Multi-Objective Optimization

Sriparna Saha

Assistant Professor
Department of Computer Science & Engineering
Indian Institute of Technology Patna

sriparna@iitp.ac.in

11 March 2016

IIT Patna, Patna

Outline

Outline

Motivation

Ticket	Travel Time (Hrs.)	Ticket Cost (Rs.)
X^1	18	300
X^2	12	600
X^3	6	1800
X^4	18	1200
X^5	12	2400

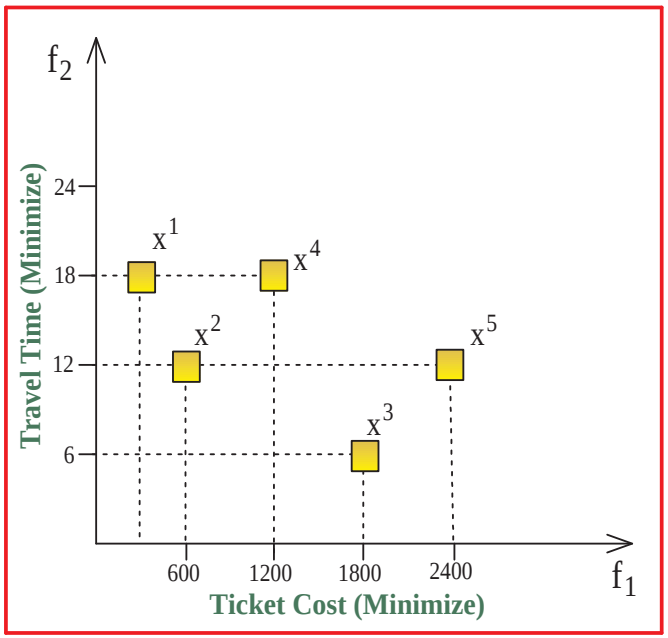


Figure 1 : Train Ticket Option

Outline

Problem Statement

Definition (Multiobjective optimization: MOO)

When an optimization problem involves more than one objective function, the task of finding one or more optimal solutions is known as Multiobjective optimization.

Example

Find out ticket in the train with minimum cost and minimum travel time with some constraints.

Example: MOO

Example

1. Optimizing criteria
   - Minimizing the ticket cost
   - Minimizing the travel time
2. Constraints
   - Not have more than 2 stoppage between source and destination
   - Should have pantry car.
3. Decision variables
   - The available trains

In many real world problems we have to simultaneously optimize two or more different objectives.

- Finding a single solution in these cases is very difficult.
- Optimizing each criterion separately may lead to good value of one objective while some unacceptably low value of the other objective(s).

Mathematical Definition: MOO

Definition (Mathematical Definition)

The Multiobjective optimization can be formally stated as:
Find the vector of decision variables

$$\mathbf{x} = [x_1, x_2, \ldots, x_n]^T$$

which will satisfy the P inequality constraints:

$$g_i(\mathbf{x}) \geq 0, i = 1, 2, \ldots, P$$

And the Q equality constraints

$$h_i(\mathbf{x}) = 0, i = 1, 2, \ldots, Q$$

And simultaneously optimizes M objective functions

$$f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_M(\mathbf{x})$$

.

Outline

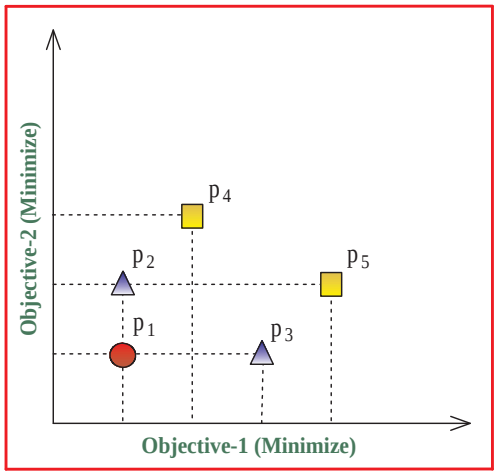
Concept of Domination

Definition (Domination)

A solution $x^i = \{f_1(x^i), f_2(x^i), \ldots, f_M(x^i)\}$ is said to dominate solution $x^j = \{f_1(x^j), f_2(x^j), \ldots, f_M(x^j)\}$ denoted as $x^i \prec x^j$ iff

1. $f_m(x^i) < f_m(x^j), \quad \exists m \in \{1, 2, \ldots, M\}$
2. $f_m(x^i) \leq f_m(x^j), \quad \forall m \in \{1, 2, \ldots, M\}$

Two solutions x^i and x^j are said to be non-dominated with each other iff neither $x^i \prec x^j$ nor $x^j \prec x^i$.



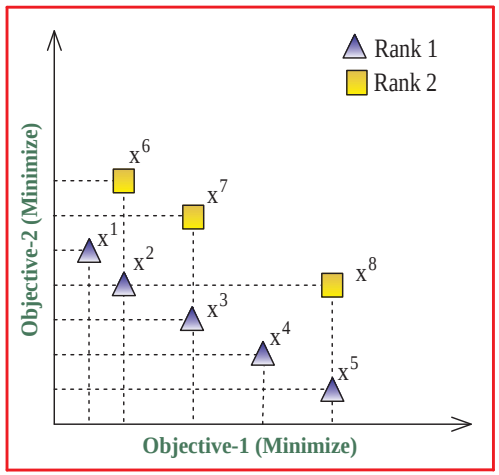
Pareto-Optimality

Definition (Pareto-optimal solution)

A solution $x \in P$ is called Pareto-optimal with respect to P if there is no solution $x' \in P$ such that x is dominated by x'.

Definition (Pareto set)

The set of Pareto-optimal solutions is known as Pareto set.



Non-dominated Sorting Genetic Algorithm-II

NSGA-II [1]

1. Approach to solve MOO.
2. Find Pareto-optimal front in a single run.
3. Find a set of solutions as diverse as possible.
4. Developed to overcome the limitation of NSGA [2].
   - High computational complexity of nondominated sorting
   - Need for specifying the sharing parameter σ_{share} for diversity preservation
   - Lack of elitism

Outline

High Computational Complexity

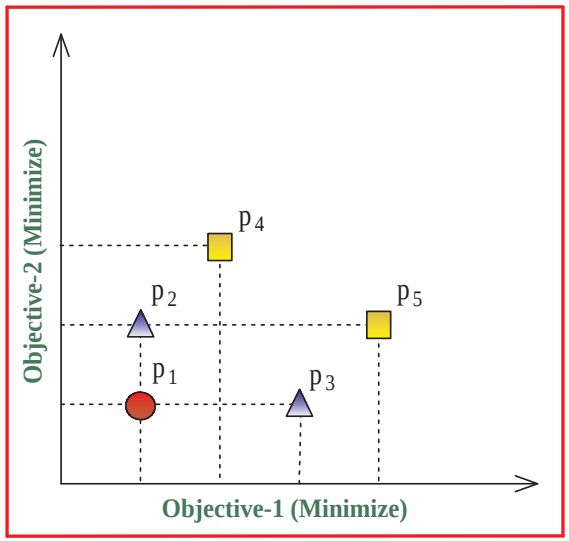
Definition (Non-Dominated Sorting)

Non-Dominated Sorting is to divide the population $\mathbb{P}$ in $K(1 \leq K \leq N)$ fronts. Let the set of these K fronts in decreasing order of their dominance is $\mathcal{F} = \{F_1, F_2, \ldots, F_K\}$. The division of the solutions is such that

1. Each solution in a front is non-dominated with each other.
2. Each solution in a front F_k is dominated by at-least one solution in its preceding front $F_{k'}, k' < k \bigwedge 1 \leq k, k' \leq K$.

