NTNU

Norwegian University of Science and Technology

IT3708 Sub-symbolic AI Methods

Lecture 10 – Multi-Objective Evolutionary Algorithms (Part II)

Evolution Strategies

Genetic Programming

Kazi Shah Nawaz Ripon

ksripon@idi.ntnu.no

2016-03-15

Suggested Reading

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

NSGA- II: Quick Recap

- 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 $\boldsymbol{Q_t}$ is first created by using parent population $\boldsymbol{P_t}$.
  - Instead of finding the non-dominated front of $\boldsymbol{Q_t}$ only, the two populations are combined together to form $\boldsymbol{R_t}$ of size $\boldsymbol{2N}$.
  - Then a non-dominated sorting is used to classify the entire population $\boldsymbol{R_t}$.

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.
- 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

- Choose individuals having large crowding distance.

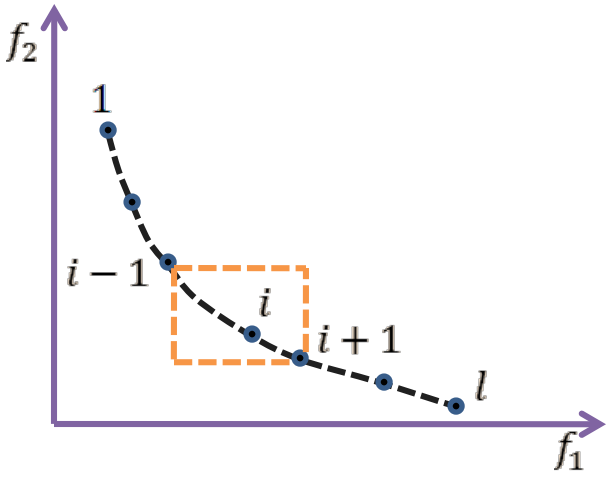
- Help for obtaining uniformly distribution.

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

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

$$where$$

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



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.

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	∞

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

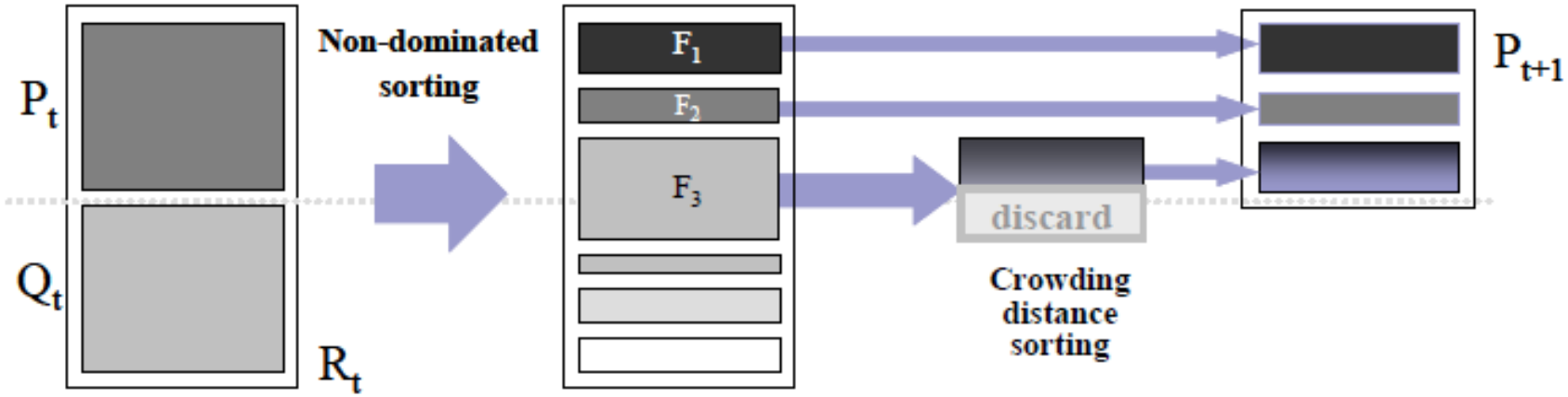
0.818	0.669	1.396	1	1.378

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

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

0.818	0.669	1.396	1	1.378

Selection for Next Generation



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.

NSGA II: Example *

- Parent and offspring populations used in this example:

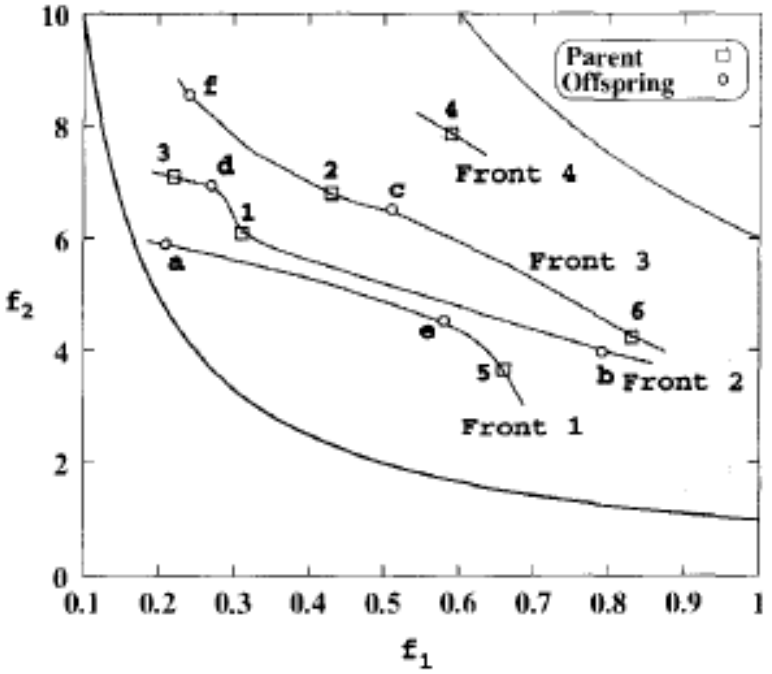
Parent population, P_t					Offspring population, Q_t				
Solution	x_1	x_2	f_1	f_2	Solution	x_1	x_2	f_1	f_2
1	0.31	0.89	0.31	6.10	a	0.21	0.24	0.21	5.90
2	0.43	1.92	0.43	6.79	b	0.79	2.14	0.79	3.97
3	0.22	0.56	0.22	7.09	c	0.51	2.32	0.51	6.51
4	0.59	3.63	0.59	7.85	d	0.27	0.87	0.27	6.93
5	0.66	1.41	0.66	3.65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

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



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. Note that if fronts 3 and 4 had not been classified earlier, we could have saved these computations.

NSGA II: Example

- 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.

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:

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

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

$$I^1 = \{3, d, 1, b\}.$$

	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

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:

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

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

	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

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:

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

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

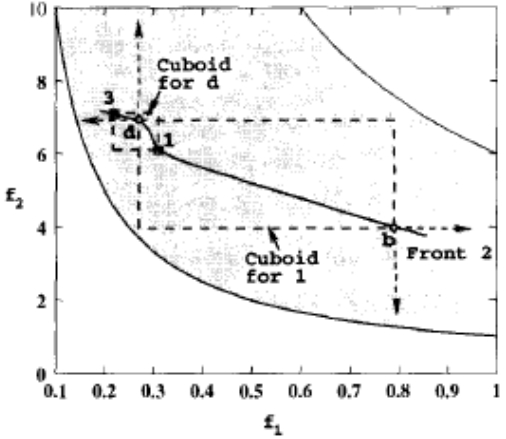
	Front 2				Sorting in		Distance
Solution	x_1	x_2	f_1	f_2	f_1	f_2	
1	0.31	0.89	0.31	6.10	third	second	0.63
3	0.22	0.56	0.22	7.09	first	fourth	∞
b	0.79	2.14	0.79	3.97	fourth	first	∞
d	0.27	0.87	0.27	6.93	second	third	0.12

Crowding Distance Assignment

The overall crowded distances of the four solutions are:

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

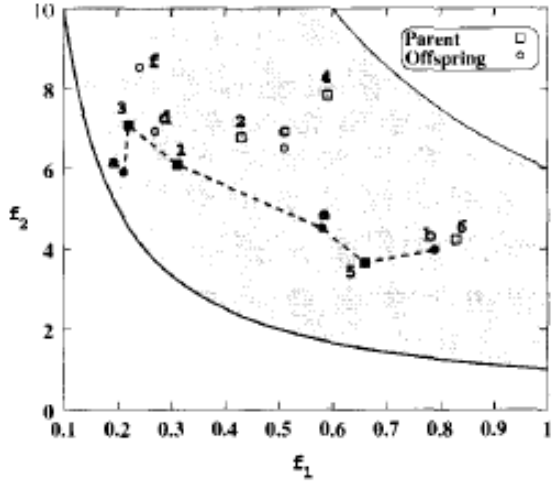
The cuboids (rectangles here) for these solutions are schematically shown in Figure 140. Solution d has the smallest perimeter of the hypercube around it than any other solution in the set $\mathcal{F}_2$, as evident from the figure. Now, we move to the main algorithm.



