Evolutionary Multi-Objective Optimization

Ning Xiong
Mälardalen University

Agenda

- What are multi-objective optimization problems (MOPs) and some basic concepts
- Why is evolutionary computing (EC) good for MOPs?
- Basics about Genetic Algorithm
- Multi-objective genetic algorithm (i.e. NSGA-II)
- Simple application of NSGA-II

What are MOPs (including basic concepts)

What is MOP

The Multiobjective Optimization Problem (MOP) (also called multicriteria optimization) can be considered as a problem of finding:

a vector of decision variables in the feasible region which ***optimizes a vector function whose elements represent the individual objective functions****.*

These objective functions are actually performance criteria which are usually in conflict with each other. Hence, the term “optimize” means finding a set of good compromise solutions.

Mathematical Definition

The general Multiobjective Optimization Problem (MOP) can be formally defined as:

$$\text{Minimize} \quad F(X) = \left(f_1(X), f_2(X), \cdots f_m(X)\right)$$

$$\text{subject to} \quad X = (x_1, x_2, x_n) \in \Omega$$

where Ω is the feasible region for decision variables
$f_i(X)$ are the individual objective functions

Some Notes on the MOPs

- The individual objective functions f_i conflict with each other
- It is not possible to find a solution which is optimal with respect to all objectives
- In MOPs our goal is to find multiple good compromise (or “trade-offs”) solutions rather than a single solution. The decision maker can then make final choice from the trade-off solutions based on her/his preference.

Basic Concepts

- An objective vector **u**=(u_1, …u_n) is said to **dominate** another **objective vector v**=(v_1, …v_n) if and only if

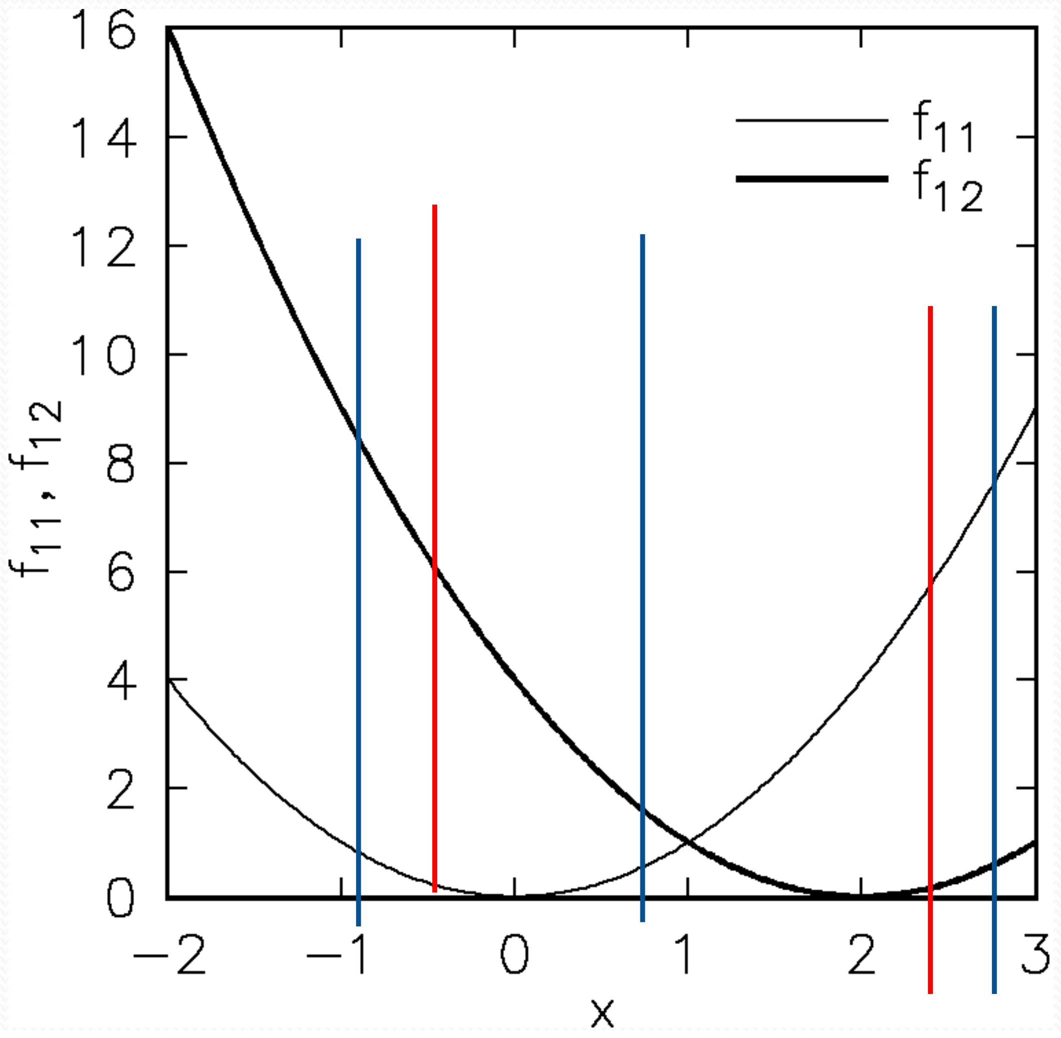
$$\forall i \quad u_i \leq v_i \qquad and \qquad \exists j \quad u_j < v_j$$

- A solution $\mathbf{x_u}$ is said to be nondominated if and only if there is no solution $\mathbf{x_v}$ such that objective vector v=F($\mathbf{x_v}$)=(v_1, …v_n) dominates u=F($\mathbf{x_u}$)=(u_1, …u_n)

