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

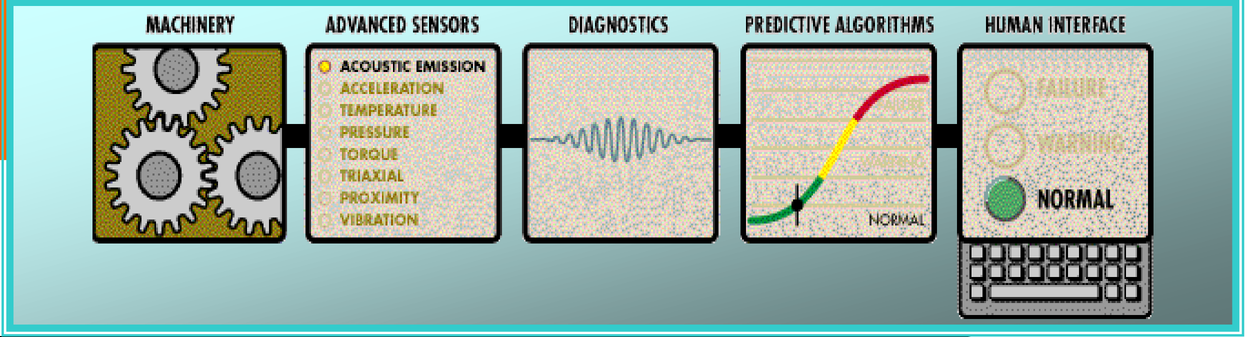
Sichuan University, Chengdu, CHINA

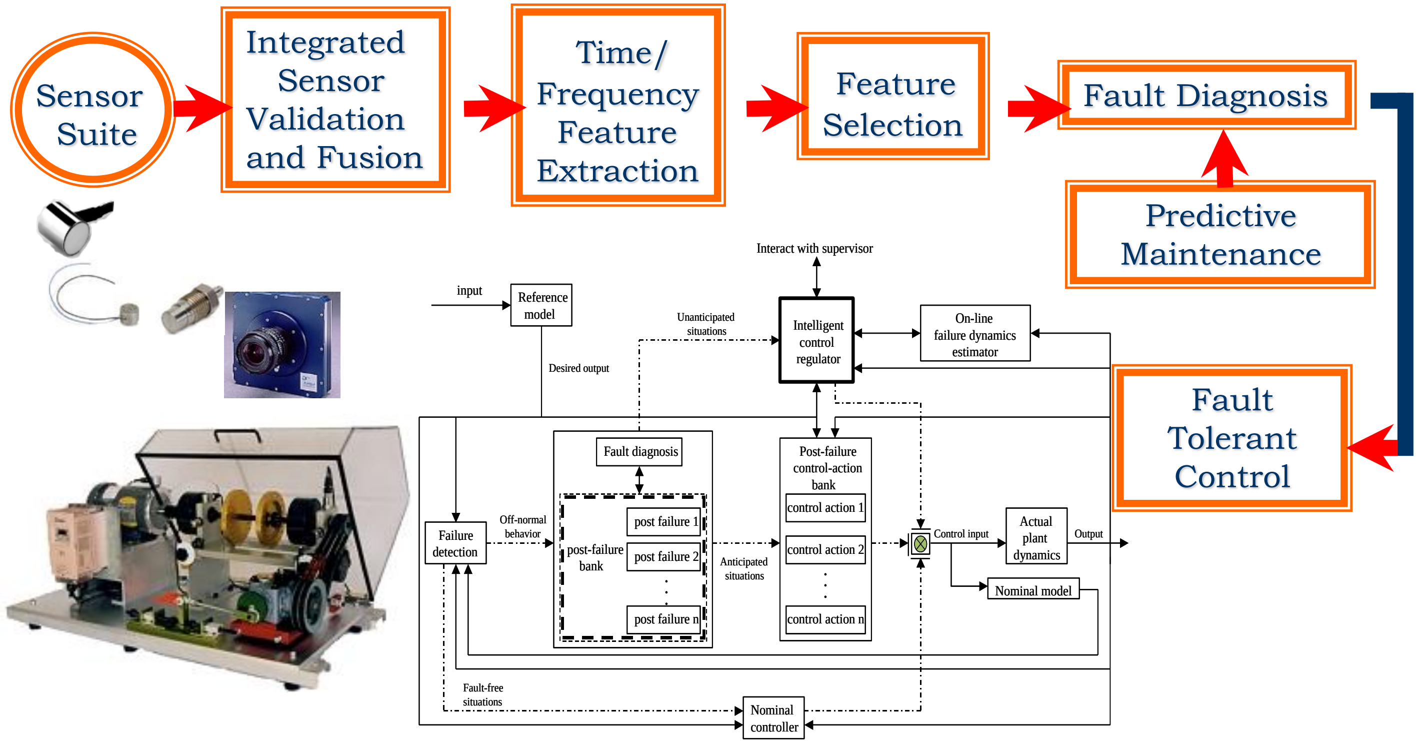
Day Five of EIGHT, July 4, 2022

Case Study 5: Conditional Health Monitoring

- **Motivation:** This research calls for the design of an on-board intelligent health assessment system using multi-channel vibration accelerometers. The system developed is capable of detecting, identifying and accommodating the gradual material degradation and catastrophic component failures of structures in an adverse operating environment.

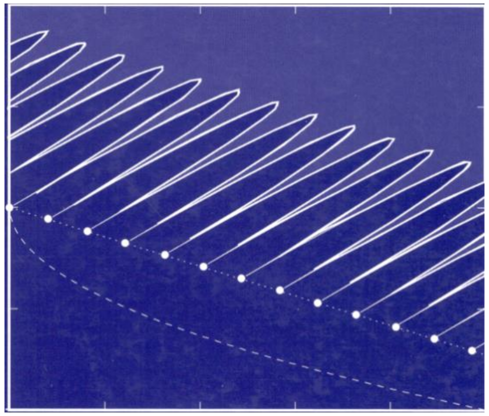
- **Approach:** Vibration signatures are extracted using wavelet packet transform and Fisher's criteria. A hybrid network with an on-line real-time adaptive critics learning is developed to perform expert advising. A hierarchical fault diagnosis architecture is advocated to fulfill the time-critical and on-board needs in different levels of structural integrity over a global working envelope. The potential of spin-off applications on aeropropulsion engine, on-orbit satellite, and reusable launch vehicle is imminent.


MACHINERY
ADVANCED SENSORS
ACOUSTIC EMISSION
ACCELERATION
TEMPERATURE
PRESSURE
TORQUE
TRIAXIAL
PROXIMITY
VIBRATION
DIAGNOSTICS
PREDICTIVE ALGORITHMS
NORMAL
HUMAN INTERFACE
FAILURE
WARNING
NORMAL


Sensor Suite
Integrated Sensor Validation and Fusion
Time/ Frequency Feature Extraction
Feature Selection
Fault Diagnosis
Predictive Maintenance
Fault Tolerant Control
Interact with supervisor
input
Reference model
Desired output
Unanticipated situations
Intelligent control regulator
On-line failure dynamics estimator
Fault diagnosis
Post-failure control-action bank
control action 1
control action 2
control action n
post-failure bank
post failure 1
post failure 2
post failure n
Failure detection
Off-normal behavior
Anticipated situations
Control input
Actual plant dynamics
Output
Nominal model
Fault-free situations
Nominal controller

7. MULTIOBJECTIVE OPTIMIZATION EVOLUTIONARY ALGORITHMS- Part I

多目标优化进化算法

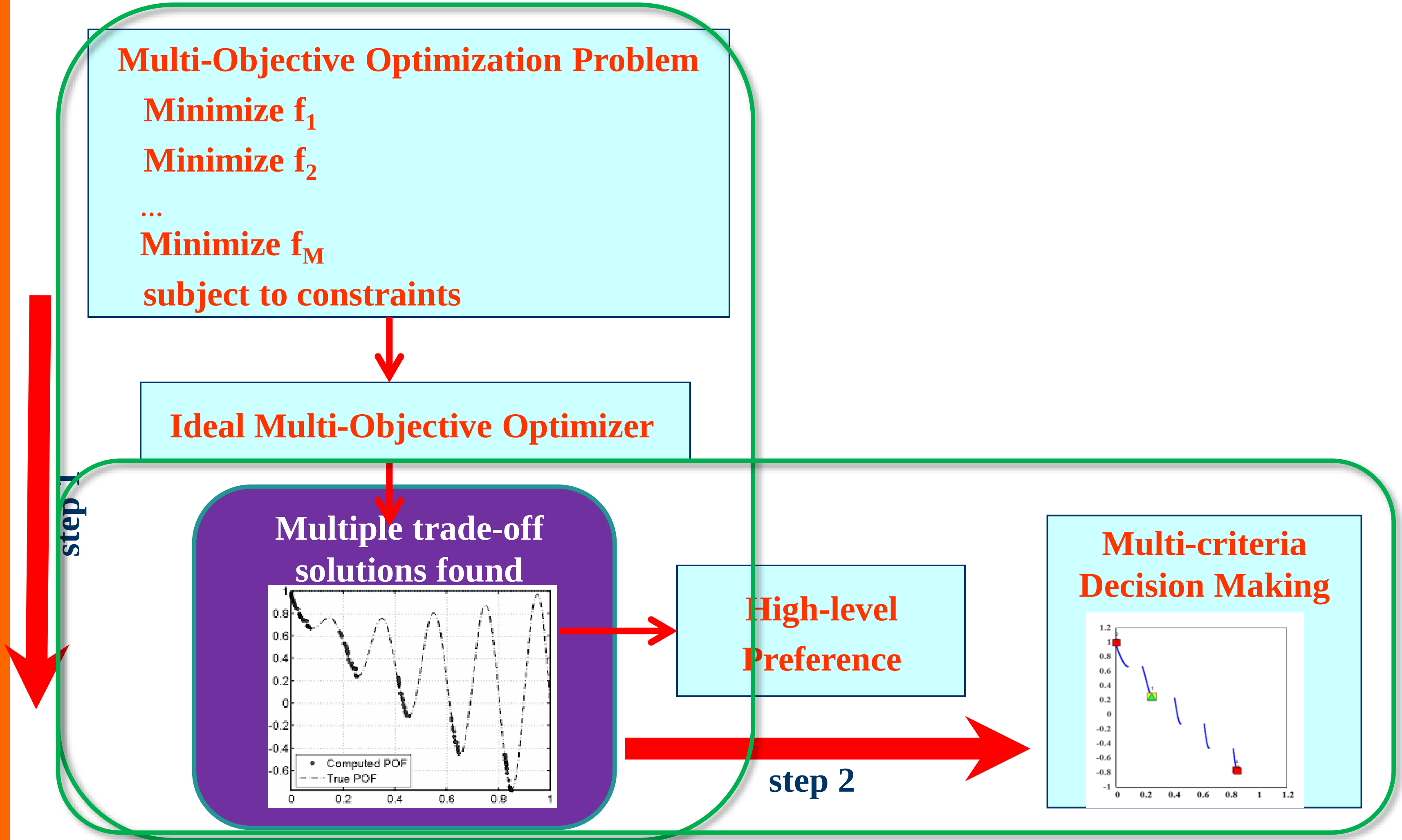


Motivation for Pareto-Optimal Solutions

- If the exact trade off among objective solutions is known, a weight based classical method will be good enough to find the corresponding solution.

- But a user is usually not sure of exact trade off relationship among objectives. In this case it is better to find a set of Pareto-optimal solutions first and then choose one solution from the set by using some other higher-level information or consideration.

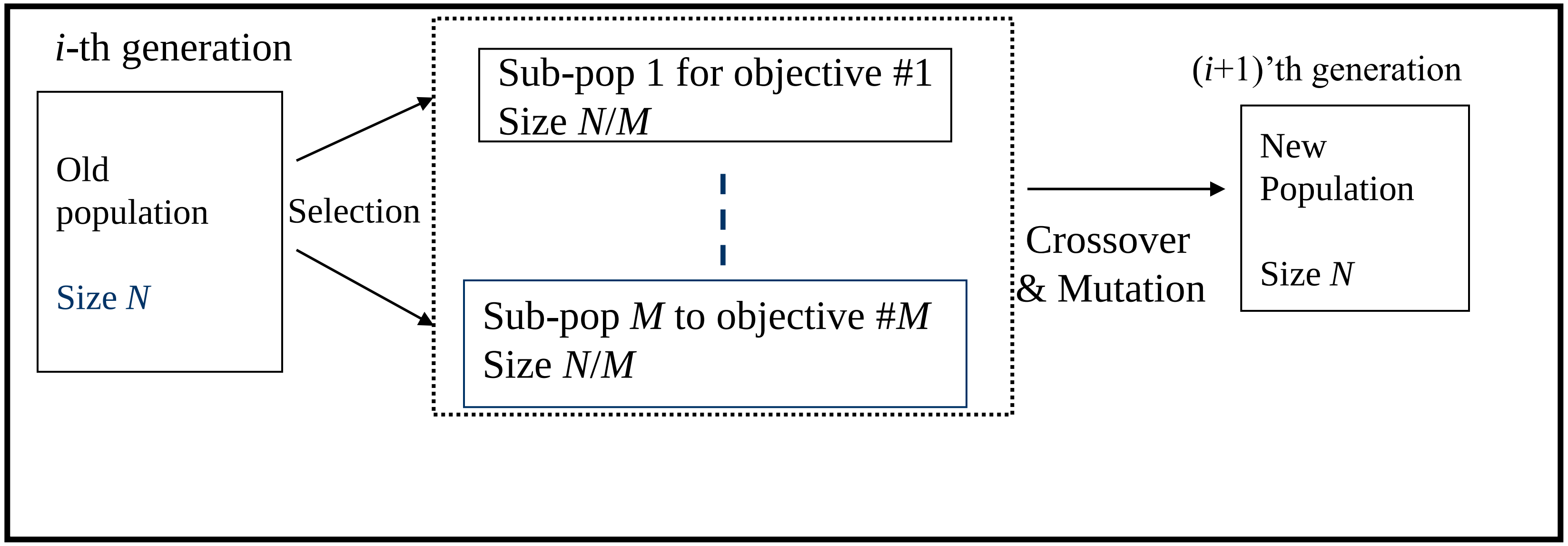
Multi-criteria Decision Making


Multi-Objective Optimization Problem
Minimize f_1
Minimize f_2
…
Minimize f_M
subject to constraints
Ideal Multi-Objective Optimizer
step 1
Multiple trade-off solutions found
Computed POF
True POF
High-level Preference
step 2
Multi-criteria Decision Making

- After a non-dominated set of solutions are found, each solution can be associated with an artificial weight vector estimated from the location of the solutions in the non-dominated set.
- This method gives a user an overall perspective of the other possible optimal solutions that the underlying MOP offers before choosing a solution and also allows to choose a solution according to some desired degree of importance to some objective.

1- Vector Evaluated GA (VEGA)

- Simplest possible multi-objective GA.
- Since a number of objectives (say *M*) have to be handled, Schafer divided the GA population into *M* equal subpopulations randomly.
- Each subpopulation is assigned a fitness based on a different objective function.
- Each of the *M* objectives is used to evaluate some members in the population.



- Divide population into *M* equal blocks for *M* objectives at every generation
- Each block is reproduced with one objective function
- Fitness proportionate selection operator is used
- Complete population participates in crossover and mutation
- In order to reduce the *positional bias* in the population, it is better to shuffle the population before it is partitioned.

