EVOLUTIONARY COMPUTATION AND MULTIOBJECTIVE OPTIMIZATION

Gary G. Yen, *FIEEE*, *FIET*, *FIAPR*

gyen@okstate.edu

***Regents Professor*, Oklahoma State University**

四川大学高端外籍讲座教授

Summer Course at

四川大学计算机学院

Day Four of EIGHT, July 1, 2022

Case Study 4:
Financial Portfolio Optimization

- **Motivation:** Classical Markowitz mean-variance model is widely used for portfolio assets selection and allocation, which aims at simultaneously maximizing the expected return of the portfolio and minimizing portfolio variance. However, introducing various realistic constraints inadvertently leads to a non-convex search space, which has hindered the application of many classic, exact algorithms such as quadratic programming. The increasing size of available assets and complex constraints has made the effectiveness of metaheuristic algorithms deteriorated.

- **Approach:** This study proposes a hybrid bi-objective algorithm combining with the respective advantages of local search algorithm, evolutionary algorithm and QP with a pre-selection strategy. The experimental study demonstrates that the proposed hybrid approach can obtain faster and better convergence compared with eight state-of the-art multi-objective evolutionary algorithms.

$$\text{minimize } f_1 = \sum_{i=1}^{N}\sum_{j=1}^{N} w_i \sigma_{ij} w_j$$

Markowitz's mean-variance model

$$\text{maximize } f_2 = \sum_{i=1}^{N} r_i w_i$$

constraints & dynamism

$$\text{Budget constraint} \sum_{i=1}^{N} w_i = 1$$

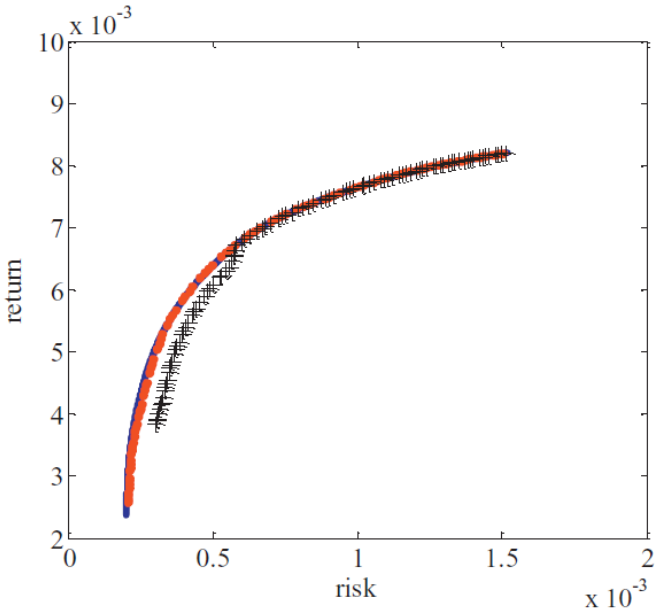
$$\text{Floor} - \text{ceiling constraint} : w_i = 0 \text{ or } a \le w_i \le b, 0 < a \le b \le 1$$

$$\text{Cardinality constraint: } Z_L \le Z \le Z_U, Z = \sum_{i=1}^{N} z_i,$$

$$z_i = 1 \text{ if } w_i \neq 0;\ z_i = 0 \text{ if } w_i = 0$$

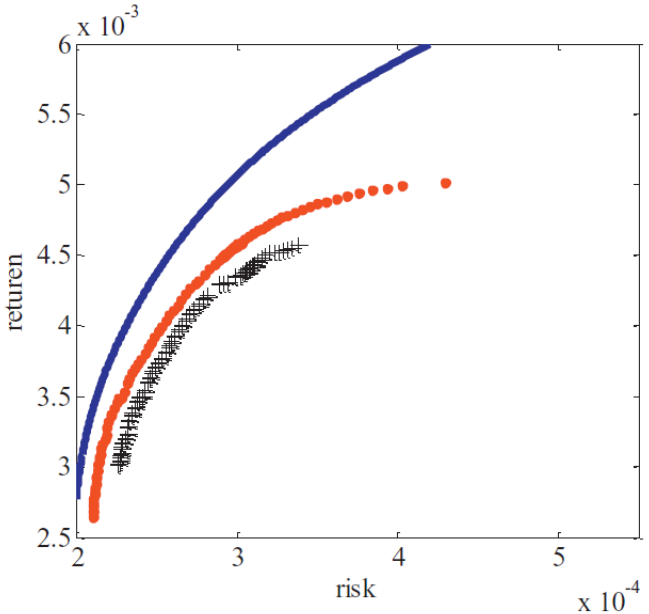
"Hybrid bi-objective portfolio optimization with pre-selection strategy," Qi R. and Yen G.G., ***Information Sciences***, Vol. 417, November 2017, pp. 401-419.

Problem index	Data source	Number of assets
Port 1	Hong Kong Hang Seng	31
Port 2	German DAX 100	85
Port 3	British FTSE 100	89
Port 4	US S & P 100	98
Port 5	Japan Nikkei 225	225



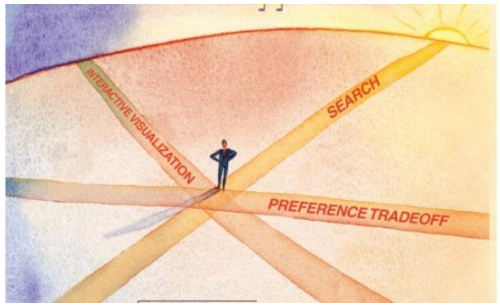
$2 \leq Z \leq 3$

$0 \leq Z \leq 55$



6. MULTIOBJECTIVE OPTIMIZATION PROBLEMS

多目标优化问题



Why MOPs?

- In real life environments we always strive to optimize a number of parameters in any design and these parameters are usually highly correlated. Hence, some tradeoff between the criteria is needed to ensure a satisfactory design. For examples, …

- In ***bridge construction***, a good design is characterized by *low* total mass and *high* stiffness.
- ***Aircraft design*** requires simultaneous optimization of fuel efficiency, payload, and weight.
- A good ***sunroof design*** in a sports car could aim to minimize the noise the driver hears and maximize the ventilation.
- The business ***portfolio management*** attempts to simultaneously minimize the risk and maximize the fiscal return.

Essence in Solving MOPs…

Multi-Objective Optimization Problem

Minimize f_1

Minimize f_2

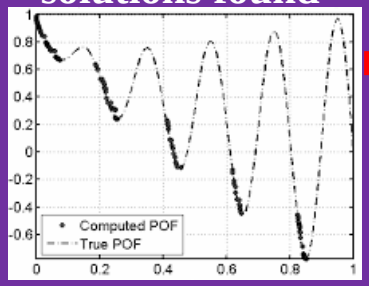
...

