Crowding is used both to determine which individuals to keep from the last allowable front (F3 in graph) AND in Tournament Selection to generate new offspring

NSGA-II

Step 1 Combine parent and offspring populations and create $R_t = P_t \cup Q_t$. Perform a non-dominated sorting to R_t and identify different fronts: $\mathcal{F}_i$, $i = 1, 2, \ldots$, etc.

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

Step 3 Perform the Crowding-sort$(\mathcal{F}_i, <_c)$ procedure (described below on page 236) and include the most widely spread $(N - |P_{t+1}|)$ solutions by using the crowding distance values in the sorted $\mathcal{F}_i$ to P_{t+1}.

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

The overall crowded distances of the four solutions are:

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

Handout 11-16-11

Board Distance for Points in Fronts 1 & 2(p. 238)

Table 21 The fitness assignment procedure under NSGA-II.

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

So the numbers computed on previous slide are imput to this table.

Fig 140: Cuboids for points d and 1
Fig 141:New Parent Population

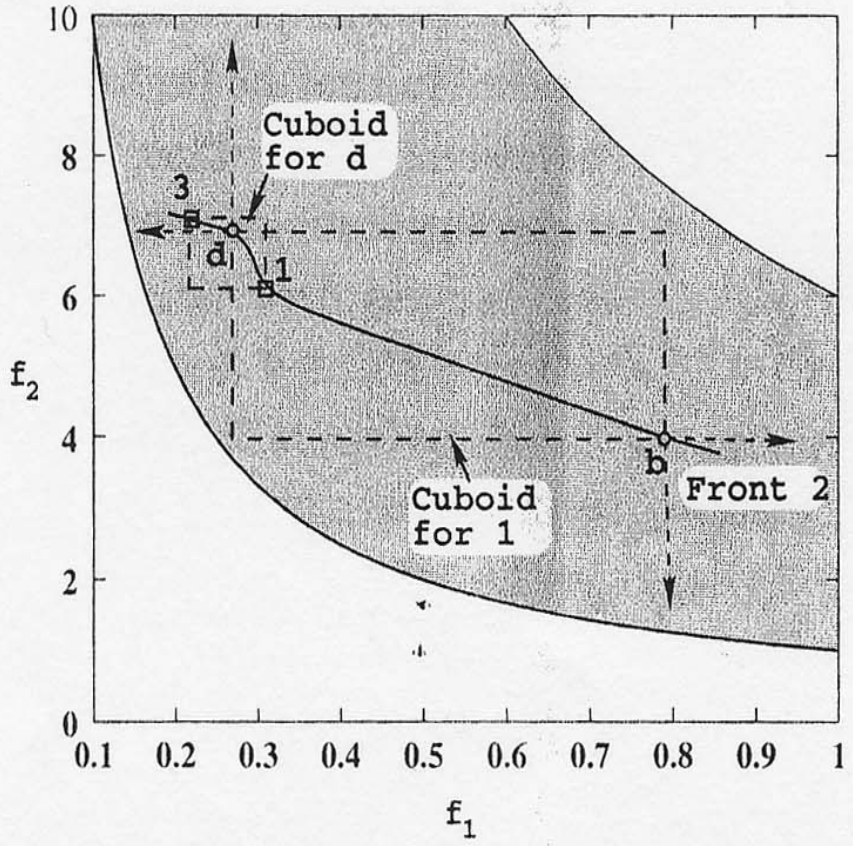


Figure 140 The cuboids of solutions 1 and d. The cuboids for solutions 3 and b extend to infinity.

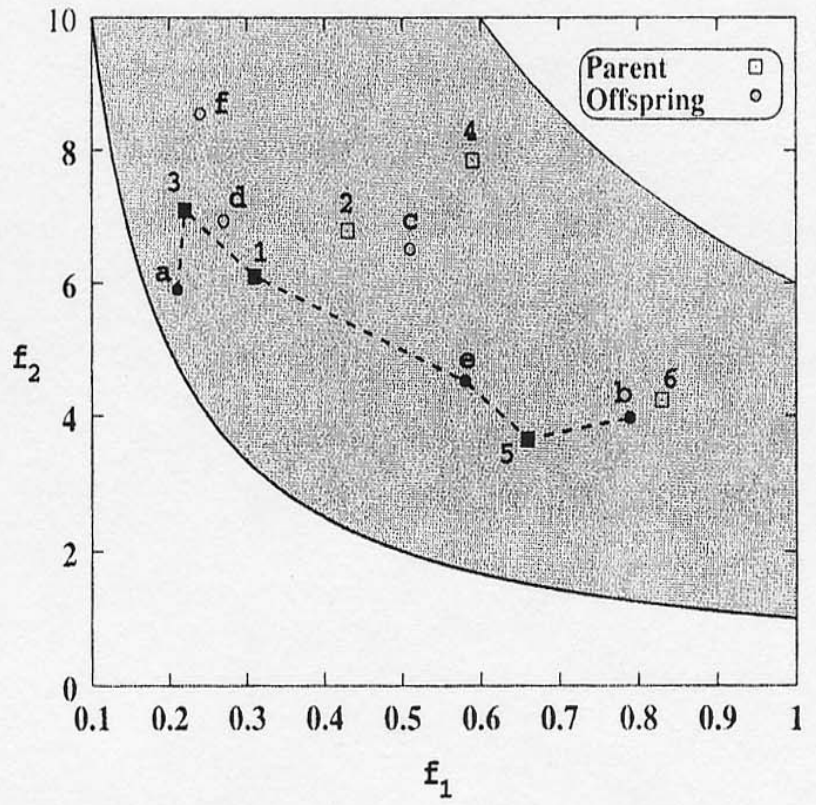


Figure 141 The new parent population P_{t+1} joined by dashed lines.

