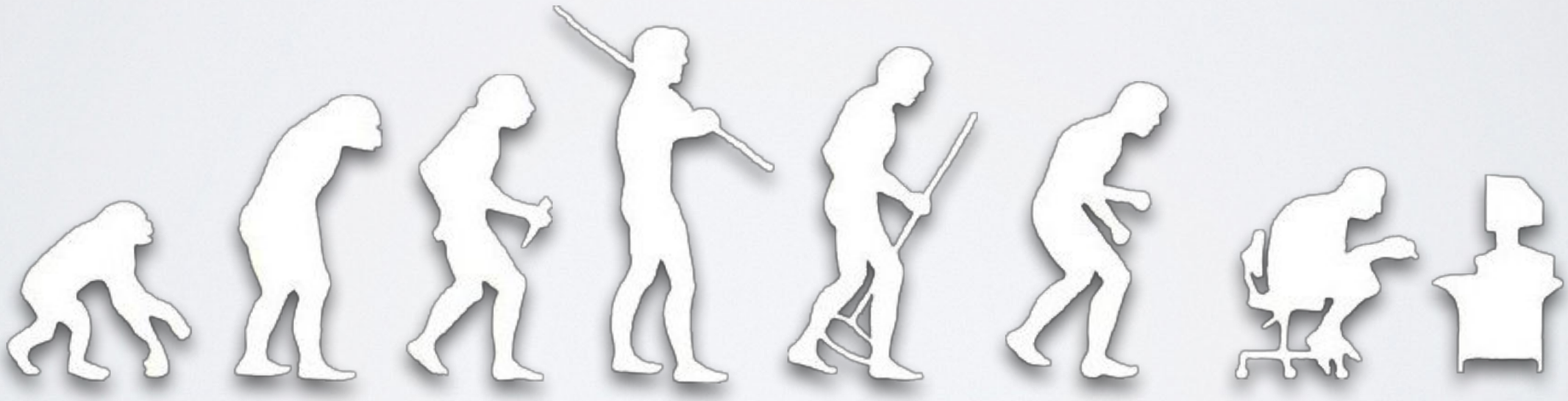
TRAVELER SALESMAN PROBLEM

Group 15

THE ALGORITHM

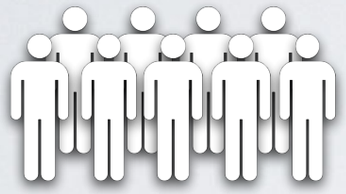


1. Create a new population

First individual is a tour of subsequent cities as they appear in the input cities list

City1 > City2 > City3…

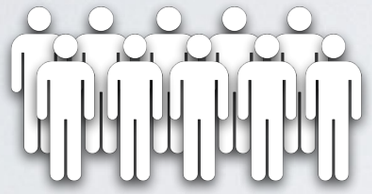
All the other individuals are random tours



1. Create a new population

First individual is a tour of subsequent cities as they appear in the input cities list

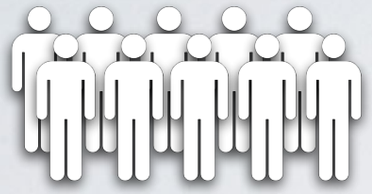
All the other individuals are random tours



2. Evolution stage

First individual is a tour of subsequent cities as they appear in the input cities list

All the other individuals are random tours

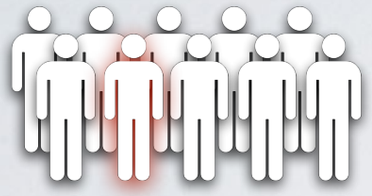


2. Evolution stage

We select the best individual of the population to be saved into the new population

2. Evolution stage

We select the best individual of the population to be saved into the new population

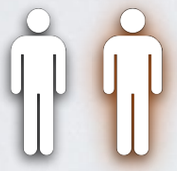


Then we take random 20% of the population and select the best out of it. This process is executed two times and the selected individuals go into the new population

2. Evolution stage

We select the best individual of the population to be saved into the new population

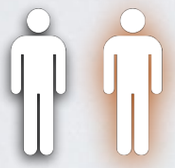
Then we take random 20% of the population and select the best out of it. This process is executed two times and the selected individuals go into the new population

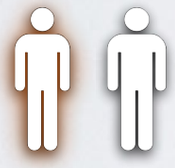


2. Evolution stage

We select the best individual of the population to be saved into the new population

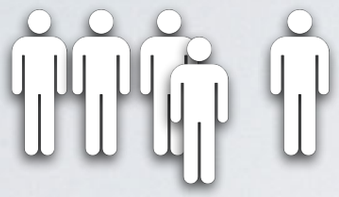
Then we take random 20% of the population and select the best out of it. This process is executed two times and the selected individuals go into the new population





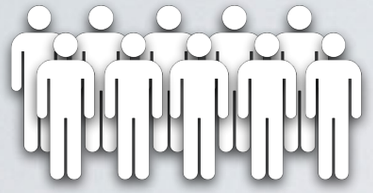
2. Evolution stage

We select the best individual of the population to be saved into the new population



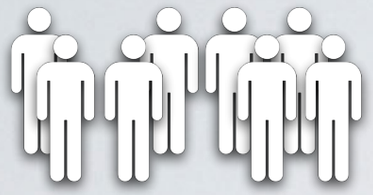
Then we take random 20% of the population and select the best out of it. This process is executed two times and the selected individuals go into the new population

NEW POPULATION:



We now have to select two parents to create a child to be inserted into the new population. We generate again a random subgroup of individuals (20%) and choose the best out of it as a parent

NEW POPULATION:



We now have to select two parents to create a child to be inserted into the new population. We generate again a random subgroup of individuals (20%) and choose the best out of it as a parent

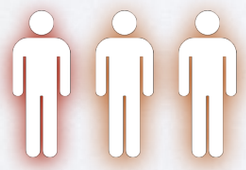
Parent 1

Parent 2

Crossover

Child

NEW POPULATION:

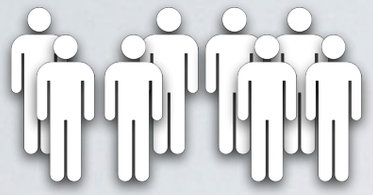


Crossover

Parent 1 / 2 / 3 / 5 / 1 / 6 / 9 / 8 / 7 / 4 /

Parent 2 / 1 / 4 / 5 / 2 / 9 / 6 / 7 / 8 / 3 /

CHILD / 4 / 2 / 5 / 1 / 6 / 9 / 8 / 7 / 3 /



We now have to select two parents to create a child to be inserted into the new population. We generate again a random subgroup of individuals (20%) and choose the best out of it as a parent

