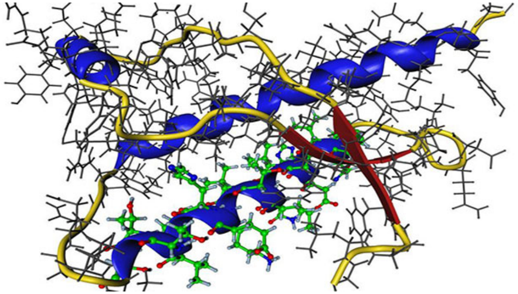


Lecture 5

Multi-objective Evolutionary Algorithms

Håken Jevne,
Kazi Ripon and Pauline Haddow

NTNU

Outline

- Introduction to MOO
  - Conceptual example
- Pareto-Optimality and Metrics
- Diversity Preservation
  - Example: NSGA II
- Other MOO Algorithms
- MOO Application example

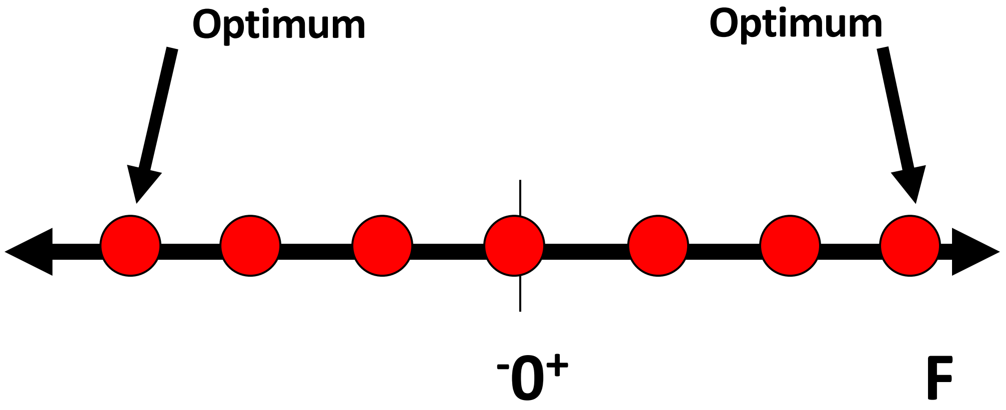
Outline

- Introduction to MOO
  - Conceptual example
- Pareto-Optimality and Metrics
- Diversity Preservation
  - Example: NSGA II
- Other MOO Algorithms
- MOO Application example

Single Objective Optimization

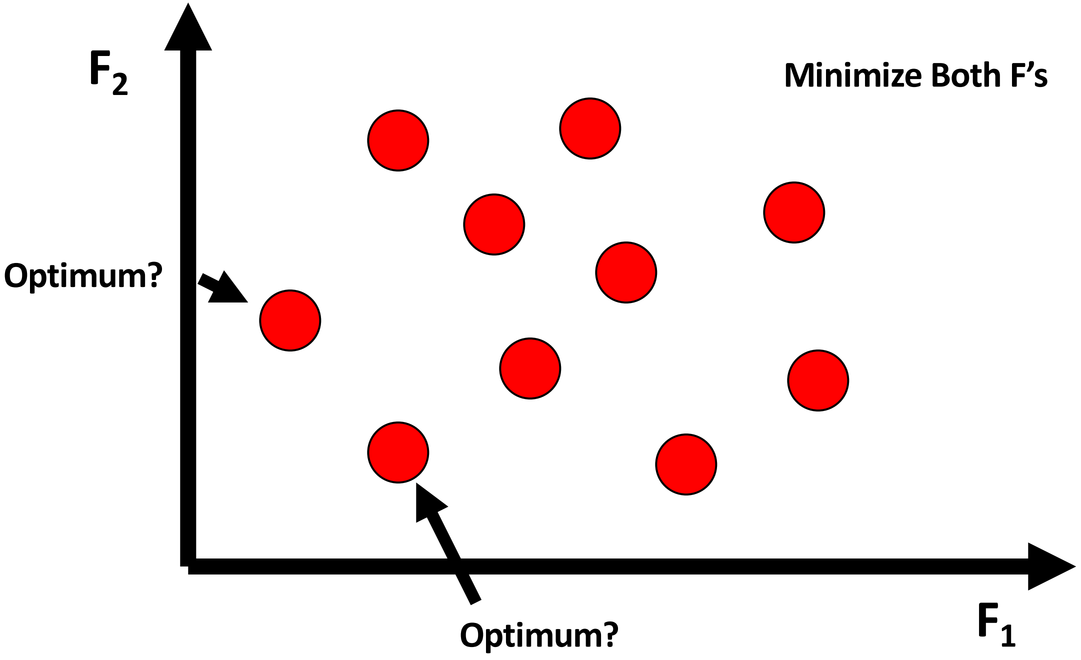
The problem has a 1 dimensional performance space and the optimum point is the one that is the furthest toward the desired extreme.

There is typically only a single solution that gives the best objective value.



Best in all dimensions?

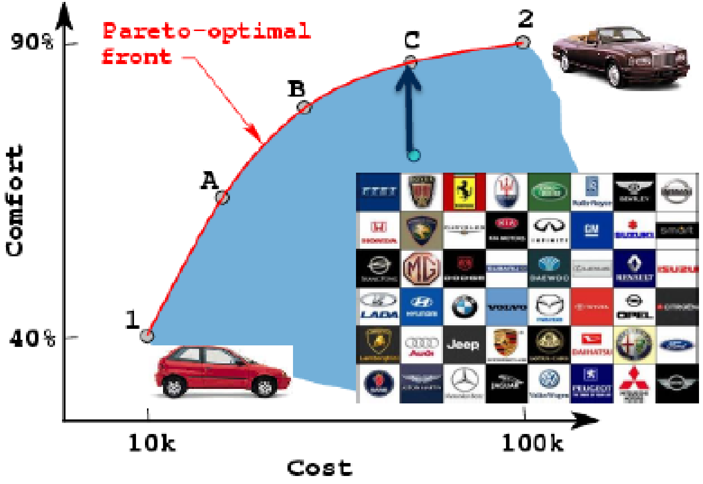
But what happens in a case like this (**conflicting**):



Why Multi-Objective Optimization?

Buying an Automobile

- Objective = reduce cost, while maximize comfort.
- Which solution (1, A, B, C, 2) is best ???
- No solution from this set makes both objectives look better than any other solution from the set.
- No single optimal solution.
- Trade off between conflicting objectives- cost and comfort.



MO optimization means:

"Take from Peter to pay Paul"

NTNU

Formal Definition

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

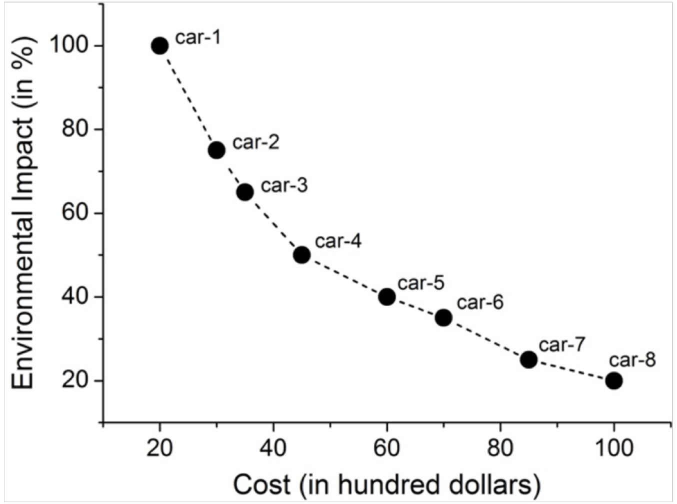
General form of multi-objective optimization:

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

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

Multi-Objective Optimization

- A MOOP will have many alternative solutions in the feasible region.
- This is because a solution that is optimal with respect to one objective might be a poor candidate for another objective.
- 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**.



Outline

- Introduction to MOO
  - Conceptual example
- Evolutionary MOO concepts
- Pareto-Optimality and Metrics
- Diversity Preservation
  - Example: NSGA II
- Other MOO Algorithms
- MOO Application example

Evolutionary Approach: General Concept

Multi-objective optimization is the process of simultaneously optimizing **several incommensurable and often competing 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.

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.

