Genetic Algorithms

CE 377K

February 5, 2015

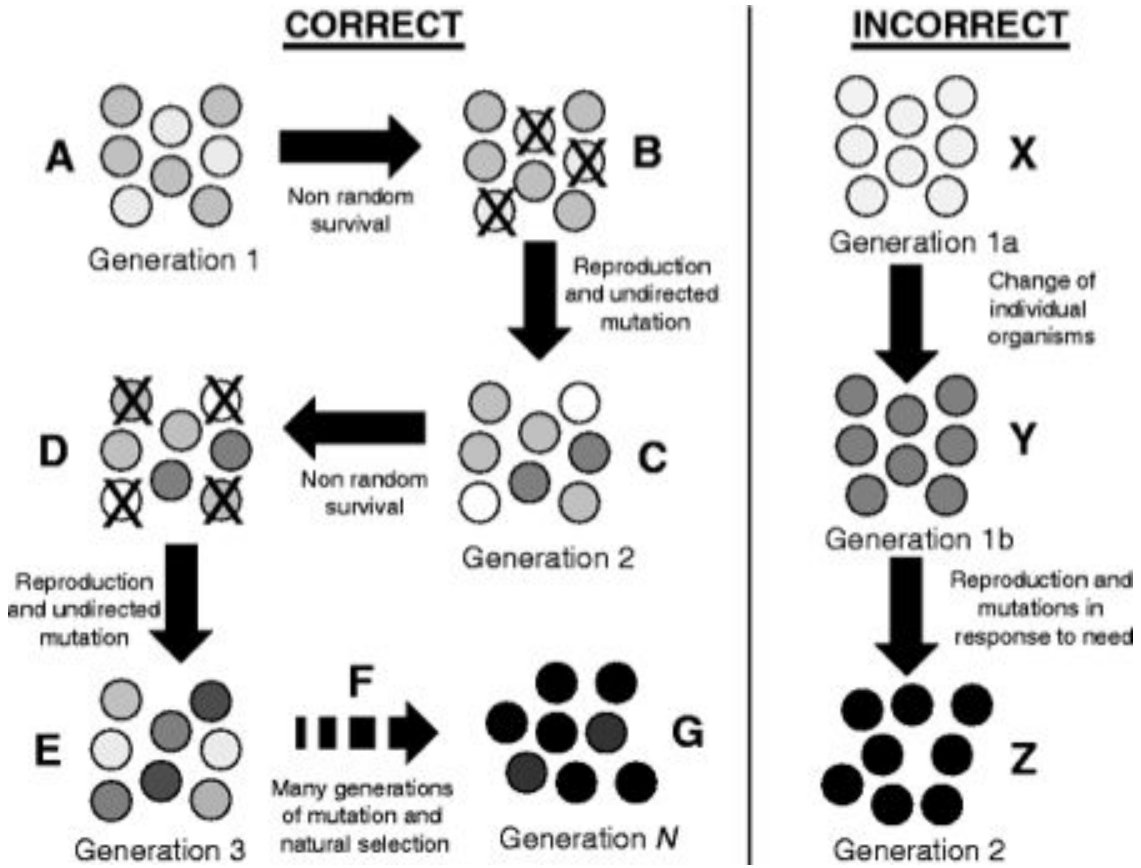
REVIEW

HW 1 posted, due February 10

Heuristics
Simulated annealing

GENETIC ALGORITHMS

Genetic algorithms attempt to find a good-quality solution by mimicing the process of natural selection.



Like simulated annealing, it is just a heuristic by can often find high-quality solutions to complicated problems.

In a nutshell, natural selection in most animals and plants involves the following principles:

- "Survival of the fittest": organisms better suited to their environment are more likely to reproduce.
- Sexual reproduction: two organisms produce a new organism by combining DNA from the parents.
- Mutation: rare, random changes in a gene due to damaged DNA or a copying error.

We want to search for good solutions to an optimization problem by making an analogy to these principles.

In contrast to simulated annealing, which had just one current solution, in genetic algorithms we maintain a *population* of many feasible solutions.

Furthermore, there will be multiple *generations*, each with their own population of solutions.

Starting with an initial generation, we want to create a generation of "offspring" which ideally have lower objective function values.