VEGA Pseudo Codes

Step1: Set an objective function counter *i* = 1 and define *q* = *N*/*M*

Step 2: For all solutions *j* = 1+(*i*-1) * *q* to *j* = *i* **q*, assign fitness as

$$\boxed{F(x^{(j)}) = f_i(x^{(j)})}$$

Step 3: Perform fitness proportionate selection on all *q* solutions to create a mating pool P_i

Step 4: If *i* = *M*, go to Step 5. Otherwise, increment *i* by one and go to Step 2

Step 5: Combine all mating pools together: $\boxed{P = \bigcup_{i=1}^{M} P_i}$

Perform crossover and mutation on *P* to create a new population

VEGA Example

- Consider the problem

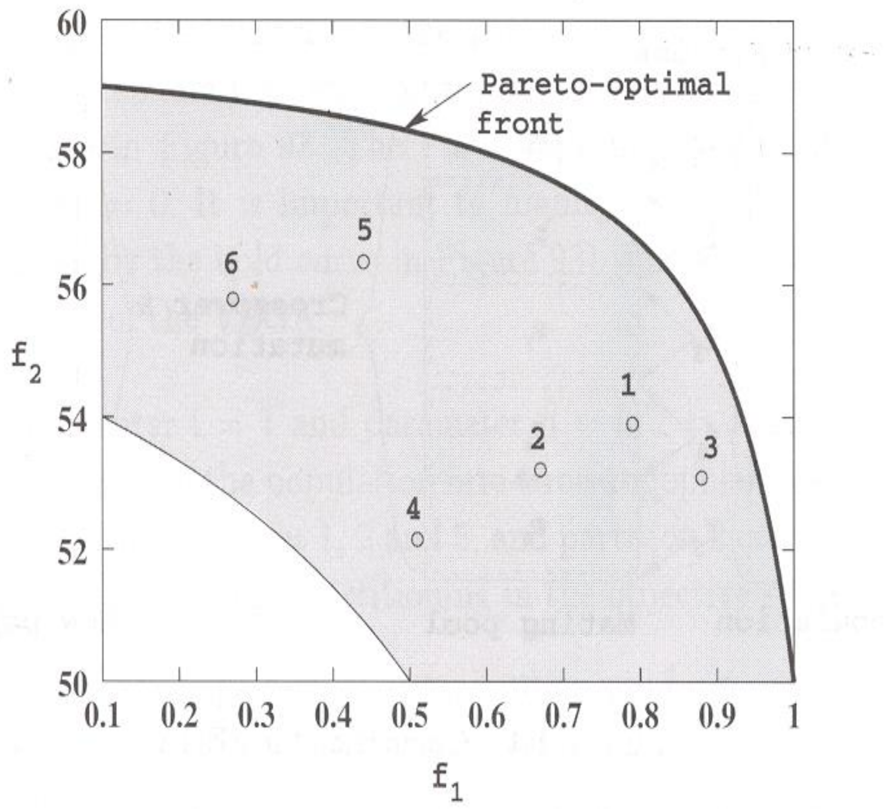
Maximize $f_1(x) = 1.1 - x_1$

Maximize $f_2(x) = 60 - \frac{1 + x_2}{x_1}$

Subject to

$0.1 \le x_1 \le 1,$

$0 \le x_2 \le 5$



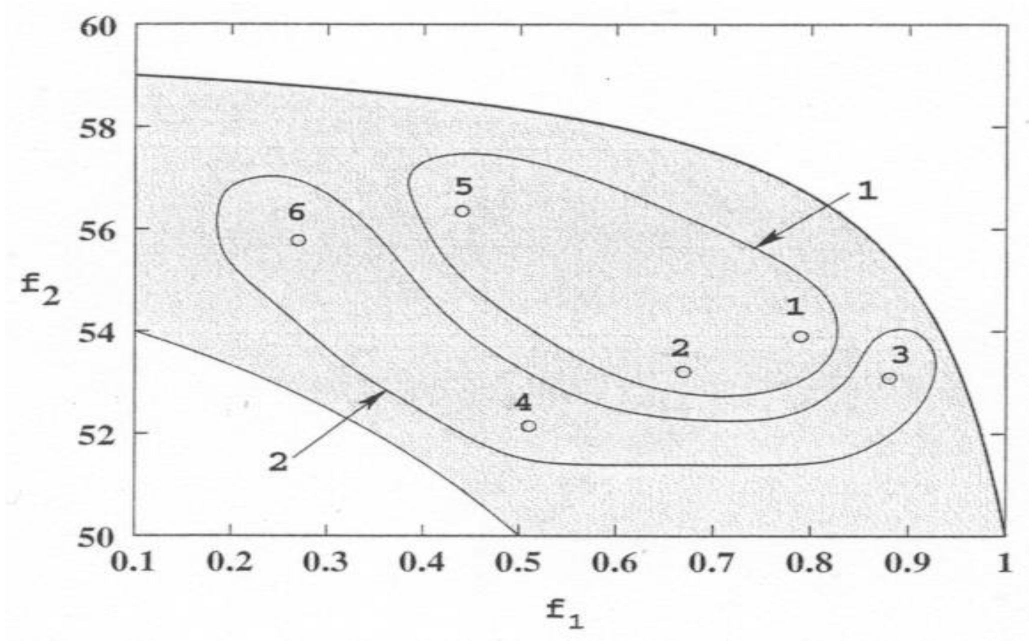
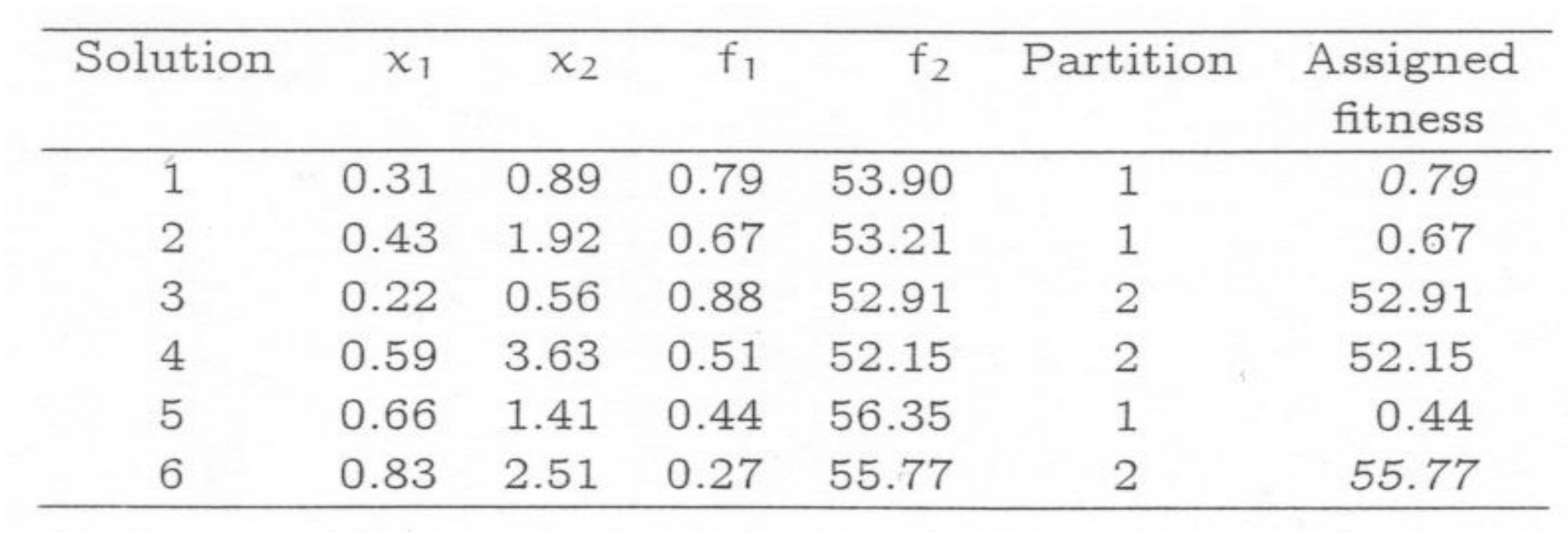
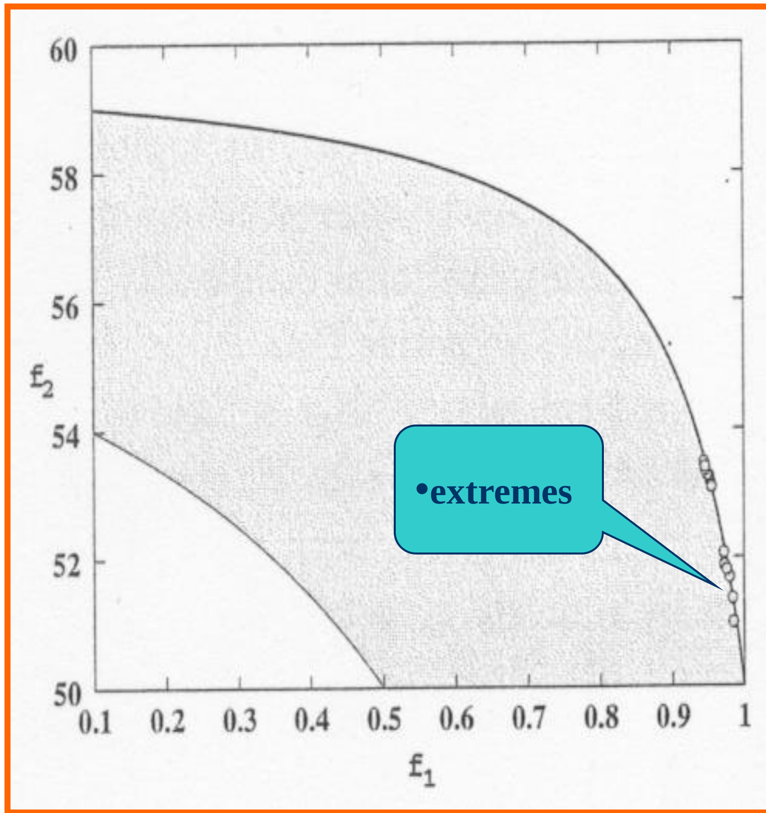


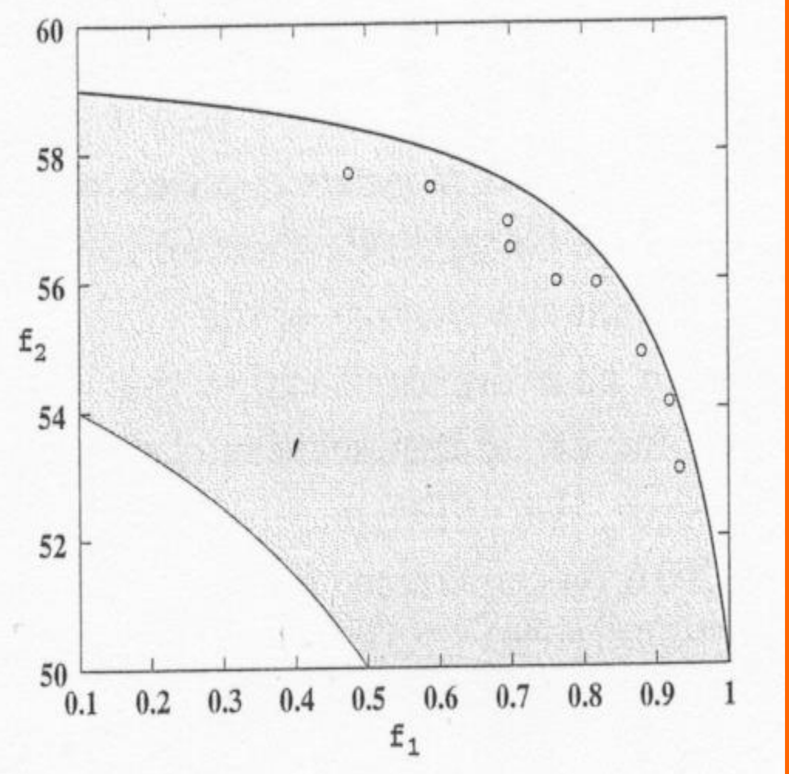
Figure above shows 2 subpopulations, one for each objective function (i.e., q = 6/2 = 3)



Solution	x_1	x_2	f_1	f_2	Partition	Assigned fitness
1	0.31	0.89	0.79	53.90	1	*0.79*
2	0.43	1.92	0.67	53.21	1	0.67
3	0.22	0.56	0.88	52.91	2	52.91
4	0.59	3.63	0.51	52.15	2	52.15
5	0.66	1.41	0.44	56.35	1	0.44
6	0.83	2.51	0.27	55.77	2	*55.77*

Table above shows fitness assignment for individuals in each subpopulation





VEGA without mutation

VEGA with mutation

VEGA Advantages & Disadvantages

- Advantages
  - Simple idea and easy to implement.
  - Only simple changes need to be made to the GA to convert it to a Multiobjective GA.
- Disadvantages
  - Solutions near the optimum of an individual objective function would be preferred by the selection operator in a subpopulation.
  - Crossover between individual champion solutions does not necessarily find diverse solutions in the population.
  - Eventually VEGA converges to individual champion solutions only. A non-dominated selection: non-dominated solutions are assigned more copies
- Computational complexity: O(N)

2- Weight Based GA (WBGA)

- Each objective function f_i is multiplied by a weight w_i.
- A GA string represents all decision variables x_i as well as their associated weights w_i.
- The weighted objective function values are then added together to calculate the fitness of a solution.
- Each individual of the population is assigned a different weight vector, thereby multiple Pareto-optimal solutions are found in a single run.
- Key issue is to maintain diversity in weight vectors among the population members.