The crowding distance is the SUM of the LENGTH of 2 adjacent sides of the cuboid

Left-most column is combination of parents P_t and children Q_t. Middle column is the combined population sorted into fronts. P_{t+1} is the new population of parents

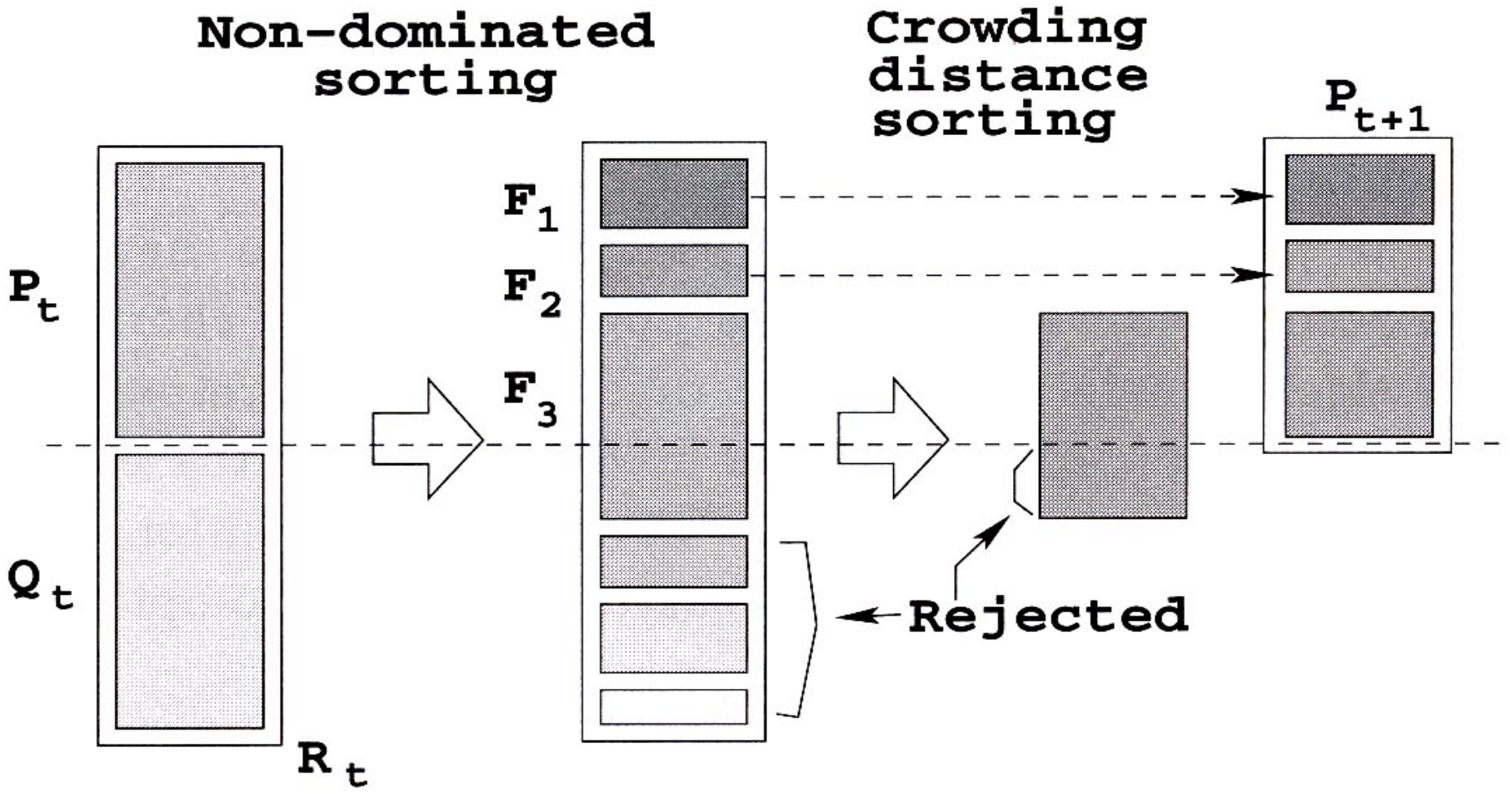


Figure 137 Schematic of the NSGA-II procedure.

