model-based algorithms.

Experiment-based Optimization

Minimize $f(\boldsymbol{x})$ subject to $\boldsymbol{x} \in \boldsymbol{X}$

where $\boldsymbol{x}$: Decision vector

$\boldsymbol{X}$: Feasible region

$f(\boldsymbol{x})$: Objective function

Main Difficulty:

The calculation (evaluation) of $f(\boldsymbol{x})$ needs an experiment (simulation), which is time-consuming, expensive, and noisy.

Experiment-based Optimization

Minimize $f(\boldsymbol{x})$ subject to $\boldsymbol{x} \in \boldsymbol{X}$

where $\boldsymbol{x}$: Decision vector

$\boldsymbol{X}$: Feasible region

$f(\boldsymbol{x})$: Objective function

Main Difficulty:

The calculation (evaluation) of $f(\boldsymbol{x})$ needs an experiment (simulation), which is time-consuming, expensive, and noisy.

A noisy optimization problem with a very limited number of solution evaluations (e.g., 100 solution evaluations).

Experiment-based Optimization

Minimize $f(\boldsymbol{x})$ subject to $\boldsymbol{x} \in \boldsymbol{X}$

where $\boldsymbol{x}$: Decision vector

$\boldsymbol{X}$: Feasible region

$f(\boldsymbol{x})$: Objective function

Main Difficulty:

The calculation (evaluation) of $f(\boldsymbol{x})$ needs an experiment (simulation), which is time-consuming, expensive, and noisy.

A noisy optimization problem with a very limited number of solution evaluations (e.g., 100 solution evaluations).

A totally different approach may be needed from standard optimization algorithms (no noise, 1,000,000 evaluations).

Experiment-based Optimization

Minimize $f(\boldsymbol{x})$ subject to $\boldsymbol{x} \in \boldsymbol{X}$

where $\boldsymbol{x}$: Decision vector

$\boldsymbol{X}$: Feasible region

$f(\boldsymbol{x})$: Objective function

The level of an experiment (simulation) is another problem.

Real-World Problem

Optimization Problem

Minimize $f(x)$

Experiment-based Optimization

Minimize $f(\boldsymbol{x})$ subject to $\boldsymbol{x} \in \boldsymbol{X}$

where $\boldsymbol{x}$: Decision vector

$\boldsymbol{X}$: Feasible region

$f(\boldsymbol{x})$: Objective function

The level of an experiment (simulation) is another problem. If we use a complicated experiment, $f(\boldsymbol{x})$ may be very accurate but its evaluation needs a long computation time and a large cost. If we use a simple experiment, its result is not accurate.

Real-World Problem

Optimization Problem

Minimize $f(x)$

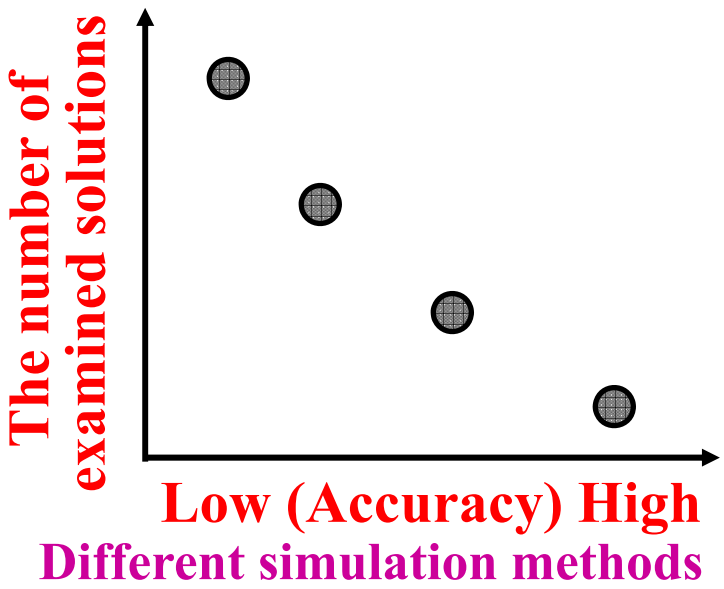
Experiment-based Optimization

Minimize $f(\boldsymbol{x})$ subject to $\boldsymbol{x} \in \boldsymbol{X}$

where $\boldsymbol{x}$: Decision vector

$\boldsymbol{X}$: Feasible region

$f(\boldsymbol{x})$: Objective function



The level of an experiment (simulation) is another problem. If we use a complicated experiment, $f(\boldsymbol{x})$ may be very accurate but its evaluation needs a long computation time and a large cost. If we use a simple experiment, its result is not accurate.

Real-World Problem

Optimization Problem

Minimize $f(x)$

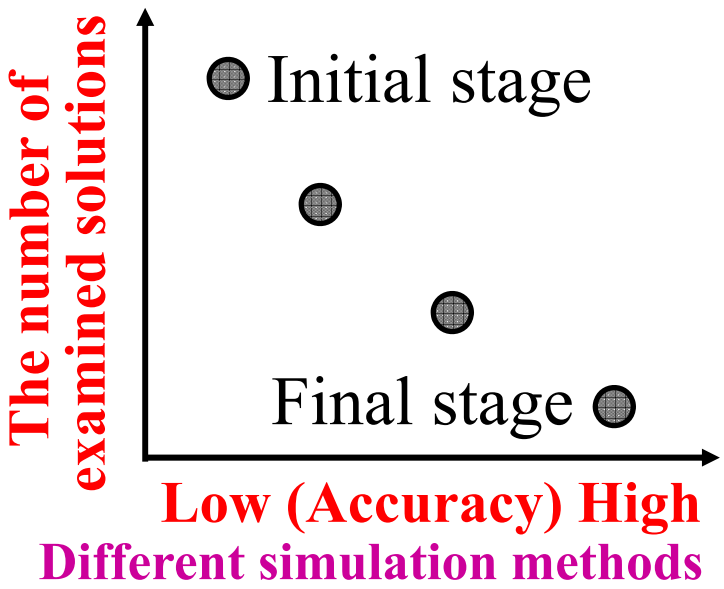
Experiment-based Optimization

Minimize $f(\boldsymbol{x})$ subject to $\boldsymbol{x} \in \boldsymbol{X}$

where $\boldsymbol{x}$: Decision vector

$\boldsymbol{X}$: Feasible region

$f(\boldsymbol{x})$: Objective function



The level of an experiment (simulation) is another problem. If we use a complicated experiment, $f(\boldsymbol{x})$ may be very accurate but its evaluation needs a long computation time and a large cost. If we use a simple experiment, its result is not accurate.

Real-World Problem

Optimization Problem

Minimize $f(x)$

Today's Plan (April 30)

Part 1:

- Noisy Optimization
- Experiment-based Optimization
- (Simulation-based Optimization)
- (Expensive Optimization)
- Surrogate-based Optimization

Part 2:

- Transfer Optimization
- Multi-Tasking Optimization

Surrogate-based Optimization

Minimize $f(\boldsymbol{x})$ subject to $\boldsymbol{x} \in \boldsymbol{X}$

where $\boldsymbol{x}$: Decision vector

$\boldsymbol{X}$: Feasible region

$f(\boldsymbol{x})$: Objective function

Main Difficulty:

The calculation of $f(\boldsymbol{x})$ is time-consuming and expensive.

Surrogate-based Optimization

Minimize $f(\boldsymbol{x})$ subject to $\boldsymbol{x} \in \boldsymbol{X}$

where $\boldsymbol{x}$: Decision vector

$\boldsymbol{X}$: Feasible region

$f(\boldsymbol{x})$: Objective function

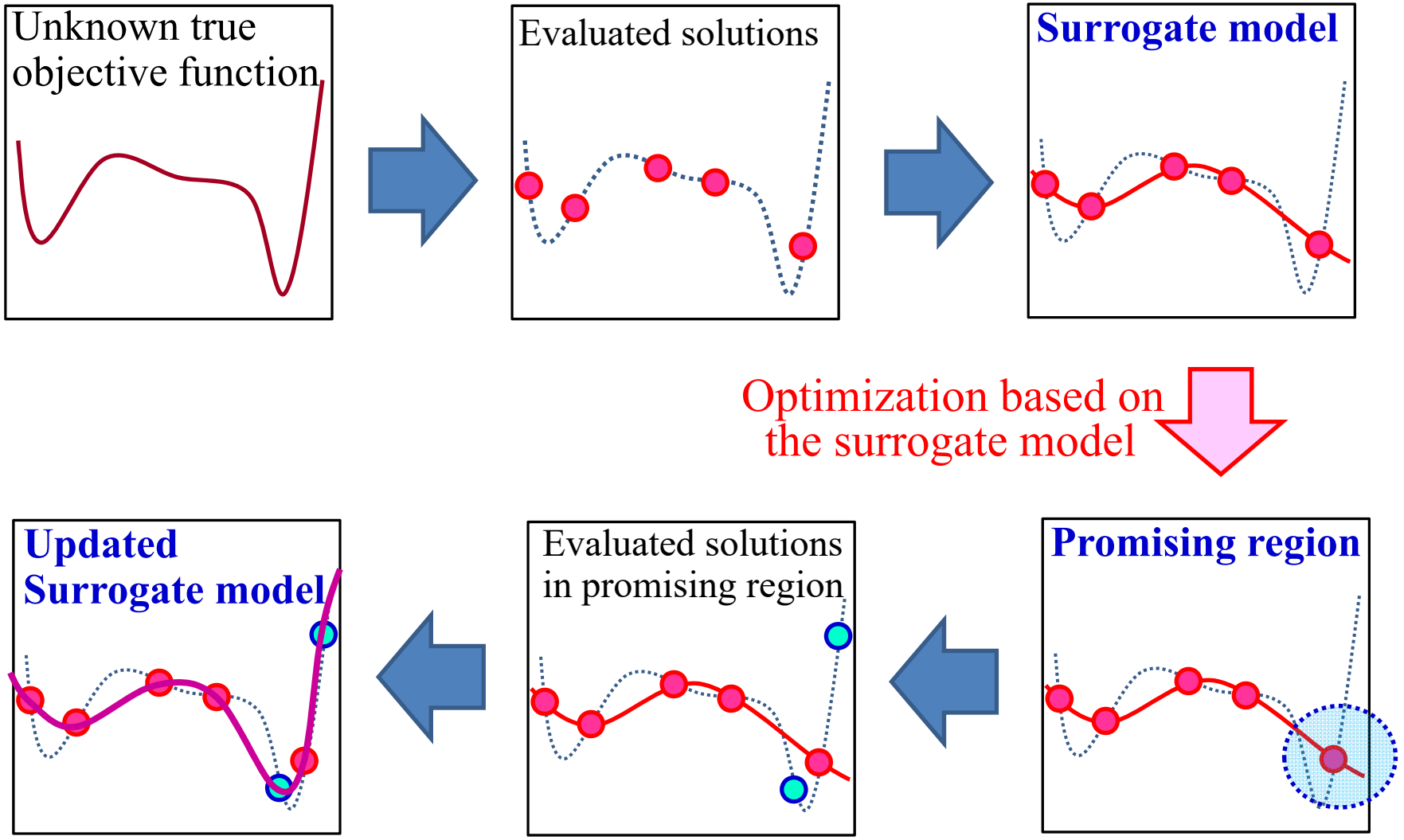
Main Difficulty:

The calculation of $f(\boldsymbol{x})$ is time-consuming and expensive.

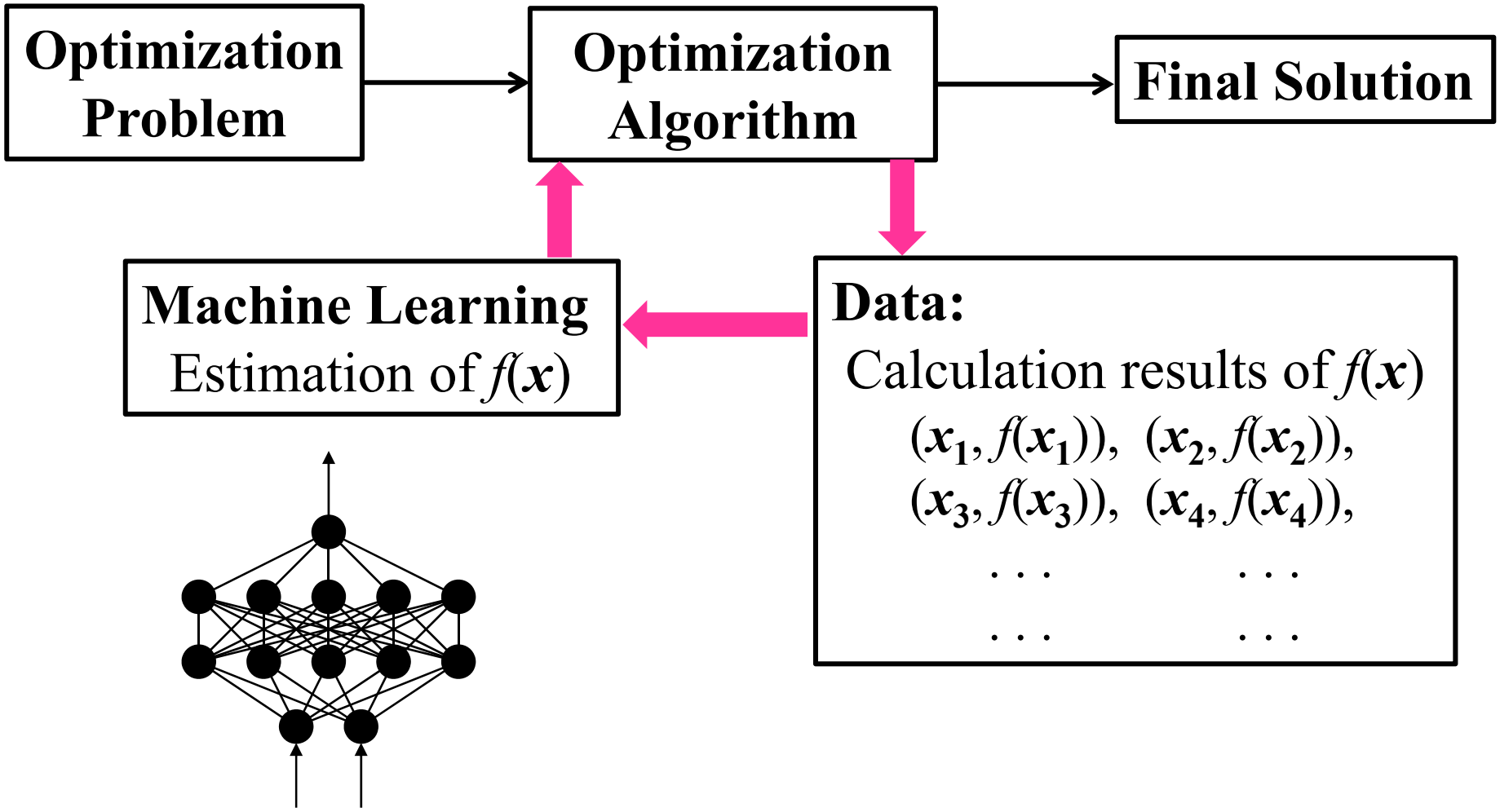
During the search (e.g., by evolutionary computation):

- $f(\boldsymbol{x})$ is calculated for only a small number of solutions.
- $f(\boldsymbol{x})$ is estimated for other solutions using the calculated solutions (using a machine learning method such as neural networks).

