IT3708 Sub-symbolic AI Methods

Lecture 9 – Multi-Objective Evolutionary Algorithms

Kazi Shah Nawaz Ripon
ksripon@idi.ntnu.no

2016-03-08

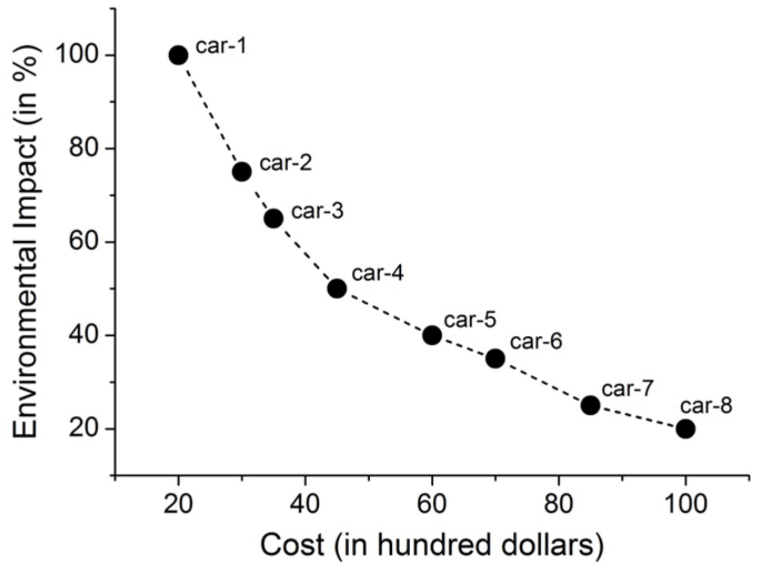
Suggested Reading

K. Deb, *Multi-Objective Optimization using Evolutionary Algorithms,* John Wiley & Sons, Inc., 2001

NTNU

Multi-Objective Optimization

- In single-objective optimization, there is typically a single solution that gives the best objective value.
- In MOOPs, there is no single optimal solution; there are, instead, a set of alternative solutions.
- This is because a solution that is optimal with respect to one objective might be a poor candidate for another objective.



General Concept

Multi-objective optimization is the process of simultaneously optimizing two or more **conflicting** objectives subject to certain constraints.

- Maximizing profit and minimizing the cost of a product.
- Maximizing performance and minimizing fuel consumption of a vehicle.
- Minimizing weight while maximizing the strength of a particular component.

Difference

Single Objective Optimization

- ❖ Optimize only one objective function
- ❖ Single optimal solution
- ❖ Maximum/Minimum fitness value is selected as the best solution.

Multi-Objective Optimization

- ❖ Optimize two or more objective functions
- ❖ Set of optimal solutions
- ❖ Comparison of solutions by
  - Domination
  - Non-domination

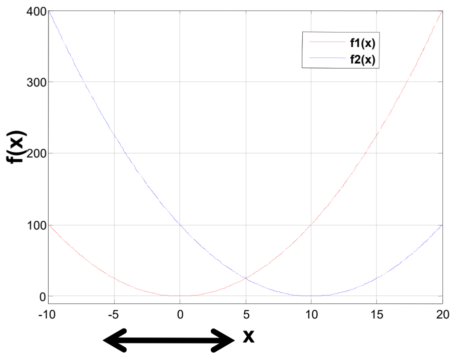
Minimize

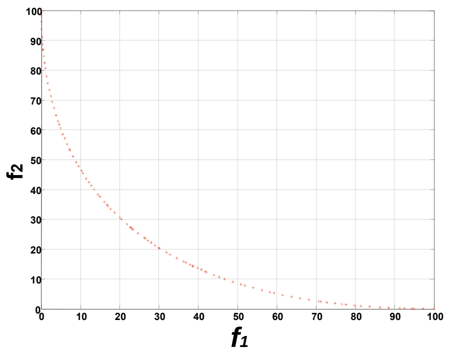
$$f_1(x) = x^2, \qquad f_2(x) = (x-10)^2;$$

where $-10 < x < 20$

Optimal solution:-

$$x \in [0,10]$$





NTNU

Formal Definition

- A multi-objective optimization problem has a number of objective functions which are to be minimized or maximized.

<u>General form of multi-objective optimization:</u>

Minimize/maximize	$f_m(x)$,	$m = 1,2,3,\ldots\ldots..,m;$
Subject to	$g_i\ (x) \geq 0$,	$j = 1,2,3,\ldots\ldots\ldots.,j;$
	$h_k(x) = 0$,	$k = 1,2,3,\ldots\ldots\ldots,k;$
	$x_i^{(L)} \leq x_i \leq x_i^{(U)}$,	$i = 1,2,3,\ldots\ldots\ldots.,n$

- A solution x that does not satisfy all of the $(J + K)$ constraints and all of the $2N$ Variable bounds is called an infeasible solution.
- On the other hand, if any solution x satisfies all the constraints and variable bounds, it is called a feasible solution.

Why Multi-Objective Optimization?

- In some optimization problems, it is not possible to find a way to link competing objectives.
- Sometimes the differences are qualitative and the relative importance of these objectives can’t be numerically quantified.

Conceptual Example

- Suppose you need to fly on a long trip:
  - Should you choose the **cheapest ticket** (more connections) or **shortest flying time** (more expensive)?
- It is impossible to put a value on time, so these two objectives can't be linked.
- Also, the relative importance will vary.
  - There may be a business emergency you need to go fix quickly.
  - Or, maybe you are on a very tight budget.

NTNU

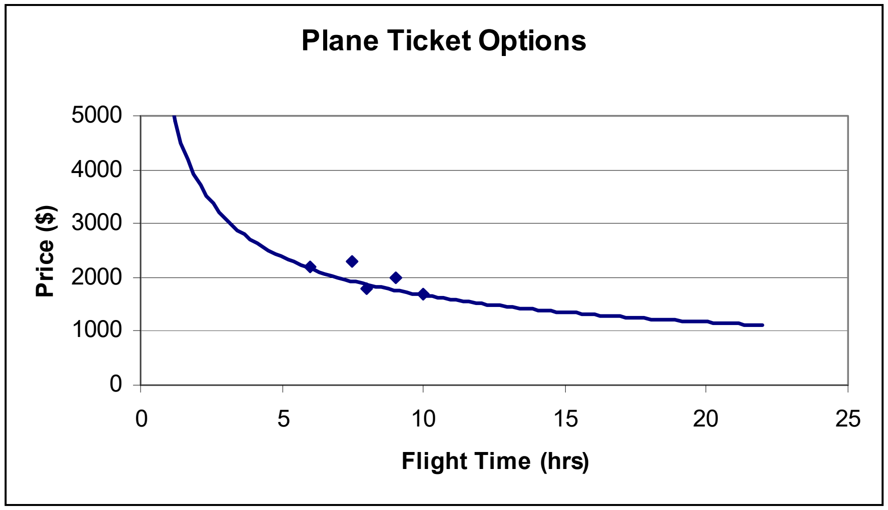
Example

- Airplane-Trip Tickets (Travel Time vs. Price):

Ticket	Travel Time (hrs)	Ticket Price ($)
A	10	1700
B	9	2000
C	8	1800
D	7.5	2300
E	6	2200

Multiple Solutions

- A MOOP will have many solutions in the feasible region.
- Even though we may not be able to assign numerical relative importance to multiple objectives, we can still classify some possible solutions as better than others.



Example

- Airplane-Trip Tickets (Travel Time vs. Price):

Ticket	Travel Time (hrs)	Ticket Price ($)
A	10	1700
B	9	2000
C	8	1800
D	7.5	2300
E	6	2200

A, C, E

Comparison of Solutions

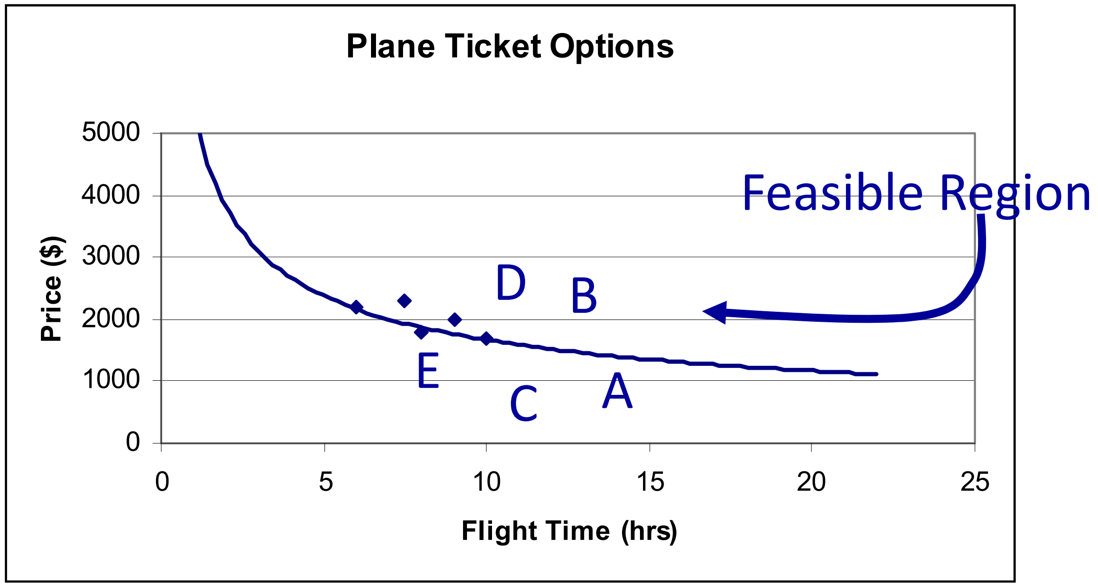
- If we compare tickets **A & B**, we can't say that either is superior without knowing the relative importance of Travel Time vs. Price.
- However, comparing tickets **B & C** shows that **C** is better than **B** in both objectives, so we can say that **C** "*dominates*" **B**.
- So, as long as **C** is a feasible option, there is no reason we would choose **B**.

Comparison of Solutions

- If we finish the comparisons, we also see that **D** is dominated by **E**.
- The rest of the options (**A, C, & E**) have a trade-off associated with Time vs. Price, so none is clearly superior to the others.
- We call this the “*non-dominated*” set of solutions become none of the solutions are dominated.

Graph of Solutions

Usually, solutions of this type form a typical shape, shown in the chart below:



Solution to MOOP

- The solution to MOOP consists of sets of **trade-offs between objectives**.
- The goal of MOO algorithms is to generate these trade-offs.
- Exploring all these trade-offs is particularly important because it provides the system designer/operator with the ability to understand and weigh the different choices available to them.

Traditional Approaches

- Weighted Sum Method.
- Lexicographic Ordering Method.
- The Ɛ-Constraint Method.

Weighted Sum Method

- Multiple objectives are combined into a single objective using weighted co-efficients.

$$Minimize \quad : \{ f_1(\vec{x}),\ f_2(\vec{x}), \dots \dots ,\ f_m(\vec{x}) \}$$

- Formulate as a single objective with weighted sum of all objective functions:

$$g(\vec{x}) = \lambda_1 f_1(\vec{x}) + \lambda_2 f_2(\vec{x}) + \dots \dots + \lambda_m f_m(\vec{x})$$

  where $\lambda_1, \lambda_2, \dots \dots , \lambda_m$ are weights values $\lambda_1 + \lambda_2 + \dots \dots + \lambda_m = 1$

  and *m* represents the number of objective functions.