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

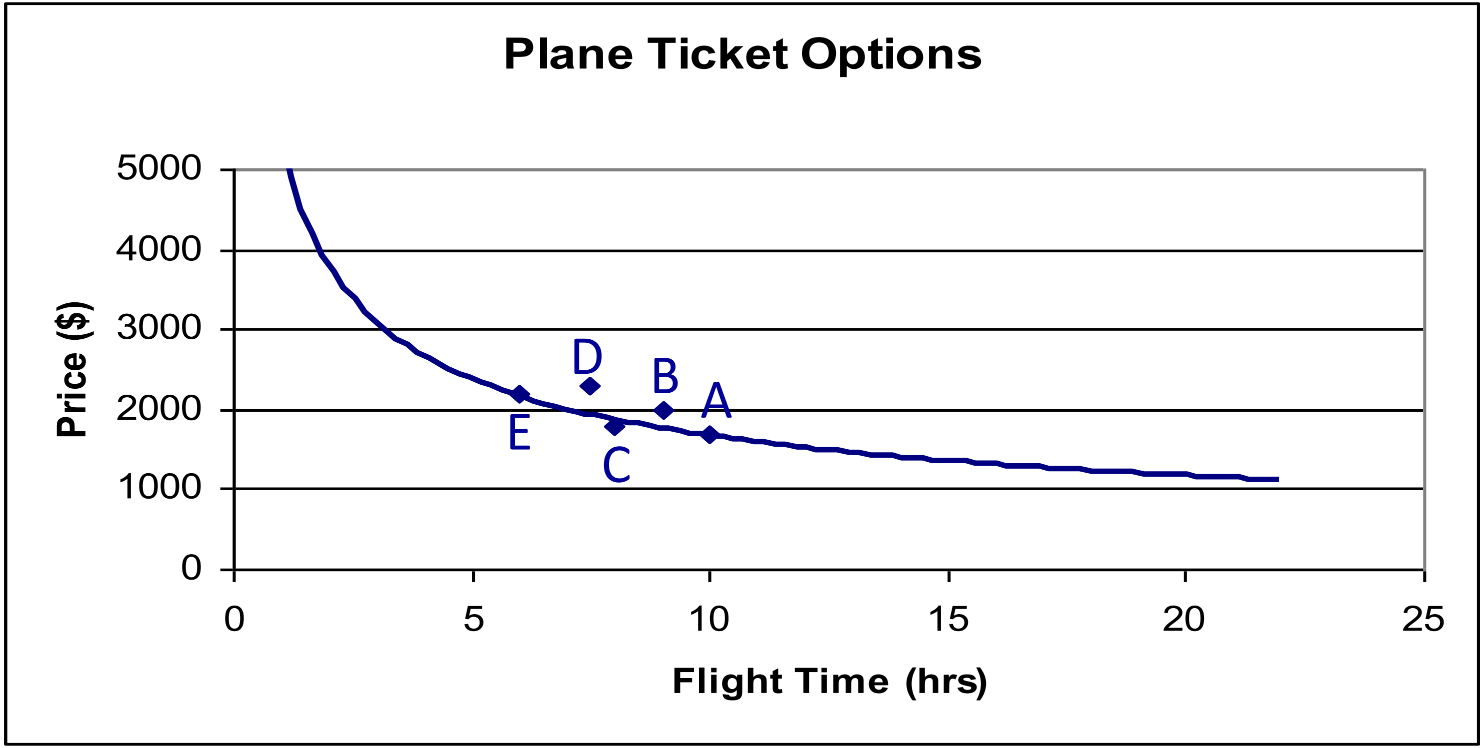
A, C, E

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**.
- 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.

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

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

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

where $\lambda_1, \lambda_2, \ldots\ldots, \lambda_m$ are weights values $\lambda_1 + \lambda_2 + \ldots\ldots + \lambda_m = 1$
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.
- Sometimes the differences are qualitative and the relative importance of these objectives can’t be numerically quantified.

Evolutionary vs Traditional Approaches

Evolutionary Approaches	Traditional Approaches
Can simultaneously deal with a set of possible solutions (the so-called population).	Normally work with a single solution.
Allow us to find several members of the Pareto-Optimal set in a single run of the algorithm.	Need to perform a series of separate runs to find a set of alternative solutions.
less susceptible to the shape or continuity of the Pareto front -- can easily deal with discontinuous or concave Pareto fronts.	These two issues are a real concern for mathematical programming techniques
Can be implemented in a parallel environment.	Difficult for parallel implementation.

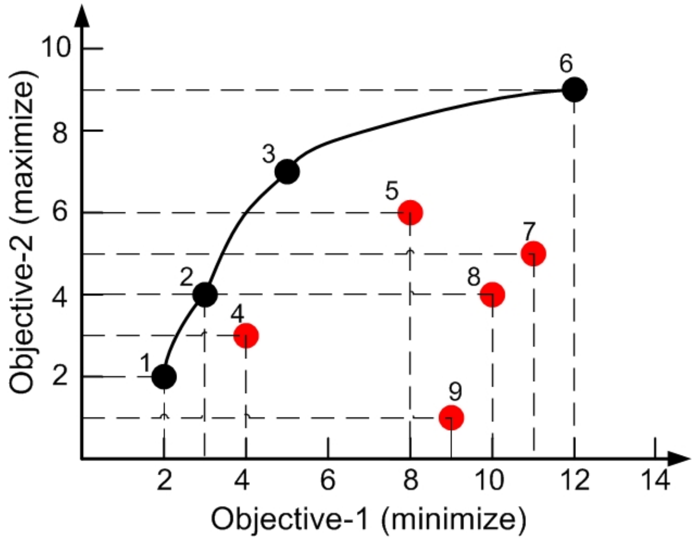
NTNU

Outline

- Introduction to MOO
  - Conceptual example
- Pareto-Optimality and Metrics
- Diversity Preservation
  - Example: NSGA II
- Other MOO Algorithms
- MOO Application example

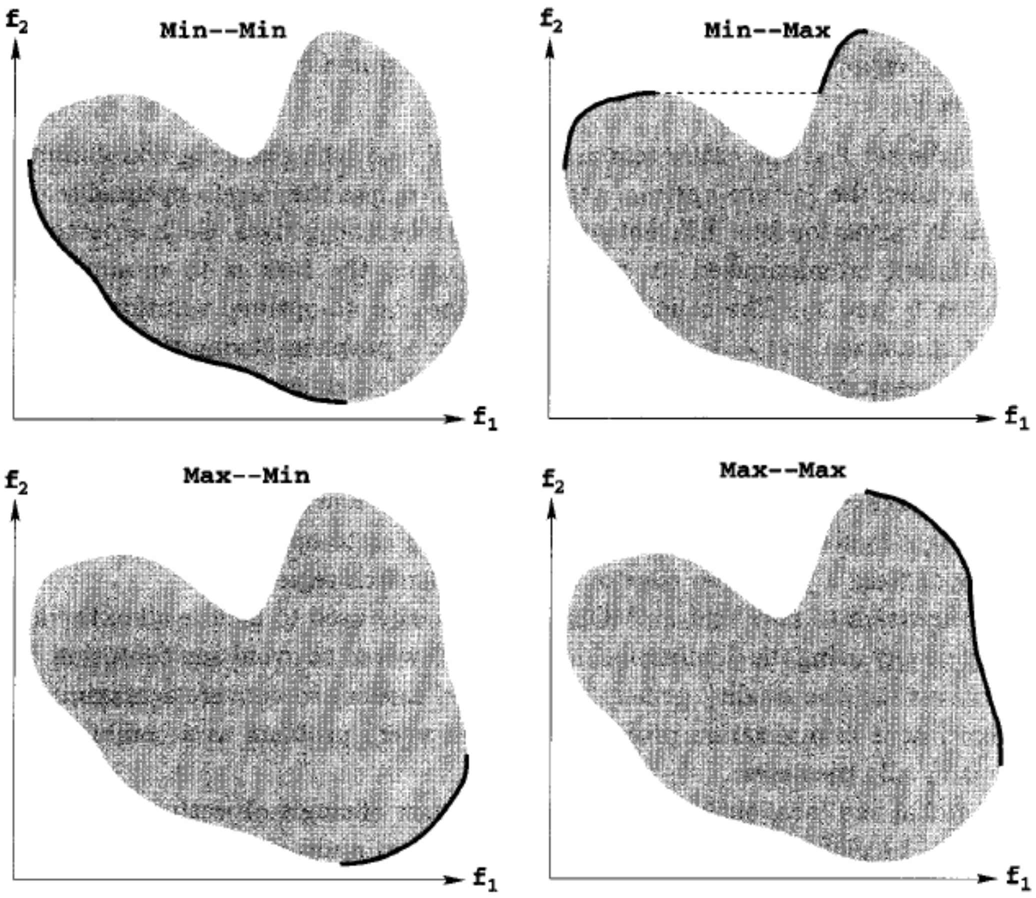
Pareto-Optimality

- Pareto optimization can handle the problems associated with weighted-sum approach 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.