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 141 by joining them with dashed lines. It is important to note that this population is formed by choosing solutions from the better non-dominated fronts.



Niched-Pareto Genetic Algorithm (NPGA)

- Horn et al. (1994) proposed NPGA based on non-domination concept.
- It differs from the previous methods in **selection operator**.
  - Unlike proportionate selection method (VEGA, NSGA, MOGA), it uses **binary tournament selection**.
- During tournament selection, sharing function approach is applied.
- A dynamic updates niching strategy is used.

Tournament Selection (NPGA)

- During the tournament selection, two solutions i and j are picked at random from the parent population P.
- They are compared by first picking a sub-population T of size t_{dom} ($<< N$) solutions from the population.
- Thereafter, each solution i and j is compared with every solution of the sub-population for domination.
- Two scenarios.

Scenario : 1

- Of the two solutions (***i*** and ***j)***, ***i is chosen***, if
  - ***i*** dominates all solutions of the sub-population, **AND**
  - ***j*** is dominated by at least one solution from the sub-population.

Scenario: 2

- If both solutions (***i*** and ***j)*** are either
  - Dominated by at least one solution in the sub-population.

OR

  - Are not dominated by any solution in the sub-population.

- Both are checked with the current (partially filled) offspring population Q.
  - Each solution is placed in the offspring population and its **niche count** is calculated.
  - The solution with the smaller niche count wins the tournament.

Scenario: 2 (Exception)

In the beginning when the offspring population is empty:

- One of the two solutions is chosen as a parent at random.
- The same procedure is followed for another pair of solutions to choose the second parent.
- These two parents are then mated and mutated to create two offspring.
- From the third tournament onwards, the original procedure is adopted.

NPGA Procedure

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

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

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

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

Step 5 Update offspring population $Q = Q \cup \{c_1, c_2\}$

Step 6 Set $i = i + 1$. If $i < N$, go to Step 2.

Otherwise,

if $|Q| = N/2$, shuffle P, set $i = 1$, go to step 2.

Otherwise, the process is complete.

NPGA: Tournament Selection Procedure

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

Step T1 Pick a sub-population T_{ij} of size t_{dom} from the parent population P.

Step T2 Find α_i as the number of solutions in T_{ij} that dominates i. Calculate α_j as the the number of solutions in T_{ij} that dominates j.

Step T3 If $\alpha_i = 0$ and $\alpha_j > 0$, then i is the winner. The process is complete.

Step T4 Otherwise, if $\alpha_i > 0$ and $\alpha_j = 0$, then j is the winner. The process is complete.

25

Tournament Selection Procedure (contd.)

Step T5 Otherwise, if $|Q| < 2$,

- i or j is chosen as a winner with probability 0.5.
- The process is complete.

- Alternatively, the niche count nc_i and nc_j are calculated by placing i and j in the current offspring population Q, independently.
- With the niching parameter α_{share}, nc_i is calculated as the number of offspring ($k \in Q$) with in a α_{share} distance d_{ik} from i.
  - The distance d_{ik} is the Euclidean distance between solutions i and k in the objective space.

Step T6 If $nc_i <= nc_{i,}$ solution i is the winner. Otherwise, solution j is the winner.

26

NPGA : Advantages and Disadvantages

Advantages:

- ✓ No explicit fitness assignment is needed, while VEGA, NSGA, MOGA require it.
- ✓ The complexity of NPGA does not depend upon the number of objectives. Thus it may found to be computationally efficient in solving problems with many objectives.

Disadvantages:

- × It requires fixing two important parameters: α_{share} and t_{dom}.
- × The parameter α_{share} has more effect on an NPGA than an NSGA.
- × The performance of NPGA depends on the choice of t_{dom}. If t_{dom} is too small, the non-domination check will be noisy. If it is too large, the computational complexity of NPGA would be greater.

Strength Pareto Evolutionary Algorithm (SPEA)

- Proposed by Zitzler and Thiele (1998).
- Introduces elitism by explicitly maintaining an **external population $\overline{P}$**.
  - $\overline{P}$ stores a fixed number of the non-dominated solutions that are found until the beginning of a simulation.
- At every generation, newly found non-dominated solutions are compared with $\overline{P}$.
  - The resulting non-dominated solutions are preserved.
- **SPEA also uses these elites to participate in the genetic operations along with the current population**.

SPEA

- Begins with a randomly created population P_0 of size N and an empty external population $\overline{P_0}$ with a maximum capacity of $\overline{N}$.
- In any generation $\boldsymbol{t}$, the best non-dominated solutions (belonging to the first non-dominated front) of the population P_t are copied to the external population $\overline{P_t}$.
- Thereafter, the dominated solutions in the modified external population are found and deleted from the external population.
- The external population consists of the best non-dominated solutions of a combined population containing old and new elites.

SPEA: Size of $\overline{P}$

- In order to restrict the external population to over-grow, the size of the external population ($\overline{\boldsymbol{P}}$) is bounded to a limit ($\overline{N}$).
- When the size exceeds $\overline{N}$, only that many solutions which are needed to maintain the fixed population size $\overline{N}$ are retained.
  - Elites which are less crowded in the non-dominated front are kept.
  - This is where SPEA maintains ***diversity***.
- The investigators suggested ***a clustering method*** to achieve this task.

SPEA: Fitness Assignment

- SPEA uses the external population $\overline{P_t}$ in its genetic operations.
- In addition to the assigning of fitness to the current population members, fitness is also assigned to external population members.
- SPEA assigns a fitness (called the ***strength***) S_i to each member i of the external population first.
- Strength S_i is proportional to the number (n_i) of current population members that an external solution i dominates:

$$S_i = \frac{n_i}{N+1}$$

- This equation assigns more strength to an elite which dominates more solutions in the current population.

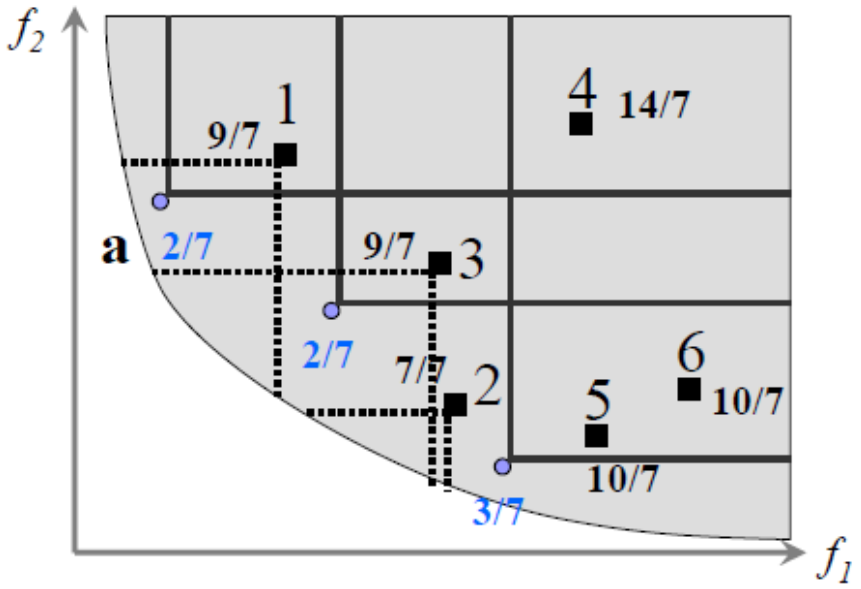
SPEA: Fitness Assignment – Current Population

- Fitness of a current population member j is assigned as one more than the sum of S_i of all external population members which weakly dominate j.

$$F_i = 1 + \sum_{i \in \overline{P_t} \wedge i \leq j} S_j$$

- The addition of one makes the fitness of any current population member P_t to be more than the fitness of any external population member $\overline{P_t}$.
- **A solution with a smaller fitness is better**.