50 Generations with ELITISM Without Mutation

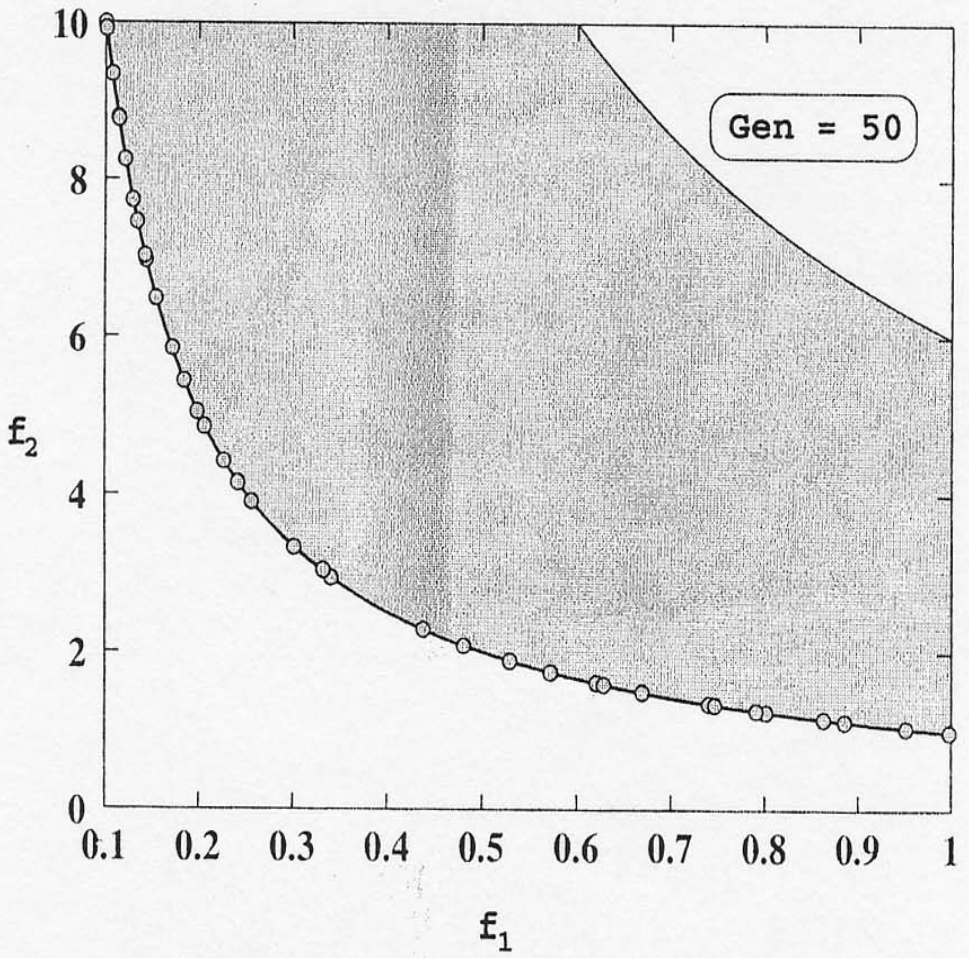


Figure 142 The population after 50 generations with the NSGA-II without mutation.

p. 241

NSGA Solution after 50 generations (without elistism or mutation)

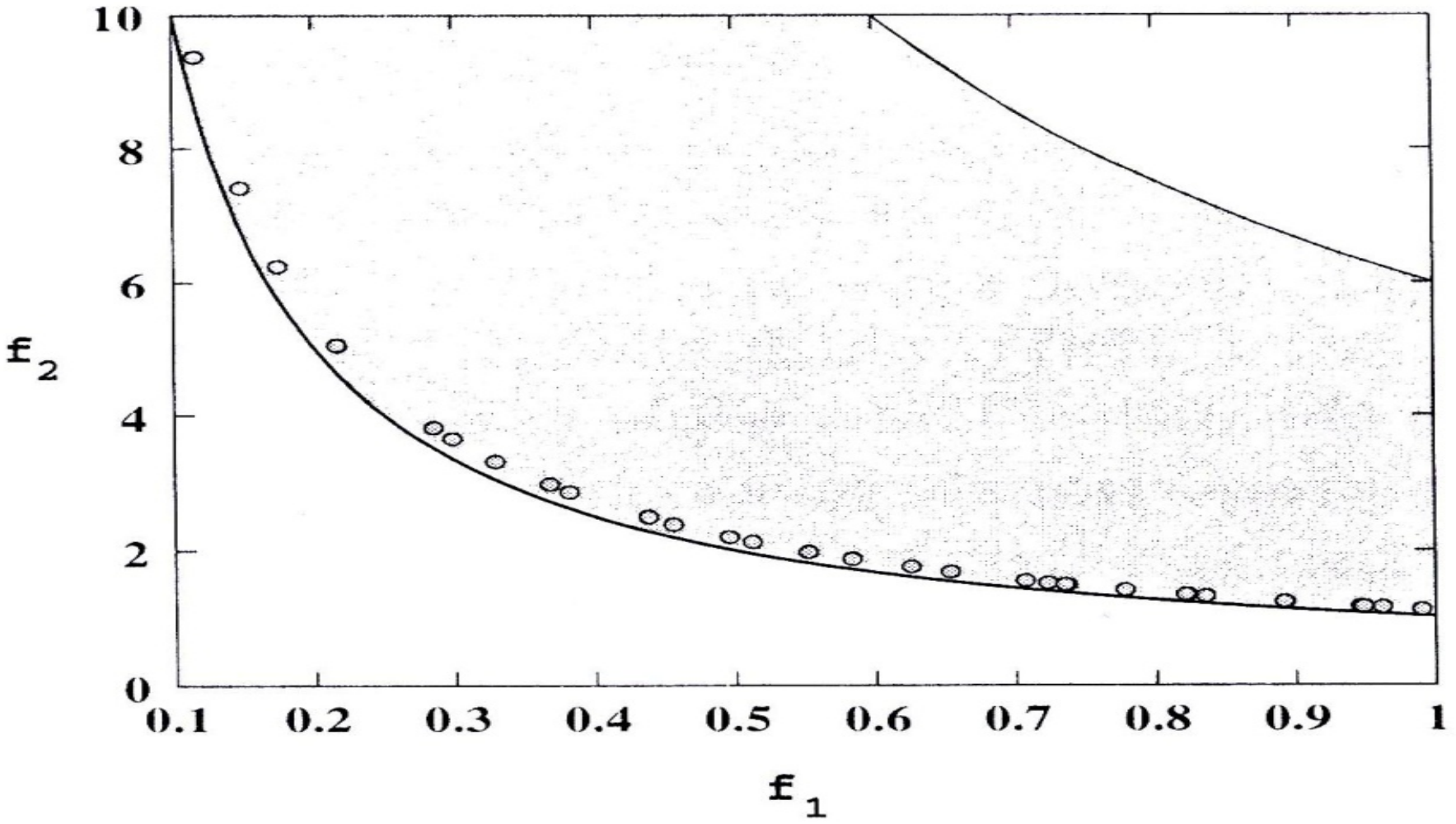


Figure 117 The NSGA distributes solutions near the Pareto-optimal region at the 50-th generation.