WBGA Pseudo Codes

Step 1: For each objective function *j*, set upper and lower bounds as $f_j^{\max}$ and $f_j^{\min}$

Step 2: For each solution *i* = 1, …, *N*, calculate the distance $d_{ik} = \left| x_w^{(i)} - x_w^{(k)} \right|$ with all solutions *k* = 1, …,*N*. Then calculate the sharing function value as

$$Sh(d_{ik}) = \begin{cases} 1 - \dfrac{d_{ik}}{\sigma_{\text{share}}}, \text{if } d_{ik} \le \sigma_{\text{share}} \\ 0, \quad \text{otherwise} \end{cases}$$

Thereafter, calculate the niche count of the solution *i* as $nc_i = \sum_{k=1}^{N} Sh(d_{ik})$

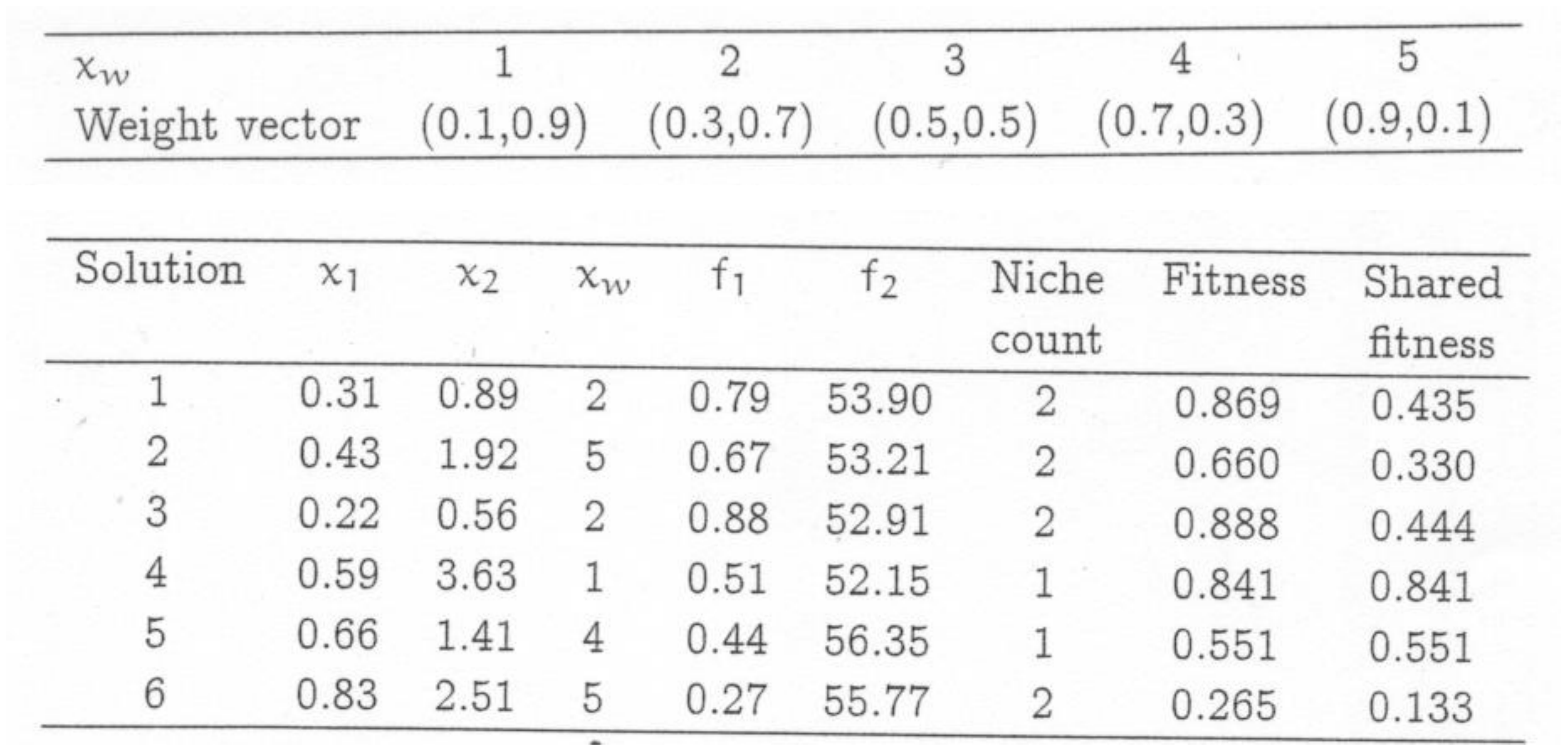
Step 3: For each solution *i* = 1,…,*N*, follow the entire procedure below.

Corresponding to the $x_w^{(i)}$ value, identify the weight vector $w^{(i)}$ from the user defined mapping between the integer variable $x_w^{(i)}$ and the weight vector $w^{(i)}$ Assign fitness F_i according to

$$F(x^{(i)}) = \sum_{j=1}^{M} w_j^{x_W^{(i)}} \frac{f_j(x^{(i)}) - f_j^{\min}}{f_j^{\max} - f_j^{\min}}$$

Calculate the *shared fitness* as $F_i' = F_i / nc_i$ for each individual. Then proportionate selection is applied to create the mating pool. Thereafter crossover and mutation operator are applied on the entire string.

WBGA Example



x_w	1	2	3	4	5
Weight vector	(0.1,0.9)	(0.3,0.7)	(0.5,0.5)	(0.7,0.3)	(0.9,0.1)

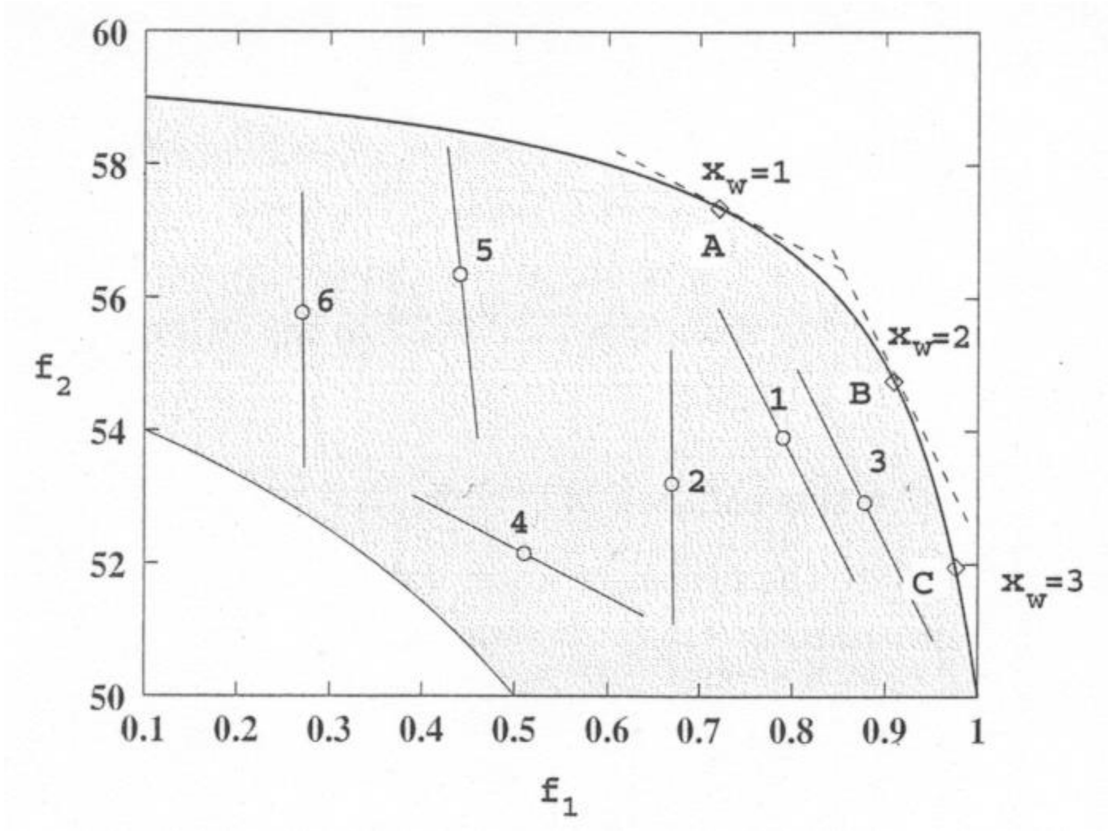
Solution	x_1	x_2	x_w	f_1	f_2	Niche count	Fitness	Shared fitness
1	0.31	0.89	2	0.79	53.90	2	0.869	0.435
2	0.43	1.92	5	0.67	53.21	2	0.660	0.330
3	0.22	0.56	2	0.88	52.91	2	0.888	0.444
4	0.59	3.63	1	0.51	52.15	1	0.841	0.841
5	0.66	1.41	4	0.44	56.35	1	0.551	0.551
6	0.83	2.51	5	0.27	55.77	2	0.265	0.133

- Step 1: Observing the lower and upper bounds, set $f_1^{max}=1.0$, $f_1^{min}=0.1$, $f_2^{max}=59.0$ and $f_2^{min}=0.0$
- Step 2: Calculate the niche count of each of the six solutions, using the x_w variables, $d_{11}=0$, $d_{12}=|2-5|=3$, $d_{13}=0$, $d_{14}=1$, $d_{15}=2$, $d_{16}=3$. Use $\sigma_{share}=1$ and calculate the sharing function values as $Sh(d_{11})=1$, $Sh(d_{12})=0$, $Sh(d_{13})=1$, $Sh(d_{14})=0$, $Sh(d_{15})=0$, $Sh(d_{16})=0$. The niche count of the first solution is the sum of the above sharing function values, $nc_1=2$. Similarly, find the niche counts of the other solutions, $nc_2=2$, $nc_3=2$, $nc_4=1$, $nc_5=1$, $nc_6=2$.
- Step 3: Assign the fitness to each solution. First calculate the fitness of the first solution. Variable x_w takes the value 2 for this solution. The corresponding weight vector is (0.3, 0.7). Thus

$$F1 = \omega_1^{(1)} \frac{f_1^{(1)} - f_1^{min}}{f_1^{max} - f_1^{min}} + \omega_2^{(1)} \frac{f_2^{(1)} - f_2^{min}}{f_2^{max} - f_2^{min}}$$

$$= 0.3\frac{0.79 - 0.1}{1 - 0.1} + 0.7\frac{53.90 - 0.0}{59.0 - 0.0} = 0.869$$

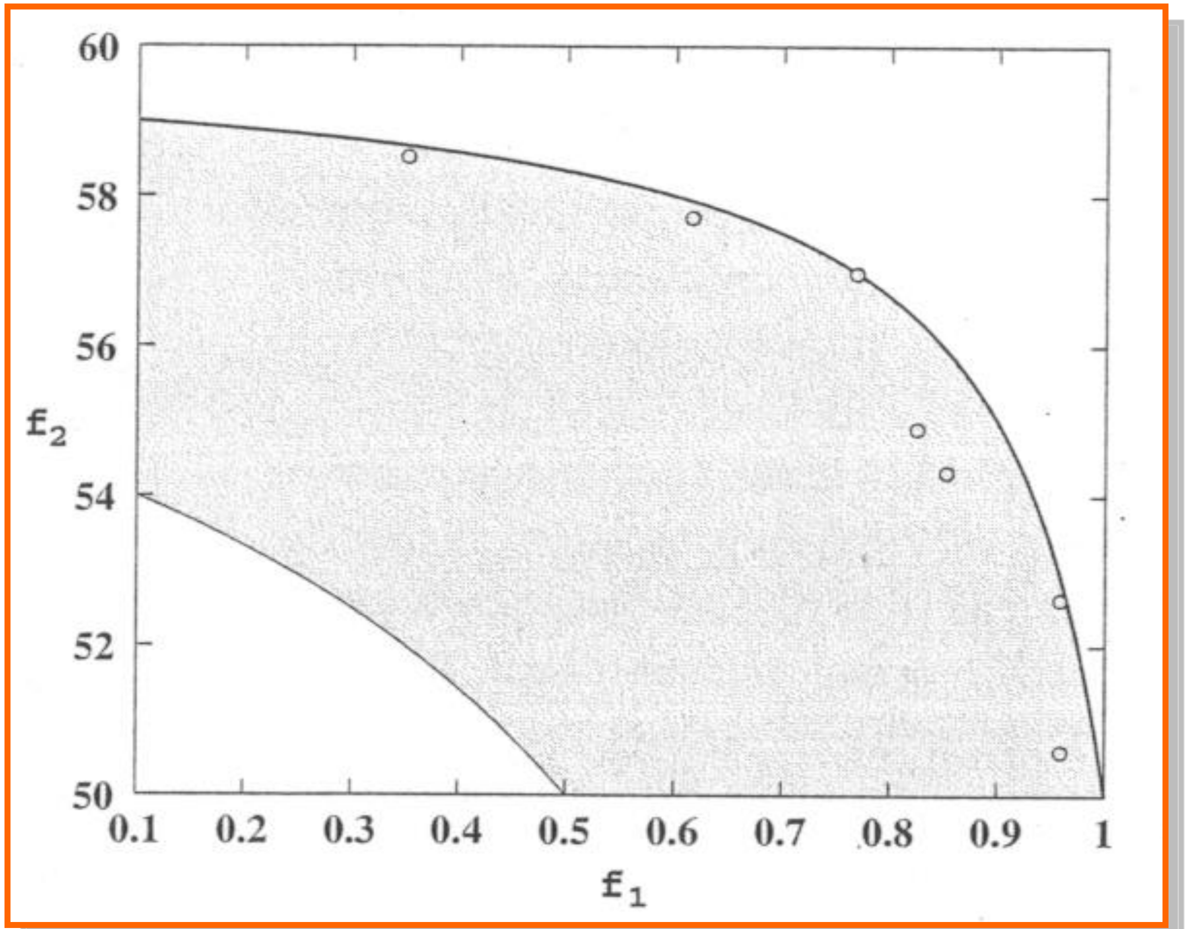
The shared fitness of this solution is $F_1' = F_1/nc_1 = 0.869/2 = 0.435$. Similarly, calculate the fitness of other solutions, $F_2=0.660$, $F_3=0.888$, $F_4=0.841$, $F_5=0.551$, and $F_6=0.265$. The corresponding shared fitness values are $F_2'=0.330$, $F_3'=0.444$, $F_4'=0.841$, $F_5'=0.551$, and $F_6'=0.133$.



Comparing solutions with the same weight vector:

Solution 3 is better than Solution 1 and solution 2 is better than solution 6.

- Weighted direction for each solution



Sample result after 500 generations of WBGA.

WBGA Advantages & Disadvantages

- Advantages
  - Easy to convert GA implementation into WBGA.
  - Lower complexity compared to other algorithms.
- Disadvantages
  - All problems have to be posed as maximization problems (due to proportionate selection on the shared fitness).
  - For mixed type objective functions, complications in choosing a fitness function may arise
  - Cannot find Pareto-optimal solutions in non-convex problems
  - Uniformly distributed weight vectors do not produce uniformly distributed Pareto-optimal solutions.
- Computational complexity: $O(MN^2)$

3- Multiple Objective GA (F-MOGA)

- The first MOEA focuses on non-dominated solutions and simultaneously maintains diversity in the solutions.