Minimize f_M

subject to constraints

Ideal Multi-Objective Optimizer

Multiple trade-off solutions found

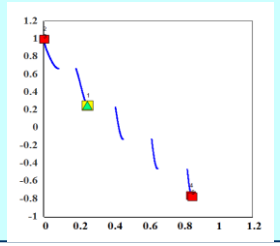


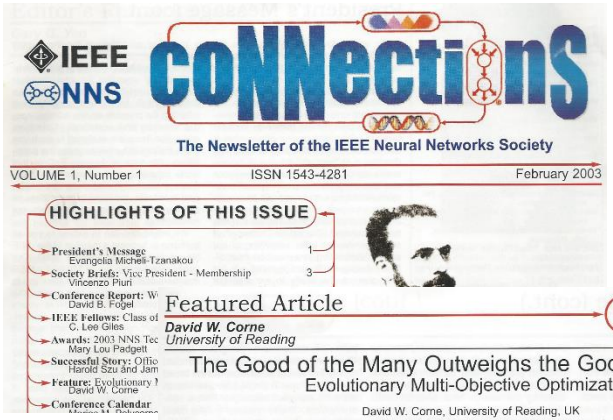
step 1

High-level information

step 2

Choose one solution





Mathematical Definition

- Mathematical model to formulate the optimization problem

Objective vectors | Decision vectors | Environment States | Equality constraints | Inequality constraints | Variable bounds

$$\min_{\mathbf{x}\in\Re^{\mathbf{n}}} \{\mathbf{y} \cong \mathbf{f}(\mathbf{x}, e) : \mathbf{h}(\mathbf{x}, e) = 0, \mathbf{g}(\mathbf{x}, e) \leq 0, x^L \leq x \leq x^U\}$$

  - Design Variables: decision and objective vector
  - Constraints: equality and inequality
  - Greater-than-equal-to inequality constraint can be converted to less-than-equal-to constraint by multiplying -1
  - Objective Function: maximization can be converted to minimization due to the *duality principle*

$$\max\ f(x) = \min\ (-f(x))$$

- Search for the decision vectors $x = [x_1^*, \cdots, x_n^*]^T$ which will satisfy

  the *l* inequality constraints:

  $$g_i(x) \leq 0; \quad i = 1, \cdots, l;$$

  the *p* equality constraints:

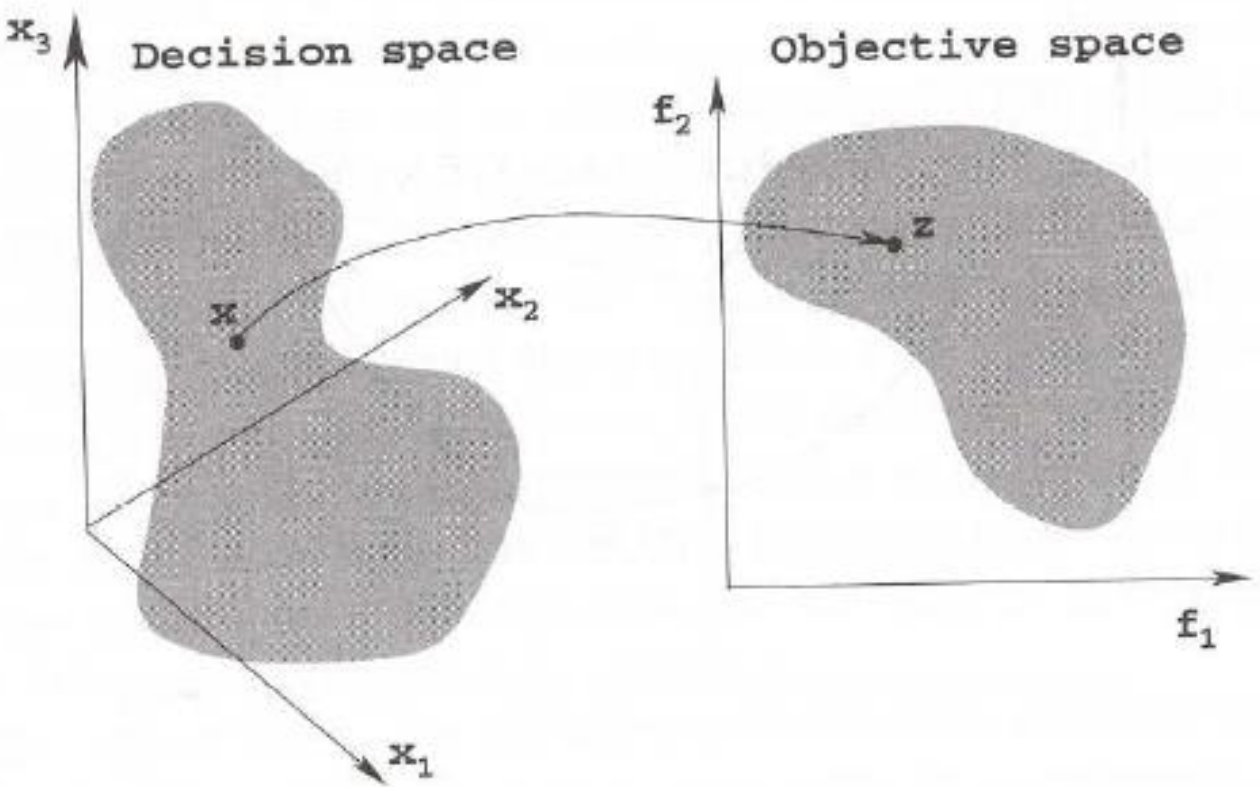
  $$h_i(x) = 0; \quad i = 1, \cdots, p;$$

  while optimizes the objective vector function

  $$f(x) = [f_1(x), f_2(x), \ldots\ldots f_m(x)]^T$$

- Set of feasible solutions (feasible region, or search space) is a subset of decision space

Mapping



Mapping takes place from an *n*-dimensional decision space to an *m*-dimensional objective space

Pareto Optimality

- Formal Definition: the minimization of the *n* components

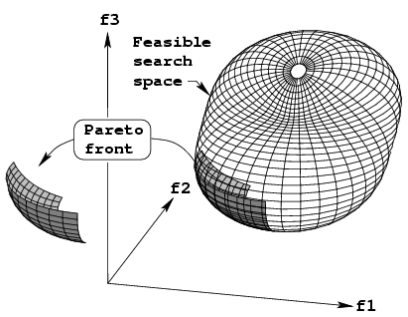
$$f_k, k = 1, \cdots, n$$

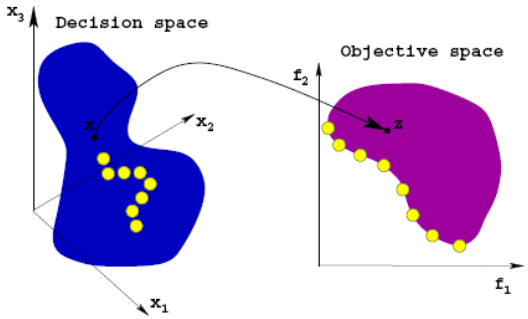
of a vector function **f** of a vector variable **x** in a universe μ, where

$$\mathbf{f}(\mathbf{x}) = (f_1(\mathbf{x}), f_2(\mathbf{x}), \cdots, f_n(\mathbf{x}))$$