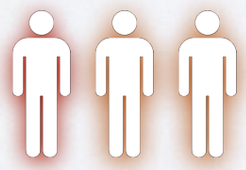
Parent 1

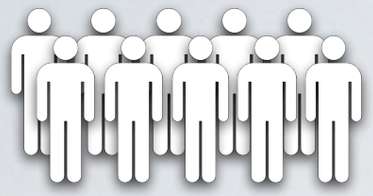
Parent 2

Crossover

Child

NEW POPULATION:



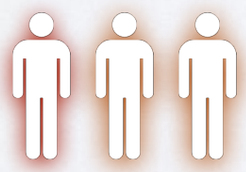


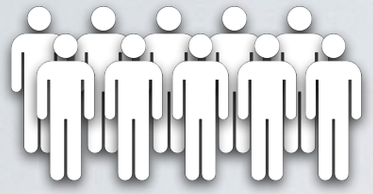
We now have to select two parents to create a child to be inserted into the new population. We generate again a random subgroup of individuals (20%) and choose the best out of it as a parent

Child

2-Opt

NEW POPULATION:



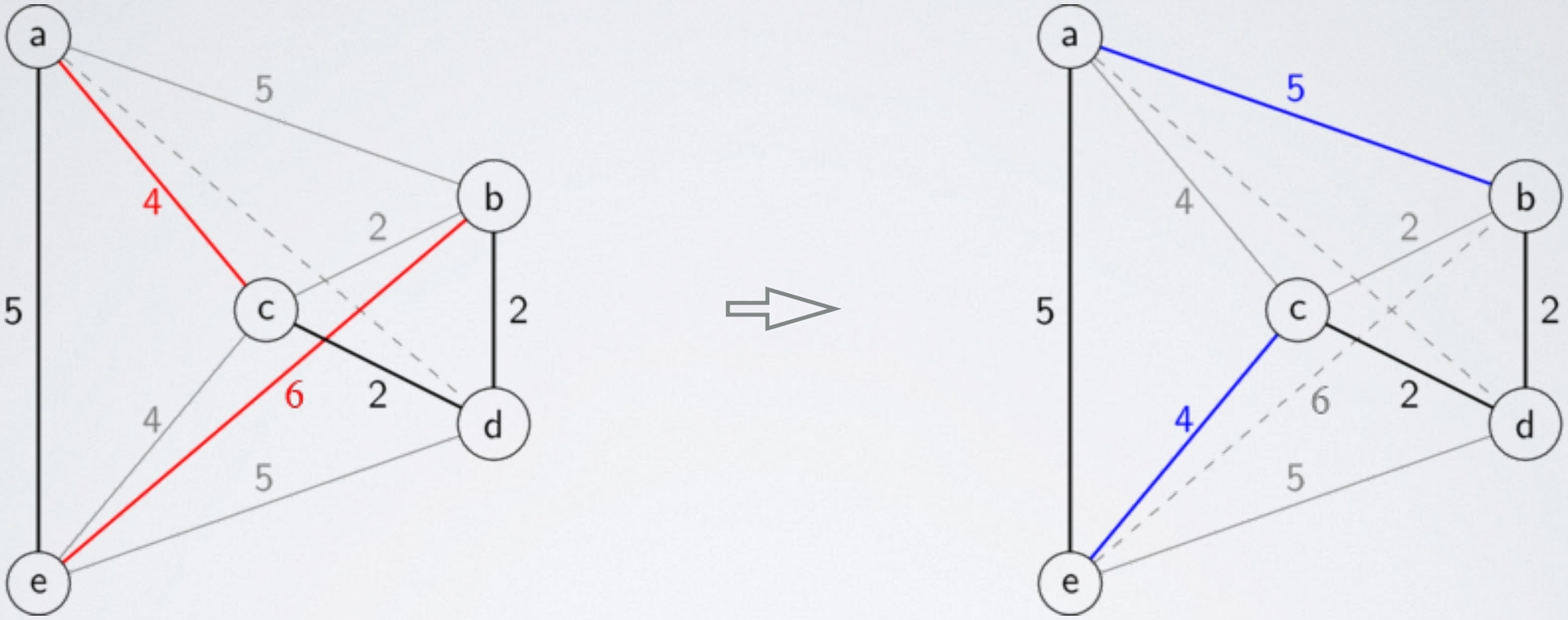


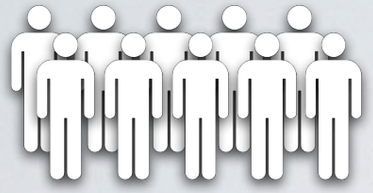
We now have to select two parents to create a child to be inserted into the new population. We generate again a random subgroup of individuals (20%) and choose the best out of it as a parent

2-Opt

NEW POPULATION:

2-Opt





2-Opt

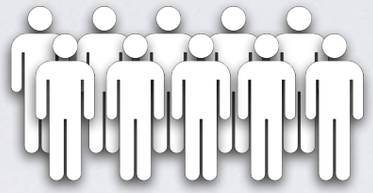
If the final individual is equal to another one already present into the new population, we discard the clone

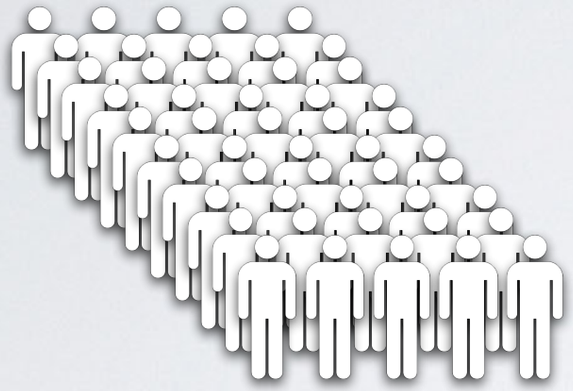
is done for a maximum of 15 times each child creation (in order to avoid eccessive slowdown in case of eccessive clones generation)

NEW POPULATION:

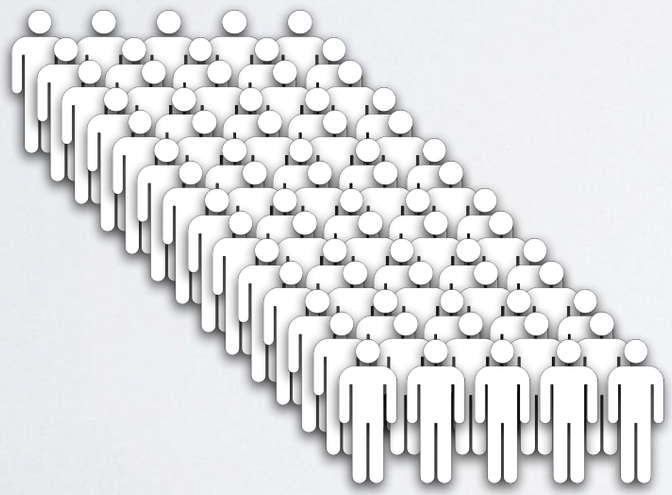
NEW POPULATION:

NEW POPULATION:



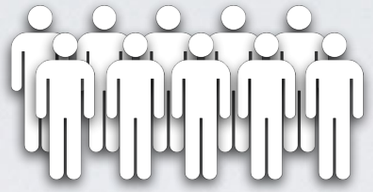


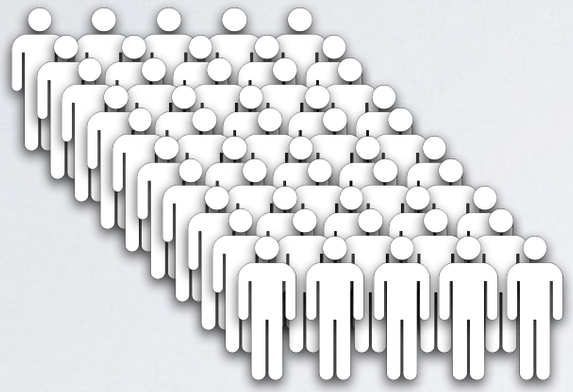
Every 5 evolutions we execute a mutation on each child before the execution of the 2-Opt

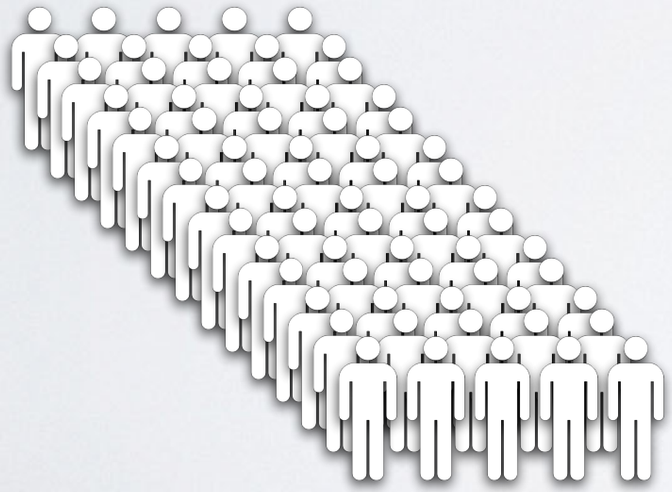


If our best individual doesn't change within 7 evolution, we increase the rate of mutations: once every 2 evolutions until we find a new best tour

NEW POPULATION:







Mutation

We swap two cities randomly (preserving the inner sub-tour).

For each city of each tour there is a default value of 0.8% of probability that a mutation occurs

Our work

Tabu

- Improvement of the function *MyGreedyStartSolution* by adopting a different approach: random selection of the **five nearest cities** instead of the closest one (thus increasing diversification).

Tabu + Genetic

- Creation of ad-hoc functions to **merge Tabu and Genetic**: in particular we needed to convert the format adopted to represent Tour and City inside the two libraries.

- When performing a new evolution, we tried to save into the new population the best individuals of the previous one. We tried to **save different percentage of the best previous individuals of the population** (up to 50% of the whole population). To do this we had to implement a function to compare and order the individuals according to their decreasing value of fitness. This approach brought scarse diversification and it was discarded (only the best individual is maintained).

- We tried to implement a various number of **crossover functions**:
  - Classic crossover (what we actually chose in the end);
  - We tried to fix the length of the sub-tour taken from the first parent (various lengths have been tried): the idea was to maintain a big part of the first parent and change it a little according to the sequence of the second parent. Anyway this method appeared to have a too small level of diversification;
- We tried to select two random segments instead of one from parent 1. This actually increased complexity without improvement;
- We tried a sequential crossover;
- We tried GSX Greedy Sub-Tour Crossover;

Our work

2-OPT

- We chose 2-OPT as our final algorithm. We initially tried to **combine genetic, tabù and 2-opt.** In particular we tried to pass to the 2-opt the child return by a tabù process. Actually this method was extremely slow and the behaviour (and results) of the 2-opt were not affected by the previous tabù.

- Later we worked with 2-opt and 3-opt with many different combinations and behaviours. We firstly tried to adopt **3-opt only** but it was too slow. We also tried to perform a **2-opt before the 3-opt** so that the 3-opt was faster (but not enough). After this we tried something smarter and after the 2-opt and we managed to divide the tour into sub-tours of 8 cities (we could set the length of the sub-tour); we then **adapted the 3-opt to work on the small sub-tours,** so that the general timing performances were not too low (in fact the cubic algorithm of the 3-opt were applied on paths of 8 cities only). After the recombination of the different paths the overall result might have improved thanks to the "local" 3-opt. We also tried the following: we set the 2-opt as default but we performed the **3-opt** (on the overall child) **only if it was already close to the temporary actual best of the population** (attempting to improve only those solution already close to the actual best). All these ideas brought to actual improvement in terms of best distances, but the timing was always too large to permit the adoption of these algorithms.