We can adapt the principles of natural selection in the following way:

- "Survival of the fittest": solutions with lower objective function values are more likely to "reproduce."
- Sexual reproduction: a new feasible solution is created by combining aspects of two "parents."
- Mutation: rare, random changes in a solution.

1. Generate an initial population of N feasible solutions (generation 0), set generation counter $g \leftarrow 0$.
2. Create generation $g + 1$ in the following way, repeating each step N times:
   (a) Choose two "parent" solutions from generation g.
   (b) Combine the two parent solutions to create a new solution.
   (c) With probability p, mutate the new solution.
3. Increase g by 1 and return to step 2 unless done.

How to generate an initial population?

It is more important for the initial population to be *highly diverse* than for the initial solutions to have low objective function values.

(Genetic algorithms get most of their power from breeding solutions together, mutation plays a secondary role.)

There is nothing wrong with randomly generating all of the initial solutions (just make sure they are feasible).

How to select parent solutions?

The *tournament selection* rule works like this: pick a tournament size t.

From the current generation, randomly select t solutions. The one with the lowest objective function value is the first parent.

Repeat the tournament by selecting t more solutions randomly. The one with the lowest objective function value is the second parent.

How to "breed" solutions together?

This part can be trickier, and varies from one problem to the next. We need to combine two feasible solutions in a way that results in a new feasible solution that resembles its parents in some way.

Some examples (again, not the only choices for these problems):

Transit frequency setting: List the routes to which each fleet in the bus is assigned. Generate a new solution by randomly assigning a bus to the route in one of the two parent solutions.

Scheduling maintenance: List the facility to which each maintenance action in each year is assigned. Generate a new solution by randomly assigning that maintenance action to a facility in one of the two parent solutions.

Facility location: Randomly assign each facility to its location in one of the two parent solutions.

How to mutate solutions?

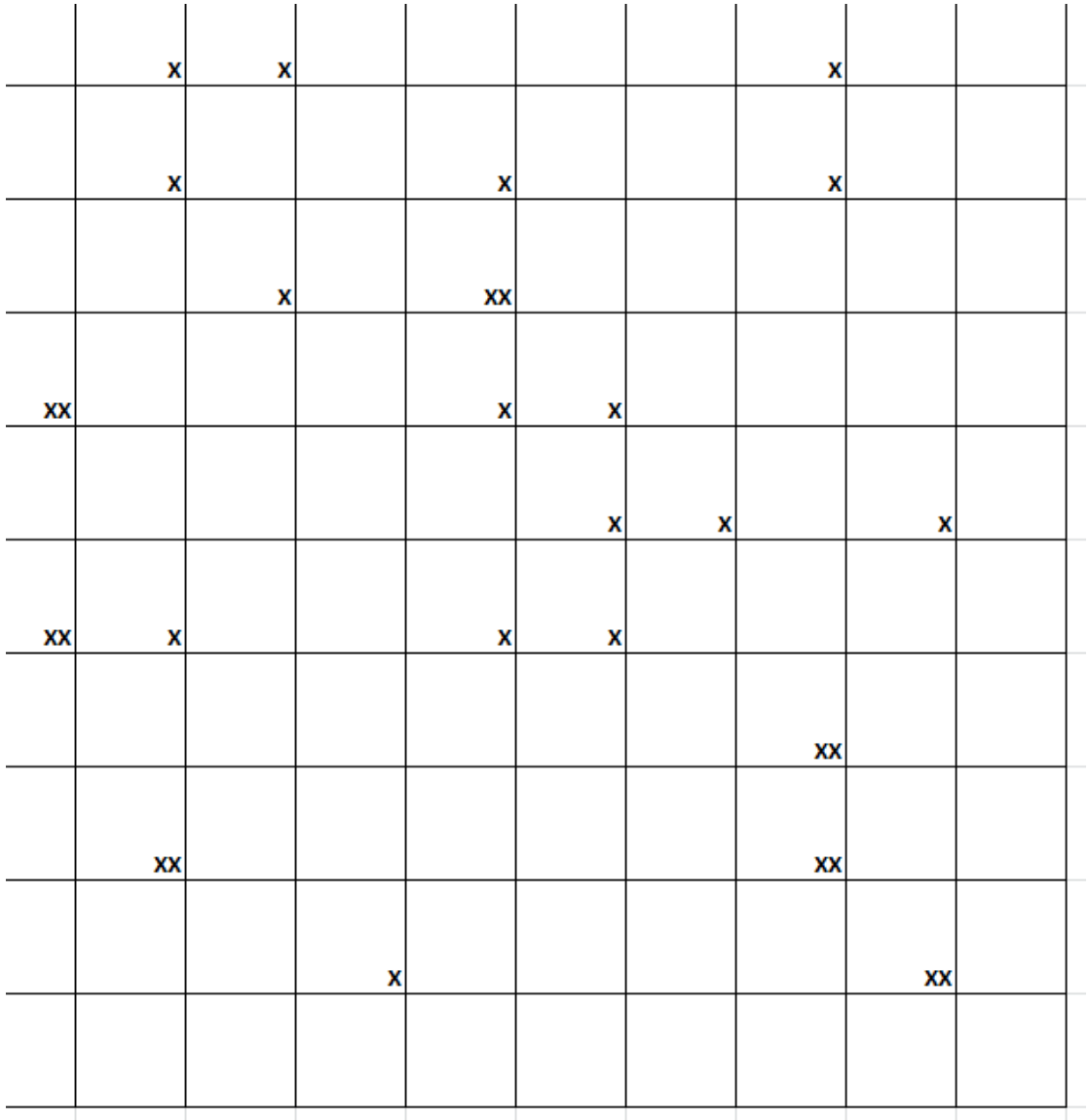
The same way as neighbors were generated in simulated annealing: randomly change the solution in a minor way.