Complexity

NSGA: $\mathcal{O}(MN^3)$
NSGA-II: $\mathcal{O}(MN^2)$
M: No. of objectives
N: Population size



Non-dominated Sorting

Algorithm 1 Non-dominated Sorting

1: **for each** $p \in \mathbb{P}$ **do**
2: $S_p \leftarrow \Phi$, $n_p \leftarrow 0$
3: **for each** $q \in \mathbb{P}$ **do**
4: **if** $p \prec q$ **then**
5: $S_p \leftarrow S_p \cup \{q\}$ // Add q to the set of solutions dominated by p
6: **else if** $q \prec p$ **then**
7: $n_p \leftarrow n_p + 1$ // Increment the domination counter of p
8: **if** $n_p = 0$ **then**
9: $p_{\text{rank}} \leftarrow 1$
10: $F_1 \leftarrow F_1 \cup \{p\}$
11: $i \leftarrow 1$ // Initialize the front counter
12: **while** $F_i \neq \Phi$ **do**
13: $Q \leftarrow \Phi$ // Used to store the members of the next front
14: **for each** $p \in F_i$ **do**
15: **for each** $q \in S_p$ **do**
16: $n_q \leftarrow n_q - 1$
17: **if** $n_q = 0$ **then**
18: $q_{\text{rank}} \leftarrow i + 1$
19: $Q \leftarrow Q \cup \{q\}$
20: $i \leftarrow i + 1$
21: $F_i \leftarrow Q$

Non-dominated Sorting: Example

$n_{p_1} = 0$	$S_{p_1} = \{p_2, p_3, p_4, p_5\}$
$n_{p_2} = 1$	$S_{p_1} = \{p_4, p_5\}$
$n_{p_3} = 1$	$S_{p_1} = \{p_5\}$
$n_{p_4} = 2$	$S_{p_1} = \{\}$
$n_{p_5} = 3$	$S_{p_1} = \{\}$

$F_1 = \{p_1\}$

$n_{p_1} = 0$	$S_{p_1} = \{p_2, p_3, p_4, p_5\}$
$n_{p_2} = 0$	$S_{p_1} = \{p_4, p_5\}$
$n_{p_3} = 0$	$S_{p_1} = \{p_5\}$
$n_{p_4} = 1$	$S_{p_1} = \{\}$
$n_{p_5} = 2$	$S_{p_1} = \{\}$

$F_2 = \{p_2, p_3\}$

$n_{p_1} = 0$	$S_{p_1} = \{p_2, p_3, p_4, p_5\}$
$n_{p_2} = 0$	$S_{p_1} = \{p_4, p_5\}$
$n_{p_3} = 0$	$S_{p_1} = \{p_5\}$
$n_{p_4} = 0$	$S_{p_1} = \{\}$
$n_{p_5} = 0$	$S_{p_1} = \{\}$

$F_3 = \{p_4, p_5\}$

Outline

Diversity Preservation

1. The performance of the sharing function method in maintaining a spread of solutions depends largely on the σ_{share} (user defined parameter) [3].
2. Since each solution must be compared with all other solutions in the population, the overall complexity of the sharing function approach is $\mathcal{O}(MN^2)$ [4].

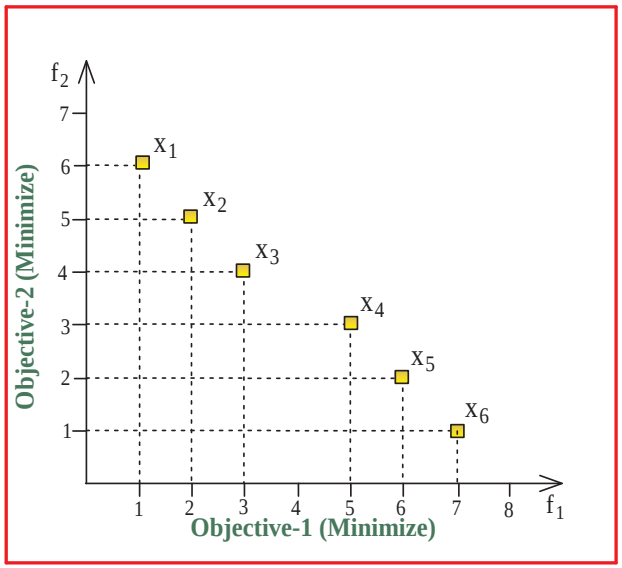


Figure 2 : A Pareto Optimal Front

Crowding Distance

	x^1	x^2	x^3	x^4	x^5	x^6
f_1	1	2	3	5	6	7
f_2	6	5	4	3	2	1

Sorted solution based on f_1 is $\langle x^1, x^2, x^3, x^4, x^5, x^6 \rangle$

Solution	x^1	x^2	x^3	x^4	x^5	x^6
Crowding distance	∞	$\frac{2}{6}$	$\frac{3}{6}$	$\frac{3}{6}$	$\frac{2}{6}$	∞

Sorted solution based on f_2 is $\langle x^6, x^5, x^4, x^3, x^2, x^1 \rangle$

Solution	x^6	x^5	x^4	x^3	x^2	x^1
Crowding distance	∞	$\frac{2}{6}+\frac{2}{5}$	$\frac{3}{6}+\frac{2}{5}$	$\frac{3}{6}+\frac{2}{5}$	$\frac{2}{6}+\frac{2}{5}$	∞

So final Crowding distance is

x^1	x^2	x^3	x^4	x^5	x^6
∞	$\frac{22}{30}$	$\frac{27}{30}$	$\frac{27}{30}$	$\frac{22}{30}$	∞

$\xrightarrow[\text{Crowding distance}]{\text{Sort based on}}$

x^1	x^6	x^3	x^4	x^2	x^5
∞	∞	$\frac{27}{30}$	$\frac{27}{30}$	$\frac{22}{30}$	$\frac{22}{30}$

Crowding Distance

Algorithm 2 Crowding-distance-Assignment(F)

1: $l \leftarrow |F|$ // Number of solutions in F
2: **for** $i \leftarrow 1$ to l **do**
3: $F[i]_{distance} \leftarrow 0$ // Distance Initialization
4: **for each** objective m **do**
5: $F \leftarrow \text{Sort}(F, m)$ // Sort using each objective value
6: $F[1]_{distance} \leftarrow \infty$ // Select boundary points
7: $F[l]_{distance} \leftarrow \infty$ // Select boundary points
8: **for** $i \leftarrow 2$ to $l-1$ **do**
9: $F[i]_{distance} \leftarrow F[i]_{distance} + \frac{(F[i+1].m - F[i-1].m)}{f_m^{max} - f_m^{min}}$

Complexity

Complexity: $\mathcal{O}(MN \log N)$

Crowded-Comparison Operator

The crowded-comparison operator ($\prec_n$) guides the selection process at the various stages of the algorithm toward a uniformly spread-out Pare-tooptimal front. Assume that every individual in the population has two attributes:

1. Non-domination Rank i_{rank}
2. Crowding distance i_{distance}

The comparison operator $\prec_n$ is defined as

$$i \prec_n j = \begin{cases} i_{\text{rank}} < j_{\text{rank}} \\ i_{\text{rank}} = j_{\text{rank}} \text{ and } i_{\text{distance}} > j_{\text{distance}} \end{cases}$$

