Solving MOOP: Pareto-based MOEA Approaches

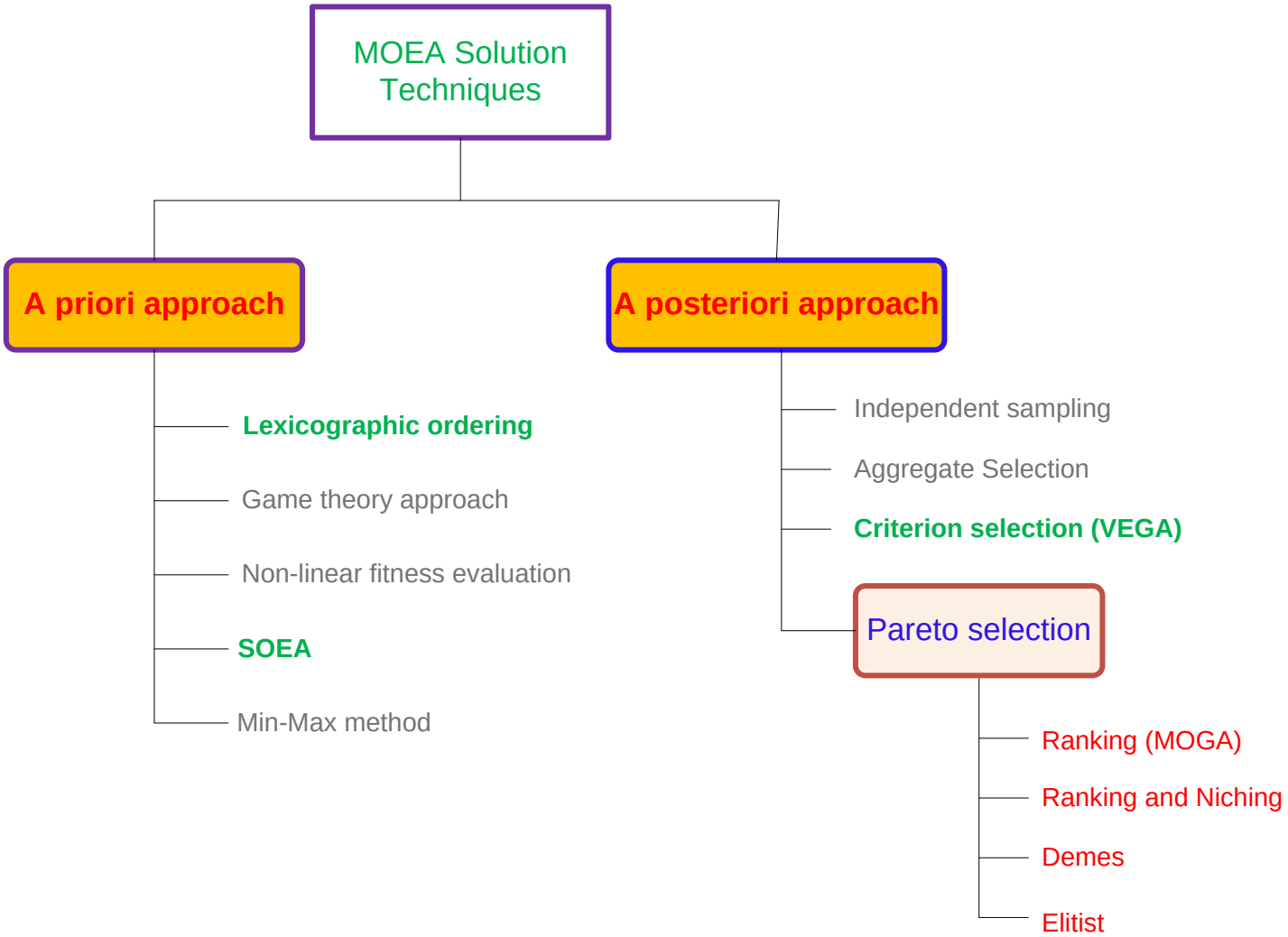
Debasis Samanta

Indian Institute of Technology Kharagpur

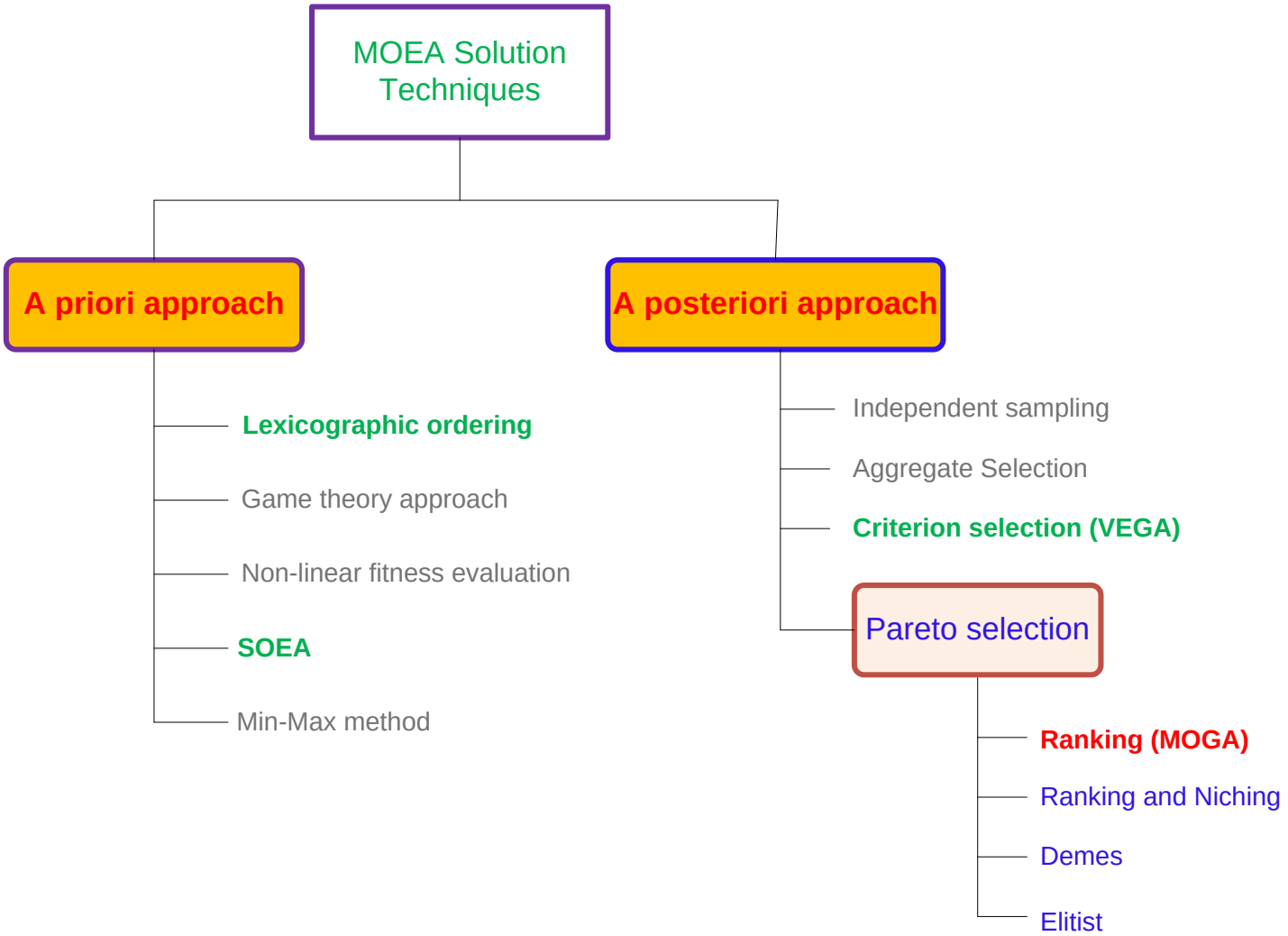
dsamanta@iitkgp.ac.in

09.04.2023

MOEA strategies



MOEA strategies



MOGA : Multi-Objective Genetic Algorithm

MOGA : Multi-Objective Genetic Algorithm

- It is Pareto-based approach based on the principle of ranking mechanism prposed by Carlos M. Fonseca and Peter J. Fleming (1993).

  Reference :

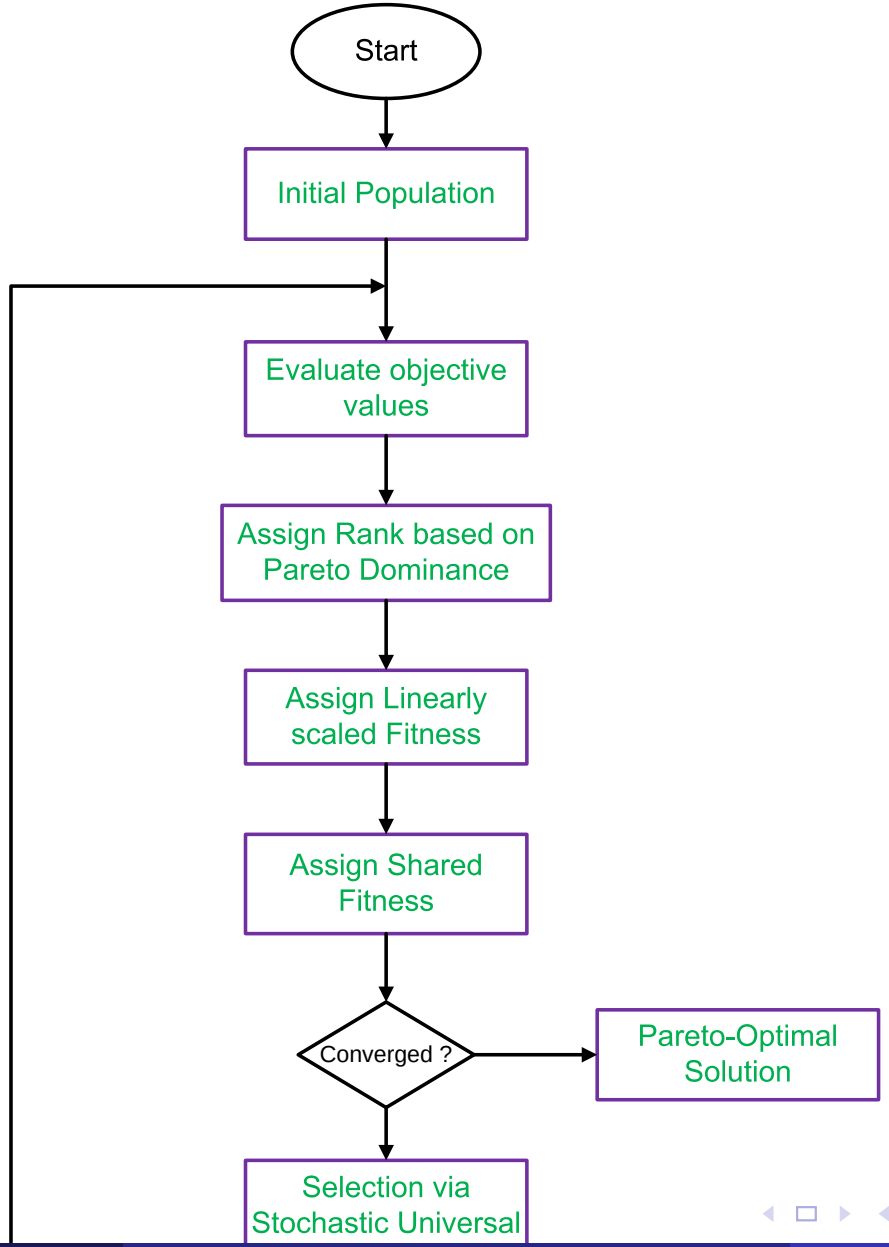
  C. M. Fonseca and P. J. Fleming, ”Genetic Algorithm for multi-objective Optimization : Formulation, Discussion and Generalization” in Proceeding of the 5^{th} International Conference on Genetic Algorithm, Page 416-423, 1993.