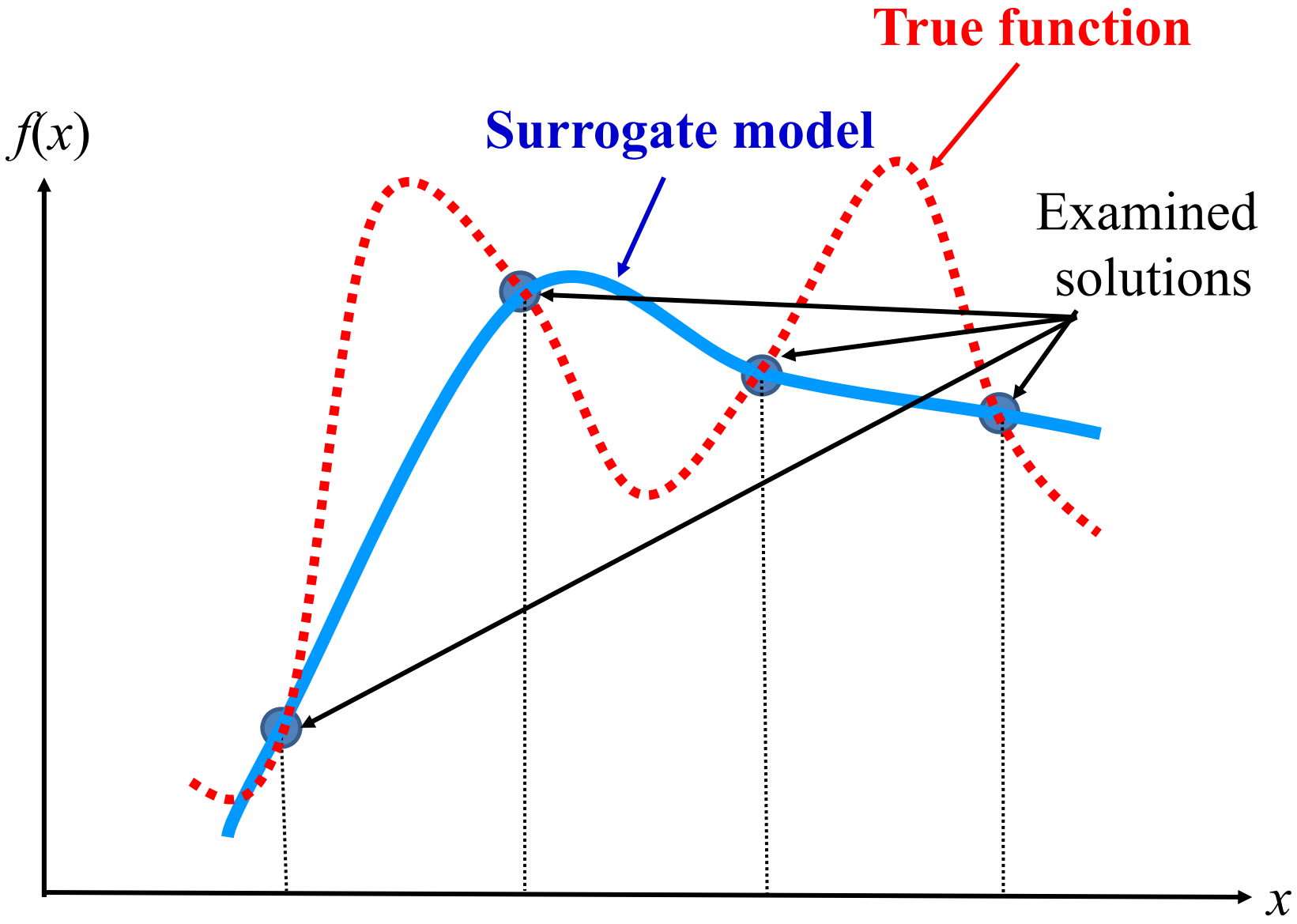
Surrogate-based Optimization



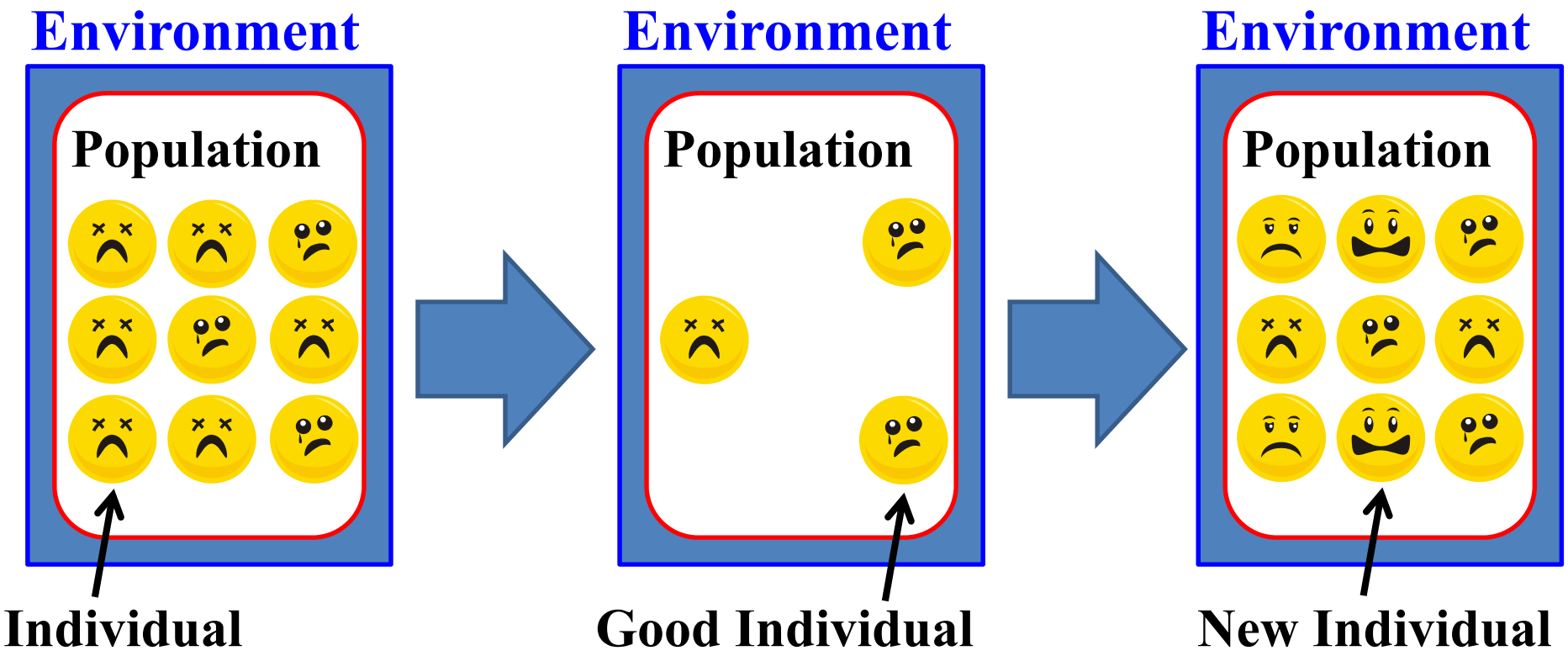
Online Data-Driven Optimization



Surrogate Model

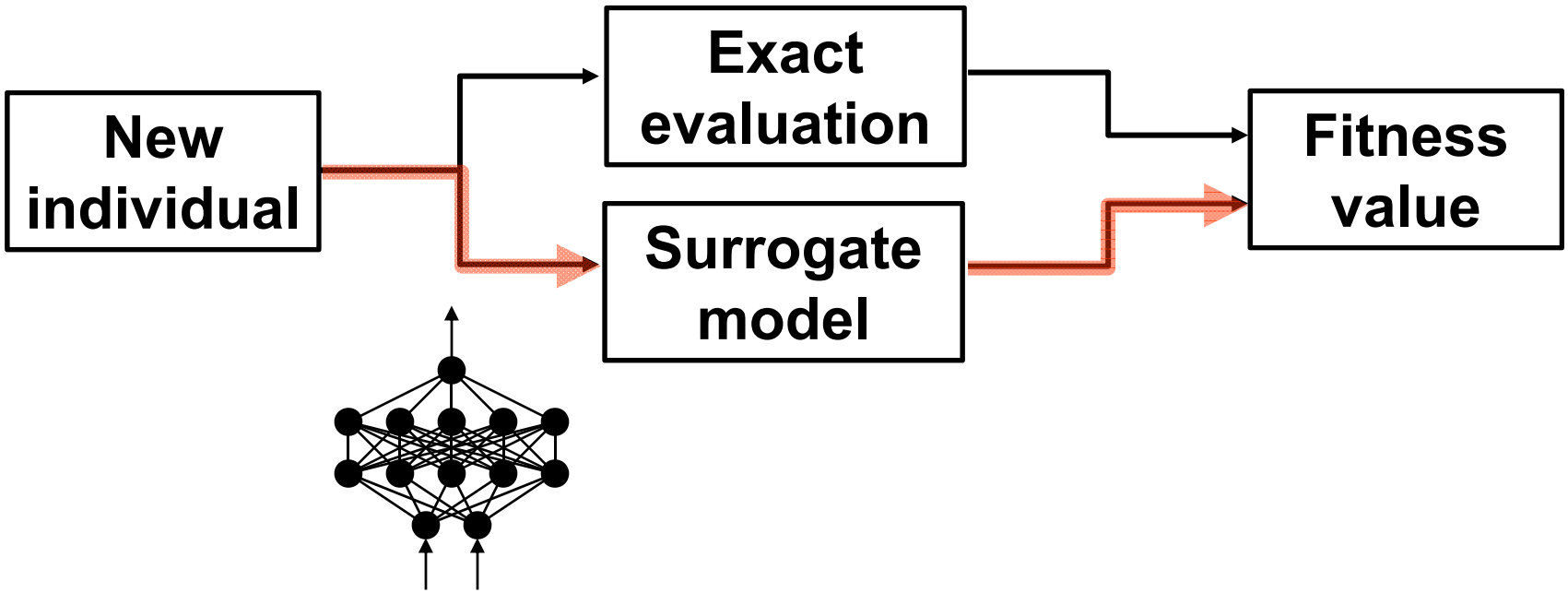


Basic Idea of Evolutionary Computation



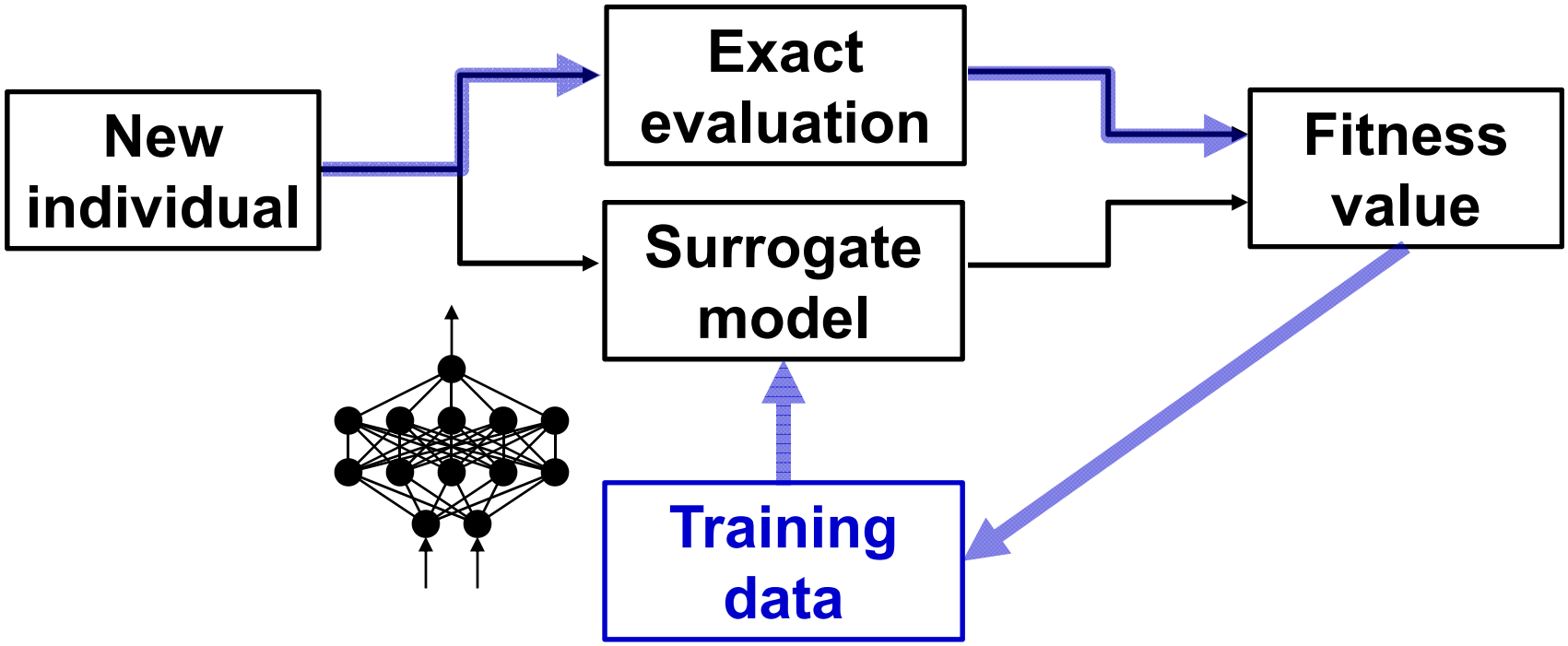
(1) A population of individuals is randomly generated.
(2) Each individual is evaluated in the environment. ⇦ Surrogate
(3) Good individuals survive probabilistically.
(4) New individuals are generated from the good individuals.

Surrogate Model



Basic Idea: To decrease the computation load for fitness evaluation by using a surrogate model.

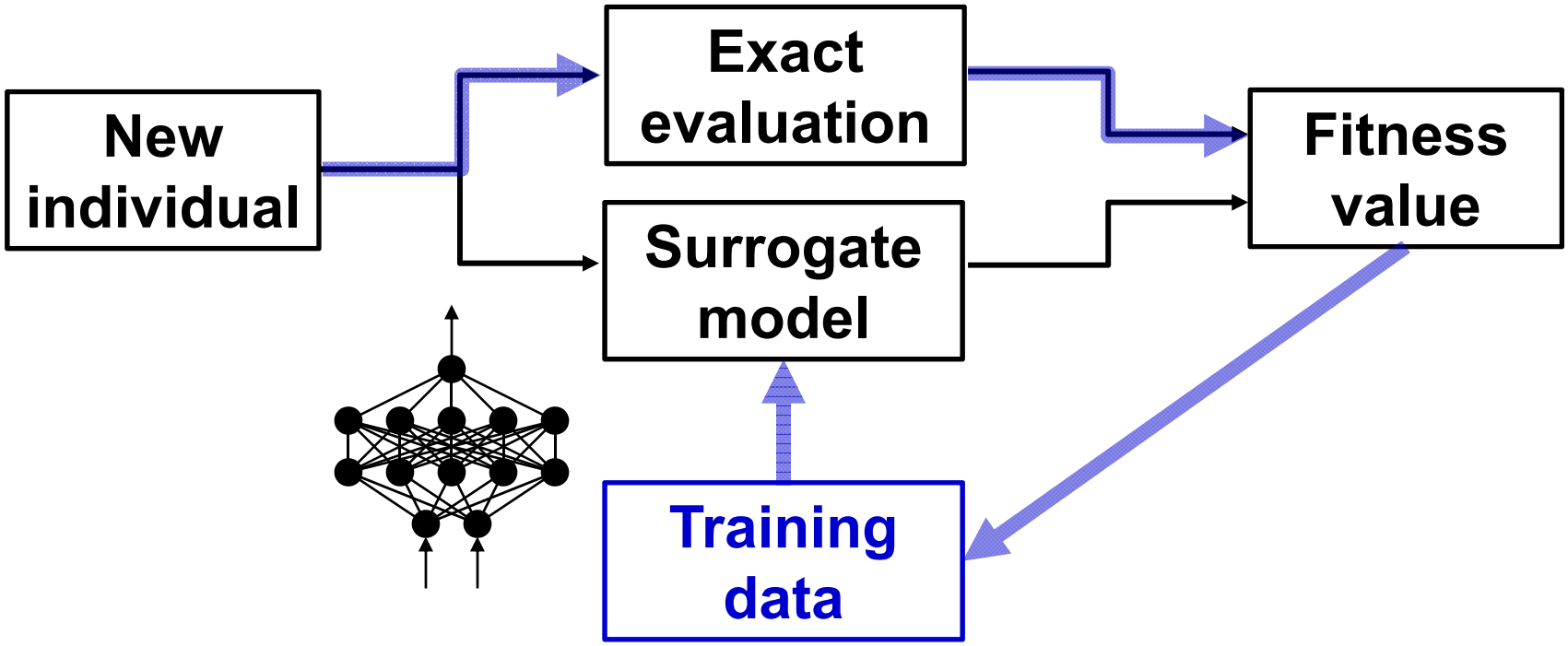
Data Collection and Model Building



1st Step: Collect training data by evaluating some solutions.