Outline

NSGA-II

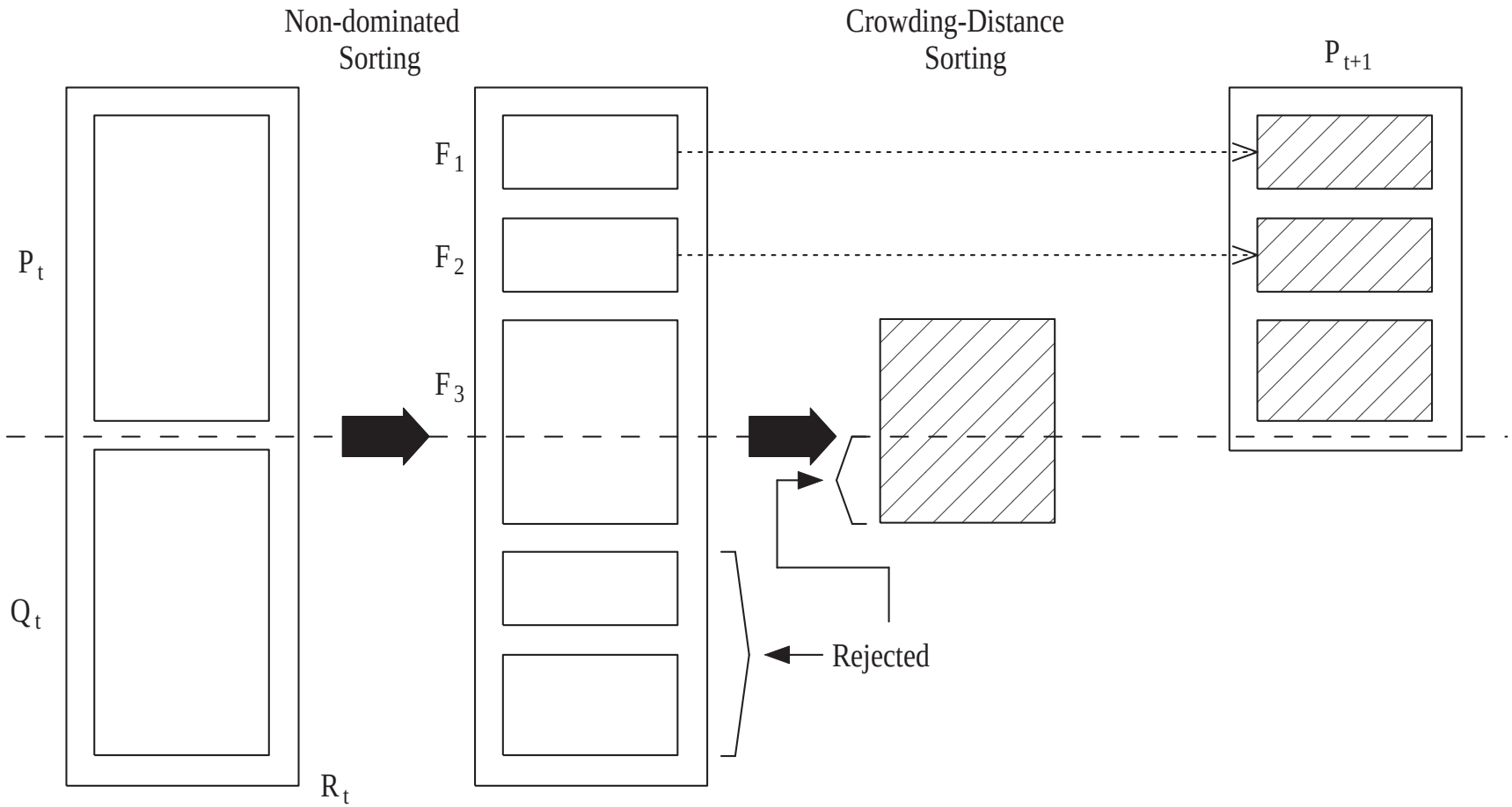


Figure 3 : NSGA-II Procedure

NSGA-II Algorithm

Algorithm 3 Working of one generation of NSGA-II

1: $R_t \leftarrow P_t \cup Q_t$ // Combine parent and offspring population
2: $\mathcal{F} \leftarrow$ Fast-Non-dominated-Sort(R_t)
3: $P_{t+1} \leftarrow \Phi$, $i \leftarrow 1$
4: **while** $|P_{t+1}| + |F_i| \leq N$ **do**
5: $\quad P_{t+1} \leftarrow P_{t+1} \cup F_i$
6: $\quad i \leftarrow i + 1$
7: Crowding-Distance-Assignment(F_i)
8: $Sort(F_i, \prec_n)$ // Sort in descending order using $\prec_n$
9: $P_{t+1} \leftarrow P_{t+1} \cup F_i[1 : (N - |P_{t+1}|)]$ // Choose the first $N - |P_{t+1}|$ elements of F_i
10: $Q_{t+1} \leftarrow$ Make-new-population(P_{t+1}) Use selection crossover and mutation to generate offspring population
11: $t \leftarrow t + 1$

NSGA-II

Complexity Analysis

- Non-dominated sorting: $\mathcal{O}(MN^2)$
- Crowding-distance assignment: $\mathcal{O}(MN \log N)$
- Sorting on $\prec_n$: $\mathcal{O}(MN \log N)$

Outline

Simulated Annealing and Multiobjective Optimization Problem

1. Simulated Annealing (SA) is another popular search algorithm.
   - utilizes the principles of statistical mechanics, regarding the behavior of a large number of atoms at low temperature, for finding minimal cost solutions to large optimization problems by minimizing the associated energy.
2. Geman and Geman provided a proof that simulated annealing, if annealed sufficiently slowly, converges to the global optimum.
3. Only a few attempts at using SA for MOOP
   - SA usually finds one solution instead of a set of solutions.
   - Difficulty in computing the acceptance probability.
   - Generally in the SA based MOOP algorithms, the set of Pareto-optimal solutions are evolved by using multiple SA runs.
   - The experimental results of the existing SA based MOOP algorithms are also not so good.

General Structure of SA

Algorithm 4 General Structure of SA

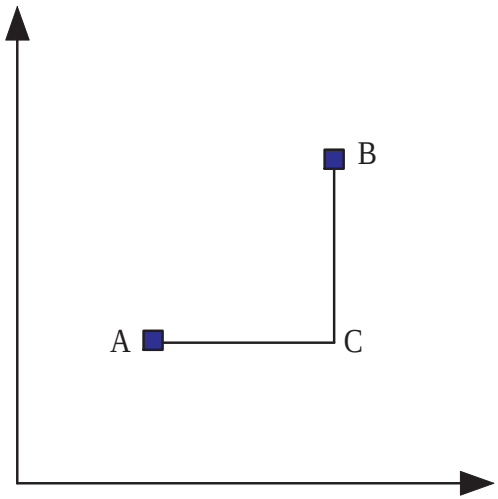
$T \leftarrow T_{\mathrm{MAX}}$
Generate current point Current randomly
$E \leftarrow$ Energy of Current
while $T > T_{\mathrm{MIN}}$ **do**
  for $i \leftarrow 1$ to N_T **do**
    Generate New with energy E^* from Current
    $\Delta E \leftarrow E^* - E$
    if $\Delta E < 0$ **then**
      Current $\leftarrow$ New
    else if $\exp -(\Delta E/T) > \mathrm{random}(0,1)$ **then**
      Current $\leftarrow$ New
  $T \leftarrow \alpha * T$ /* Cooling Schedule */

Archived Multiobjective Simulated Annealing Algorithm

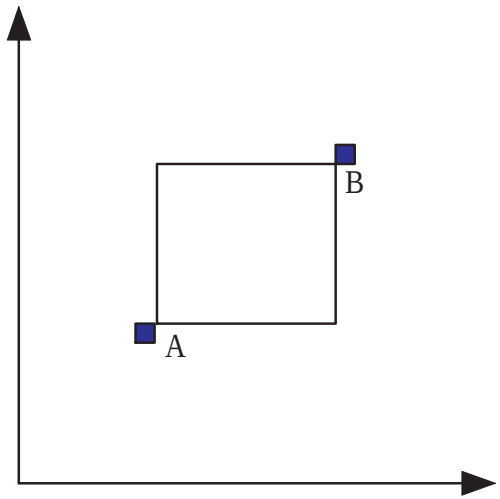
1. AMOSA
   - Based on Simulated Annealing.
   - Incorporates the concept of an archive where the non-dominated solutions seen so far are stored.
   - Uses clustering to restrict the size of the archive and to ensure diversity.
   - Uses amount of domination for computing the acceptance probability depending on domination status of the new solution, current solution and archive.
2. There are two limits kept on the size of the archive: Hard-limit and Soft-limit
3. During the process the non-dominated solutions are stored in the archive as and when they are generated until the size of the archive increases to Soft-limit.
4. There after if more non-dominated solutions are generated, the size of the archive is first reduced to Hard-limit by applying clustering.

Amount of Domination

Given two solutions a and b, let the amount of domination be $\Delta\mathrm{dom}$. This $\Delta\mathrm{dom}$ value is used in AMOSA while computing the probability of acceptance of a newly generated solution.



(a) Amount of domination between A and $B = (AC + BC)$.



(b) Amount of domination between A and $B = (AC + BC)$.

The AMOSA Algorithm

Algorithm 5 AMOSA Algorithm

1: Set $T_{\max}$, $T_{\min}$, Hard-limit, Soft-limit, iter, α
2: Set $\text{temp} \leftarrow T_{\max}$
3: Initialize the archive.
4: Current-pt $\leftarrow$ random(archive).
5: **while** $\text{temp} > T_{\min}$ **do**
6: **for** $j \leftarrow 1$ to iter **do**
7: New-pt $\leftarrow$ Perturb(Current-pt)
8: Decision about Current-pt based on domination status
9: temp $\leftarrow \alpha \times$temp
10: **if** Archive-size $>$ Soft-limit **then**
11: Archive $\leftarrow$ cluster(Archive, Hard-limit)
12: Output Archive

Procedure: Archive Initialization

Algorithm 6 Archive Initialization

1: Randomly generate $\gamma\times$ Soft-limit solutions in A
2: **for** $i \leftarrow 1$ to $\gamma\times$ Soft-limit **do**
3: $A'[i] \leftarrow$ Perturb$(A[i])$
4: **if** $A'[i]$ dominates $A[i]$ **then**
5: $A[i] \leftarrow A'[i]$
6: $ND \leftarrow$ Select all non-dominated solutions form A
7: Archive $\leftarrow$ cluster(Archive, Hard-limit)