- A nondominated solution is called Pareto-optimal solution

- The Pareto set consists of all Pareto-optimal (nondominated) solutions

A Simple Example of MOP

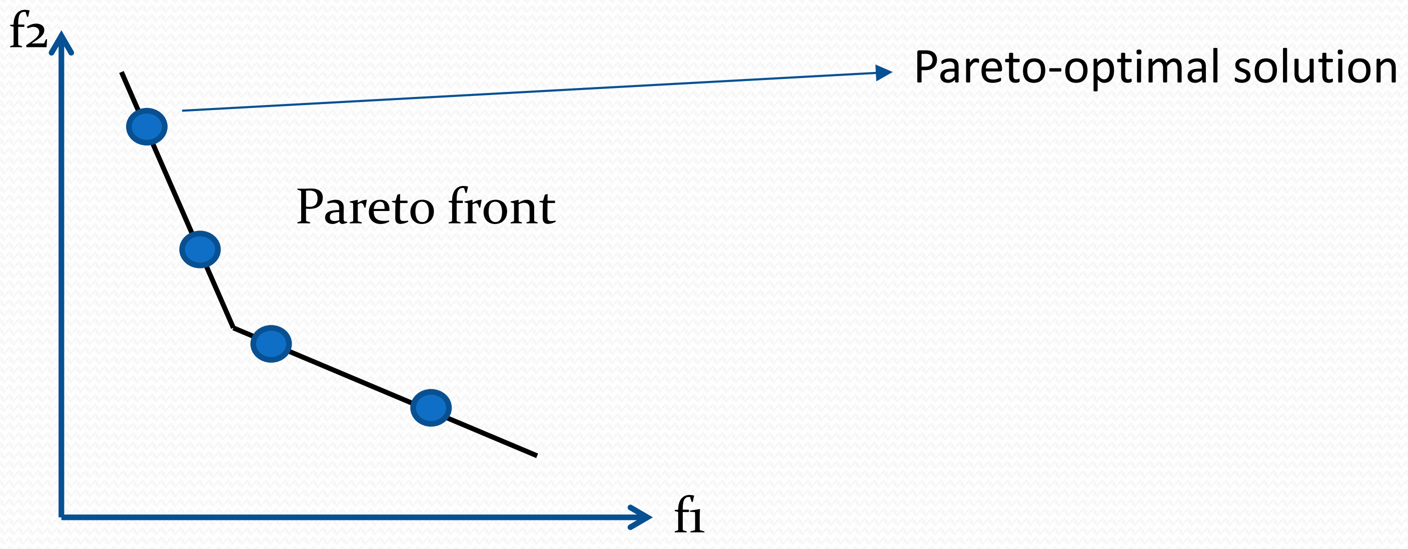


Minimize: f_{11} and f_{12}

Nondominated solution: [0, 2]

Pareto Front

Pareto front is the set of objective vectors of all Pareto optimal solutions



Goal: finding a managable number of **evenly distributed** points on the Pareto front

Why is evolutionary computing good for MOPs?

Classical Method

Multiple objective functions are combined into one overall objective function, then a single solution is found to minimze this overall objective function:

$$\min \sum_{i=1}^{n} w_i f_i(x)$$

where $w_i > 0$ are the weighting coefficients representing the relative importance of the *n* objective functions. Weights are usually normalized such that

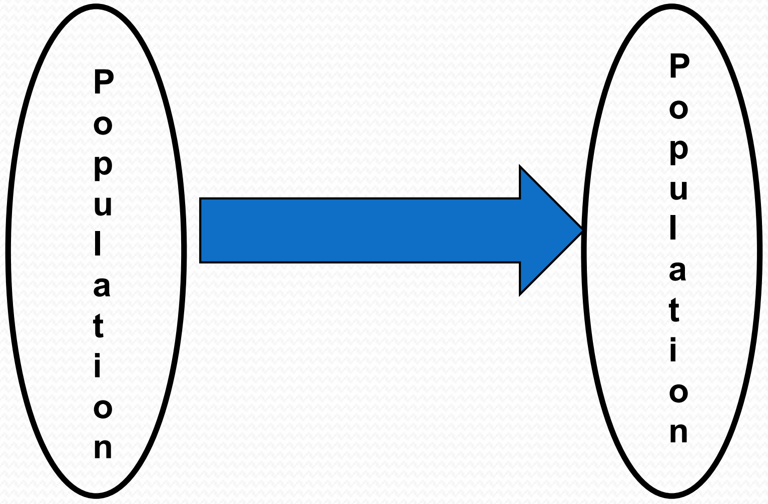
$$\sum_{i=1}^{n} w_i = 1$$

Drawbacks of Classical Method

- The solution found largely depends the weight assigned.
- It is difficult for decision maker to specify exact weight values to get most preferred result.
- Obtimization has to be done in a number of times using different sets weights in order to acquire a set of Pareto-optimal solutions.
- Not all Pareto-optimal solutions can always be found via weighted summation of multiple objectives.

Merit of Evolutionary Computing

Evolutionary computing is population based optimization methods, they simultaneously deal with a set of possible solutions. They have the potential ability to find a group of non-dominated solutions with a single run of the algorithm when applied to MOPs.