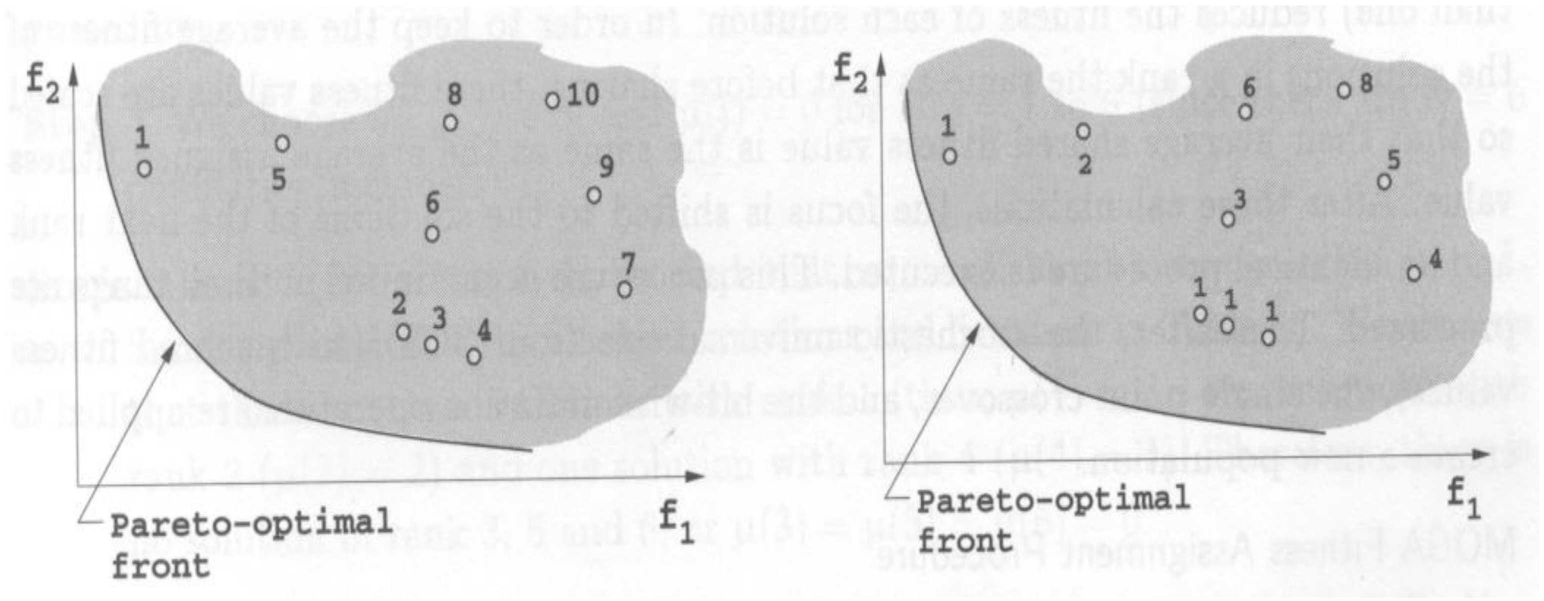
- First, each solution is checked for its domination in the population. To a solution *i*, a rank equal to one plus the *number* of solutions, n_i, that dominate solution *i* is assigned.

$$\boxed{r_i = 1 + n_i}$$

- Once the ranking is performed, a raw fitness to a solution is assigned based on its rank by using a *linear mapping* function. Thereafter, solutions of each rank are considered at a time and their raw fitness are averaged.

- The rest of the algorithm (stochastic universal selection, single-point crossover, bit-wise mutation) is the same as a classical GA.

Ranking in F-MOGA



A MOP and some of its solutions

Ranking of MOP solutions

- All non-dominated points will have a rank of 1.
- Maximum rank cannot be more than *N*.
- All possible ranks between 1 and *N* need not be assigned

Diversity Maintenance in F-MOGA

- MOGA introduced niching among solutions of each rank. The niche count with σ_{share} is found first. The sharing function with $\alpha=1$ is used, but the distance metric is computed with the objective function values, instead of the parameter values. Thus the normalized distance between any two solution *i* and *j* in a rank is calculated

$$d_{ij} = \sqrt{\sum_{k=1}^{M} \left(\frac{f_k^{(i)} - f_k^{(j)}}{f_k^{\max} - f_k^{\min}} \right)^2}$$

- For the solution *i*, d_{ij} is computed for each solution *j* having the *same* rank. Therefore the niche count is calculated by summing the sharing function values:

$$Sh(d_{ij}) = \begin{cases} 1 - \dfrac{d_{ij}}{\sigma_{\text{share}}}, & \text{if } d_{ij} \le \sigma_{\text{share}} \\ 0, & \text{otherwise} \end{cases}$$

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

where $\mu(r_i)$ is the number of solutions with rank r_i.

F-MOGA Pseudo Codes

Step 1: Choose a σ_{share} . Initialize $\mu(j)=0$ for all possible ranks *j* = 1,…,*N*. Set solution counter *i* = 1

Step 2: Calculate the number of solutions n_i that dominates solution *i*. Compute the rank of the *i*th solution as $r_i = 1 + n_i$. Increment the count for the number of solutions in rank r_i by one, that is $\mu(r_i) = \mu(r_i) + 1$

Step 3: If *i* < *N*, increment *i* by one and go to Step 1. Otherwise, go to Step 4.

Step 4: Identify the maximum rank *r** by checking the largest r_i which has $\mu(r_i) > 0$ The sorting according to rank and fitness-averaging yields the following assignment of the average fitness to any solution *i* = 1,…,*N*:

$$\boxed{F_i = N - \sum_{k=1}^{r_i - 1} \mu(k) - 0.5(\mu(r_i) - 1)}$$

To each solution *i* with rank r_i = 1, the above equation assigns a fitness equal to $F_i = N - 0.5(\mu(1) - 1)$, which is the average value of $\mu(1)$ consecutive integers from *N* to $N - \mu(1) + 1$. Set a rank counter *r* = 1.

Step 5: For each solution *i* in rank *r*, calculate the niche count nc_i with other solutions of the same rank by using

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

Calculate the shared fitness using $F_j^{'} = F_j / nc_j$. To preserve the same average fitness, scale the shared fitness by

$$F_j \leftarrow \frac{F_j \mu(r)}{\sum_{k=1}^{\mu(r)} F_k^{'}} F_j^{'}$$

Step 6: If $r < r^*$, increment *r* by one and go to Step 5. Otherwise, the process is complete.

- Note sharing distance is calculated *rank based* with *objective function values*, instead of *decision values* in NSGA.

F-MOGA Example

- Consider the problem

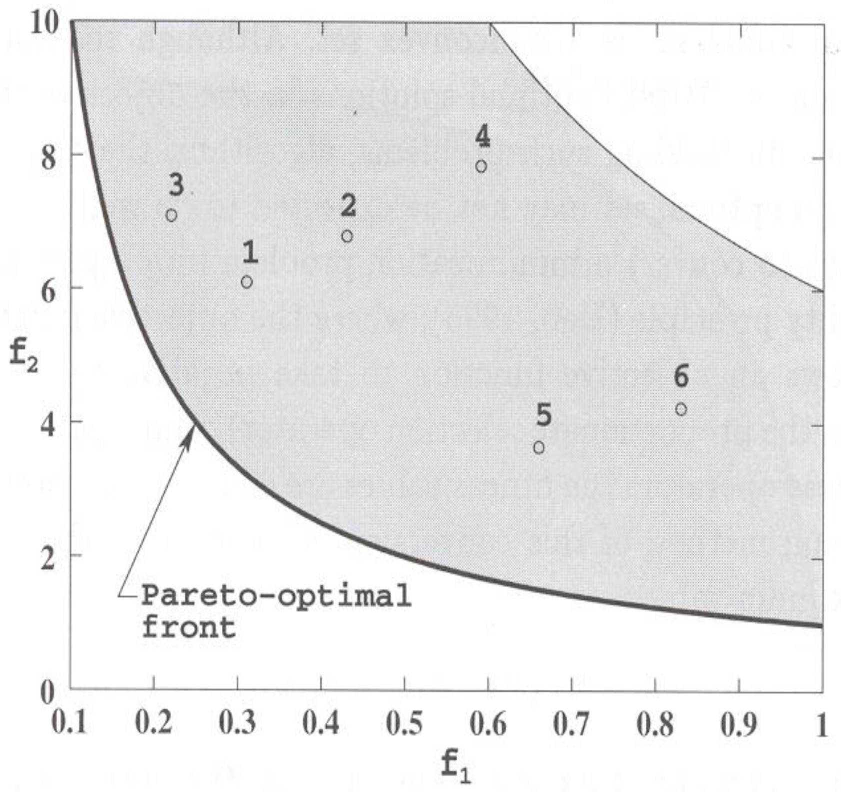
Minimize $f_1(x) = x_1$

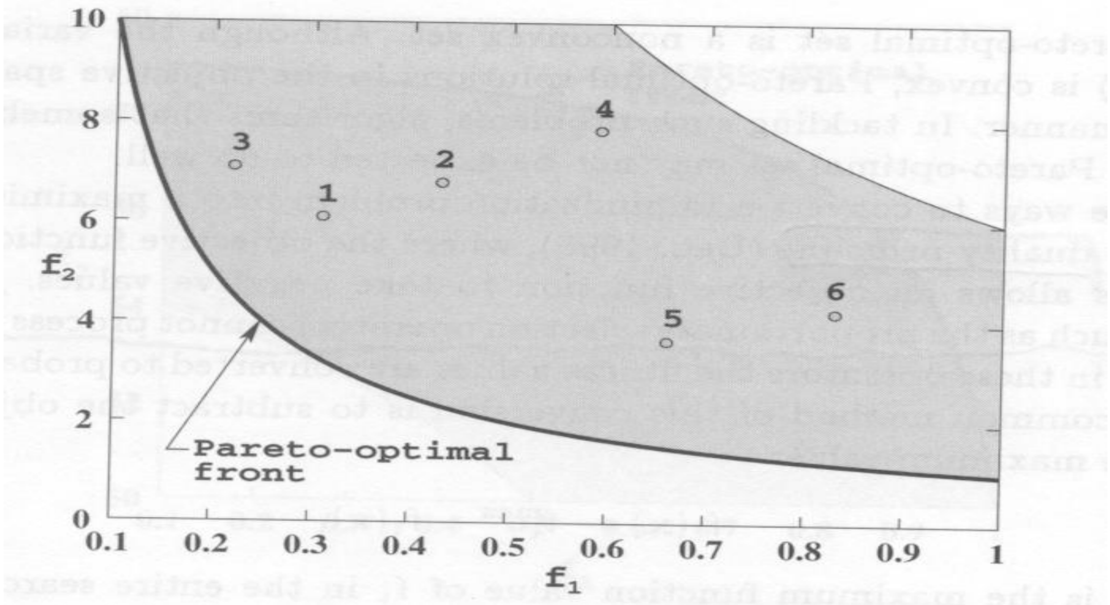
Minimize $f_2(x) = \dfrac{1 + x_2}{x_1}$

Subject to

$0.1 \le x_1 \le 1,$

$0 \le x_2 \le 5$

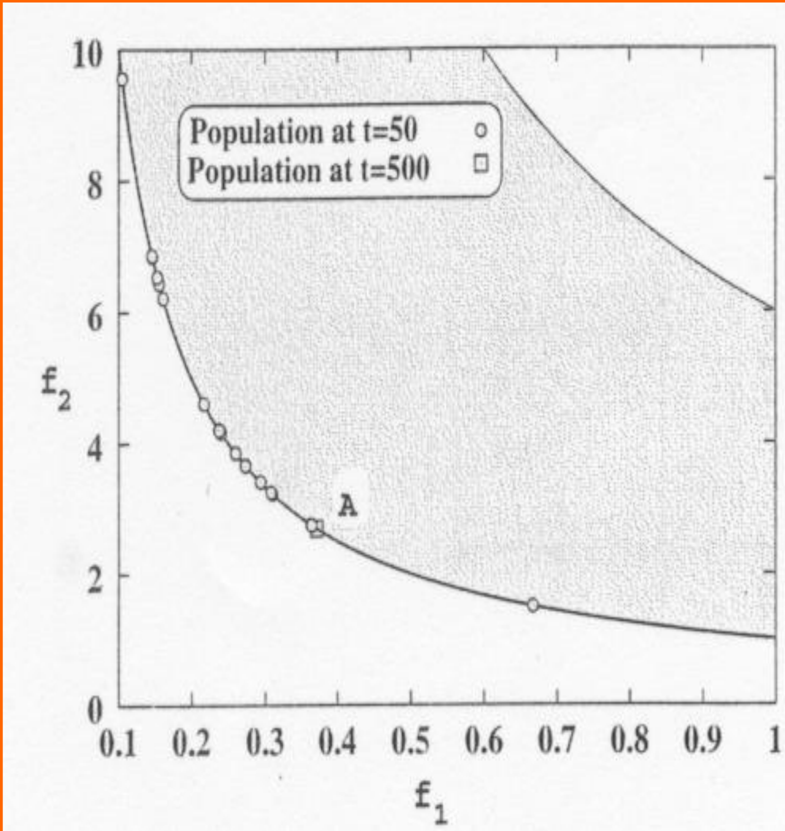


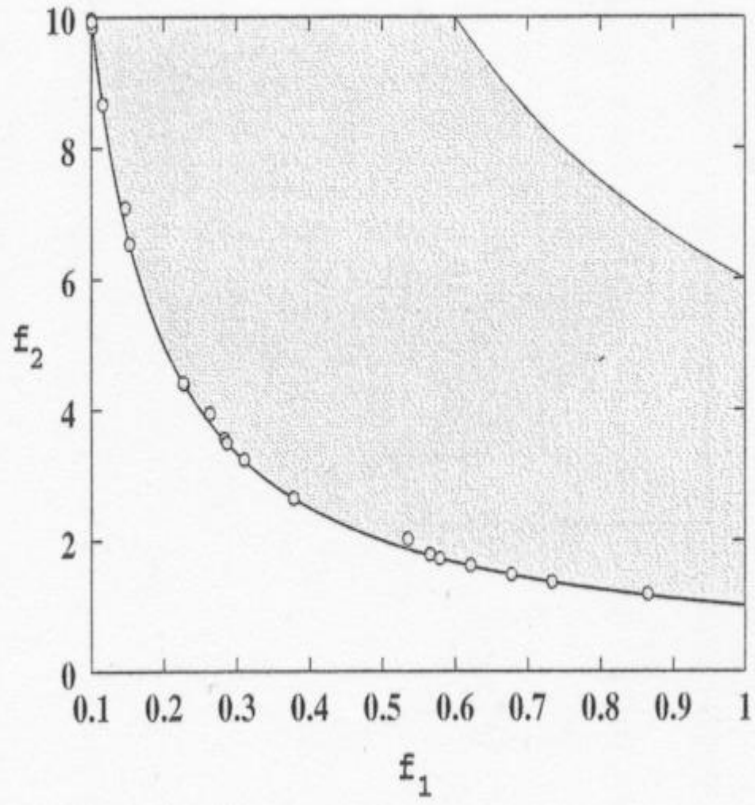


Solution	x_1	x_2	f_1	f_2	Rank	Average fitness	Niche count	Shared fitness	Scaled fitness
1	0.31	0.89	0.31	6.10	1	5.0	1.752	2.854	4.056
2	0.43	1.92	0.43	6.79	2	2.5	1.000	2.500	2.500
3	0.22	0.56	0.22	7.09	1	5.0	1.702	2.938	4.176
4	0.59	3.63	0.59	7.85	4	1.0	1.000	1.000	1.000
5	0.66	1.41	0.66	3.65	1	5.0	1.050	4.762	6.768
6	0.83	2.51	0.83	4.23	2	2.5	1.000	2.500	2.500

- Step 1: Choose σ_{share}=0.5, $\mu(j)$=0 for all j=1,…,6
- Steps 2 and 3: For solution 1, find that n_1=0, and thus r_1=1+0=1. Similarly find r_i for all other solutions. Observe that there are three solutions with rank 1($\mu(1)$=3), and $\mu(2)$=2 and $\mu(4)$=1.
- Step 4: The maximum rank r^*=4. Sorting of the population according to rank yields (1,3,5) (2,6) () (4). Assign a fitness of 6.00 to solution 1, a raw fitness of 5.00 to solution 3, and so on. Finally solution 4 gets a raw fitness equal to 1. Average the raw fitness of all solutions in each rank and reassign that value as the average fitness of each solution having the same rank. $F_1=F_3=F_5$=5.0, $F_2=F_6$=2.5, F_4=1.0.
- Step 5: For three solutions in the first rank, calculate the niche count in the objective space. Assume f_1^{min}=0.1, f_1^{max}=1.0, f_2^{min}=1, f_2^{max}=10. d_{13}=0.149, d_{15}=0.475, d_{35}=0.621. Calculate the sharing function with α=1, Sh(d_{13})=0.702, Sh(d_{15})=0.050, Sh(d_{35})=0. Of course Sh(d_{11})=Sh(d_{33})=Sh(d_{55})=1. So the niche counts are nc_1=1+0.702+0.050=1.752, nc_3=1+0.702+0=1.702, nc_5=1+0.050+0=1.050. Dividing the average fitness values by the niche counts, we calculate the shared fitness F_j' (column 9)

  Scale these fitness values so that their average is the same as the original average fitness values. Scaling factor is $F_1\mu(1)/(F_1'+F_3'+F_5')$= 5 x 3/(2.854+2.938+4.762)=1.421, Multiply column 9 by 1.421 to calculate the scaled fitness values of solutions 1, 3 and 5.
- Step 6: Move to Step 5 with solutions of rank 2.
- Step 5: Compute d_{26}=0.528. Thus Sh(d_{26})=0 and niche counts $nc_2=nc_6$=1. The shared fitness values are the same as the average fitness values and the scaling factor is one.
- Step 6: Move to Step 5 for rank 3 solutions. Since there are none, move to Step 5 for rank 4 solution Since there is only one, the scaled fitness value is the same as its average fitness.