Case-1: current-pt dominates new-pt

1. Find the number of points $h(h \geq 0)$ that dominate the new-pt.
2. $\Delta\text{dom} = \sum h_{i=1}(\Delta\text{dom}_{i,\text{new-pt}}) + (\Delta\text{dom}_{\text{current-pt,new-pt}})$
3. $\text{prob} = \frac{1}{1+\exp(\Delta\text{dom}/h \times \text{temp})}$
4. Set new-pt as current-pt with probability prob

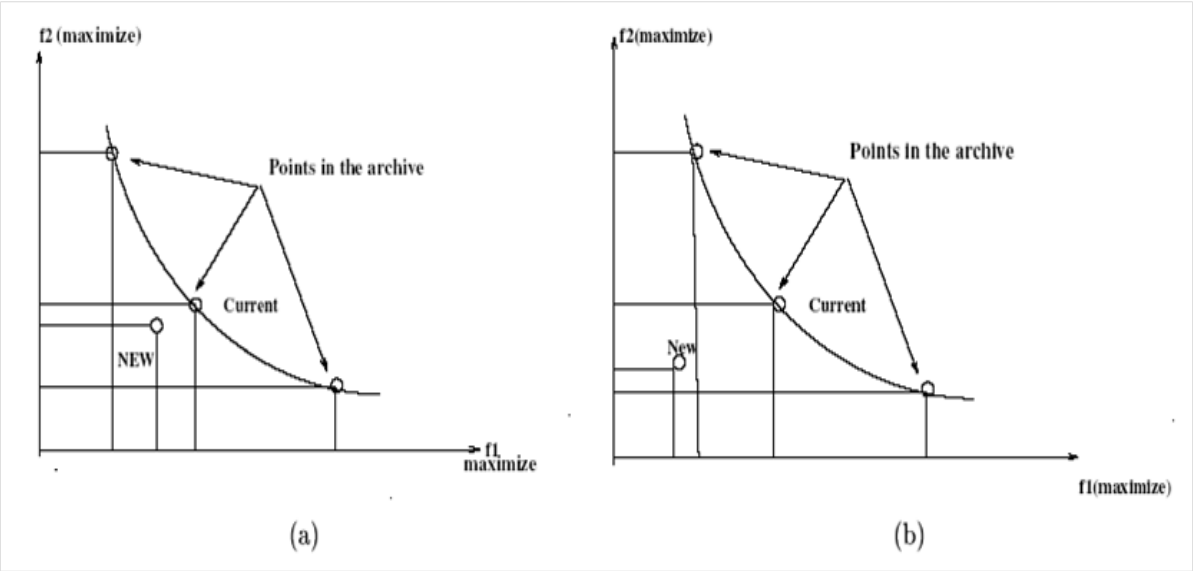
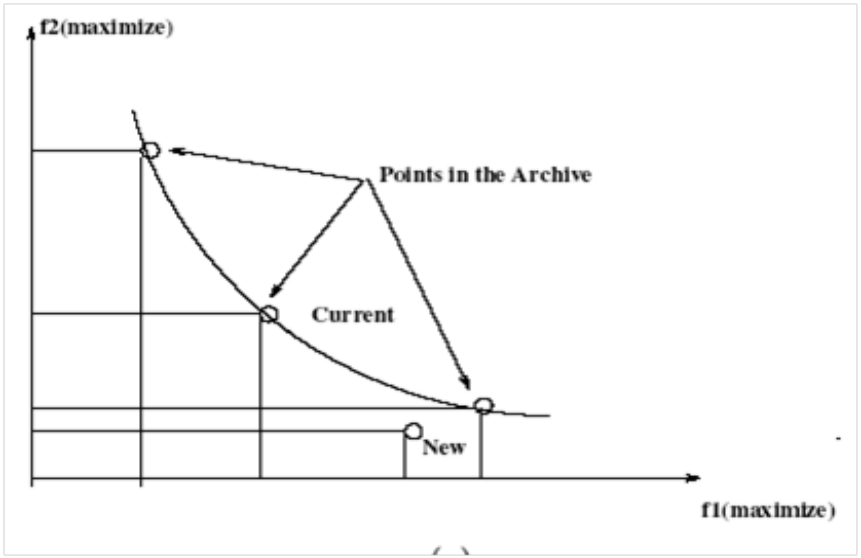


Figure 5 : (a): New is non-dominating w.r.t all solutions of archive (b) Some solutions in the archive dominates New

Case-2: current-pt is non-dominating w.r.t new-pt

Case 2(a)

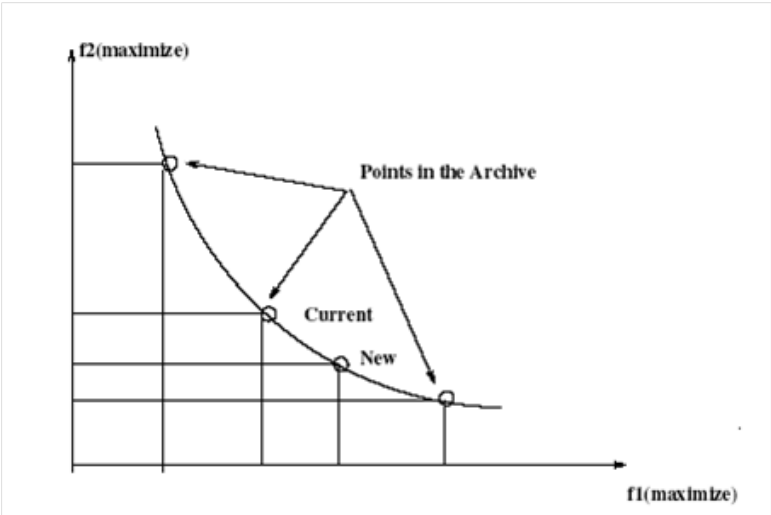
1. New-pt is dominated by h points in the archive ($h \geq 1$).
2. The new-pt is selected as the current-pt with $\text{prob} = \frac{1}{1+\exp(\Delta\text{dom}/h \times \text{temp})}$ where $\Delta dom = \sum_{i=1}^{h} (\Delta dom_{i,\text{new-pt}})$



Case-2: current-pt is non-dominating w.r.t new-pt

Case 2(b)

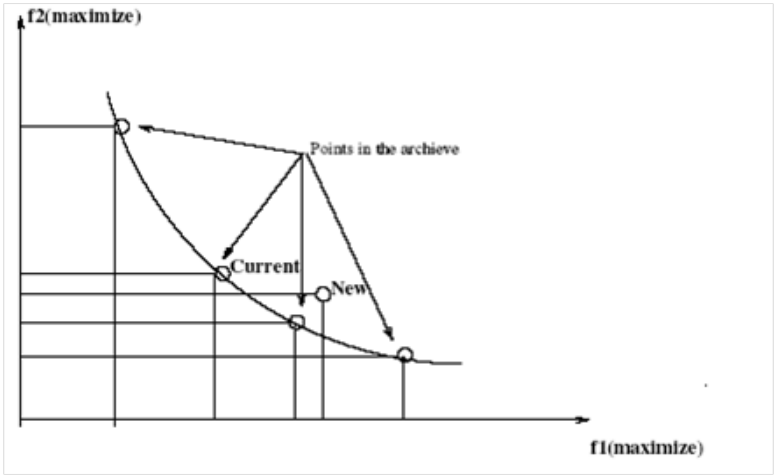
1. New-pt is non-dominating with respect to all the other points in the archive as well.
2. Here new-pt is selected as the current-pt and added to the archive.
3. In case archive becomes overfull (i.e., Soft-limit is exceeded) clustering is performed to reduce the number of points to Hard-limit.



Case-2: current-pt is non-dominating w.r.t new-pt

Case 2(c)

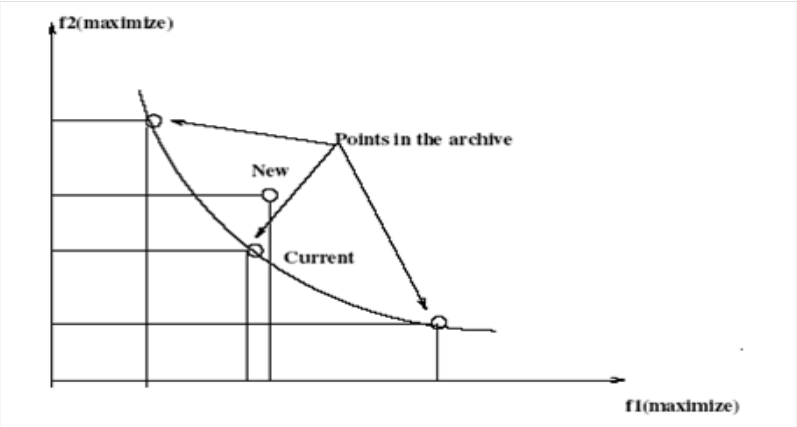
1. new-pt is non-dominating w.r.t current-pt but it dominates $k(k \geq 1)$ points of the archive.
2. Select the new-pt as the current-pt, and added to the archive
3. All the k dominated points are removed from the archive.
4. Here current-pt may or may not be on the archival front.