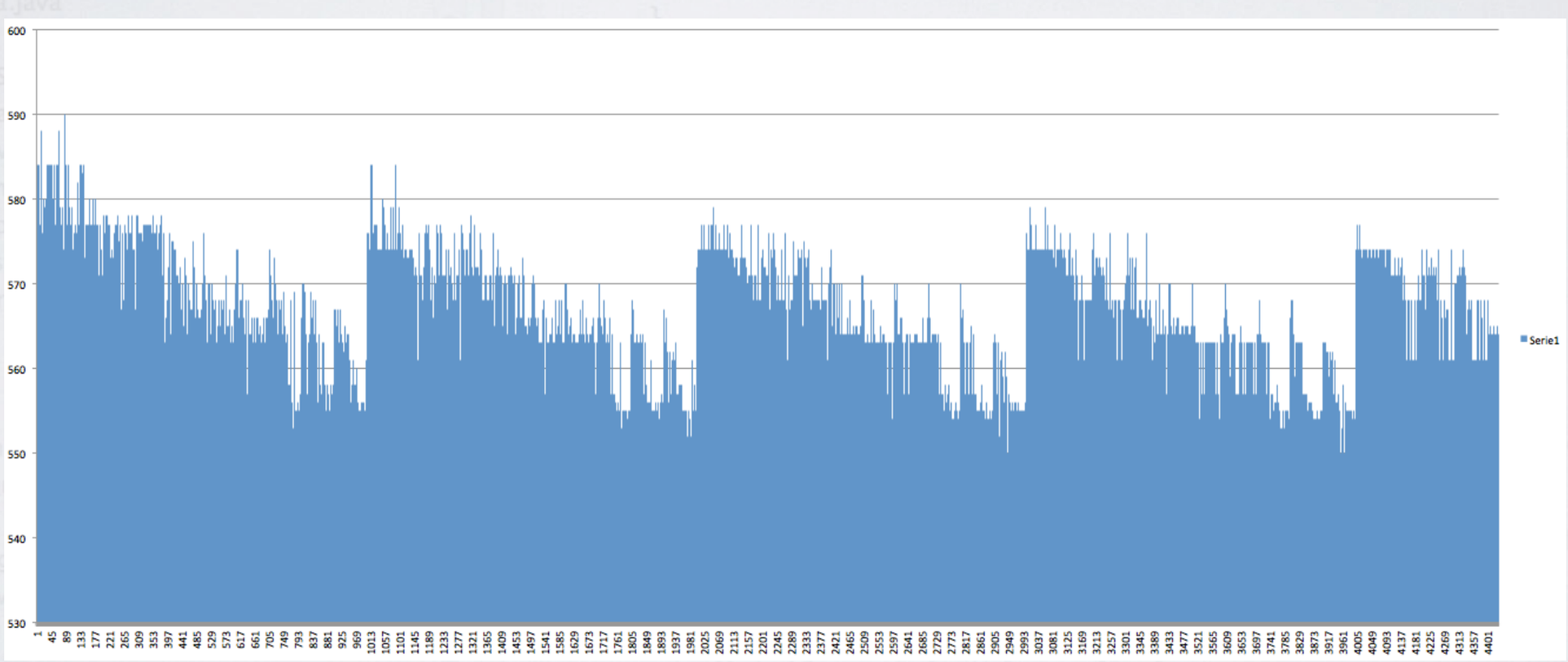
- We realised that the probability of having very similar children during our evolutions was very high (almost independently from the single parameters), so we opted to **manage the clones:** we used the function previously written to compare tours and checked every time if a clone was to be inserted into the population. If this was the case we tried to perform a mutation on it and we also tried to substitute it with a total random tour. Later we decided to manage the clones by simply trying to generate new children (max. 15 times, otherwise the clone is NOT inserted at all and we reduce the actual population size).

Our work

2-OPT

- We also worked on the **mutation function**. Actually we didn't change it very much but we tested statically which was the best value of mutate probability (0.8% for each node in each tour) and the frequency of execution. Eventually we set the values as described in the summary table.

- All the parameters that we set as default value, have been selected through statistical tests performed automatically through **recursive functions** capable of checking various ranges of values with all the possible combination (example of a graphical output is reported below);



Our parameters

- **Population** size: **50** (0-500 cities), **30** (>500 cities);
- **Evolutions**: **30** (0-160 cities), **150** (160-500 cities), **75** (>500 cities);
- **Mutate rate** (standard): **5** evolutions; mutate rate (fast): **2** evolutions;
- Number of not improving evolutions (**maxAge**) to fasten the mutate rate: **7**;
- **Mutate probability**: **0.8%**;
- **Tournament Size**: **20%** (of the total population size);
- Maximum number of **clone reprocessing** per child: **15**;
- Total number of **iterations** per instance: **10**;

Population and Evolutions parameters has been chosen to better perform the algorithm according to the instance dimension and also to be consistent with the time constraints imposed by the project guideline

Our results

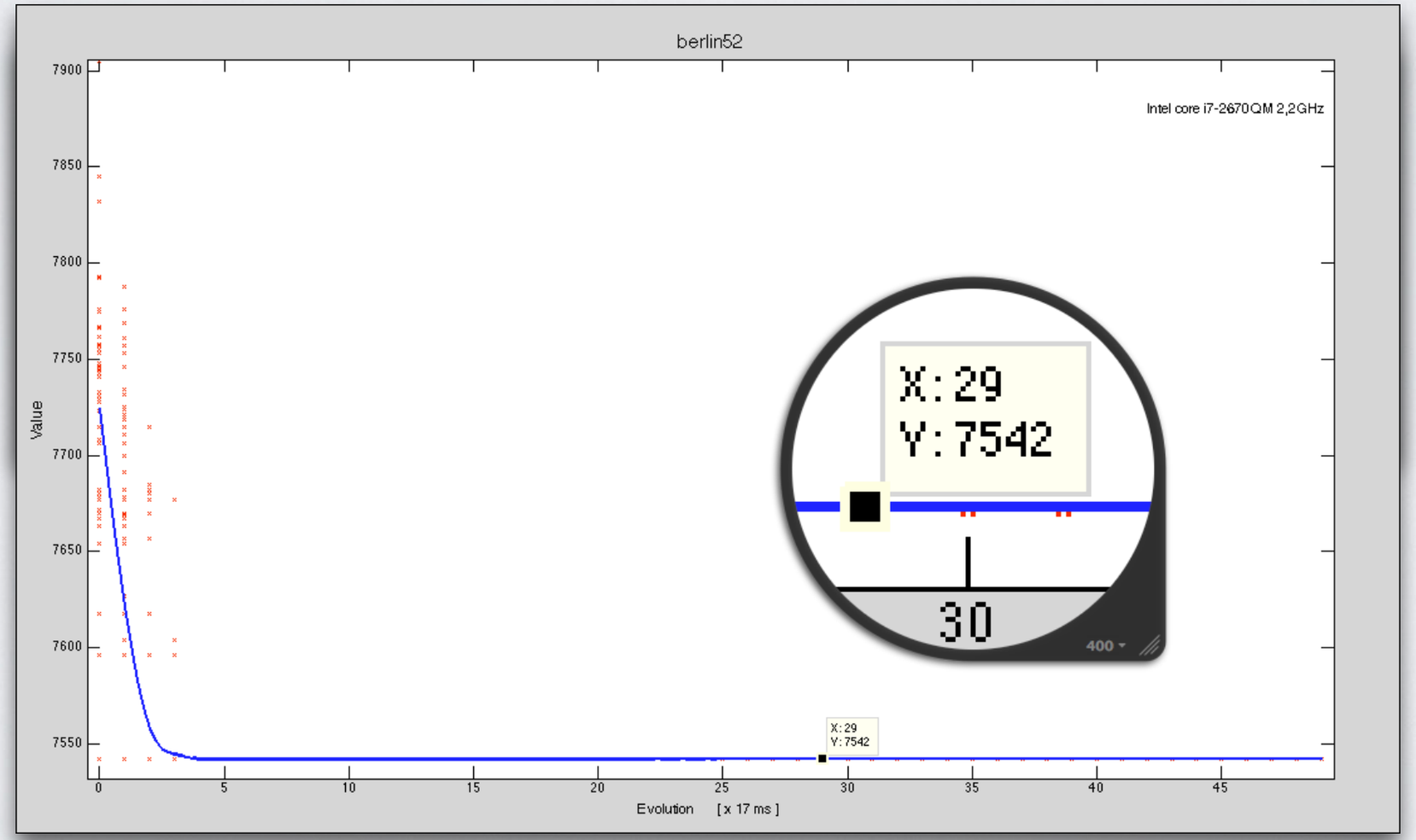
Instance	Mean sol	Min sol	Max sol	Time Best	Time mean	Best known
berlin52	7542	7542	7542	17 ms	510 ms	7542
eil51	426,5	426	427	28 ms	420 ms	426
eli76	538,1	538	540	99 ms	990 ms	538
pr152	73736,4	73682	73818	465 ms	4,65 s	73682
pr1002	266100	263800	268700	4,9 m	4,9 m	259045
rat195	2334	2323	2348	10.3 s	39,7 s	2323