EXAMPLE

Let's return to the facility location problem. In this instance, there are 100 eligible facility locations (costs shown below):

8.7	13.5	11.0	10.2	6.3	6.4	12.4	13.5	9.7	6.8
5.8	12.4	8.8	15.1	14.7	13.9	9.0	5.2	8.7	12.7
11.8	9.8	12.4	12.6	6.3	13.3	7.7	10.2	13.6	13.5
5.4	6.9	11.1	8.4	14.2	10.9	10.7	11.7	8.1	8.9
4.9	12.5	10.8	6.6	12.0	6.8	11.9	9.2	9.5	10.0
14.4	9.7	8.6	11.6	8.0	6.7	12.7	5.9	7.6	14.1
7.7	9.6	6.4	9.9	9.2	13.3	12.3	14.7	15.0	5.1
7.6	14.7	7.1	5.5	5.5	9.7	9.7	14.4	7.9	15.1
10.9	12.5	9.2	12.1	14.8	6.4	6.1	10.5	12.5	10.8
12.0	5.9	13.1	10.0	11.9	10.0	8.6	8.4	10.7	5.3

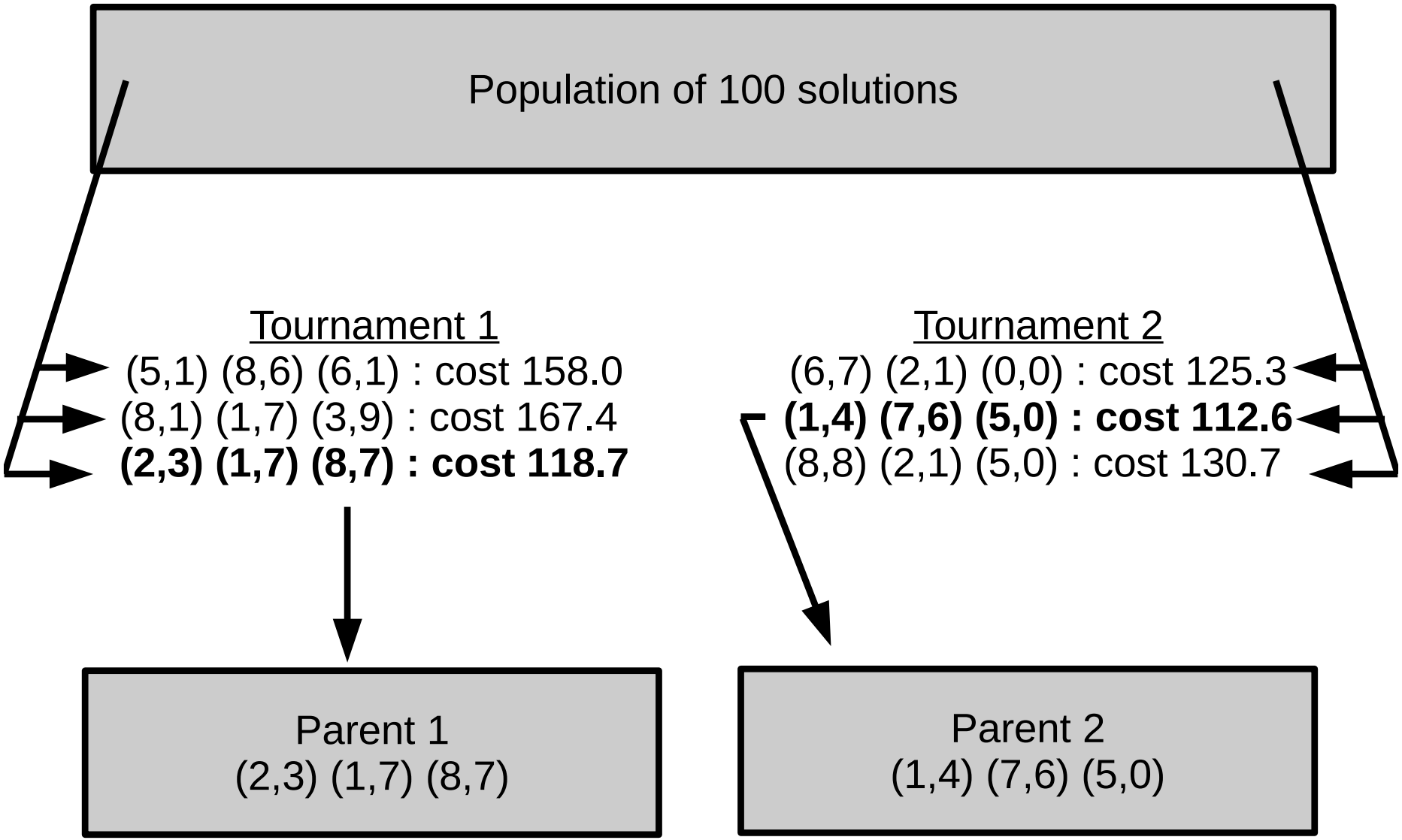
Here are the locations of the 30 customers.