Example: Two-Objective Minimization



N= 6, N̲=3

■ P

○ P̲

$$S_a = \frac{\#\text{ of current population members dominated by } \mathbf{a}}{N+1} = \frac{2}{7}$$

$$F_1 = 1 + \sum_{i \in D_j} S_i$$

$$= 1 + \frac{2}{7} = \frac{9}{7}$$

Example: Two-Objective Minimization

- External population members ($\overline{P_t}$) get smaller fitness values than the EA population members (P_t) .
- EA population members (P_t) dominated by many external members ($\overline{P_t}$) get large fitness values.
- A binary tournament selection procedure is applied to the combined ($\overline{P_t}$ U P_t) population to choose **solutions with smaller fitness values**.
- It is likely that external elites will be emphasized during this tournament procedure.

SPEA: Algorithm

Step 1 Find the best non-dominated set $\mathcal{F}_1(P_t)$ of P_t. Copy these solutions to $\overline{P}_t$, or perform $\overline{P}_t = \overline{P}_t \cup \mathcal{F}_1(P_t)$.

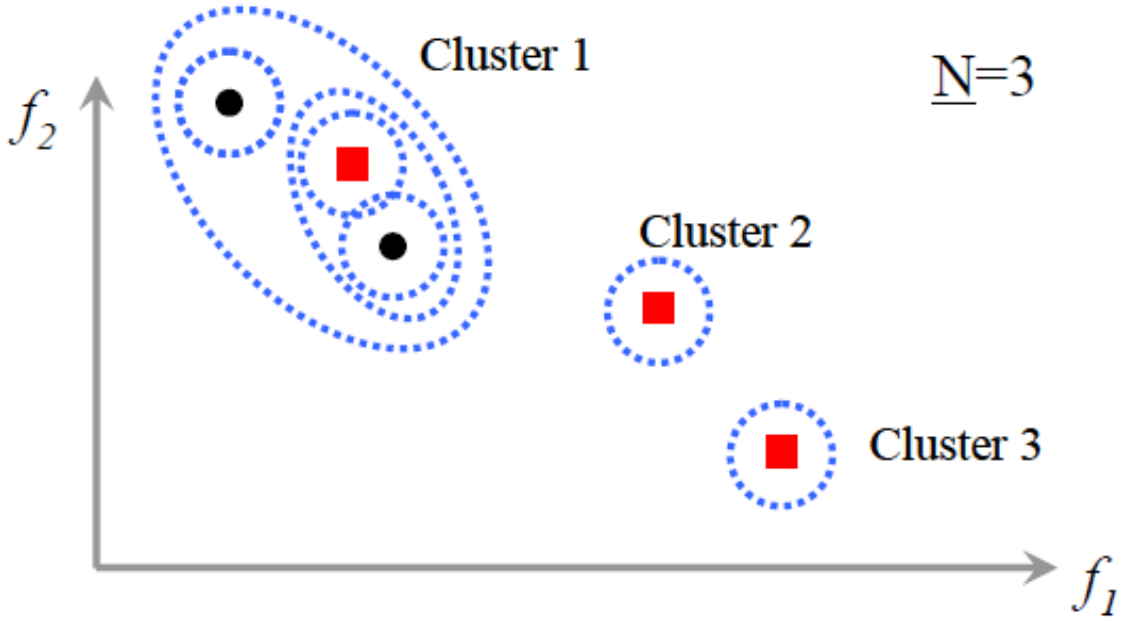
Step 2 Find the best non-dominated solutions $\mathcal{F}_1(\overline{P}_t)$ of the modified population $\overline{P}_t$ and delete all dominated solutions, or perform $\overline{P}_t = \mathcal{F}_1(\overline{P}_t)$.

Step 3 If $|\overline{P}_t| > \overline{N}$, use a clustering technique to reduce the size to $\overline{N}$. Otherwise, keep $\overline{P}_t$ unchanged. The resulting population is the external population $\overline{P}_{t+1}$ of the next generation.

Step 4 Assign fitness to each elite solution $i \in \overline{P}_{t+1}$ by using equation (6.4). Then, assign fitness to each population member $j \in P_t$ by using equation (6.5).

Step 5 Apply a binary tournament selection with these fitness values (in a minimization sense), a crossover and a mutation operator to create the new population P_{t+1} of size N from the combined population $(\overline{P}_{t+1} \cup P_t)$ of size $(\overline{N} + N)$.

SPEA: Clustering Algorithm



SPEA: Clustering Algorithm

Step C1 Initially, each solution belongs to a distinct cluster or $C_i = \{i\}$, so that $\mathcal{C} = \{C_1, C_2, \ldots, C_{\overline{N}}\}$.

Step C2 If $|\mathcal{C}| \leq \overline{N}$, go to Step C5. Otherwise, go to Step C3.

Step C3 For each pair of clusters (there are $\binom{|\mathcal{C}|}{2}$ of them), calculate the cluster-distance by using equation (6.6). Find the pair (i_1, i_2) which corresponds to the minimum cluster-distance.

$$d_{12} = \frac{1}{|C_1||C_2|} \sum_{i \in C_1, j \in C_2} d(i,j)$$

Step C4 Merge the two clusters C_{i_1} and C_{i_2} together. This reduces the size of $\mathcal{C}$ by one. Go to Step C2.

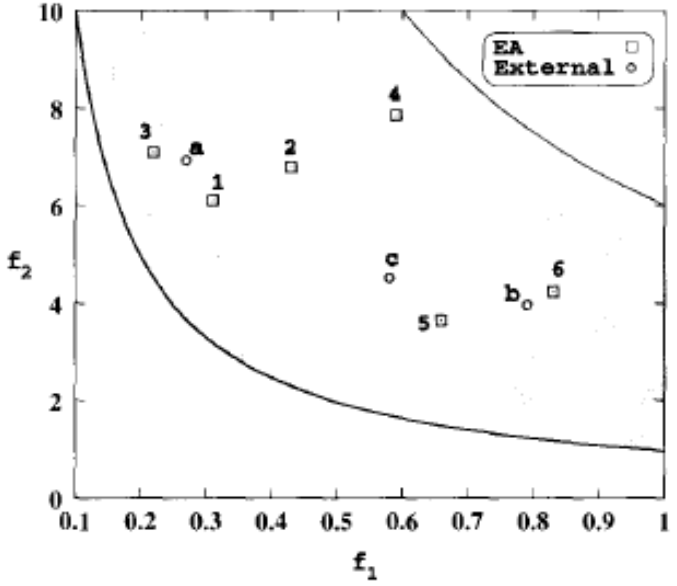
Step C5 Choose only one solution from each cluster and remove the others from the clusters. The solution having the minimum average distance from other solutions in the cluster can be chosen as the representative solution of a cluster.

SPEA: Example *

- Two-objective (minimization) problem.

EA population P_t				
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

External population $\overline{P}_t$				
Solution	x_1	x_2	f_1	f_2
a	0.27	0.87	0.27	6.93
b	0.79	2.14	0.79	3.97
c	0.58	1.62	0.58	4.52



* Example taken from K. Deb, 2001.

SPEA: Example

Step 1:

- First, we find the non-dominated solutions of P_t.
- From the figure: $\mathcal{F}_1(P_t) = \{1, 3, 5\}$
- Include these solutions to in $\overline{P_t}$
- Thus: $\overline{P_t} = \{a, b, c, 1, 3, 5\}$

SPEA: Example

Step 2 We now calculate the non-dominated solutions of this modified population $\overline{P}_t$ and observe $\mathcal{F}_1(\overline{P}_t) = \{a, c, 1, 3, 5\}$. We set this population as the new external population.