NGSA with Mutation 500 generations

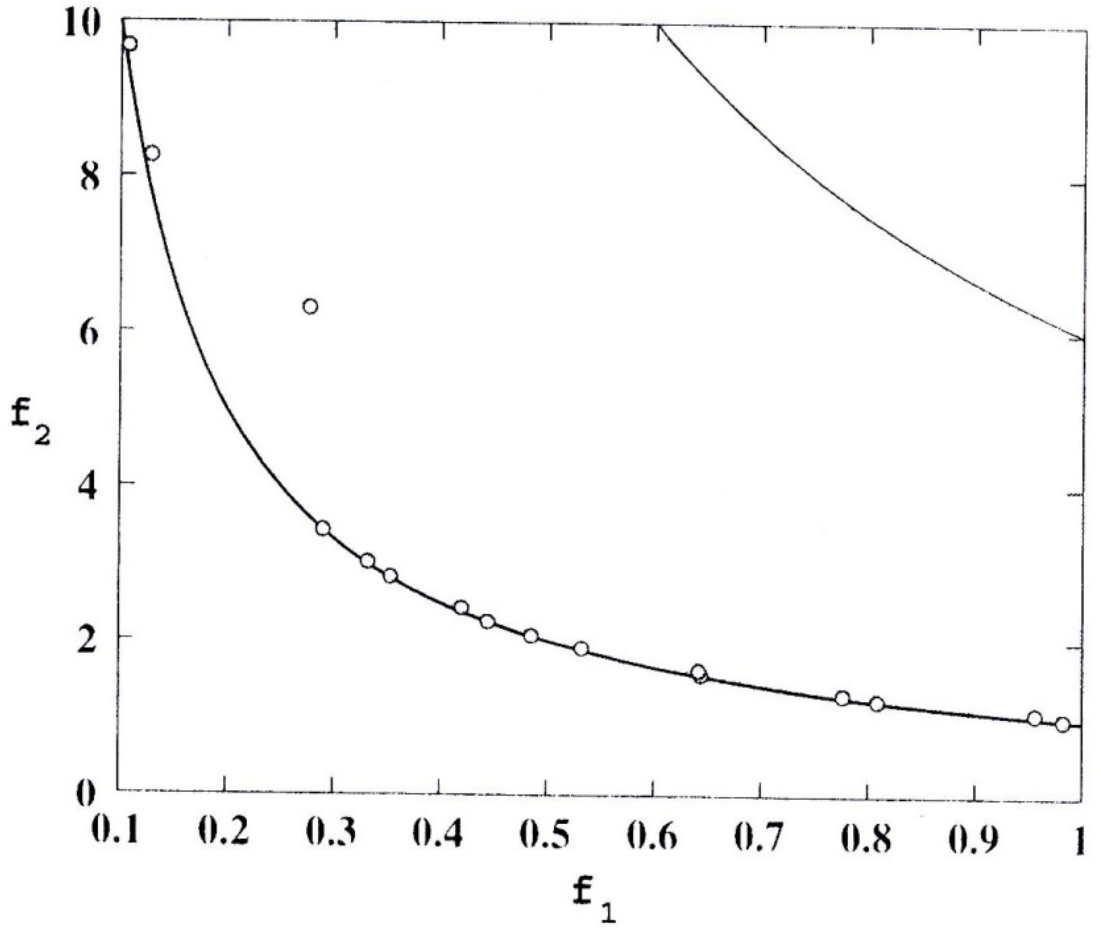


Figure 120 Population at the 500-th generation with an NSGA having a mutation operator.

500 Generations with Elitism with and without Mutation

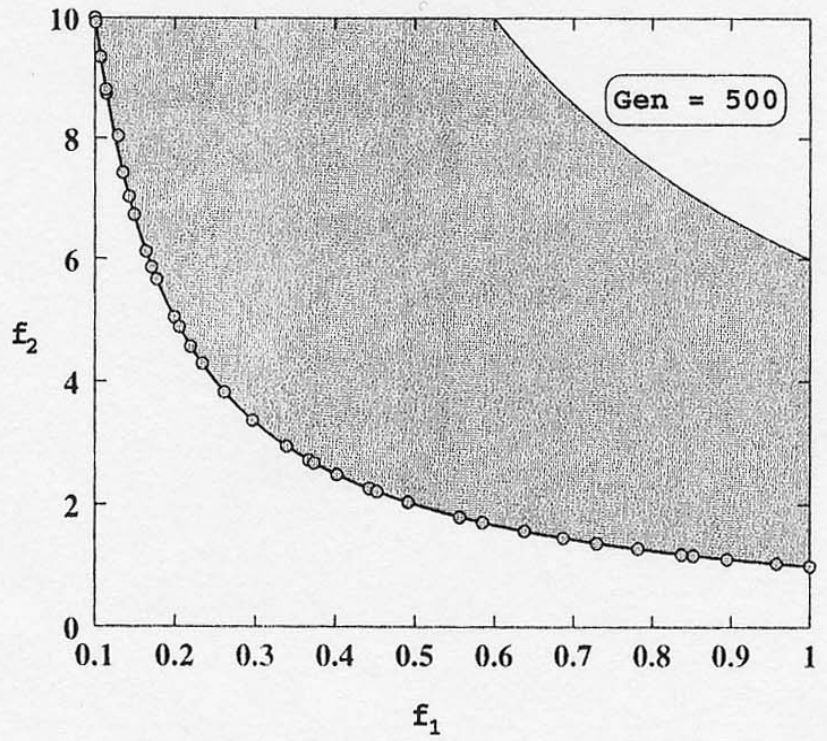


Figure 143 The population after 500 generations with the NSGA-II without mutation.