Population dynamics at 50 and 500 Generations without mutation

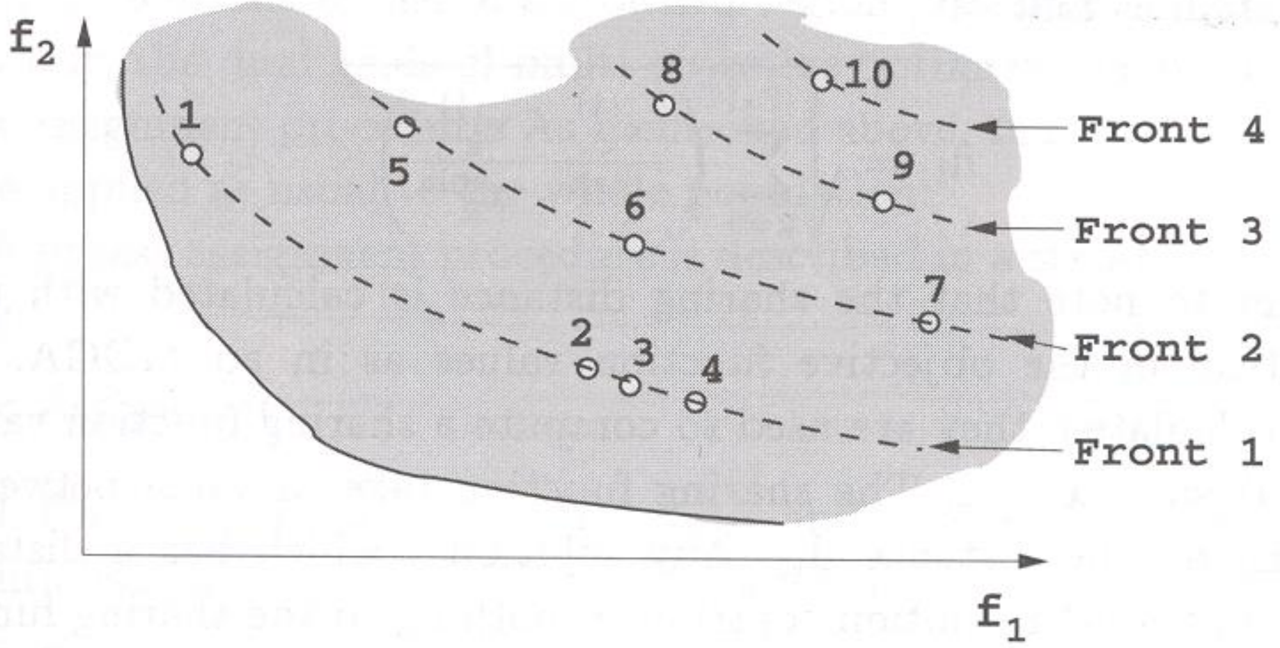
Population after 500 generations with mutation

F-MOGA Advantages & Disadvantages

- Advantages:
  - Fitness assignment scheme is simple
  - Since niching is performed in objective space, MOGA can easily be applied to other optimization problems such as combinational optimization problems.
- Disadvantages:
  - All solutions in a particular non-dominated front, need not have the same assigned fitness. Introduces unwanted bias towards some solutions in the search space.
  - The shared fitness computation procedure *does not ensure that every solution in a poorer rank will always have a worse scaled fitness F' than every solution in a better rank*. This happens if there is crowding of good solutions, then their shared fitness will be small. This leads to slow convergence or instability.
- Computational complexity: $O(MN^2)$

4- Non-Dominated Sorting GA (NSGA)

- The first step of NSGA is to sort the population *P* according to non-domination, $P = \cup_{j=1}^{\rho} P_j$
- The fitness assignment procedure begins from the first non-dominated set and successively proceeds to dominated sets. An solution *i* of the first (or best) non-dominated set is assigned a fitness equal to $F_i = N$ (the population size).
- Assigning more fitness to solutions belonging to a better non-dominated set ensures a selection pressure towards the Pareto-optimal front.



Whole population is classified into four different fronts.

- In NSGA, diversity is maintained by degrading the assigned fitness based on the number of *neighboring solutions* (crowdedness) sharing function
- The sharing function is used front wise, i.e. for each solution *i* in the front *P*, its distance from another solution *j* in the same front is calculated as

$$d_{ij} = \sqrt{\sum_{k=1}^{|P|} \left(\frac{x_k^{(i)} - x_k^{(j)}}{x_k^{\max} - x_k^{\min}} \right)^2}$$

- Note sharing distance is calculated *front-wise* with *decision variables*, instead of *objective function* values in MOGA.

$$Sh(d_{ij}) = \begin{cases} 1 - \left(\dfrac{d_{ij}}{\sigma_{\text{share}}}\right)^2, & \text{if } d_{ij} \le \sigma_{\text{share}} \\ 0, & \text{otherwise} \end{cases}$$

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

- The final task is to reduce the fitness of the *i*-th solution by its niche count and obtain the shared fitness value.

- The process of degrading fitness of a solution which is crowded by many solutions helps emphasize the solutions residing in less crowded regions

NSGA Features

- Fitness Assignment

  Fitness according to class of non-dominated sorting

- Diversity Maintenance

  Fitness sharing strategy among non-dominated front

- Selection

  Proportional selection (roulette wheel)

  Assign copies in the mating pool proportional to the shared fitness, so each solution in the first front has a better chance of surviving that that in the second front. On the other hand, sharing function approach among solutions in each front makes sure that solutions in the less crowded region get more copies in the mating pool.

NSGA Pseudo Codes

Step 1: Choose sharing parameter σ_{share} and a small positive number ε and initialize $F_{\min} = N + \varepsilon$. Set front counter $j = 1$

Step 2: Classify population P according to non-domination: $(P_1, P_2, \ldots, P_\rho)$

Step 3: For each $q \in P_j$

Step 3a: Assign fitness $F_j^{(q)} = F_{\min} - \varepsilon$

Step 3b: Calculate niche count nc_q using the following equation among solutions of P_j only

$$nc_q = \sum_{k=1}^{|P_j|} Sh(d_{qk})$$

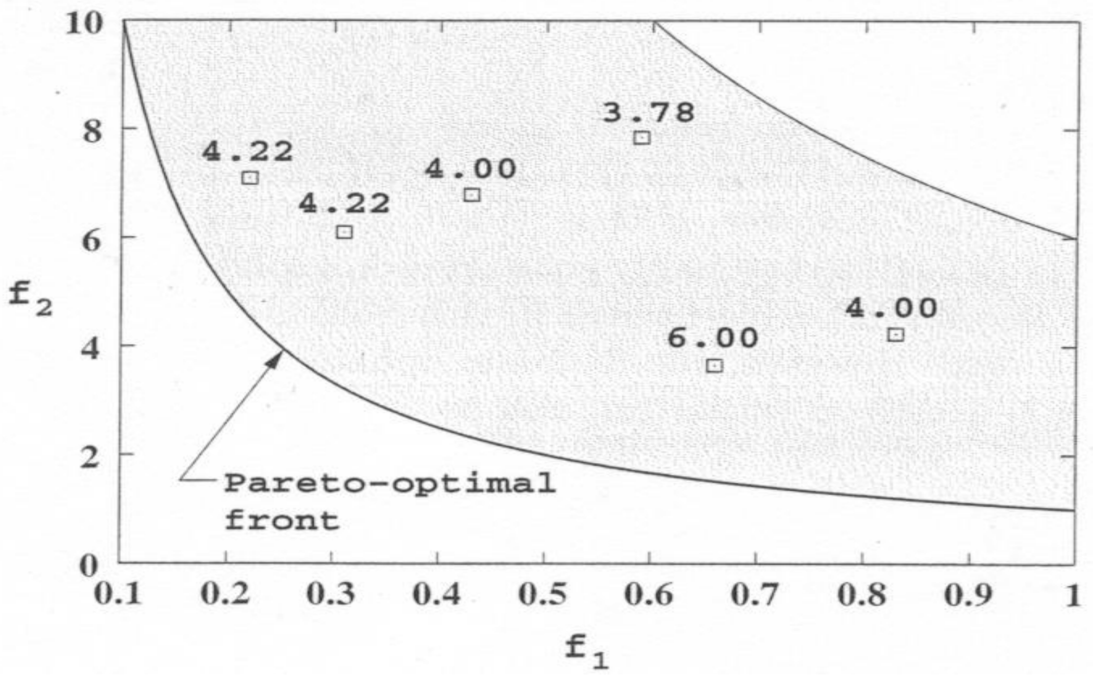
Step 3c: Calculate shared fitness $F_j^{'(q)} = F_j^{(q)} / nc_q$

Step 4: $F_{\min} = \min(F_j^{'(q)} : q \in P_j)$ and set $j = j + 1$

Step 5: If $j \le \rho$, go to Step 3. Otherwise, the process is complete.

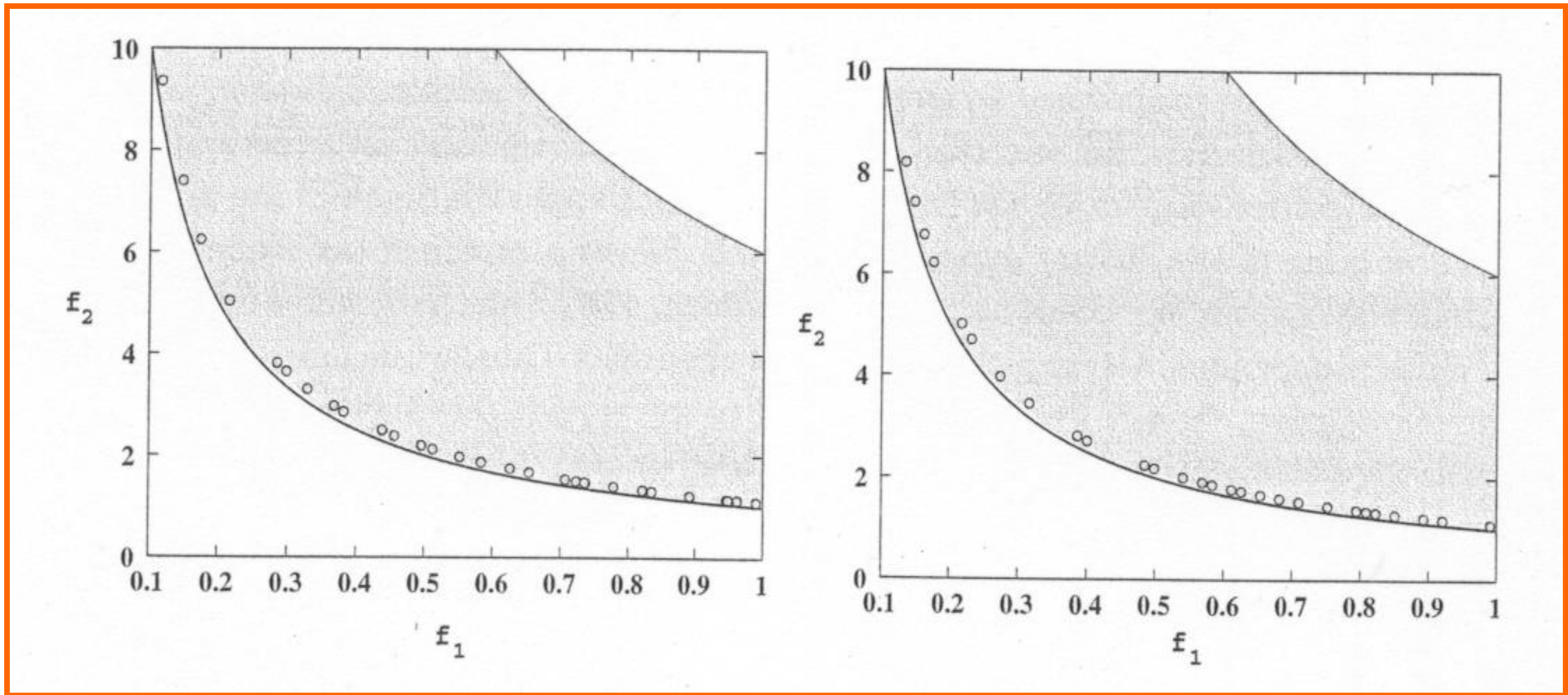
NSGA Example

Solution	x_1	x_2	f_1	f_2	Front Id	Assigned fitness	Shared fitness
1	0.31	0.89	0.31	6.10	1	6.00	4.22
2	0.43	1.92	0.43	6.79	2	4.00	4.00
3	0.22	0.56	0.22	7.09	1	6.00	4.22
4	0.59	3.63	0.59	7.85	3	3.78	3.78
5	0.66	1.41	0.66	3.65	1	6.00	6.00
6	0.83	2.51	0.83	4.23	2	4.00	4.00



Shared fitness values of the six solutions

- Step 1: choose σ_{share}=0.158 and ε=0.22, so F_{min}=6+0.22=6.22 and j=1.
- Step 2: classify the population into different non-dominated sets. Solutions 1, 3 and 5 belongs to the first non-dominated set P_1, solutions 2 and 6 belongs to P_2, while solution 4 belongs to P_3.
- Step 3: Being P_1, assign a fitness equal to the population size, 6.00 to solutions 1, 3 and 5. Modify these fitness values to emphasize diversity among the solutions. Calculate the normalized Euclidean distance, d_{13}=0.120, d_{15}=0.403, d_{35}=0.518. Calculate the sharing function values, Sh(d_{13})=0.423, Sh(d_{15})=0, Sh(d_{35})=0. Calculate the niche count for all three solutions: nc_1=Sh(d_{11})+Sh(d_{13})+Sh(d_{15})=1.423, nc_3=Sh(d_{31})+Sh(d_{33})+Sh(d_{35})=1.423, nc_5=Sh(d_{51})+Sh(d_{53})+Sh(d_{55})=1.000. Compute the shared fitness by dividing the assigned fitness with the niche count.
- Step 4: Now, F_{min}=min(4.22, 4.22, 6.00)=4.22, set j=2.
- Step 5: Continue the same procedure for assigning fitness to solutions in the second and the third fronts.



NSGA after 50 generation

NSGA after 500 generation

Results are close to Pareto front after 50 generations but are more evenly spread after 500 generations.