- Problem is then treated as a single objective problem.

Weighted Sum Method (Limitation)

- Relative weights of the objectives are not exactly known in advance.
  - Objective function that has the largest variance value may dominate the multi-objective evaluation.
- Some solutions may be missed.
- A single solution is obtained at one time.
  - Multiple runs of the algorithm are required in order to get the whole range of solutions.
- Difficult to select proper combination of weights.
- Selection of weights depends on user and this restricts the final varies from user to user.

Lexicographic Ordering Method

- User is asked to rank the objectives in order of importance.
- The optimum solution is then obtained by minimizing the objective functions, starting with the most important one and proceeding according to the assigned order of importance of the objectives.
- It is also possible to select randomly a single objective to optimize at each run of a GA.

Lexicographic Ordering Method (Limited)

- Requires a pre-defined ordering of objectives and its performance will be affected by it.
- Selecting randomly an objective is equivalent to a weighted combination of objectives.
- Inappropriate when there is a large amount of objectives.

Ɛ-Constraint Method

- This method is based on optimization of one (the most preferred or primary) objective function, and considering the other objectives as constraints bound by some allowable levels ε_i.
- Hence, a single objective optimization is carried out for the most relevant objective function subject to additional constraints on the other objective functions.
- The levels ε_i are then altered to generate the entire Pareto-Optimal set.

Ɛ-Constraint Method (Limitation)

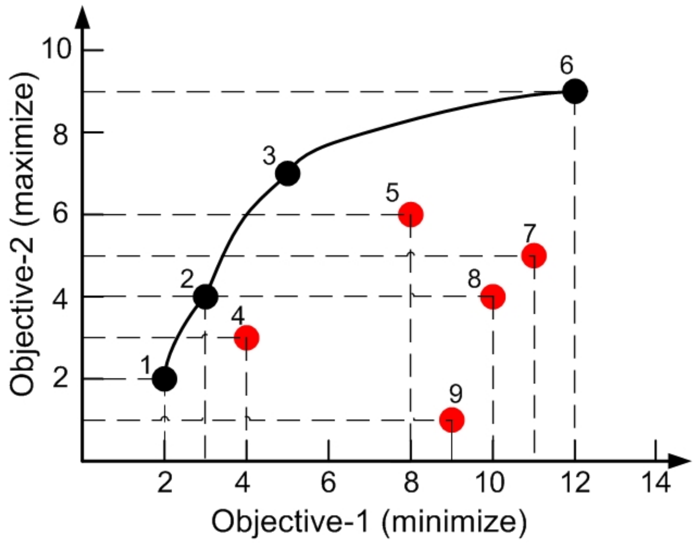
- Potentially high computational cost (many runs may be required).

Pareto-Optimality

- Pareto optimization can handle these situations very efficiently.
- The term domination is used to find the trade-offs solutions.
- A solution $x^{(1)}$ is said to *dominate* the other solution $x^{(2)}$, if both the following conditions are true:

  (1) The solution $x^{(1)}$ is no worse than $x^{(2)}$ in all objectives.

  (2) The solution $x^{(1)}$ is strictly better than $x^{(2)}$ in at least one objective.



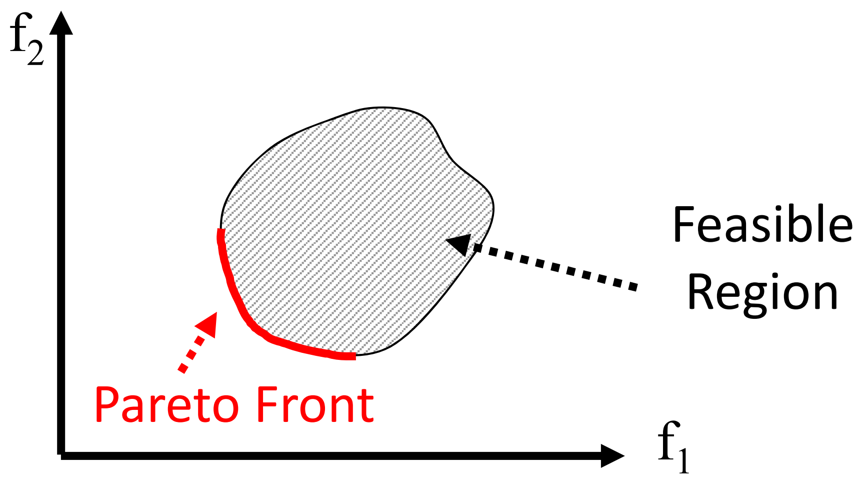
NTNU

Pareto-Optimal Solutions

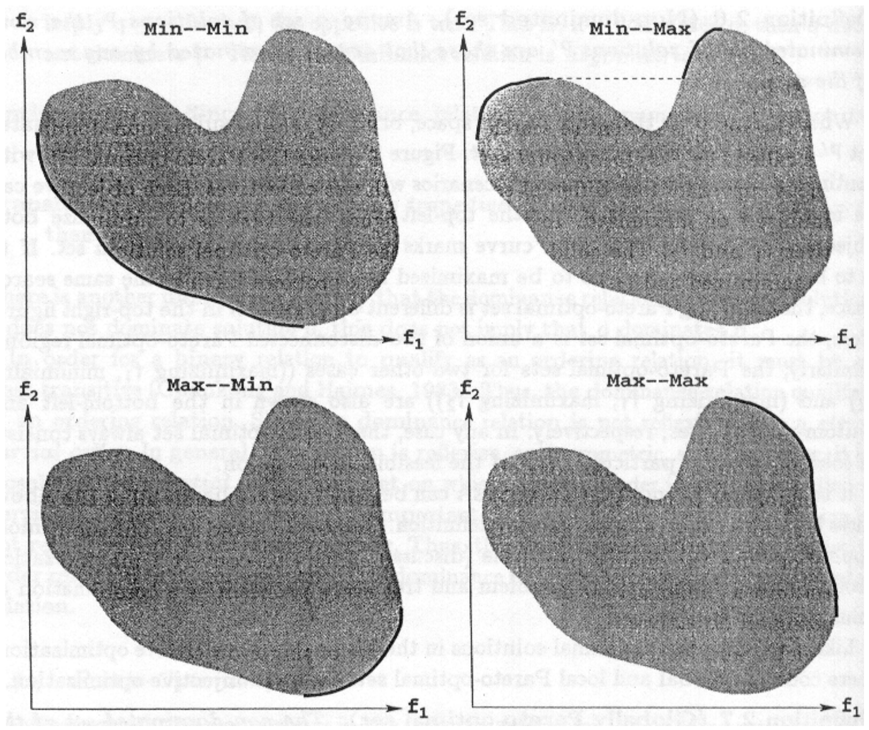
- The line is called the **Pareto front** and solutions on it are called **Pareto-Optimal**.
- All Pareto-optimal solutions are non-dominated.
- If x^1 dominates x^2 ($x^1 \leq x^2$), it can be said that
  - x^2 is dominated by x^1
  - x^1 is non-dominated by x^2
  - x^1 is non-inferior to x^2 (Deb)

Graphical Example

For the following feasible region with objectives f_1 & f_2 where both f_1 & f_2 are minimized:

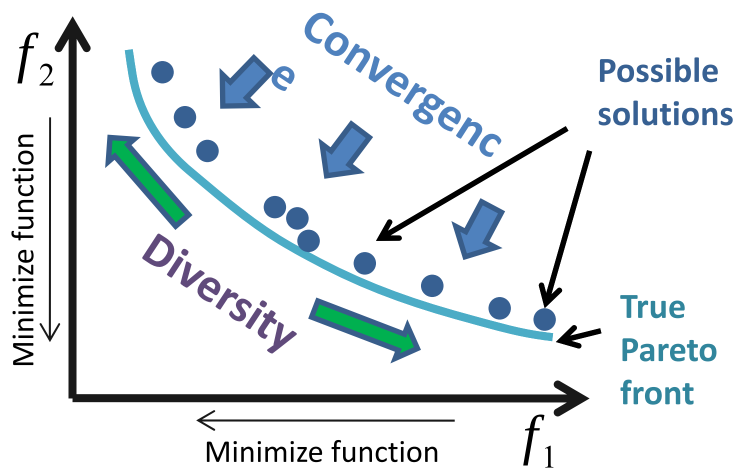


Pareto-Optimality Example



NTNU

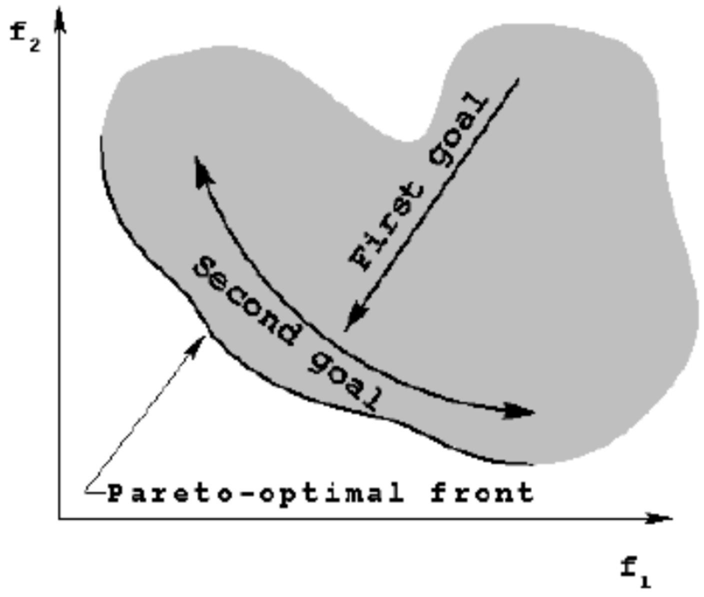
Objectives in MOOP



- To find a set as close as possible to the Pareto-optimal front (**Convergence**).
- To find a set of solutions as diverse as possible (**Diversity**).
    - Representation of the entire Pareto-Optimal front.