Step 3 Since the size of $\overline{P}_t$ is 5, which is greater than the external population size ($\overline{N} = 3$), we need to use the clustering algorithm to find which three will remain in the external population.

SPEA: Clustering Example

Step C1 Initially, all five solutions belong to a separate cluster:

$$C_1 = \{a\}, \quad C_2 = \{c\}, \quad C_3 = \{1\}, \quad C_4 = \{3\}, \quad C_5 = \{5\}.$$

Step C2 Since there are 5 clusters, we move to Step C3.

Step C3 We use $f_1^{max} = 1$, $f_1^{min} = 0.1$, $f_2^{max} = 60$ and $f_2^{min} = 1$. All $\binom{5}{2}$ or 10 cluster-distances are as follows:

$$d_{12} = 0.35, \quad d_{13} = 0.05, \quad d_{14} = 0.06, \quad d_{15} = 0.44, \quad d_{23} = 0.30,$$
$$d_{24} = 0.40, \quad d_{25} = 0.09, \quad d_{34} = 0.09, \quad d_{35} = 0.39, \quad d_{45} = 0.49.$$

We observe that the minimum cluster-distance occurs between the first and the third clusters.

Step C4 We merge these clusters together and have only four clusters:

$$C_1 = \{a, 1\}, \quad C_2 = \{c\}, \quad C_3 = \{3\}, \quad C_4 = \{5\}.$$

SPEA: Clustering Example

Step C2 Since there are four clusters, we move to Step C3 to reduce one more cluster.

Step C3 Now, the average distance between the first and second cluster is the average distance between the two pairs of solutions (a,c) and (1,c). The distance between solutions a and c is 0.35 and that between solutions 1 and c is 0.30. Thus, the average distance d_{12} is 0.325. Similarly, we can find other distances of all $\binom{4}{2}$ or 6 pairs of clusters:

$$d_{12} = 0.325, \quad d_{13} = 0.075, \quad d_{14} = 0.415,$$
$$d_{23} = 0.400, \quad d_{24} = 0.090, \quad d_{34} = 0.490.$$

The minimum distance occurs between clusters 1 and 3.

Step C4 Thus, we merge clusters 1 and 3 and have the following three clusters:

$$C_1 = \{a, 1, 3\}, \quad C_2 = \{c\}, \quad C_3 = \{5\}.$$

Step C2 Since we now have an adequate number of clusters, we move to Step C5.

SPEA: Example

Step C5 In this step, we choose only one solution in every cluster. Since the second and third clusters have one solution each, we accept them as they are. However, we need to choose only one solution for cluster 1. The first step is to find the centroid of the solutions belonging to the cluster. We observe that the centroid of solutions a, 1 and 3 is $c_1 = (0.27, 6.71)$. Now the normalized distance of each solution from this centroid is as follows:

$$d(a, c_1) = 0.005, \quad d(1, c_1) = 0.049, \quad d(3, c_1) = 0.052.$$

Solution $\boldsymbol{a}$ is the closest to the centroid $\boldsymbol{c_1}$.

We choose solution $\boldsymbol{a}$ and delete solutions $\boldsymbol{1}$ and $\boldsymbol{3}$ from this cluster.

New external population is $\overline{P_{t+1}} = \{a, c, 5\}$

SPEA: Example

Step 4 Now, we assign fitness values to the solutions of populations P_t and $\overline{P}_{t+1}$. Note that solution 5 is a member of both P_t and $\overline{P}_{t+1}$ and is treated as two different solutions. First, we concentrate on the external population. We observe that solution a dominates only one solution (solution 4) in P_t, Thus, its fitness (or strength) is assigned as:

$$F_a = \frac{n_a}{N+1} = \frac{1}{6+1} = 0.143.$$

Similarly, we find $n_c = n_5 = 1$, and their fitness values are also $F_c = F_5 = 0.143$.

Next, we calculate the fitness values of the solutions of P_t. Solution 1 is dominated by no solution in the external population. Thus, its fitness is $F_1 = 1.0$. Similarly, solutions 2 and 3 are not dominated by any external population members and hence their fitness values are also 1.0. However, solution 4 is dominated by two external members (solutions a and c) and hence its fitness is $f_4 = 1 + 0.143 + 0.143 = 1.286$. We also find that $F_5 = 1.0$ and $F_6 = 1.143$. These

SPEA: Example

EA population P_t		External population P_{t+1}	
Solution	Fitness	Solution	Fitness
1	1.000	a	0.143
2	1.000	c	0.143
3	1.000	5	0.143
4	1.286		
5	1.000		
6	1.143		

SPEA: Example

Step 5 Now, using the above fitness values we would perform six tournaments by randomly picking solutions from the combined population of size nine (in effect, there are only eight distinct solutions) and form the mating pool. Thereafter, crossover and mutation operators will be applied to the mating pool to create the new population P_{t+1} of size six.

SPEA : Advantages and Disadvantages

Advantages:

- ✓ Once a solution Pareto-optimal front is found, it is stored in the external population.
- ✓ Clustering ensures a spread among the non-dominated solutions.
- ✓ The proposed clustering algorithm is parameter less.

Disadvantages:

- × A balance between the regular population size N and the external population size $\overline{N}$ is important
  - × If $\overline{N}$ is too large, selection pressure for the elites is too large and the algorithm may not converge to the Pareto-optimal front.
  - × If $\overline{N}$ is too small, the effect of elitism will be lost.

Pareto-Archived Evolution Strategy (PAES)

- Proposed by Knowles and Corne (2000).
- PAES is a **(1+1) Evolution Strategy (ES)** employing local search.
  - But using a reference archive of previously found solutions.
- (1+1)-ES uses only mutation on a single parent to create a single offspring → **a local search strategy**.
  - PAES is capable of generating diverse solutions in the Pareto-optimal set.

Evolution Strategies

Evolution Strategies (ES)

Quick Overview

- Developed: Germany in the 1970's.
- Early names: I. Rechenberg, H.-P. Schwefel.
- Typically applied to:
  - Numerical optimization.
  - ***Continuous parameter optimization.***
- Attributed features:
  - Fast.
  - good optimizer for real-valued optimization.
  - relatively much theory.
- Special:
  - ***Self-adaptation*** of (mutation) parameters standard.

Quick Overview (cont'd)

- **Early Evolutionary Strategy:**
  - Only uses mutation and does not use crossover.
  - Normally applied to real numbers (continuous variables) rather than discrete values.
  - Only a single solution was maintained and this was improved upon.

- **Later developments:**
  - Uses both mutation and crossover.
  - Also applies to discrete variables.
  - ES's are a population based approach.

ES - How They Work

- An individual in an ES is represented as a pair of real vectors:

$$\boldsymbol{v} = \langle x_1,\ldots,x_n, \sigma_1,\ldots,\sigma_n \rangle$$

- $\boldsymbol{x_1,\ldots,x_n}$ represent points in the search space and consists of a number of real valued variables

- The second vector, $\boldsymbol{\sigma_1,\ldots,\sigma_n}$ represents Mutation step sizes.

ES - How They Work

- Mutation is performed by replacing $\boldsymbol{x}$ by

$$\boldsymbol{x}^{t+1} = \boldsymbol{x}^{t} + N(0, \boldsymbol{\sigma})$$

  $N(0, \boldsymbol{\sigma})$ is a random Gaussian number with a mean of zero and standard deviations of $\boldsymbol{\sigma}$.

- Key idea:
  - σ is part of the chromosome $\langle x_1,\ldots,x_n, \sigma \rangle$
  - σ is also mutated into σ'.
- Thus, mutation step size σ is coevolving with the solution x.

ES - How They Work

- In the earliest ES's (where only a single solution was maintained), the new individual replaced its parent if it had a higher fitness.
- In addition, these early ES's, maintained the same value for σ throughout the duration of the algorithm.
  - It has been proven that if this vector remains constant throughout the run then the algorithm will converge to the optimal solution.

ES's - How They Work

- **Problem**
  - Although the global optimum can be proved to be found with a probability of one, it also states that the theorem holds for **sufficiently** long search time.
  - The theorem tells us nothing about how long that search time might be.