Case-3: new-pt dominates current-pt

Case 3(a)

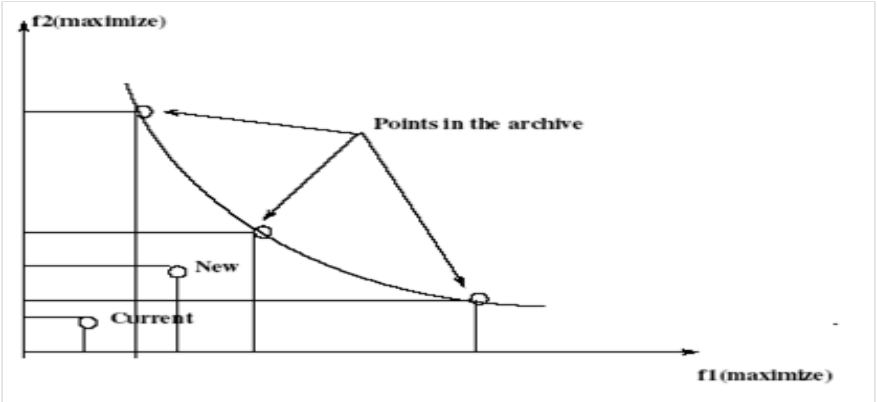
1. new-pt is non-dominating w.r.t. all points in the archive
2. The new-pt is added to the archive as it is considered to be a new non-dominated solution that must be stored in the archive.
3. If current-pt is in the archive then remove it.
4. If the number number of points become more than the Soft-limit clustering is performed to reduce the number of points to Hard-limit



Case-3: new-pt dominates current-pt

Case 3(b)

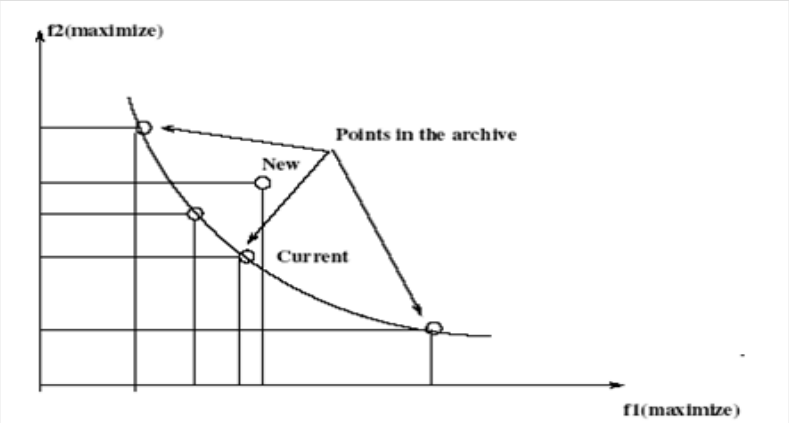
1. New-pt dominates the current-pt but $k(k \geq 1)$ points in the archive dominate this new-pt.
2. The minimum of the difference of domination amounts between the new-pt and k points denoted by $\Delta dom_{\min}$ of the archive is computed.
3. The point from the archive which corresponds to the minimum difference is selected as the current-pt with $prob = \frac{1}{1+\exp(-\Delta dom_{\min})}$
4. Otherwise the new-pt is selected as the current-pt.



Case-3: new-pt dominates current-pt

Case 3(c)

1. new-pt dominates not only the current-pt but also $k(k \geq= 1)$ other points in the archive.
2. New-pt is selected as the current-pt and added to the archive, while all the dominated points of the archive are removed.
3. Note that here current-pt may or may not be on the archival front.



Outline

Characteristics and advantages of our proposed algorithm

1. Concept of amount of domination is used to determine the acceptance probability
2. Clustering used to enforce diversity of solutions.
3. In AMOSA a new solution worse than the current solution may be selected.
   - In contrast to most other MOEA's where if a choice needs to be made between two solutions x and y and if x dominates y then x is always selected.
   - leads to reduced possibility of getting stuck at suboptimal regions.
     - Characteristic of single objective EAs or SAs
4. All MOEAs are so designed that this characteristics is lost.
5. The AMOSA algorithm provide a way of incorporating this feature.
   - Good performance for problems where other algorithms got stuck at local optima.

Performance Measures for Comparing MOO Techniques

In multiobjective optimization problem, there are two primary functionalities that an MOO strategy must achieve regarding the obtained solution set.

1. Converge as close to the true pareto optimal front as possible
   - ensures that the obtained solutions are near optimal
   - Convergence Measure γ, purity
2. Maintain as diverse a solution set as possible.
   - ensures that a wide range of trade-off solutions is obtained.
   - Spacing, Minimal Spacing

Simulation Result

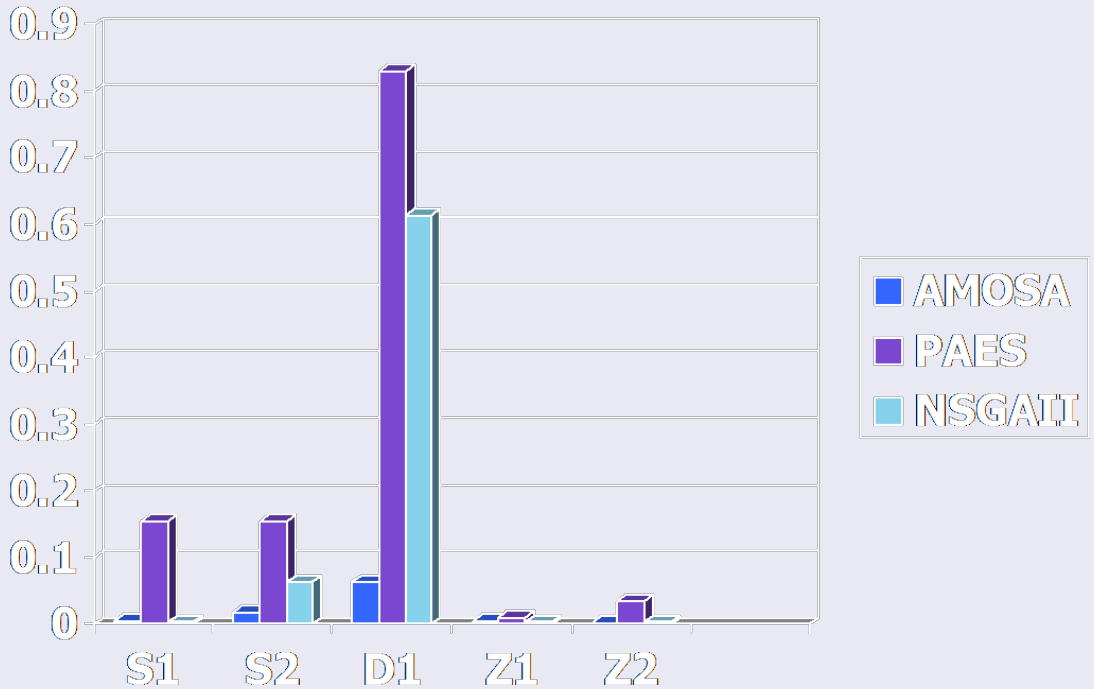
1. The performance of proposed AMOSA algorithm is compared with two existing well-known and widely used algorithms,
   - Pareto Achieved Evolutionary Strategy (PAES) proposed by Knowles and Corne .
   - Non-dominated Sorting Genetic AlgorithmII (NSGAII) proposed by Kalyanmoy Deb et. al.
2. Eight test problems are used: SCH1, SCH2, Dev1, Dev4, ZDT1, ZDT2, ZDT3, ZDT6.
3. The Population size in NSGAII is 30, archive size of PAES and AMOSA are also set to 30.
4. All the algorithms are executed 10 times per problem and results reported are the average values obtained for the ten runs.