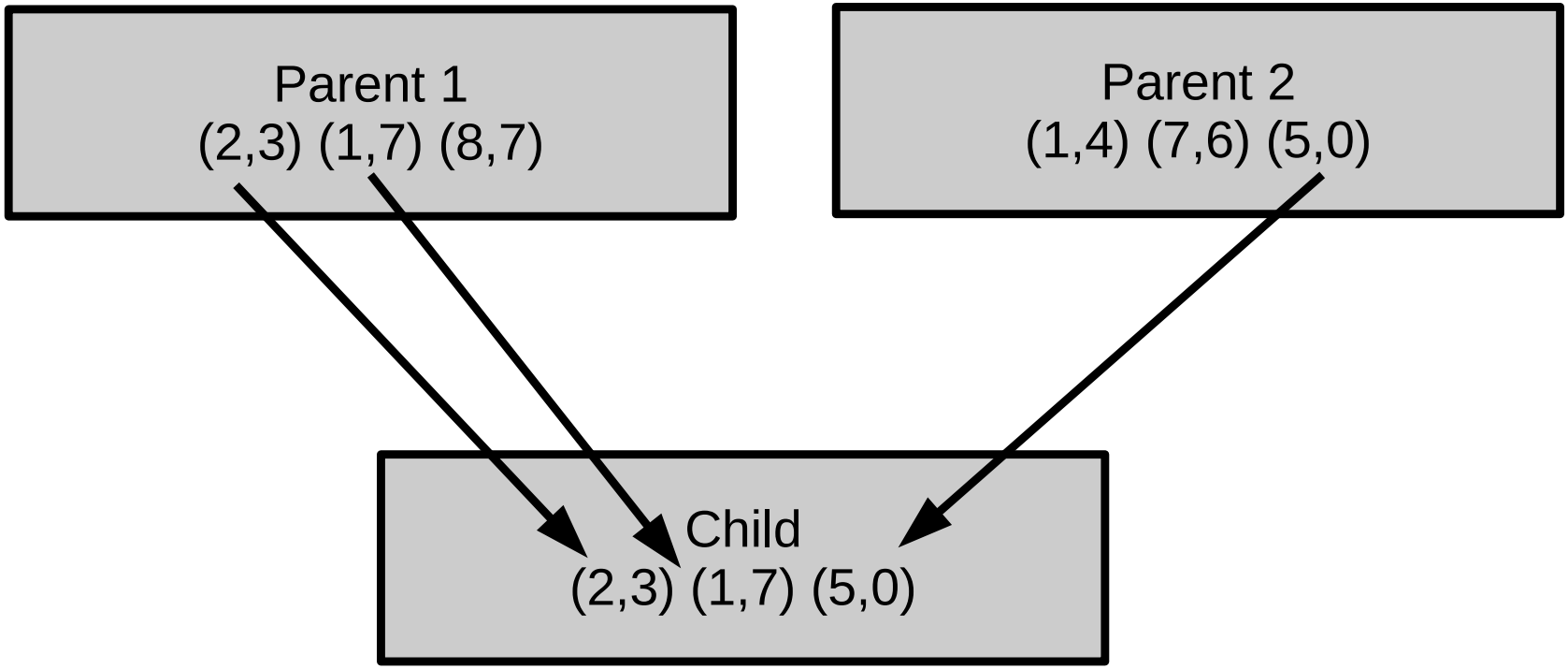


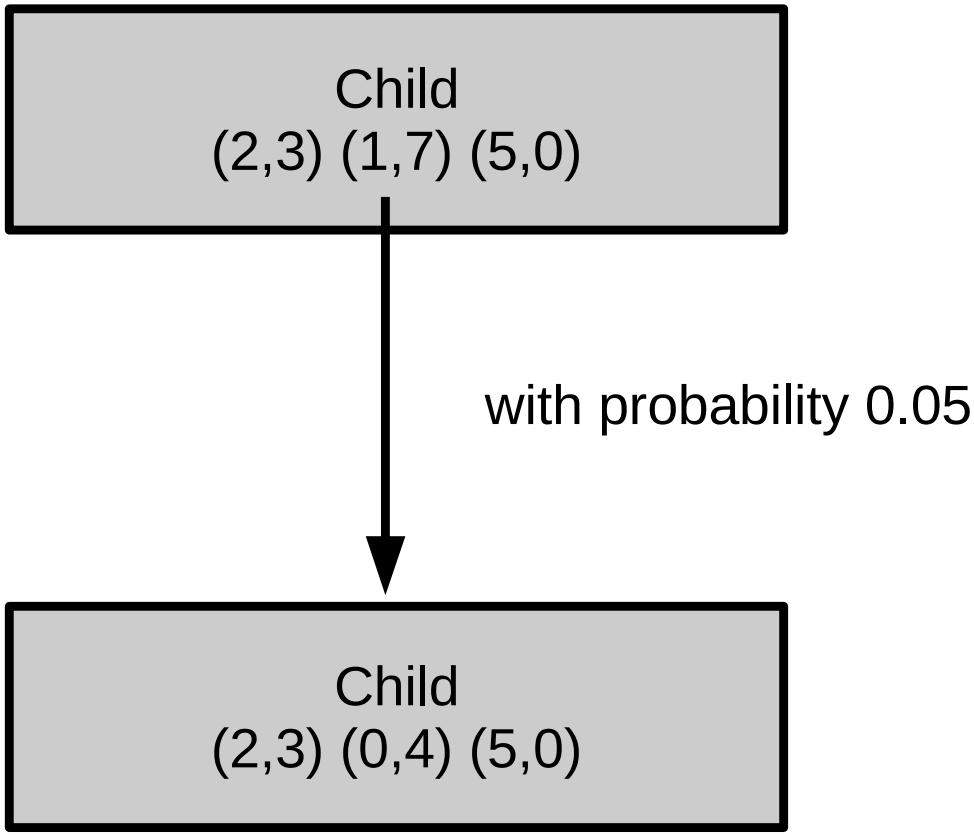
Through trial and error, the following parameters were chosen:

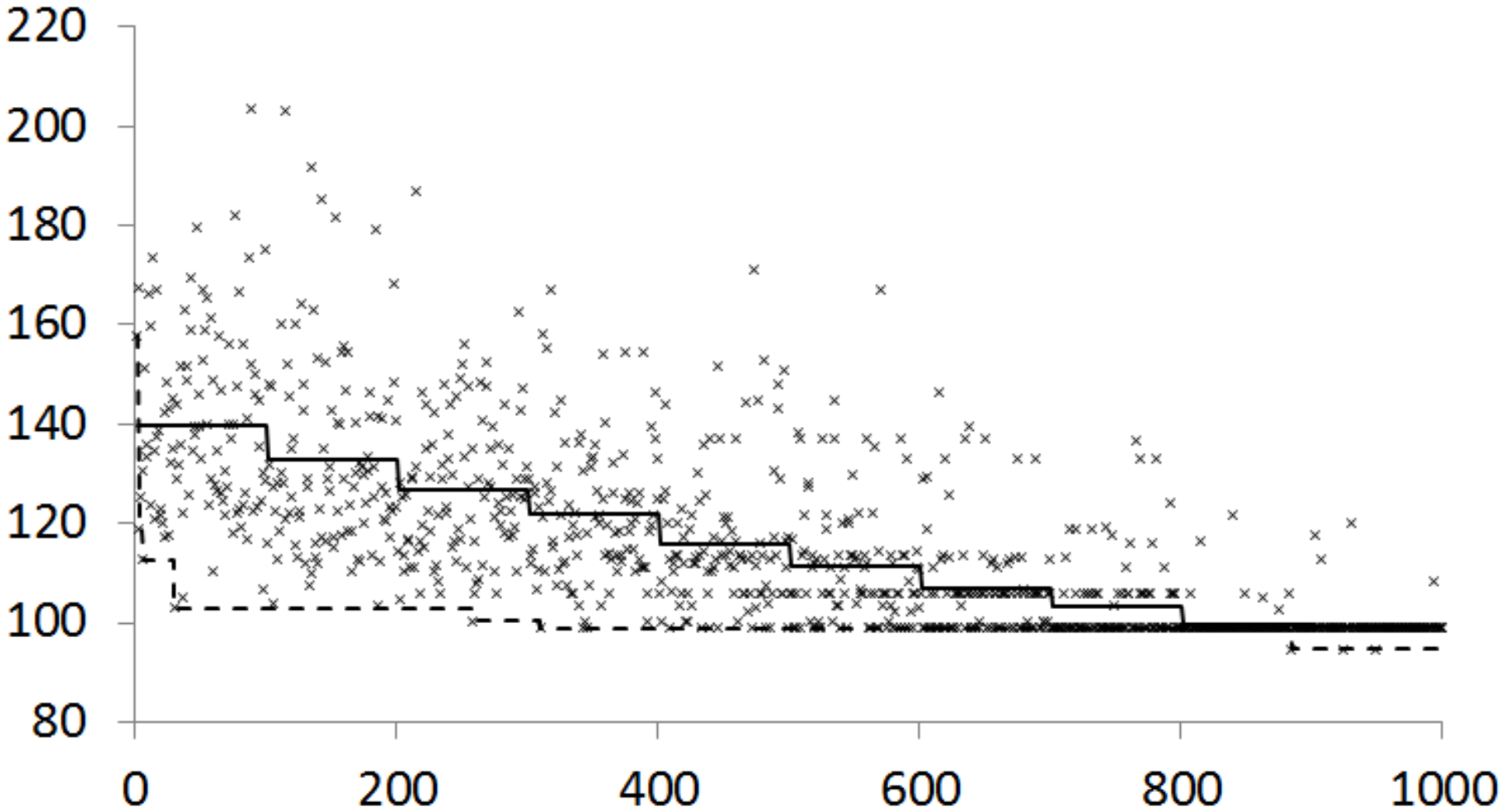
- Population size of $N = 100$
- Ten generations
- Tournament size $t = 3$
- Mutation probability $p = 0.05$