- To try and speed up convergence Rechenberg has proposed the ". **1/5 success rule**".

ES's - How They Work

- **Motivation behind 1/5 rule**
  - If we are finding lots of successful moves then we should try larger steps in order to try and improve the efficiency of the search
  - If we are not finding many successful moves then we should proceed in smaller steps

1/5 Success Rule

- This rule resets σ after every k iterations by

$$\sigma = \sigma / c \quad \text{if } p_s > 1/5$$

$$\sigma = \sigma \bullet c \quad \text{if } p_s < 1/5$$

$$\sigma = \sigma \quad \text{if } p_s = 1/5$$

where p_s = % of successful mutations

$$= \frac{\text{\# of Successful munations}}{\text{total \# of mutations}}$$

Suggested value, $0.817 \leq c \leq 1$

ES's - Recombination

- Creates one child.

- Acts per variable / position by either
  - Averaging parental values (Intermediate)
  - Selecting one of the parental values (Discrete)

- From two or more parents by either:
  - Using two selected parents to make a child (Local)
  - Selecting two parents for each position a new (Global)

Recombination: Intermediate

- Often used to adapt the strategy parameters.
- Each child parameter is the mean value of the corresponding parent parameters.

(8, 12, 31, … ,5) (2, 5, 23, … , 14)

(5, 8.5, 27, … , 9.5)

Recombination: Discrete

- Similar to crossover of genetic algorithms.
- Equal probability of receiving each parameter from each parent.

(8, 12, 31, … ,5) (2, 5, 23, … , 14)

(2, 12, 31, … , 14)

ES's - Names of Recombinations

	Two fixed parents	Two parents selected for each i
$z_i = (x_i + y_i)/2$	Local intermediary	Global intermediary
z_i is x_i or y_i chosen randomly	Local discrete	Global discrete

ES's - Survivor Selection

- In ES's there are two variations as to how we create the new generation.
- Deterministically chops off the "bad stuff".
- Applied after creating λ children from the μ parents by mutation and recombination.
- Basis of selection is either:
  - $(\mu + \lambda)$: The set of parents and children.
  - (μ , λ) : The set of children only.

Survivor Selection: ($\mu + \lambda$)

- ($\mu + \lambda$), uses μ parents and creates λ offspring.
- After mutation, there will be $\mu + \lambda$ members in the population.
- All these solutions compete for survival, with the μ best selected as parents for the next generation.
- **Elitist strategy.**

Survivor Selection: (μ , λ)

- (μ, λ), works by the μ parents producing λ offspring (where λ > μ).
- Only the λ compete for survival.
  - Thus, the parents are completely replaced at each new generation.
- A single solution only has a life span of a single generation.

ES’s - Survivor Selection

- The original work on ES’s (Schwefel, 1965) used a $(1 + 1)$ strategy.
- This took a single parent and produced a single offspring.
- Both these solutions competed to survive to the next generation.

ES's vs GA's

- **In summary ES's are**
  - Like Gas, but only use mutation and not crossover.
  - They operate on real numbers.
  - They are not a population based approach.
  - **But we can break any, or all, of these rules if we wish!**

Back to PAES

Quick Recap: PAES

- Proposed by Knowles and Corne (2000).
- PAES is a **(1+1) Evolution Strategy (ES)** employing local search.
  - But using **a reference archive** of previously found solutions.
- (1+1)-ES uses only mutation on a single parent to create a single offspring → **a local search strategy**.
  - PAES is capable of generating diverse solutions in the Pareto-optimal set.

PAES: How to Choose The Winner

- At any generation t, along with the parent (p_t) and the offspring (c_t), there is an archive of the best solutions found so far.
- Initially, the archive is empty.
- As the generations proceed, good solutions are added to the archive and updated.
- To maintain a maximum size of the archive, p_t and c_t are compared for domination.

PAES: Working Procedure

- PAES is composed of three parts:
    1. The candidate solution generator.
    2. The candidate solution acceptance function.
    3. The non-dominated solutions (NDS) archive.

Candidate Solution Generator

- At first, a random solution x_0 (parent) is chosen.
- x_0 is then mutated by using a normally distributed probability function with zero-mean and with a fixed mutation strength $\rightarrow$ c_0 (the mutated offspring).

Candidate Solution Acceptance Function

- The parent (x_0) and the offspring (c_0) are compared.
  - The winner becomes the parent of the next generation.
- The most important issue of the PAES is **the way that a winner is chosen in the midst of multiple objectives**.
- This will result in three scenarios.

Scenario - 1

p_t dominates c_t:

- The offspring c_t is not accepted.
- A new mutated solution is created for further processing.

Scenario - 2

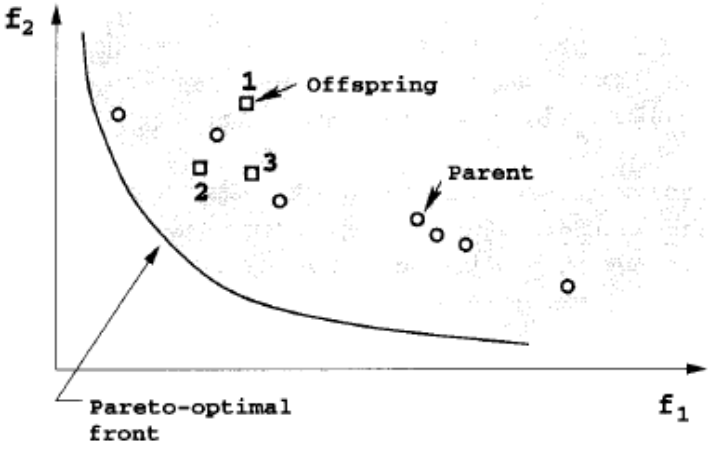
c_t dominates p_t:

- Solution c_t is accepted as a parent of the next generation.
- A copy of c_t is kept in the archive.
- This is how the archive gets populated by non-dominated solutions.

Scenario - 3

p_t and c_t are non-dominated:

- c_t is compared with the current archive (contains the set of non-dominated solutions found so far).
- Again, **three cases are possible here**.



Archive Update: First Case

c_t is dominated by one / more member(s) of the archive:

- c_t is not worth including in the archive.
- c_t is then rejected, and
- Parent p_t, is mutated again to find a new offspring for further processing.

Archive Update: 2nd Case

c_t dominates one / more member(s) of the archive:

- c_t accepted without any condition.
- c_t then becomes the parent of the next generation.
- **Dominated member(s) of the archive are deleted**.
- This process does not increase the size of the archive.

Archive Update: 3rd Case

c_t is not dominated by any member of the archive and it also does not dominate any member of the archive:

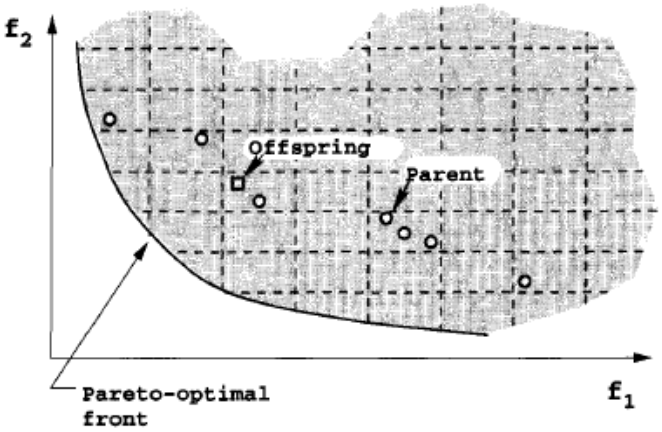
- c_t belongs to that non-dominated front in which the archive solutions belong.
- **c_t is added to the archive only if there is a slot available in the archive.**
- If the archive is full, the density-based comparison is performed between p_t and c_t to decide who remains in the archive.
  - The one residing in the least crowded area in the search space remains in the archive.

Archive Update: 3rd Case (contd)

- As for the offspring $\boldsymbol{c_t}$, the current parent $\boldsymbol{p_t}$, is also a member of the archive.
  - The acceptance of $\boldsymbol{c_t}$ in the achieve does not qualify it to become the parent of the next generation.
- To decide who qualifies as a parent of the next generation, the density of solutions in their neighborhood is checked.
- The one residing in the least crowded area in the search space qualifies as the parent.