2nd Step: Train a surrogate model. This step is a machine learning problem to train a function approximator (i.e., nonlinear modelling).

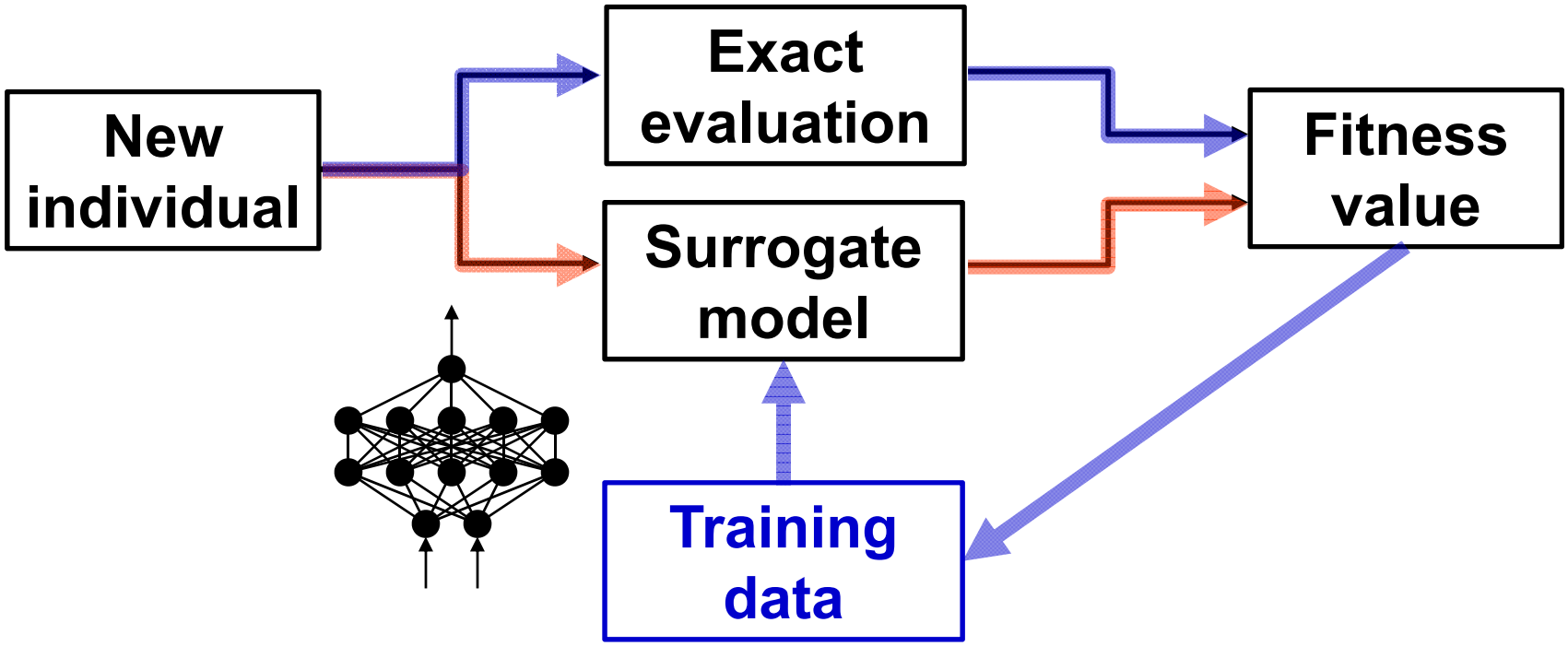
Data Collection and Model Building



Q1: Which method should be used for function approximation (e.g., multi-layer feedforward neural networks, RBF networks, fuzzy systems, polynomial models, ...).

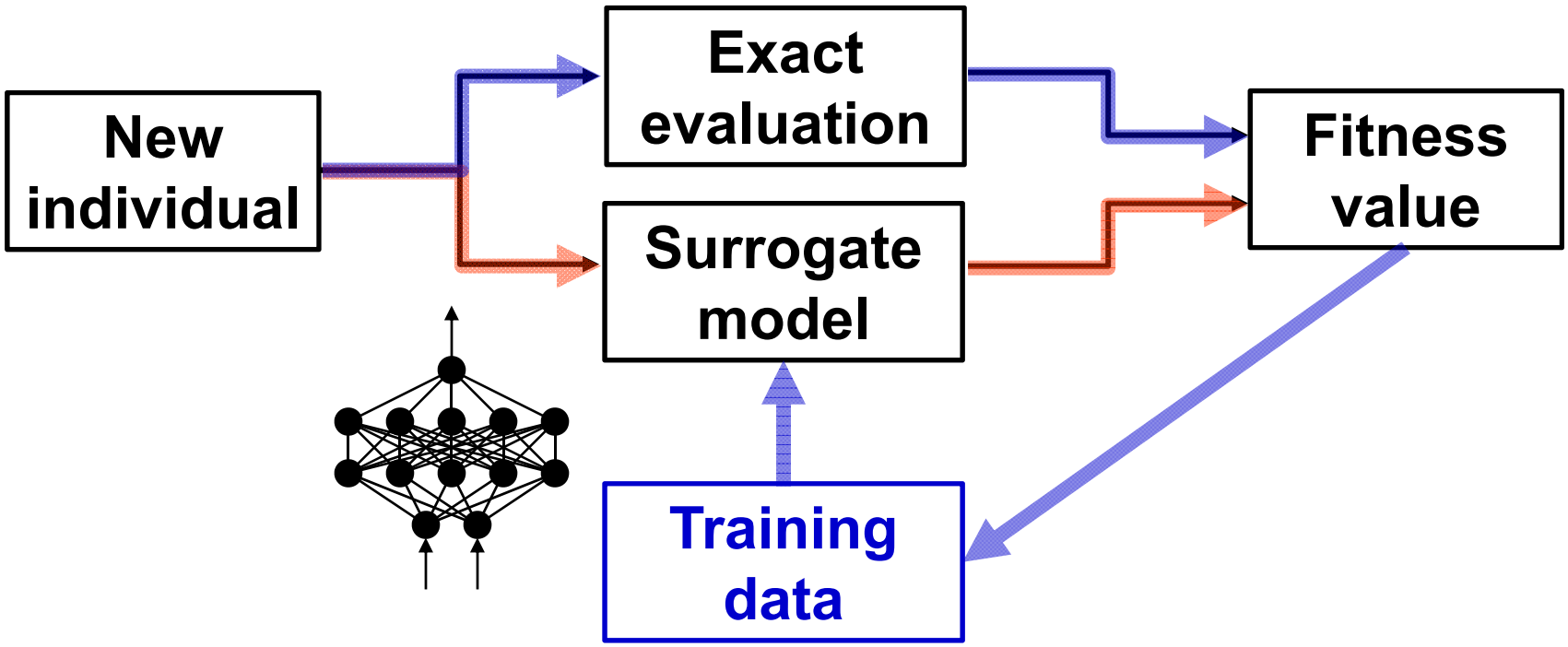
Q2: How many solutions are needed as training data?

Use of Surrogate Model



The trained surrogate model is used for fitness evaluation of each individual. However, the exact evaluation of the fitness value is also periodically performed for updating (re-training) the surrogate model in an online manner. **Q.** How often?

Use of Surrogate Model



The point is how to assign the available computation load for the exact fitness evaluation to different phases of search. One extreme strategy is to assign it only to the initial phase. Another strategy is to evenly assign it to the entire search process.

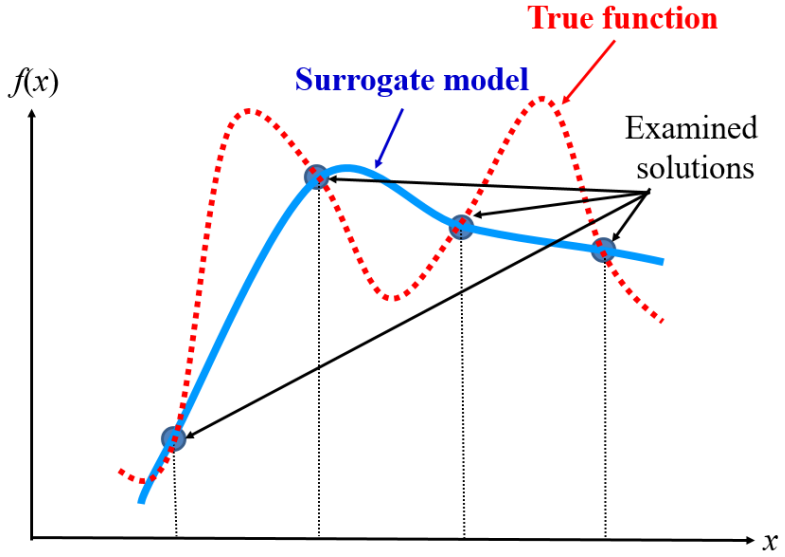
Examples (when 200 solutions can be exactly evaluated):

Implementation 1:

Use an optimization algorithm under the termination condition of 200 solution evaluations (e.g., local search with 200 solution evaluations, 10 generations of GA with population size 20, 4 generations of GA with population size 50, random creation of 200 solutions). Then train a surrogate model using the evaluated 200 solutions. After that, an algorithm is applied to the surrogate model. The final solution is the solution with the best objective value based on the surrogate model.

Q. Is this a good implementation? What is the main difficulty in this implementation?

First 200 solutions are evaluated.



Examples (when 200 solutions can be exactly evaluated):

Implementation 2:

Use an optimization algorithm under the termination condition of 100 solution evaluations. Then train a surrogate model using the evaluated 100 solutions. After that, an algorithm is applied to the surrogate model. Every 100 new solutions, one solution is evaluated. When a solution is evaluated, the surrogate model is updated. The objective function values of the other 99 solutions are estimated by the updated surrogate model. In this manner, 100 + 100 x 100 solutions are examined (i.e., evaluated or estimated). The final solution is the best solution among the actually evaluated solutions.

Q. Is this a good implementation? What is the main difficulty?

1

2

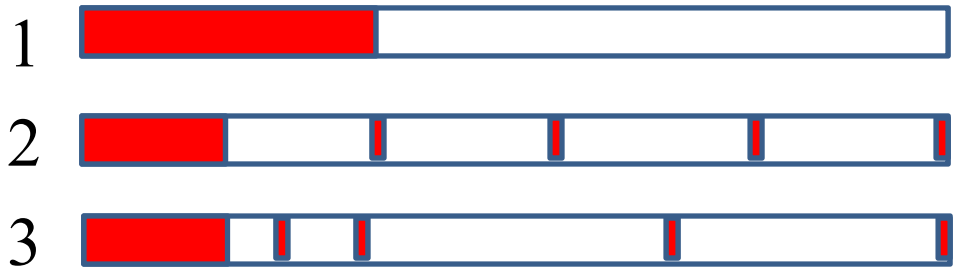
3

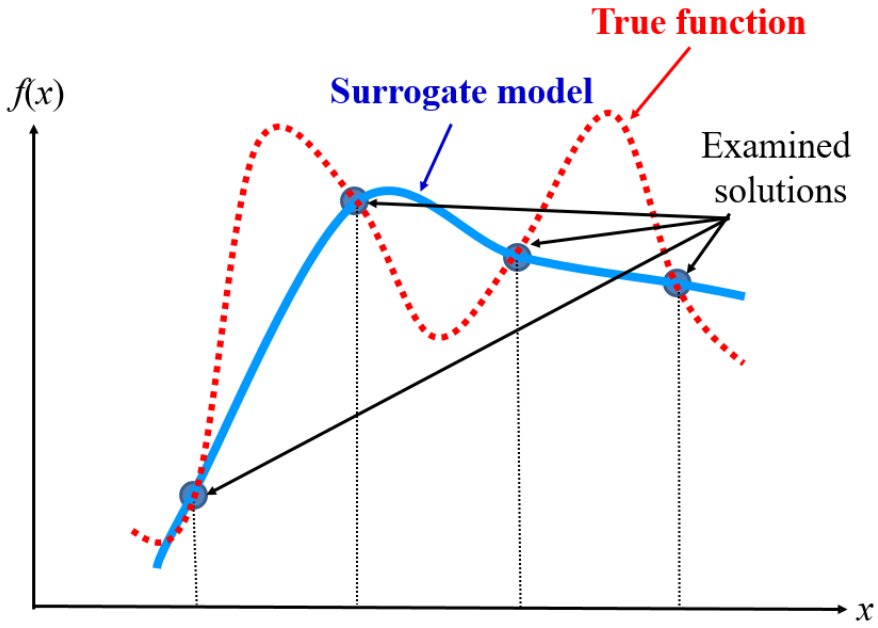
Examples (when 200 solutions can be exactly evaluated):

Implementation 3:

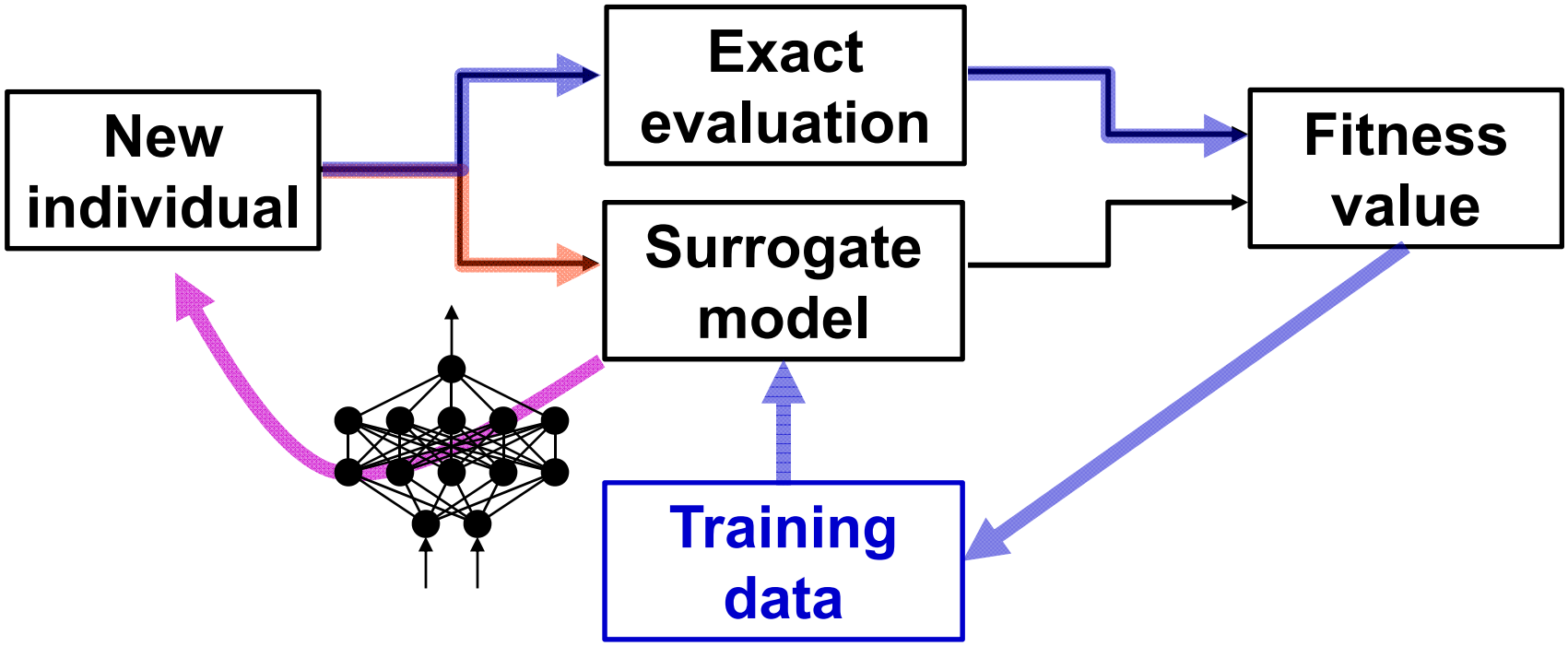
This is almost the same as Implementation 2. After the surrogate model is obtained, a solution is actually evaluated only when a better solution (based on the surrogate model) than the current best solution (based on the actual evaluation) is found.