- Regarding the ”generation” and ”selection” of the Pareto-optimal set, ordering and scaling techniques are required.

- MOGA follows the following methodologies:

  For ordering: Dominance-based ranking,
  For scaling: Linearized fitness assignment and fitness averaging.

Flowchart of MOGA



Dominance-based ranking

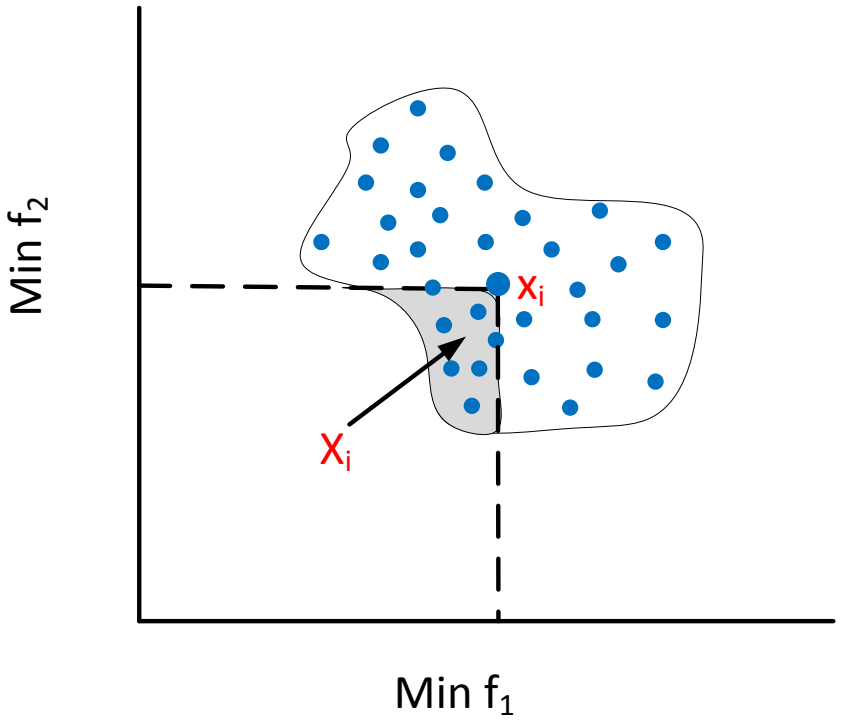
Definition 6 : Rank of a solution

The rank of a certain individual corresponds to the number of chromosomes in the current population by which it is dominated.

More formally,

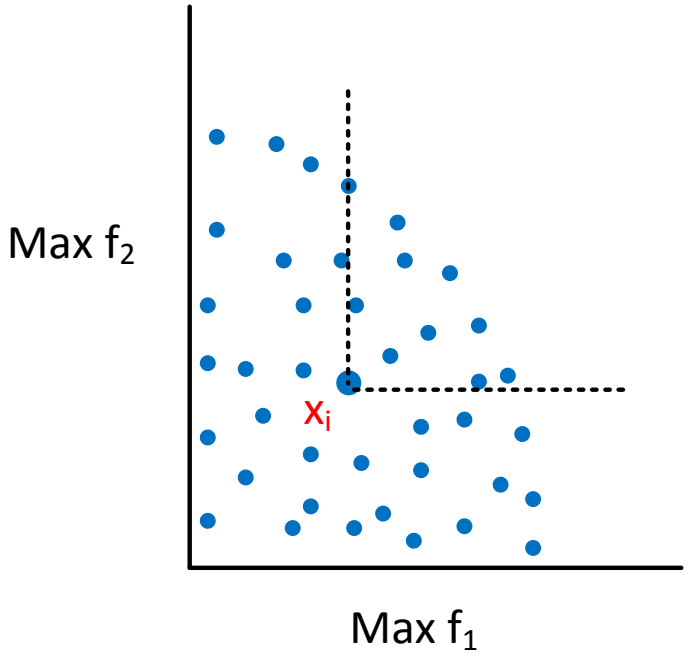
If an individual x_i is dominated by p_i individuals in the current generation, then $rank(x_i) = 1 + p_i$

Example 1: Dominance-based ranking



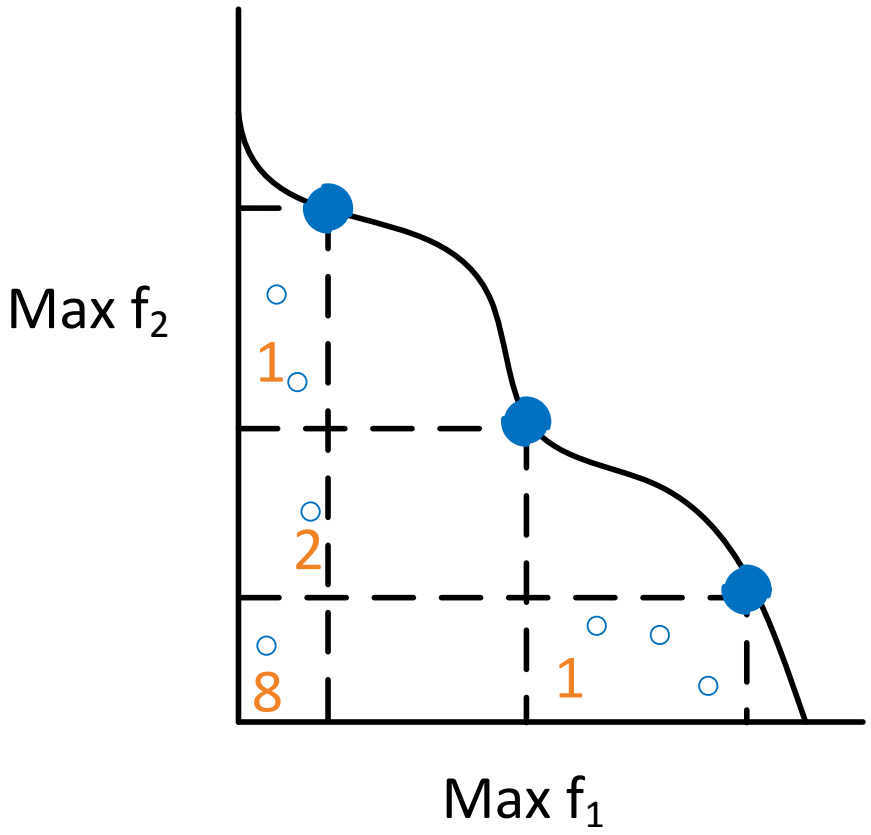
Rank(x_1)=1+$|x_i|$
Where $|x_i|$ = number of solutions in the shaded region

Example 2: Dominance-based ranking



Rank(x_1)=1+11=12

Example 3: Dominance-based ranking



Number of dominated points with their domination count

Interpretation : Dominance-based ranking

Note :

1. Domination count = How many individual does an individual dominates
2. All non-dominated individuals are assigned rank 1.
3. All dominated individuals are penalized according to the population density of the corresponding region of the trade-off surface.