Density Calculation (Archive is not Full)

- Each objective is divided into 2^d equal divisions.
  - d is a user-defined depth parameter.
- The entire search space is divided into $\left(2^d\right)^M$ unique, equal-sized M-dimensional hypercubes.
- The archived solutions are placed in these hypercubes according to their locations in the objective space.
- Thereafter, the number of solutions in each hypercube is counted.
- If the offspring resides in a less crowded hypercube than the parent
  - The offspring becomes the parent of the next generation.
- Otherwise, the parent solution continues to be the parent of the next generation.



Density Calculation (Archive is Full)

- Offspring cannot be included automatically.
- The hypercube with the highest number of solutions is identified.
- If the offspring does not belong to that hypercube, it is included in the archive, and
  - At random one of the solutions from the highest-count hypercube is eliminated.
- Whether the offspring or the parent qualifies to be the parent of the next generation is decided by same parent-offspring density count mentioned earlier.

Archive and Parent Update in PAES

Step 1 If c_t is dominated by any member of A_t, set $p_{t+1} = p_t$ (A_t is not updated). Process is complete. Otherwise, if c_t dominates a set of members from A_t: $D(c_t) = \{i : i \in A_t \wedge c_t \preceq i\}$, perform the following steps:

$$\begin{aligned} A_t &= A_t \backslash D(c_t), \\ A_t &= A_t \cup \{c_t\}, \\ p_{t+1} &= c_t. \end{aligned}$$

Process is complete. Otherwise, go to Step 2.

Archive and Parent Update in PAES

Step 2 Count the number of archived solutions in each hypercube. The parent p_t belongs to a hypercube having n_p solutions, while the offspring belongs to a hypercube having n_c solutions. The highest-count hypercube contains the maximum number of archived solutions.

If $|A_t| < N$, include the offspring in the archive, or $A_t = A_t \cup \{c_t\}$ and $p_{t+1} = \text{Winner}(c_t, p_t)$, and return. Otherwise (that is, if $|A_t| = N$), check if c_t belongs to the highest-count hypercube. If yes, reject c_t, set $p_{t+1} = p_t$, and return. Otherwise, replace a random solution r from the highest-count hypercube with c_t:

$$\begin{aligned} A_t &= A_t \backslash \{r\}, \\ A_t &= A_t \cup \{c_t\}, \\ p_{t+1} &= \text{Winner}(c_t, p_t). \end{aligned}$$

The process is complete.

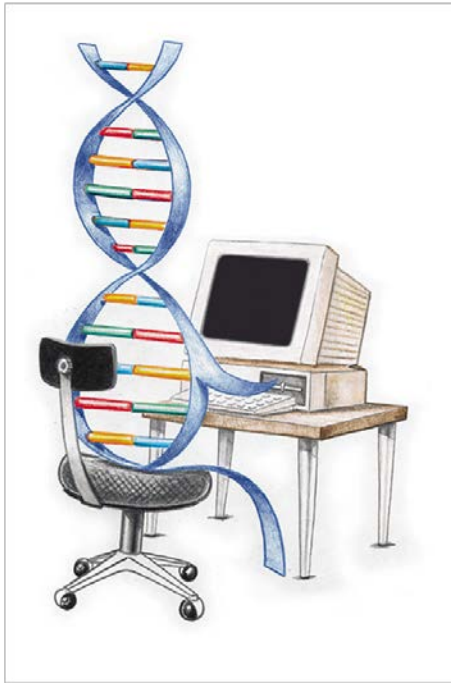
PAES : Advantages and Disadvantages

Advantages:

✓ PAES has a direct control on the diversity that can be achieved in the Pareto-optimal solutions by using the appropriate size of the depth size d.

Disadvantages:

× In addition to choosing an appropriate archive size (N), the depth parameter (d) is also an important parameter.



Genetic Programming

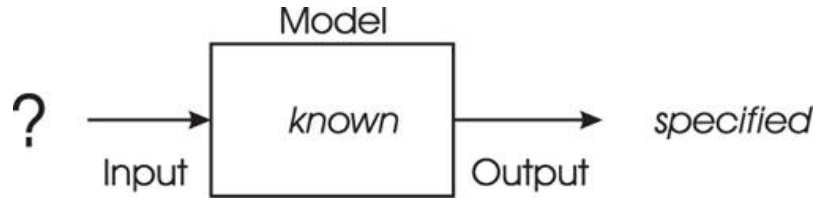
Genetic Programming (GP)

- One of the central problems in computer science is how to make computers solve problems without being explicitly programmed to do so.
- GP offers a solution through the evolution of computer programs by methods of natural selection.
- According to Koza, GP searches the space of possible computer programs for a program that is highly fit for solving the problem at hand.

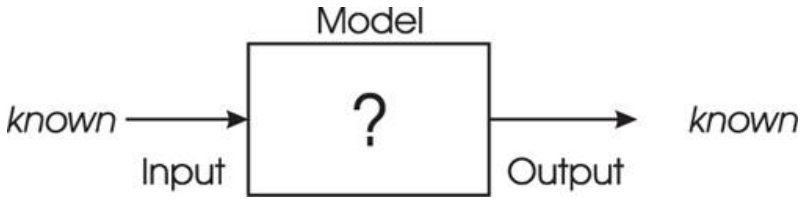
GP Quick Overview

- Developed by J. Koza in the 1990's.
- Relatively young member of the EA family.
- Differs from other EA strands:
  - In its application area.
  - Particular representation (**using trees as chromosomes**).
- EAs are typically applied to optimization problems, GP could instead be positioned in **machine learning**.
- Special:
  - non-linear chromosomes: trees, graphs.
  - **mutation possible but not necessary (disputed!)**.

GP in Machine Learning



Most EAs are for finding some input realising maximum payoff.



GP seeks models with maximum fit.

Models

↓

Represented as parse trees

↓

Treated as individual

↓

Their fitness is the model quality to be optimized.

Historical GP Perspective

- Specialized form of GA.
- Original application was to design computer program.
  - Now applied in alternative areas eg. Analog Circuits.
- **Does not make distinction between search and solution space.**
- It saves time by **freeing the human** from having to design complex algorithms.
  - Not only designing the algorithms but creating ones that give **optimal** solutions.

Introductory Example: Credit Scoring

- Bank wants to distinguish good from bad loan applicants.
- Model needed that matches historical data.

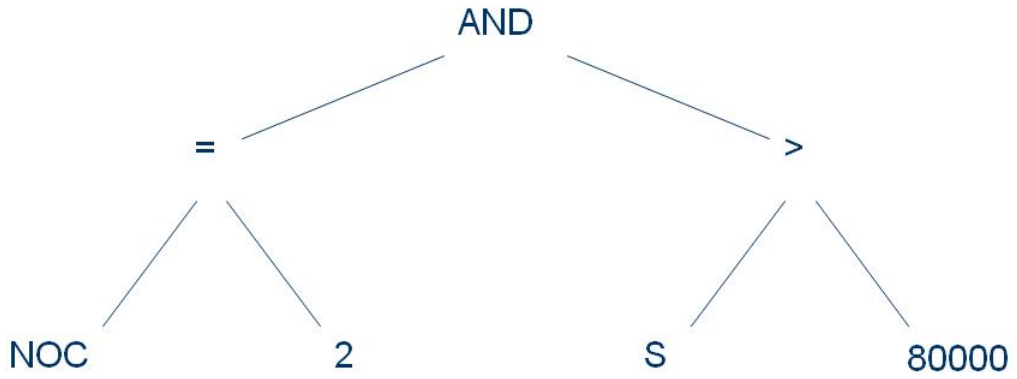
ID	No of children	Salary	Marital status	OK?
ID-1	2	45000	Married	0
ID-2	0	30000	Single	1
ID-3	1	40000	Divorced	1
…				

Introductory Example: Credit Scoring

- A possible model:
  - IF **(NOC = 2) AND (S > 80000)** THEN **good** ELSE **bad**
- In general:
  - IF **formula** THEN **good** ELSE **bad**
- Only unknown is the right formula, hence
  - Our search space (phenotypes) is the set of formulas
- Natural representation of formulas (genotypes) is: **parse trees**.