$\alpha = 8,\ T_{\max} = 200,\ T_{\min} = 0.00001,$ iter $= 200$

Problem	n	*variable bounds*	*Objective functions*	*Optimal Solution*	*Comments*
$SCH1$	1	$[-10, 10]$	$f_1(x) = x^2$ $f_1(x) = (x-2)^2$	$x \in [0, 2]$	*convex*
$SCH2$	1	 $[-5, 10]$	$f_1(x) = -x$ if $x \leq 1$ $f_1(x) = x - 2$ if $1 < x \leq 3$, $f_1(x) = 4 - x$ if $3 < x \leq 4$, $f_1(x) = x - 4$ if $x > 4$, $f_2(x) = (x-5)^2$	$x \in [1, 2] \cup [4, 5]$	*convex, disconnected*
$Dev1$	2	$x_1 \in [0.1, 1.0]$ $x_2 \in [0, 1.0]$	$f_1(x) = x_1$ $g_1(x) = 2.0 - exp\{-(\frac{x_2-0.2}{0.004})^2\}$ $f_2(x) = (g_1(x) - 0.8exp\{-(\frac{x_2-0.6}{0.4})^2\})/x_1$	$x_1 \in [0, 1]$ $x_2 = 0.2$	*convex*
$Dev4$	2	$x_1 \in [0, 1]$ $x_2 \in [0, 1]$	$f_1(x_1) = x_1$; $g(x_2) = 1 + 10x_2$ $h(f_1, g) = 1 - (\frac{f-1}{g})^\alpha - \frac{f_1}{g} sin(2\pi q f_1)$ $f_2 = h(f_1, g)g(x2)$		*convex, disconnected*
$ZDT1$	30	$[0, 1]$	$f_1(x) = x_1$ $f_2(x) = g(x)[1 - \sqrt{x_1/g(x)}]$ $g(x) = 1 + 9(\sum_{i=2}^{n} x_i)/(n-1)$	$x_1 \in [0, 1]$ $x_i = 0$ $i = 2, ...n$	*convex*
$ZDT2$	30	$[0, 1]$	$f_(x) = x_1$ $f_2(x) = g(x)[1 - (x_1/g(x))^2]$ $g(x) = 1 + 9(\sum_{i=2}^{n} x_i)/(n-1)$	$x_1 \in [0, 1]$ $x_i = 0,$ $i = 2, ...n$	*nonconvex*
$ZDT3$	30	$[0, 1]$	$f_(x) = x_1$ $f_2(x) = g(x)[1 - \sqrt{x_1/g(x)} - \frac{x_1}{g(x)} * sin(10\pi x_1]$ $g(x) = 1 + 9(\sum_{i=2}^{n} x_i)/(n-1)$	$x_1 \in [0, 1]$ $x_i = 0,$ $i = 2, ...n$	*convex disconnected*
$ZDT6$	10	$[0, 1]$	$f_(x) = 1 - exp(-4x_1)sin^6(4\pi x_1)$ $f_2(x) = g(x)[1 - (f_1(x)/g(x))^2]$ $g(x) = 1 + 9[(\sum_{i=2}^{n} x_i)/(n-1)]^{0.25}$	$x_1 \in [0, 1]$ $x_i = 0$ $i = 2, ...n$	*nonconvex, non-uniformly spaced*

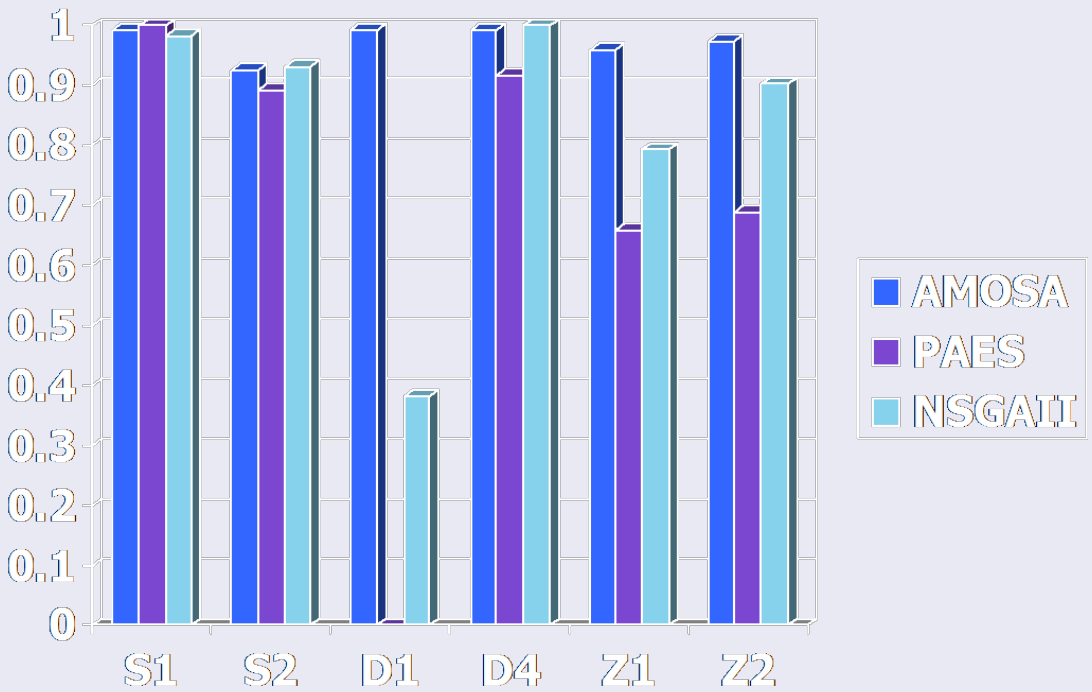
Convergence Measure

On the Problems SCH1, SCH2, Dev1, ZDT1, ZDT2



Purity Measurements

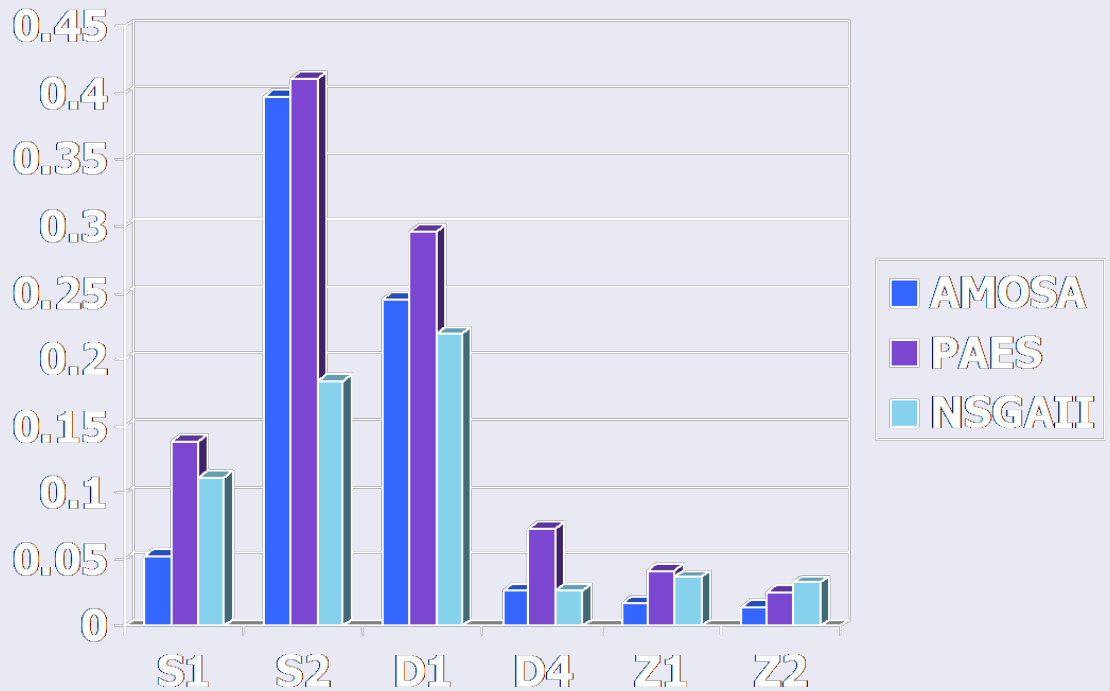
On the Problems SCH1, SCH2, Dev1, Dev4, ZDT1, ZDT2