NSGA Observations

- No solution in a front has a worse shared fitness value than a solution in a worse front. For example, any solution in front 1 has a better fitness than any solution in fronts 2 and 3.
  - Provide a selection pressure towards the non-dominated solutions
- With a front, less crowded solutions (in the parameter space) have better fitness values.
  - Ensure that diversity is maintained among the non-dominated solutions

NSGA Advantages & Disadvantages

- Advantages:
  - The assignment of fitness according to non-dominated sets
  - Since better non-dominated sets are emphasized systematically, the NSGA progresses to the Pareto-optimal region front-wise
  - Performing sharing in the parameter space allows phenotypically diverse solutions to emerge.
  - If desired, the sharing can be performed in the objective space.
- Disadvantages:
  - The sharing function approach requires the fixing of the sharing parameter σ_{share}
  - The performance of the NSGA is sensitive to the sharing parameter σ_{share}, What about a dynamic update procedure?
- Computational complexity: $O(MN^2)$

5- Niched Pareto GA (NPGA)

- Uses the binary tournament selection scheme, instead of proportionate selection method used in VEGA, F-MOGA, and NSGA.
- It has been shown in SOP that tournament selection has better growth and convergence properties compared to proportionate selection
- Solutions in a tournament are checked for domination with respect to a small subpopulation *T* of size t_{dom} Therefore each solution *i* and *j* is compared with every solution in *T* for domination.
- If one dominates all solutions in T, but the other is dominated by at least one solution in *T*, the former is chosen.
- If both solutions *i* and *j* are either dominated by at least one solution in *T* or are not dominated by any solution in *T*, both are checked with the current offspring population *Q*. Each solution is placed in the offspring population and its niche count is calculated. The solution with the smaller niche count wins the tournament.
- The NPGA requires fixing two parameters: σ_{share} and t_{dom}
- The parameter σ_{share} has more effect on NPGA than on NSGA, because the niche count nc_i of a solution *i* is calculated as the number of population members present within a distance σ_{share} from solution *i*
- The performance of the NPGA depends on the choice of t_{dom}

NPGA Tournament Selection

Winner = NPGA-tournament(i, j, Q)

- Step T1: Pick a subpopulation T_{ij} of size t_{dom} from the parent population *P*.
- Step T2: Find α_i as the number of solutions in T_{ij} that dominates *i*, Calculate α_j as the number of solutions in T_{ij} that dominates *j*.
- Step T3: If $\alpha_i=0$ and $\alpha_j>0$, then *i* is the winner. The process is complete.
- Step T4: Otherwise, if $\alpha_i>0$ and $\alpha_j=0$, then *j* is the winner. The process is complete.
- Step T5: Otherwise, if $|Q|<2$, *i* or *j* is chosen as a winner with probability 0.5. The process is complete. Alternatively, the niche count nc_i and nc_j are calculated by placing *i* and *j* in the current offspring population Q., independently. With the niching parameter σ_{share}, nc_i is calculated as the number of offspring within a σ_{share} distance d_{ik} from *i.* The distance d_{ik} is the Euclidean distance between solutions *i* and *k* in the objective space

$$d_{ij} = \sqrt{\sum_{m=1}^{M} \left(\frac{f_m^{(i)} - f_m^{(k)}}{f_m^{\max} - f_m^{\min}} \right)^2}$$

  Note that f_m^{max} and f_m^{min} are the maximum and minimum bounds of the *m*th objective function in the search space.
- Step T6: If $nc_i < nc_j$, solution *i* is the winner. Otherwise, solution *j* is the winner.

NPGA Pseudo Codes

Step 1: Shuffle P, set i=1 and $Q=\varnothing$.

Step 2: Perform the tournament selection and find the first parent p_1=NPGA-tournament(i,i+1,Q).

Step 3: Set $i=i+2$, and find the second parent p_2=NPGA-tournament(i,i+1,Q).

Step 4: Perform crossover with p_1 and p_2 and create offspring c_1 and c_2. Perform mutation on c_1 and c_2.

Step 5: Update offspring population $Q=Q\text{U}(c_1,c_2)$.

Step 6: Set $i=i+1$. If $i<N$, go to Step 2. Otherwise, if $|Q|=N/2$, shuffle P, set i=1, and go to Step 2. Otherwise, the process is complete.

NPGA Advantages & Disadvantages

- Advantages:
  - No explicit fitness assignment is needed.
  - First MOGA proposed to use the tournament selection. However, the dynamically updated tournament selection somewhat reduces the elegance with which tournament selection is applied.
- Disadvantages:
  - Require fixing two parameters: σ_{share} and t_{dom}
  - σ_{share} has more effect on NPGA than on NSGA. This is because the niche count is calculated as the number of population members present within a distance σ_{share} from solution.
  - Performance of the NPGA depends on the choice of t_{dom}. No guideline exists. Experiment shows that t_{dom} should be an order of magnitude smaller than the population size. If t_{dom} is too small, the non-dominated check would be noisy. On the other hand, if t_{dom} is too large, true non-dominated solution will be emphasized, but the computational complexity would be greater. Ideally this parameter should also depends on the number of objectives being optimized.
- Computational complexity: $O(MN^2)$

Summary of Non-Elitist MOEAs

- Elite-preservation is missing
- Elite-preservation is important for proper convergence in single-objective EAs
- Same is true in EMOs
- Three tasks
  - Progress towards the Pareto-optimal front
  - Maintain diversity among solutions
  - Elite preservation

Non-Elitist MOEA

Elitist MOEA

Fundamental Concept of Elitism

- Elite-preserving operator favors the elitist of a population by giving them an opportunity to be directly carried over to the next generation.
- In single objective optimization, crossover is one possible design.
- This makes sure that the fitness of the population-best solution does not deteriorate. In this design, a good solution found early on in the run will never be lost unless a better solution is discovered.
- Elitism can be implemented to different degrees. For example, keep track of the best solution in a population but not use the elitist in any genetic operations. Or, all elites present in the current population can be carried over to the new population. The interesting choice will be an intermediate degree of elitism.

6- Elitist Non-dominated Sorting GA

- NSGA II has both elite-preservation strategy and an *explicit* diversity-preserving mechanism.
- After creating an offspring population Q_t from the parent population P_t, the two are combined and then a non-dominated sorting is used to classify the entire population R_t of size $2N$. This allows global non-domination check among the offsprings and parent solutions.
- Then the new population of size N is filled with solutions of different non-dominated fronts one at a time starting with the best non dominated front. All fronts which could not be accommodated are just deleted.

"A fast and elitist multiobjective genetic algorithm: NSGA-II," Deb K., Pratap A., Agarwal S., and Meyarivan T., *IEEE Transactions on Evolutionary Computation*, 6(2), 2002, pp. 182-197.

- When the last allowed front is being considered, there may exist more solutions than the number of remaining slots in the new population, so a niching strategy is used to choose the members which reside in the least crowded region in that front.
- The above especially useful in the later stages when most of the solutions lie in the first non dominated front, ensuring better spread in the Pareto-optimal front.
- Binary tournament selection (with a crowded tournament operator), recombination and mutation operator are used to create a offspring population.
- Code is available at www.iitk.ac.in/kangal/soft.htm

NSGA II Pseudo Codes

Step 1: Combine parent and offspring populations and create $R_t = P_t \cup Q_t$. Perform a non-dominated sorting to R_t and identify different fronts:

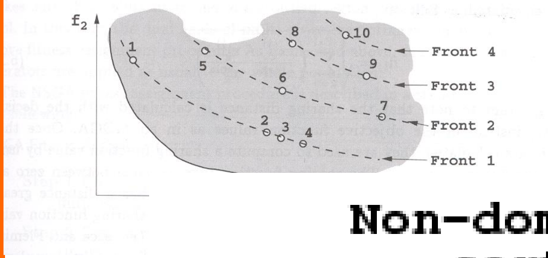
$$F_i, i = 1,2,\ldots, etc.$$

Step 2: Set new population $P_{t+1} = \emptyset$ Set a counter i=1. Until $|P_{t+1}| + |F_i| < N$, perform $P_{t+1} = P_{t+1} \cup F_i$ and i=i+1.

Step 3: Perform the Crowding-sort $(F_i, <_c)$ procedure and include the most widely spread $(N - |P_{t+1}|)$ solutions by using the crowding distance values in the sorted F_i to P_{t+1}

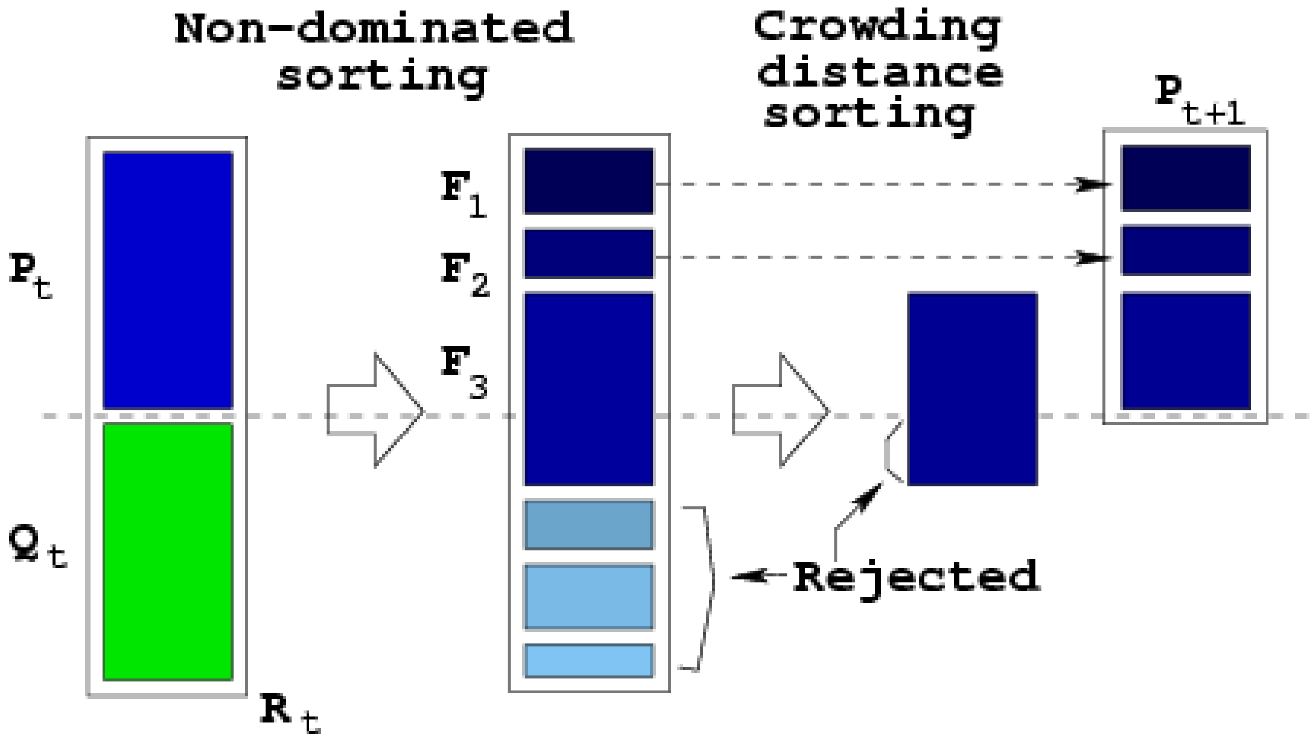
Step 4: Create offspring population Q_{t+1} from P_{t+1} by using the crowded tournament selection, crossover and mutation operators.

NSGA II Procedure



Elites are preserved

Non-dominated solutions are emphasized



Crowded Tournament Selection

- The crowded comparison operator $(<_c)$ compares two solutions and returns the winner of the tournament based on the non-dominated rank r_i and the local crowding distance d_i.
- A solution *i* wins a tournament with another solution *j* if any of the following conditions are true:
    - $r_i < r_j$
    - $r_i = r_j$ and $d_i > d_j$

Crowding Distance

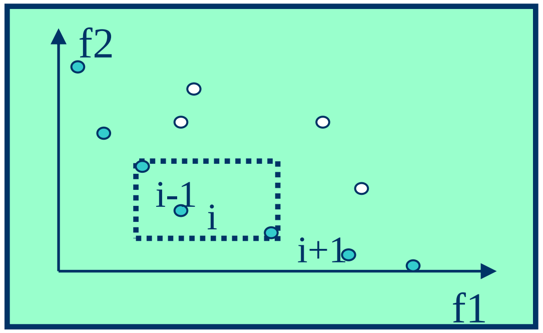
- Crowing-sort ($<_c$)

  Step C1: Call the number of solutions in F as $l = |F|$. For each i in the set, first assign $d_i = 0$

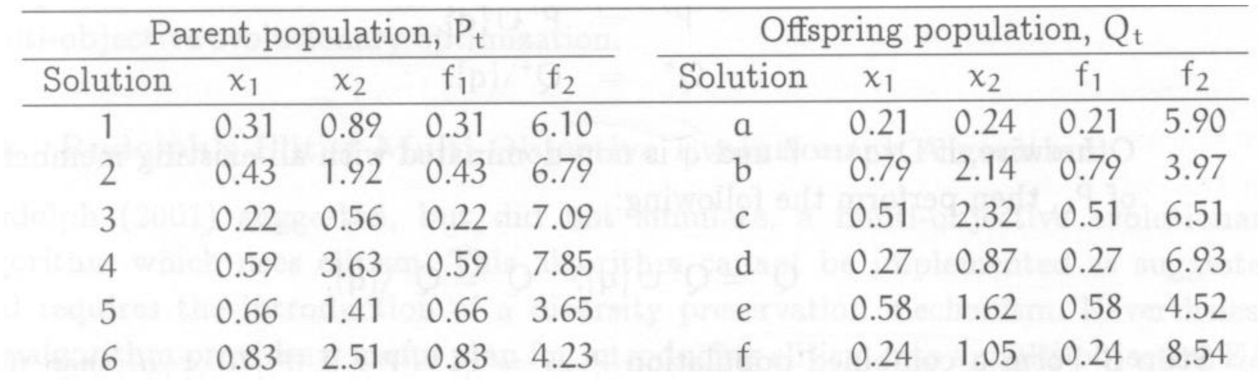
  Step C2: For each objective function $m = 1,2,\ldots,M$, sort the set in worse order of f_m or, find the sorted indices vector: $I^m = \text{sort}(f_m,>)$.