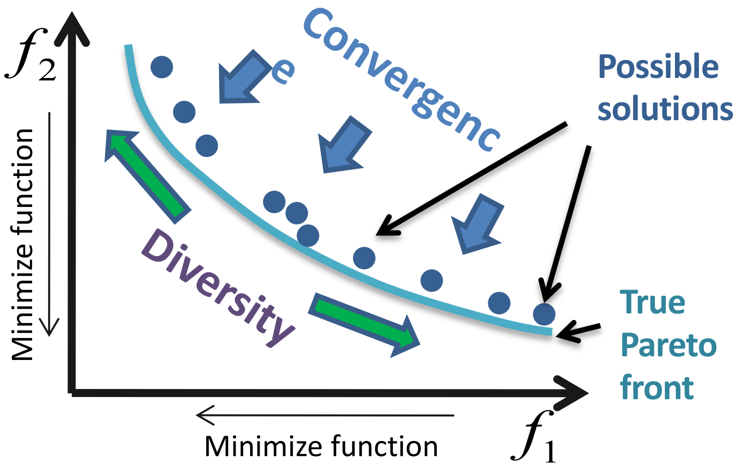


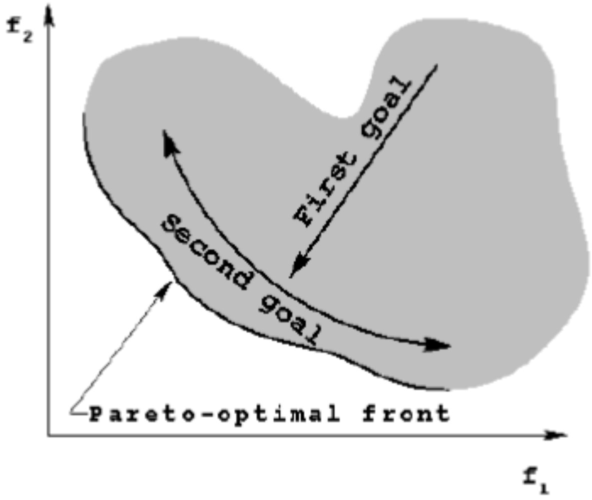
- The line is called the **Pareto front** and solutions on it are called **Pareto-Optimal**.

Shape of Pareto-Front



Objectives in MOOP

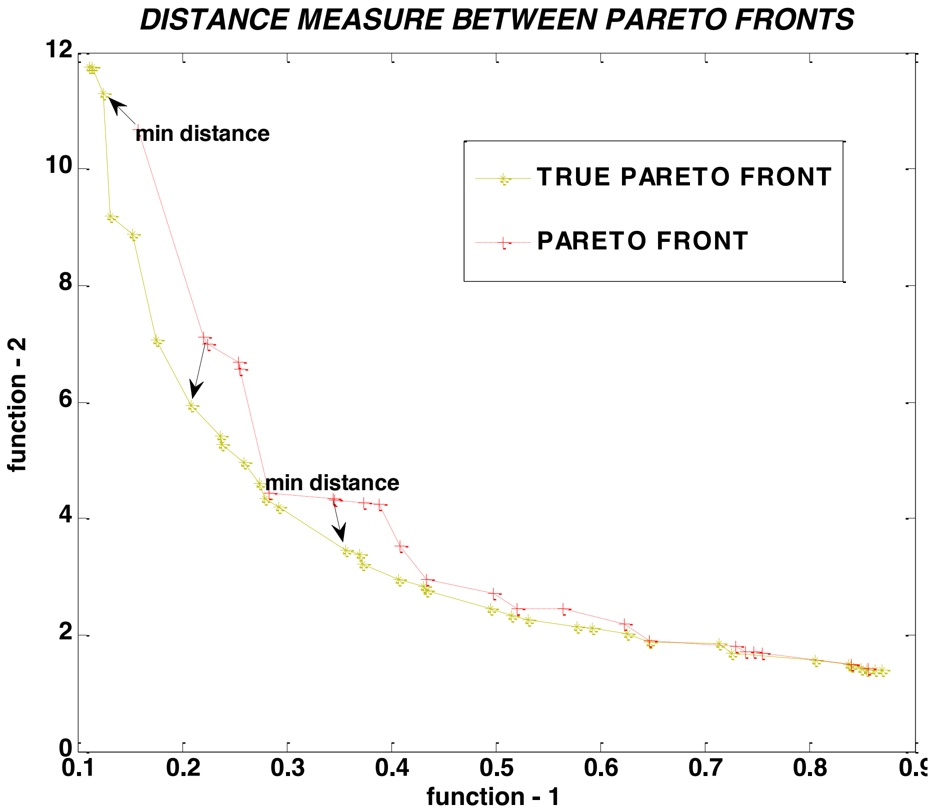




- To find a set as close as possible to the Pareto-optimal front (***Convergence***).
- To find a set of solutions as diverse as possible (***Diversity***).
    - Representation of the entire Pareto-Optimal front.

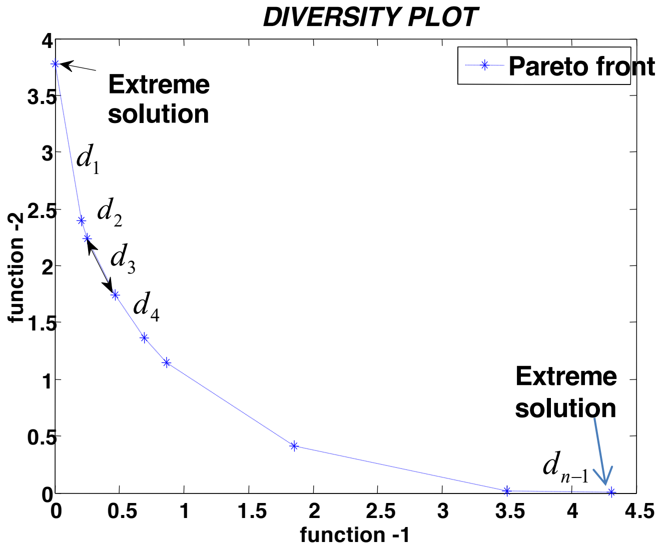
Convergence Metrics: One Example

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



Diversity Metrics: One Example

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



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.

d_i can be any distance measure between neighboring solutions, and $\bar{d}$ is the mean value of these distance measures.

Outline

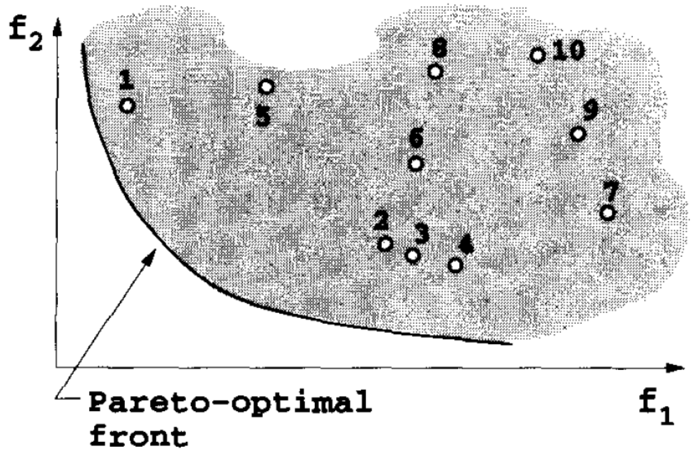
- Introduction to MOO
  - Conceptual example
- Pareto-Optimality and Metrics
- Diversity Preservation
  - Example: NSGA II
- Other MOO Algorithms
- MOO Application example

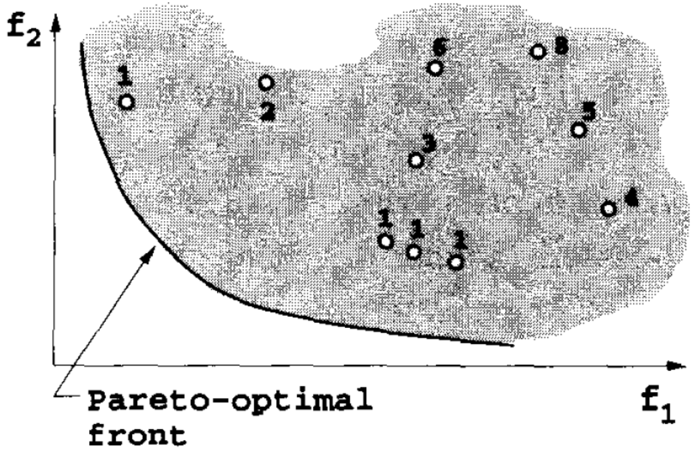
Diversity Control based on Non-Domination Concept

- 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 (**1 + 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.

Niching: ranking (MOGA)

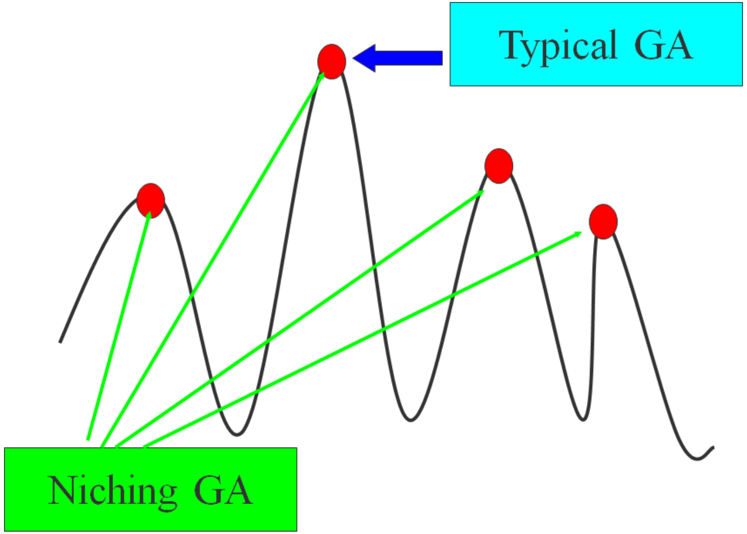
- Upper figure shows a two-objective minimization problem having 10 solutions.
- Lower figure shows the rank of each solution.
- The ranking procedure may not assign all possible ranks (between 1 and N).
- Ranks 7, 9 and 10 are missing.