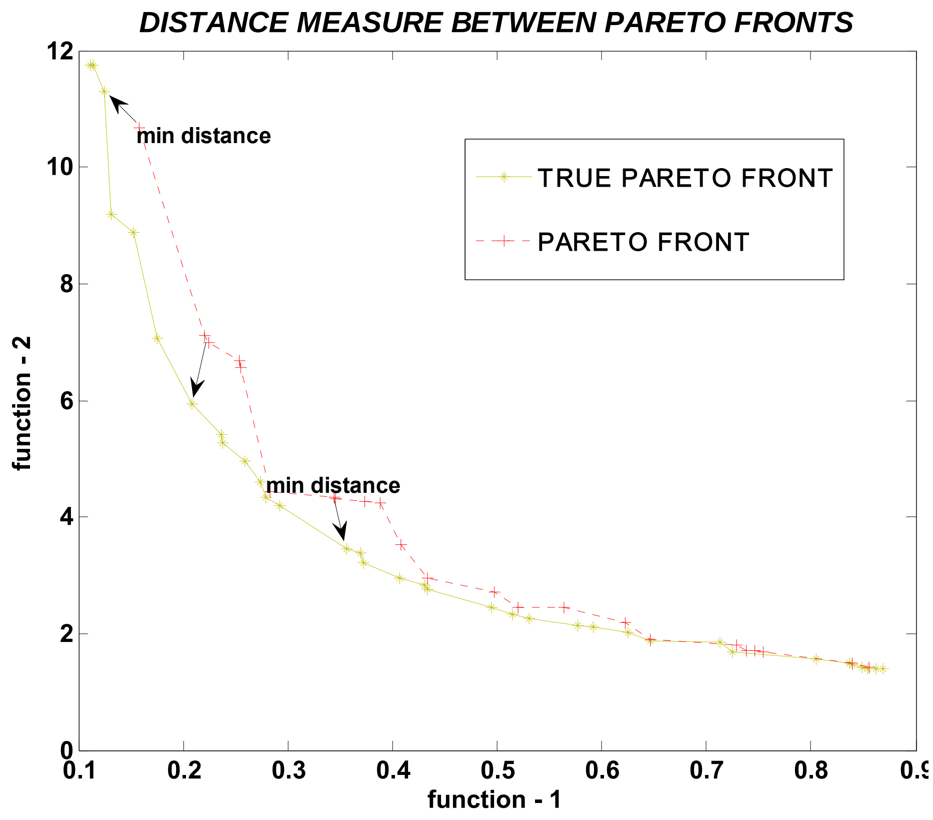
Performance Metrics

- To quantify the performance of MOEA according two essential metrics dictated by Pareto-Optimality.
  - Convergence measure
  - Diversity measure



Convergence Metrics (One Example)

$$\gamma = avg\left(\sum_{N} (\min\_distance)\right)$$

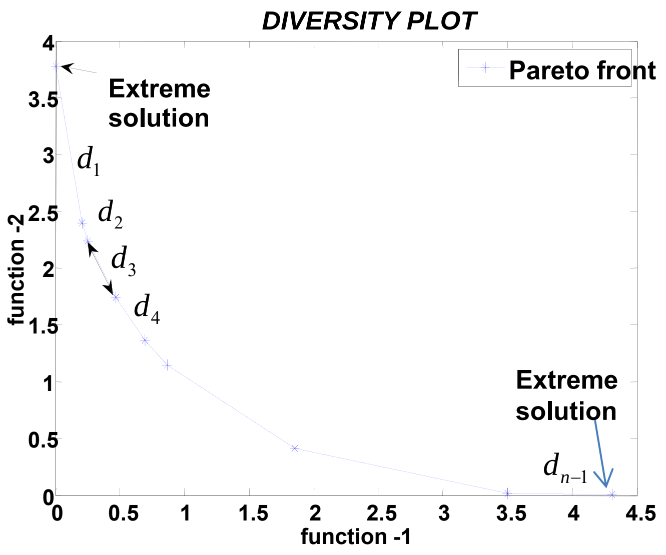


NTNU

Diversity Metrics (One Example)

$$\Delta = \frac{d_f + d_l + \sum_{i=1}^{N-1} \left| (d_i - \bar{d}) \right|}{d_f + d_l + (N-1)\bar{d}}$$

- ➢ If distance between the solutions d_i is equal to average distance $\bar{d}$, that gives uniformly distribution.



The parameters d_f and d_l are the Euclidean distances between the extreme solutions of true Pareto front and the boundary solutions of the obtained non-dominated set.

The parameter $\bar{d}$ is the average of all distances , d_i ,$i = 1,2, \ldots . N - 1$ assuming that there are N solutions on the best non- dominated front.

Why EAs in MO Optimization?

- EAs deal simultaneously with a set of possible solutions (the so-called population).
- This allows us to find several members of the Pareto-Optimal set in a single run of the algorithm
  - instead of having to perform a series of separate runs as in the case of the traditional techniques.
- EAs are less susceptible to the shape or continuity of the Pareto front (e.g., they can easily deal with discontinuous or concave Pareto fronts),
  - whereas these two issues are a real concern for mathematical programming techniques.

NTNU

Why EAs in MO Optimization?

- For MOOP, EAs can yield whole set of potentials solutions, which are all optimal in some sense.
- Some of the other advantages of having EAs is that they are easy to implement, robust and could be implemented in a parallel environment.

MOEA: Early Suggestions

- **Vector Evaluated Genetic Algorithm: VEGA** (Schaffer, 1984)
  - First real implementation of MOEA.
  - Independent selection cycles according to each objective.
  - Worked efficiently for some generations.
  - It some cases suffered from its bias towards individual objective champions.

NTNU

MOEA: Early Suggestions

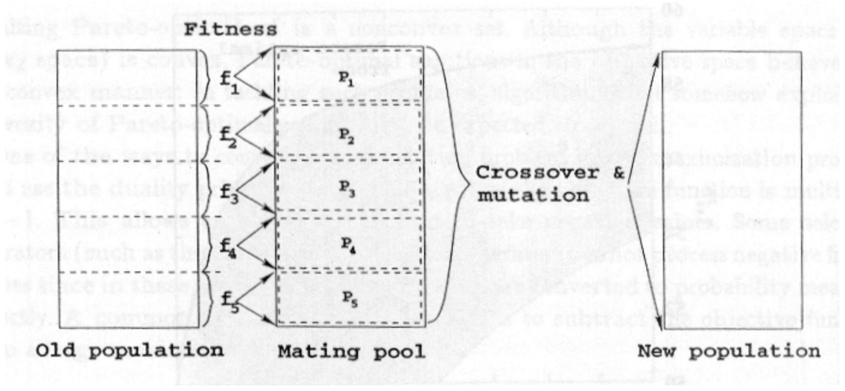
- **Goldberg, 1989**
  - Suggested the **concept of domination**.
  - Use of **niching strategy** among solutions of a non-dominated class.
  - Had impact on MOGA, NPGA, NSGA.
    - These algorithms differ in the way a fitness is assigned to each individual.

VEGA

- Simplest, straightforward extension of GA.
- First multi-objective GA to find a set of non-dominated solutions.
- **Only the selection mechanism of the GA is modified.**
- Evaluated an **objective vector** instead of a scalar objective function.
  - Each element of the vector represents each objective function.
- At each generation **a number of sub-populations is generated by performing proportional selection according to each objective function** in turn.

VEGA

- For a problem with k objectives and a population size of M, k sub-populations of size M/k each would be generated.
- These sub-populations would be shuffled together to obtain a new population of size M, on which the GA would apply the crossover and mutation operators in the usual way.
  - To reduce positional bias in the population, it is better to shuffle the population before it is partitioned into equal subpopulations.
- **Selection operator is restricted only within a sub-population**.



NTNU

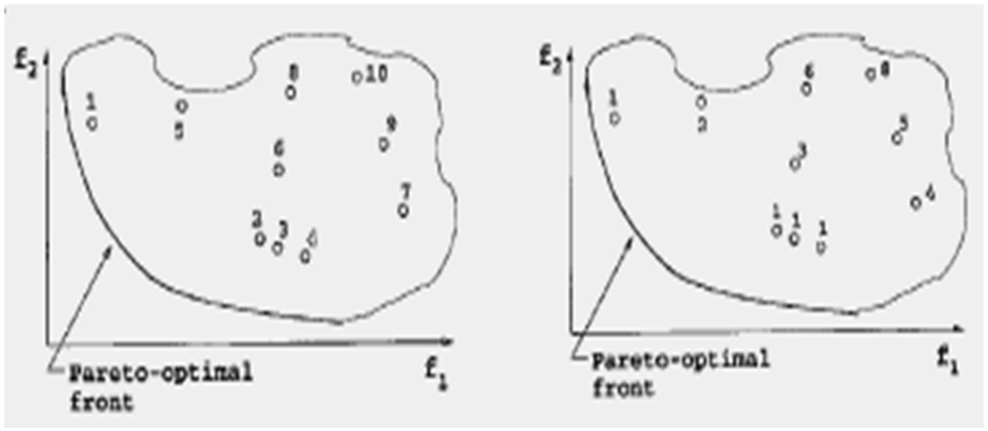
Advantages and Disadvantages of VEGA

- ✓ Efficient and easy to implement.
- × Every solution is tested for only one objective.
  - × Not tested for other $(M\text{-}1)$ objectives.
- × Hard to maintain a good spread solutions in the obtained front.

Multi-Objective Genetic Algorithm (MOGA)

- Fonseca and fleming (1993) first introduced a multi-objective GA which used *non-dominated* classification of a GA population, and simultaneously maintained *diversity* in the non-dominated solutions.
- Each solution is checked for its domination in the population. A *rank* equal to one plus the number of solutions n_i that dominates the solution i is assigned to solution i.
- In order to maintain diversity among non-dominated solutions, *niching* has been introduced among solutions of each rank.

Ranking Procedure of MOGA



(a) A two-objective search space and corresponding Pareto-Optimal front (b) Ranking of the individual

- There must be at least one solution with rank equal to *1*.
- The maximum number of any solution cannot be more than *N* (the population size).
- May not assign all possible ranks (between *1* and *N*) to any population.

Niching (Sharing Function Model)

- To introduce diversity among solutions of a population.
- Focusing on **degrading the fitness** of similar solutions.

<u>Shared fitness calculation</u>:

- Calculate the sharing function value $Sh(d_{ij})$ with all population member by using

$$\mathrm{Sh}(d) = \begin{cases} 1 - \left(\frac{d}{\sigma_{\mathrm{share}}}\right)^{\alpha}, & \text{if } d \leq \sigma_{\mathrm{share}}; \\ 0, & \text{otherwise.} \end{cases}$$

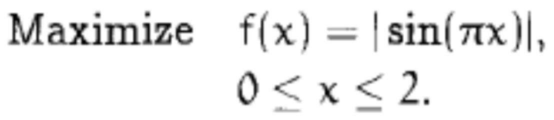
  d is the distance between two solutions.

- Add all sharing function value to calculate the niche count nc_i by using

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

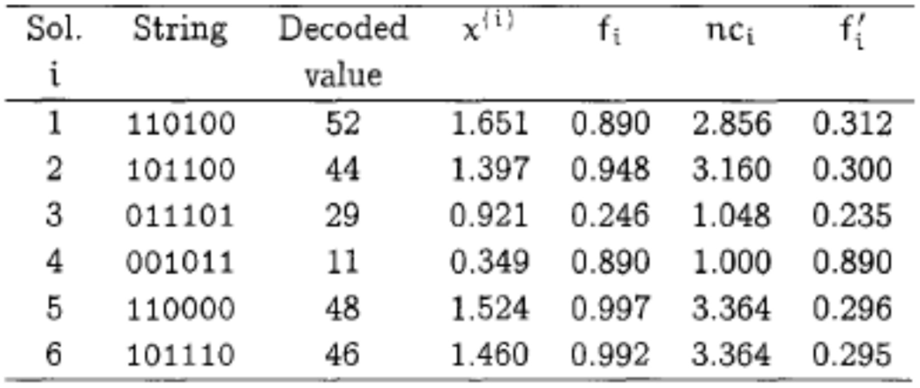
- Calculate the shared fitness value as $f'_i = f_i / nc_i$

Example *



Maximize $f(x) = |\sin(\pi x)|$,

$0 \le x \le 2.$



Sol. i	String	Decoded value	$x^{(i)}$	f_i	nc_i	f'_i
1	110100	52	1.651	0.890	2.856	0.312
2	101100	44	1.397	0.948	3.160	0.300
3	011101	29	0.921	0.246	1.048	0.235
4	001011	11	0.349	0.890	1.000	0.890
5	110000	48	1.524	0.997	3.364	0.296
6	101110	46	1.460	0.992	3.364	0.295

Assume:

$\sigma_{share} = 0.5$

$\alpha = 1$

* Example taken from Deb, 2001.