Q. Is this a good implementation? What is the main difficulty?



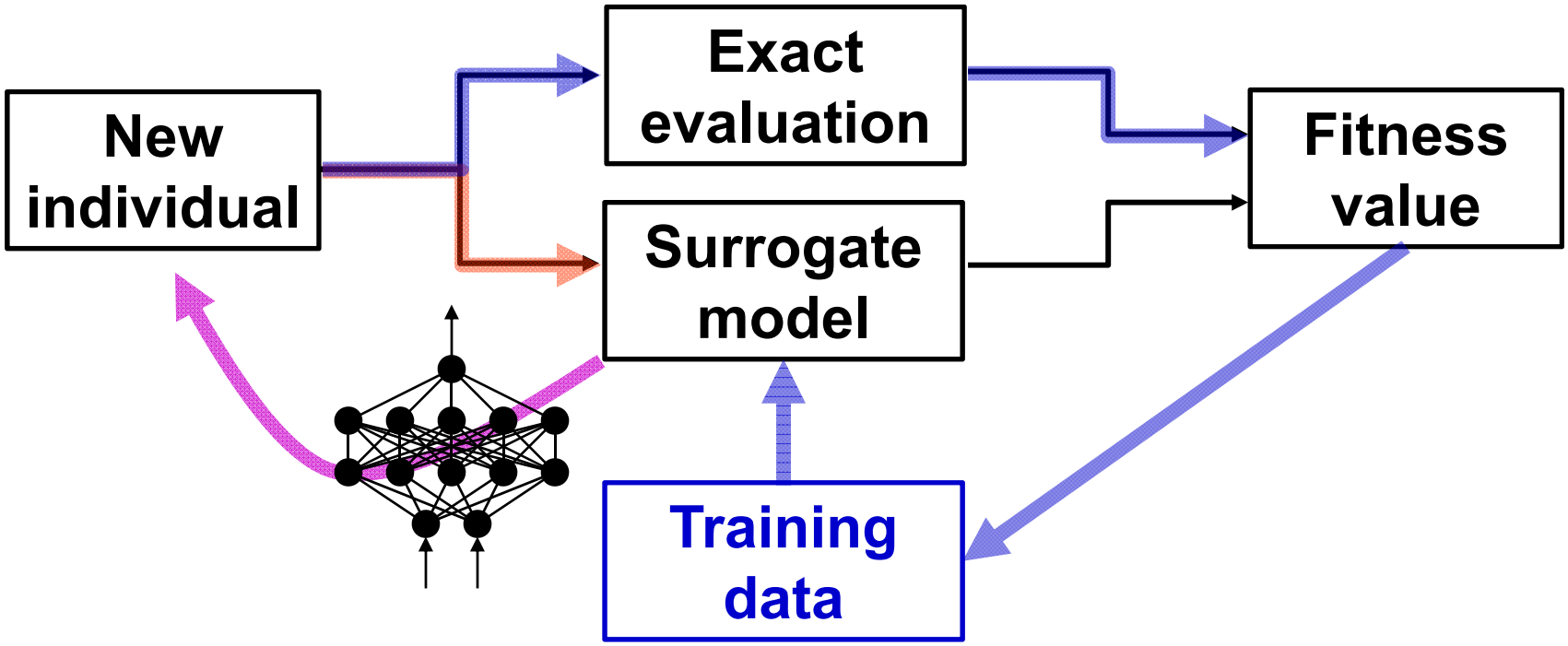


Surrogate Model for Solution Selection



The surrogate model can be used for choosing a solution to be exactly evaluated. For example, 100 solutions are evaluated using the surrogate model. Then one solution is selected for the exact evaluation.

Surrogate Model for Solution Selection



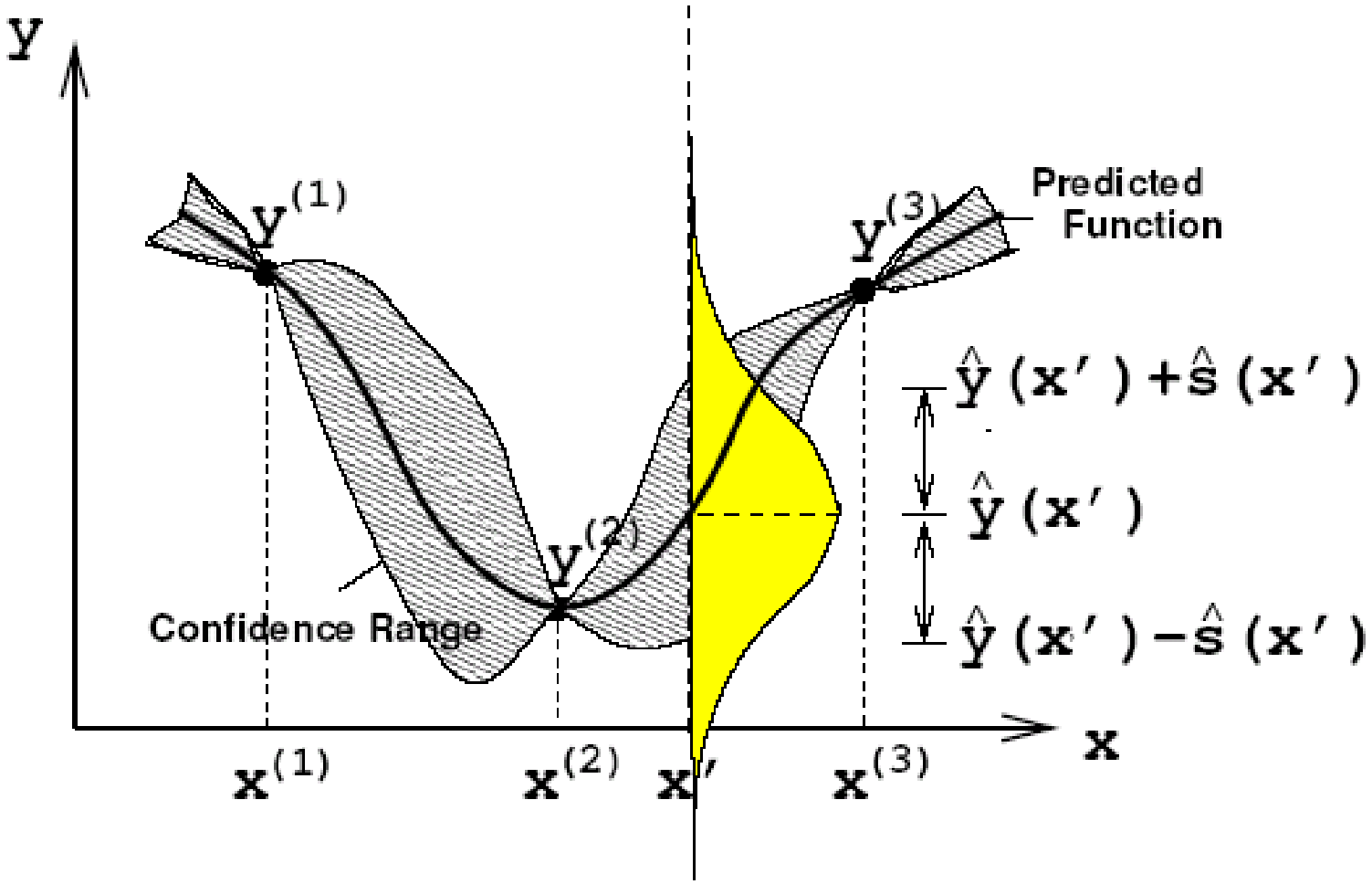
Q. How to choose a small number of solutions to be exactly evaluated from a large number of candidate solutions which have been evaluated by the surrogate model.

Surrogate Model for Solution Selection

Q. How to choose a small number of solutions to be exactly evaluated from a large number of solutions which have been evaluated by the surrogate model.

1. The solution which has the highest estimated objective value by the surrogate model.
2. The solution which is the most isolated from the evaluated solutions (i.e., with no evaluated solutions in the neighborhood).
3. The best solution by a combined criterion of 1 and 2.

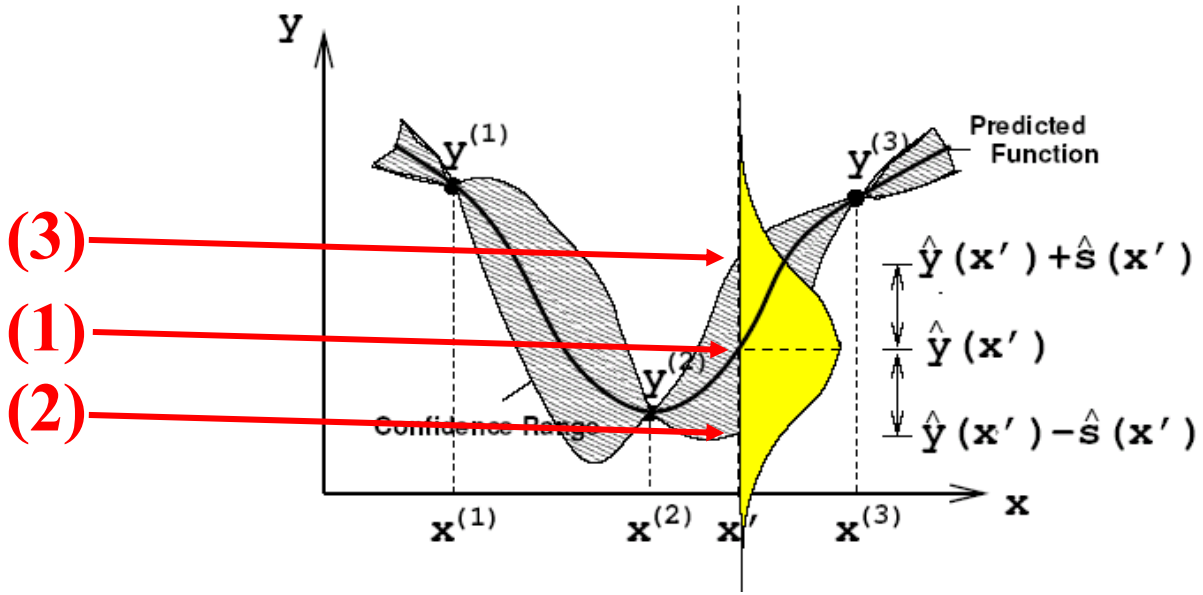
When a stochastic model is used for function approximation (i.e., when an estimated objective function value is a probability distribution or an interval):



Taken from Emmerich et al, 2005

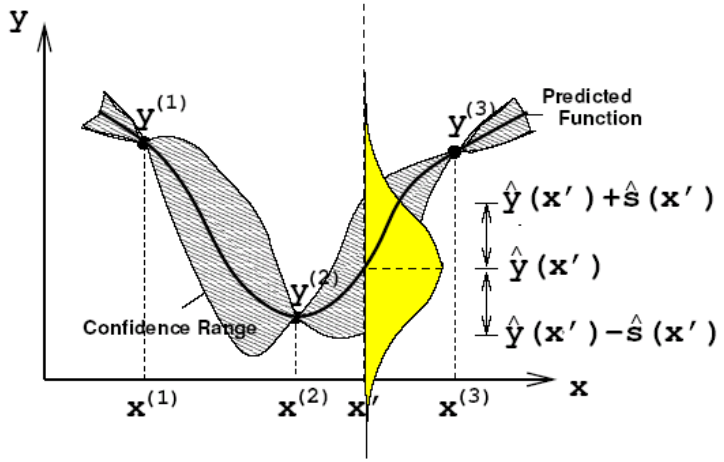
When a stochastic model is used for function approximation (i.e., when an estimated objective function value is a probability distribution or an interval):

1. The solution which has the best expected objective function value by the surrogate model (i.e., smallest center value).
2. The solution which has the best optimistic value by the surrogate model (i.e., smallest best possible value).
3. The solution which has the best pessimistic value by the surrogate model (i.e., smallest worst possible value).



When a stochastic model is used for function approximation (i.e., when an estimated objective function value is a probability distribution or an interval):

1. The solution which has the best expected objective function value by the surrogate model (i.e., smallest center value).
2. The solution which has the best optimistic value by the surrogate model (i.e., smallest best possible value).
3. The solution which has the best pessimistic value by the surrogate model (i.e., smallest worst possible value).
4. The solution which has the largest uncertainty.



Surrogate Model Evaluation

Q. How to evaluate the surrogate model?

Given: A training data set $\{(\boldsymbol{x}_i, y_i) \mid i = 1, 2, ..., N\}$

A function approximator (e.g., a neural network)

Surrogate Model Evaluation

Q. How to evaluate the surrogate model?

Given: A training data set $\{(\boldsymbol{x}_i, y_i) \mid i = 1, 2, ..., N\}$
A function approximator (e.g., a neural network)

1. Use of the mean square error for testing data in the cross-validation method:

$$\{(y_1 - f(\boldsymbol{x}_1))^2 + (y_2 - f(\boldsymbol{x}_2))^2 + \ldots \} / |\text{Testing Data}|$$

Example: 10-fold cross-validation (10-CV): The given training data are randomly divided into ten subsets of the same size. One subset is used as testing data and the other are used as training data. A model is trained by the training data and evaluated by the testing data. This is iterated 10 times for the current data set subdivision (and iterated for other subdivisions).

More straight forward method: Calculate the average errors over the actually evaluated solutions in the optimization process.

Surrogate Model Evaluation

Q. How to evaluate the surrogate model?

Given: A training data set $\{(\boldsymbol{x}_i, y_i) \mid i = 1, 2, ..., N\}$
A function approximator (e.g., a neural network)

2. Use of the average accuracy with respect to the comparison of two solutions in testing data in the cross-validation method (i.e., average accuracy with respect to the following question: Which solution is better between solutions $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$?)

This can be performed using the actually evaluated solutions in the optimization process.

Q. Which do you think a better evaluation method between the approximation accuracy and the comparison accuracy?

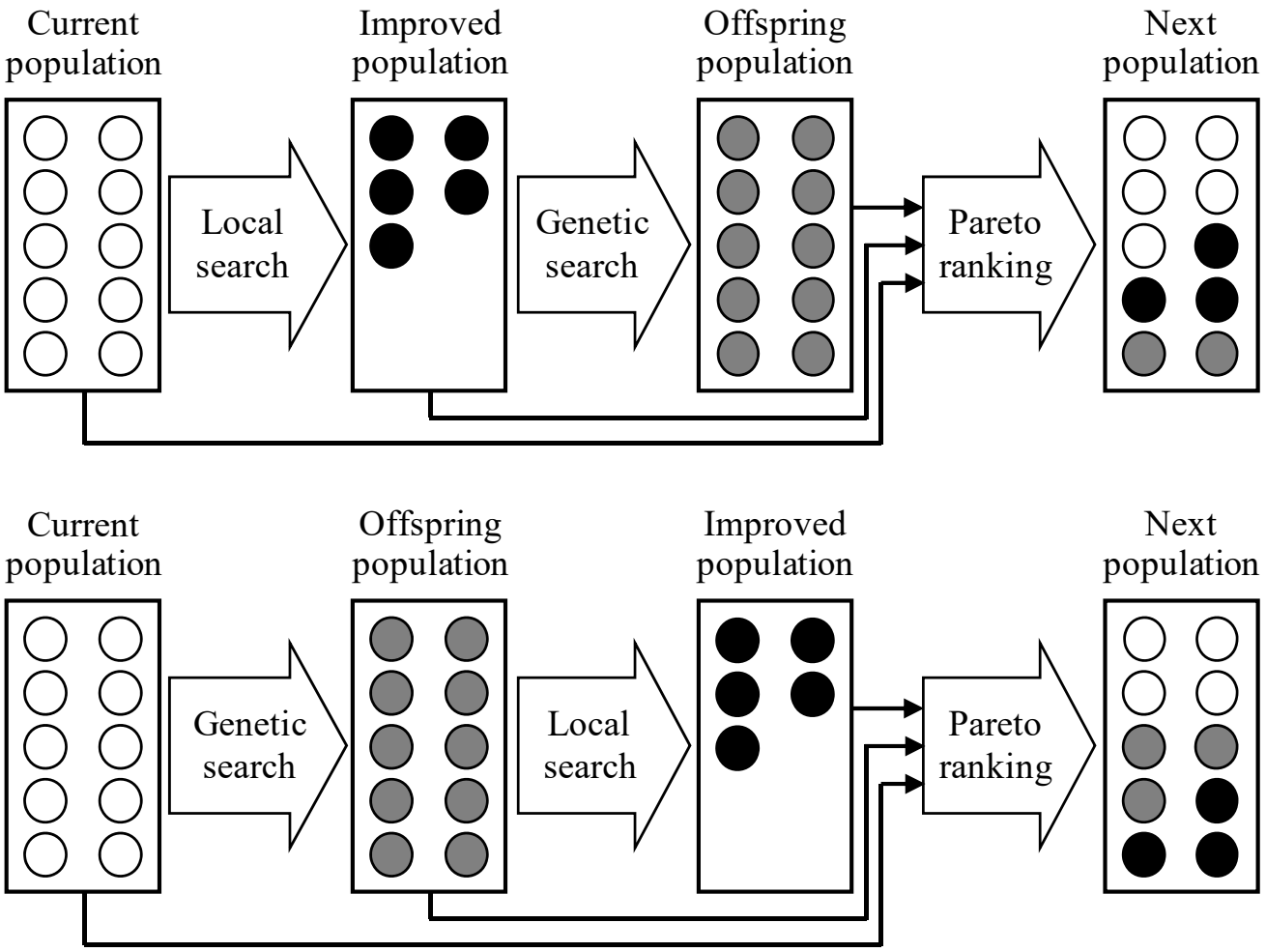
Example of Implementation:

Ensemble-based surrogate model

Multiple surrogate models are used for approximating a single fitness function. The average value or the weighted average value is the final estimated fitness value.