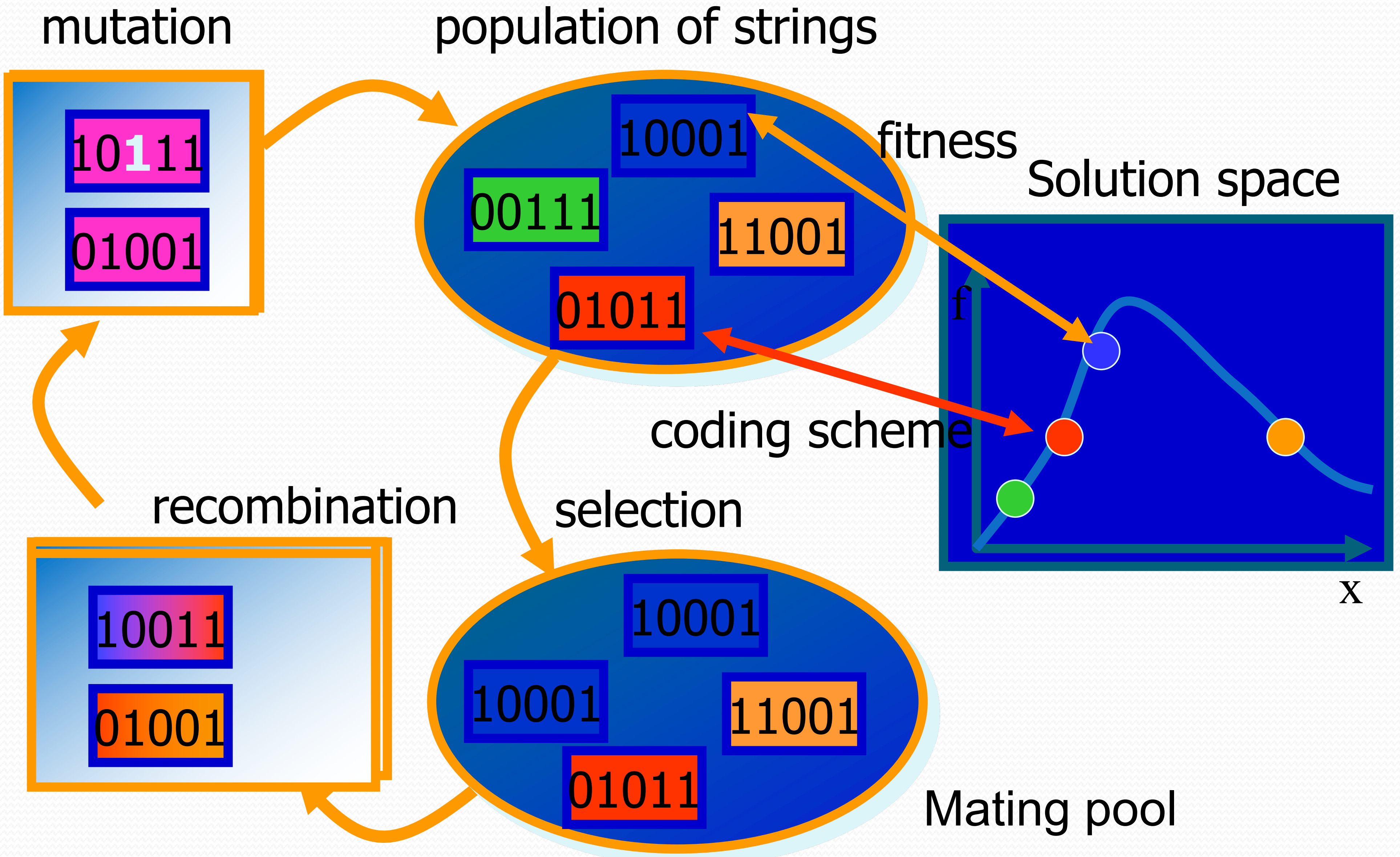


Evolutionary Computation (EC)

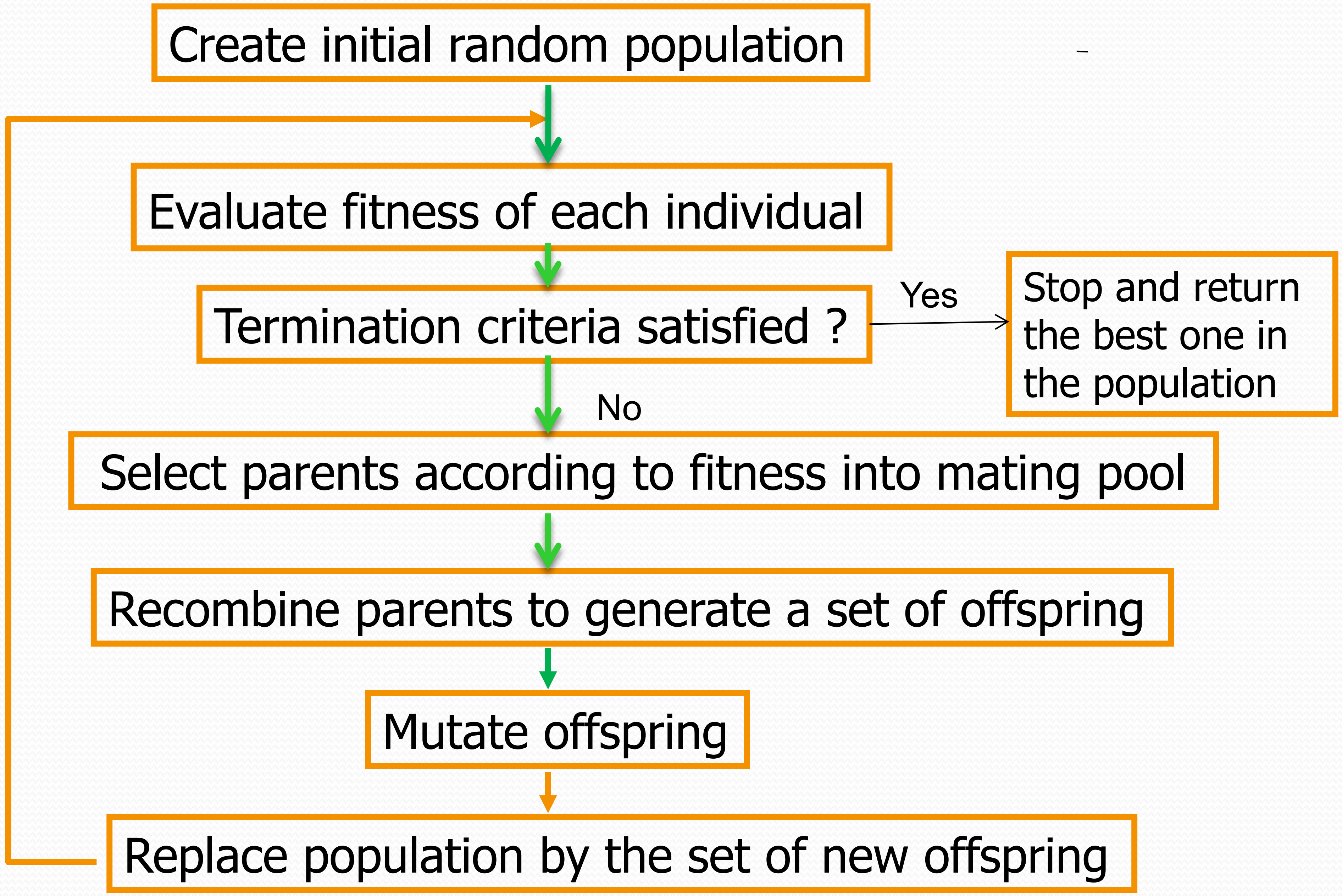
- EC simulates natural evolution in a computer program to improve system performance automatically
- Like natural selection, EC aims to facilitate stochastic search in problem space
- EC conducts randomized, parallel beam search. It becomes more popular with advancement of computer hardware..
- Genetic algorithms (GAs) are the most applied technique in EC.