Mean		
Istanze	**Best (%)**	**Time [ms]**
berlin52	0,00	510
eil51	0,12	420
eli76	0,02	990
pr152	0,07	4650
pr1002	2,72	294000
rat195	0,47	39700
Mean	**0,57**	**56711,7**

Best		
Istanze	**Best (%)**	**Time [ms]**
berlin52	0,00	17
eil51	0,00	28
eil76	0,00	99
pr152	0,00	465
pr1002	1,84	294000
rat195	0,00	10300
Mean	**0,31**	**50818,2**

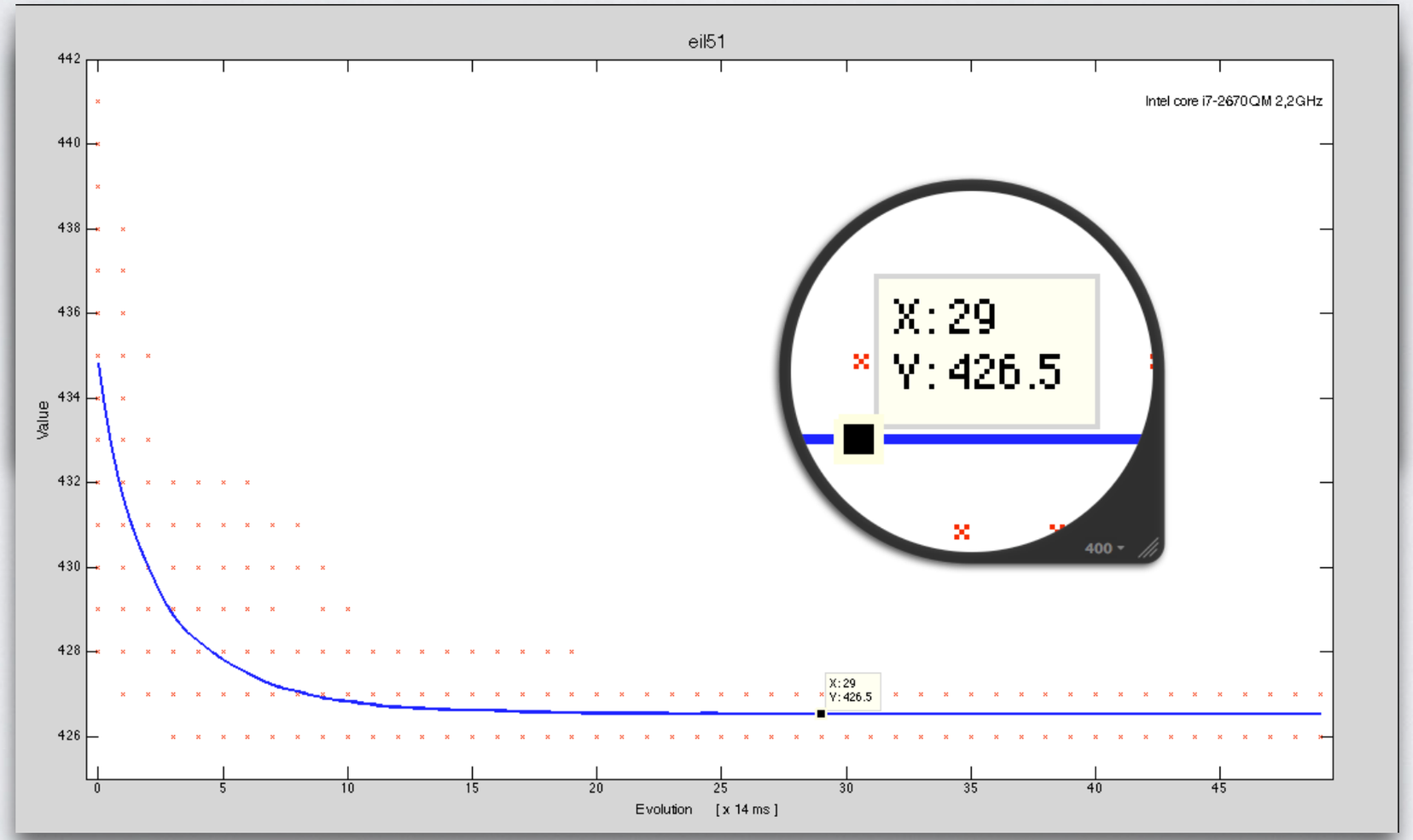
Our results

Instance	Mean sol	Min sol	Max sol	Time Best	Time mean	Best known
berlin52	7542	7542	7542	17 ms	510 ms	7542
eil51	426,5	426	427	28 ms	420 ms	426
eli76	538,1	538	540	99 ms	990 ms	538
pr152	73736,4	73682	73818	465 ms	4,65 s	73682
pr1002	266100	263800	268700	4,9 m	4,9 m	259045
rat195	2334	2323	2348	10.3 s	39,7 s	2323