Niching

- The idea of segmenting the population of the GA into disjoint sets to have at least one member in each region of the fitness function that is "*interesting*“ – local/global optima;
  - General intention is to cover more than one local optima.
- Niching is used to maintain the diversity of the population.



Niching : Sharing Function Model

- To maintain diversity among non-dominated solutions, ***niching among solutions of each rank***.
- Focusing on **degrading the fitness** of similar solutions.
- **Niche count** provides an estimate of the extent of crowding near a solution.
- The normalized distance between any two solutions *i* and *j* ***in a rank*** is calculated as follows:

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

- Where f_k^{max} and f_k^{min} are the maximum and minimum objective function value of the *k*-th objective.

NTNU

Niching : Sharing Function Model

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

- For the solution i, d_{ij} is computed for each solution j (including i) having the same rank.
- With $\alpha = 1$, the sharing function value is computed as:

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

- Thereafter, the *niche count* is calculated by summing the sharing function values:

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

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

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

Niching : Sharing Function Model

- *Sharing function* is used to obtain an estimate of the no. of solutions belonging to each optimum.
- The parameter d is the distance between any two solutions in the population.
- The above function takes a value in $[0,1]$, depending on the values of d and σ_{share}.
- If d is zero (two solutions are identical or their distance is zero), $Sh(d) = 1$.
  - A solution has full sharing effect on itself.
- If $d \geq \sigma_{share}$ (two solutions are at least a distance of σ_{share} away from each other, $Sh(d) = 0$.
  - Two solutions do not have any sharing effect on each other.
- Any other distance d between two solutions will have a partial effect on each.

Example * (*fitness sharing, not MO*)

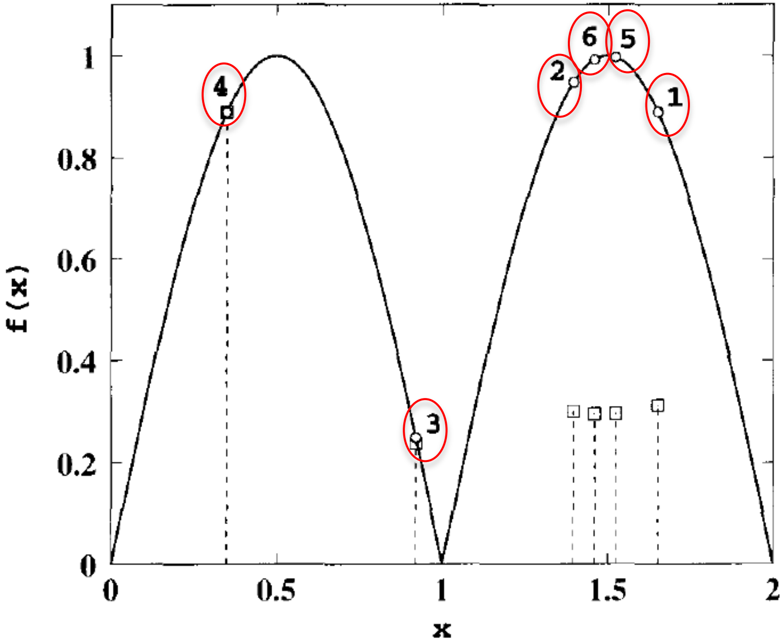
$$\text{Maximize} \quad \left. \begin{array}{l} f(x) = |\sin(\pi x)|, \\ 0 \le x \le 2. \end{array} \right\}$$

Sol. i	String	Decoded value	$x^{(i)}$	f_i
1	110100	52	1.651	0.890
2	101100	44	1.397	0.948
3	011101	29	0.921	0.246
4	001011	11	0.349	0.890
5	110000	48	1.524	0.997
6	101110	46	1.460	0.992

Assume:

$\sigma_{share} = 0.5$

$\alpha = 1$



* Example taken from Deb, 2001.

NTNU

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

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

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:

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

Example

Similarly, the niche count calculations of all six solutions:

	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

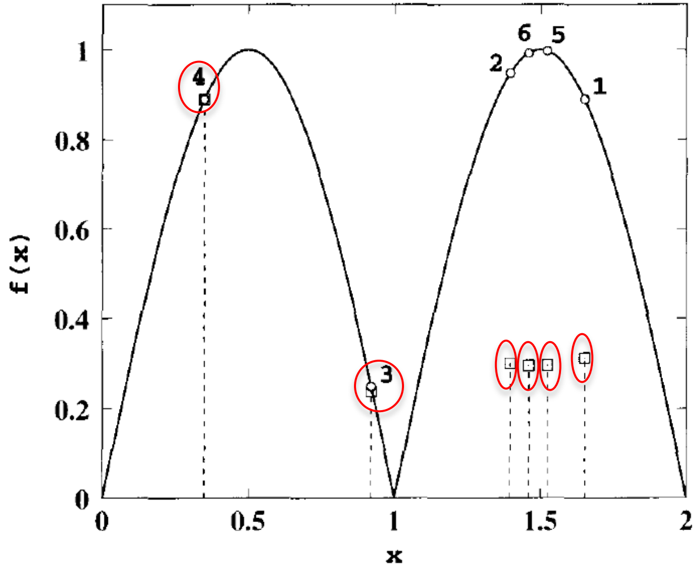
Example

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

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

Similarly, six solutions and corresponding shared fitness values:

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



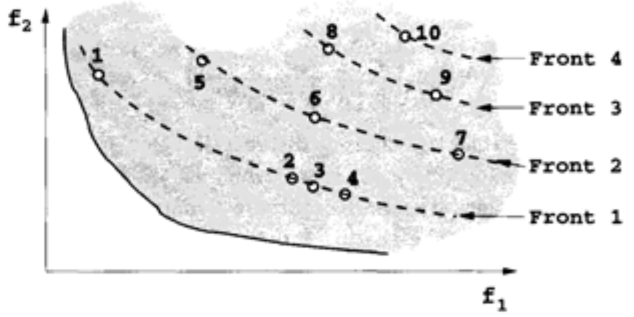
Crowding Distance Assignment

- The crowding operator ($\leq_n$) guides the selection process at the various stages of the algorithm toward a uniformly spread-out Pareto optimal front.
- IN NSGA-II, it is used to choose the **members of the last front (rank),** which reside in the least crowded region in that front using a niching strategy.
- 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:
  - If solution i has a better rank: $r_i < r_j$.
  - If they have the same rank but solution i has a *better crowding distance* (lower dense) than solution j:
    - that is, $r_i = r_j$ and $d_i > d_j$.

.



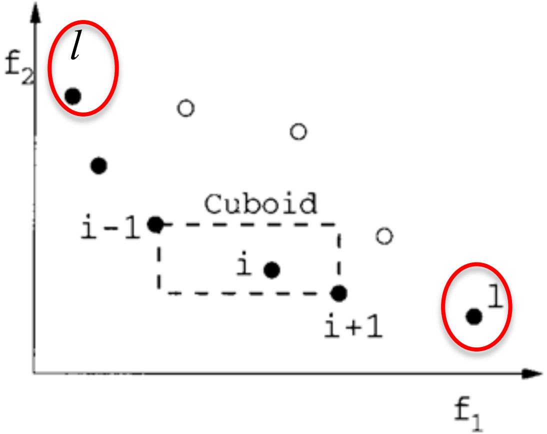
Crowding Distance Assignment Procedure: Crowding-sort($\mathcal{F}$, $<_c$)

Step C1 Call the number of solutions in $\mathcal{F}$ as $l = |\mathcal{F}|$. 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 $d_{I_1^m} = d_{I_l^m} = \infty$, and for all other solutions $j = 2$ to $(l - 1)$, assign:

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



I_1 , I_l = the lowest and highest objective function values (different for each objective, based on sorting), respectively.