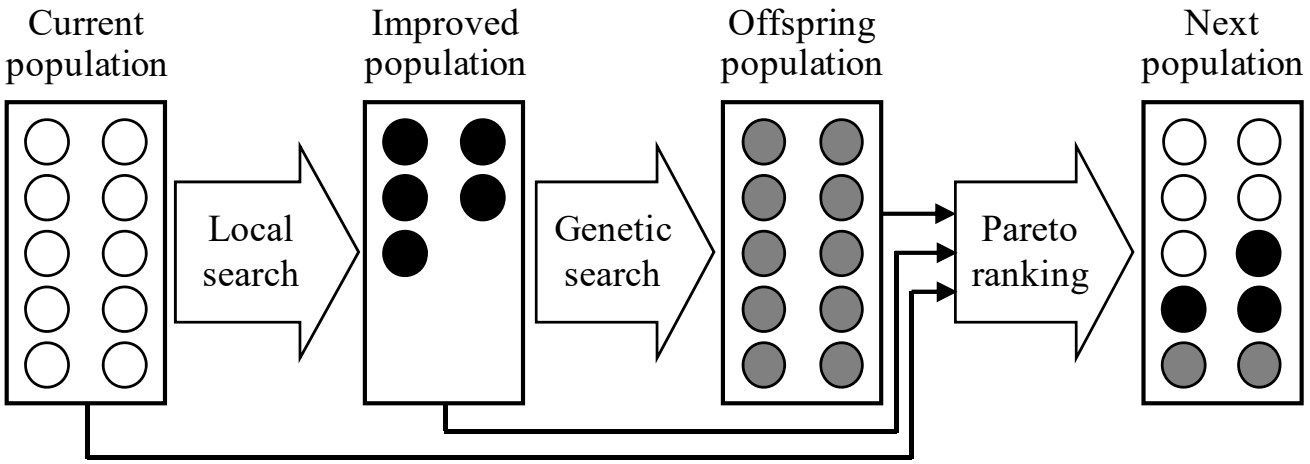
Example of Implementation:

Surrogate-based Memetic Algorithm



Example of Implementation:

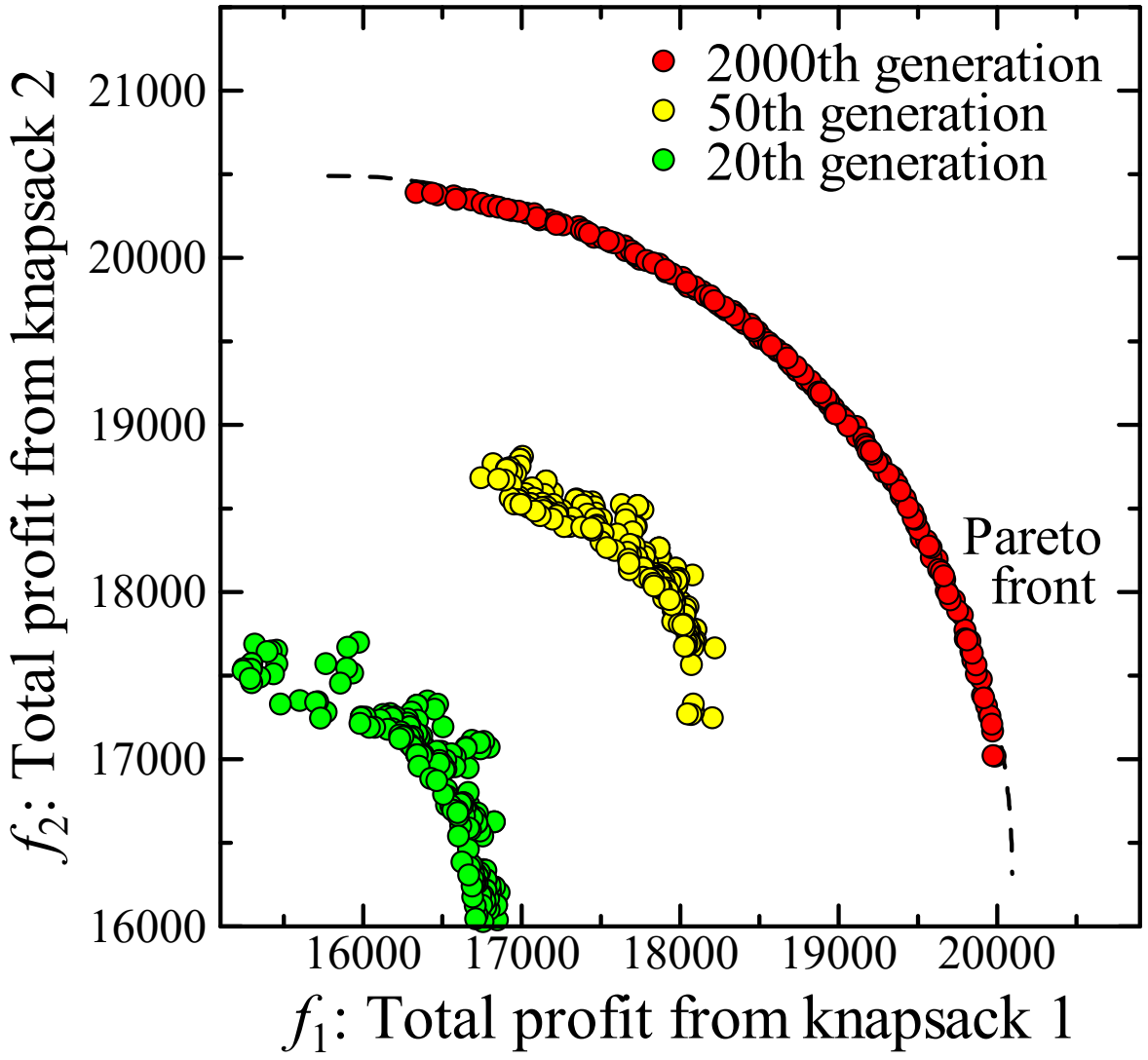
Surrogate-based Memetic Algorithm



- Local search is performed using the surrogate model.
- If a current solution is improved by local search, the improved solution is evaluated (i.e., exact fitness evaluation).
- All solutions in the main evolutionary part are evaluated exactly whereas local search is performed based on the surrogate model.

Example of Implementation:

Surrogate-based Multiobjective Algorithm

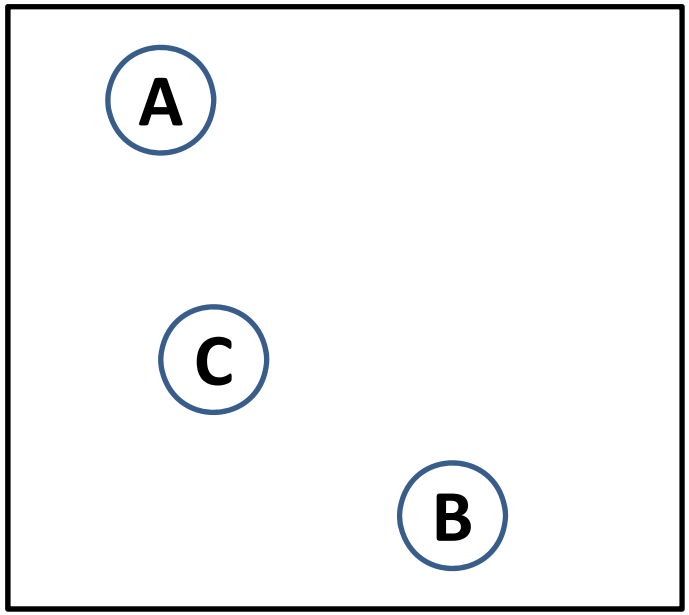


How to Use Surrogate Models in EMO Algorithms?

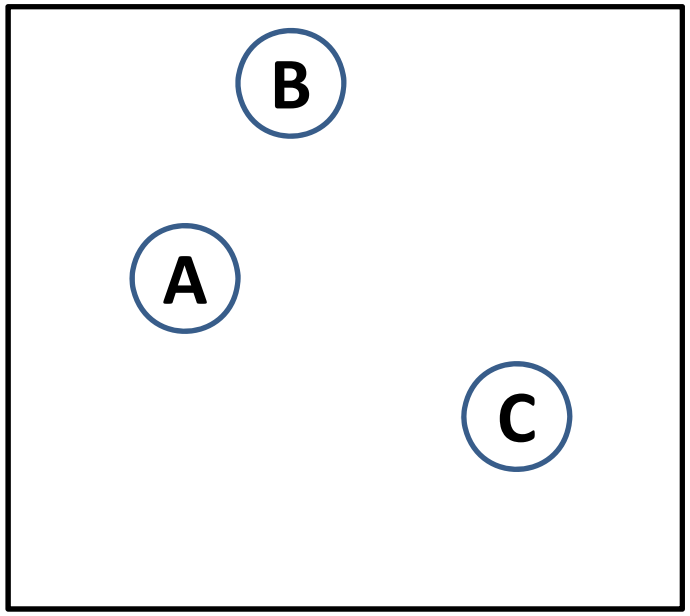
(1) Function approximation for each objective function.

(2) Function approximation for aggregated functions (e.g., scalarizing functions in MOEA/D).

(3) Approximation of the Pareto dominance relation between two solutions $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$: Non-dominated with each other, $\boldsymbol{x}_i$ is better, or $\boldsymbol{x}_j$ is better. (Special three-class classification problem)



Decision space

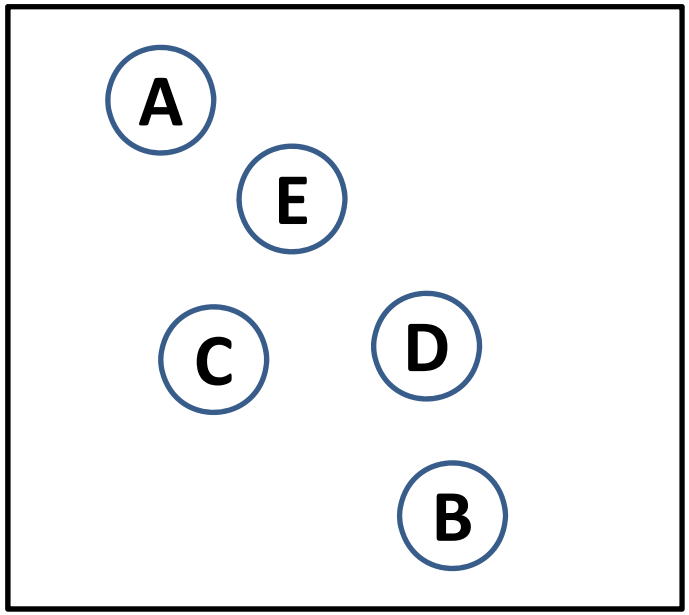


Objective space (minimization)

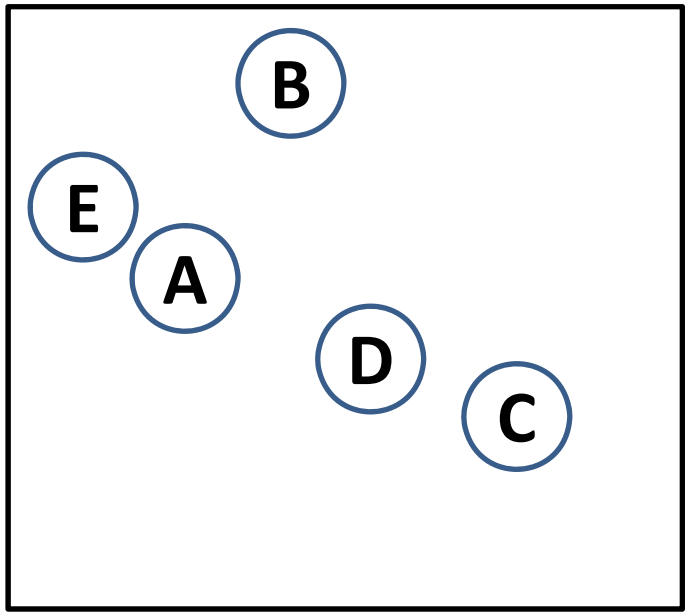
How to Use Surrogate Models in EMO Algorithms?

(4) Approximation of the comparison between two solutions $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ by a particular fitness evaluation mechanism in an EMO algorithm (e.g., NSGA-II): $\boldsymbol{x}_i$ is better or $\boldsymbol{x}_j$ is better.

(Special two-class classification problem)



Decision space



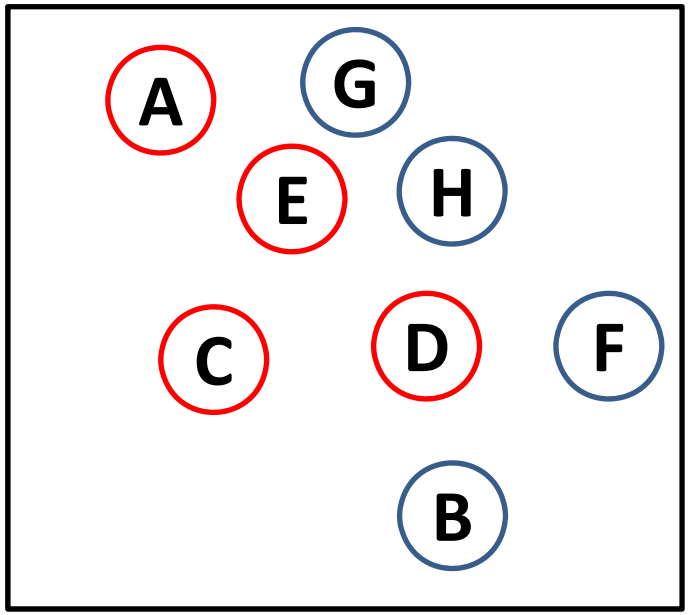
Objective space (minimization)

How to Use Surrogate Models in EMO Algorithms?

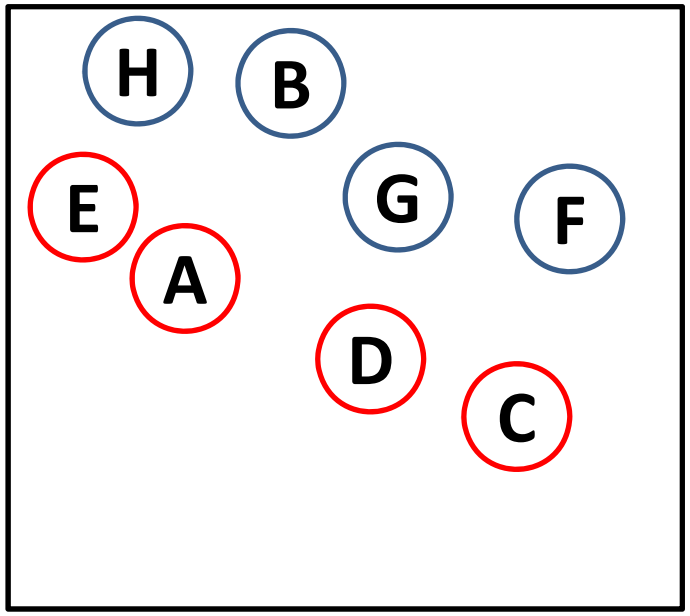
(4) Approximation of the comparison between two solutions $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ by a particular fitness evaluation mechanism in an EMO algorithm (e.g., NSGA-II): $\boldsymbol{x}_i$ is better or $\boldsymbol{x}_j$ is better.