Example

Step 1 From the first solution, the distances from all population members are as follows:

$$d_{11} = 0, \quad d_{12} = 0.254, \quad d_{13} = 0.731, \quad d_{14} = 1.302,$$
$$d_{15} = 0.127, \quad d_{16} = 0.191.$$

The corresponding sharing function values are calculated by using equation (4.59):

$$\mathrm{Sh}(d_{11}) = 1, \quad \mathrm{Sh}(d_{12}) = 0.492, \quad \mathrm{Sh}(d_{13}) = 0, \quad \mathrm{Sh}(d_{14}) = 0,$$
$$\mathrm{Sh}(d_{15}) = 0.746, \quad \mathrm{Sh}(d_{16}) = 0.618.$$

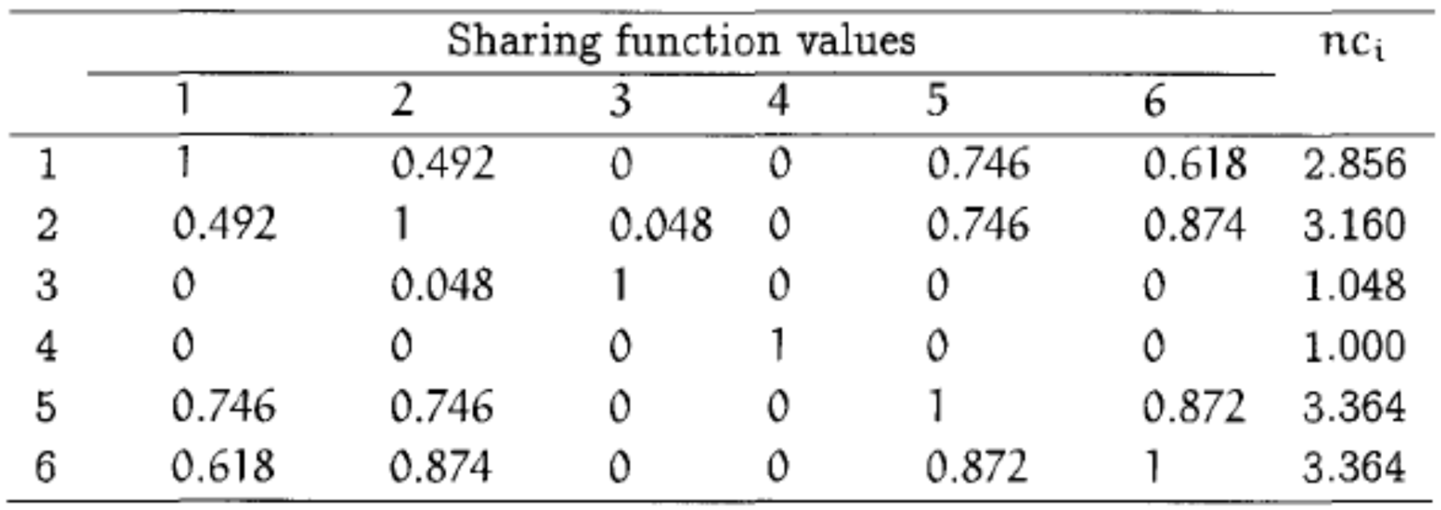
Note that since solutions 3 and 4 are more than a 0.5 unit away from solution 1, their sharing effect is zero.

Step 2 The niche count of the first solution is simply the addition of the above sharing function values, or:

$$nc_1 = 1 + 0.492 + 0 + 0 + 0.746 + 0.618 = 2.856.$$

NTNU

Example



	Sharing function values						nc_i
	1	2	3	4	5	6	
1	1	0.492	0	0	0.746	0.618	2.856
2	0.492	1	0.048	0	0.746	0.874	3.160
3	0	0.048	1	0	0	0	1.048
4	0	0	0	1	0	0	1.000
5	0.746	0.746	0	0	1	0.872	3.364
6	0.618	0.874	0	0	0.872	1	3.364

Example

Step 3 The shared fitness value of the first solution is:

$$f'_1 = f(x^{(1)})/nc_1 = 0.890/2.856 = 0.312.$$

Sol. i	String	Decoded value	$x^{(i)}$	f_i	nc_i	f'_i
1	110100	52	1.651	0.890	2.856	0.312
2	101100	44	1.397	0.948	3.160	0.300
3	011101	29	0.921	0.246	1.048	0.235
4	001011	11	0.349	0.890	1.000	0.890
5	110000	48	1.524	0.997	3.364	0.296
6	101110	46	1.460	0.992	3.364	0.295

- Calculation of the shared fitness value: f'_i / f'_{avg}
- For example: f'_4 / f'_{avg}

$$= 0.890/ ((0.312 + 0.3 + 0.235 + 0.890 + 0.296 + 0.295) / 6)$$

$$= 2.29$$

NTNU

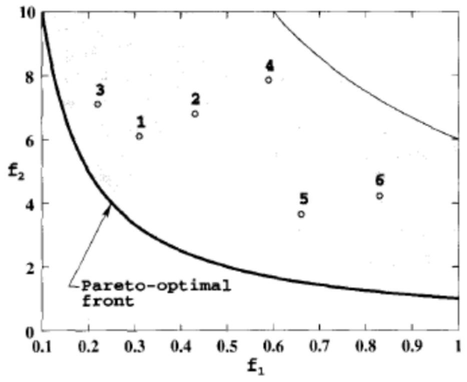
MOGA: Step by Step with Example (Deb, 2001)

- Two objective (minimization) optimization problem:

$$\text{Min-Ex:} \quad \begin{cases} \text{Minimize} & f_1(\mathbf{x}) = x_1, \\ \text{Minimize} & f_2(\mathbf{x}) = \dfrac{1+x_2}{x_1}, \\ \text{subject to} & 0.1 \le x_1 \le 1, \\ & 0 \le x_2 \le 5. \end{cases}$$

- Six-membered population:

Solution	x_1	x_2	f_1	f_2
1	0.31	0.89	0.31	6.10
2	0.43	1.92	0.43	6.79
3	0.22	0.56	0.22	7.09
4	0.59	3.63	0.59	7.85
5	0.66	1.41	0.66	3.65
6	0.83	2.51	0.83	4.23



NTNU

MOGA: Step by Step with Example

- **Step 1**

  Choose a α_{share} .

  Initialize $\mu(j)$ = 0 for all possible ranks $j = 1, 2 ,\ldots.., N$.

  Set solution counter $i = 1$.

Example:

α_{share} = 0.5

$\mu(j)$ = 0 for all $j = 1$ to 6 (Population size = $N = 6$)

MOGA: Step by Step with Example

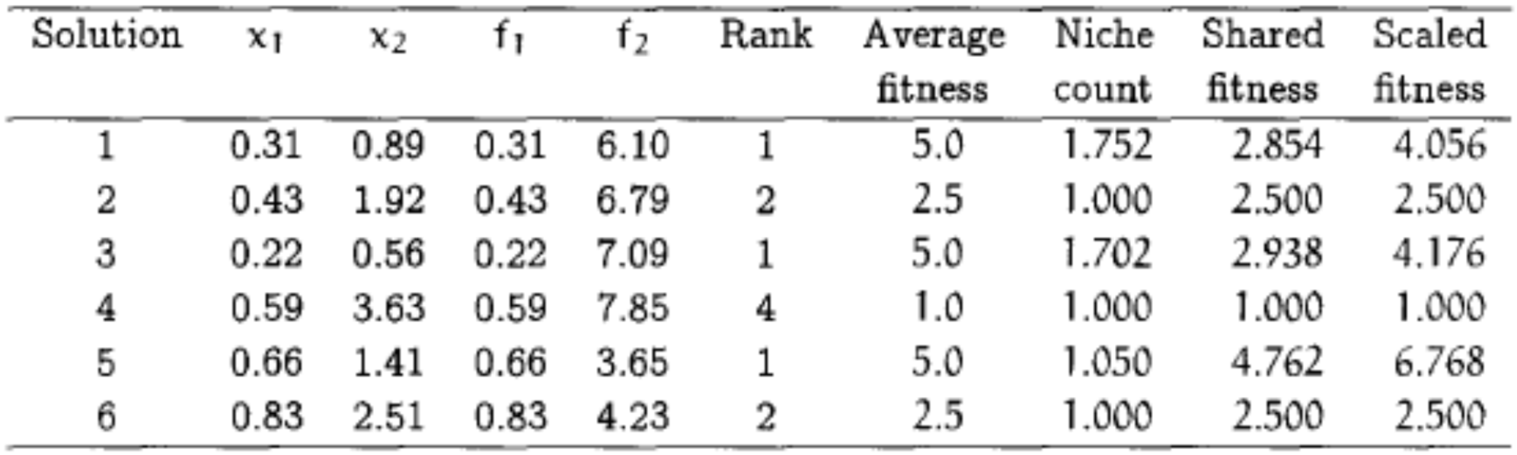
Step 2 Calculate the number of solutions (n_i) that dominates solution i. Compute the rank of the i-th solution as $r_i = 1 + n_i$. Increment the count for the number of solutions in rank r_i by one, that is, $\mu(r_i) = \mu(r_i) + 1$.

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

Example:

Steps 2 and 3 For solution 1, we find that $n_1 = 0$, and thus, $r_1 = 1 + 0 = 1$. Similarly, we find r_i for all other solutions and list them in Table 16. We observe that there are three solutions with rank 1 (thus, $\mu(1) = 3$), two solutions with rank 2 ($\mu(2) = 2$) and one solution with rank 4 ($\mu(4) = 1$). Therefore, there is no solution in rank 3, 5 and 6, or $\mu(3) = \mu(5) = \mu(6) = 0$.

MOGA: Step by Step with Example



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

MOGA: Step by Step with Example

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 $i = 1, \ldots, N$:

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

Example:

Step 4 The maximum rank r^* assigned to any solution in the population is 4. Sorting of the population according to rank yields the following sequence (Table 16):

(1,3,5) (2,6) ( ) (4).

NTNU

MOGA: Step by Step with Example

$$F_1 = N - 0.5(\mu(1) - 1) = 6 - 0.5(3 - 1) = 5.0,$$

$$F_2 = N - \mu(1) - 0.5(\mu(2) - 1) = 6 - 3 - 0.5(2 - 1) = 2.5,$$

$$F_3 = N - 0.5(\mu(1) - 1) = 6 - 0.5(3 - 1) = 5.0,$$

$$\begin{aligned} F_4 &= N - \mu(1) - \mu(2) - \mu(3) - 0.5(\mu(4) - 1) \\ &= 6 - 3 - 2 - 0 - 0.5(1 - 1) = 1.0, \end{aligned}$$

$$F_5 = N - 0.5(\mu(1) - 1) = 6 - 0.5(3 - 1) = 5.0,$$

$$F_6 = N - \mu(1) - 0.5(\mu(2) - 1) = 6 - 3 - 0.5(2 - 1) = 2.5.$$

It is seen that all three solutions of the first rank has the same average fitness of 5.0.