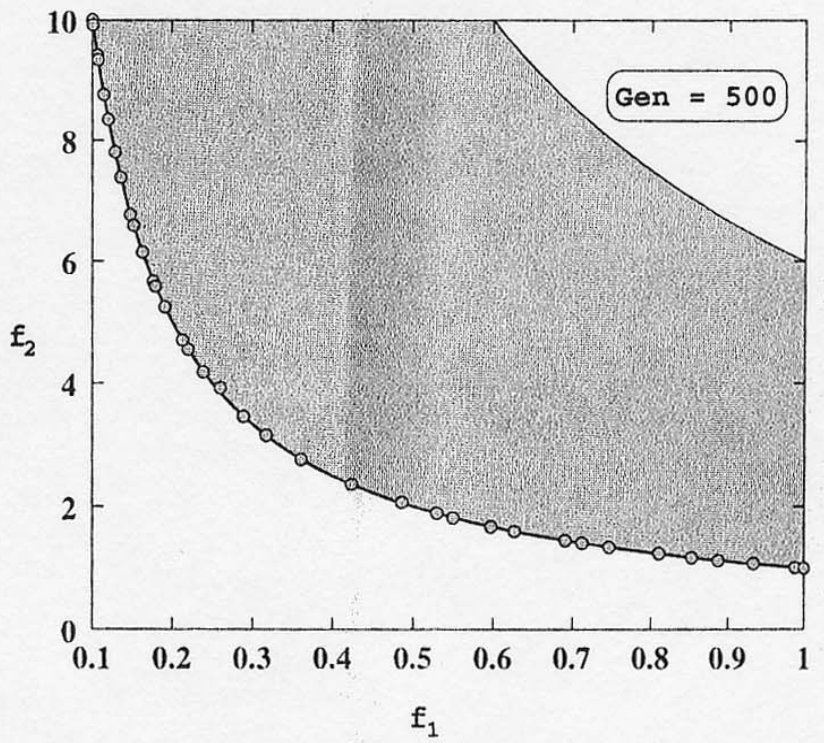


Figure 144 The population after 500 generations with the NSGA-II with mutation.

NSGA after 500 generations (without Mutation)

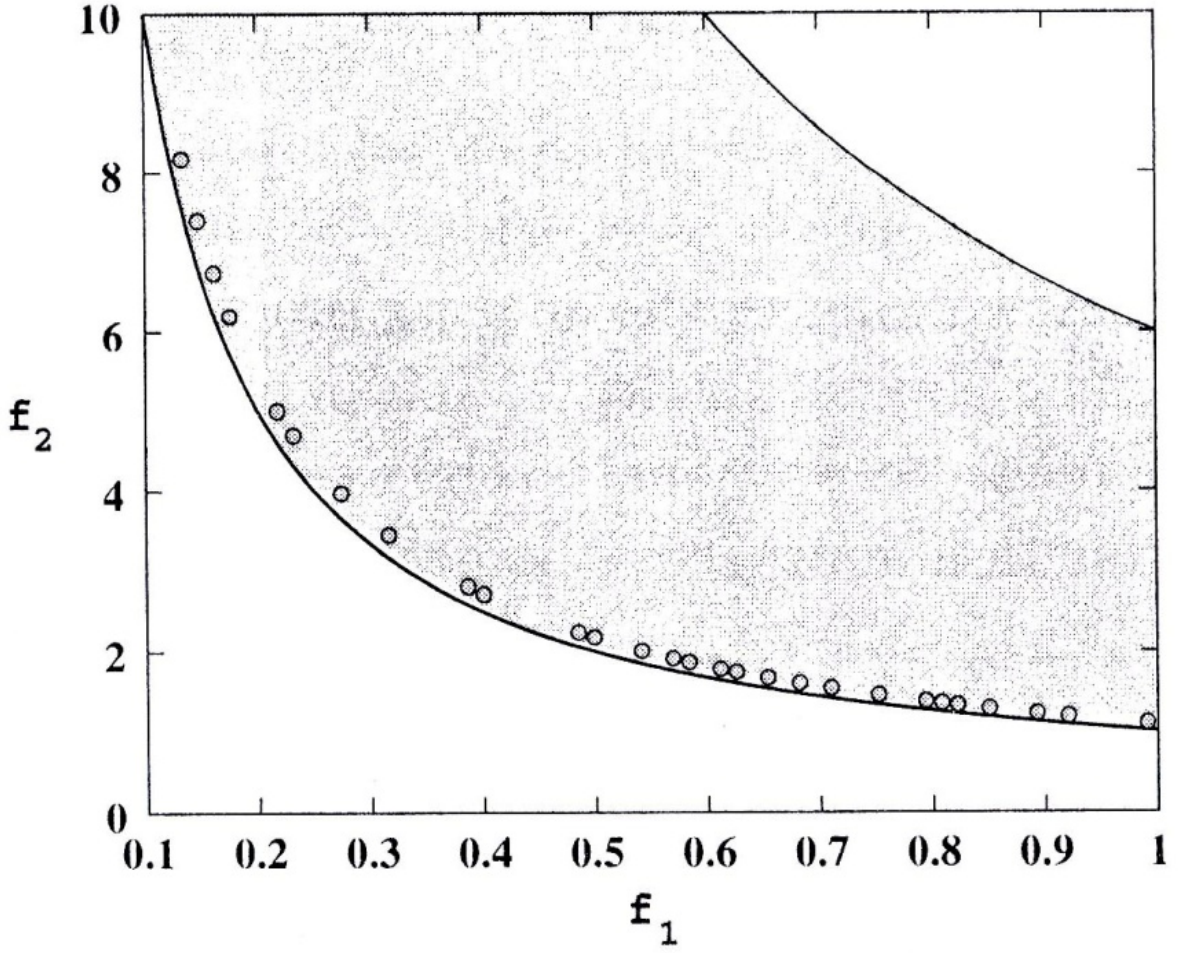


Figure 118 The NSGA distributes solutions near the Pareto-optimal region at the 500-th generation.