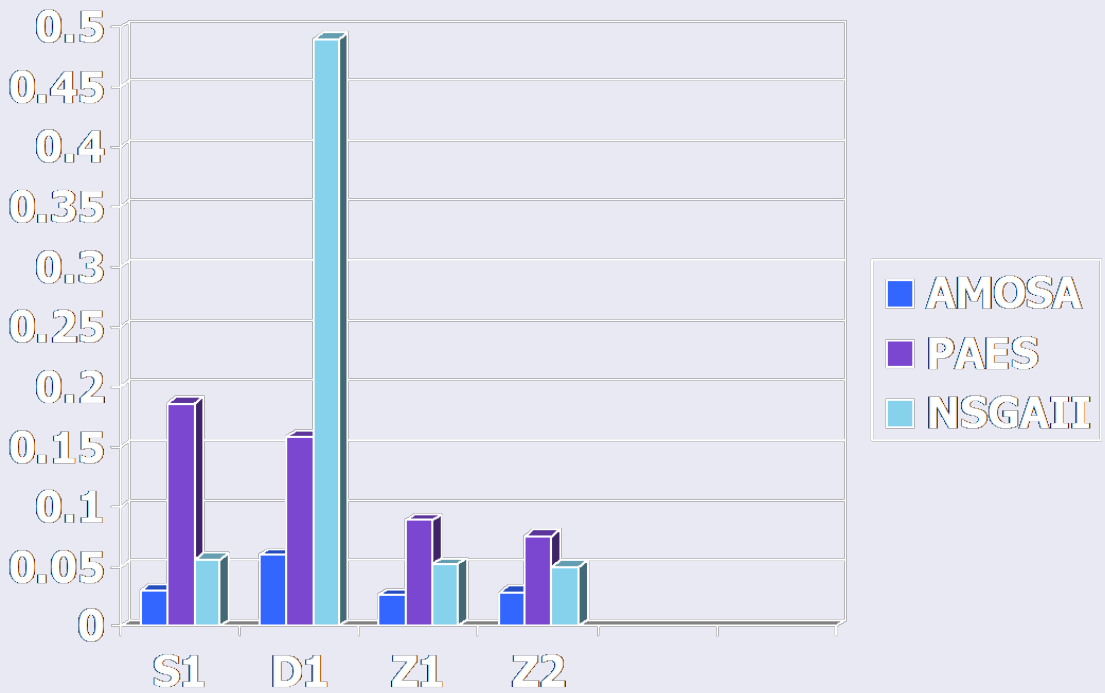
Note: Number of distinct solutions was always more for AMOSA

Spacing Measurements

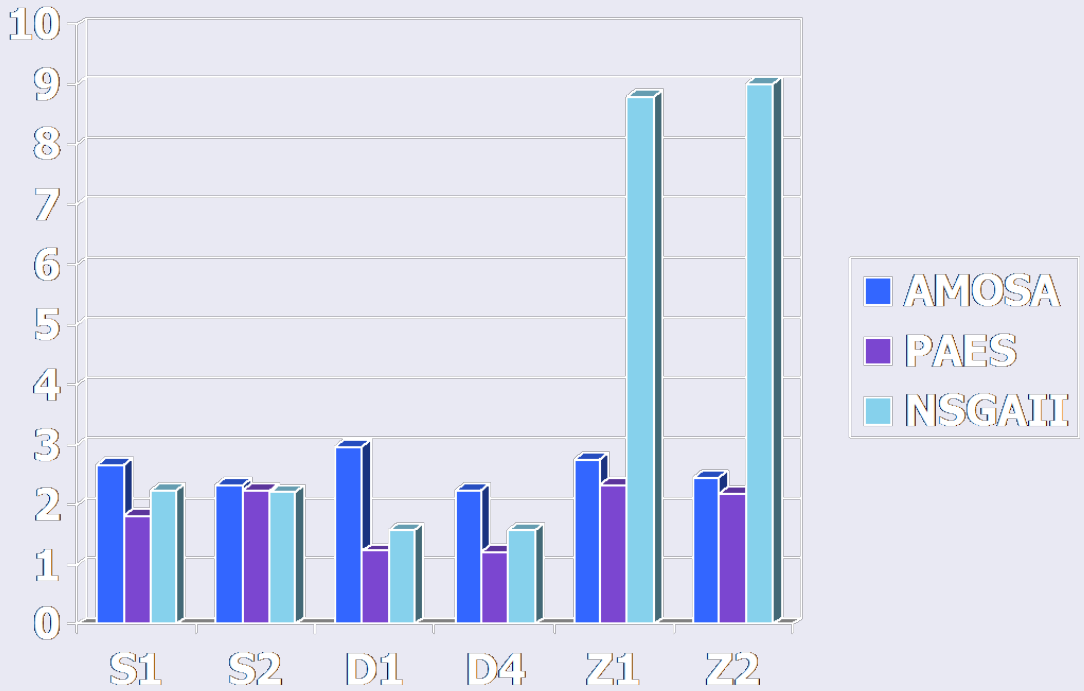
On the Test Problems SCH1, SCH2, Dev1, Dev4, ZDT1, ZDT2



Minimal Spacing on the Test Problems SCH1, Dev1, ZDT1, ZDT2



Timing Measurement on the Test Problems SCH1, SCH2, Dev1, Dev4, ZDT1, ZDT2



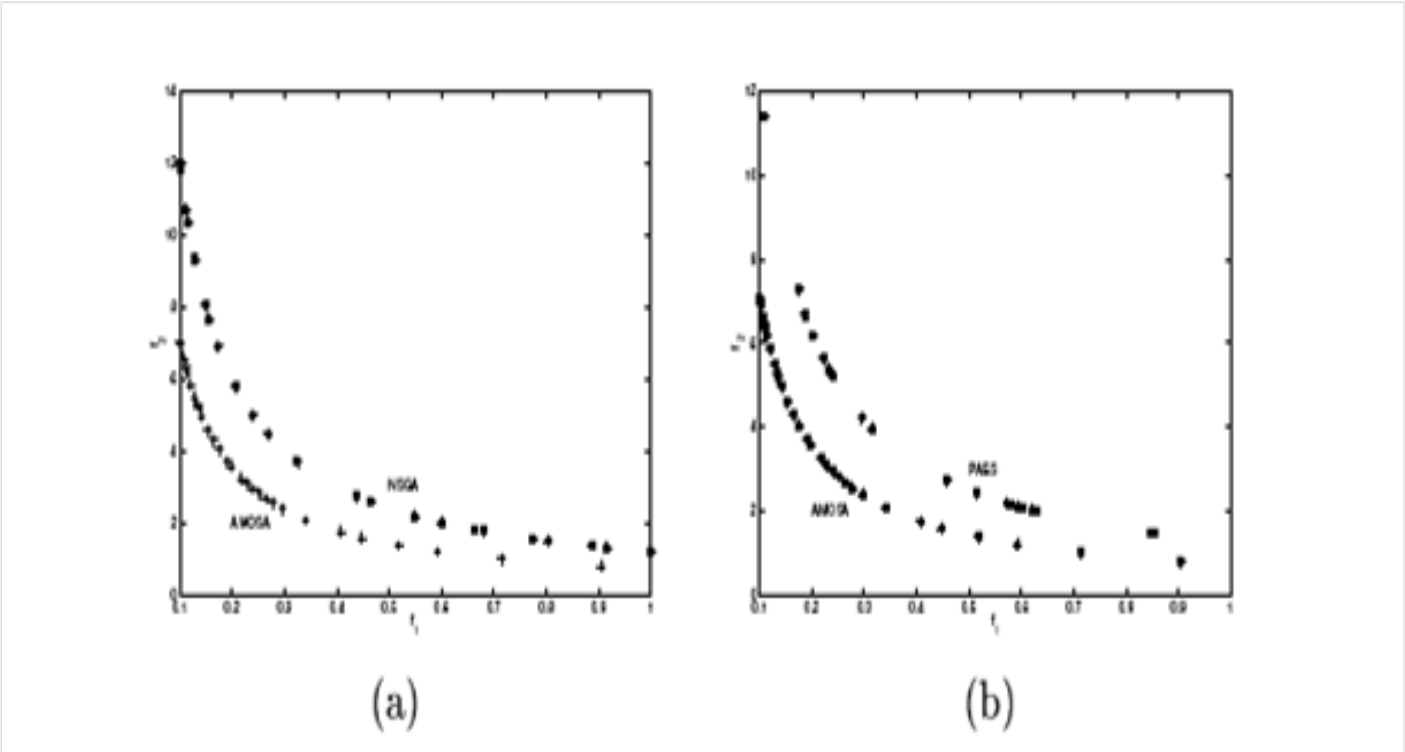
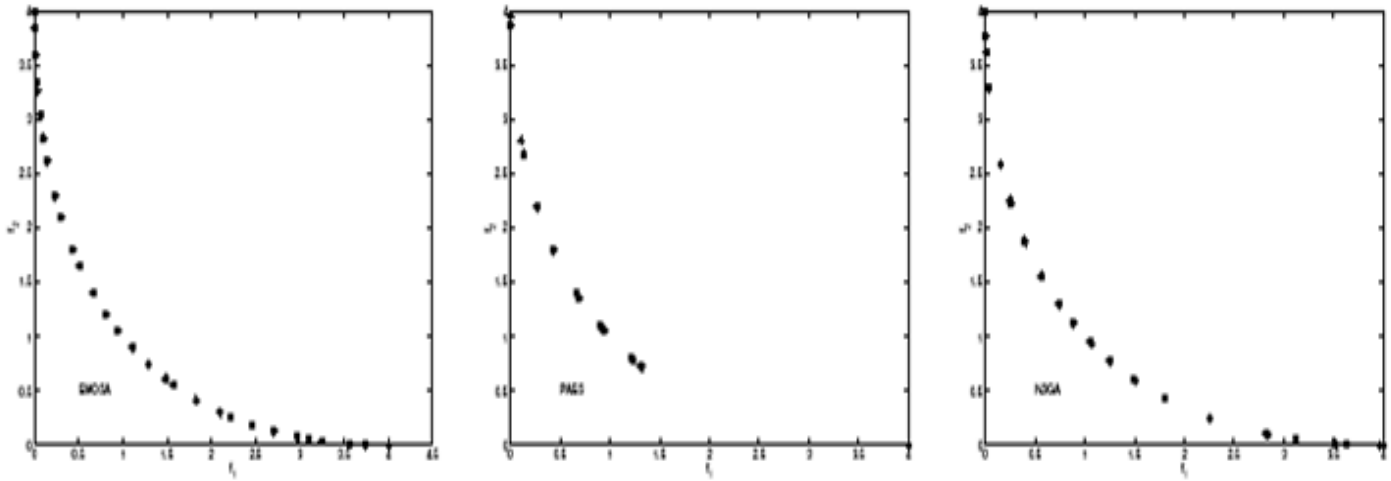