MOGA: Step by Step with Example

Step 5 For each solution i in rank r, calculate the niche count nc_i with other solutions ($\mu(r)$ of them) of the same rank by using equation (5.15). Calculate the shared fitness using $F'_j = F_j / nc_j$. To preserve the same average fitness, scale the shared fitness as follows:

Scaling factor: $$F'_j \leftarrow \frac{F_j \mu(r)}{\sum_{k=1}^{\mu(r)} F'_k} F'_j.$$

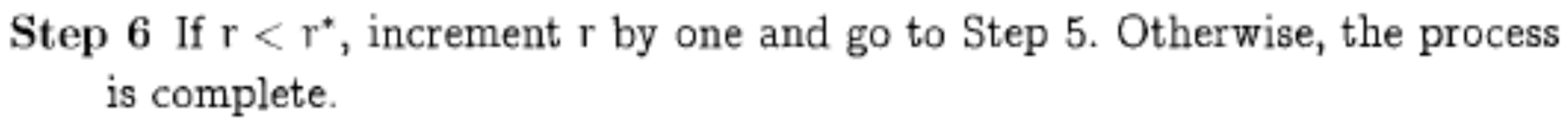
Example:

- ***Niche count*** and ***shared fitness***: Earlier slides.
- ***Scaling factor***: $F_1 \mu(1)/(F'_1 + F'_3 + F'_5)$

  $5 \times 3/(2.854 + 2.938 + 4.762)$

  $= 1.421$
- Multiply column *9* by *1.421* to calculate the scaled fitness values of solutions *1*, *3*, and *5*.

MOGA: Step by Step with Example



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

Example:

- Move to Step 5 with solutions of **rank 2** and then solutions of **rank 3**.

MOGA (Advantages and Disadvantages)

Advantages:

- ✓ The fitness assignment is simple in MOGA.
- ✓ Since niching is performed in the objective space, MOGA can be easily applied to other optimization problems.
- ✓ If a **spread** of Pareto-optimal solutions is required on the objective space, MOGA is the suitable choice.

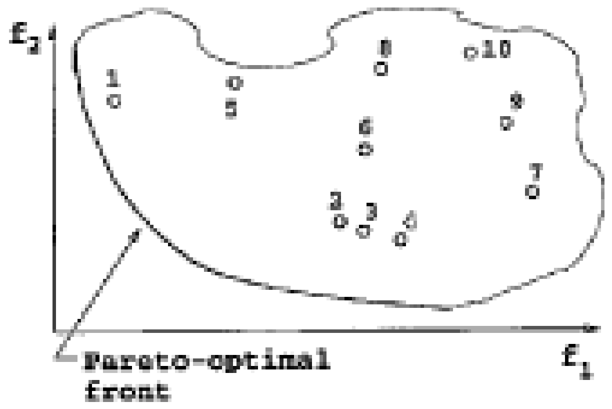
Disadvantages:

- ➢ Like other GAs which use niche counts to maintain diversity, the σ_{share} parameter needs fixing.
- ➢ The shared fitness computation procedure does not make sure that a solution in a poorer rank will always have a worse scaled fitness f' than every solution in a better rank.
- ➢ It may cause a slow convergence.

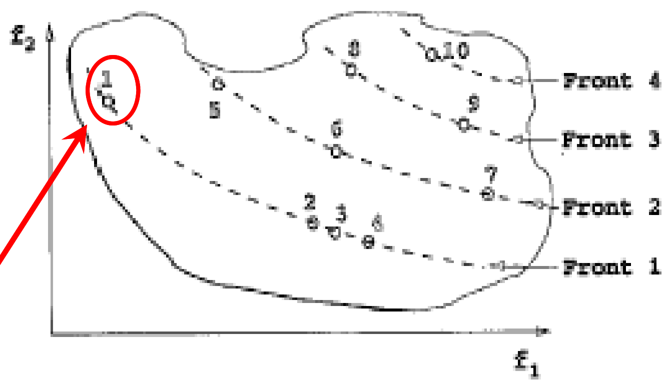
Non-Dominated Sorting Genetic Algorithm(NSGA)

- Srinivas and Deb (1994) used Goldberg's (1989) idea of using the **non-dominated sorting** concept.
- Dual objectives of a MO optimization are maintained by
  - Assigning more fitness to better non-dominated set which ensures a selection pressure toward the Pareto-Optimal front.
  - By degrading the assigned fitness based on the number of neighboring solutions (**for diversity**).
- The fitness sharing method is used **front-wise**.
- The sharing distance is calculated with decision variables, instead of the objective function values as in MOGA.

Necessity for Diversity Preservation



(a) A two-objective search space



(b) Ten solutions are classified into different non-domination front

What will happen, if lost?

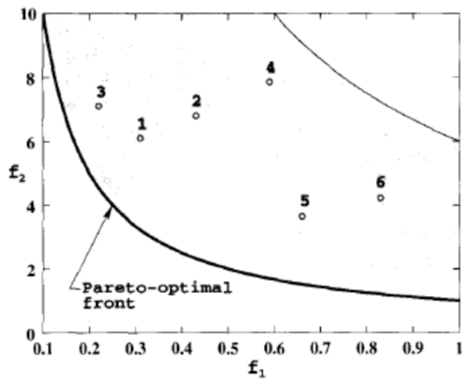
NSGA: Step by Step with Example (Deb, 2001)

- Two objective (minimization) optimization problem:

$$\text{Min-Ex:} \begin{cases} \text{Minimize} & f_1(\mathbf{x}) = x_1, \\ \text{Minimize} & f_2(\mathbf{x}) = \dfrac{1+x_2}{x_1}, \\ \text{subject to} & 0.1 \le x_1 \le 1, \\ & 0 \le x_2 \le 5. \end{cases}$$

- Six-membered population:

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



Pareto-Optimal front for $0.1 \le x_1^* \le 1$ and $x_2^* = 0$.

NSGA: Step by Step with Example

Fitness Assignment

Step 1 Choose sharing parameter α_{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:

$$(P_1, P_2, \ldots\ldots, P_\rho) = Sort\ (P, \leq).$$

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 among solutions of P_j only.

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 $j \leq \rho$, go to Step 3. Otherwise, the process is complete.

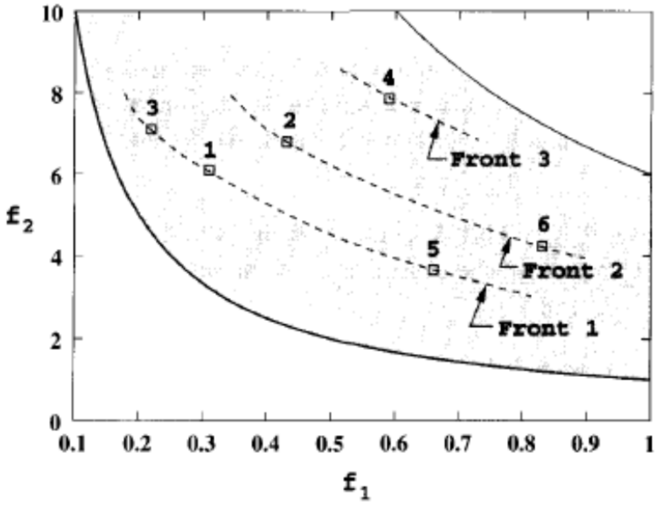
NSGA: Step by Step with Example

Step 1 We choose $\sigma_{\text{share}} = 0.158$

$\epsilon = 0.22$

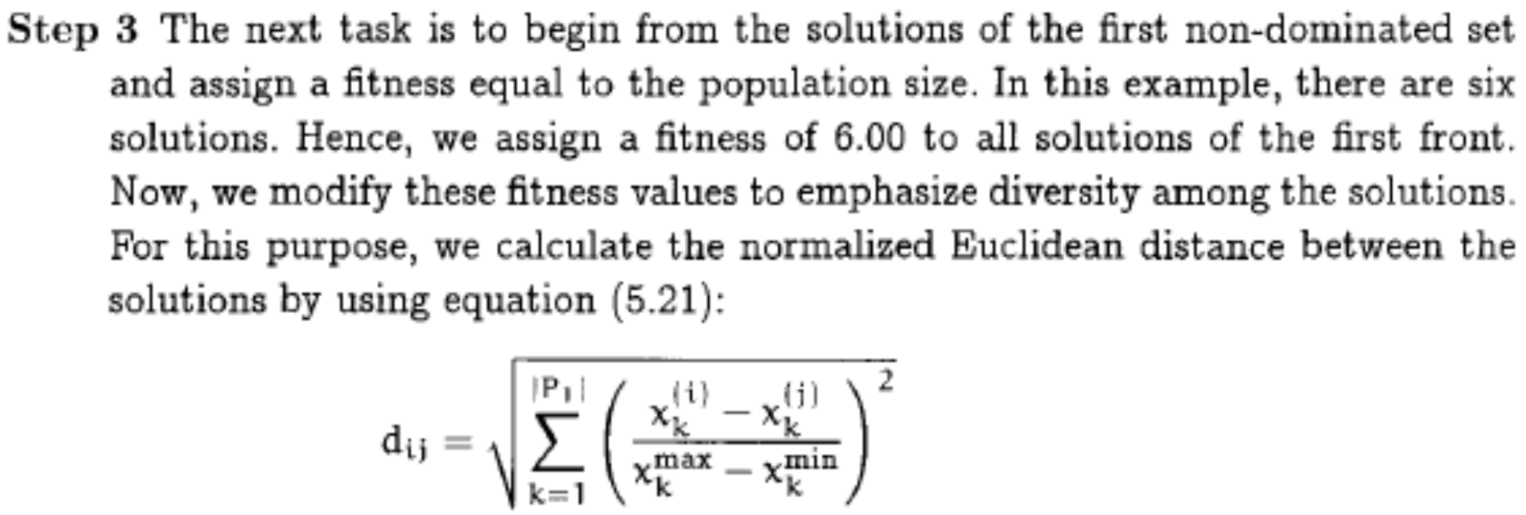
$F_{\min} = 6.22$ and $j = 1$

Step 2 Next, we classify the population into different non-domination sets.



NTNU

NSGA: Step by Step with Example



Step 3 The next task is to begin from the solutions of the first non-dominated set and assign a fitness equal to the population size. In this example, there are six solutions. Hence, we assign a fitness of 6.00 to all solutions of the first front. Now, we modify these fitness values to emphasize diversity among the solutions. For this purpose, we calculate the normalized Euclidean distance between the solutions by using equation (5.21):

$$d_{ij} = \sqrt{\sum_{k=1}^{|P_1|} \left(\frac{x_k^{(i)} - x_k^{(j)}}{x_k^{\max} - x_k^{\min}} \right)^2}$$

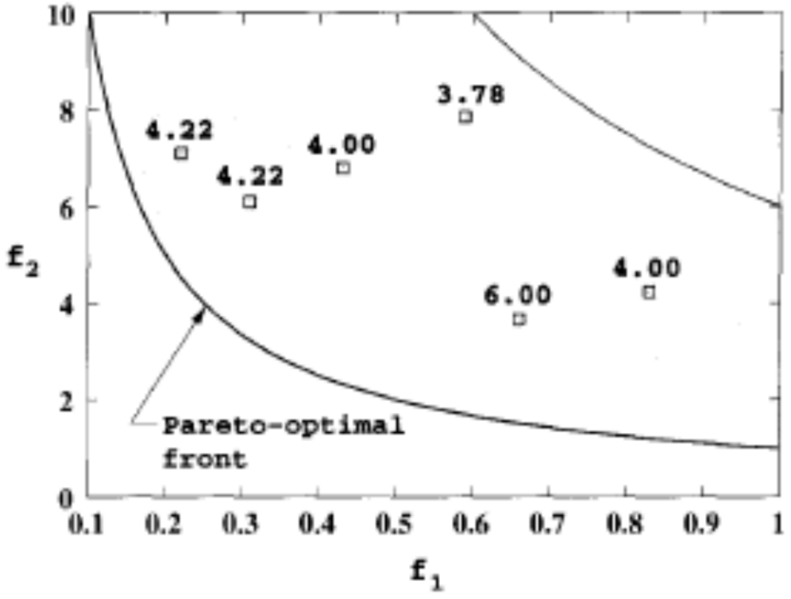
After calculating the sharing function (as earlier) and niche count (replacing N with the number of non-dominated front members), we get:

$$nc_1 = 1.423$$
$$nc_3 = 1.423$$
$$nc_5 = 1.000$$

NSGA: Step by Step with Example

Step 4 Now, $F_{min} = 4.22$. We set $j = 2$.