Two types of Errors

Poor Distribution

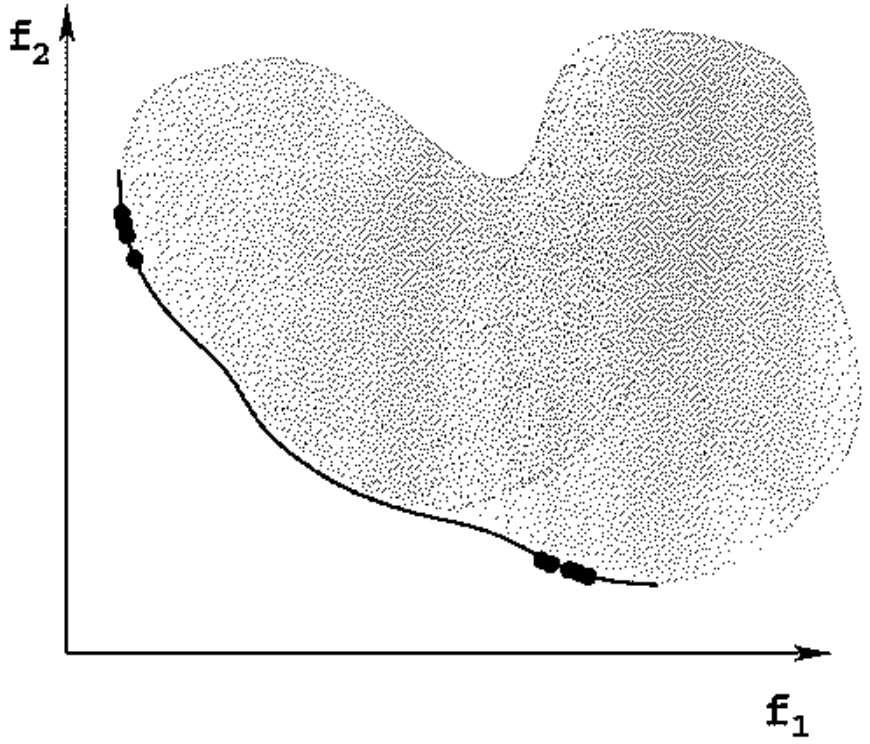


Figure 183 Convergence is good, but distribution is poor (by Algorithm 1).

Poor Convergence

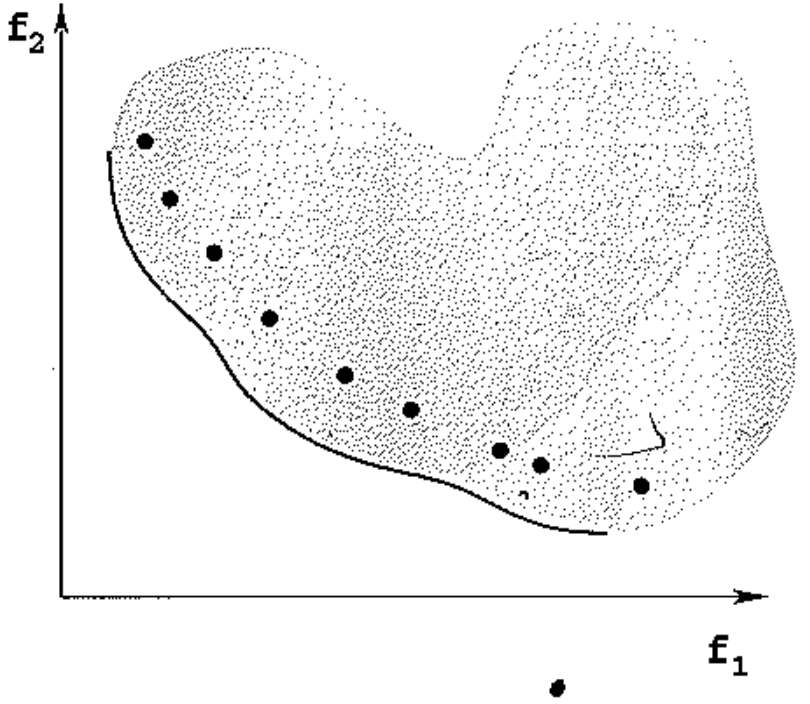


Figure 184 Convergence is poor, but distribution is good (by Algorithm 2).

Conclusions on Multi Objective Optimization

- Most problems are multi objective problems
- We usually use single objective methods (and perhaps incorporate other objectives as constraint) because
  - Multi objective tends to be much more computationally expensive than single objective optimization
  - It can be complex to explain multi objective results (which can be helped by visualization methods)
  - It can be difficult to quantify all the objectives.

Conclusions on Multi Objective Optimization

- If the feasible space is not convex, nonlinear programming or other methods involving changing weights between objectives, may miss part of the Pareto front.
- NSGA-II is an effective way to find the Pareto Front, but it does take a large number of objective function evaluations.
- Part of the goal for Pareto Front calculation is to find points on (or very close to) the true Pareto Front and also not to have big gaps between those points.

Conclusions on Multi Objective

- The inclusion of Elitism via Deb's method adds complexity to the code.

- The inclusion of Elitism appears to significantly improve performance.

Genetic Programming

- Genetic Programming applies the concept of Genetic Algorithms to Automatic Programming.
- Automatic Programming refers to algorithms that generate a computer program to do a specific task.
- Why would you want to use Automatic Programming?