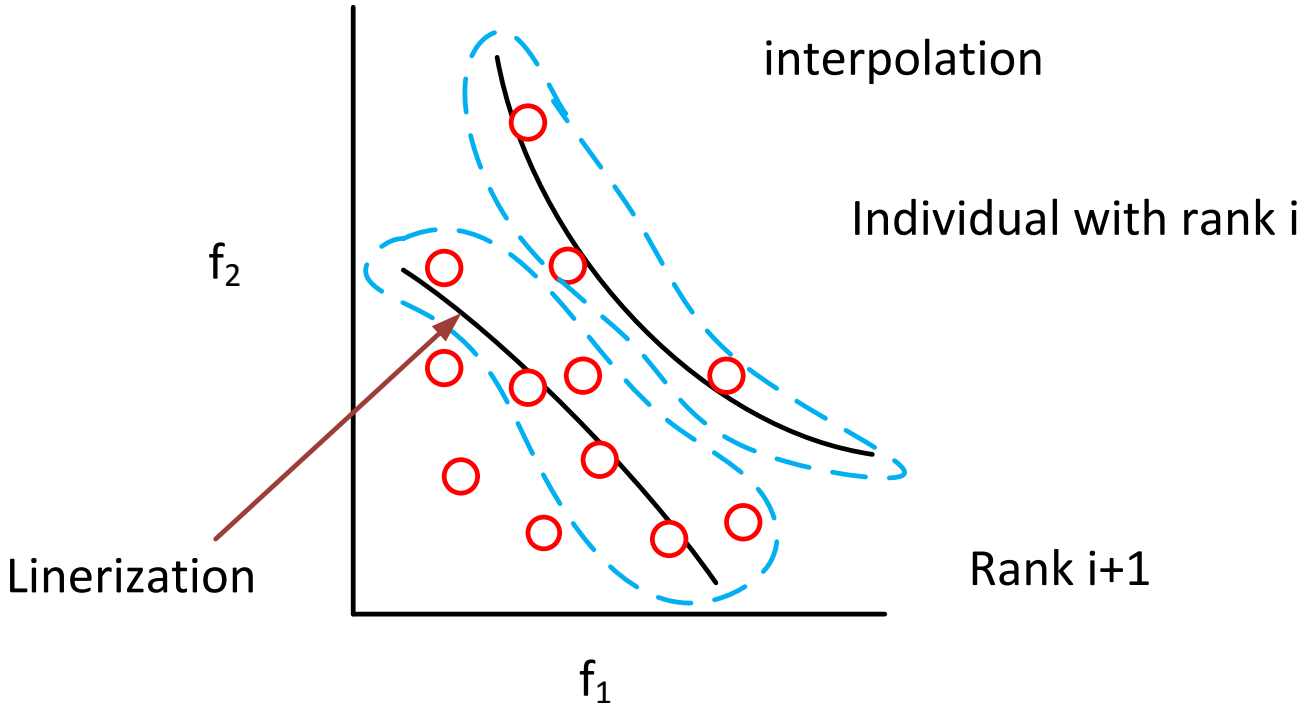
Fitness Assignment in MOGA

Steps :

1. Sort the population in ascending order according to their ranks.
2. Assign fitness to individuals by interpolating the best (rank 1) to the worst (rank $\leq N$, N being the population size) according to some linear function.
3. Average the fitness of individual with the same rank, so that all of them are sampled at the same rate.

This procedure keeps the global population fitness constant while maintaining appropriate selective pressure, as defined by the function used.

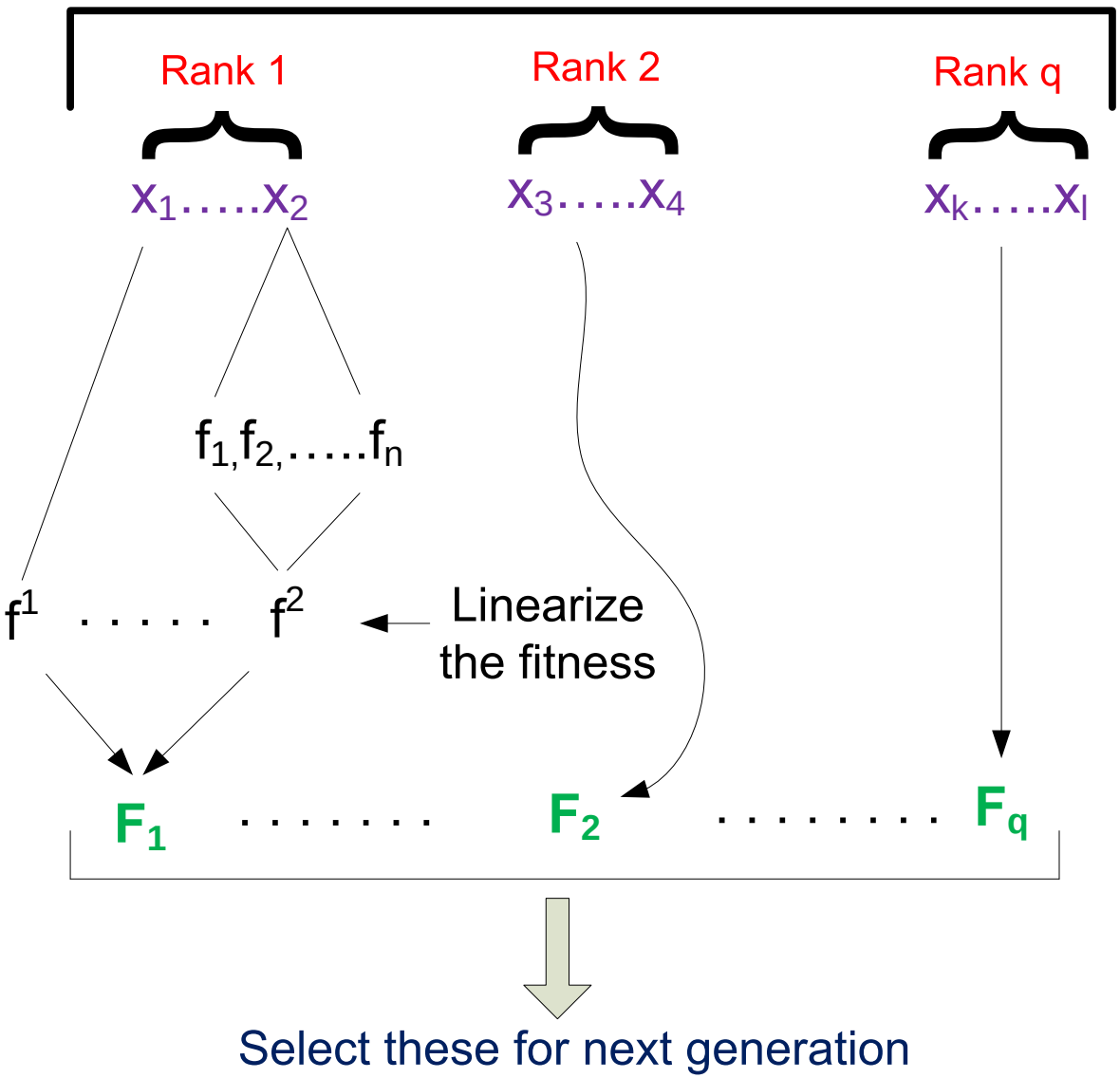
Fitness Assignment in MOGA



Example : Linearization = $\bar{f}_i = \sum_{j=1}^{k} \frac{f_j^i}{\bar{f}_j^i}$

where f_j^i denotes the j-th objective function of a solution in the i-th rank and $\bar{f}_j^i$ denotes the average value of the j-th objectives of all the solutions in the i-th rank.

Illustration of MOGA



Remarks on MOGA

- The fitness assignment (Step 3) in MOGA attempts to keep global population fitness constant while maintaining appropriate selection pressure.
- MOGA follows blocked fitness assignment which is likely to produce a large selection pressure that might lead to premature convergence.
- MOGA founds to produce better result (near optimal) in majority of MOOPs.

Niched Pareto Genetic Algorithm (NPGA)

J. Horn and N. Nafploitis, 1993 **Reference :** *Multiobjective Optimization using the Niched Pareto Genetic Algorithm* by J.Horn and N.Nafpliotis, Technical Report University of Illionis at Urbans-Champaign, Urbana, Illionis, USA, 1993

Niched Pareto Genetic Algorithm (NPGA)

- NPGA is based on the concept of tournament selection scheme (based on Pareto dominance principle).
- In this techniques, first two individuals are randomly selected for tournament.
- To find the winner solution, a comparison set that contains a number of other individuals in the population is randomly selected.
- Then the dominance of both candidates with respect to the comparison set is tested.
- If one candidate only dominates the comparison set, then the candidate is selected as the winner.
- Otherwise, niched sharing is followed to decide the winner candidate.

The above can be specified as follows.

Niched Pareto Genetic Algorithm (NPGA)

Pareto-domination tournament

let $N =$ size of the population, K is the no of objective functions.

Steps :

1. i=1 (The first iteration)
2. Randomly select any two candidates C_1 and C_2
3. Randomly select a "Comparison Set (CS)" of individuals from the current population
   Let its size be N^* (Where $N^* = P\%N$; P is decided by the programmer)
4. Check the dominance of C_1 and C_2 against each individual in CS

Niched Pareto Genetic Algorithm (NPGA)

4. If C_1 is dominated by CS but not by C_2 than select C_2 as the winner

   Else if C_2 is dominated by CS but not C_1 than select C_1 as the winner
   Otherwise Neither C_1 nor C_2 dominated by CS

   do_sharing (C_1, C_2) and choose the winner.
5. If $i = N'$ than exit (Selection is done)

   Else i=i+1, go to step 2

Niched Pareto Genetic Algorithm (NPGA)

- A sharing is followed, when there is no preference in the candidates.
- This maintains the genetic diversity allows to develop a reasonable representation of Pareto-optimal front.
- The basic idea behind sharing is that the more individuals are located in the neighborhood of a certain individual, the more its fitness value is degraded.
- The sharing procedure for any candidate is as follows.

Niched Pareto Genetic Algorithm (NPGA)

Procedure do_sharing(C_1, C_2)

1. j=1. Let $x = C_1$
2. Compute a normalized (Euclidean distance) measure with the individual x_j in the current population as follows,

$$d_{xj} = \sqrt{\sum_{i=1}^{k} \left(\frac{f_i^x - f_i^j}{f_i^U - f_i^L} \right)^2}$$

where f_i^j denotes the i-th objective function of the j-th individual

f_i^U and f_i^L denote the upper and lower values of the i-th objective function.