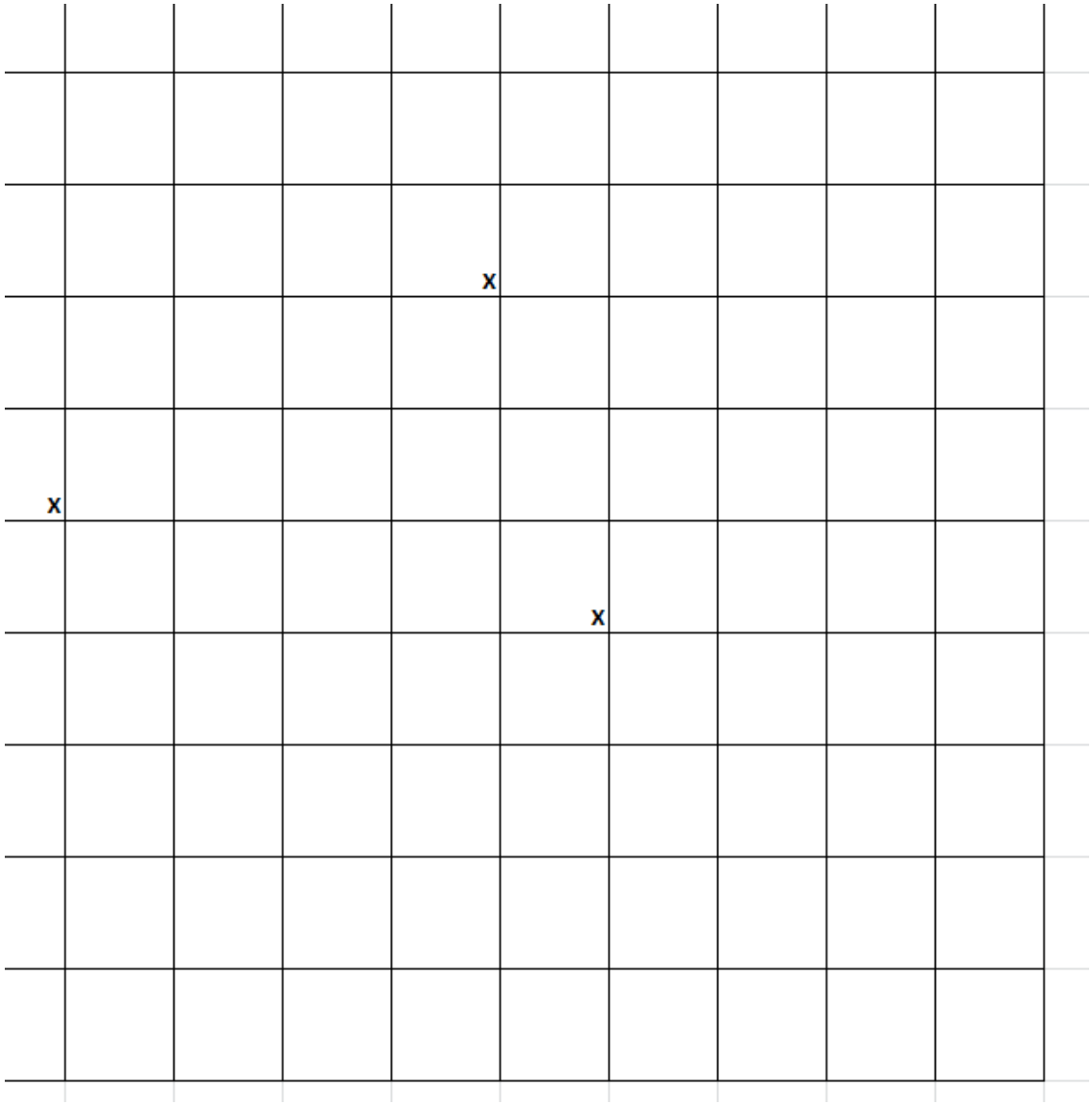
Example of how the next generation is produced




Parent 1
(2,3) (1,7) (8,7)
Parent 2
(1,4) (7,6) (5,0)
Child
(2,3) (1,7) (5,0)


Child
(2,3) (1,7) (5,0)
with probability 0.05
Child
(2,3) (0,4) (5,0)


220
200
180
160
140
120
100
80
0
200
400
600
800
1000


X
X
X