(5) Classification of solutions in the current population between dominated and non-dominated solutions.

(Standard two-class classification problem)



Decision space



Objective space (minimization)

How to Use Surrogate Models in EMO Algorithms?

(1) Function approximation for each objective function.

(2) Function approximation for aggregated functions (e.g., scalarizing functions in MOEA/D).

(3) Approximation of the Pareto dominance relation between two solutions $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$: Non-dominated with each other, $\boldsymbol{x}_i$ is better, or $\boldsymbol{x}_j$ is better.

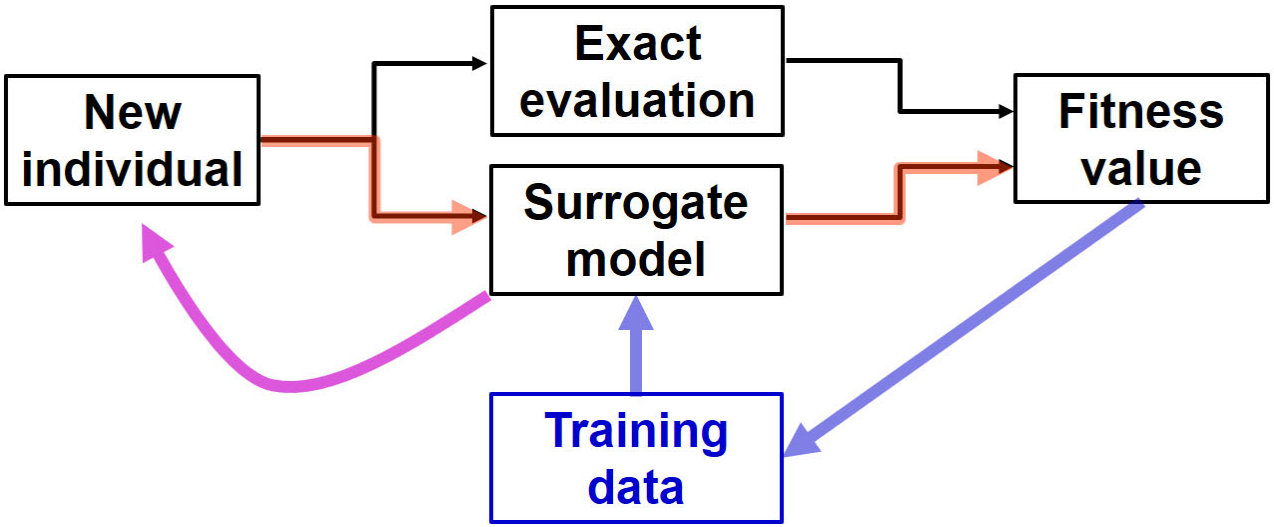
(4) Approximation of the comparison between two solutions $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ by a particular fitness evaluation mechanism in an EMO algorithm (e.g., NSGA-II): $\boldsymbol{x}_i$ is better or $\boldsymbol{x}_j$ is better.

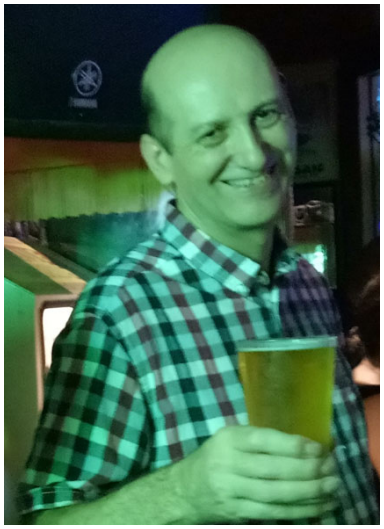
(5) Classification of solutions in the current population between dominated and non-dominated solutions.

Q1. Which do you think a good strategy?

Example of Implementation:

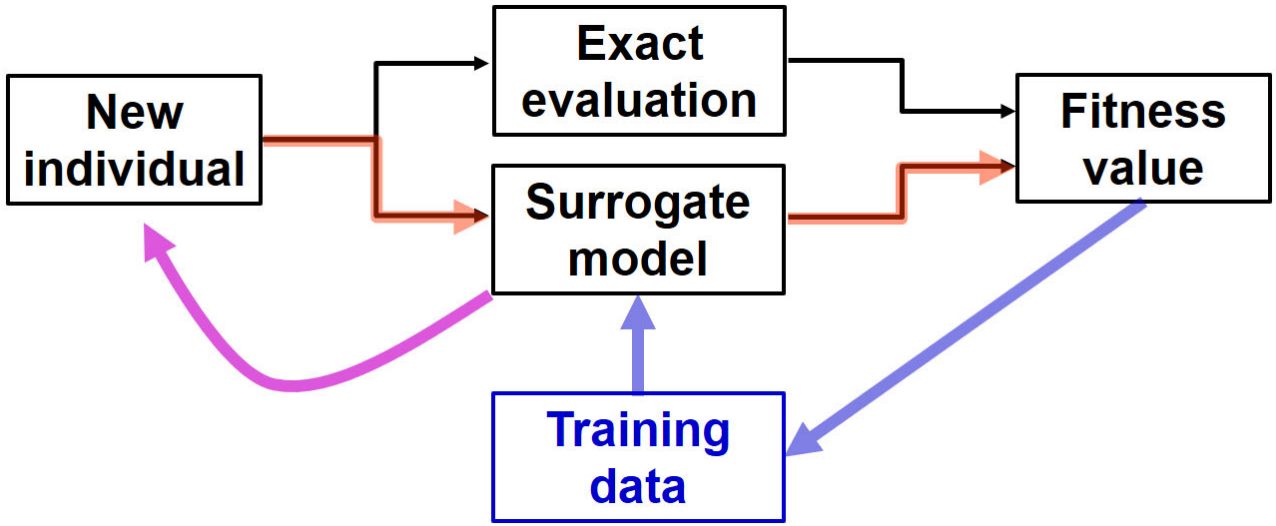
Surrogate-based Interactive Algorithm





A surrogate model can be used to replace a human evaluation.

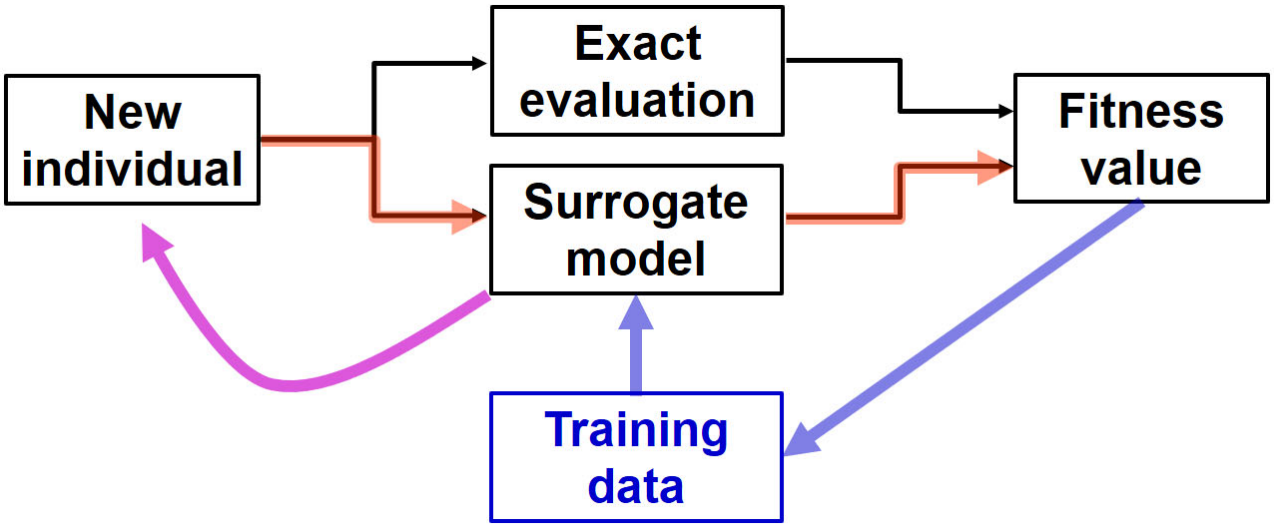
Other Related Topics to Surrogate Model



Model Complexity

An appropriate surrogate model may be different between an initial phase with a small training data set and a final phase with a large training data set. The requested accuracy for the surrogate model may be different between the initial phase and the final phase of search.

Other Related Topics to Surrogate Model



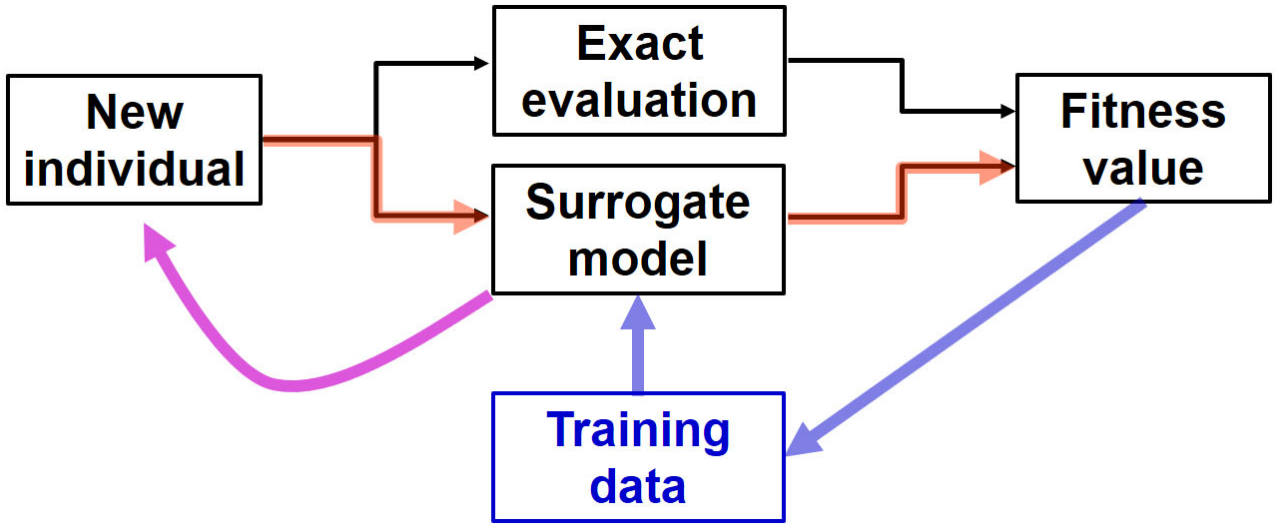
Semi-Supervised Learning

Through the execution of a surrogate-based algorithm, we have a small number of exactly evaluated solutions and a large number of solutions evaluated by the surrogate model.

Training Data: $\{ (\boldsymbol{x}_1, y_1), ..., (\boldsymbol{x}_i, y_i), (\boldsymbol{x}_{i+1}, ?), ..., (\boldsymbol{x}_{i+j}, ?) \}$ where i is small (e.g., 200) and j is large (e.g., 200,000).

Q. How can we utilize $\{(\boldsymbol{x}_{i+1}, ?), ..., (\boldsymbol{x}_{i+j}, ?)\}$ in the learning of the surrogate model?

Other Related Topics to Surrogate Model



Active Learning

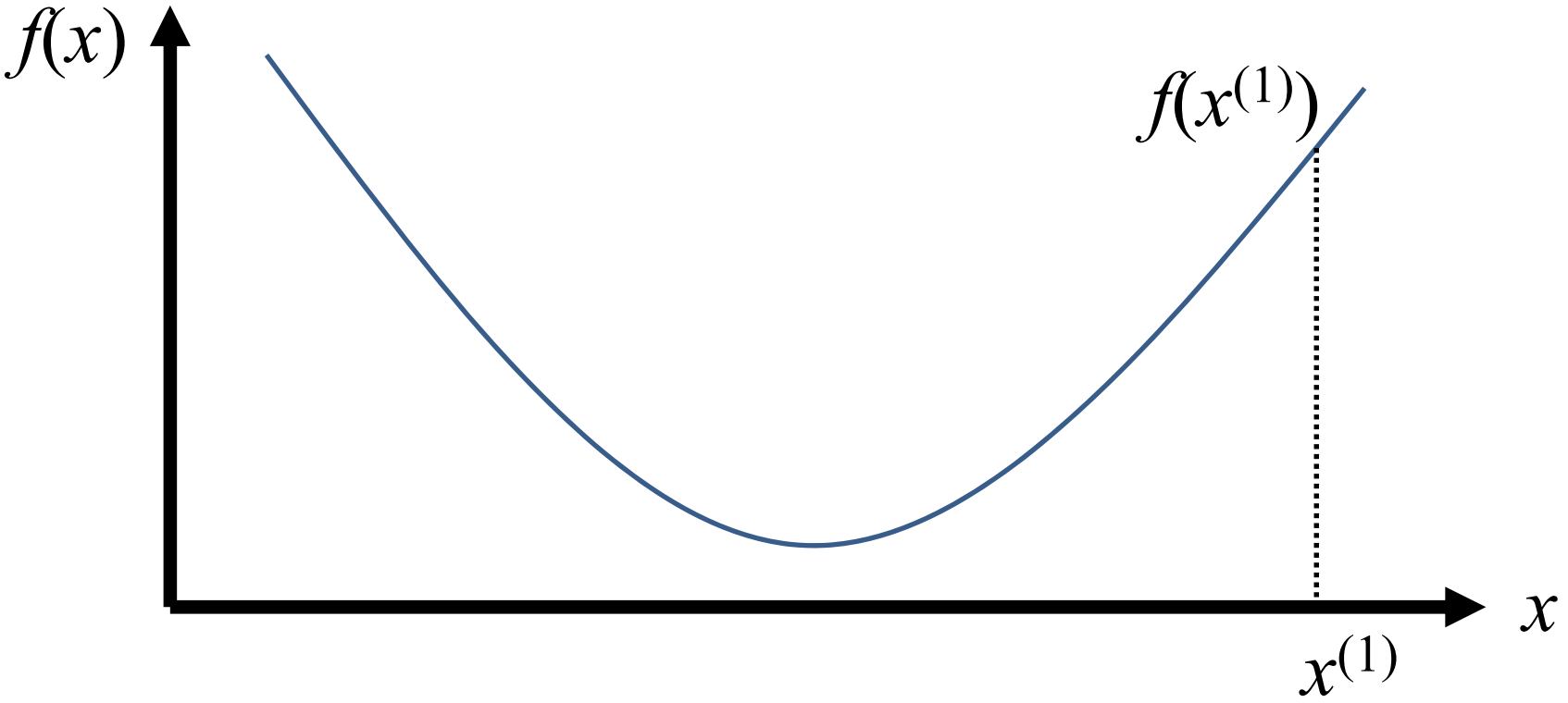
Training Data: $\{ (\boldsymbol{x}_1, ?), (\boldsymbol{x}_2, ?), ..., (\boldsymbol{x}_N, ?) \}$. The problem is to choose inputs for which the corresponding outputs should be calculated. (For a classification task, the problem is to choose patterns for which the corresponding classes should be assigned).

Optimization Methods

1. Introduction.
2. Greedy algorithms for combinatorial optimization.
3. LS and neighborhood structures for combinatorial optimization.
4. Variable neighborhood search, neighborhood descent, SA, TS, EC.
5. Branch and bound algorithms, and subset selection algorithms.
6. Linear programming problem formulations and applications.
7. Linear programming algorithms.
8. Integer linear programming algorithms.
9. Unconstrained nonlinear optimization and gradient descent.
10. Newton's methods and Levenberg-Marquardt modification.
11. Quasi-Newton methods and conjugate direction methods.
12. Nonlinear optimization with equality constraints.
13. Nonlinear optimization with inequality constraints.
14. Problem formulation and concepts in multi-objective optimization.
15. Search for single final solution in multi-objective optimization.
16: Search for multiple solutions in multi-objective optimization.