Niched Pareto Genetic Algorithm (NPGA)

3. Let σ_{share}= Niched Radius
   Compute the following sharing value

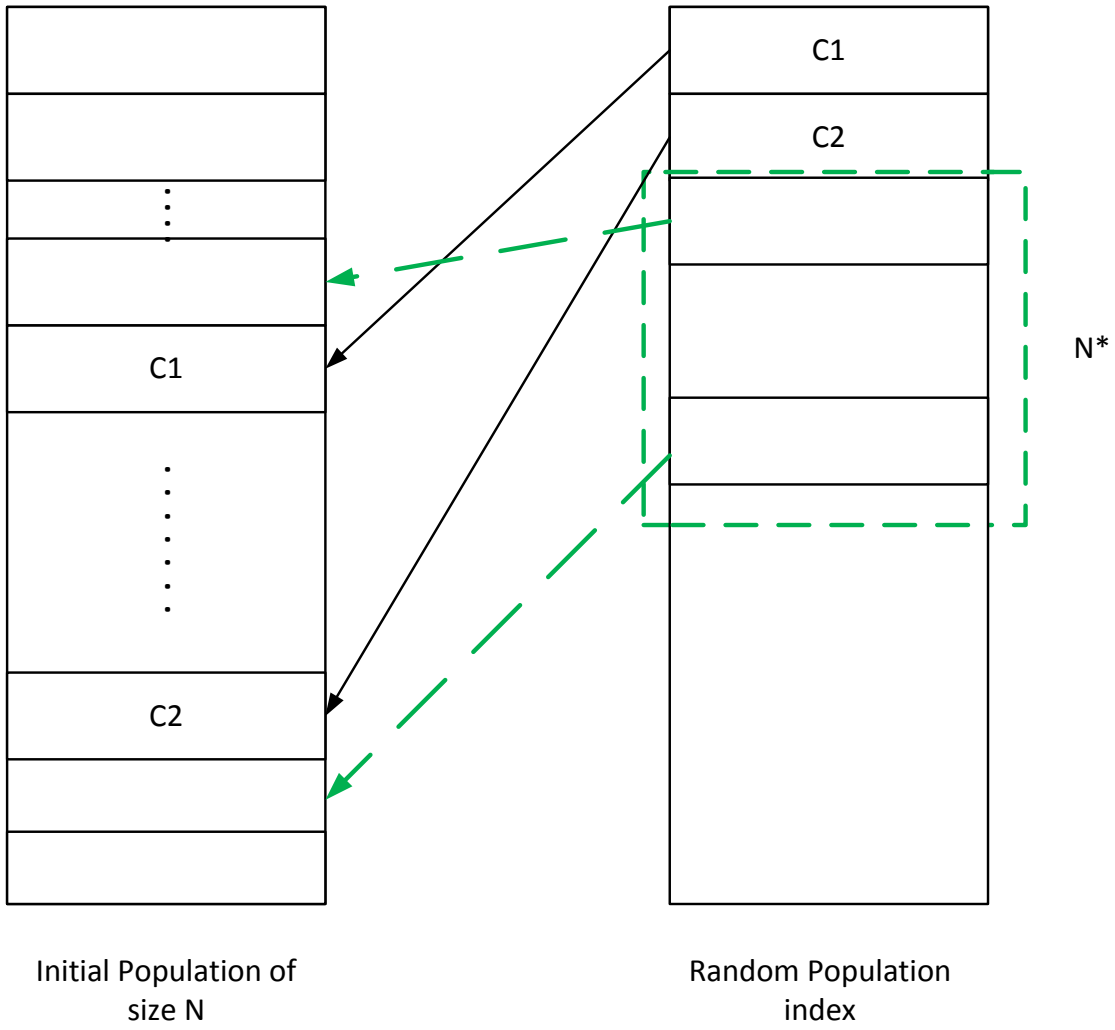
$$sh(d_{xj}) = \begin{cases} 1 - \left(\frac{d_{xj}}{\sigma_{share}}\right)^2 & \text{, if } d_{xj} < \sigma_{share} \\ 0 & \text{, otherwise} \end{cases}$$

4. Set $j = j + 1$, if $j < N$, go to step 2 else calculate "Niched Count" for the candidate as follows

$$n_1 = \sum_{j=1}^{N} sh\left(d_{ij}\right)$$

5. Repeat step 1-4 for C_2.
   Let the niched count for C_2 be n_2
6. if $n_1 < n_2$ then choose C_1 as the winner else C_2 as the winner.

Niched Pareto Genetic Algorithm (NPGA)



Niched Pareto Genetic Algorithm (NPGA)

This approach proposed by Horn and Nafploitis [1993]. The approach is based on tournament scheme and Pareto dominance. In this approach, a comparison was made among a number of individuals (typically 10%) to determine the dominance. When both competitors are dominated or non-dominated (that is, there is a tie) the result of the tournament is decided through fitness sharing (also called equivalent class sharing)
The pseudo code for Pareto domination tournament assuming that all of the objectives are to be maximized is presented below. Let us consider the following.

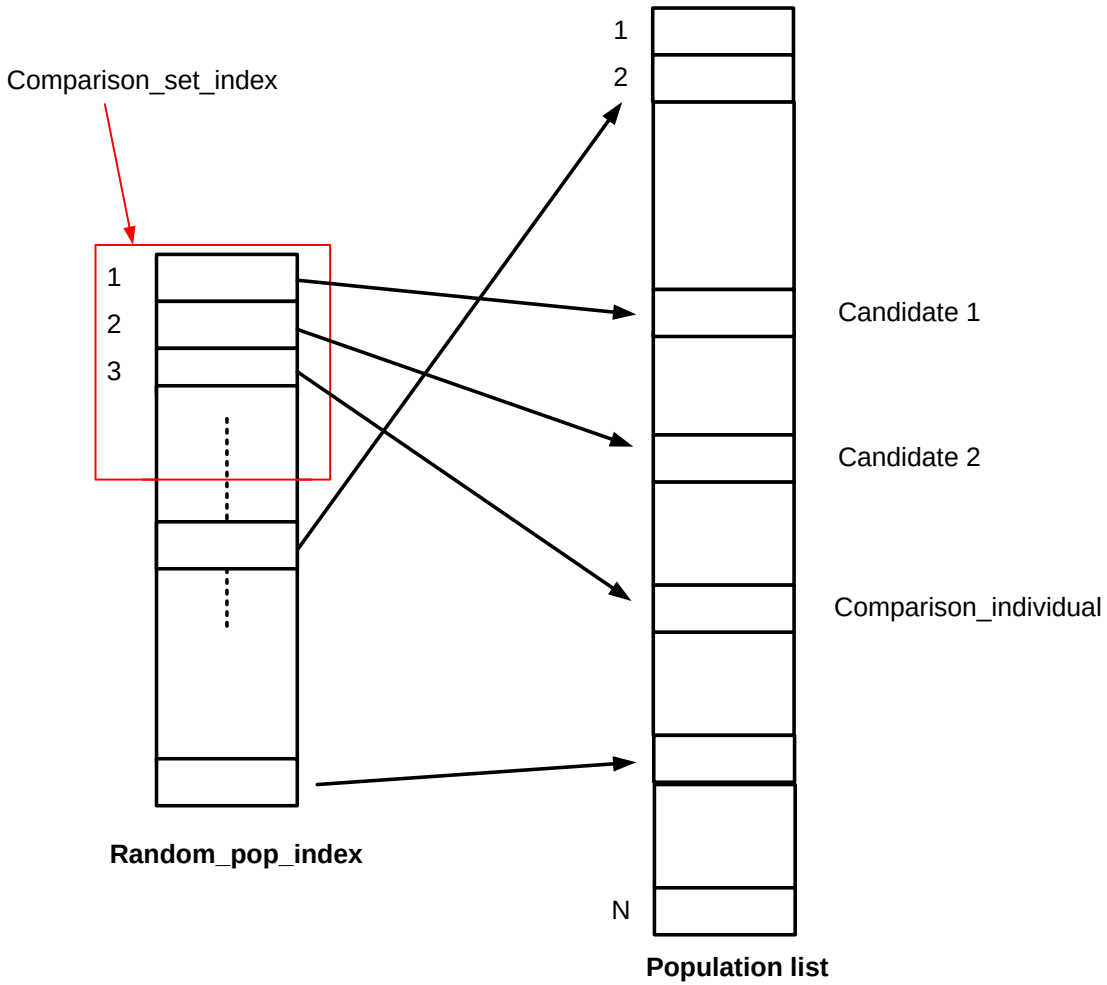
Pareto domination Tournament

S = an array of N individuals in the current population.

random_pop_index = an array holding N individuals of S, in a random order.

t_{dom}= the size of the comparison set.

Algorithm Selection



Algorithm Selection

This algorithm returns an individual from the current population S.

Begin

shuffle(*random_pop_index*)
candidate_1 = *random_pop_index*[1];
candidate_2 = *random_pop_index*[2];
candidate_1_dominated = *F*;
candidate_2_dominated = *F*;
for *comparison_set_index* = 3 to $t_{dom} + 3$ **do**
comparison_individual = *random_pop_index*[*comparison_set_index*];
    if *s*[*comparison_set_index*]*dominatess*[*candidate_1*] **then**
        candidate_1_dominated = *TRUE*;
    end if
    if *s*[*comparison_set_index*]*dominatess*[*candidate_2*] **then**
        candidate_2_dominated = *TRUE*;
    end if
end for

Algorithm Selection

if *candidate_1_dominatedand* $\sim$ *candidate_2_dominated* **then**
    return candidate_1
else if *candidate_2_dominatedand* $\sim$ *candidate_1_dominated* **then**
    return candidate_2
else
    if *nichecount*(*candidate_1*) $>$ *nichecount*(*candidate_2*) **then**
        return candidate_2
    else
        return candidate_1
    end if
end if
END

Algorithm Selection

This approach does not apply Pareto selection to the entire population, but only to a segment of it at each num, the technique is very first and produces good non-dominated num that can be kept for a large number of generation.

However, besides requiring a sharing factor, this approach also requires a good choice of the value of t_{dom} to perform well, complicating its appropriate use in practice.

Points to Ponder on MOEA

- How you should solve an optimization problem?
  - Strategy 1 : Solve individually, that is, one objective at a time ignoring the other objective(s)
  - Strategy 2 : Solve one objective as the main and other(s) as the constraint(s)

  Justify the above-mentioned two strategies.
- What are the issues with if one objective is to minimize and another to maximize?
- Explain weighted-sum approach. What are the issues with this? How Pareto-based approach addresses these issues?