Crowding Distance : Example

Minimize

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

where $-5 \le x \le 5$

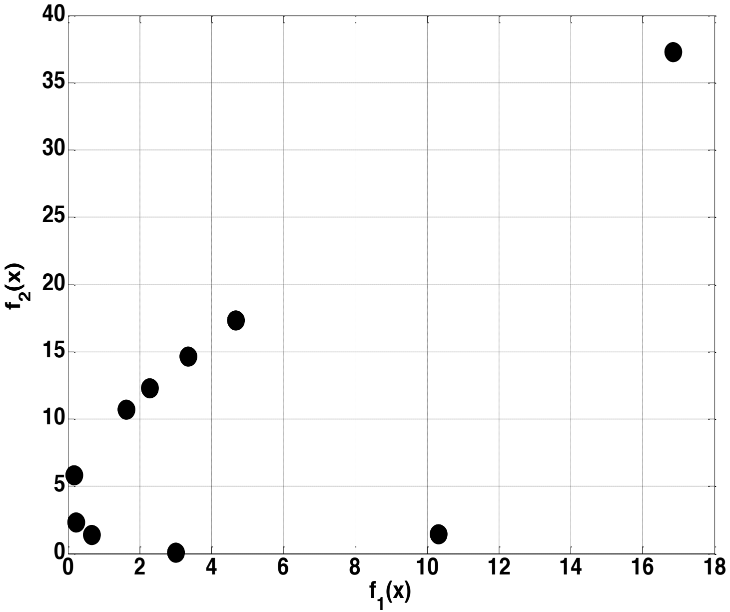
- Search space is of single dimension (given).
- Objective space is of two dimension (given).
- Let population size = ***10***
- Initialize population with 10 chromosomes having single dimensioned real value.
- These values are randomly distributed in between ***[-5, 5]***.

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

Crowding Distance : Example

- Find out all objective functions values for all chromosomes.

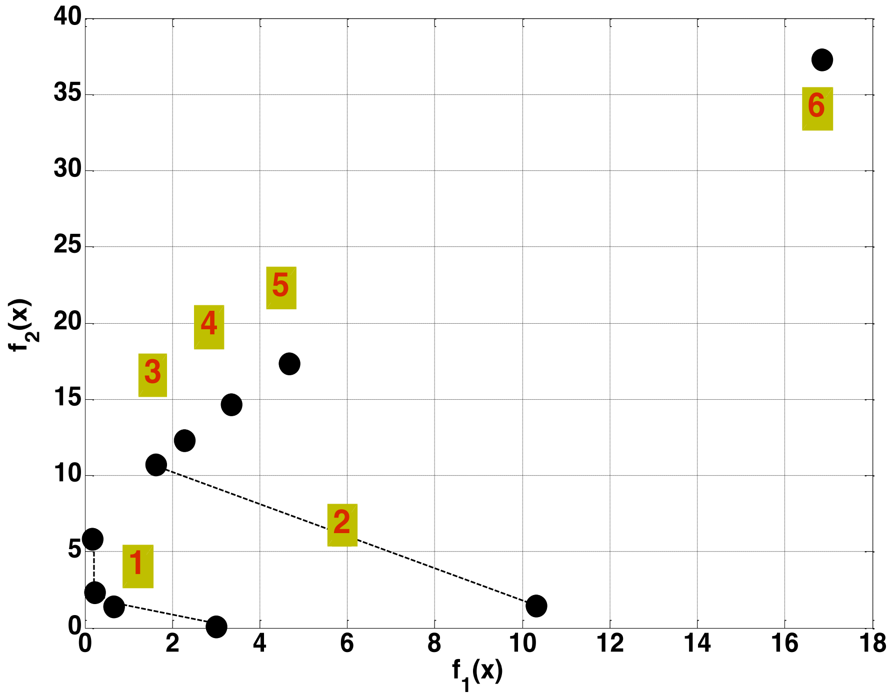
x	$f_1(x)$	$f_2(x)$
-0.414	0.171	5.829
0.467	0.218	2.347
0.818	0.669	1.396
1.735	3.011	0.07
3.210	10.308	1.465
-1.272	1.618	10.708
-1.508	2.275	12.308
-1.832	3.355	14.682
-2.161	4.671	17.317
-4.105	16.854	37.275



Crowding Distance : Example

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

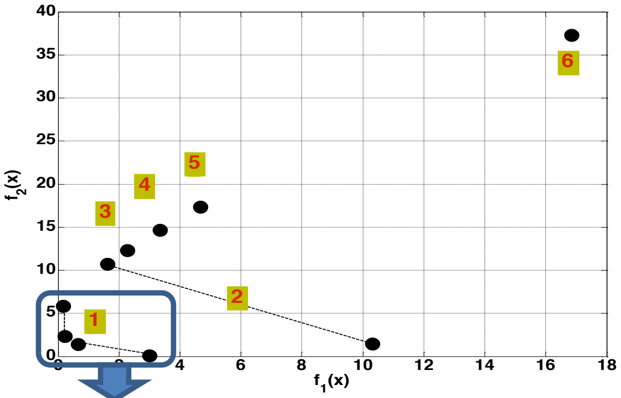
x	$f_1(x)$	$f_2(x)$	$Rank$
-0.414	0.171	5.829	1
0.467	0.218	2.347	1
0.818	0.669	1.396	1
1.735	3.011	0.07	1
3.210	10.308	1.465	2
-1.272	1.618	10.708	2
-1.508	2.275	12.308	3
-1.832	3.355	14.682	4
-2.161	4.671	17.317	5
-4.105	16.854	37.275	6

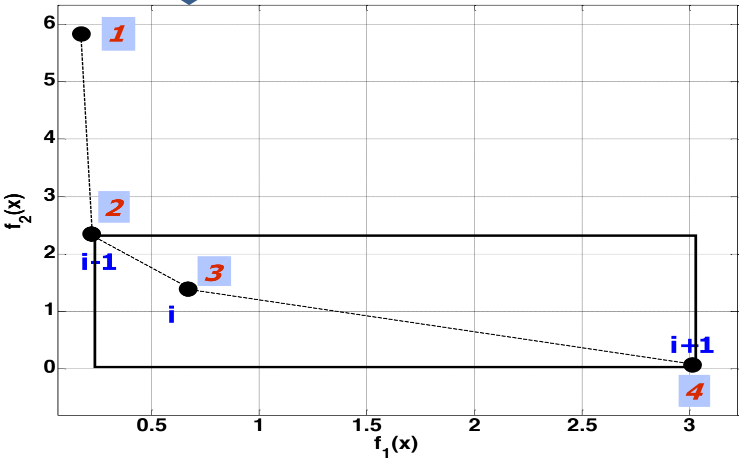


Crowding Distance Assignment

- 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

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

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

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

0.818	0.669	1.396	1	1.378

Outline

- Introduction to MOO
  - Conceptual example
- Pareto-Optimality and Metrics
- Diversity Preservation
  - Example: NSGA II
- Other MOO Algorithms
- MOO Application example

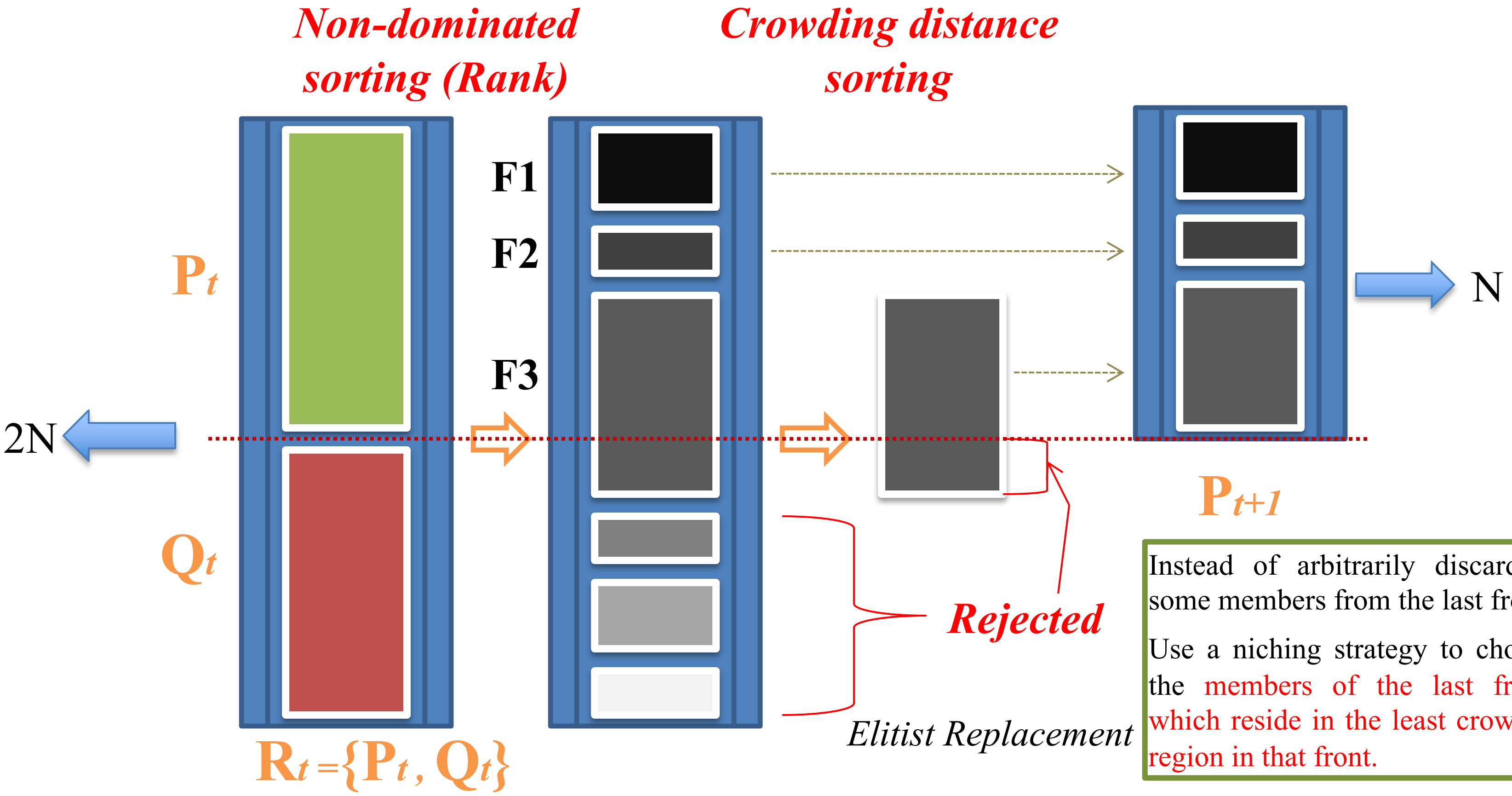
NSGA- II

- Developed by Prof. K. Deb and his students at Kanpur Genetic Algorithms Laboratory (2002). Citation: 23228
- 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:*

NSGA- II: *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.
- It allows a global non-domination check among the offspring and parent solutions.