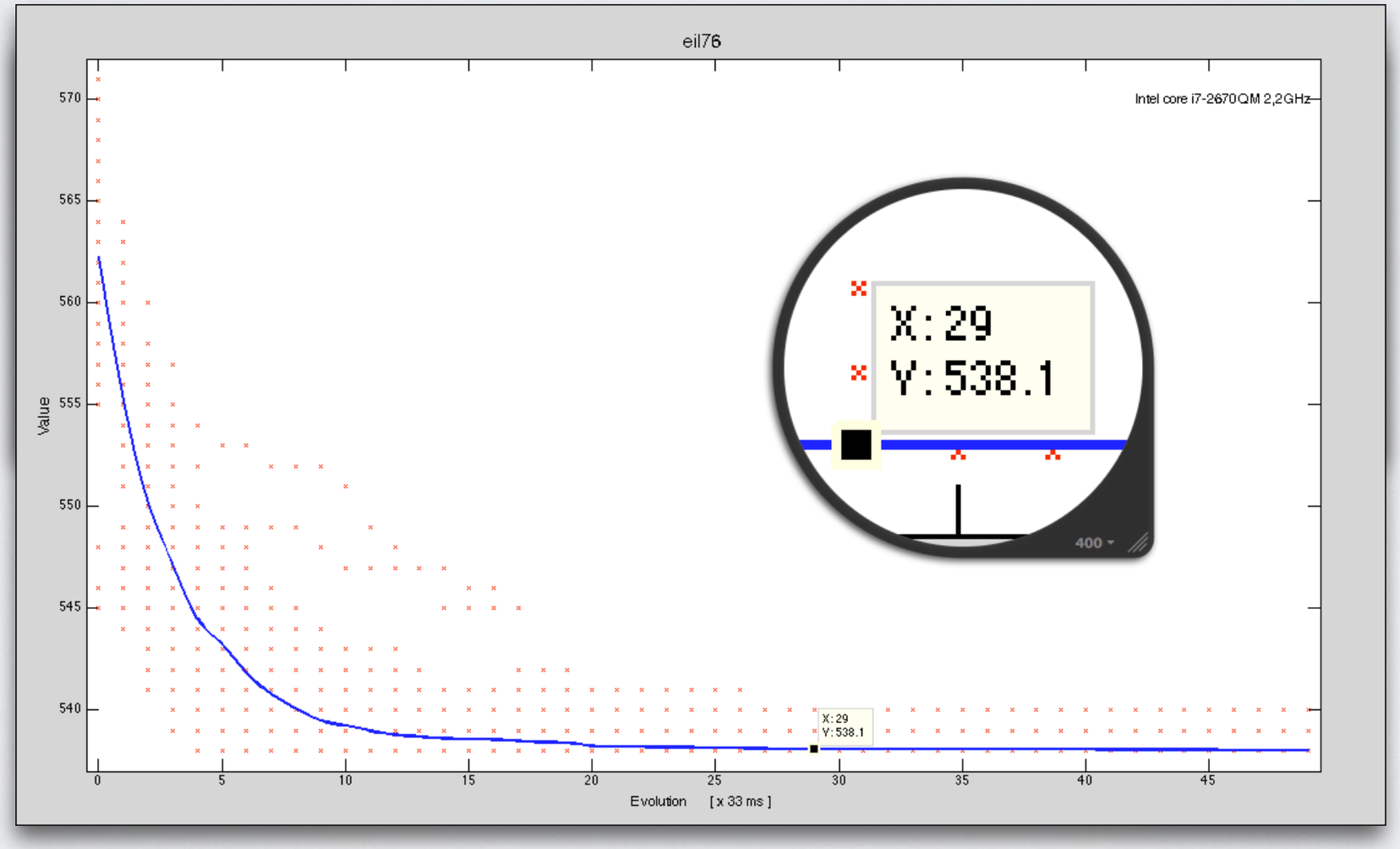
Our results

Instance	Mean sol	Min sol	Max sol	Time Best	Time mean	Best known
berlin52	7542	7542	7542	17 ms	510 ms	7542
eil51	426,5	426	427	28 ms	420 ms	426
eli76	538,1	538	540	99 ms	990 ms	538
pr152	73736,4	73682	73818	465 ms	4,65 s	73682
pr1002	266100	263800	268700	4,9 m	4,9 m	259045
rat195	2334	2323	2348	10.3 s	39,7 s	2323