One Dimensional Search Methods (Line Search)

$$x^{(1)} \Rightarrow x^{(2)} \Rightarrow x^{(3)} \Rightarrow x^{(4)} \Rightarrow x^{(5)} \Rightarrow \ldots$$



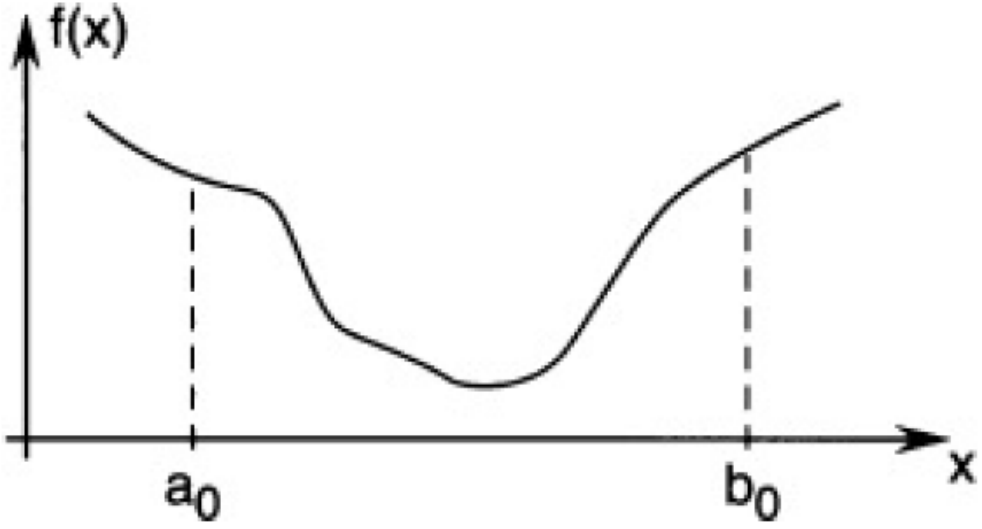
1. Golden Section Search

Problem: Minimization of $f(x)$ subject to $a_0 \le x \le b_0$

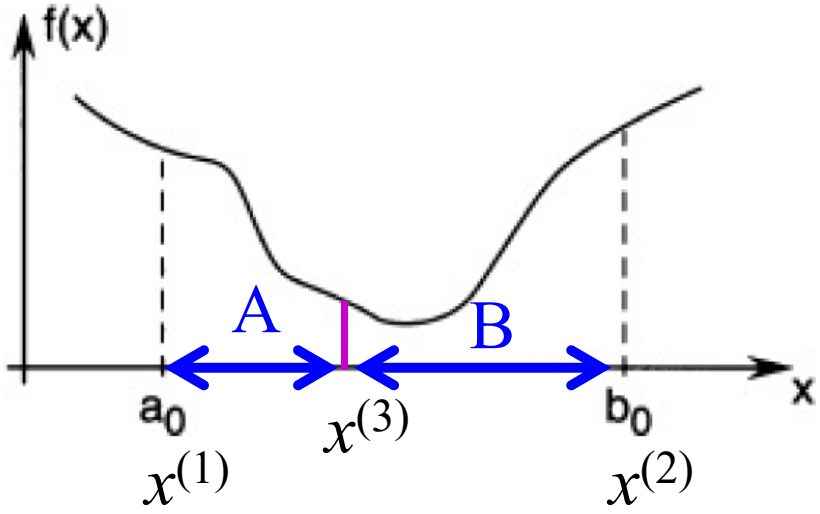
Assumption: $f(x)$ is unimodal ($f(x)$ has a single local minimum)
We do not know the mathematical form of $f(x)$.

Algorithm (Your Task): To determine $x^{(1)}$, $x^{(2)}$, $x^{(3)}$, ...

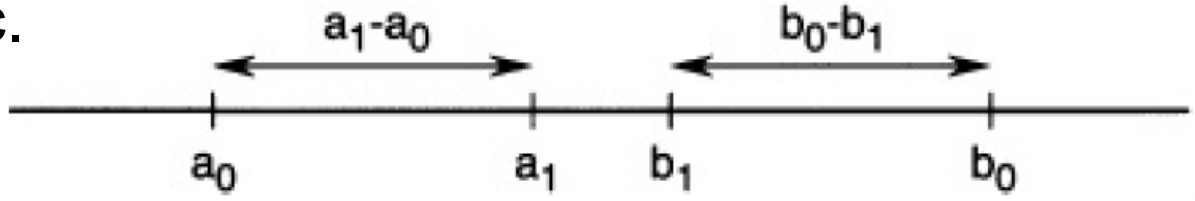
We choose these points in such a way that an approximation to the minimizer of *f* may be achieved in as few evaluations as possible. Our goal is to narrow the range progressively until the minimizer is "boxed in" with sufficient accuracy.



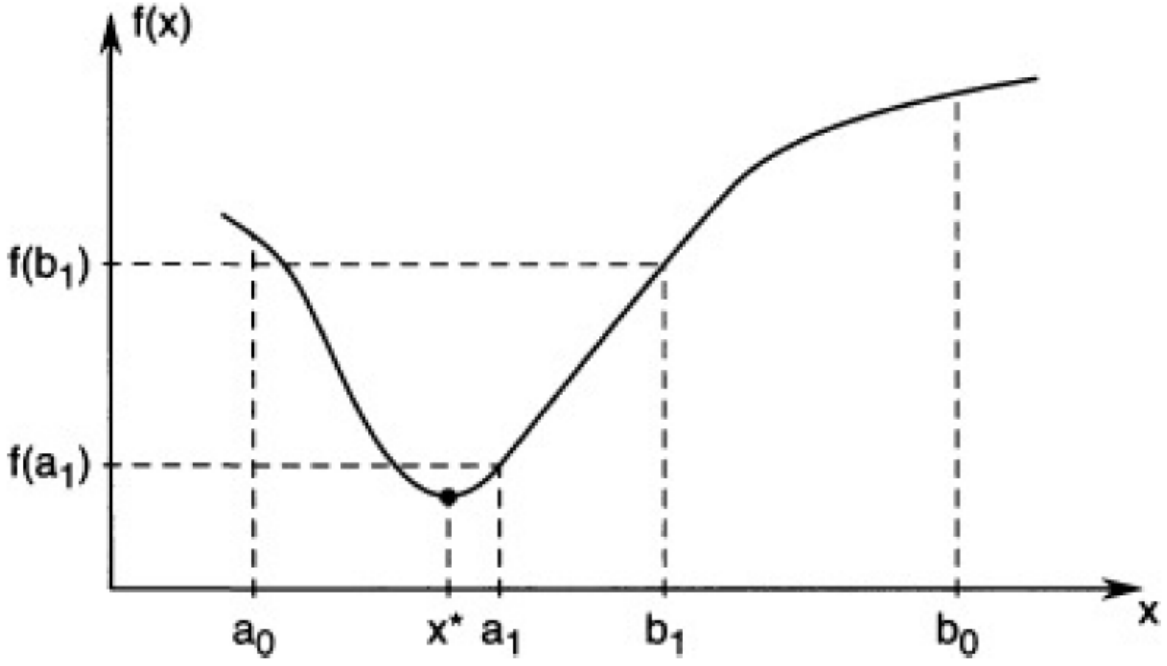
If we evaluate *f* at only one intermediate point of the interval, we cannot narrow the range within which we know the minimizer is located (A or B).

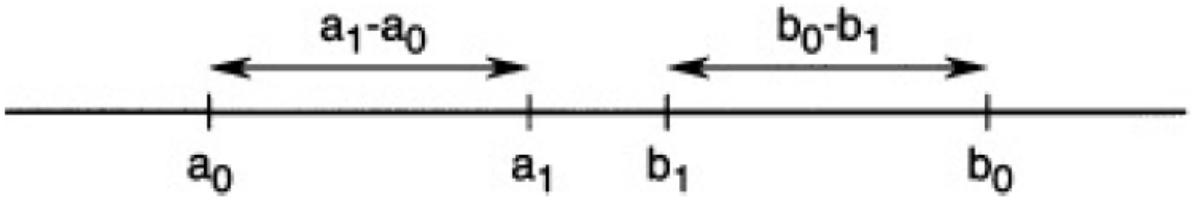


We have to evaluate *f* at two intermediate points. We choose the intermediate points in such a way that the reduction in the range is symmetric.



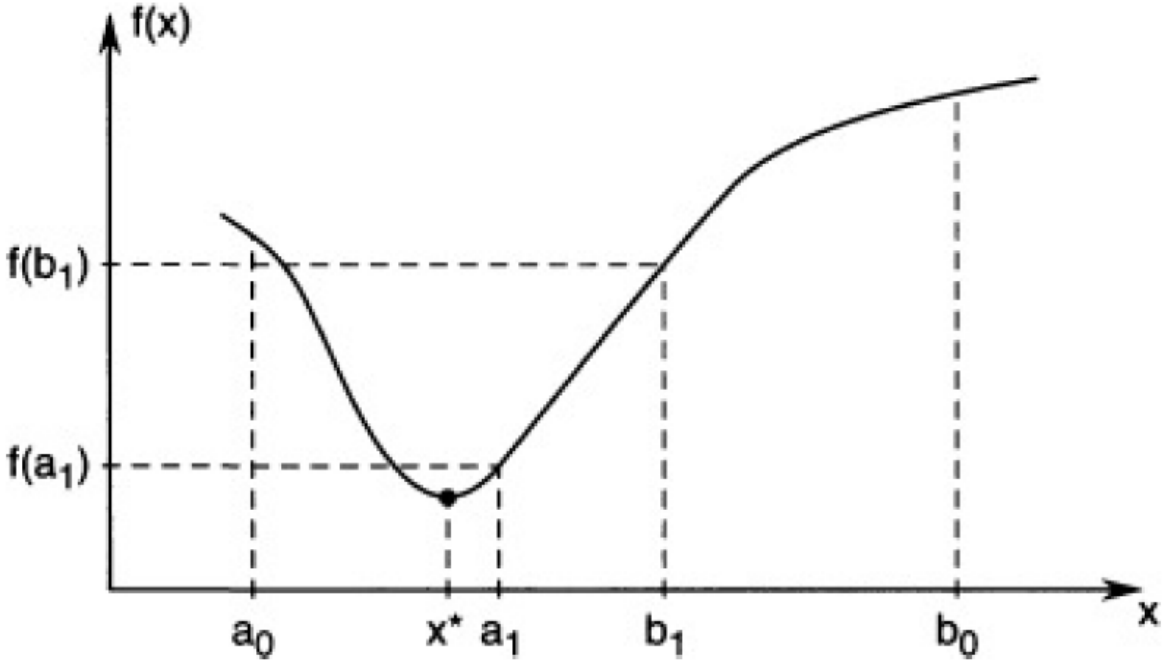
$$a_1 - a_0 = b_0 - b_1 = \rho(b_0 - a_0), \quad \rho < \frac{1}{2}.$$

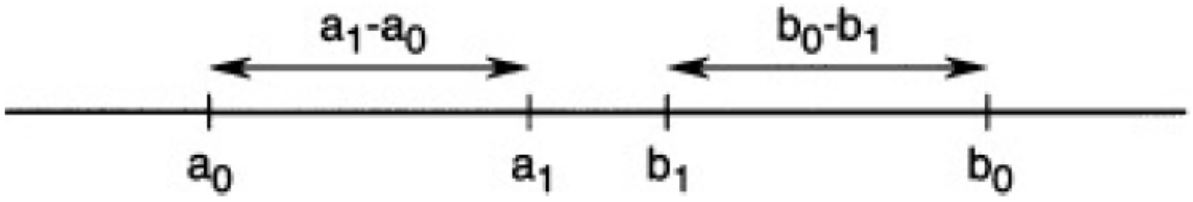




$$a_1 - a_0 = b_0 - b_1 = \rho(b_0 - a_0), \quad \rho < \frac{1}{2}.$$

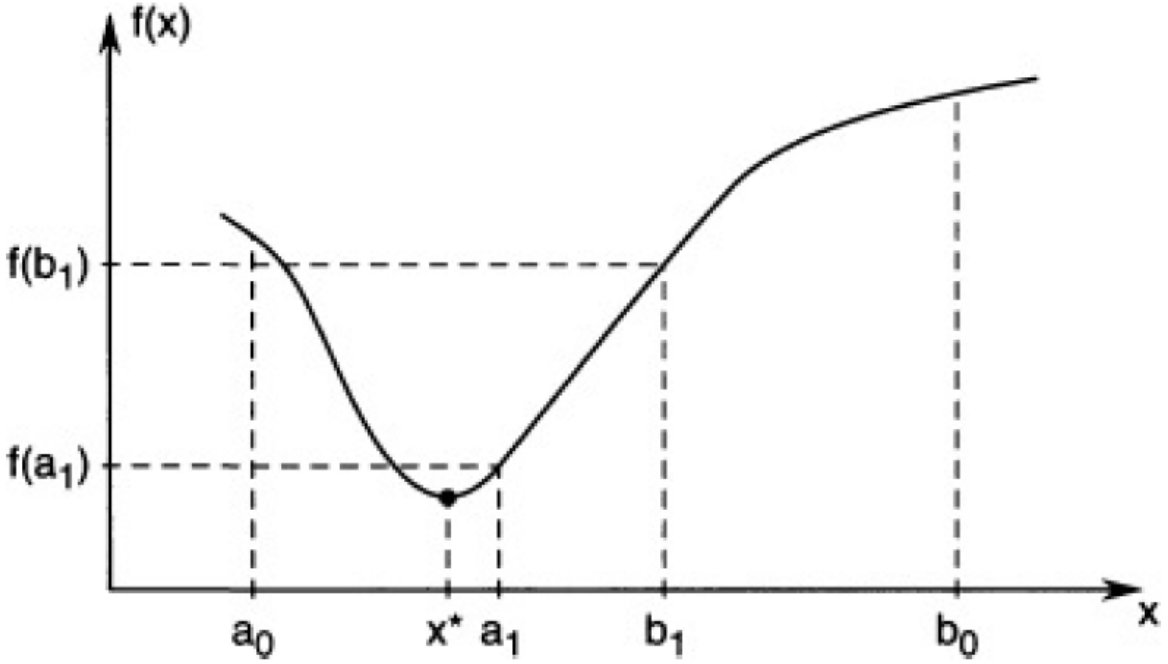
We can decrease the range depending on $f(a_1)$ and $f(b_1)$. For example, if $f(a_1) < f(b_1)$, we know that x^* is in $[a_0, b_1]$

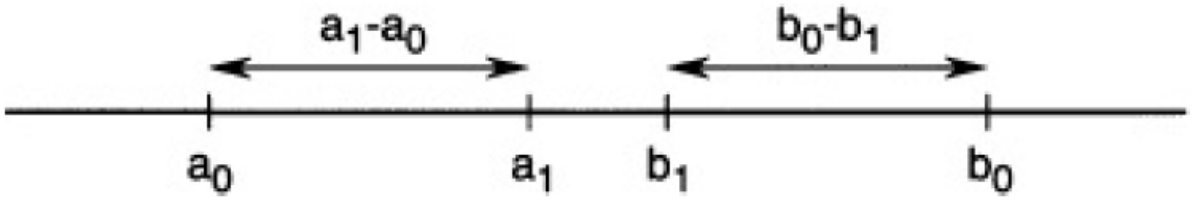




$$a_1 - a_0 = b_0 - b_1 = \rho(b_0 - a_0), \quad \rho < \frac{1}{2}.$$

We can decrease the range depending on $f(a_1)$ and $f(b_1)$. For example, if $f(a_1) < f(b_1)$, we know that x^* is in $[a_0, b_1]$