Non-dominated Sorting Genetic Algorithm(NSGA)

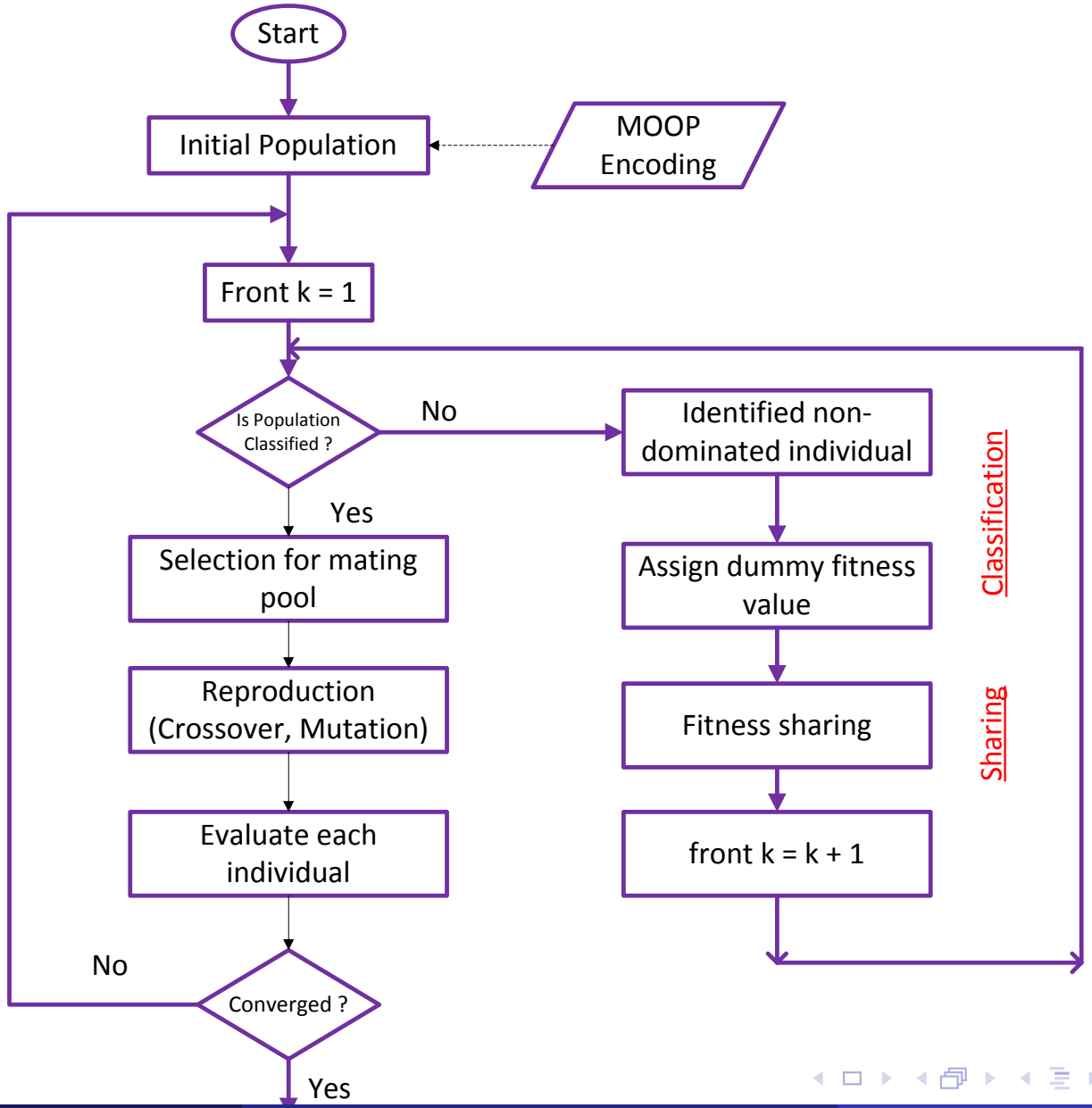
N.Srinivas and K.Deb, 1994

Reference : *Multi-objective Optimization using Non-dominated Sorting in Genetic Algorithm* by N.Srinivas and K.Deb, IEEE Transaction on Evolutionary Computing, Vol. 2, No. 3, Pages 221-248, 1994.

The algorithm is based on the concept of

1. Non-dominated sorting procedure (a rank-based selection method to select good non-dominated points.
2. Niche method (is used to maintain a subpopulation with a chance for population diversity)

Flowchart of NSGA

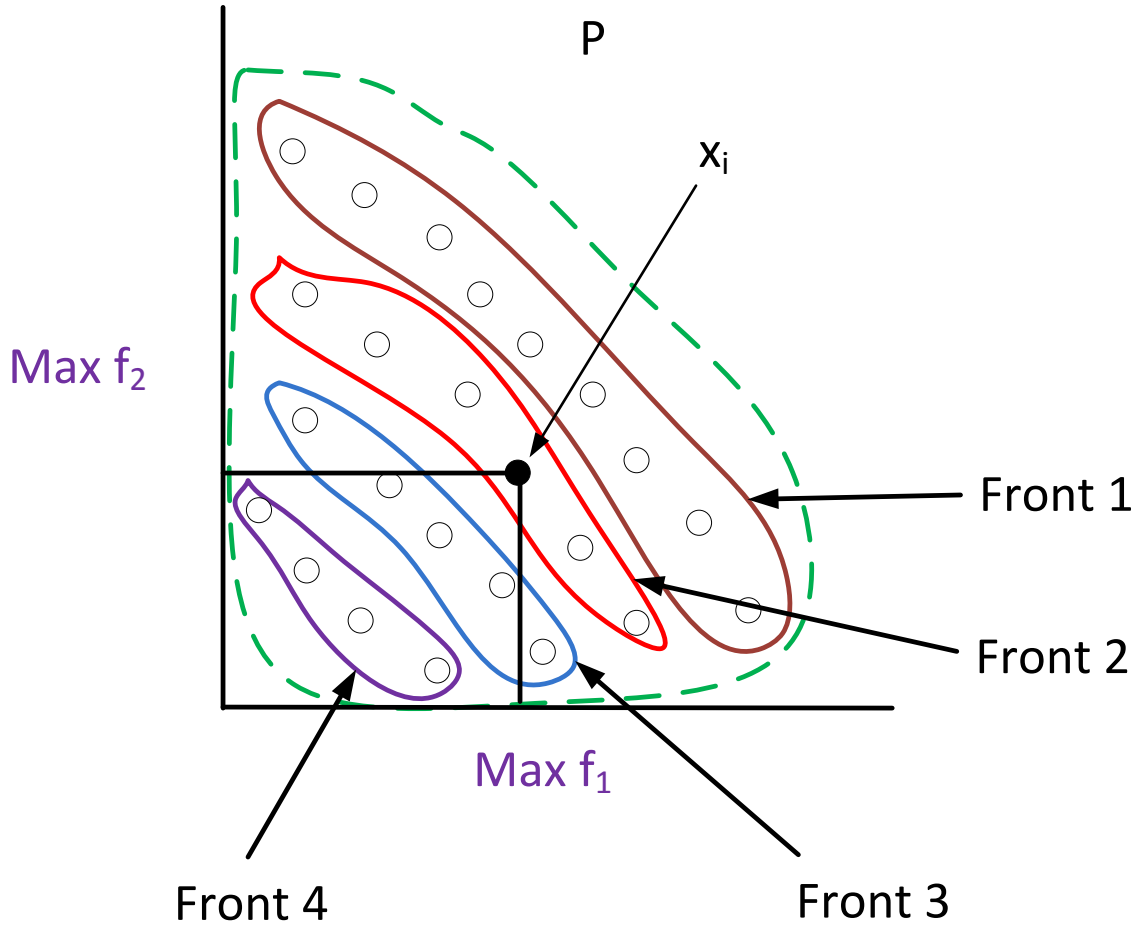


NSGA Technique

- NSGA is based on the classification of the several fronts of individuals.
- It finds a front (of non-dominated individuals) at a particular rank and then assign a fitness value (called dummy fitness value) to each solution in the front followed by sharing of fitness value.
- It then selects some individuals based on the shared fitness values of the individuals for mating pool.
- Reproduce offspring for next generation and thus completes an iteration.

Identification of Non-dominated Fonts

Non-dominated Fonts of Different Orders



Non-dominated Sorting Procedure

Notations:

- x_i - denotes an i-th solution
- S_i - denotes a set of solutions which dominate the solution x_i
- P - denotes a set of solutions (i.e., the current population)
- P_k - denotes a non-domination front at the k-th level (output as sorting based on the non-domination rank of solution)
- n_i denotes the domination count, that is, the number of solutions which dominates x_i

Non-dominated Sorting Procedure

Steps :

1. For each $x_i \in P$ do
   1. For all $x_j \neq x_i$ and $x_j \in P$

      If $x_i \leq x_j$ (i.e. if x_i) dominates x_j

      then $s_i = s_i \cup x_j$

      else if $x_j \leq x_i$ then $n_i = n_i + 1$
   2. If $n_i = 0$, keep x_i in P_1 (the first front). Set a front counter $K = 1$

   $< --$ This completes the finding of first front $-- >$
2. Enumerating other fonts

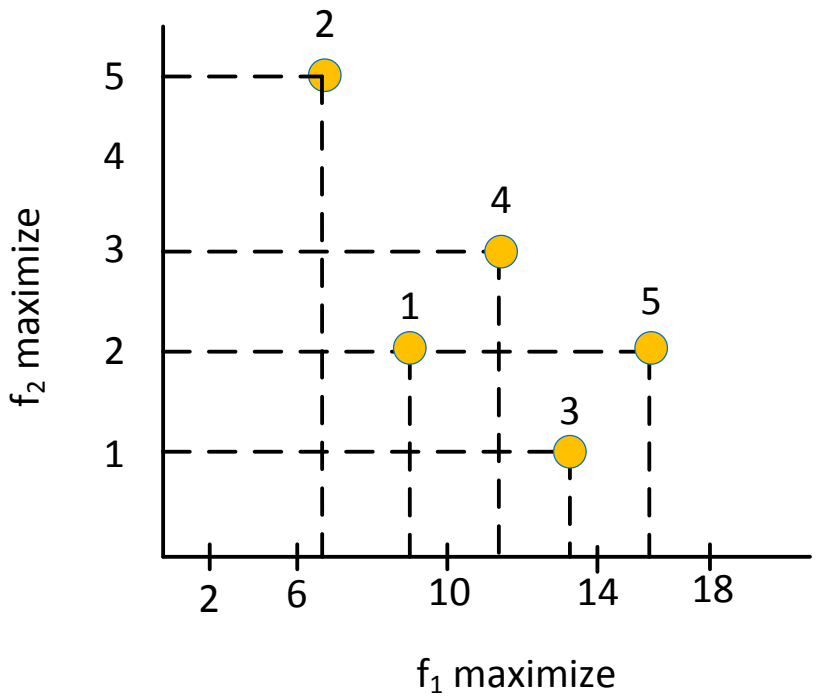
Non-dominated Sorting Procedure

3. While $P_k = \phi$ do
   1. [*Initialize*] $Q = \phi$ (for sorting next non-dominated solutions)
   2. For each $x_i \in P_k$ and For each $x_j \in S_i$ do
      - Update $n_j = n_j - 1$
      - If $n_j = 0$ then x_j in Q Else $Q = Q \cup x_j$
   3. Set $k = k + 1$ and $P_k = Q$ (Go for the next front)

Non-dominated Sorting Procedure

Note : Time complexity of this procedure is $O(MN^2)$, where M is the number of objective and N is the population size.