NSGA- II: *Selection for Next Generation*



Instead of arbitrarily discarding some members from the last front,

Use a niching strategy to choose the members of the last front, which reside in the least crowded region in that front.

NSGA II: *Example* *

Min-Ex:

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

- Parent and offspring populations used in this example:

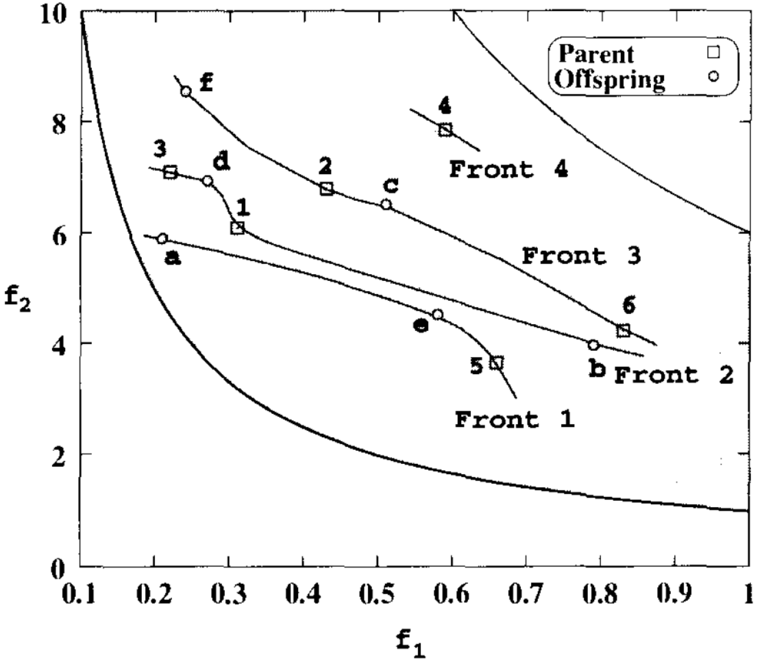
Parent population, P_t					Offspring population, Q_t				
Solution	x_1	x_2	f_1	f_2	Solution	x_1	x_2	f_1	f_2
1	0.31	0.89	0.31	6.10	a	0.21	0.24	0.21	5.90
2	0.43	1.92	0.43	6.79	b	0.79	2.14	0.79	3.97
3	0.22	0.56	0.22	7.09	c	0.51	2.32	0.51	6.51
4	0.59	3.63	0.59	7.85	d	0.27	0.87	0.27	6.93
5	0.66	1.41	0.66	3.65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

* Example taken from K. Deb, 2001.

NSGA II: *Example*

Step 1 We first combine the populations P_t and Q_t and form $R_t = \{1, 2, 3, 4, 5, 6, a, b, c, d, e, f\}$. Next, we perform a non-dominated sorting on R_t. We obtain the following non-dominated fronts:

$$\mathcal{F}_1 = \{5, a, e\},$$
$$\mathcal{F}_2 = \{1, 3, b, d\},$$
$$\mathcal{F}_3 = \{2, 6, c, f\},$$
$$\mathcal{F}_4 = \{4\}.$$



NSGA II: *Example*

Step 2 We set $P_{t+1} = \emptyset$ and $i = 1$. Next, we observe that $|P_{t+1}| + |\mathcal{F}_1| = 0 + 3 = 3$. Since this is less than the population size N $(= 6)$, we include this front in P_{t+1}. We set $P_{t+1} = \{5, a, e\}$. With these three solutions, we now need three more solutions to fill up the new parent population.

Now, with the inclusion of the second front, the size of $|P_{t+1}| + |\mathcal{F}_2|$ is $(3 + 4)$ or 7. Since this is greater than 6, we stop including any more fronts into the population.

NTNU

NSGA II: *Example*

Step 3:

- Next, we consider solutions of the second front only and observe that **3 of the 4 solutions must be chosen to fill up 3 remaining slots** in the new population.

- This requires that we first sort this sub-population (solutions 1, 3, a, and d) by **using crowding distance operator**.

NSGA II: *Crowding Distance Assignment*

Step C1 Call the number of solutions in $\mathcal{F}$ as $l = |\mathcal{F}|$. 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 $d_{I_1^m} = d_{I_l^m} = \infty$, and for all other solutions $j = 2$ to $(l - 1)$, assign:

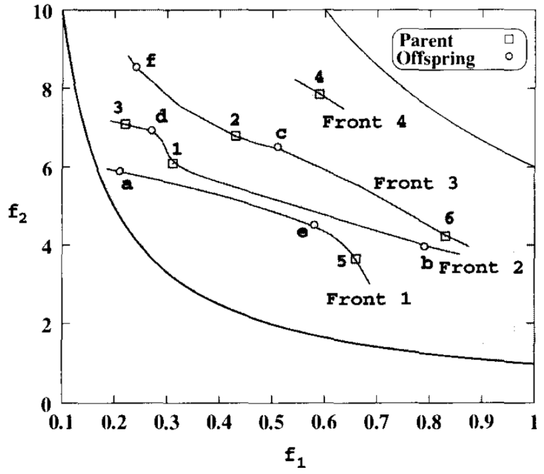
$$d_{I_j^m} = d_{I_j^m} + \frac{f_m^{(I_{j+1}^m)} - f_m^{(I_{j-1}^m)}}{f_m^{\max} - f_m^{\min}}.$$

NSGA II: *Crowding Distance Assignment*

Step C1 We notice that $l = 4$ and set $d_1 = d_3 = d_b = d_d = 0$. We also set $f_1^{max} = 1$, $f_1^{min} = 0.1$, $f_2^{max} = 60$ and $f_2^{min} = 0$.

Step C2 For the first objective function, the sorting of these solutions is shown in Table and is as follows: $I^1 = \{3, d, 1, b\}$.

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

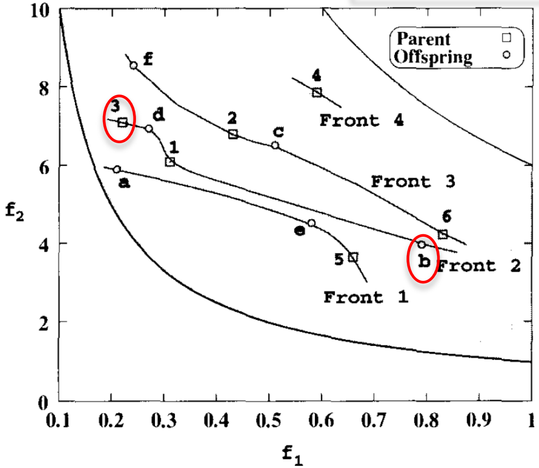


NSGA II: *Crowding Distance Assignment*

Step C3 Since solutions 3 and b are boundary solutions for f_1, we set $d_3 = d_b = \infty$. For the other two solutions, we obtain:

$$d_d = 0 + \frac{f_1^{(1)} - f_1^{(3)}}{f_1^{max} - f_1^{min}} = 0 + \frac{0.31 - 0.22}{1 - 0.1} = 0.10.$$

$$d_1 = 0 + \frac{f_1^{(b)} - f_1^{(d)}}{f_1^{max} - f_1^{min}} = 0 + \frac{0.79 - 0.27}{1 - 0.1} = 0.58.$$