Step 5 We continue the above procedure for assigning fitness to solutions in the second and third fronts. Corresponding shared fitness values are shown in figure.



NSGA (Advantages and Disadvantages)

Advantages:

- ✓ Its main advantage is the assignment of fitness according to non dominated sets.
- ✓ No solution in a front has a worse shared fitness value than a solution in a worse front.
  - ✓ This provides a selection pressure towards non-dominated solutions which would hopefully lead towards the Pareto-optimal solutions.
- ✓ Within a front, less dense solutions have better fitness values.
  - ✓ It ensures that diversity is maintained among the non-dominated solutions.

Disadvantages:

- × The sharing function approach requires fixing the parameter $\sigma_{share.}$
- × The performance of NSGA is sensitive to the parameter σ_{share}

Non Dominated Sorting based Genetic Algorithm II (NSGA- II)

- Developed by Prof. K. Deb at Kanpur Genetic Algorithms Laboratory (2002)
- Famous for *Fast non-dominated search*
- Fitness assignment - *Ranking based* on non-domination sorting
- It uses an explicit diversity-preserving mechanism *Crowding distance*
- Uses *Elitism*
  - The offspring population Q_t is first created by using parent population P_t. Instead of finding the non-dominated front of Q_t only, the two populations are combined together to form R_t of size $2N$. Then a non-dominated sorting is used to classify the entire population R_t.

Selection

- **Selection** is the stage of a GA in which individual are chosen from a population for later breeding (recombination or crossover).
- The crowding operator $\leq_n$ guides the selection process at the various stages of the algorithm toward a uniformly spread-out Pareto optimal front.

Crowding Distance Assignment

- It helps to maintain a *good spread* of solutions in the obtained set of solutions.
- It does not require any user-defined parameter for maintaining diversity among populations.

Crowding Tournament Selection Operator:

A solution i wins a tournament with another solution j if any of the following conditions are true:

1. If solution i has a better rank, that is, $r_i < r_j$.
2. If they have the same rank but solution i has a better (lower) *crowding distance* than solution j, that is, $r_i = r_j$ and $d_i > d_j$.

Crowding Distance Assignment

- The crowding distance d_i of a solution i is a measure of the search space around i which is not occupied by any other solution in the population.
- This quantity d_i serves as an estimate of the perimeter of the cuboid formed by using the nearest neighbors as the vertices.
- To get an estimate of the density of solutions surrounding a particular solution i in the population, the average distance of the two solutions on either side of solution i along each objectives are taken.
- *If both solutions are of the same front, then the solution located in less crowded region is selected.*

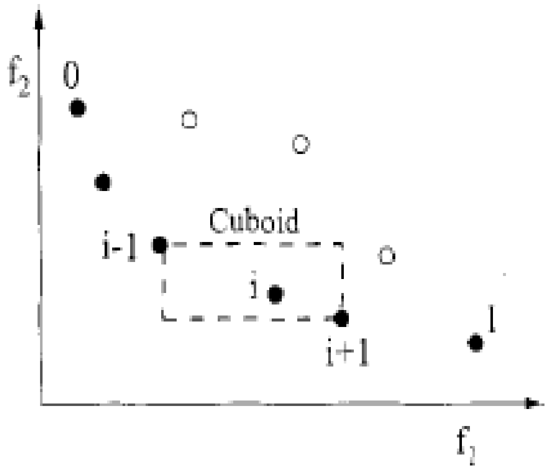


Fig: Crowding distance calculation. Points marked in filled circles are solutions of the same non-dominated front

Crowding Distance Assignment

- Choose individuals having large crowding distance.
- Help for obtaining uniformly distribution.

$$1_{C.D.} = l_{C.D.} = \infty,$$

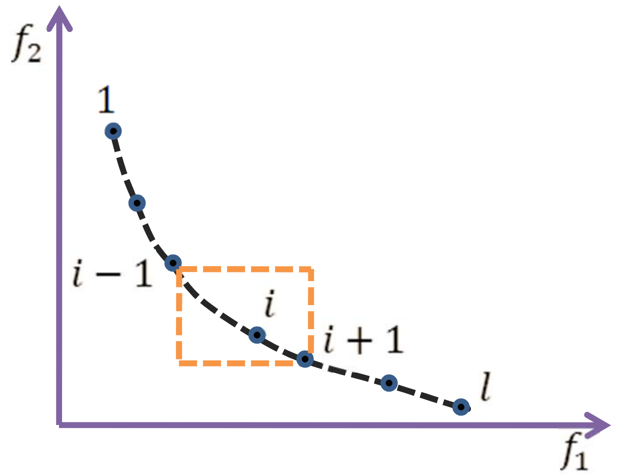
$$i_{C.D.} = \sum_m \left(\frac{f[i+1]_m - f[i-1]_m}{f_m^{\max} - f_m^{\min}} \right)$$

where

$i = 2, 3, \dots, (l-1)$

where $f[i]_m$ represent m^{th} objective function value of i^{th} solution.

and $f_m^{\max}$ is the maximum value of function $\vec{f}_m$ in the Pareto front.



NTNU

Initialize Population

Minimize

$$f_1(x) = x^2,\ \ f_2(x) = (x-2)^2$$

where $-5 \le x \le 5$

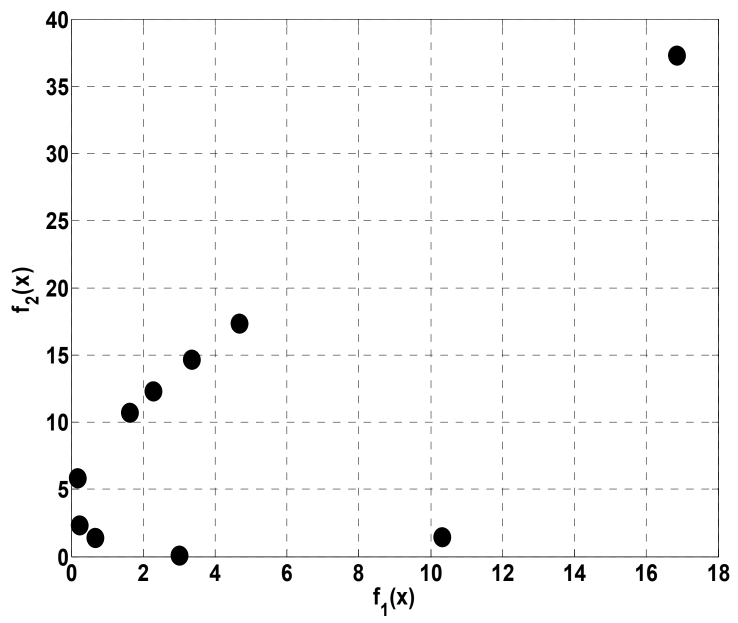
- Search space is of single dimension (given).
- Objective space is of two dimension (given).
- Let population size = ***10***
- Initialize population with 10 chromosomes having single dimensioned real value.
- These values are randomly distributed in between ***[-5, 5]***.

x
0.4678
1.7355
0.8183
-0.414
3.2105
-1.272
-1.508
-1.832
-2.161
-4.105

Evaluate Fitness values

- Find out all objective functions values (fitness values) for all chromosomes.

x	$f_1(x)$	$f_2(x)$
-0.414	0.171	5.829
0.467	0.218	2.347
0.818	0.669	1.396
1.735	3.011	0.07
3.210	10.308	1.465
-1.272	1.618	10.708
-1.508	2.275	12.308
-1.832	3.355	14.682
-2.161	4.671	17.317
-4.105	16.854	37.275



Fast Non-domination Sorting

- Assigning the rank to each individual of the population.
- Rank based on the *non-domination sorting* (front wise).
- It helps in selection and sorting.

x	$f_1(x)$	$f_2(x)$
-0.414	0.171	5.829
0.467	0.218	2.347
0.818	0.669	1.396
1.735	3.011	0.07
3.210	10.308	1.465
-1.272	1.618	10.708
-1.508	2.275	12.308
-1.832	3.355	14.682
-2.161	4.671	17.317
-4.105	16.854	37.275

Reference chromosome **Dominated chromosomes** **Counter**

$$x^1 = \{\dots\} \Rightarrow n^1 = 0$$

$$x^2 = \{\dots\} \Rightarrow n^2 = 0$$

$$x^3 = \{\dots\} \Rightarrow n^3 = 0$$

$$x^4 = \{\dots\} \Rightarrow n^4 = 0$$

$$x^5 = \{\underline{x^3, x^4}\} \Rightarrow n^5 = 1$$

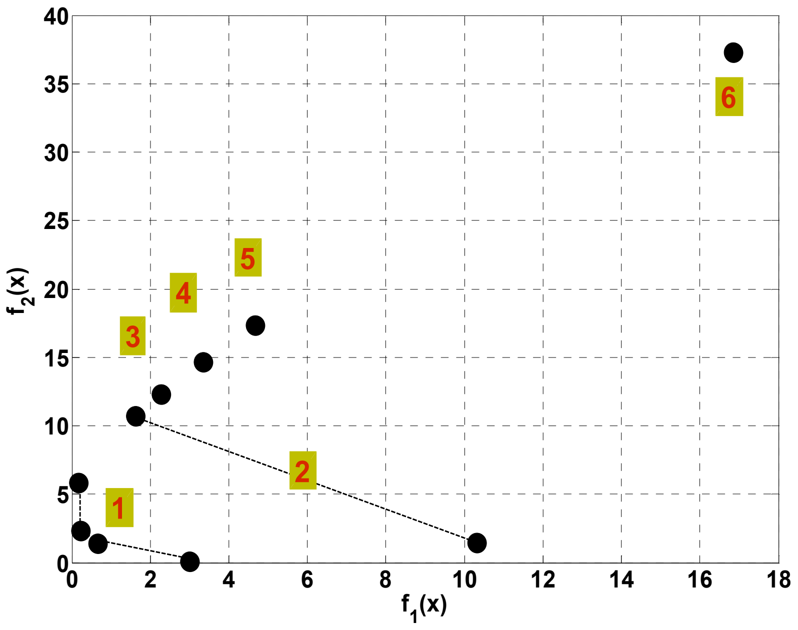
$$x^6 = \{\underline{x^1, x^2, x^3}\} \Rightarrow n^6 = 1$$

$$x^7 = \{\underline{x^1, x^2, x^3}, \underline{\underline{x^6}}\} \Rightarrow n^7 = 2$$