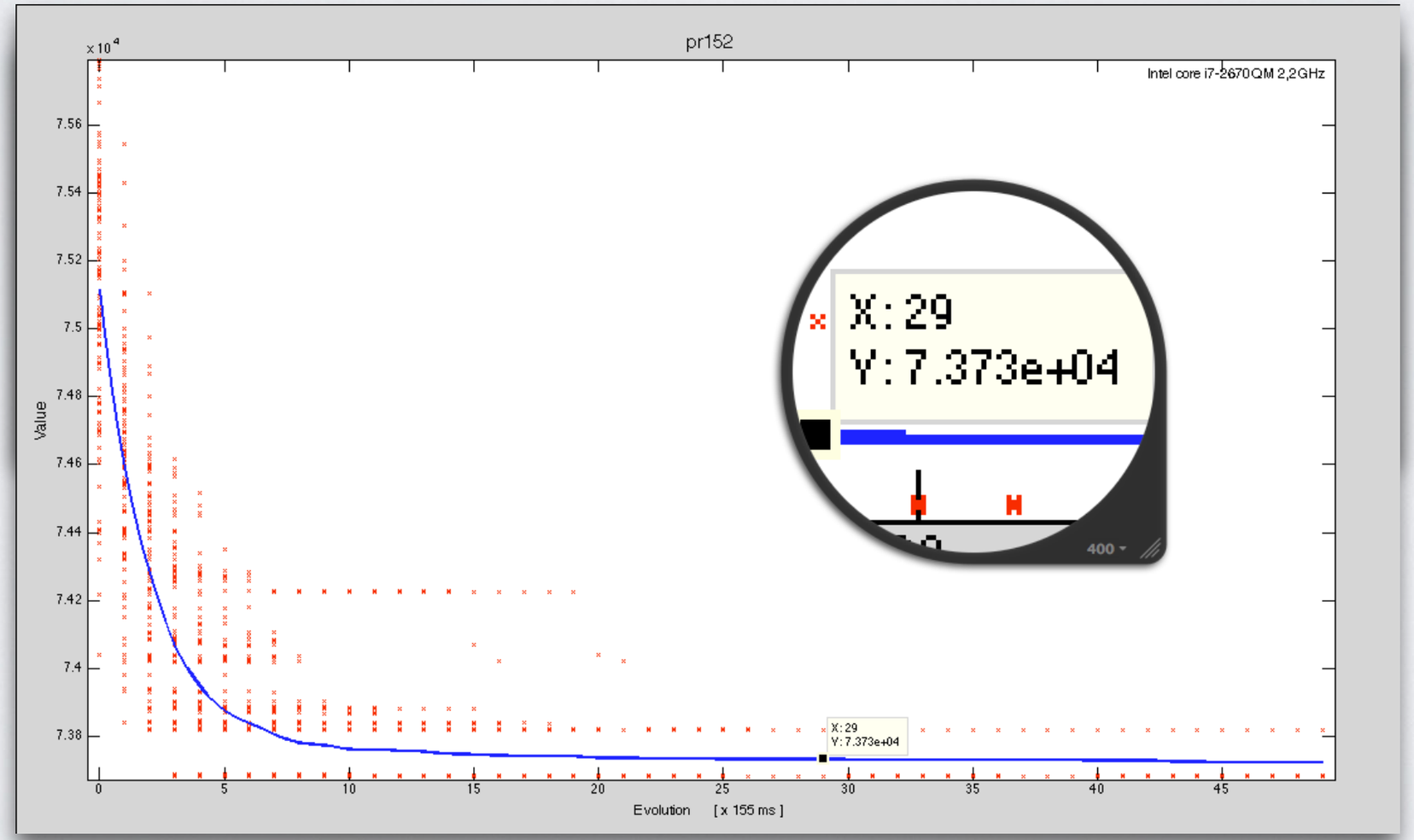
Our results

Instance	Mean sol	Min sol	Max sol	Time Best	Time mean	Best known
berlin52	7542	7542	7542	17 ms	510 ms	7542
eil51	426,5	426	427	28 ms	420 ms	426
eli76	538,1	538	540	99 ms	990 ms	538
pr152	73736,4	73682	73818	465 ms	4,65 s	73682
pr1002	266100	263800	268700	4,9 m	4,9 m	259045
rat195	2334	2323	2348	10.3 s	39,7 s	2323



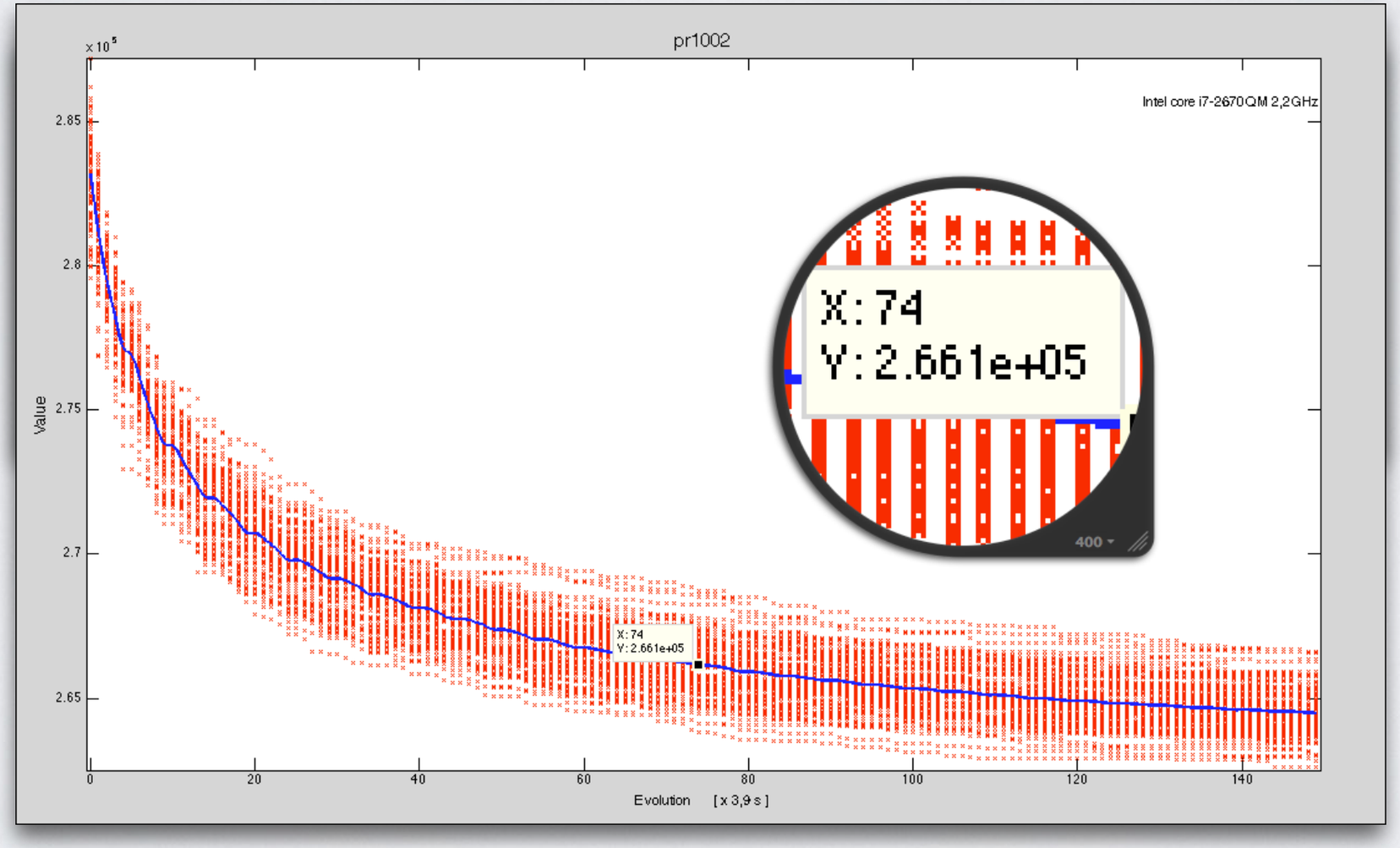
Our results

Instance	Mean sol	Min sol	Max sol	Time Best	Time mean	Best known
berlin52	7542	7542	7542	17 ms	510 ms	7542
eil51	426,5	426	427	28 ms	420 ms	426
eli76	538,1	538	540	99 ms	990 ms	538
pr152	73736,4	73682	73818	465 ms	4,65 s	73682
pr1002	266100	263800	268700	4,9 m	4,9 m	259045
rat195	2334	2323	2348	10.3 s	39,7 s	2323