Basics about Genetic Algorithm (GA)

Simple Genetic Algorithm



Flow Chart of Genetic Algorithm



Create initial random population
Evaluate fitness of each individual
Termination criteria satisfied ?
Yes
Stop and return the best one in the population
No
Select parents according to fitness into mating pool
Recombine parents to generate a set of offspring
Mutate offspring
Replace population by the set of new offspring

Selection Probability Based on Fitness

- Darwinian selection: individuals with higher fitness have a better chance of survival

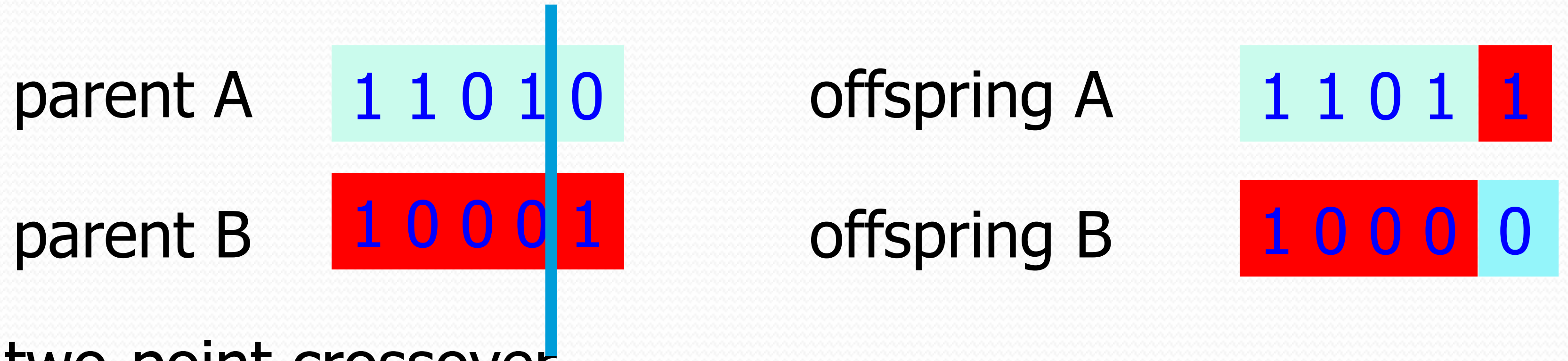
- Fitness proportionate selection:

$$P(S_i) = \frac{Fit(S_i)}{\sum_j Fit(S_j)}$$

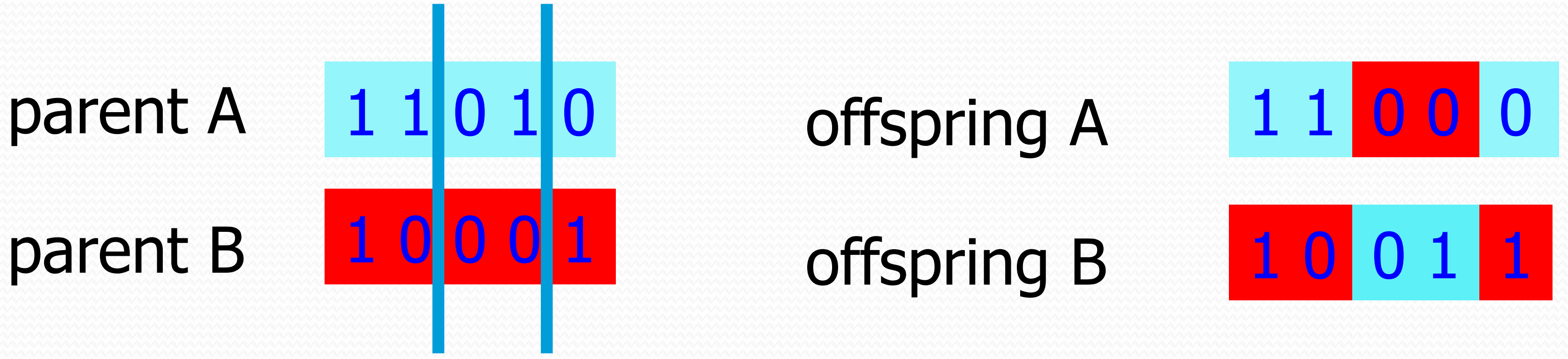
Crossover

- crossover applied to a pair of parent strings with probability $p_c \in [0.6, 1.0]$
- crossover site chosen randomly with uniform probability