- Then a decision vector $\mathbf{x}_u \in \mu$ is said to be Pareto-optimal if and only if there is no $\mathbf{x}_v \in \mu$ for which $\mathbf{v} = \mathbf{f}(\mathbf{x}_v) = (v_1, \cdots, v_n)$ dominates $\mathbf{u} = \mathbf{f}(\mathbf{x}_u) = (u_1, \cdots, u_n)$, that is, there is no $\mathbf{x}_v \in \mu$ such that

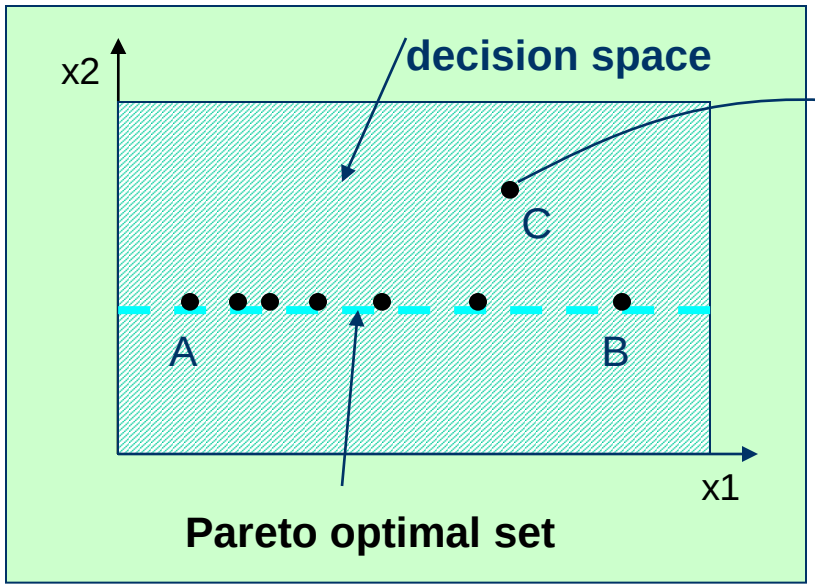
$$\forall i \in \{1, \cdots, n\}, v_i \leq u_i \quad \text{and} \quad \exists i \in \{1, \cdots, n\} \mid v_i < u_i$$

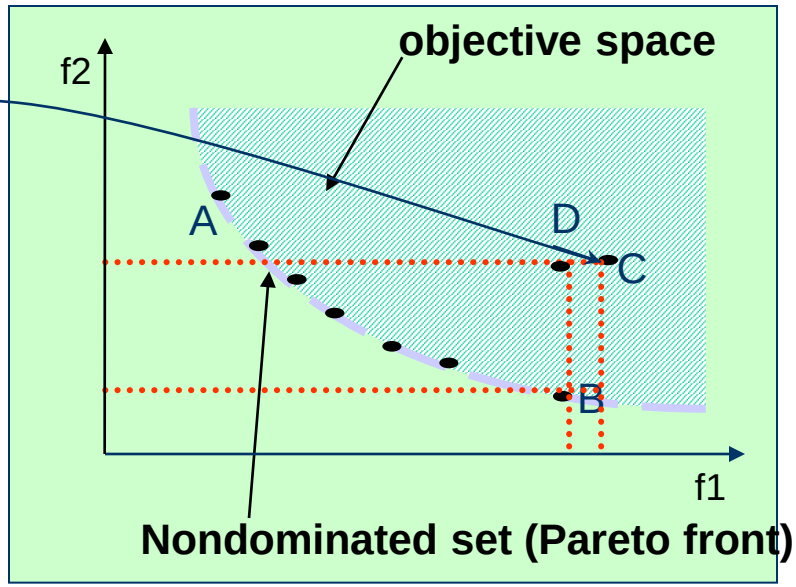




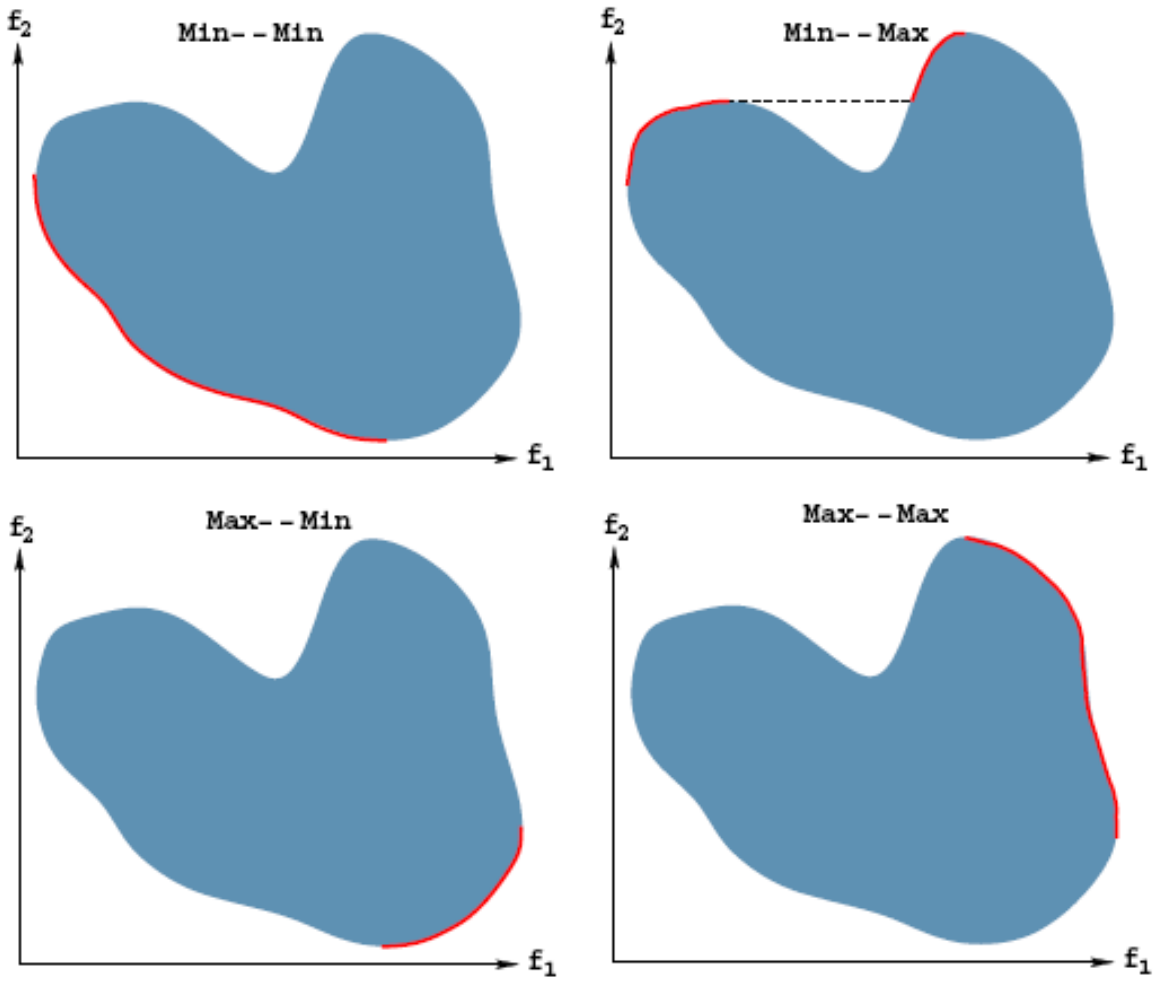
Pareto Optimal Set– the set of all Pareto-optimal *decision vectors*, which yields a set of nondominated solutions