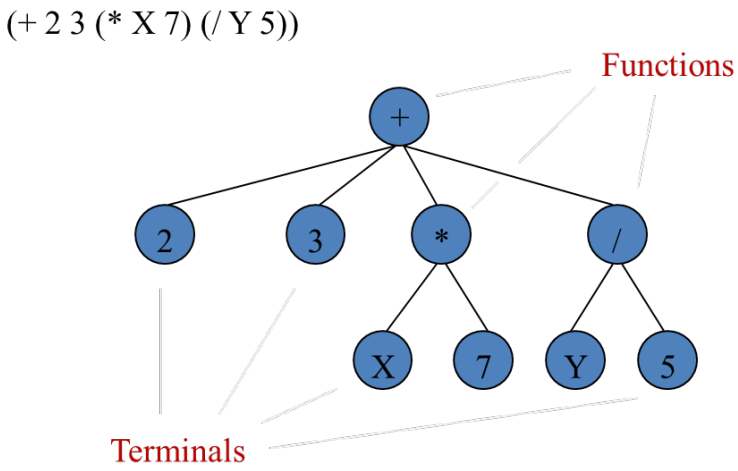
Introductory Example: Credit Scoring

IF (NOC = 2) AND (S > 80000) THEN good ELSE bad can be represented by the following tree



Randomly Generating Programs

- Randomly generate a program that takes two arguments and uses basic arithmetic to return an answer.
  - Function set = {+, -, *, /}
  - Terminal set = {integers, X, Y}
- Randomly select either a function or a terminal to represent our program.
- If a function was selected, recursively generate random programs to act as arguments.

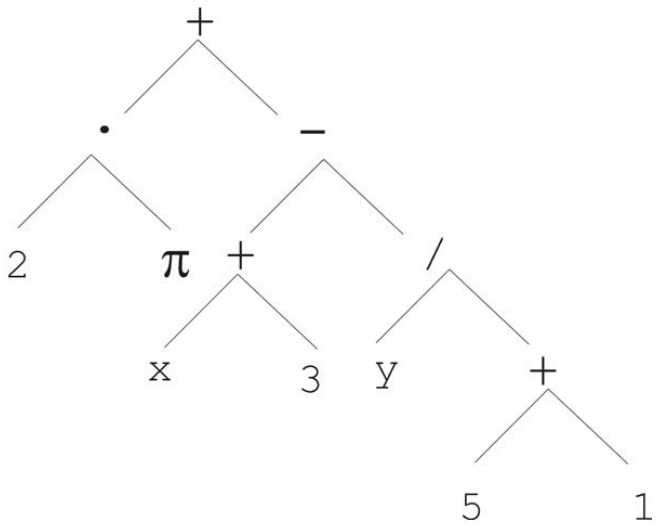


Tree based Representation

- The parse trees used by GP as chromosomes capture expressions in a given formal syntax.
- Depending on the problem at hand, and the users' perceptions on what the solutions must look like, this can be
  - syntax of arithmetic expressions $2 \cdot \pi + \left((x+3) - \frac{y}{5+1} \right)$
  - Logical formula $(x \wedge true) \rightarrow ((x \vee y) \vee (z \leftrightarrow (x \wedge y)))$
  - Program

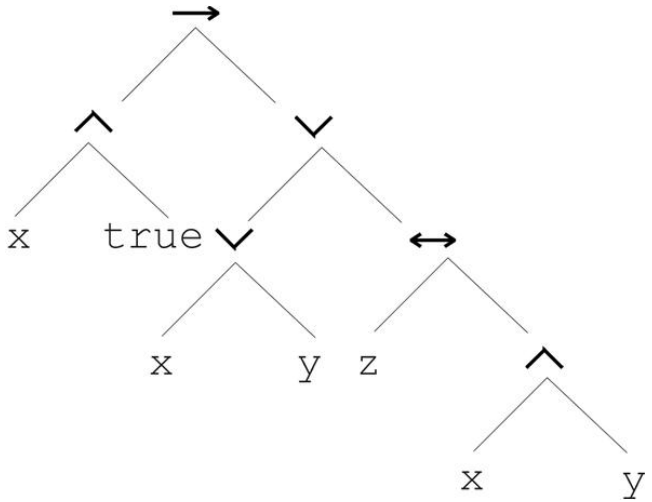
```
i =1;
while (i < 20)
{
        i = i +1
}
```

Tree based Representation



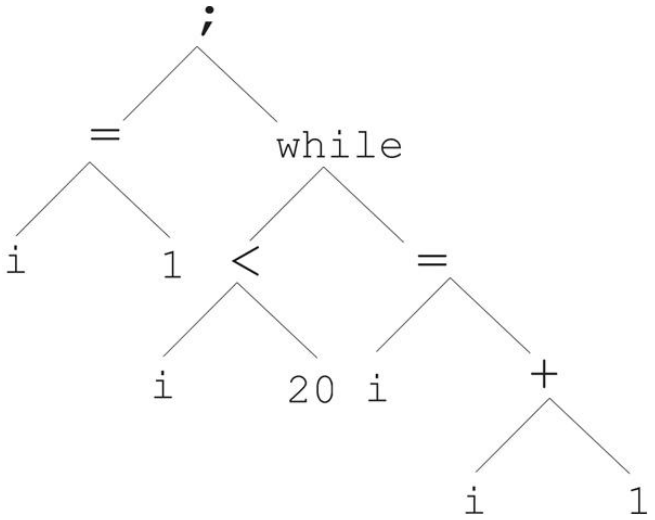
$$2 \cdot \pi + \left((x+3) - \frac{y}{5+1} \right)$$

Tree based Representation



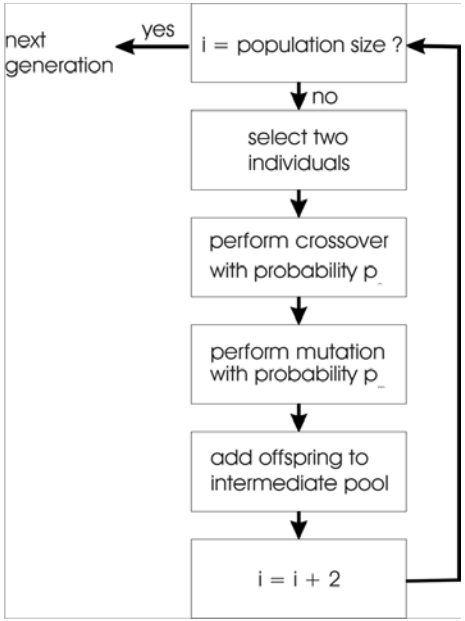
$$(x \wedge true) \rightarrow ((x \vee y) \vee (z \leftrightarrow (x \wedge y)))$$

Tree based Representation

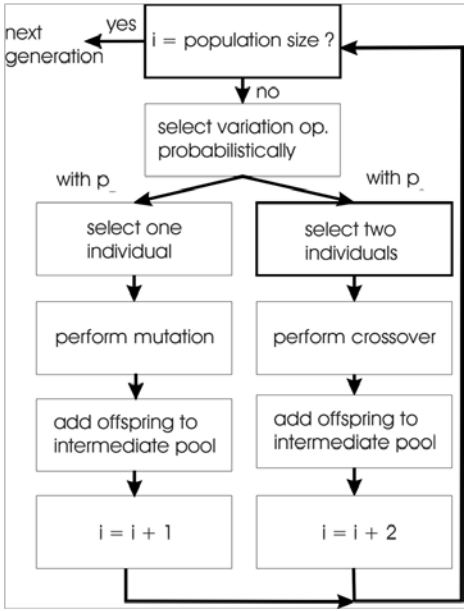



```
i = 1;
while (i < 20)
{
        i = i + 1
}
```

Offspring Creation Scheme



GA flowchart



GP flowchart

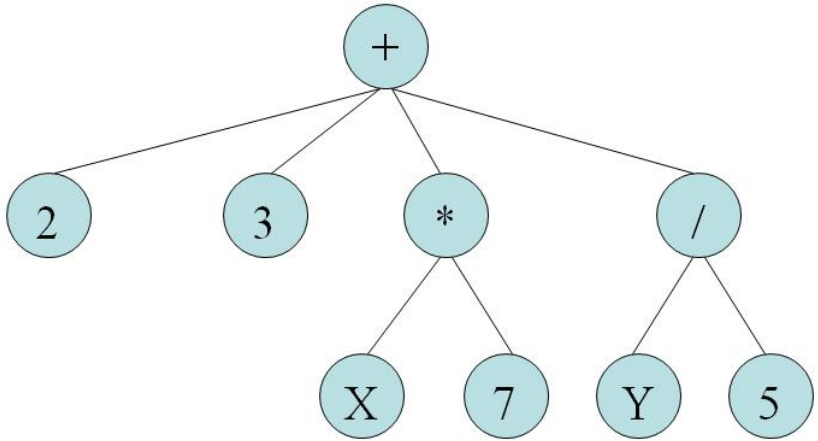
Koza advises users to set the mutation rate at 0 !