  Step C3: For $m=1,2,\ldots,M$, assign a large distance to the boundary solutions, or ∞, and for all other solutions $j = 2$ to $(l-1)$, assign

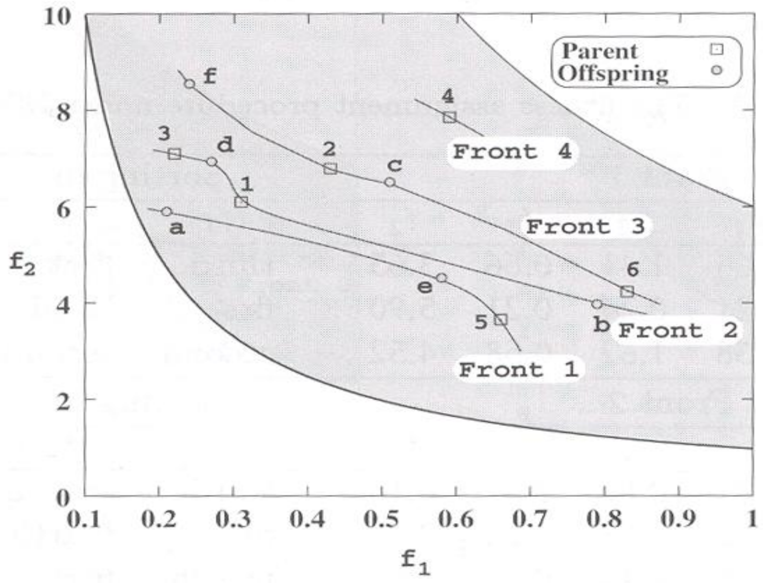
$$d_{I_j^m} = d_{I_j^m} + \frac{f_m^{(I_{j+1}^m)} - f_m^{(I_{j-1}^m)}}{f_m^{\max} - f_m^{\min}}$$



NSGA II Example

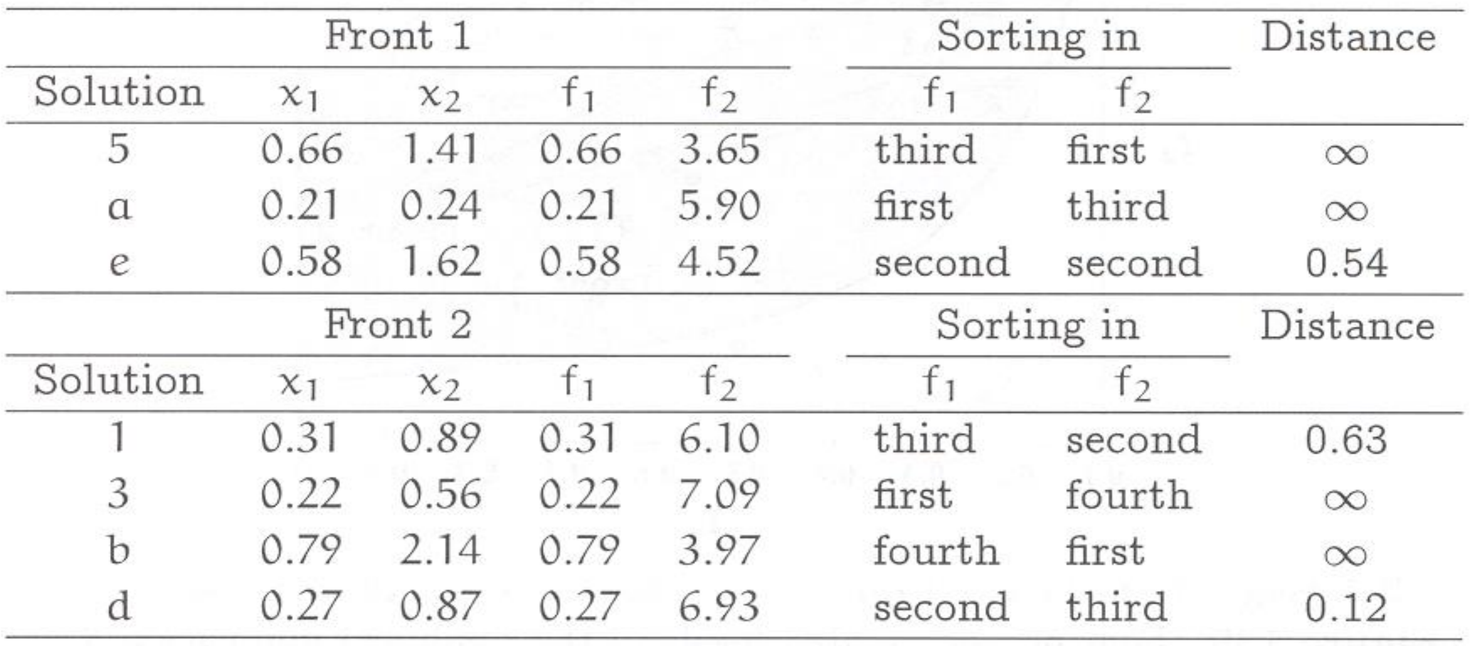


Parent population, P_t					Offspring population, Q_t				
Solution	x_1	x_2	f_1	f_2	Solution	x_1	x_2	f_1	f_2
1	0.31	0.89	0.31	6.10	a	0.21	0.24	0.21	5.90
2	0.43	1.92	0.43	6.79	b	0.79	2.14	0.79	3.97
3	0.22	0.56	0.22	7.09	c	0.51	2.32	0.51	6.51
4	0.59	3.63	0.59	7.85	d	0.27	0.87	0.27	6.93
5	0.66	1.41	0.66	3.65	e	0.58	1.62	0.58	4.52
6	0.83	2.51	0.83	4.23	f	0.24	1.05	0.24	8.54

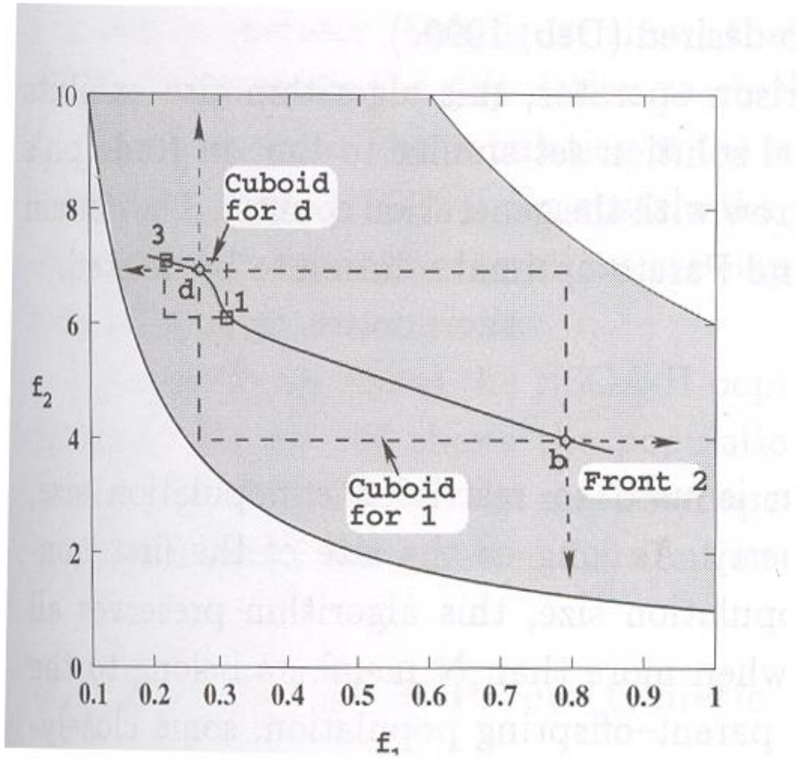


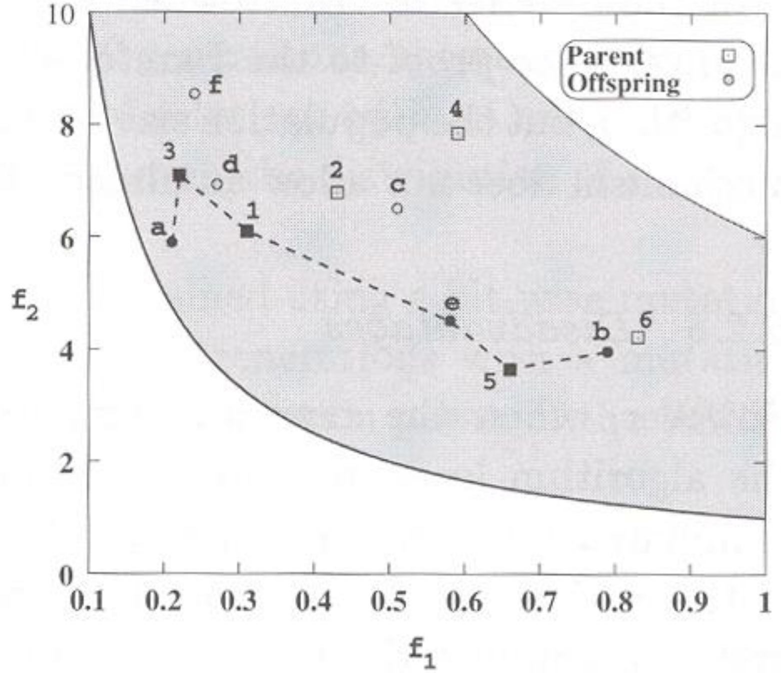
R_t = {1, 2, 3, 4, 5, 6, a, b, c, d, e, f}
Four non-dominated fronts of the combined population R_t;

F_1 = {5, a, e} $\longrightarrow$ P_{t+1}
F_2 = {1, 3, b, d} stop!
F_3 = {2, 6, c, f}
F_4 = {4}



Front 1					Sorting in		Distance
Solution	x_1	x_2	f_1	f_2	f_1	f_2	
5	0.66	1.41	0.66	3.65	third	first	∞
a	0.21	0.24	0.21	5.90	first	third	∞
e	0.58	1.62	0.58	4.52	second	second	0.54
Front 2					**Sorting in**		**Distance**
Solution	x_1	x_2	f_1	f_2	f_1	f_2	
1	0.31	0.89	0.31	6.10	third	second	0.63
3	0.22	0.56	0.22	7.09	first	fourth	∞
b	0.79	2.14	0.79	3.97	fourth	first	∞
d	0.27	0.87	0.27	6.93	second	third	0.12

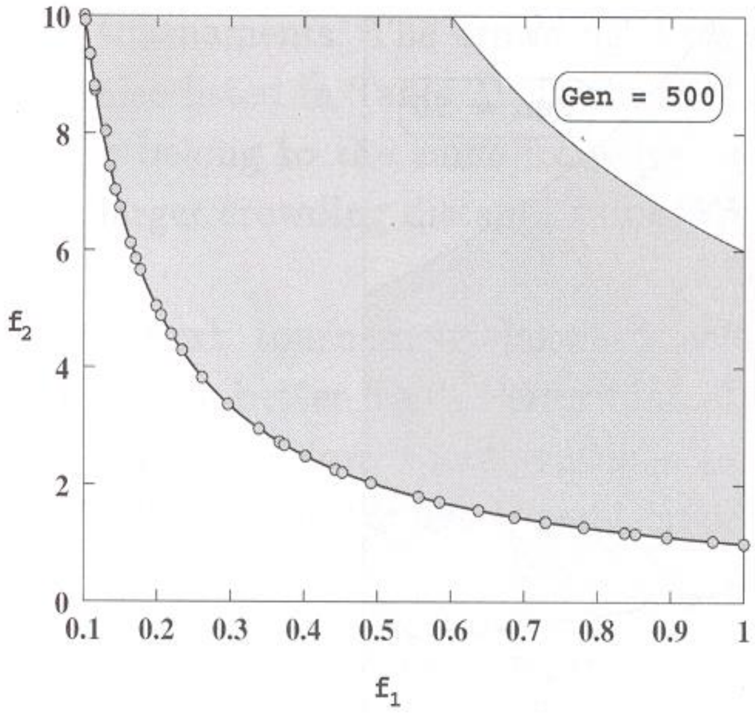


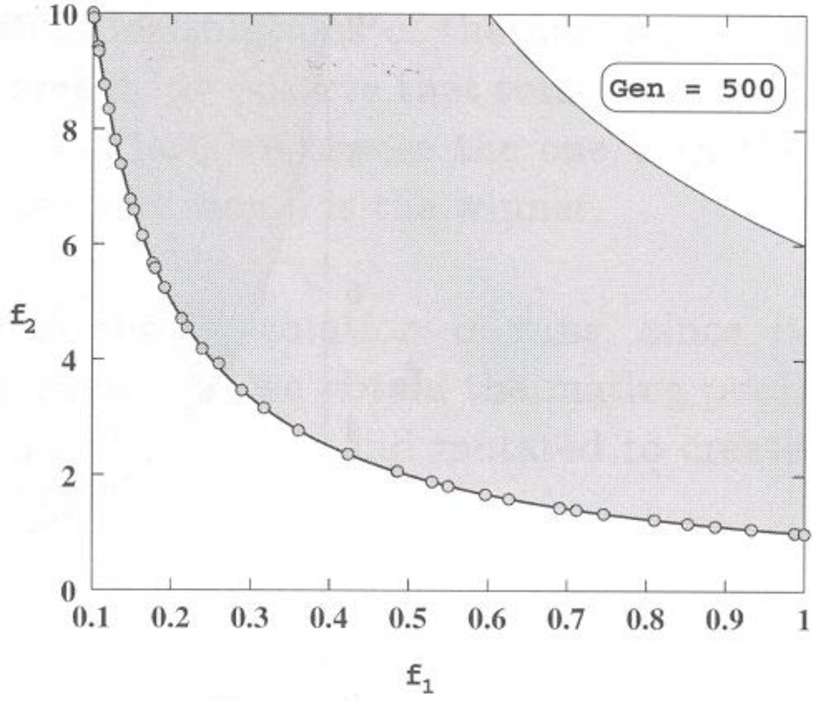


The cuboids of solutions 1 and d.
The cuboids of 3 and b extend to infinity

The parent population P_{t+1} joined by line.

- Tournament selection to generate the next offspring
  - Assume we pair the solutions (5,e), (a,3), (1,b), (a,1), (e,b), and (3,5), so that each solution participates in exactly two tournaments.
  - In the first tournament, we observe that solution 5 and e belongs to the same front. Thus we choose the one with the larger crowding distance value. Solution 5 is the winner.
  - In the next, solution a wins. We obtain the mating pool: {5, a, b, a, e, 5}. Now, these solutions can be mated pair-wise and mutated to create Q_{t+1}. This complete one generation of the NSGA-II





Population after 500 generations without mutation.