Non-dominated Set– corresponding objective vector set-Pareto Front

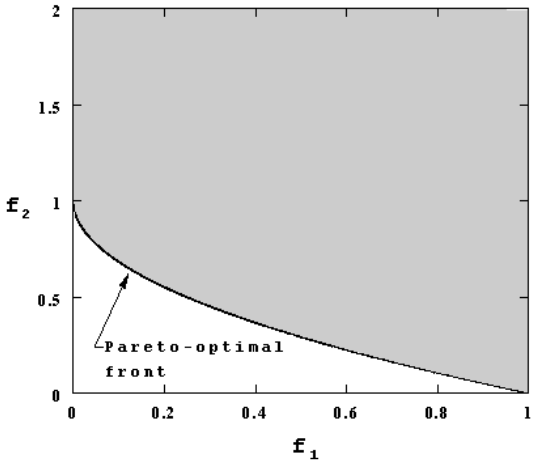


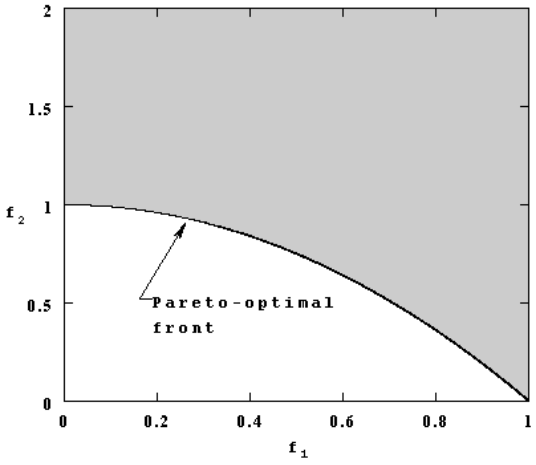


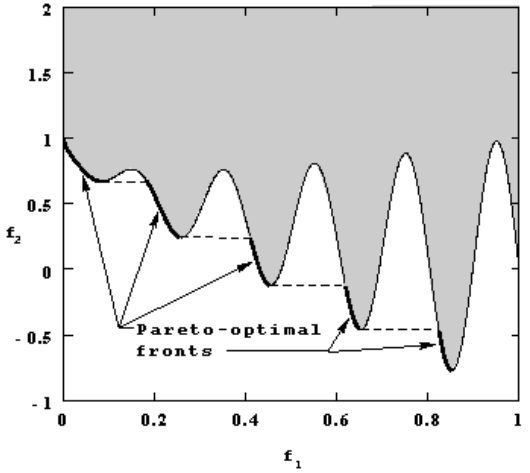
Pareto-Optimal Fronts

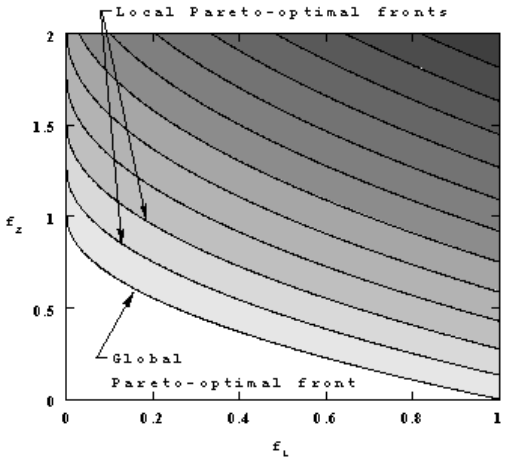


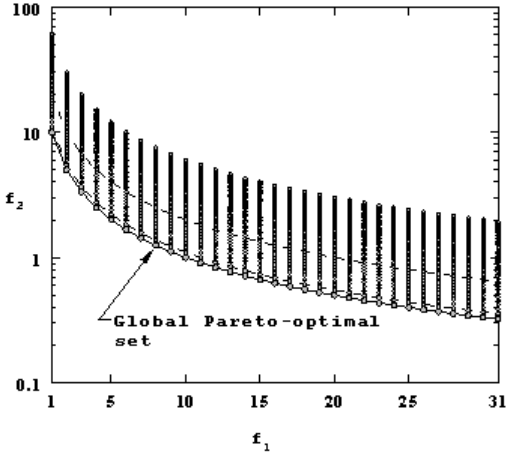
ZDT Test Suite

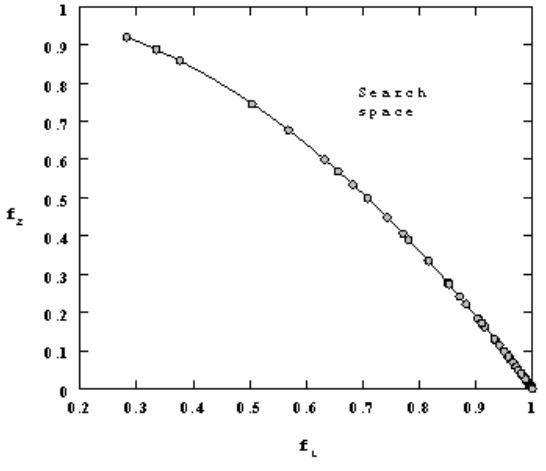










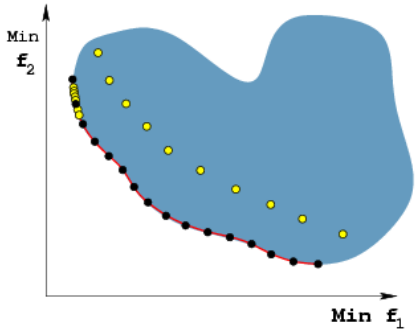


Distinctions from SOP

- Multiple conflicting objectives as opposed to single one
- Multiple optima vs. single optimum
- Two goals instead of one
  - Progressing towards the Pareto front

    - ***Convergence***
  - Maintaining a diverse set of solutions in the non-dominated front

    - ***Diversity***
- Dealing with two search spaces
  - A decision variable space plus an objective space
  - A proximity of two solutions in one space does not mean a proximity in the other space
  - Search is performed in the decision space



Concept of Domination

- Most multi-objective optimization algorithms use the concept of domination.
- In these algorithms, *two* solutions are compared on the basis of whether one dominates the other solution or not.

If $x^{(1)}$ dominates solution $x^{(2)}$, it is customary to write any of the following

$x^{(2)}$ is dominated by $x^{(1)}$

$x^{(1)}$ is non-dominated by $x^{(2)}$

$x^{(1)}$ is non-inferior to $x^{(2)}$

Non-Dominated Set

- Among a set of solutions *P*, the non dominated set of solutions *P*' are those that are not dominated by any member of the set *P*.