Rank
1
1
1
1
2
2
3
4
5
6

Fast Non-domination Sorting

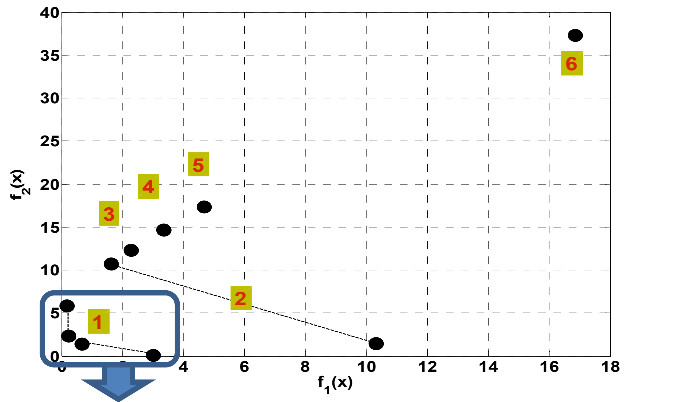
x	$f_1(x)$	$f_2(x)$	*Rank*
-0.414	0.171	5.829	1
0.467	0.218	2.347	1
0.818	0.669	1.396	1
1.735	3.011	0.07	1
3.210	10.308	1.465	2
-1.272	1.618	10.708	2
-1.508	2.275	12.308	3
-1.832	3.355	14.682	4
-2.161	4.671	17.317	5
-4.105	16.854	37.275	6

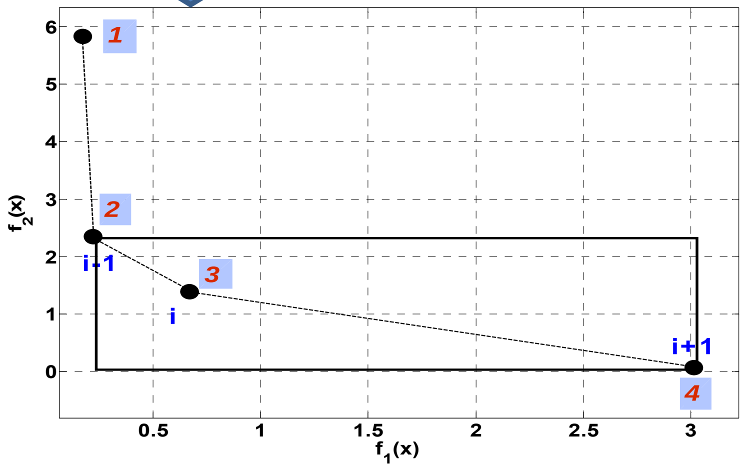


Crowding Distance Assignment

- Crowning distance can be calculated for all chromosomes of same Pareto front.

x	$f_1(x)$	$f_2(x)$	$Rank$	$C.D.$
-0.414	0.171	5.829	1	∞
0.467	0.218	2.347	1	0.945
0.818	0.669	1.396	1	1.378
1.735	3.011	0.07	1	∞
3.210	10.308	1.465	2	∞
-1.272	1.618	10.708	2	∞
-1.508	2.275	12.308	3	∞
-1.832	3.355	14.682	4	∞
-2.161	4.671	17.317	5	∞
-4.105	16.854	37.275	6	∞

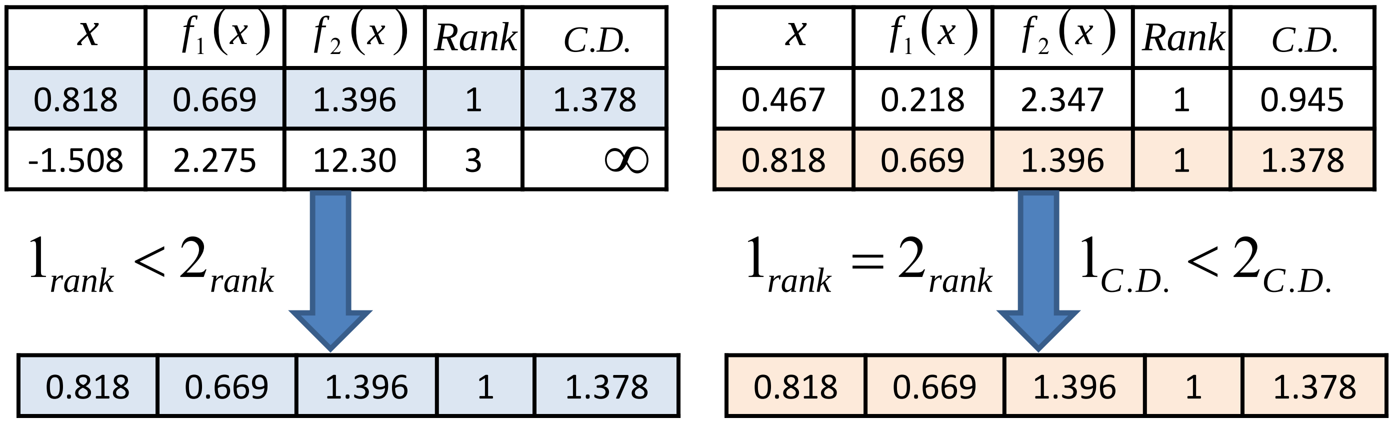




Tournament Selection

Runs a 'tournament' among a few individuals chosen at random from the population and selects the winner (the one with the best fitness) for crossover.

- In tournament selection, a number *Tour size* of individuals is chosen randomly from the population and the best individual from this group is selected as parent. **(Based on the crowding operator)**



x	$f_1(x)$	$f_2(x)$	$Rank$	$C.D.$
0.818	0.669	1.396	1	1.378
-1.508	2.275	12.30	3	∞

$$1_{rank} < 2_{rank}$$

0.818	0.669	1.396	1	1.378

x	$f_1(x)$	$f_2(x)$	$Rank$	$C.D.$
0.467	0.218	2.347	1	0.945
0.818	0.669	1.396	1	1.378

$$1_{rank} = 2_{rank} \quad 1_{C.D.} < 2_{C.D.}$$

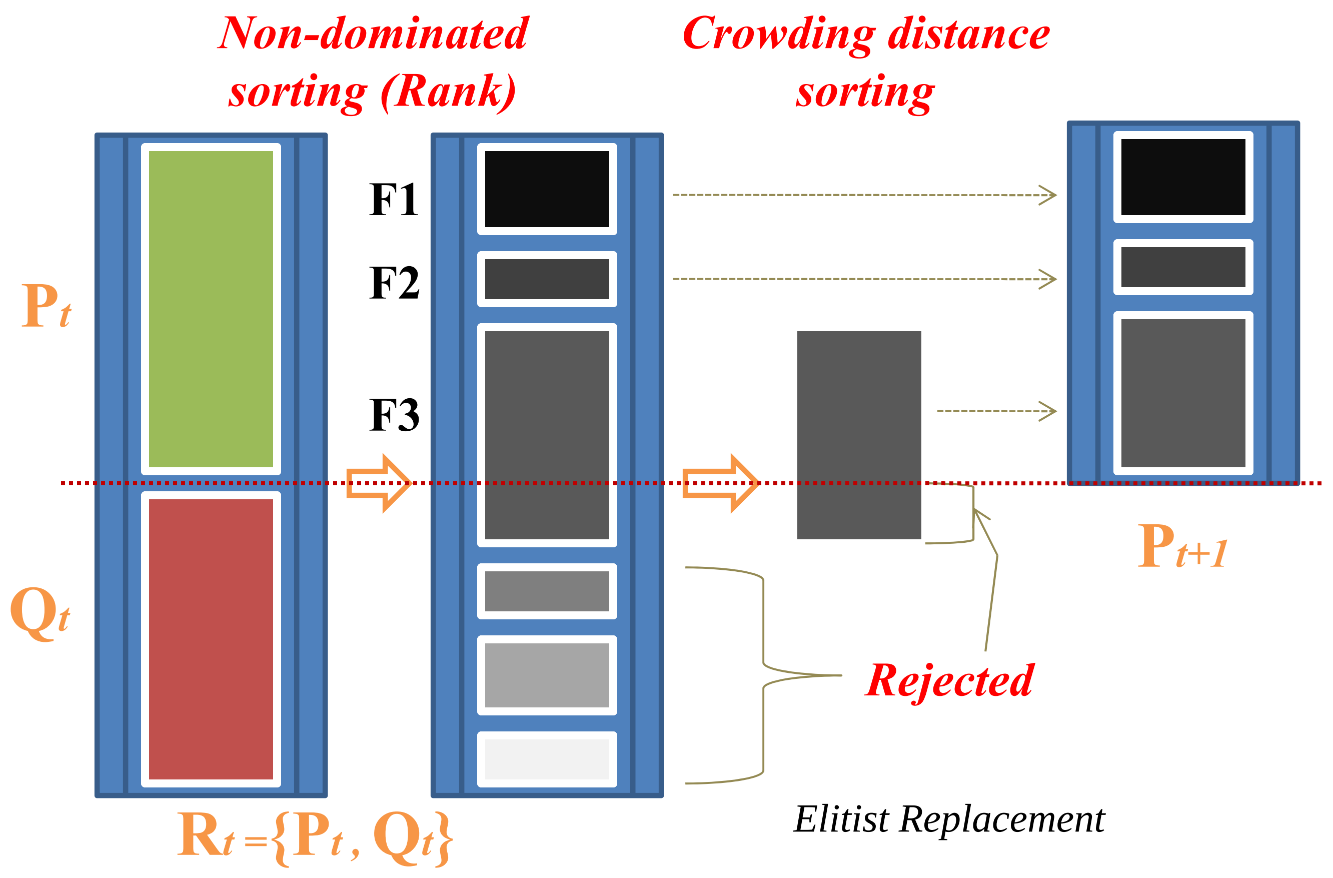
0.818	0.669	1.396	1	1.378

NTNU

Working Procedure

- Offspring population P_t is first created using the parent population Q_t.
- P_t and Q_t are combined together to form R_t of size $2N$, and then a non-dominated sorting is used to classify the entire population of R_t.
- The new population is filled by solutions of different non-dominated front, once at a time.
  - Starts with the best non-dominated front, and continues with the second non-dominated front, followed by third, and so on.
- Since the overall population size of R_t is $2N$, not all fronts may be accommodated in the new population of size N.
  - All fronts which could not be accommodated are simply deleted.
- For the last allowed front, there may exist more solutions than the remaining slots in the new population.

Selection for Next Generation



NSGA II (Advantages and Disadvantages)

Advantages:

✓ Crowding comparison procedure leads to diversity among non-dominated solutions.

✓ As solutions compete with their crowding distances, no extra niching parameter (such as α_{share} needed in MOGA, NSGA) is required here.

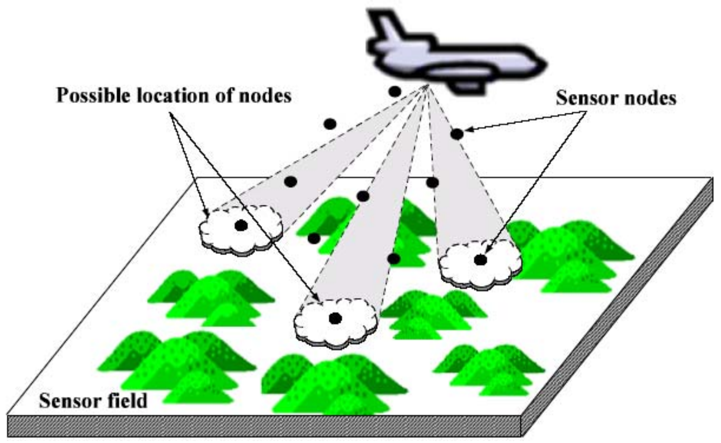
✓ It allows a global non-domination check among offspring and parent solutions.