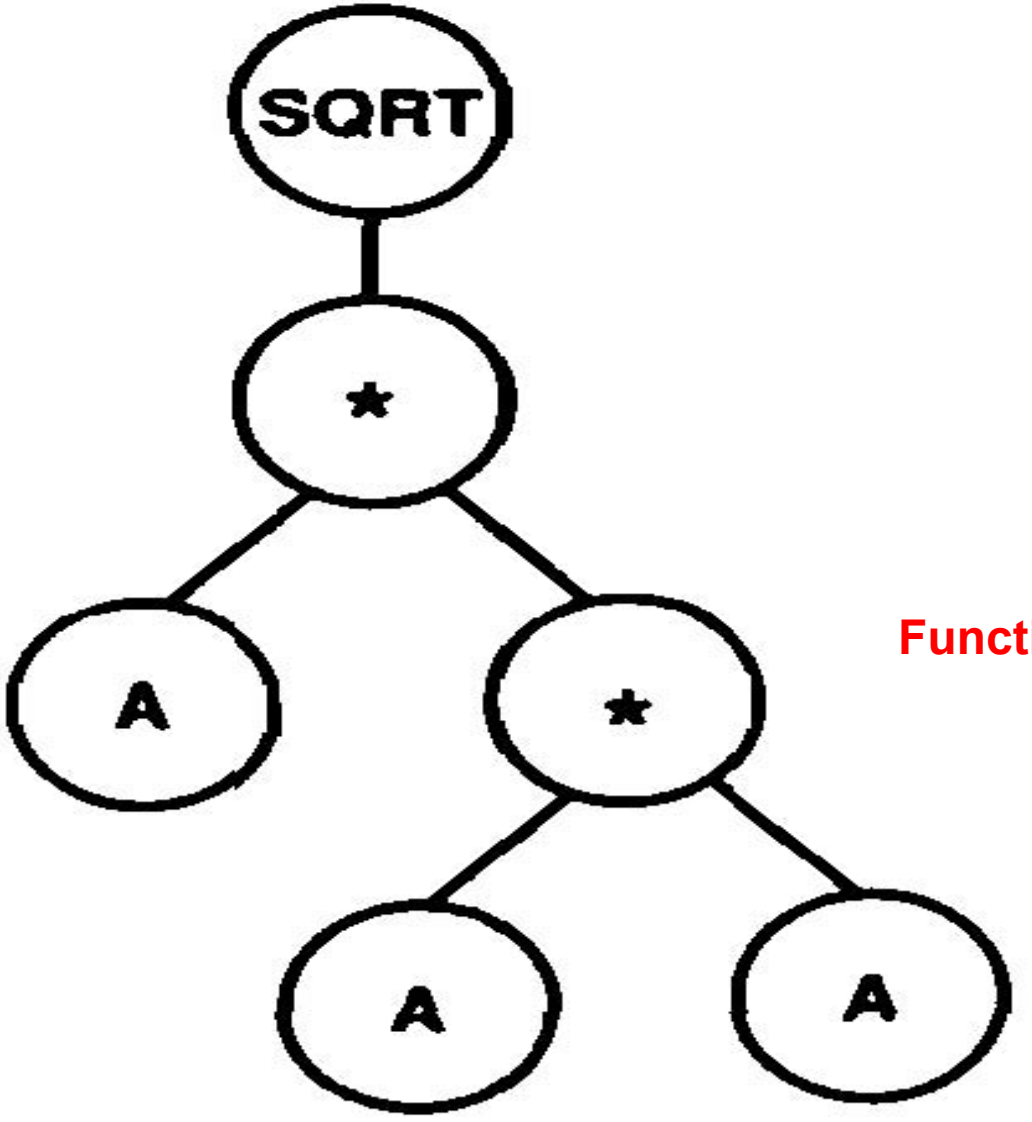
- References to this area include:
- Koza, John (1992) *Genetic Programming* MIT Press
- Mitchell, Melanie (1996) *An Introduction to Genetic Algorithms*, MIT Press (pp 35-53)

First Example: Using Genetic Programming to find Relationship between radius and orbit of planet

- Suppose that you want to find the relationship between the average **distance** of a plant from the sun (A) and the orbital **period** of the planet (P).
- You are given the following data

Planet Name	A	P
Venus	.72	.61
Earth	1.00	1.00
Mars	1.52	1.84
Jupiter	5.20	11.9
Saturn	9.23	29.4
Uranus	19.1	83.5

Genetic Programming uses Parse Trees rather than binary strings for decision variables



Function

Terminal

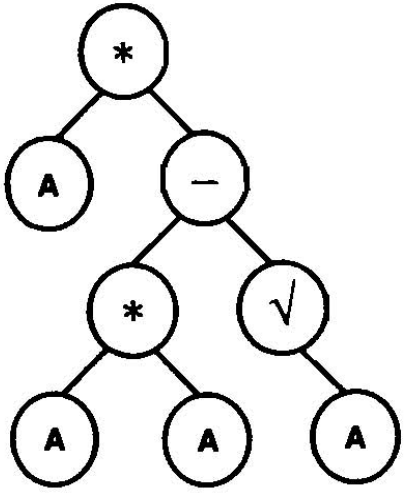
Example of a parse tree for $(A^3)^{1/2}$ = SQRT(*A(*AA))) (*Lisp*)

This is an example of simple program

Contents of Parse Tree

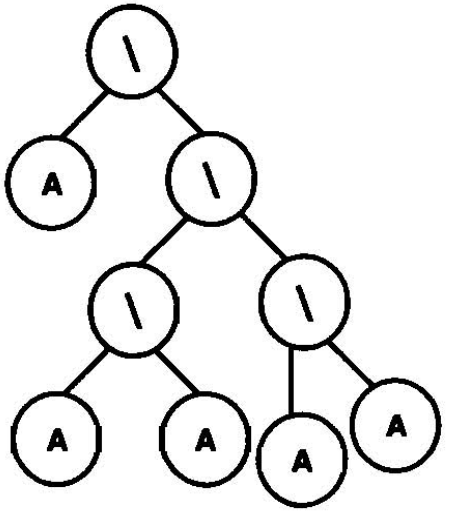
- The tree consists of
- **Functions**- the set of commands available for building solution expressions
- **Terminals**- the set of data (includes decision variables) for the solution expression
- The following shows more examples of parse trees representing a possible solution to the problem.
-