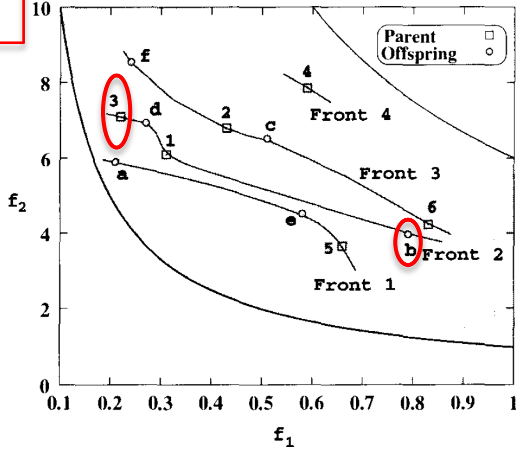
NSGA II: *Crowding Distance Assignment*

Now, we turn to the second objective function and update the above distances. First, the sorting on this objective yields $I^2 = \{b, 1, d, 3\}$. Thus, $d_b = d_3 = \infty$ and the other two distances are as follows:

$$d_1 = d_1 + \frac{f_2^{(d)} - f_2^{(b)}}{f_2^{max} - f_2^{min}} = 0.58 + \frac{6.93 - 3.97}{60 - 0} = 0.63.$$

$$d_d = d_d + \frac{f_1^{(3)} - f_2^{(1)}}{f_2^{max} - f_2^{min}} = 0.10 + \frac{7.09 - 6.10}{60 - 0} = 0.12.$$

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

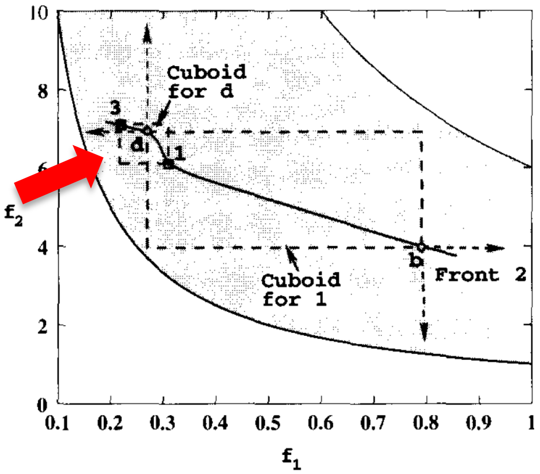


NSGA II: *Crowding Distance Assignment*

The overall crowded distances of the four solutions are:

$$d_1 = 0.63, \quad d_3 = \infty, \quad d_b = \infty, \quad d_d = 0.12.$$

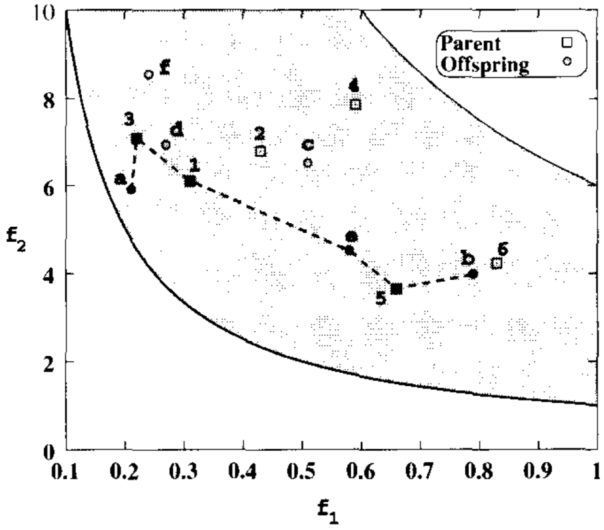
The cuboids (rectangles here) for these solutions are schematically shown in Figure . Solution d has the smallest perimeter of the hypercube around it than any other solution in the set $\mathcal{F}_2$, as evident from the figure. Now, we move to the main algorithm.



NSGA II: Example

Step 3 A sorting according to the descending order of these crowding distance values yields the sorted set {3, b, 1, d}. We choose the first three solutions.

Step 4 The new population is $P_{t+1} = \{5, a, e, 3, b, 1\}$. These population members are shown in Figure by joining them with dashed lines.



NSGA II: Example

- The offspring population Q_{t+1} has to be created next by using this parent population.

- The exact offspring population will depend on:
  - The chosen pair of solutions participating in a tournament – **Crowding Tournament Selection Operator**.
  - The chosen crossover.
  - The Chosen mutation operators.

Outline

- Introduction to MOO
  - Conceptual example
- Pareto-Optimality and Metrics
- Diversity Preservation
  - Example: NSGA II
- Other MOO Algorithms
- MOO Application example

NTNU

Other MOO Algorithms

- Multiple Objective Genetic Algorithm (MOGA)
- Strength Pareto Evolutionary Algorithm (SPEA)
- Niched Pareto Genetic Algorithm (NPGA)
- Pareto-Archived Evolution Strategy (PAES)
- Multi-Objective Messy Genetic Algorithm

Comparison of MOO Algorithms

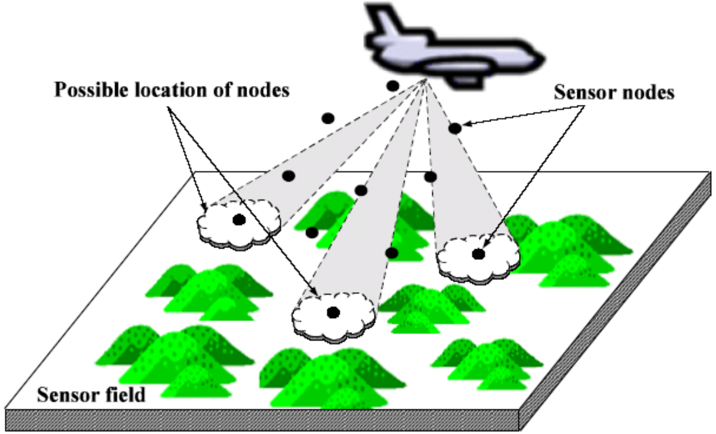
	MOGA	NSGA	NPGA	NSGAII	SPEA	PAES
Fitness	Very simple	Excellent due to assignment of fitness according to non dominated sets.	No explicit fitness assignment is needed like most other MOEAs.	Due to global non-domination check among offspring and parent solutions, elitism is implemented perfectly.	Easy to calculate.	A more direct approach is used in calculating the density while comparing the fitness.
Effect of parameters	Sharing function parameter σ_{share} needs fixing	Very sensitive to σ_{share}	Requires fixing two important parameters: α_{share} and t_{dom}	Solutions compete with their crowding distances. So, no extra niching parameter is required.	Parameter-less, thereby making it attractive to use.	In addition to archive size (N), depth parameter (d) is also important.
Convergence	Slow	Front-wise selection helps better convergence speed.	The choice of σ_{share} has more effect.	Elitism helps speedy convergence.	Not so speedy (non-dominated sorting of the whole population is not used for assigning fitness)	Depends on the choice of parameter values.
When to use	If a spread of Pareto-optimal solutions is required on the objective space.	It ensures that diversity is maintained among the non-dominated solutions.	Computationally efficient in solving problems with many objectives.	When a global non-domination check among offspring and parent solutions is necessary.	Ensures a better spread is achieved among the obtained non-dominated solutions.	Has a direct control on the diversity that can be achieved by using the appropriate size of the depth size d.

NTNU

Outline

- Introduction to MOO
  - Conceptual example
- Pareto-Optimality and Metrics
- Diversity Preservation
  - Example: NSGA II
- Other MOO Algorithms
- MOO Application example

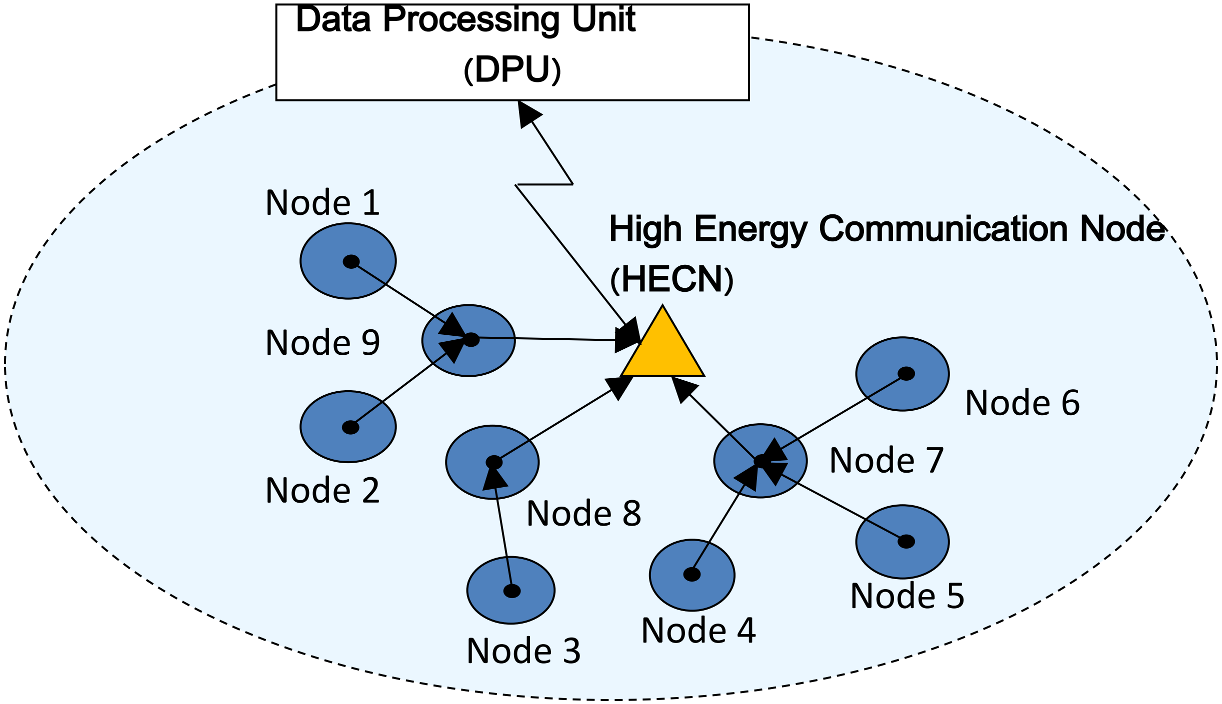
Layout Optimization for a Wireless Sensor Network using NSGA - II



a) Coverage
b) Lifetime

* Slides taken from Prof. Ganapati Panda

Wireless Sensor Network (WSN)



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.