- When *P* is the entire search space, (*P*=S), the resulting non dominated set *P*' is called the *Pareto optimal set*.

Dominance Relations

- Any two solutions of *non-dominated set* must be non-dominated with respect to each other.
- Any solution not belonging to *non-dominated set* is dominated by at least one member of the *non-dominated set*.
- If solution x_1 does not dominate solution x_2, this does not imply that x_2 dominates x_1 (i.e., solutions 3 and 5)
- Three possible outcomes of these relations: x_1 dominates x_2; x_1 is dominated by x_2; or x_1 and x_2 do not dominate each other

Strong and Weak Dominance

- Definition: A solution $x^{(1)}$ strongly dominate a solution $x^{(2)}$, if solution $x^{(1)}$ is strictly better than solution $x^{(2)}$ in all objectives

- Definition: Among a set of solutions *P*, the weakly non-dominated set of solution *P*' are those that are not strongly dominated by any other member of the set *P*.

Identifying the Non-Dominated Set

- Similar to finding the minimum of a set of real numbers
- Approach 1: Naïve and Slow
  - **Step 1:** set solution counter *i*=1 and create an empty non-dominated set *P* ‘
  - **Step 2:** for a solution $j \in P$ (but $j \neq i$) , check if solution *j* dominates solution *i.* If yes, go to Step 4
  - **Step 3:** if more solutions are left in *P*, increment *j* by one and go to Step 2; otherwise, set $P' = P' \cup \{i\}$
  - **Step 4:** increment *i* by one. If $i \leq N$, go to Step 2; otherwise stop and declare *P* ‘ as the non-dominated set
- Computational complexity: O(MN^2)

Identifying the Non-Dominated Set

- Approach 2: Kung *et al.*'s Efficient Method
  - **Step 1:** Sort the population according to the descending order of importance in the first objective function and rename the population as P of size N.
  - **Step 2: Front**(P) If $|P| = 1$, return P as the output of **Front**(P). Otherwise, T = **Front**($P^{(1)} - P^{(|P|/2)}$) and B = **Front**($P^{(|P|/2+1)} - P^{(|P|)}$).
  - If the i'_{th} solution of B is not dominated by any solution of T, create a merged set $M = T \cup \{i\}$. Return M as the output of **Front**(P).

- Step uses a self-recursion
- Computational complexity: $O(N \log N)$

Non-Dominated Sorting

- Some evolutionary algorithms which require the entire population to be classified into various non-domination levels.
- The population needs to be sorted according to an ascending level of non domination. The best non-dominated solutions are called non-dominated solutions of level 1.
- To find solutions for the next level of non-domination, the current best non-dominated set is temporarily disregarded (or removed) from the population.
- The non-dominated solutions of the remaining population are then found and are called non-dominated solutions of the next level.
- Non-dominated solutions of level 1 are better than non-dominated solutions of level 2.

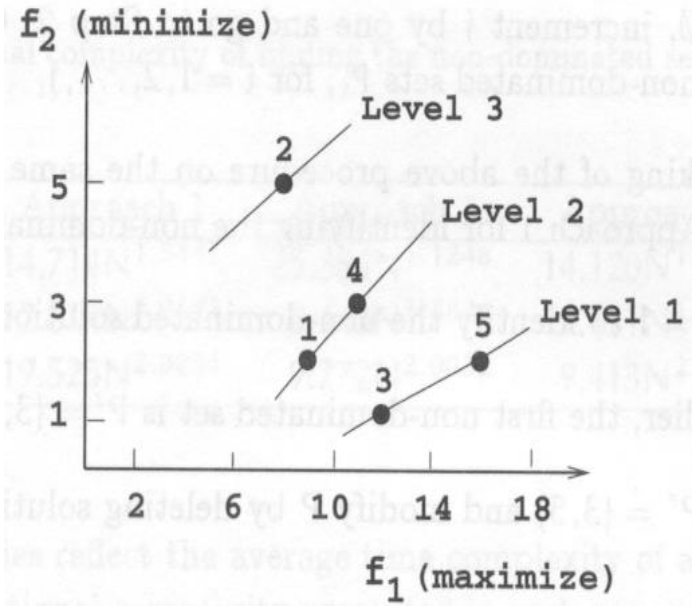
Non-Dominated Sorting of a Population

Step 1: Set all non-dominated sets P_j *(j=1,2,..)* as empty sets. Set non-dominated level counter $j = 1$

Step 2: Use any one of the approaches to find the non-dominated set *P'* of population *P*.

Step 3: Update $P_j = P'$ and $P = P \backslash P'$

Step 4: If *P* is not empty, increment *j* by one and go to Step 2. Otherwise, stop and declare all non-dominated sets P_i, for (i = 1,2,….)



Diversity Preservation

- Using a limited number of data points to approximate a continuous line segment to preserve representation power
- In order to maintain multiple optimal solutions, an explicit diversity-preserving operator must be used.
- The *mutation* operator is often used as a diversity preserving-operator (to enhance exploration behavior).
- It can create a better solution by perturbing a solution, it can also destroy a good solution.
- Since it is computationally expensive to check the worth of every possible mutation outcome, it is usually given a small probability in a GA.
- Mutation operator solely is *not* enough to maintain multiple optimal solutions.

Sharing Function Model by Goldberg (1987)

- Instead of replacing by similar solution, degrade the fitness of similar solutions.
- If we are interested in finding *q* optimal solutions with a finite population of size *N*, there is an expected occupancy m_i such that $\sum_{i=1}^{q} m_i = N$
- If an optimum has more than the expected number of representatives, this comes at the expense of other optimums. So degrade the fitness of the extra representatives.

- Solutions of an under represented optimum must be emphasized.
- Identifying solutions belonging to each true optimum demands a lot of problem knowledge and is not usually available
- Use an adaptive strategy, where *sharing function is used to obtain an estimate of the number of solutions belonging to each optimum*.

$$Sh(d) = \begin{cases} 1 - \left(\frac{d}{\sigma_{share}} \right)^{\alpha}, & d \le \sigma_{share} \\ 0, & otherwise \end{cases}$$

- The parameter *d* is the distance between any two solutions in the population.
- If *d* = 0 (meaning that two solutions are identical or their distance is zero), *Sh*(*d*)=1 and a solution has full sharing effect on itself, and if *d* > σ_{share} (meaning that two solutions are at least a distance of σ_{share} away from each other), *Sh*(*d*)=0 and the two solutions have no sharing effect on each other.

- The sharing function is calculated for all members and a niche count is defined as

$$nc_i = \sum_{j=1}^{N} Sh(d_{ij})$$

- The niche count provides an estimate of crowding near a solution
- Then, we calculate the shared fitness value as

$$f_i' = f_i / nc_i$$

- Since all over-represented optima will have large nc_i value, the fitness of all these optima would be degraded by a large amount, and for the under-represented optima the degradation will not be much.