Mutation

- Most common mutation: replace randomly chosen subtree by randomly generated tree.

Mutation

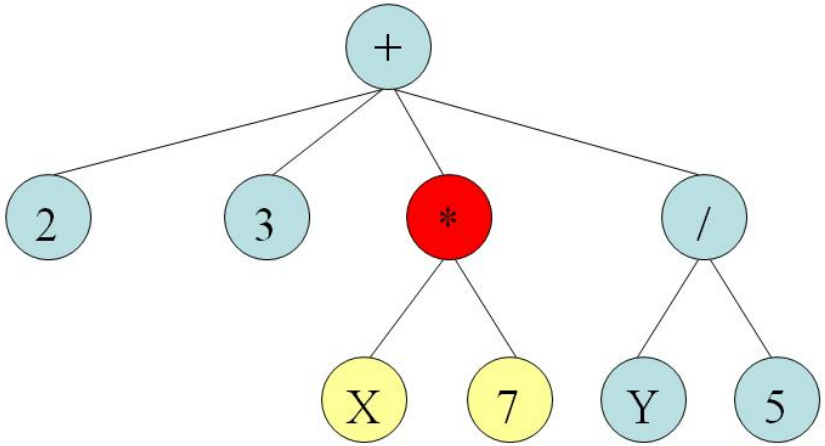
(+ 2 3 (* X 7) (/ Y 5))



Mutation

(+ 2 3 (* X 7) (/ Y 5))

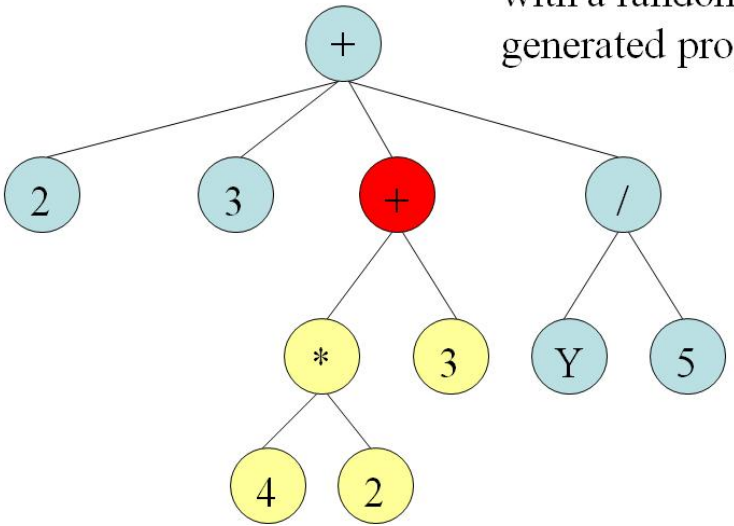
First pick a random node



Mutation

(+ 2 3 (+ (* 4 2) 3) (/ Y 5))

Delete the node and its children, and replace with a randomly generated program



Mutation

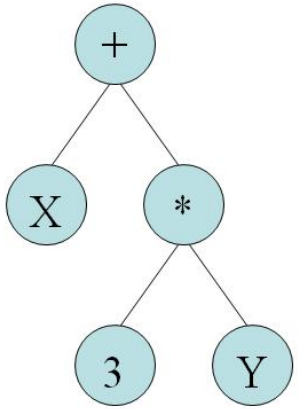
- Mutation has two parameters:
  - Probability p_m to choose mutation vs. recombination.
  - Probability to chose an internal point as the root of the subtree to be replace.
- The size of the child can **exceed** the size of the parent.

Recombination

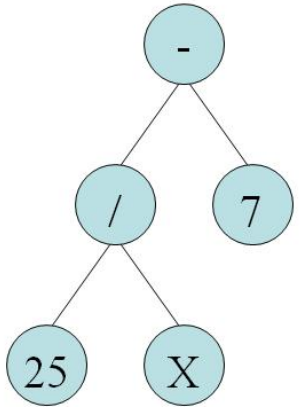
- Most common recombination: exchange two randomly chosen subtrees among the parents.
- Recombination has two parameters:
  - Probability p_c to choose recombination vs. mutation.
  - Probability to chose an internal point within each parent as crossover point.
- The size of offspring can **exceed** that of the parents.

Recombination

(+ X (* 3 Y))

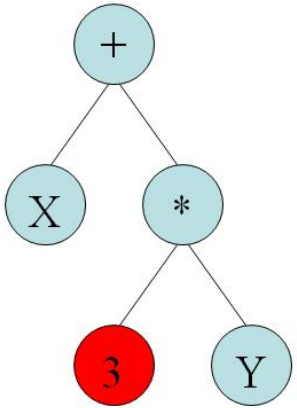


(- (/ 25 X) 7)



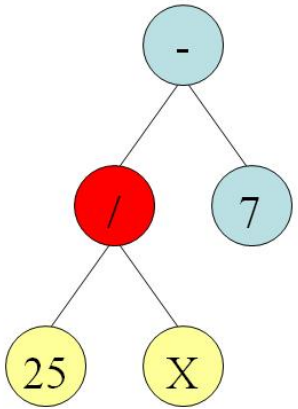
Recombination

(+ X (* 3 Y))



Pick a random node in each program

(- (/ 25 X) 7)

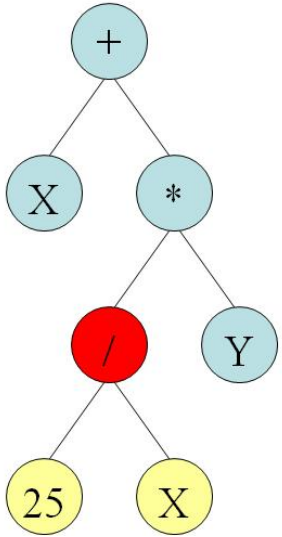


2016.03.15

106

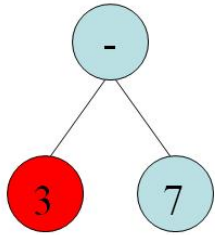
Recombination

(+ X (* (/ 25 X) Y))



Swap the two nodes

(- 3 7)



Recombination

- Parent selection typically fitness proportionate.
- **Over-selection** is used to deal with very large populations.
  - rank population by fitness and divide it into two groups:
  - group 1: best $x\%$ of population, group 2 other $(100\text{-}x)\%$.
  - 80% of selection operations chooses from group 1, 20% from group 2.
- Motivation: to increase selection pressure.
- %'s come from rule of thumb.

Survivor Selection

- Typical: generational scheme (thus none).
  - No elitism.
  - The number of offspring created is the same as the population size.
  - All individuals have a life span of one generation.
- Recently steady-state is becoming popular for its elitism.

Bloat

- Bloat = “survival of the fattest”,
  - the tree sizes in the population are increasing over time.
- Ongoing research and debate about the reasons.
- Needs countermeasures:
  - Prohibiting variation operators that would deliver “too big” children.
  - Parsimony pressure: penalty for being oversized.

Real World Applications

- Neural Network Optimization.
- Image Analysis.
- Generation of a knowledge base for expert systems.
- Fuzzy Logic Control.
- Hardware Evolution (Field-Programmable Gate Array).