- one-point crossover

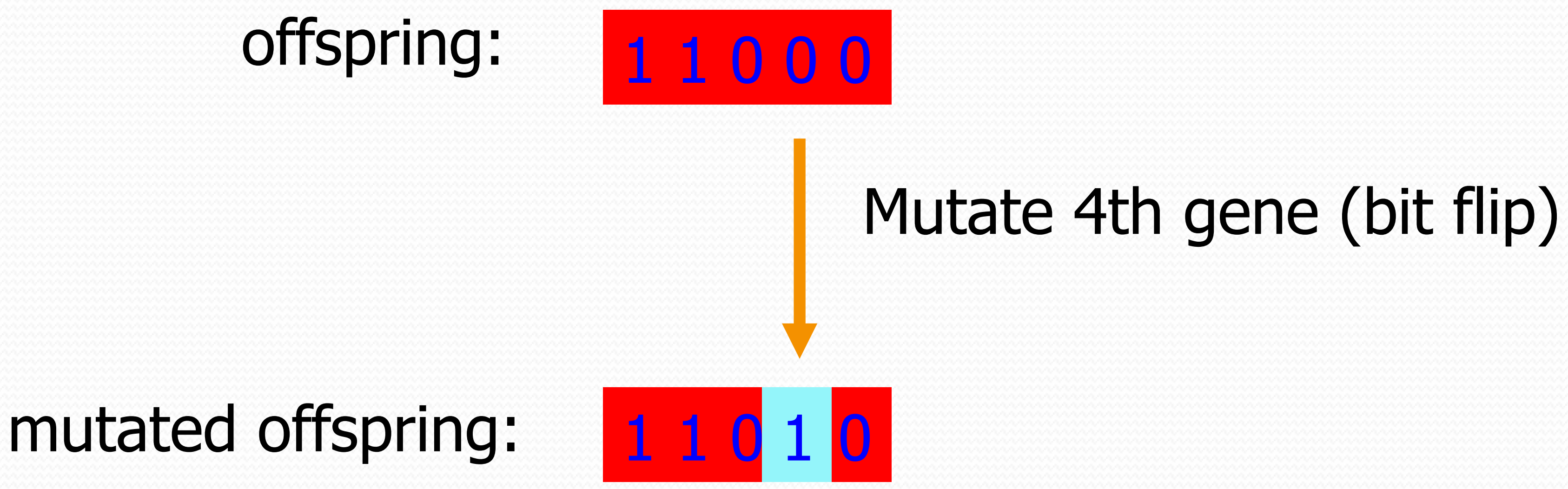


- two-point crossover



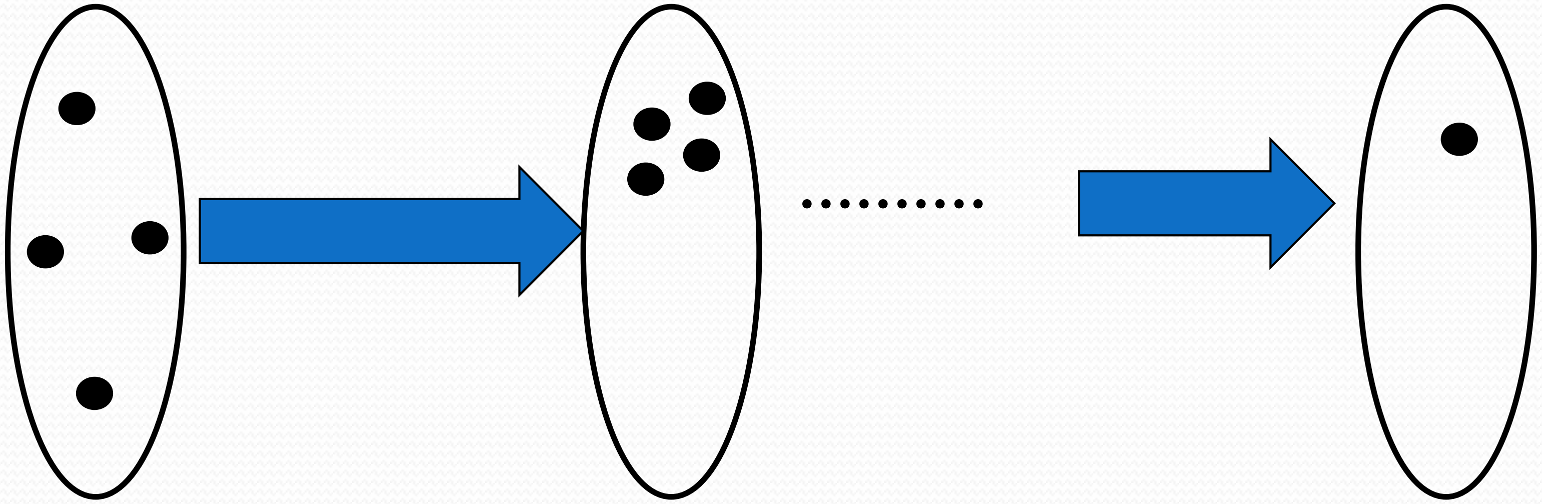
Mutation

- Mutation applied to individual bits with probability $p_m \in [0.001..0.1]$
- Role of mutation is to maintain genetic diversity



Convergence with Traditional GA

- In traditional GA, individuals in the population are evaluated with a numerical fitness function.
- The individuals with higher fitness have more chance to survive and produce offspring
- Finally the population would converge to a single solution with highest fitness value.

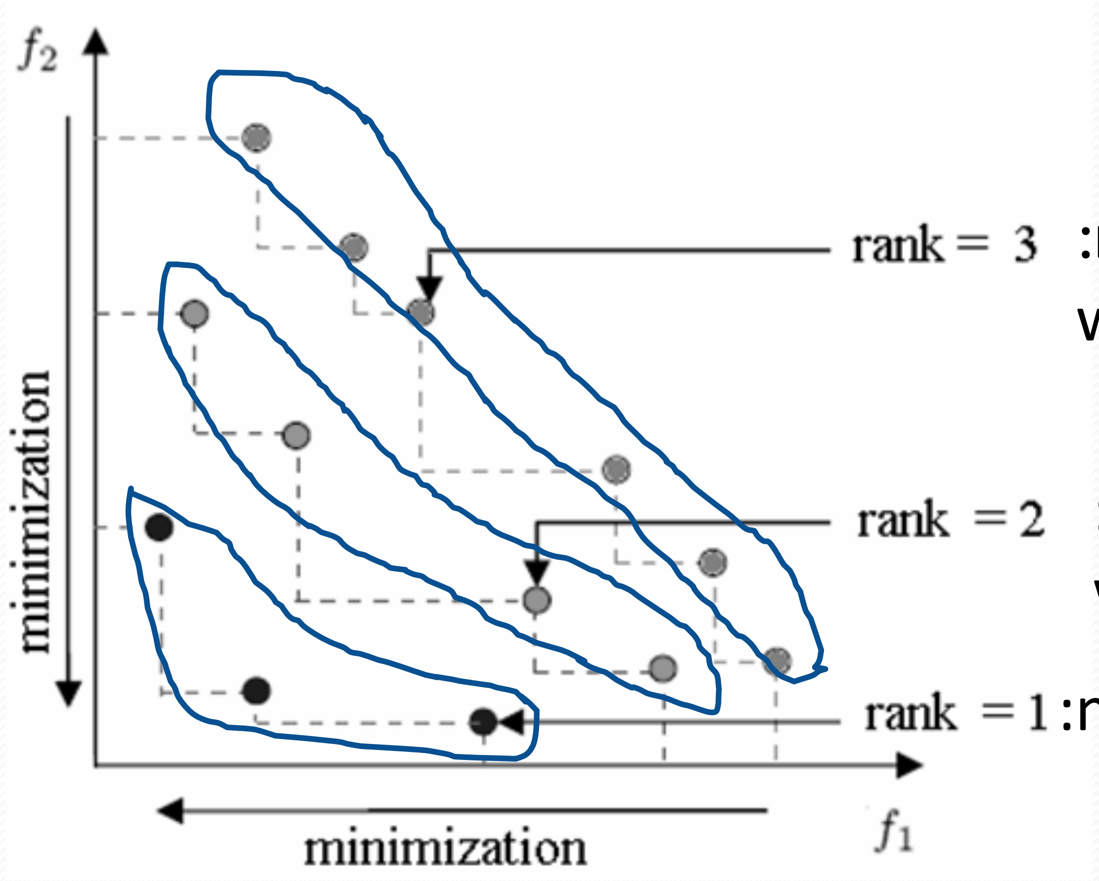


Multi-objective genetic algorithm

When Applying GA for MOPs

- We have a set of objective functions for performance evaluation
- Our goal is to find a group of nondominated solutions,
- An intuitive idea is to base individual evaluation and selection on nondominance relation among solutions

Basic Idea: Classifying Solutions Based on Nondominance



:nondominated among those with rank than two

:nondominated among those with rank more than one

:nondominated in the population

Direct sorting: repeated pairwise comparison of solutions, highly inefficient

Fast-Nondominated Sorting

- Do pairwise comparisons on the population once and get the following results:

  (i) The nondominated solutions: at rank 1 or front 1

  (ii) n_p for every individual p: the number of solutions dominating p

  (iii) S_p for every individual p: the set of solutions dominated by p

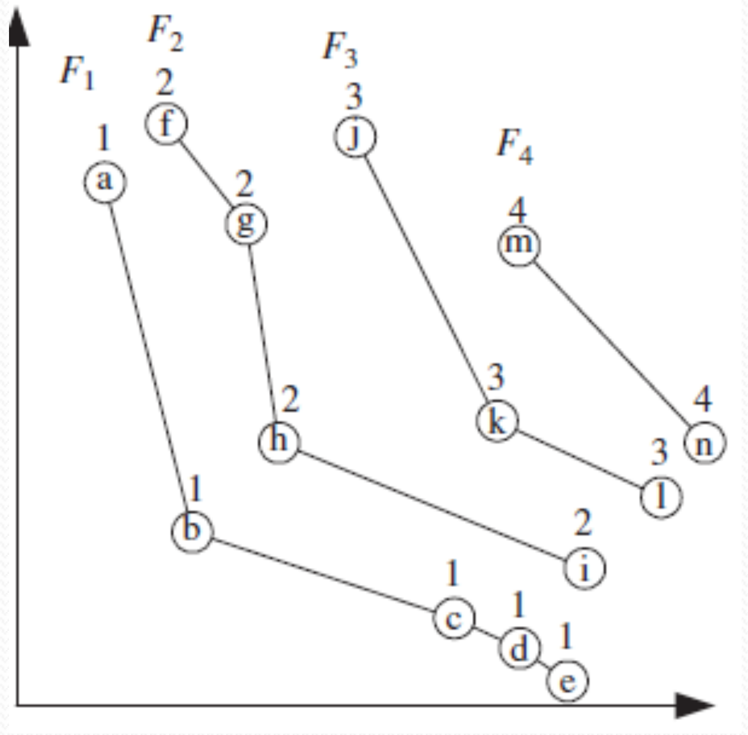
- Sets S_p of individuals at Rank k $\Rightarrow$ Individuals at Rank k+1

Fast Nondominated Sorting Procedure

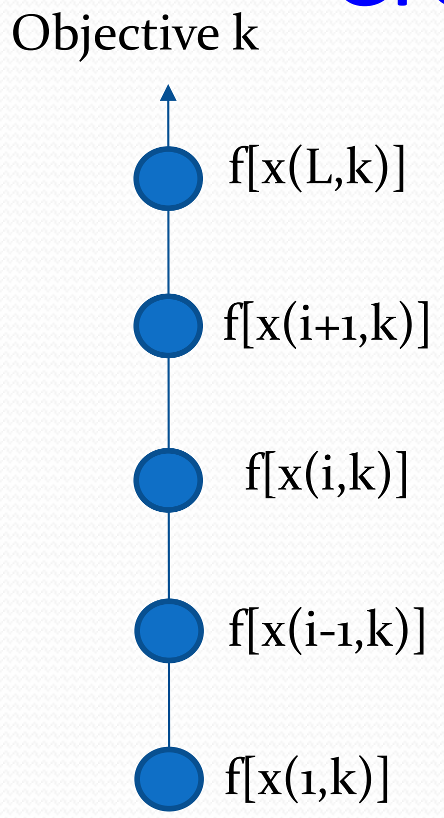
```
fast-non-dominated-sort($P$)
for each $p \in P$
    $S_p = \emptyset$
    $n_p = 0$
    for each $q \in P$
        if $(p \prec q)$ then                  If $p$ dominates $q$
            $S_p = S_p \cup \{q\}$             Add $q$ to the set of solutions dominated by $p$
        else if $(q \prec p)$ then
            $n_p = n_p + 1$                    Increment the domination counter of $p$
    if $n_p = 0$ then                          $p$ belongs to the first front
        $p_{\text{rank}} = 1$
        $\mathcal{F}_1 = \mathcal{F}_1 \cup \{p\}$
$i = 1$                                        Initialize the front counter
while $\mathcal{F}_i \neq \emptyset$
    $Q = \emptyset$                            Used to store the members of the next front
    for each $p \in \mathcal{F}_i$
        for each $q \in S_p$
            $n_q = n_q - 1$
            if $n_q = 0$ then                  $q$ belongs to the next front
                $q_{\text{rank}} = i + 1$
                $Q = Q \cup \{q\}$
    $i = i + 1$
    $\mathcal{F}_i = Q$
```

Solutions on the same rank

- Evaluation criterion: we prefer solutions in the less dense area to favour diversity of solutions in the population
- Next we introduce the crowding distance as a measure of such density



Crowding Distance



$Cd_k[x(L,k)]$=infinite

$$Cd_k\left[x(i,k)\right]=\frac{f_k\left[x(i+1,k)\right]-f_k\left[x(i-1,k)\right]}{f_k^{\max}-f_k^{\min}}$$

$Cd_k[x(1,k)]$=infinite

Step 1: For each objective function k, sort the solutions in the same rank in the ascending order, calculate the crowding distances with respect to objective k

Step 2: The total crowding distance cd(x) of a solution x is the sum of the solution’s crowding distances on all objectives.

Crowded-Comparison Operator

Between two solutions with differing non-domination ranks we prefer the solution with the lower rank.

Otherwise, if both the solutions belong to the same rank, then we prefer the point with larger crowding distance (located in a region with lower density)

Tournament Selection

Randomly chosen two solutions from the population. With predifined probability *p* the stronger one is selected, and with probability (1-*p*) the weaker one is selected.