Parameters of Sharing Function Model

- Two new parameters have been introduced α and σ_{share}
- σ_{share} need to be set right in order to define the niche size of an optimum. The niche size is related to the basin of attraction of the optimum.
- σ_{share} depends on whether we calculate distances between genotypes or phenotypes.

Classical Approaches

Classification by Miettinen (1999):
Increasing use of preference information

1. ***No-preference methods* (heuristic based)**:
   does not assume any information about the importance of objectives, but a heuristic is used to find a single optimal solution.

2. ***Posteriori methods* (generating solutions):**
   uses preference information (knowledge on algorithmic parameters) of each objective and iteratively generate a set of Pareto-optimal solutions

3. ***A priori methods*** **(one preferred solution):**

   use more information about the preferences of objectives and usually find one preferred Pareto-optimal solution

4. ***Interactive methods*** **(involving a decision maker):**

   use the preference information progressively during the optimization process

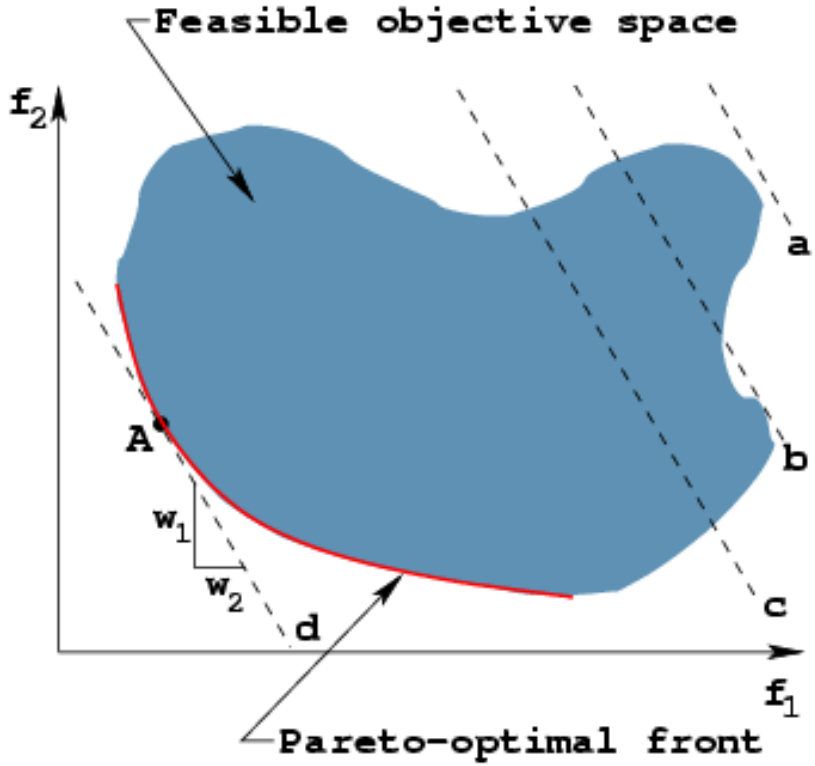
Weighted Sum Method

- The method scalarizes a set of objectives into a single objective by pre-multiplying each objective with a user supplied weight.

$$F(x) = \sum_{i=1}^{k} w_i f_i(x), \quad 0 \le w_i \le 1, \quad \sum_{i=1}^{k} w_i = 1$$

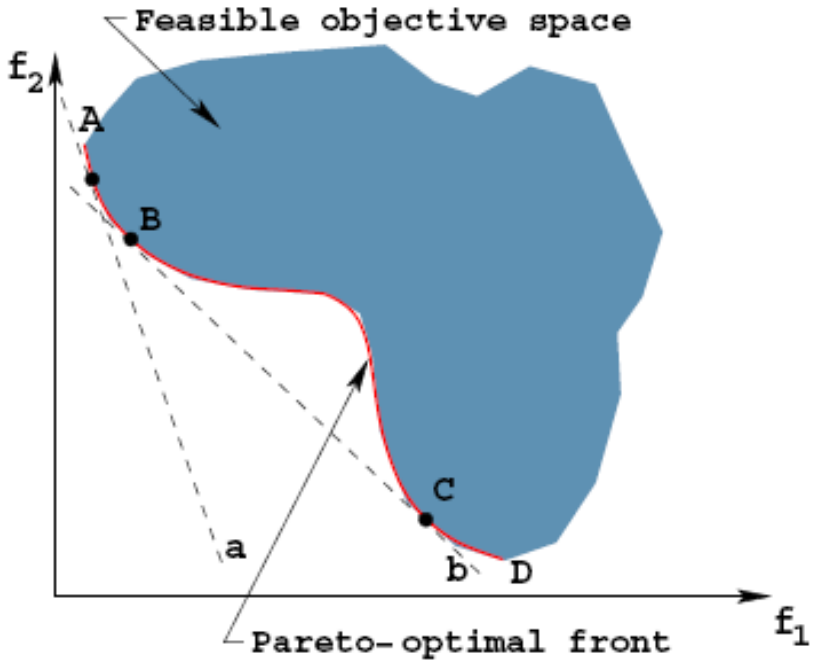
- How to choose the weights ? – Depends on the importance of each objective in the context of the problem.
- The weights also will depend on the range of each of the objective functions.

- Knowing the weights we can calculate the composite function *F*.
- The contour surfaces of *F* can then be visualized in the objective space, as shown by the lines ‘a’,’b’,’c’ and ‘d’
- Moving from ‘a’ to ‘b’ is effectively lowering the value of *F*.
- So to minimize *f* we need to find a line tangential to the feasible objective space and it must lie in the bottom left corner.



Difficulty with Non Convex Problems

- No contour line will produce a tangent point with the feasible objective space in the region BC.
- So the region BC will remain undiscovered using the weighted sum approach



Advantages

- This is the simplest way to solve an MOP.
- The concept is intuitive and easy to use.
- For problems having a convex Pareto-optimal front, this method guarantees finding solutions on the entire Pareto-optimal set.

Disadvantages

- A uniformly distributed set of weight vectors need not find a uniformly distributed set of Pareto-optimal solutions.
- Different weight vectors need not lead to different Pareto-optimal solutions.
- Cannot find certain Pareto-optimal solutions in the case of a non convex (concave) objective space.