Figure 6 : (a)The final non-dominated front for Dev1 obtained from AMOSA and for NSGAII-II (b) The final non-dominated front for Dev1 obtained from AMOSA and for PAES

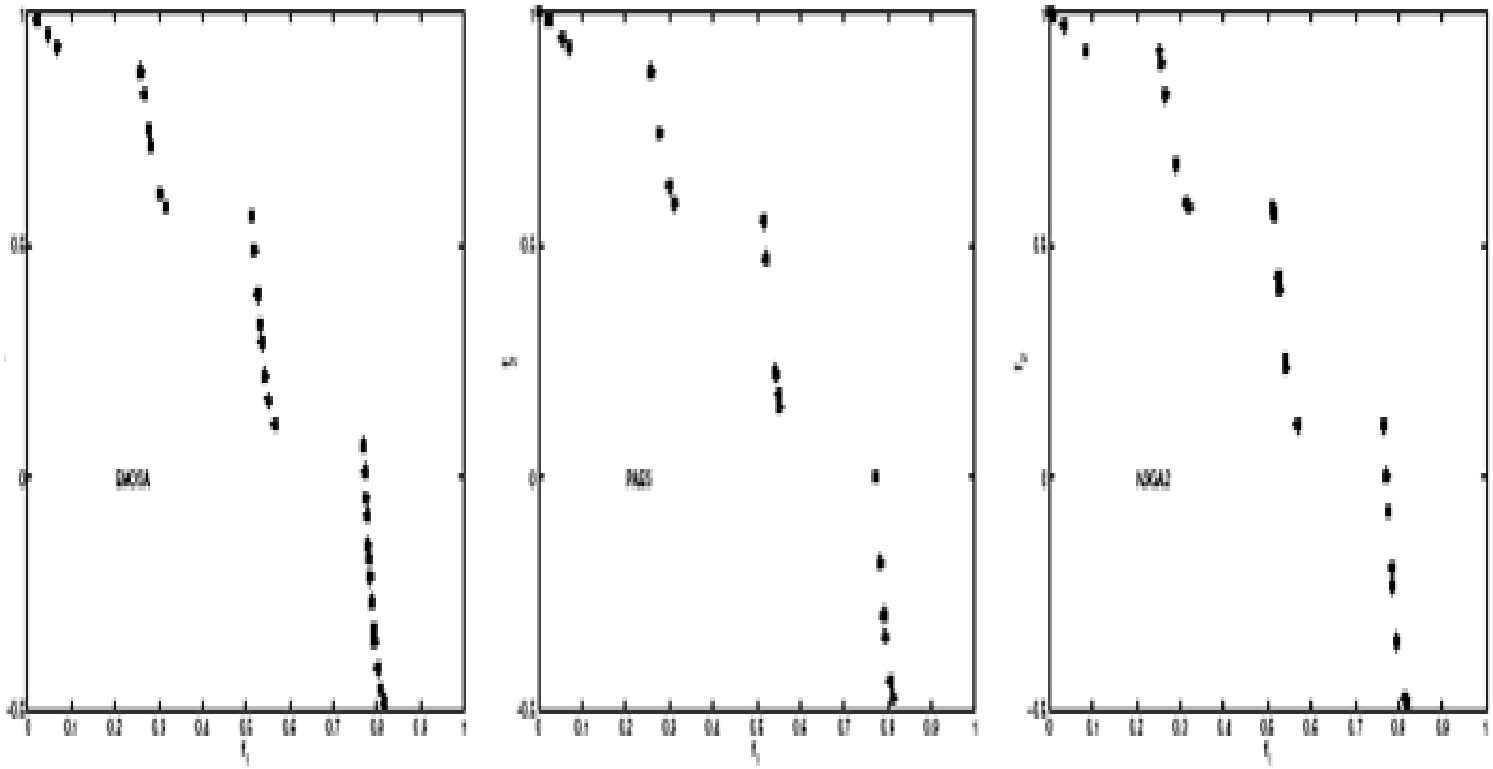
Final Solutions for SCH1



(a) (b) (c)

Figure 7.1: The final non-dominated solutions for SCH1 obtained by (a) AMOSA, (b) PAES and (c) NSGAII.

Final Solutions for Dev4



Discussion about Results

1. Performance of our proposed AMOSA is comparable with, often better than that of NSGAII, and much better than that of PAES.
2. PAES performs best in terms of time .
3. For 30 variable/ 10 variable problem such as ZDT1, ZDT2,. . .,ZDT6, AMOSA took much less time than that of NSGAII.
4. But for 2 or single variable problems SCH1, SCH2, Dev1, Dev4 AMOSA takes more time than NSGAII.
   - It is due to complexity of its clustering procedure
   - For single/two variable problem complexity of clustering procedure dominates the ranking and crossover operations of NSGA-II but in 30/10 variable problem scenario is reversed.
     - It is due to large chromosome length in 30/10 variable problem .

Some Real Life Applications of AMOSA

AMOSA is a tool for multiobjective optimization, and hence will be useful in several real-life domains:

1. Clustering (to be explained in subsequent slides) and Image Segmentation
2. Computational Biology
   - Drug designing (design a small molecule that can bind to the active side of the target protein so that several energy factors should be minimized).
   - Clustering Microarray Data
3. VLSI K way equi-partitioning of a data set (in progress)
4. E-commerce
   - Product recommendation (several desirable, often complementary, characteristics should be optimized together).

Discussion and Conclusions

- Clustering procedure in AMOSA is the most time consuming procedure-some other more efficient methods have to be incorporated to improve the performance.
- Theoretical analysis of the algorithm should be done.
- Allowing bad solutions with a probability is an interesting feature of this algorithm-this needs to be incorporated in other MOO algorithms in order to improve the performance.
- Extend multiobjective clustering method for some cluster validity indices which are more complementary in nature.
- Extend the multiobjective clustering method to automatically detect the number of clusters from a data set.
- Try different encoding to represent clusters.
- Extend to constrained optimization

Important References I


K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: Nsga-ii," *Evolutionary Computation, IEEE Transactions on*, vol. 6, no. 2, pp. 182–197, 2002.


N. Srinivasan and K. Deb, "Multi-objective function optimisation using non-dominated sorting genetic algorithm," *Evolutionary Comp*, vol. 2, no. 3, pp. 221–248, 1994.


C. M. Fonseca and P. J. Fleming, "Multiobjective optimization and multiple constraint handling with evolutionary algorithms. i. a unified formulation," *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, vol. 28, no. 1, pp. 26–37, 1998.


K. Deb and D. E. Goldberg, "An investigation of niche and species formation in genetic function optimization," in *Proceedings of the 3rd International Conference on Genetic Algorithms*, pp. 42–50, Morgan Kaufmann Publishers Inc., 1989.

Thank you!