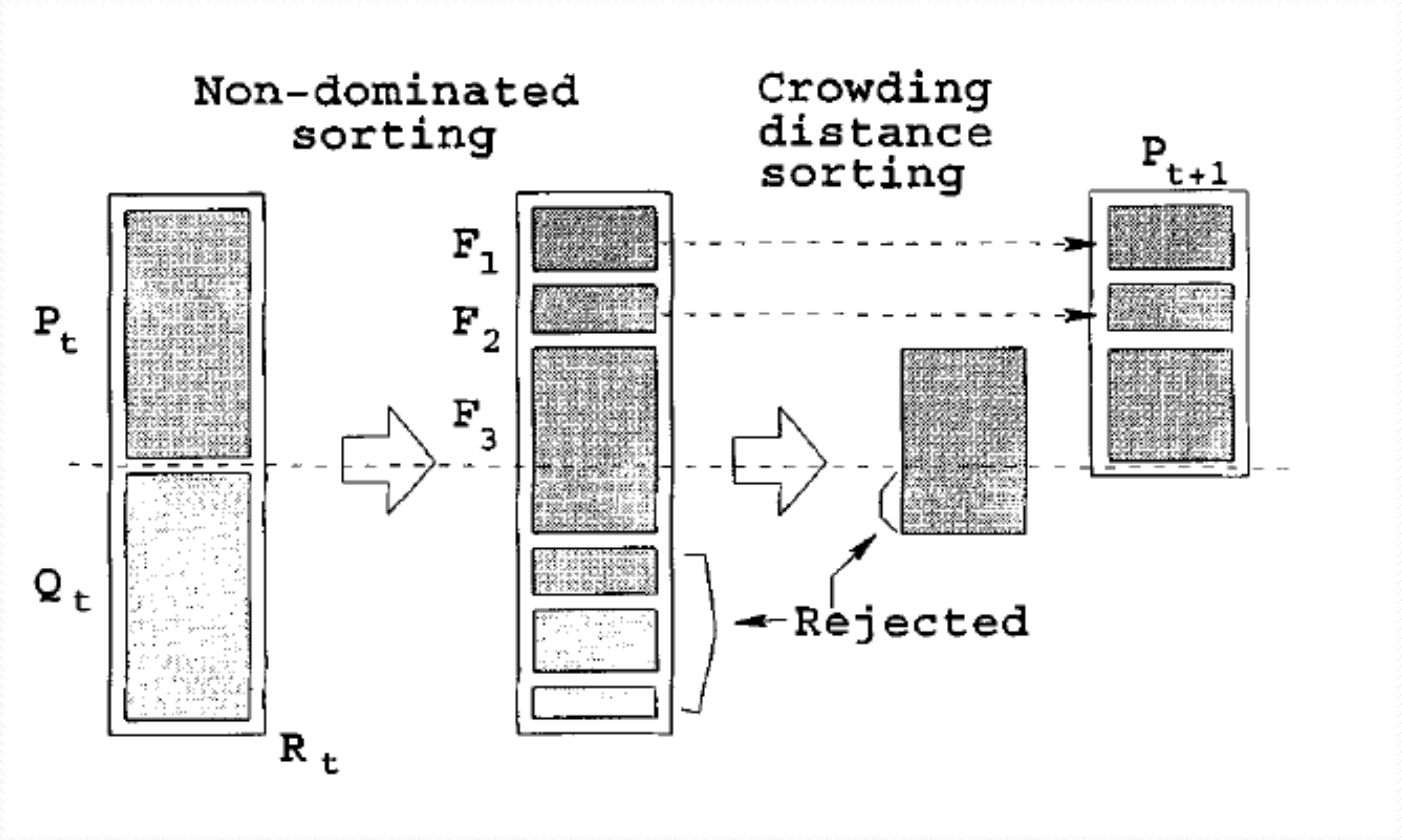
- Tournament selection only needs comparison of two solutions, not fitness values

NSGA-II Algorithm

The NSGA-II algorithm uses tournament scheme combined with rank and crowding distance, to select the parents for mating

A fast and elitist multiobjective genetic algorithm, IEEE Trans. Evolutionary Computation, Vol. 6, 2002, pp. 182-197.

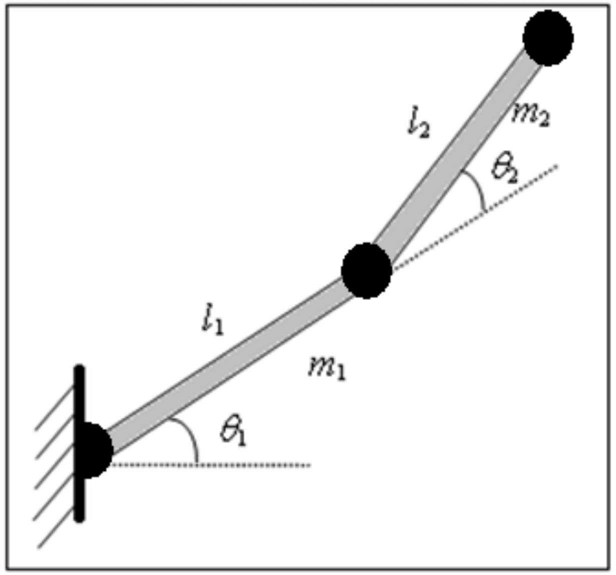
Main Loop of NSGA-II



- The offspring set Q_t is created from the population P_t by using tournament selection