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

A_i - Area covered by the i^{th} node
N - Total number of nodes
A - Area of the ROI

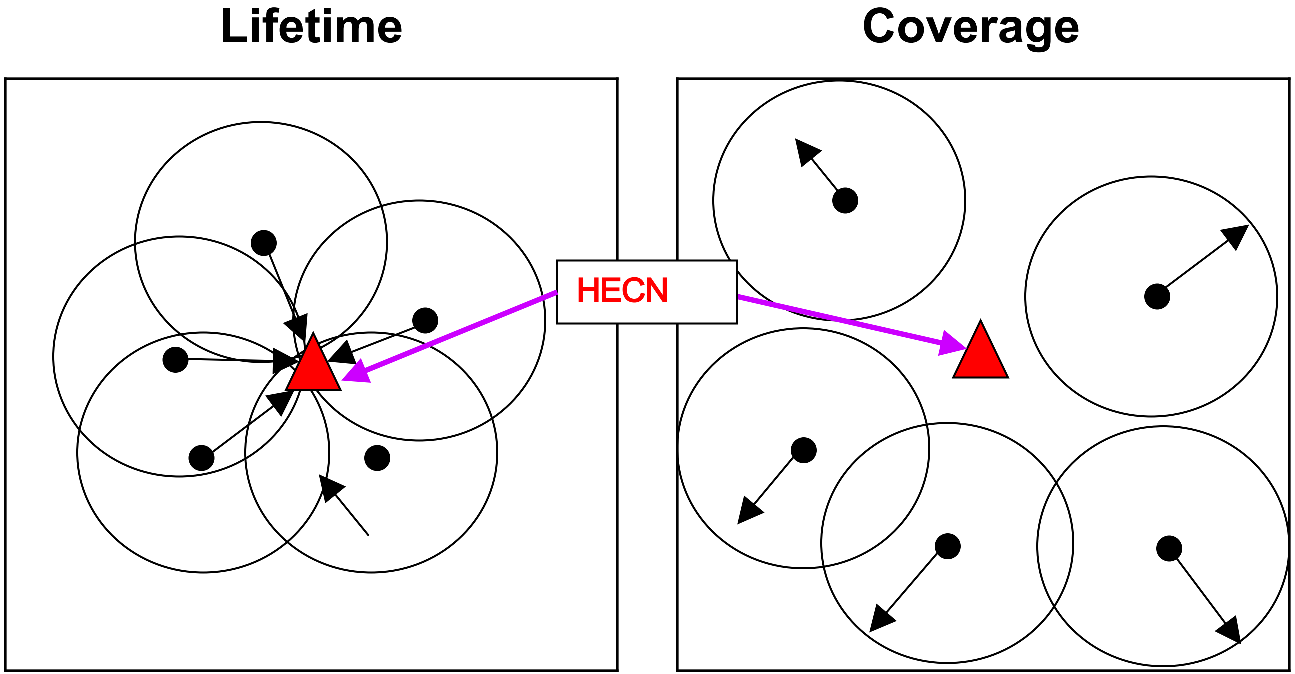
Optimization of Lifetime

- 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.

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

$T_{failure}$ = 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

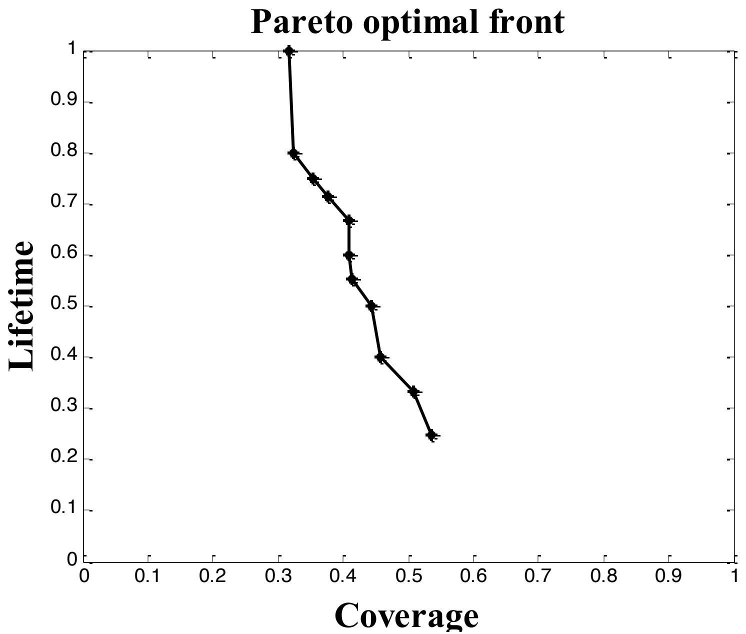
NTNU

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

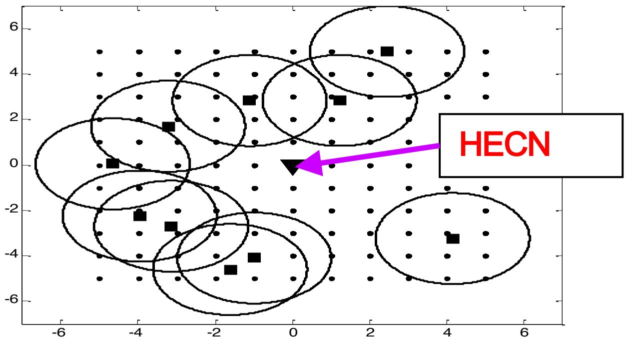
NSGA-II Results



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

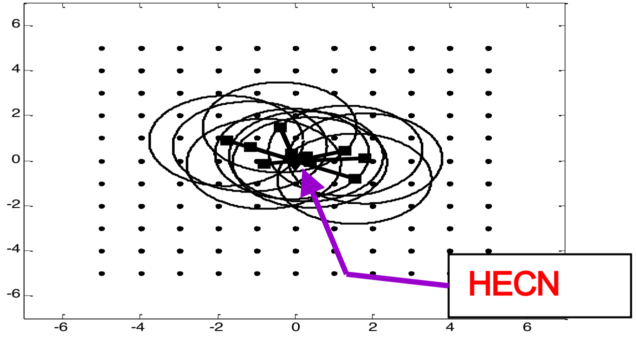
NSGA-II: Results

Initial Disconnect Network



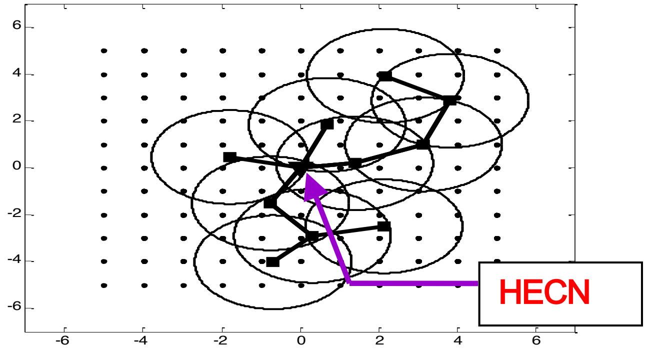
Best Lifetime

Coverage = 0.3335 Lifetime = 0.999



Best Coverage

Coverage = 0.5353 Lifetime = 0.249



Additional Reading + References

- Fonseca, C. M., & Fleming, P. J. (1993). Multiobjective genetic algorithms. In *IEE colloquium on Genetic algorithms for control systems engineering*, (pp. 6-1)
- Deb, K. (2001). *Multi-objective optimization using evolutionary algorithms* (Vol. 16). John Wiley & Sons.
- Deb, K., Pratap, A., Agarwal, S., & Meyarivan, T. A. M. T. (2002). A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE transactions on evolutionary computation*, *6*(2), 182-197.
- Horn, J., Nafpliotis, N., & Goldberg, D. E. (1994, June). A niched Pareto genetic algorithm for multiobjective optimization. In IEEE World Congress on Computational Intelligence Evolutionary Computation, 1994, pp. 82-87. IEEE.
- Zitzler, E., & Thiele, L. (1999). Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach. *IEEE transactions on Evolutionary Computation*, *3*(4), 257-271.