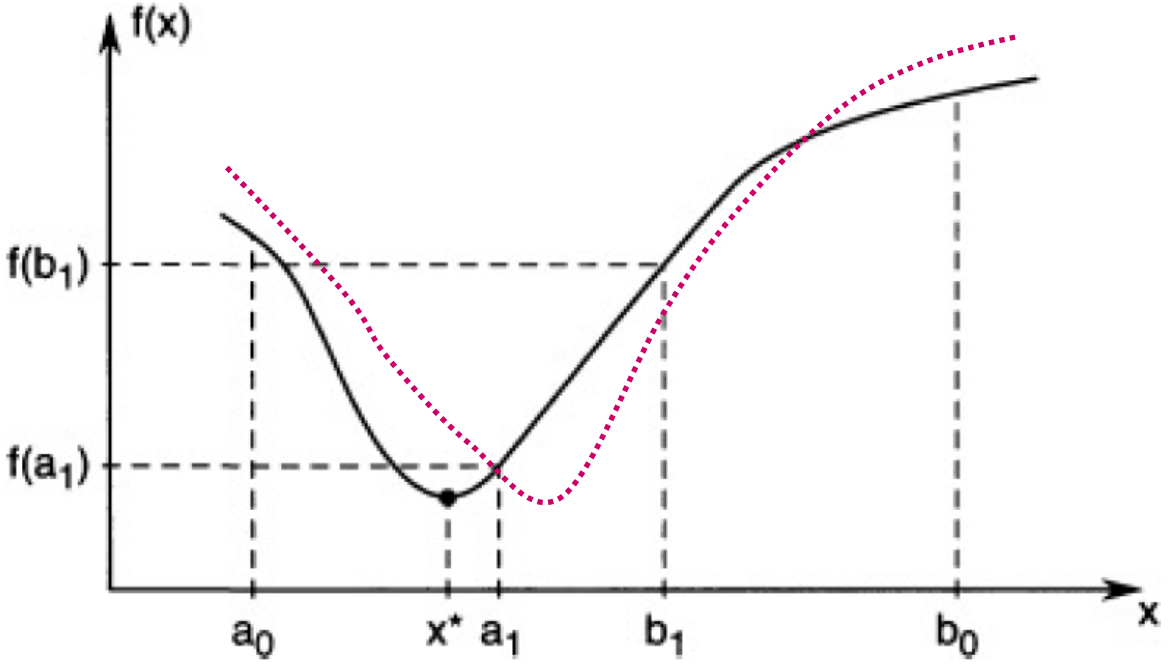




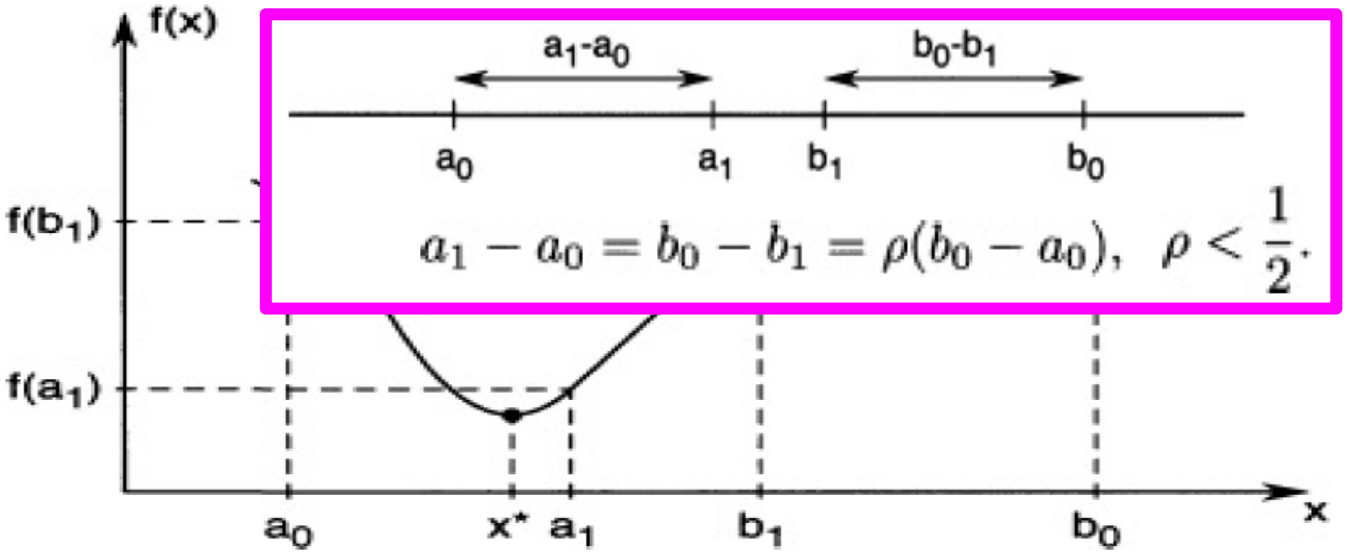
$$a_1 - a_0 = b_0 - b_1 = \rho(b_0 - a_0), \quad \rho < \frac{1}{2}.$$

We can decrease the range depending on $f(a_1)$ and $f(b_1)$.
For example, if $f(a_1) < f(b_1)$, we know that x^* is in $[a_0, b_1]$

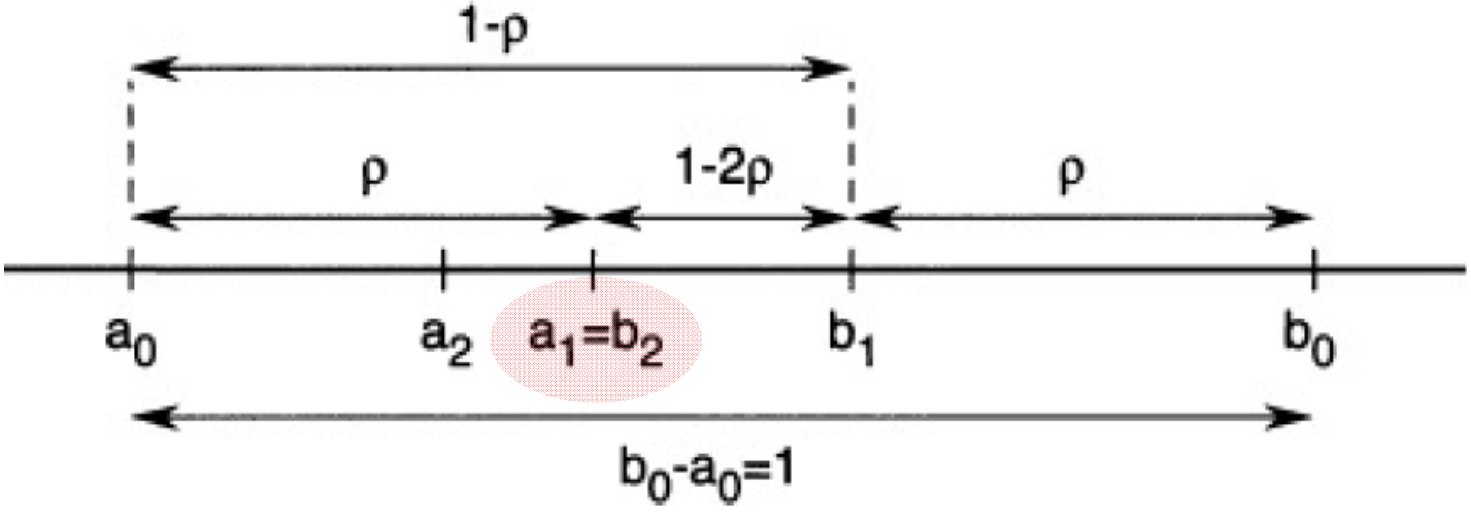
We cannot say that x^* is in $[a_0, a_1]$



Next Step:

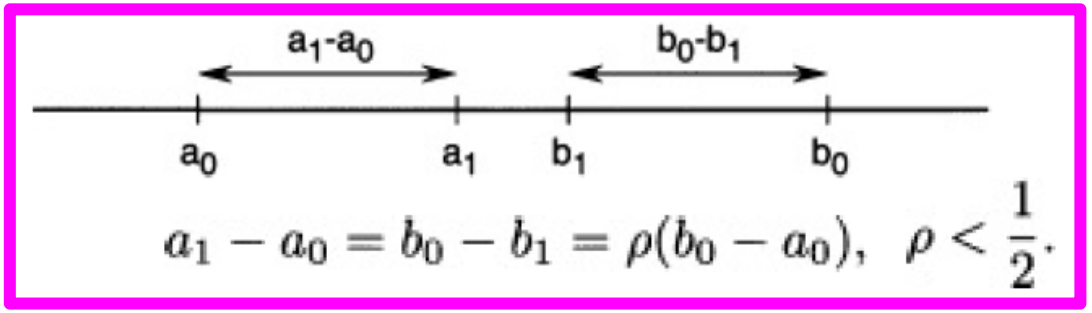


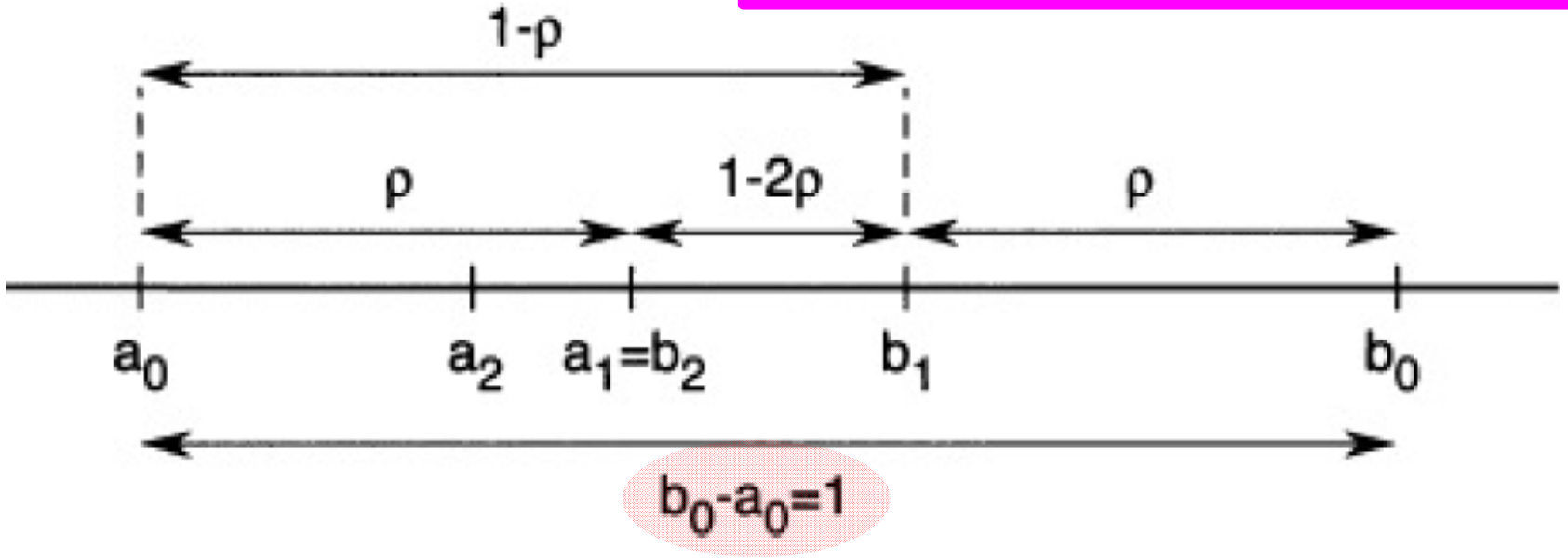
Because a_1 is already in the uncertainty interval and $f(a_1)$ is already known, we can make a_1 coincide with b_2. Thus, only one new evaluation of f at a_2 would be necessary.



For simplicity:

$b_0 - a_0 = 1$





If b_2 is specified in the same manner as in the first step using ρ from the new interval $[a_0, b_1]$, $b_1 - b_2 = \rho\,(b_1 - a_0) = \rho\,(1 - \rho)$. For $b_2 = a_1$, we have the equation $\rho\,(1 - \rho) = 1 - 2\rho$. That is, $\rho^2 - 3\rho + 1 = 0$. $\boldsymbol{a_2 : a_2 = a_0 + \rho\,(b_1 - a_0)}$ as in the first step.

$$\rho^2 - 3\rho + 1 = 0$$

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$
when $ax^2+bx+c=0$

The solutions are

$$\rho_1 = \frac{3+\sqrt{5}}{2}, \qquad \rho_2 = \frac{3-\sqrt{5}}{2}.$$

Because we require that $\rho < \frac{1}{2}$, we take $\rho = \frac{3-\sqrt{5}}{2} \approx 0.382.$

Observe that

$$1 - \rho = \frac{\sqrt{5}-1}{2}$$

and

$$\frac{\rho}{1-\rho} = \frac{3-\sqrt{5}}{\sqrt{5}-1} = \frac{\sqrt{5}-1}{2} = \frac{1-\rho}{1},$$

that is,

$$\frac{\rho}{1-\rho} = \frac{1-\rho}{1}.$$

Thus, dividing a range in the ratio of ρ to $1 - \rho$ has the effect that the ratio of the shorter segment to the longer equals the ratio of the longer to the sum of the two. This rule was referred to by ancient Greek geometers as the *golden section*.

Using the golden section rule means that at every stage of the uncertainty range reduction (except the first), the objective function f need only be evaluated at one new point. The uncertainty range is reduced by the ratio $1 - \rho \approx 0.61803$ at every stage. Hence, N steps of reduction using the golden section method reduces the range by the factor $(1-\rho)^N \approx (0.61803)^N$.

Simple Exercise

Example 7.1 Suppose that we wish to use the golden section search method to find the value of x that minimizes $f(x) = x^4 - 14x^3 + 60x^2 - 70x$

in the interval [0,2] (this function comes from an example in [21]). We wish to locate this value of x to within a range of 0.3.

Your Answer: x is in $[a_?, b_?] = [?.????, ?.????]$.

After N stages the range [0,2] is reduced by $(0.61803)^N$. So, we choose N so that $(0.61803)^N \leq 0.3/2$.

Four stages of reduction will do; that is, $N = 4$.

Iteration 1. We evaluate f at two intermediate points a_1 and b_1. We have

$$a_1 = a_0 + \rho(b_0 - a_0) = 0.7639,$$
$$b_1 = a_0 + (1 - \rho)(b_0 - a_0) = 1.236,$$
$$f(a_1) = -24.36,$$
$$f(b_1) = -18.96.$$

where $\rho = (3 - \sqrt{5})/2$. We compute

Thus, $f(a_1) < f(b_1)$, so the uncertainty interval is reduced to $[a_0, b_1] = [0, 1.236]$.

• • •

Iteration 4. We set $b_4 = \mathbf{a}_3$ and

$$a_4 = a_2 + \rho(b_3 - a_2) = 0.6525.$$

We have

$$f(a_4) = -23.84,$$
$$f(b_4) = f(a_3) = -24.36.$$

Hence, $f(a_4) > f(b_4)$. Thus, the value of x that minimizes f is located in the interval $[a_4, b_3] = [0.6525, 0.9443]$.

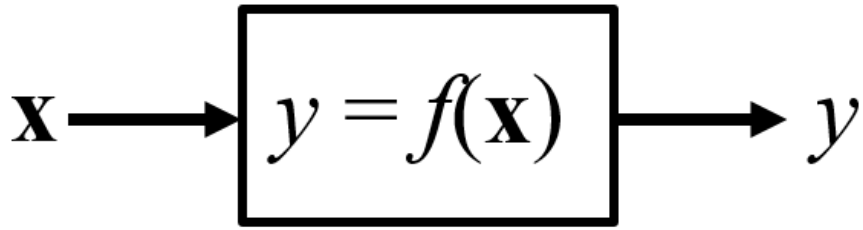
Note that $b_3 - a_4 = 0.292 < 0.3$.

Lab Session Task 1:

Create your own algorithm for black box optimization under the given computation budget of 1,000 solution evaluations. Any of the following types is OK as your report.

(1) Explain your algorithm in detail with no example.
(2) Explain your algorithm using one or more simple examples.
(3) (2) & Experimental results on some non-linear test functions.
(4) (3) & Comparison of your algorithm with basic meta-heuristic algorithms such as simulated annealing and evolutionary algorithms

Black Box Optimization



x: an n-dimensional input vector (real number vector)
y: a real number
$f(\mathbf{x})$: Unknown function