An Illustration



Here, 1 dominates 2

5 dominates 1 etc.

Non-dominated Sorting Procedure

- Three non-dominated fronts as shown.
- The solutions in the front $[3, 5]$ are the best, follows by solutions $[1, 4]$.
- Finally, solution [2] belongs to the worst non-dominated front.
- Thus, the ordering of solutions in terms of their non-domination level is: $[3, 5], [1, 4], [2]$

Assignment of Dummy Fitness Values

Assign dummy fitness value

- Once all non dominated individual are identified for a front, it assigns a unique fitness value to each belongs to that front. This unique value is called dummy fitness value.
- The dummy fitness value is a very large number and typically is proportional to the number of population (which includes the number of individuals in all fronts including the current front).
- The proportional constant (≥ 1) is decided by the programmer.

Assign dummy fitness value

Note :

- The same fitness value is assigned to give an equal reproductive potential to all the individuals belong to the front.
- A higher value is assigned to individuals in an upper layer front to ensure selection pressure, that is, having better chance to be selected for mating
- As we go from an upper front to the next lower front, count of individuals in the upper fronts are ignored (i.e. logically remove from the population set). In other words, the number of population successively decreases as we move from one front to the next front.

Fitness Assignment

Sharing the fitness value

- In the given front, all individuals are assigned with a dummy fitness value which is proportional to the number of population.
- These classified individuals are then shared with their dummy fitness values.
- Sharing is achieved by dividing the dummy fitness value by ”a quantity proportional to the number of individuals around it”. This quantity is called niche count.
- This quantity is calculated by computing ”Sharing coefficient” denoted as $sh(d_{ij})$ between the individual x_i,x_j belong to the front under process, as per following.

$$sh(d_{ij}) = \begin{cases} 1 - \left(\frac{d_{ij}}{T_{shared}}\right)^2 & \text{, if } d_{ij} < T_{shared} \\ 0 & \text{, otherwise} \end{cases}$$

In the above, the parameter d_{ij} is the ”phenotype distance” between two individuals x_i and x_j in the current front.

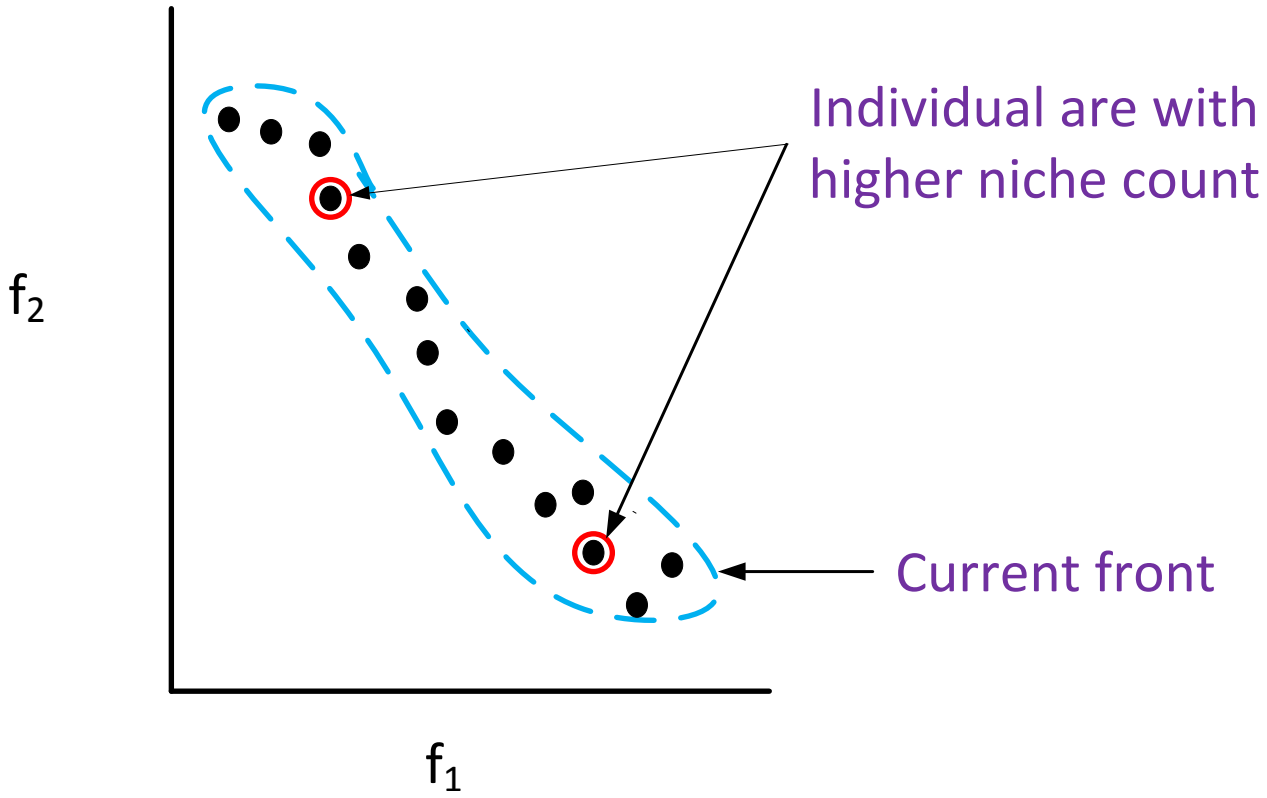
Sharing of fitness values

- T_{share} is the maximum phenotype distance allowed between any two individuals to become members of a niche. This value should be decided by the programmer.
- Niche count of x_i then can be calculated as

$$\gamma(x_i) = \sum_{\forall x_j \in p_k} Sh(d_{ij})$$

  where P_k denotes the set of non dominated individual in the current front k).
- The shared fitness value of x_i is then $\bar{f}_i = \frac{f}{\gamma(x_i)}$, where f is the dummy fitness value.

NSGA Technique



Sharing the fitness value

Note :

- The sharing is followed to ensure the population diversity. That is, give a fair chance to those individuals which are not so niche.
- $sh(d_{ij}) \neq 0$, and it always less than or equal to 1 (=1 when $d_{ij} = 0$).
- After the fitness sharing operation, all individuals in the current front are moved to a population set to be considered for mating subject to their selection.
- The assignment of dummy fitness value followed by fitness sharing is then repeated for the next front (with a smaller population size).

NSGA Selection

- Once all fronts are processed, we obtained a set of population with their individual shared fitness values.
- This population is then passed through a selection procedure.
- NSGA follows ”stochastic remainder proportionate selection”, which is as follows.
  1. Calculate cumulative probabilities of all individuals as in Roulette-Wheel scheme.
  2. Generate a random number (for first time only) r_i (this is the probability to get an individual to be selected).
  3. If $P_{j-1} \leq r_i < P_j$ (where P_j and P_{j-1} are two cumulative probabilities, then consider j-th individual for the selection.
  4. Let $E_j = P_j \times N$ (N, the population size and E_j denotes expected count).
  5. If integer part in E_j is nonzero, then select j-th solution for mating.
  6. The fractional part is used as the random number for the selection of next solution.
  7. Repeat step 3-6 until the mating pool of desired size is not reached.

Reproduction and generation of new population

Follow the standard reproduction procedure as in Simple GA.

Remark on NSGA

- Several comparative studies reveal that NSGA is outperformed by both MOGA and NPGA.
- NSGA is also highly inefficient algorithms because of the way in which it classified individuals (if is $O(MN^3)$) time complexity)
- It needs to specify a sharing parameter.
- T_{share} It is a non-elitism approach.
- K. Deb et al. proposed an improved version of NSGA algorithm called NSGA-II.

NSGA-II

Elitist Multi-objective Genetic Algorithm : NSGA-II

K. Deb, A. Protap, S. Agarwal and T. Meyarivan 2002.

Reference :

A Fast and Elitist Multi-objective Genetic Algorithm : NSGA-II by K. Deb, A. Pratap, S. Agrawal and T. Meyarivan in IEEE Transaction on Evolutionary Computation, Vol.6, No.2, Pates 182-197, April 2002.

Elitist Multi-objective Genetic Algorithm : NSGA-II

Notations

P = A set of input solution (i.e., the current population)

x_i and x_j denote any two solutions

P' = set of solutions on non-dominated front.

Approach

In this approach every solution from the population P is checked with a partially filled population P' (which is initially empty). To start with, the first solution from P is moved into initially empty P'. Thereafter, each solution $x_i \in P$ one by one is compared with all member of the set P'. If the solution x_i dominates any member of P', then that solution is removed from P'. Otherwise, if solution x_i is dominated by any member of P', the solution x_i is ignored. On the other hand, if solution x_i is not dominated by any member of P' it is moved in to P'. This is how the set P' grows with non dominated solutions. When all solutions of P are checked, the remaining members of P' constitute the solutions on non-dominated front.

Elitist Multi-objective Genetic Algorithm : NSGA-II

This front is removed and the same steps are repeated with the remaining solutions to find the next non-domination front until P is empty. This approach is precisely stated in the following.