Three possible solutions to the relationship between P and A



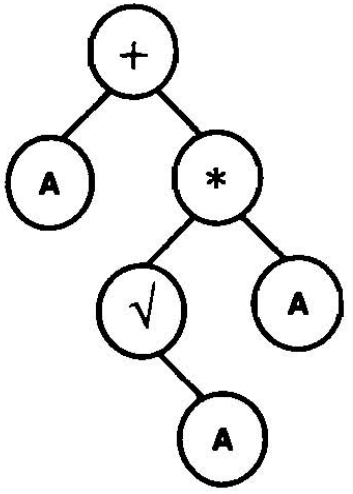
A * [(A * A) – √ A]

fitness → = 1



A \ [(A \ A) \ (A \ A)]

= 3 = fitness



A + (√ A * A)

fitness = 0

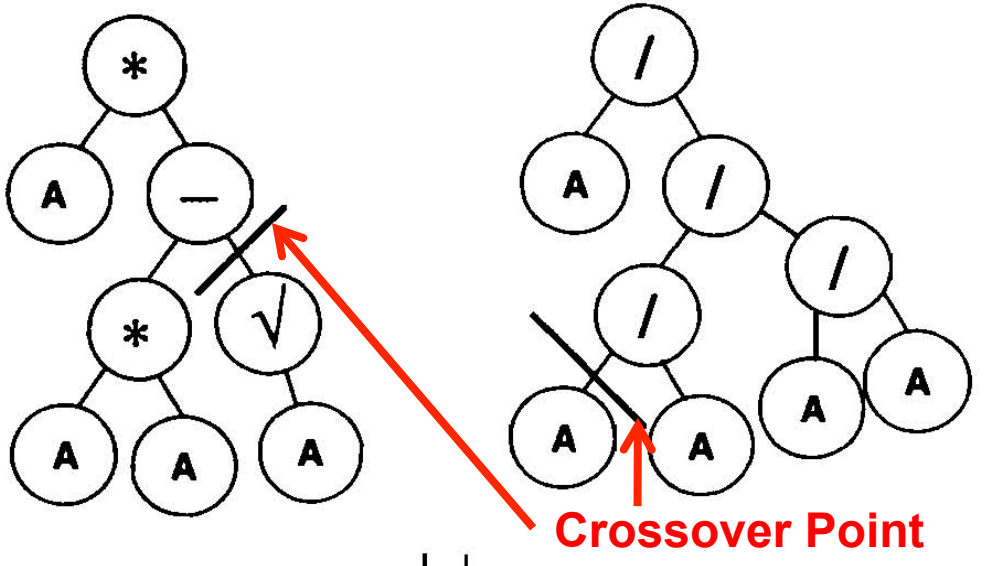
Fitness Cases:

Planet	A	Correct Output (P)
Venus	0.72	0.61
Earth	1.00	1.00
Mars	1.52	1.84
Jupiter	5.20	11.9
Saturn	9.53	29.4
Uranus	19.1	83.5

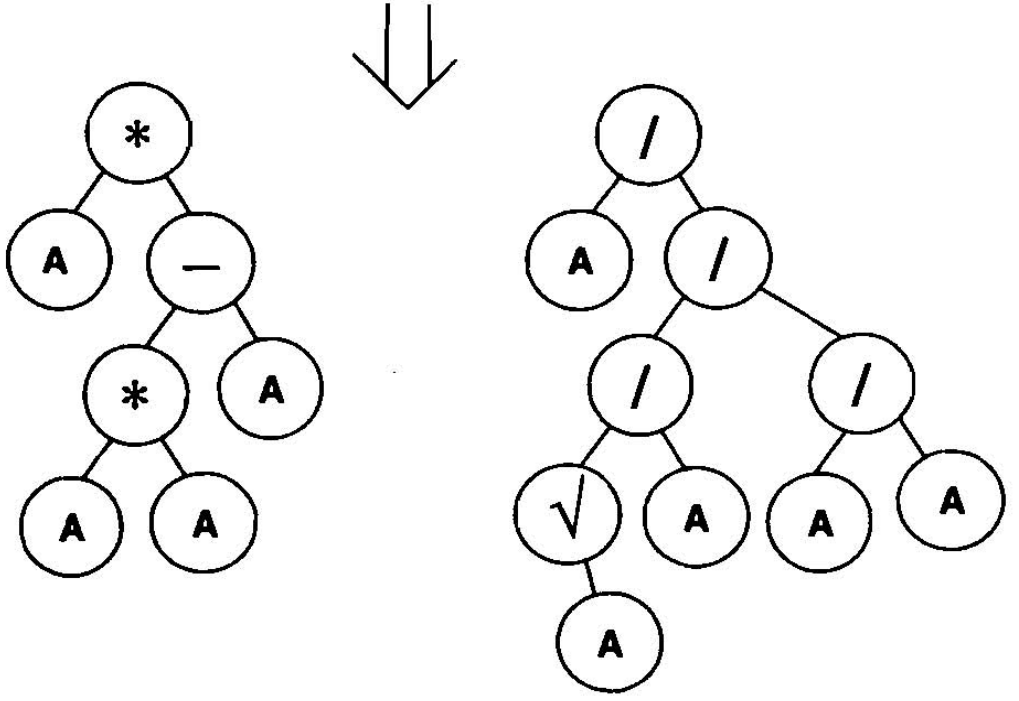
fitness depends on $f_i(A) - P$

Crossover in Genetic Programming

Parents



Offspring



Genetic Programming is …

- Essential aspects of GP
  - Dynamic length structures
  - Evaluate by Execution
  - Syntax Preserving Operators
- Non essentials of GP
  - Parentheses (Lisp Syntax)
  - Lisp Commands
  - Tree Structures

How Different from GA?