ε-Constraint Method

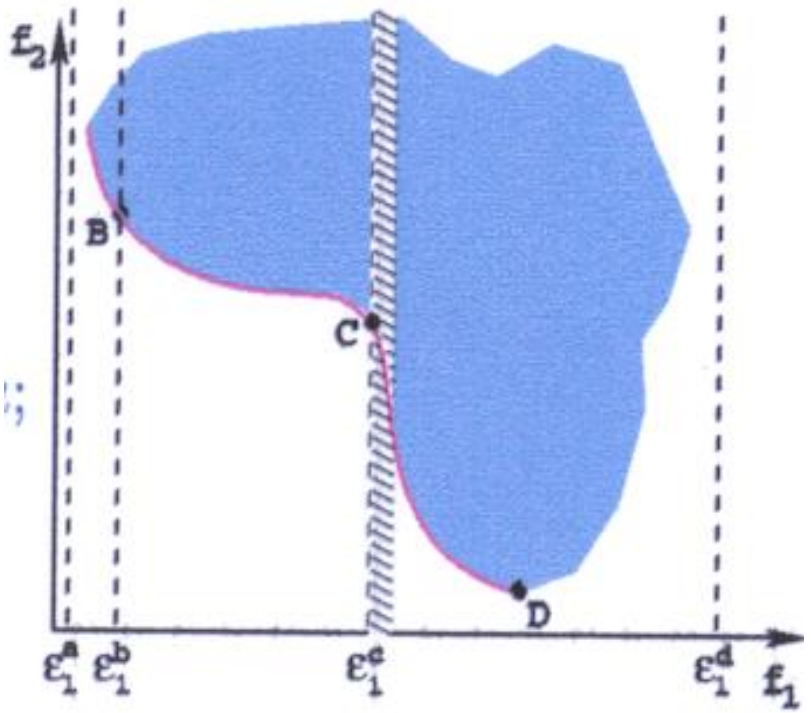
- Haimes *et al*. (1971) suggested reformulating the MOP by just keeping one of the objectives and restricting the rest of the objectives within user specified values.

- Minimize $f_{\mu}(x)$

  subject to $f_m(x) \leq \varepsilon_m \qquad m = 1,2,\ldots,M \quad m \neq \mu$

  $x_i^{(L)} \leq x_i \leq x_i^{(U)} \qquad i = 1,2,\ldots,n$

- Minimize f_2 and treat f_1 as a constraint
- Consider $\varepsilon_1 = \varepsilon_1^c$, now the portion BC of the Pareto-optimal front becomes a feasible solution of the imposed problem.
- In this way, intermediate Pareto-optimal solutions can be obtained in the case of non-convex objective spaces.



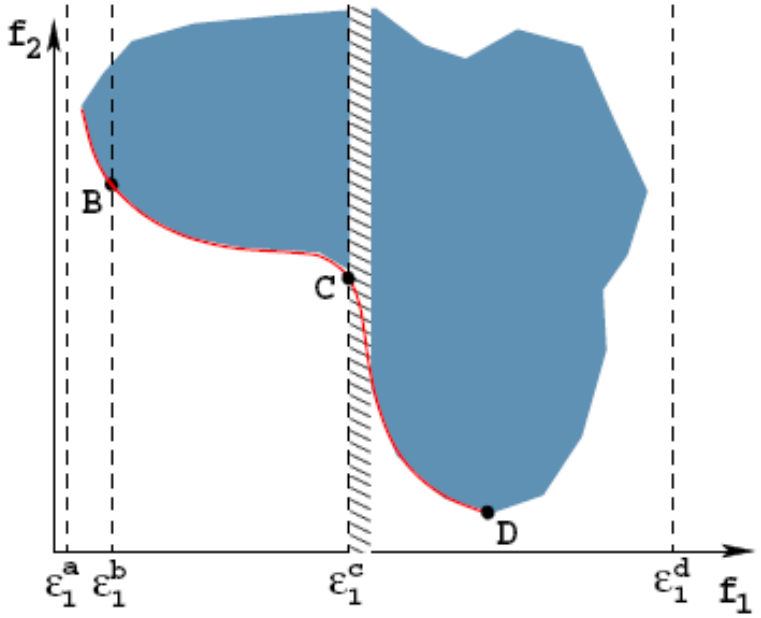
Advantages

- Different Pareto-optimal solutions can be found by using different ε_m values.

- Can be used for any arbitrary problem with either convex or non-convex objective space.

Disadvantages

- The solution to the problem largely depends on the chosen vector ε. It must be chosen such that it lies within the maximum or minimum values of the individual objective function.
- As the number of objectives increases, more information is required form the user to choose the vector ε



ε_a were chosen, there exists no easible solution to the problem.

ε_d were chosen , the entire search pace is feasible

Other Approaches

- Weighted Metric Methods
- Benson's Methods
- Value Function Methods
- Goal Programming Methods
- Interactive Methods

etc

Disadvantages of Classical Methods

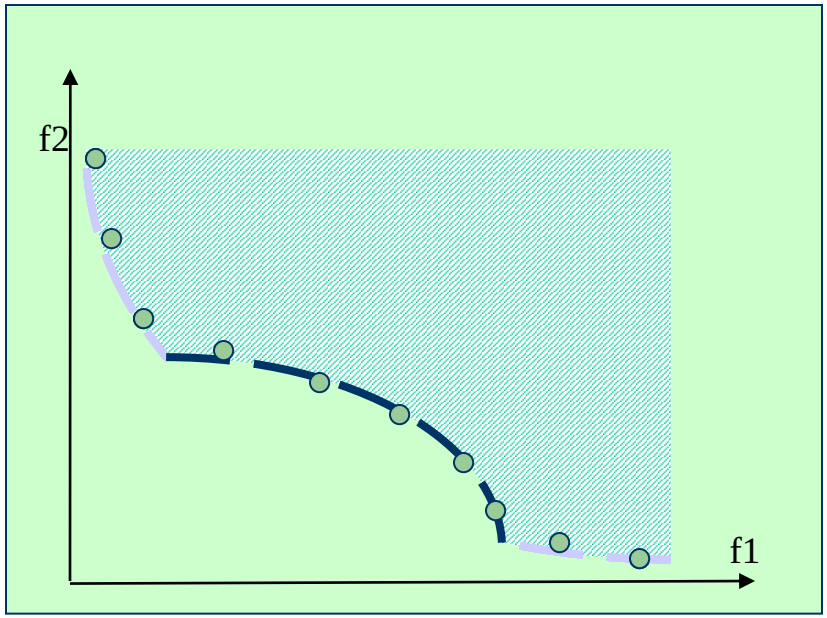
- We need prior knowledge of the problem domain to result into a single objective optimization problem (e.g., weight vector, ε constraints)
- Results in a single solution for each run
- Non-uniformity in Pareto-optimal solution
- Require fitness function to be linear, continuous and differentiable
- Cannot deal with MOPs having ***discontinuous*** and ***concave*** Pareto fronts

Why *Nature-Inspired* Heuristics?

- An unorthodox, stochastic, and population based parallel searching heuristics maybe more suitable for MOPs
- Classification of EA's–
  - ***Genetic Algorithm***;
  - Genetic Programming;
  - Evolutionary Programming;
  - Evolutionary Strategy;
  - Ant Colony;
  - ***Artificial Immune System***;
  - ***Particle Swarm Optimization***;
  - ***Differential Evolution***;
  - Memetic Algorithm

ability of handling complex problems with discontinuities, multimodality, disjoint feasible spaces and dynamism

- A population based parallel searching algorithm maybe more suitable for MOPs

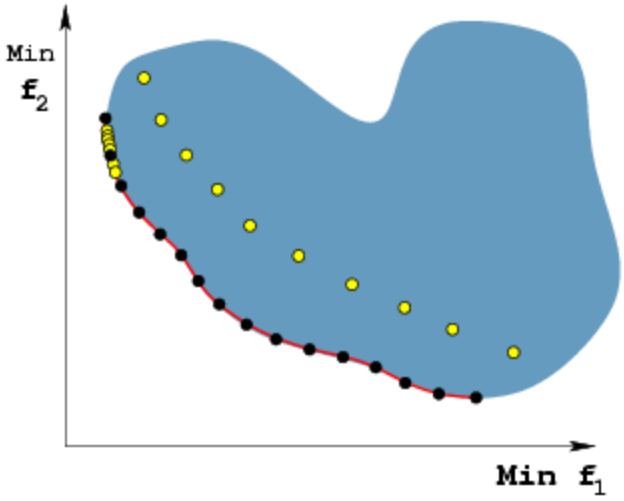


To search for a 1) ***near-optimal***, 2) ***uniformly distributed*** and 3) ***well extended*** set of solutions for MOPs

Quantitative Measurement

How to measure the performance in Objective Space (not in decision space):

- Convergence to Pareto front
- Uniform distribution or diversity
- Extension
- Anything else???