Steps :

$P = x_1, x_2, \cdots, x_N$

1. Initialize $P' = x_i$ set solution counter $j = 2$.
2. Let $j = 1$.
3. Compare solution x_i with x_j from P for domination
4. If $x_i \leq x_j$, delete x_j from P'
5. If $j < |P'|$, then increment j by one and go to step 3.
   Else Go to step 7.

Elitist Multi-objective Genetic Algorithm : NSGA-II

6. If $x_j \leq x_i$, then increment i by one and go to step 2.
7. Insert x_i in P' *i.e.* $P' = P' \cup i$
8. If $i < N$, increment i by one and go to step 2.
9. Output P' as the non-dominated from with current population
10. if $P \neq \phi$, repeat Step 1-9.
11. Stop.

Note : Time complexity of this procedure is $O(MN^2)$ and in worst case $O(MN^3)$ (when one front contains only one solution.)

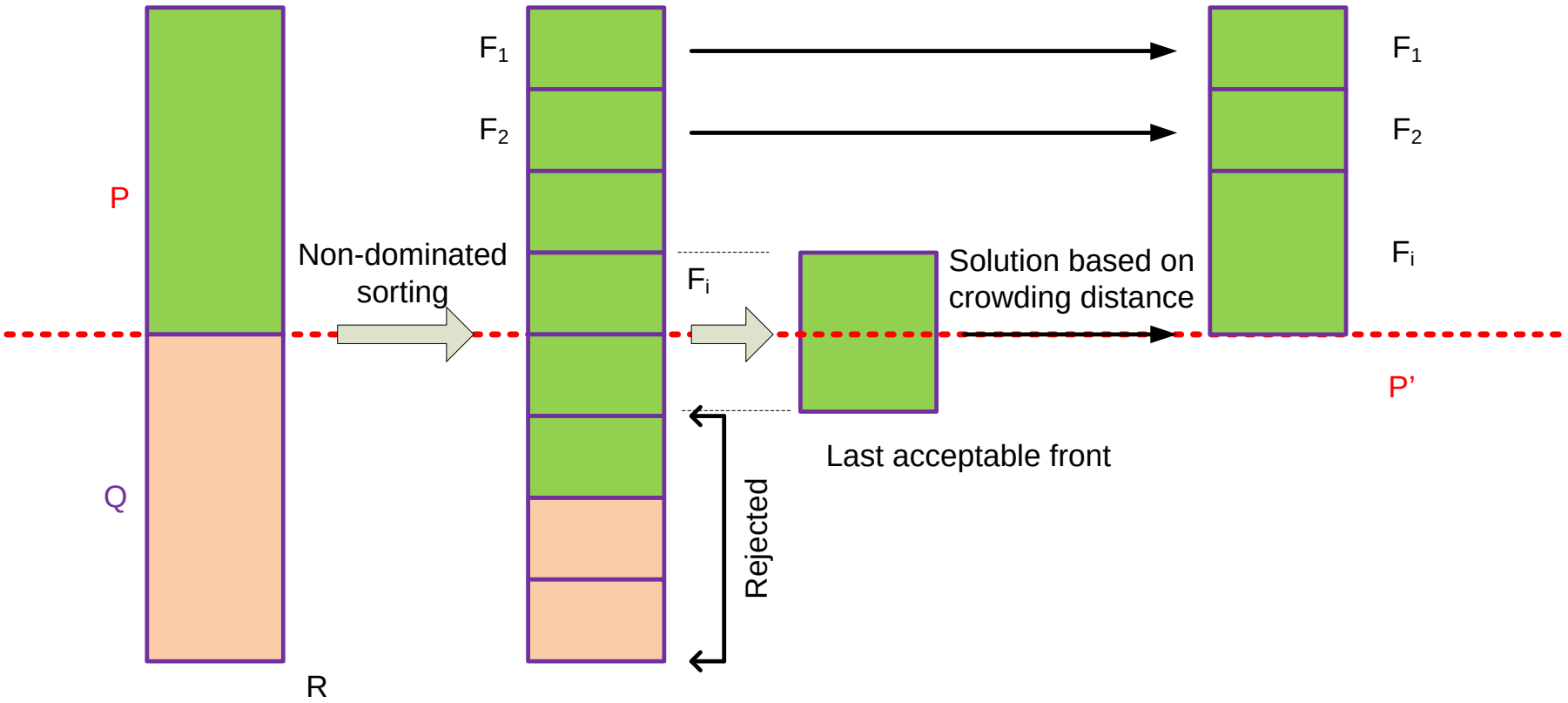
Overview of NSGA-II

- Let P be the current population
- Create offspring population Q from P using the genetic operators (mating pool, crossover and mutation)
- Two population are combined together to form a large population R of size of $2N$
- Apply non-dominating sorting procedure to classify the entire population R.
- After the non-domination sorting, the new population P is obtained by fitting the non-dominated front one-by-one until the $|P'| < |P|$
  - The filling starts with the best non-dominated front followed by the second non-dominated front, followed by the third non-dominated front and so on.

Overview of NSGA-II

- Since the total population size is $2N$, not all fonts may be accommodated in N solutions available in P'. All fronts which could not be accommodated are simply rejected. when the last allowed front is being considered, there may exist more solutions in the last front than the remaining slots in the new population.
- Instead of arbitrarily discarding some members from the last acceptable front, the solution which will make the diversity of the selected solutions the highest are chosen.
- This is accomplished by calculating crowding distance of solutions in the last acceptable front.
- This way a new generation was obtained and the steps are repeated until it satisfies a termination condition.

NSGA Technique



Basic Steps : NSGA-II

Steps :

1. Combine parent (P) and offspring (Q) populations and obtain $R = P \cup Q$
2. Perform a non-dominated sorting to R and identify different fronts of non-dominated solutions as F_i, $i = 1, 2, \cdots$, etc.
3. Set new population $P' = \phi$. Set a new counter $i = 1$ (to indicate front to be allowed to fill P'.
4. Fill P' until $|P'| + |F_i| < N$ that is, $P' \cup F_i$ and $i = i + 1$.
5. Perform Crowding sort (F_i) and include the most-widely spread $N - |P'|$ solution by using the crowding distance values in the sorted F_i to P'.
6. Create offspring population Q' from P' by using the crowded-tournament selection, crossover and mutation operator.
7. $P = P'$, $Q = Q'$.
8. Repeat step 1-7 until the convergence of solutions.

Crowding sort procedure

The crowding sort follows the following two concepts.

- Crowding distance metric (d) and
- Crowded comparison operator $<_e$

Note :

- The crowding distance d_i of a solution x_i is a measure of the search space around x_i which is not occupied by any other solutions in the population. (It is just like a population density : d_i is lower for a dense population.
- The crowded comparison operator $<_e$ (a binary operator) to compare two solutions and returns a winner.

Crowding sort procedure

Definition 8 : Crowding comparison operator

A solution x_i wins over a solution x_j, iff any of the following conditions are true

- If solution x_i has a better rank (i.e. dominance rank). That is $rank(x_i) < rank(x_j)$
- If they have the same rank but solution x_i has better crowding solution x_j.
- That is $rank(x_i) = rank(x_j)$ and $d_i > d_j$ (where d_i and d_j are crowding distance of x_i and x_j, respectively.

Crowding sort procedure

Note :

- The first condition ensures that x_i lies on a better non-domination front.
- The second condition resolve the tie of both solution being on the same non-dominated front by deciding on their crowding distances

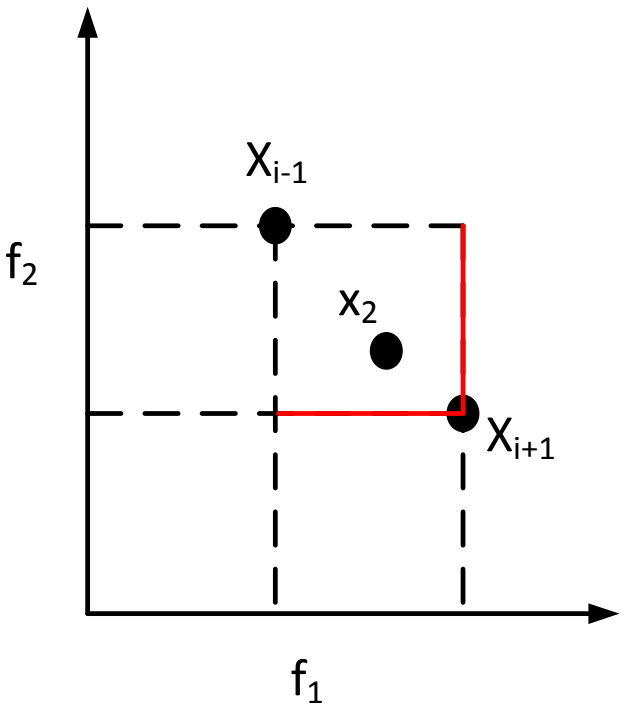
(Note : For NSGA-II, only the second condition is valid as all solution are belong to one front only.