Application of NSGA-II

Optimal PID Controller Using NSGA-II

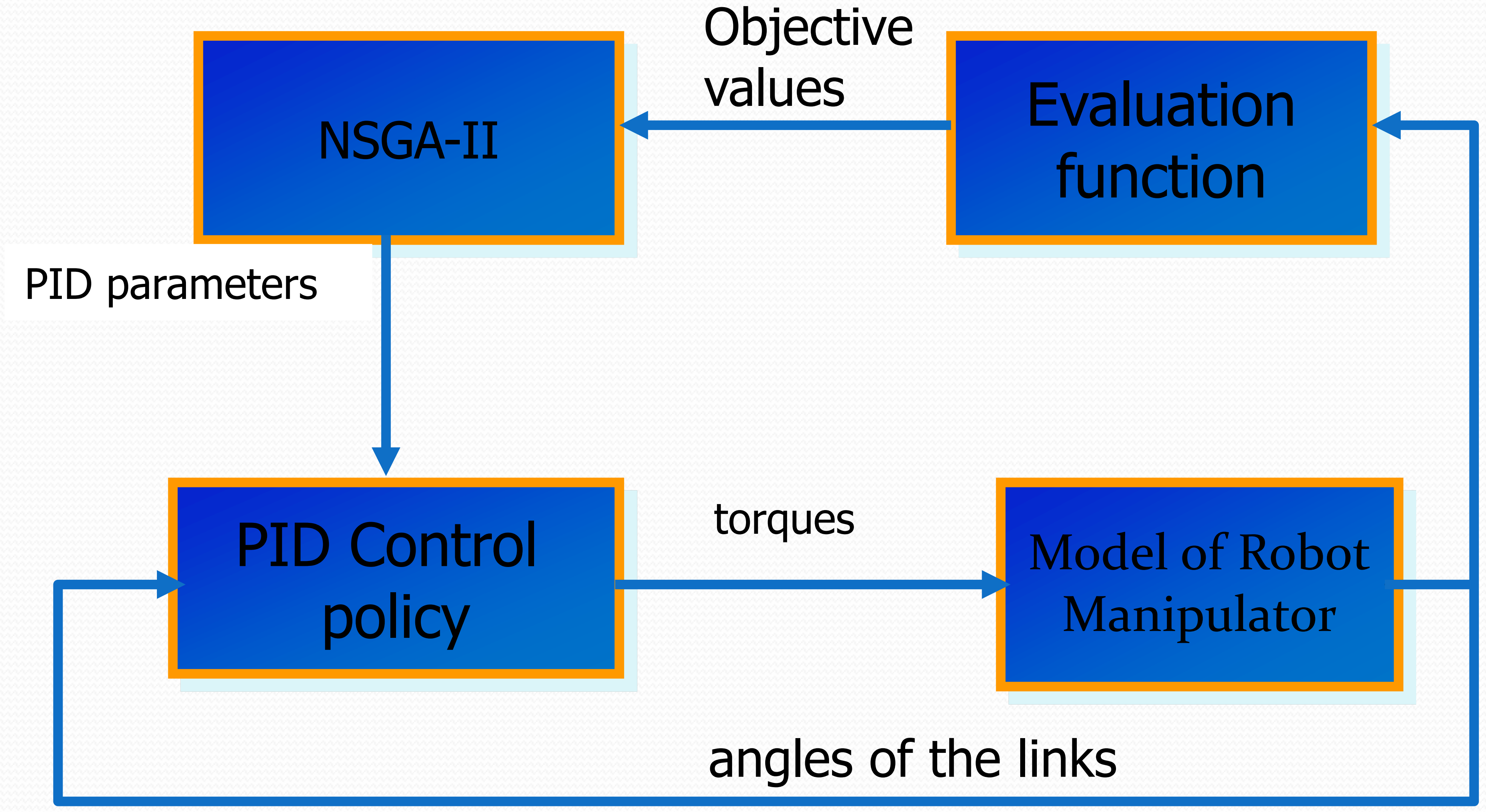


- PID control to decide the two torques as control actions for the robotic manupulator
- Optimization task: find PID parameters (6 parameters) to minimize the following 2 functions

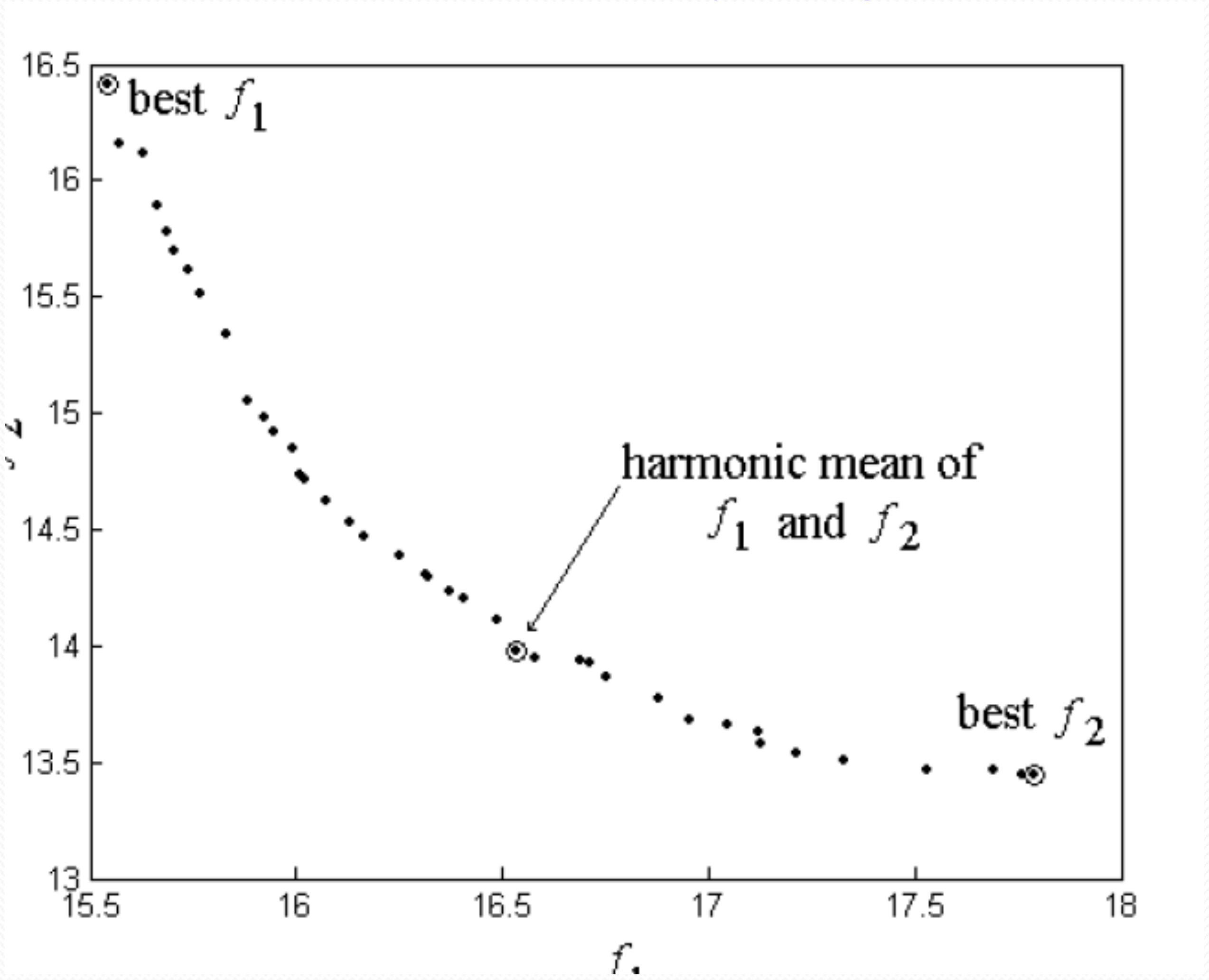
$$f_1 = \sum_{t=1}^{tf} \left|\theta_1(t) - \theta_{d,1}(t)\right| + \sum_{t=1}^{tf} \left|\theta_2(t) - \theta_{d,2}(t)\right|$$

$$f_2 = \sum_{t=1}^{tf} \left|\tau_1(t) - \tau_1(t-1)\right| + \sum_{t=1}^{tf} \left|\tau_2(t) - \tau_2(t-1)\right|$$

Evolutionary Optimal PID Control



Results of Applying NSGA-II



Acquired Pareto-optimal solutions (PID gains) with large diversity

Recommendations for Reading

1. Read and understand the contents of the slides given in the lecture.

2. Read the sections 1, 2, 3.1, 3.4, 4.2 in the paper on the NSGA algorithm, available in blackboard.

3. Read the sections I, II, and III in the paper on the NSGA-II algorithm, available in the blackboard.

4. For students planning to do projects in evolutionary-based optimization, it would be interesting to read the sections of “Simulation Results” on both the NSGA and NSGA-II papers.

5. This lecture is only for your knowledge, not included in written exam