Population after 500 generations with mutation

(Min) $f_1(x) = x_1$

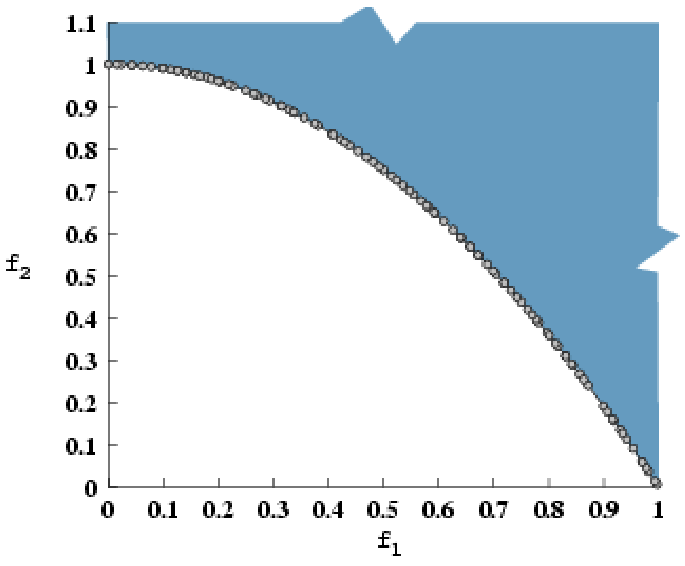
(Min) $f_2(x) = g\left[1 - \left(\frac{f_1}{g}\right)^2\right]$

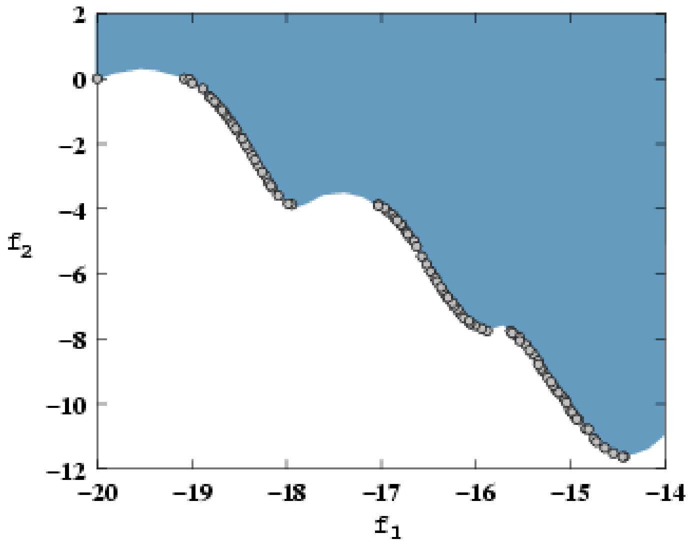
Where $g(x) = 1 + \frac{9}{n-1}\sum_{i=2}^{n} x_i$

(Min) $f_1(x) = x_1$

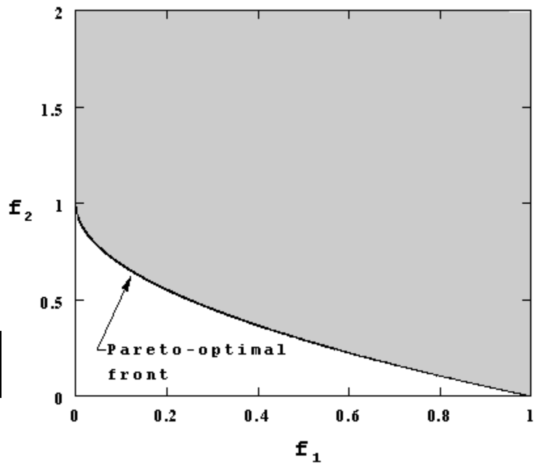
(Min) $f_2(x) = g\left[1 - \sqrt{\frac{f_1}{g}} - \frac{f_1}{g}\sin(10\pi f_1)\right]$

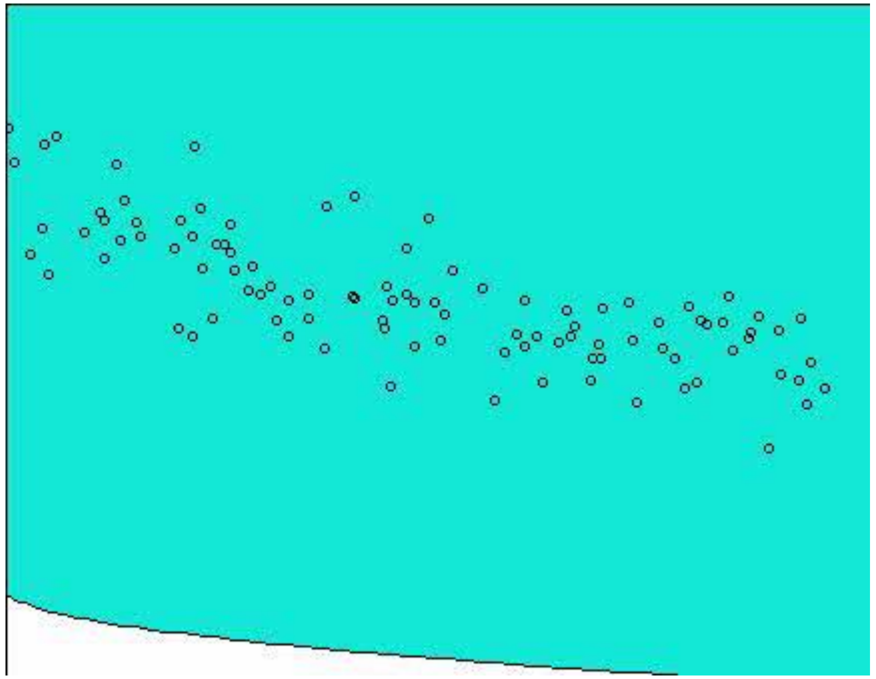
Where $g(x) = 1 + \frac{9}{n-1}\sum_{i=2}^{n} x_i$





NSGA II for ZDT1





Pros & Cons

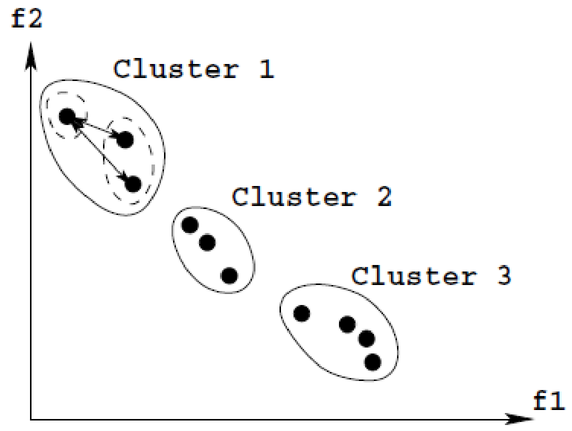
- Non-dominated Sorting for Convergence
- Crowding Distance for Diversity Preservation
- The elitism mechanism does not allow an already found Pareto-optimal solution to be deleted.

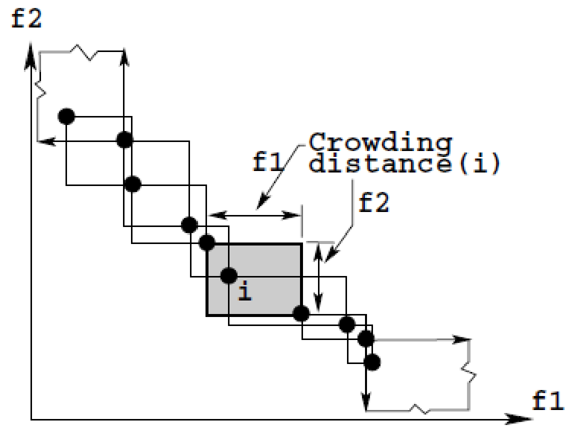
- Favor boundary solutions
- Computational complexity is still high, O(MN^2)
- Perform well in problems having a Pareto-optimal front consisting of several noncontiguous convex parts.

- Perform poorly in high-dimensional objective spaces

7- Clustered NSGA II

- Straightforward replacement of NSGA II;s crowded distance method by the clustering method used in SPEA2.
- The clustered NSGA II is expected to find a better distributed set of Pareto-optimal solutions than the original NSGA II, but at the expense of a larger computational time.





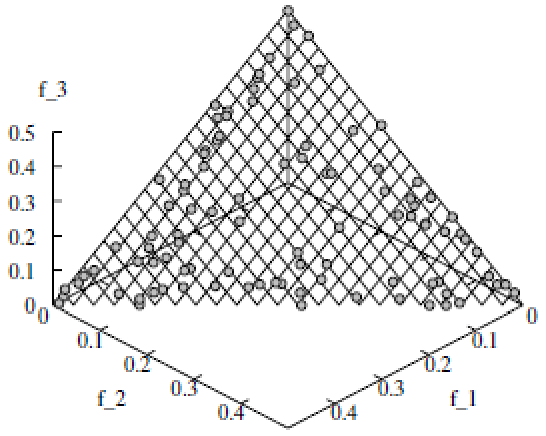


Figure 9: NSGA-II distribution on DTLZ1.

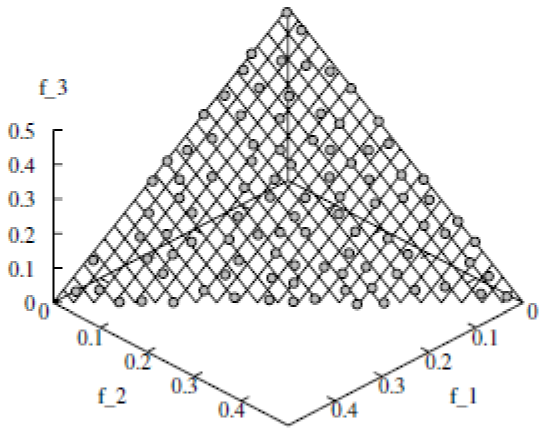
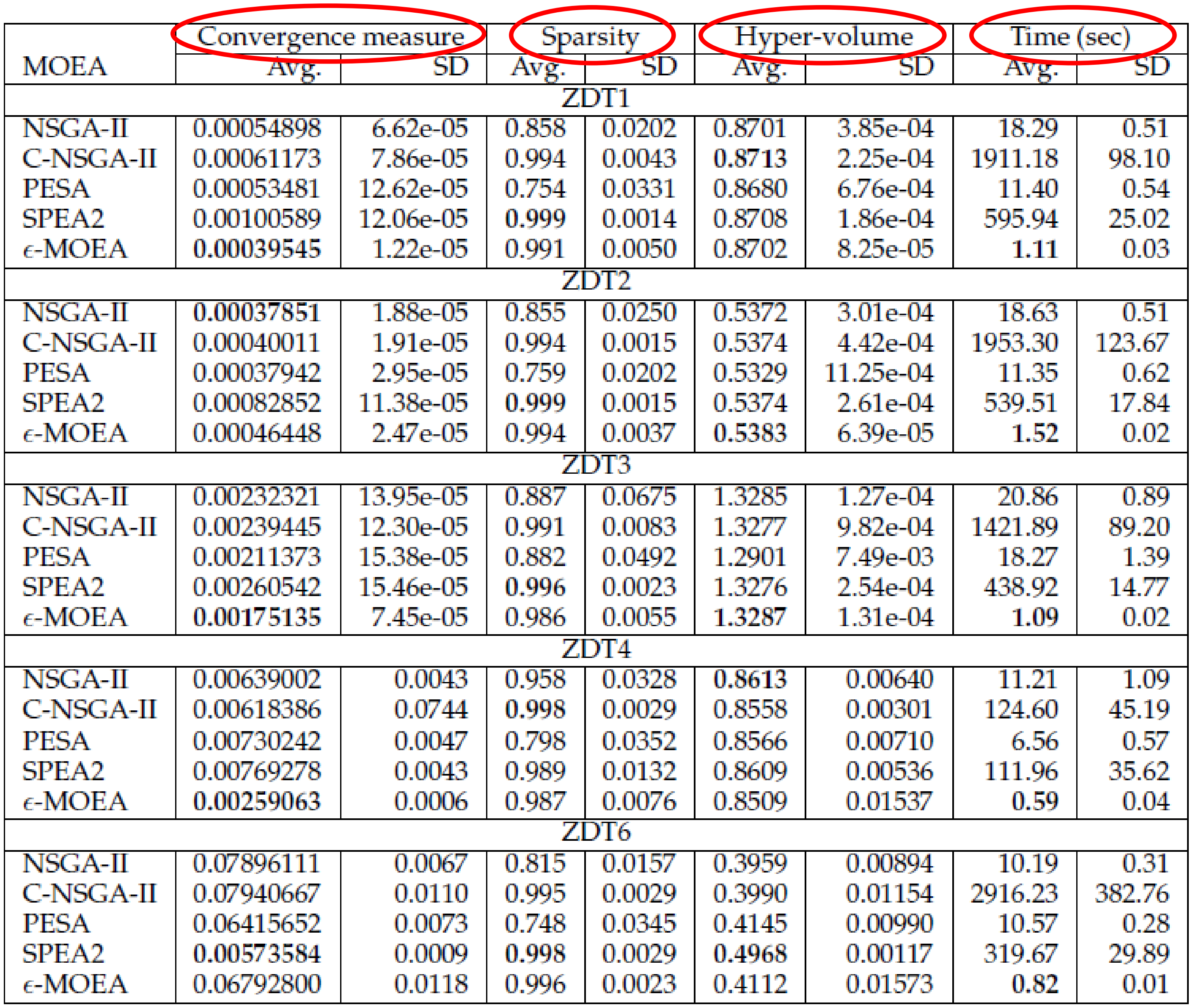


Figure 10: C-NSGA-II distribution on DTLZ1.



MOEA	Convergence measure		Sparsity		Hyper-volume		Time (sec)	
	Avg.	SD	Avg.	SD	Avg.	SD	Avg.	SD
ZDT1								
NSGA-II	0.00054898	6.62e-05	0.858	0.0202	0.8701	3.85e-04	18.29	0.51
C-NSGA-II	0.00061173	7.86e-05	0.994	0.0043	**0.8713**	2.25e-04	1911.18	98.10
PESA	0.00053481	12.62e-05	0.754	0.0331	0.8680	6.76e-04	11.40	0.54
SPEA2	0.00100589	12.06e-05	**0.999**	0.0014	0.8708	1.86e-04	595.94	25.02
ϵ-MOEA	**0.00039545**	1.22e-05	0.991	0.0050	0.8702	8.25e-05	**1.11**	0.03
ZDT2								
NSGA-II	**0.00037851**	1.88e-05	0.855	0.0250	0.5372	3.01e-04	18.63	0.51
C-NSGA-II	0.00040011	1.91e-05	0.994	0.0015	0.5374	4.42e-04	1953.30	123.67
PESA	0.00037942	2.95e-05	0.759	0.0202	0.5329	11.25e-04	11.35	0.62
SPEA2	0.00082852	11.38e-05	**0.999**	0.0015	0.5374	2.61e-04	539.51	17.84
ϵ-MOEA	0.00046448	2.47e-05	0.994	0.0037	**0.5383**	6.39e-05	**1.52**	0.02
ZDT3								
NSGA-II	0.00232321	13.95e-05	0.887	0.0675	1.3285	1.27e-04	20.86	0.89
C-NSGA-II	0.00239445	12.30e-05	0.991	0.0083	1.3277	9.82e-04	1421.89	89.20
PESA	0.00211373	15.38e-05	0.882	0.0492	1.2901	7.49e-03	18.27	1.39
SPEA2	0.00260542	15.46e-05	**0.996**	0.0023	1.3276	2.54e-04	438.92	14.77
ϵ-MOEA	**0.00175135**	7.45e-05	0.986	0.0055	**1.3287**	1.31e-04	**1.09**	0.02
ZDT4								
NSGA-II	0.00639002	0.0043	0.958	0.0328	**0.8613**	0.00640	11.21	1.09
C-NSGA-II	0.00618386	0.0744	**0.998**	0.0029	0.8558	0.00301	124.60	45.19
PESA	0.00730242	0.0047	0.798	0.0352	0.8566	0.00710	6.56	0.57
SPEA2	0.00769278	0.0043	0.989	0.0132	0.8609	0.00536	111.96	35.62
ϵ-MOEA	**0.00259063**	0.0006	0.987	0.0076	0.8509	0.01537	**0.59**	0.04
ZDT6								
NSGA-II	0.07896111	0.0067	0.815	0.0157	0.3959	0.00894	10.19	0.31
C-NSGA-II	0.07940667	0.0110	0.995	0.0029	0.3990	0.01154	2916.23	382.76
PESA	0.06415652	0.0073	0.748	0.0345	0.4145	0.00990	10.57	0.28
SPEA2	**0.00573584**	0.0009	**0.998**	0.0029	**0.4968**	0.00117	319.67	29.89
ϵ-MOEA	0.06792800	0.0118	0.996	0.0023	0.4112	0.01573	**0.82**	0.01