Crowding sort procedure

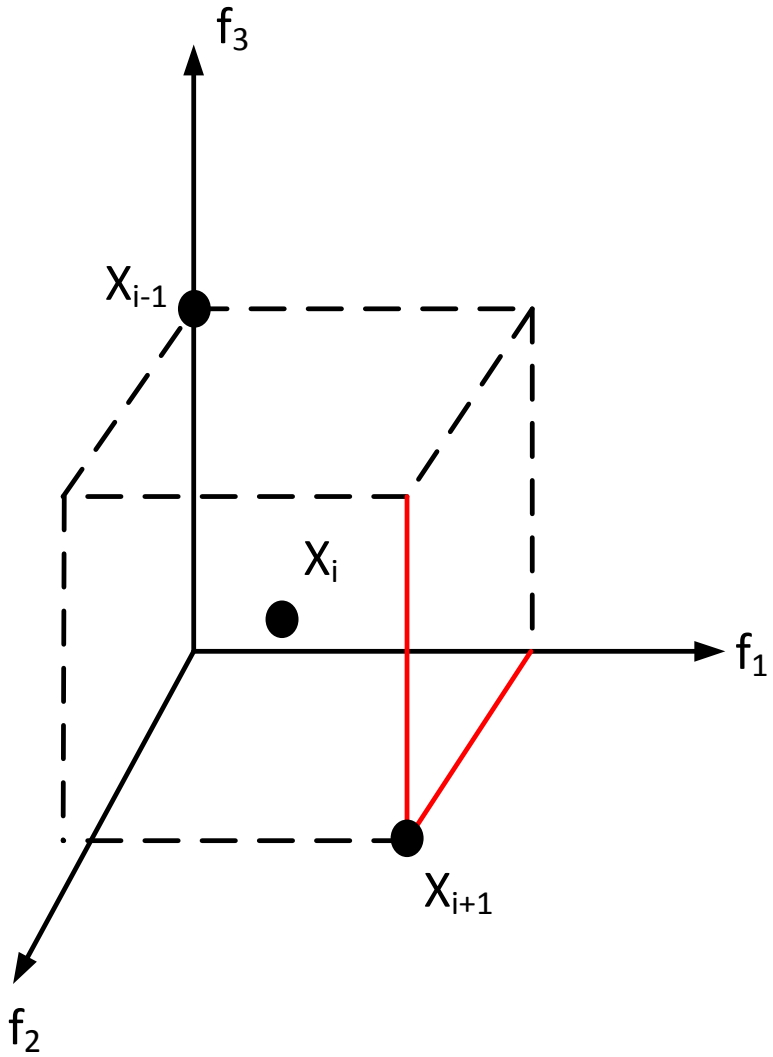
Definition 7 : Crowding distance measure

The crowding distance measure d_i is an estimate of the size of the largest cuboid enclosing the point x_i (i-th solution) without including any point in the population.

Crowding sort procedure



(a) Solutions in 2D space



(b) Solutions in 3D space

Crowding sort procedure

- In Fig(a), the crowding distance is the average side length of the rectangle (so that two nearest solutions of x_i and $x_i - 1$ at two opposite corners).
- In fig.(b), the crowding distance is the sum of three adjacent edges of a cuboid formed by the two neighbors x_{i-1} and x_{i+1} of x_i at two opposite corners of the cuboid.

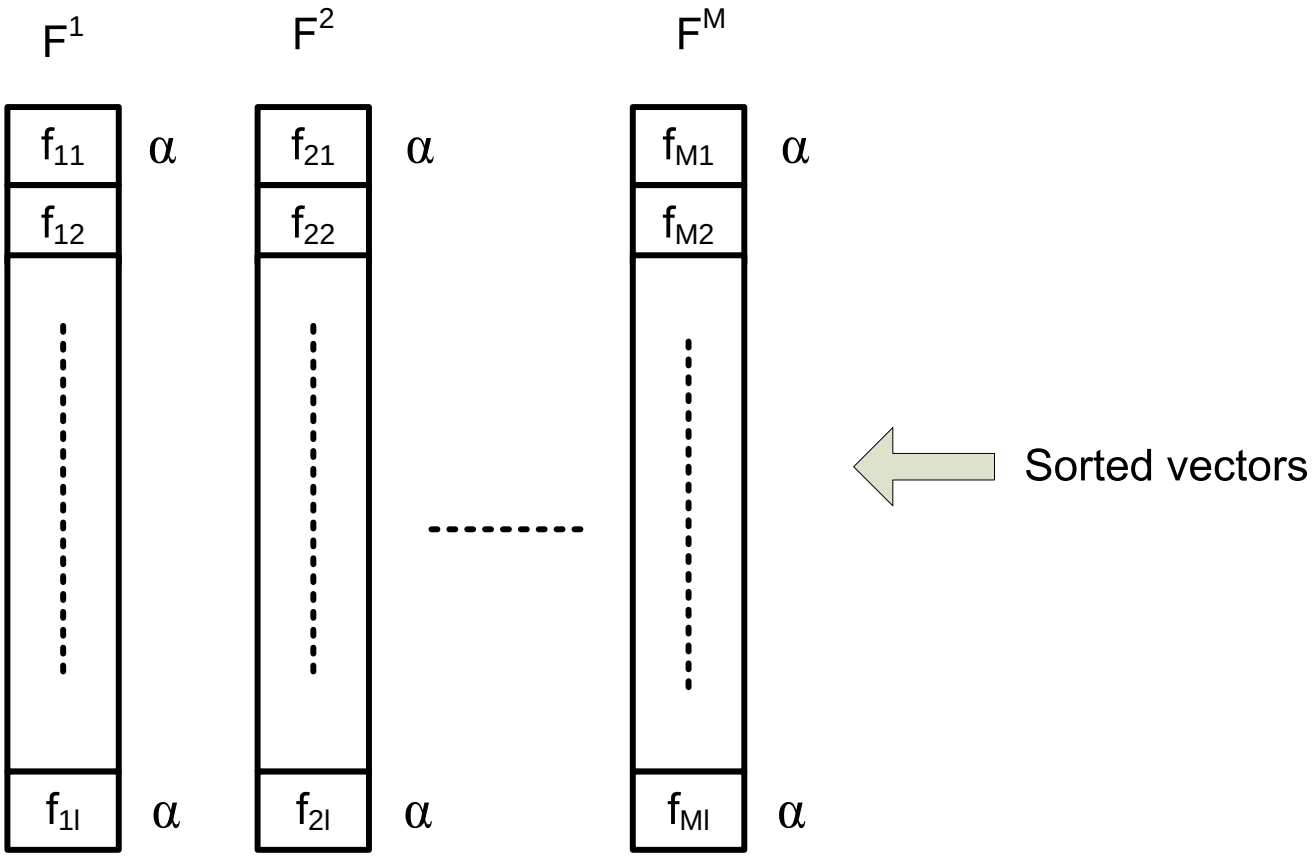
Crowding distance calculation

Given a set of non-dominated solutions say F, and objective functions $f = f_1, f_2, \cdots, f_M$, the procedure to calculate the crowding distance of each solution $x_i \in F$ is stated below. **Steps :**

1. Let $l = |F|$ (Number of solutions in F)
2. For each $x_i \in F$, set $d_i = 0$ (Initialize the crowding distance)
3. For each objective $f_i \in f$
   - $F^i = sort(F, i)$
     - Sort all solutions in F with respect to objective values f_i
     - This will result $F^1, F^2, \cdots, F^m$ sorted vector.
     - The sorting is performed in ascending order of objective values f_i

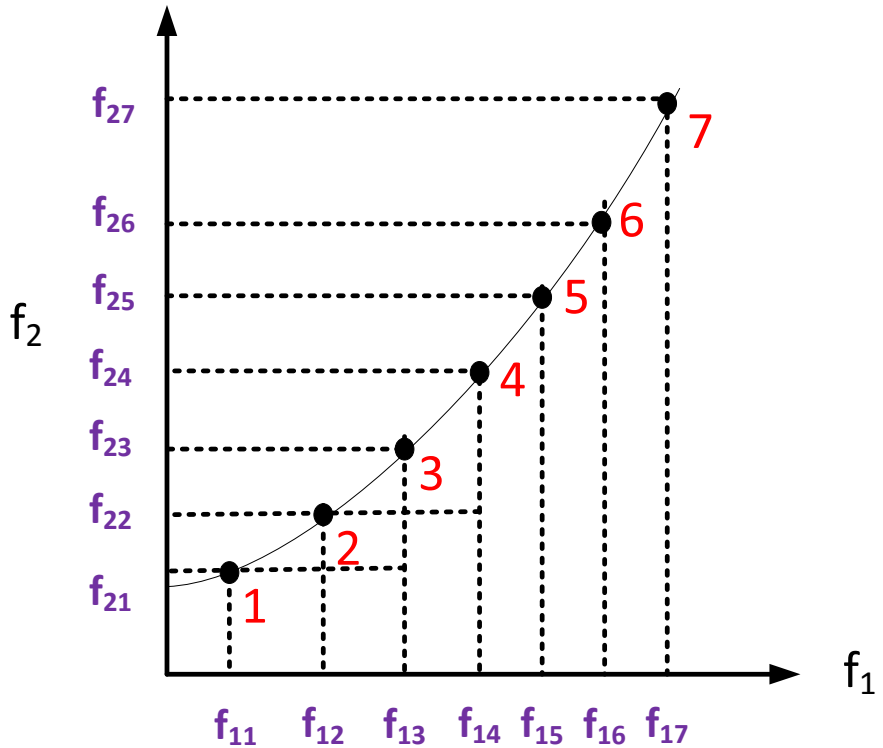
The sorted vectors are shown in the fig.

Crowding distance calculation



Crowding distance calculation

1. For each solution $j = 2$ to $l - 1$ do
$$d_j = \sum_{k=1}^{M} \frac{f_{k*j+1} - f_{k*j-1}}{f_k^{MAX} - f_k^{MIN}}$$
2. $d_i = d_l = \propto$(Each boundary solutions are not crowded and hence large crowded distance)



Crowding distance calculation

Note :

1. All objective function are assumed be minimized
2. Time complexity of the above operation is $O(MN \log_2 N)$, is the complexity of M sorting operation with N elements.
3. The parameters f_k^{MAX} and f_k^{MIN}, $K = 1, 2, \cdots M$ can be set as the population-maximum and population-minimum values of k-th objective function and is used here to normalize the objective values.

Crowding Tournament

- Once crowding distance of all solutions in the last acceptable front F is calculated we are to select $N - |P'|$ solutions from F to complete the size of $|P| = N$.
- To do this, we are to follow tournament selection operation according to $<_c$ (Crowding comparison operator). That is we select x_i over x_j ($x_i <_c x_j$) if $d_i > d_j$.
- Note that we prefer those solutions which are not in the crowded search space. This ensure a better population diversity.

Crowding Tournament

Remarks :

- NSGA-II is an elitist approach
- It consider a faster non-dominated sorting procedure with time complexity $O(MN^2)$ compared to $O(MN^3)$ in NSGA.
- It does not require explicit sharing concept rather use a crowding tournament selection with time complexity $O(MN \log N)$
- Thus, the overall time complexity of NSGA-II is $O(MN^2)$

Thank You!