✓ Its elitism property can speed up the performance significantly, which also can help preventing the loss pf good solutions once they are found.

Disadvantages:

× The non-dominated sorting needs to be performed on a population of size $2N$, instead of a population of size N required in most algorithms.

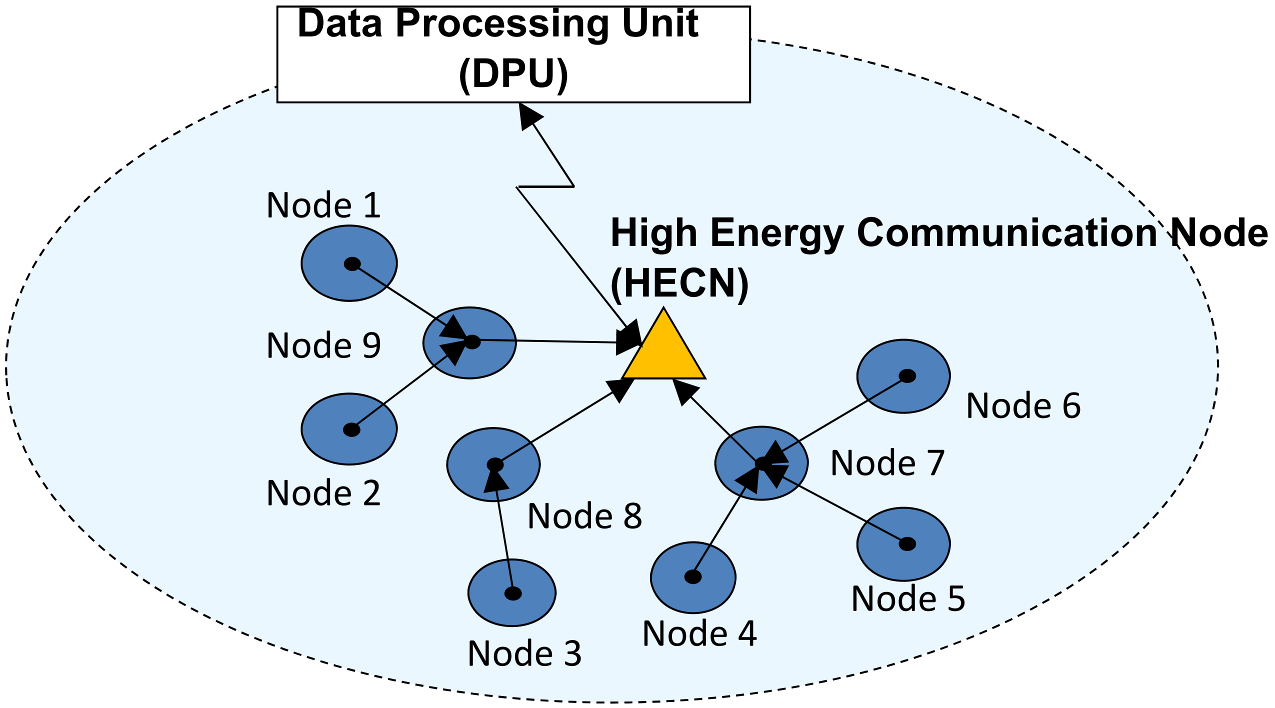
Layout Optimization for a Wireless Sensor Network using NSGA - II



a) Coverage
b) Lifetime

* Slides taken from Prof. Ganapati Panda

Wireless Sensor Network (WSN)


Data Processing Unit (DPU)
Node 1
High Energy Communication Node (HECN)
Node 9
Node 6
Node 2
Node 7
Node 8
Node 5
Node 4
Node 3

Example of a WSN where sensor nodes are communicating with the DPU through HECN

Optimization of Coverage

- Coverage is defined as the ratio of the union of areas covered by each node and the area of the entire ROI.

$$\boxed{C = \frac{\bigcup_{i=1,\ldots,N} A_i}{A}}$$

A_i	-	Area covered by the ith node
N	-	Total number of nodes
A	-	Area of the ROI

NTNU

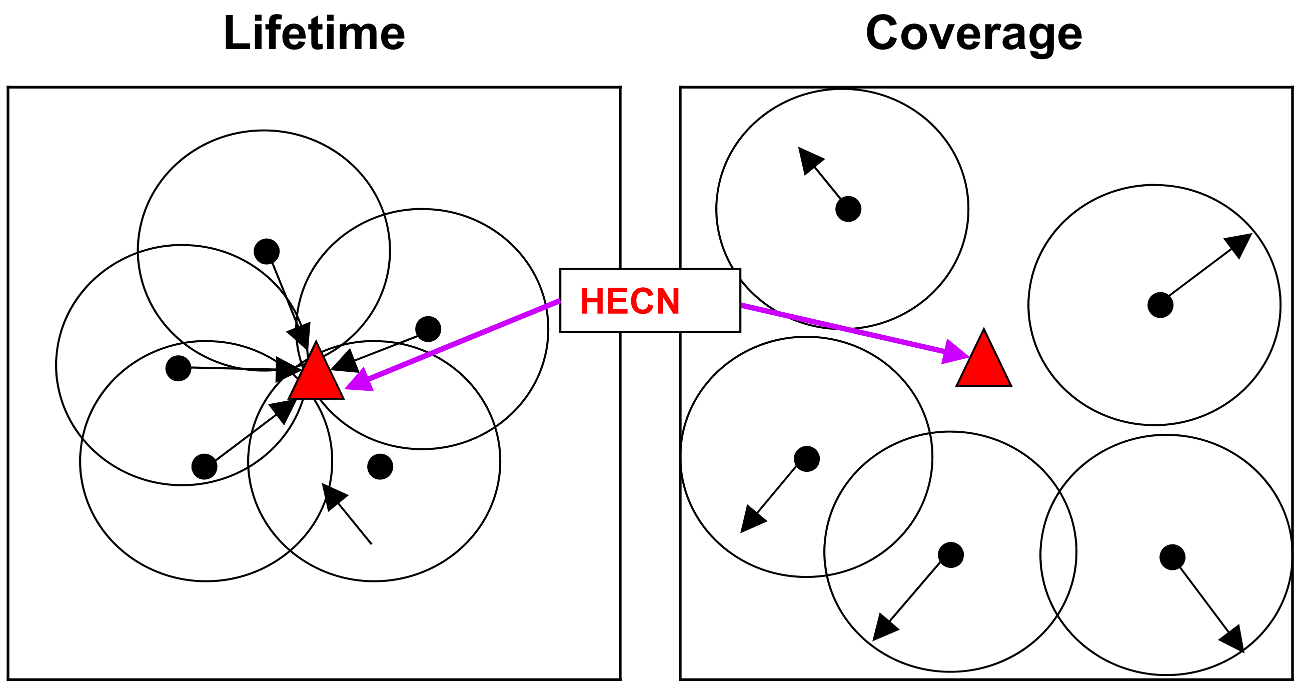
Optimization of Lifetime

- The lifetime of the whole network is the time until one of the participating nodes run out of energy.
- In every sensing cycle, the data from every node is routed to HECN through a route of minimum weight

$$\boxed{\text{Lifetime} = \frac{T_{\text{failure}}}{T_{\max}}}$$

T_{failure} = maximum number of sensing cycles before failure of any node
$T_{\max}$ = maximum number of possible sensing cycles

Competing Objectives



- try to arrange the nodes as close as possible to the HECN for maximizing lifetime

- try to spread out the nodes for maximizing coverage

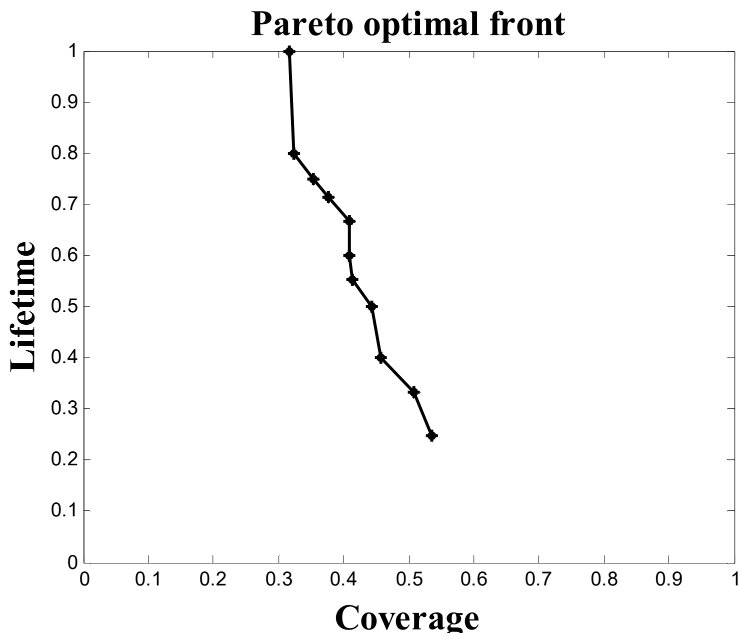
NTNU

Simulation Parameters

Parameters of NSGA-II

Number of chromosomes	100
Number of generations	50
Crossover Probability	0.9
Mutation Probability	0.5
Distribution index for crossover	20
Distribution index for mutation	20
Tour size	2

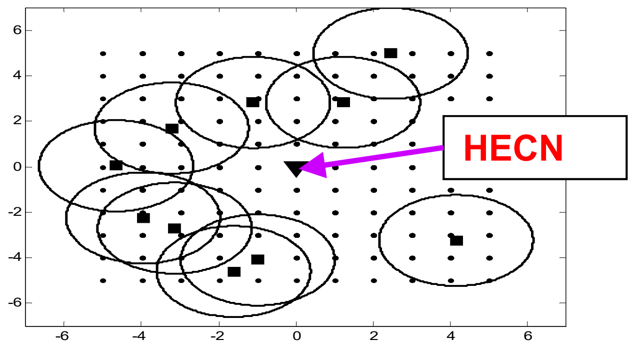
NSGA-II Results



- Pareto Front obtained for a WSN with 10 sensors, 100 chromosomes and 50 generations

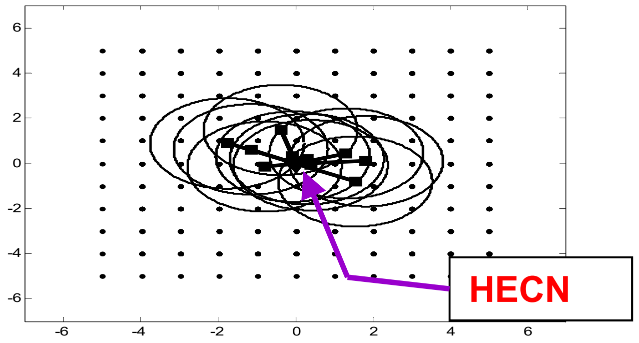
NSGA-II Results (Cont'd)

Initial Disconnect Network



Best Lifetime

Coverage = 0.3335 Lifetime = 0.999



Best Coverage

Coverage = 0.5353 Lifetime = 0.249