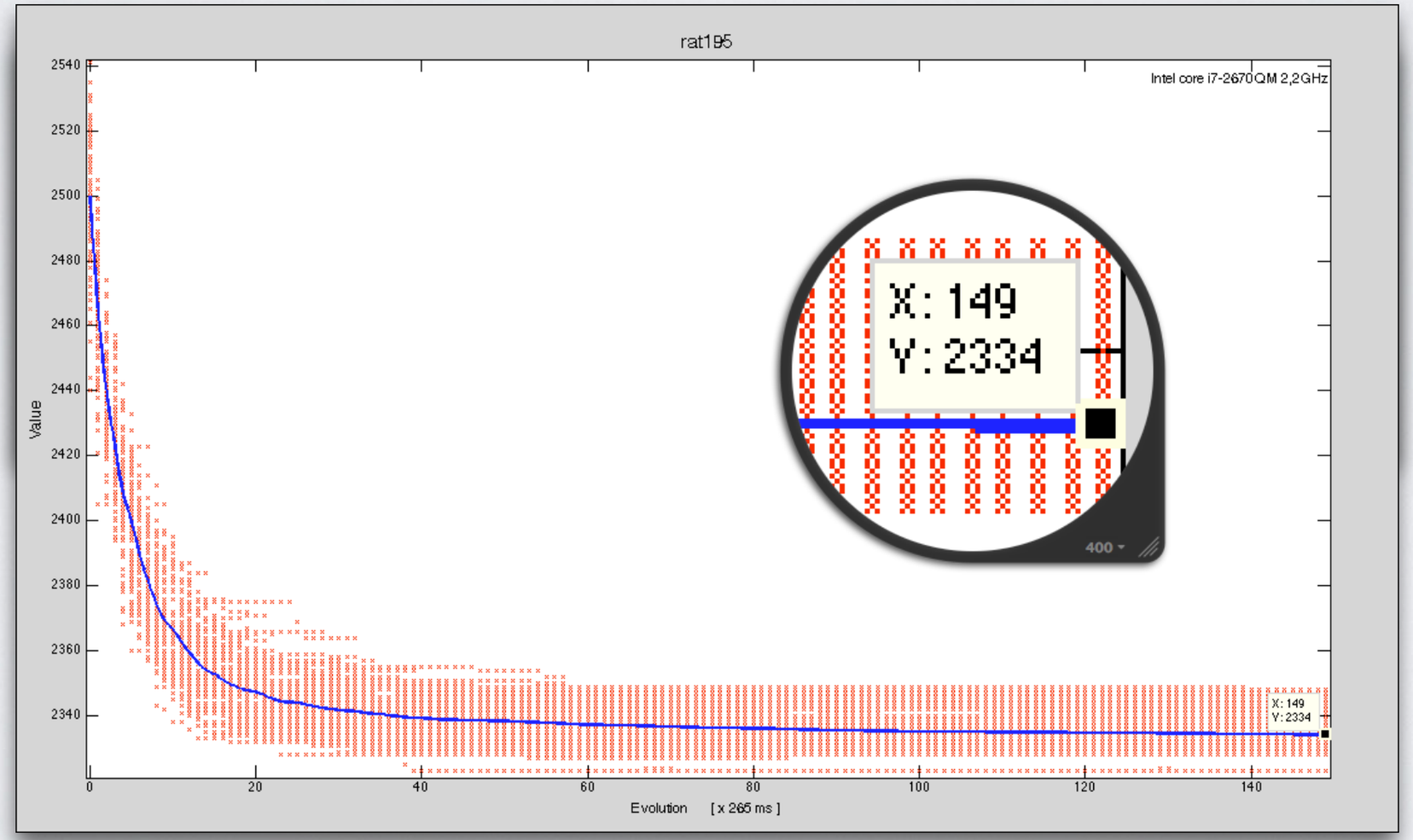
Our results

Instance	Mean sol	Min sol	Max sol	Time Best	Time mean	Best known
berlin52	7542	7542	7542	17 ms	510 ms	7542
eil51	426,5	426	427	28 ms	420 ms	426
eli76	538,1	538	540	99 ms	990 ms	538
pr152	73736,4	73682	73818	465 ms	4,65 s	73682
pr1002	266100	263800	268700	4,9 m	4,9 m	259045
rat195	2334	2323	2348	10.3 s	39,7 s	2323



Our results

Instance	Mean sol	Min sol	Max sol	Time Best	Time mean	Best known
berlin52	7542	7542	7542	17 ms	510 ms	7542
eil51	426,5	426	427	28 ms	420 ms	426
eli76	538,1	538	540	99 ms	990 ms	538
pr152	73736,4	73682	73818	465 ms	4,65 s	73682
pr1002	266100	263800	268700	4,9 m	4,9 m	259045
rat195	2334	2323	2348	10.3 s	39,7 s	2323



Our results

Instance	Mean sol	Min sol	Max sol	Time Best	Time mean	Best known
berlin52	7542	7542	7542	17 ms	510 ms	7542
eil51	426,5	426	427	28 ms	420 ms	426
eli76	538,1	538	540	99 ms	990 ms	538
pr152	73736,4	73682	73818	465 ms	4,65 s	73682
pr1002	266100	263800	268700	4,9 m	4,9 m	259045
rat195	2334	2323	2348	10.3 s	39,7 s	2323

Mean		
Istanze	**Best (%)**	**Time [ms]**
berlin52	0,00	510
eil51	0,12	420
eli76	0,02	990
pr152	0,07	4650
pr1002	2,72	294000
rat195	0,47	39700
Mean	**0,57**	**56711,7**

Best		
Istanze	**Best (%)**	**Time [ms]**
berlin52	0,00	17
eil51	0,00	28
eil76	0,00	99
pr152	0,00	465
pr1002	1,84	294000
rat195	0,00	10300
Mean	**0,31**	**50818,2**