Evolutionary Algorithms for MOPs

- Derived from the conception of evolution in Biology - Darwin's "survival of the fittest" (natural selection) law
- Classification of EA's– Genetic Algorithm; Genetic Programming; Evolutionary Programming; Evolutionary Strategy; Ant colony; Immune Algorithm; Swarm Intelligence; …
- GA:

1 **Randomly generate the initial population P(0), and set iteration index i=0;**

2 **REPEAT**

(a) Evaluate the fitness of each individual in P(i);

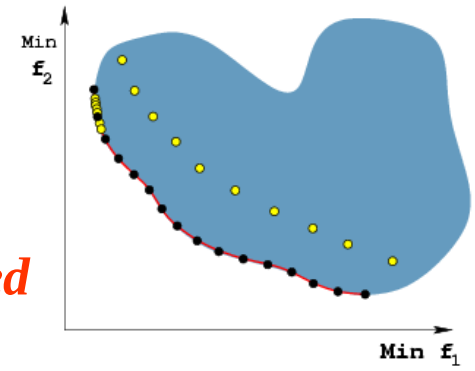
(b) Select parents from P(i) based on their fitness in P(i);

(c) Apply genetic operations to the parents and derive generation P(i+1);

UNTIL the stopping criteria are met;

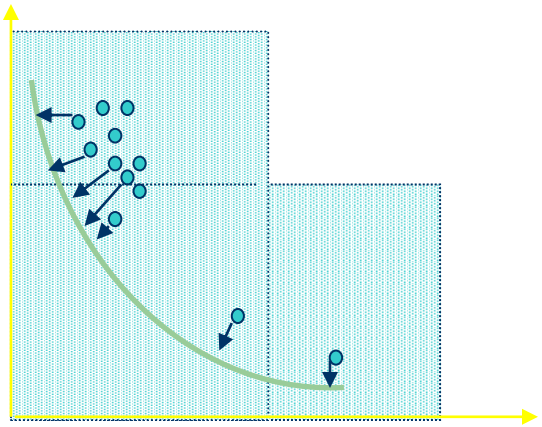
Why EA for MOPs?

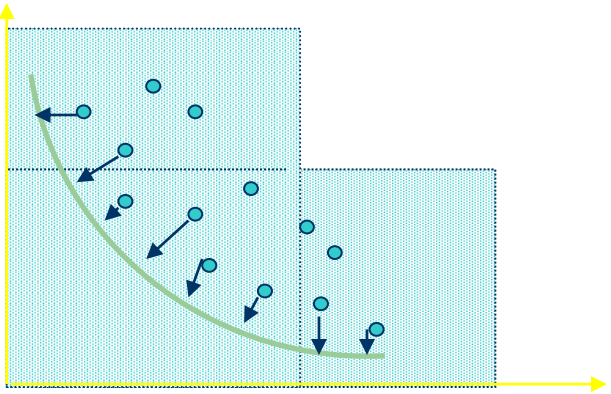
- GA/PSO's common characteristics
  - No Assumption– ability of handling complex problems
    - discontinuities, multimodality, disjoint feasible space and
  - Population based
    - Parallel search– escape from local optimum
    - Derive *a set* of solutions simultaneously in each generation– find concave Pareto front
    - Final result is a set of Pareto optimal points– reduce running times
  - Fast converging
  - Search for a **near-optimal, uniformly *distributed*** and ***well extended*** set of solutions for MOPs



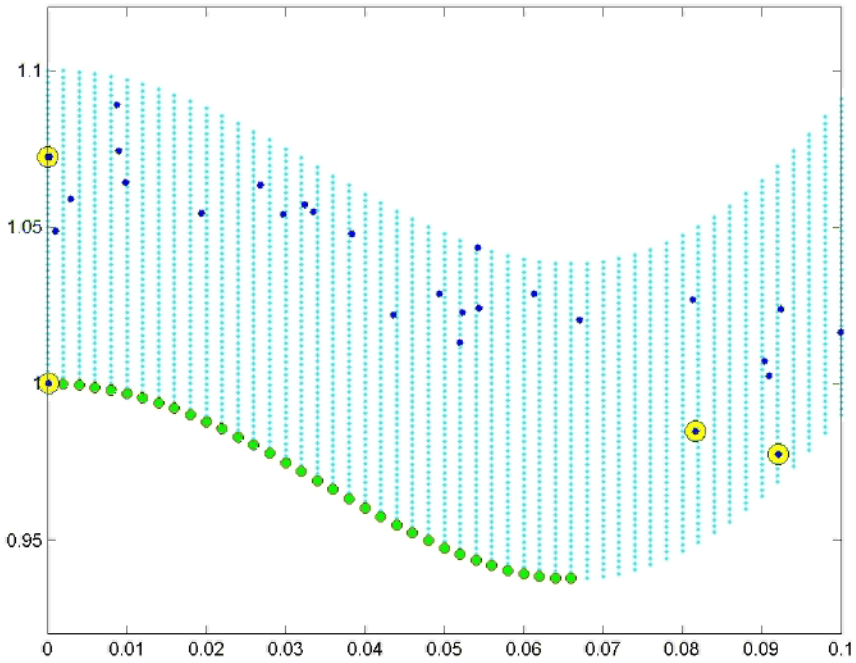
EA's Dilemma in Solving MOPs

- Limited computation resource – "genetic drift" phenomenon
  - A high fitness individual has tendency to create a congregated area around it
  - Good for SOP in refining to a unique solution– MOP needs to ***exploit*** it
  - Harmful for MOP in searching for a set of solution– MOP needs to ***avoid*** it





Demonstration: EA's Dilemma



- To ***avoid*** and ***promote*** "genetic drift" phenomenon simultaneously
  - preserve population diversity during the evolutionary process
  - use dynamic population size instead of a fixed one

Efforts in Enhancing EA for MOP

- Modifying the fitness assignment
- Improving flight mechanism/genetic operators
- Enhancing the *convergence*
- Preserving the *diversity*
- Managing the *population*
- Constraints and uncertainty handling

Critical Message Conveyed

Multiple Objective Optimization

present an opportunity to advance the next technology frontier

Q&A