IFT6751: Homework 3

Gabriel C-Parent

April 12, 2015

1 Introduction

In this homework, an Elastic Net algorithm is used to solve the traveling salesperson problem.

1.1 The TSP

The Traveling Salesperson Problem (TSP) is a landmark problem in combinatorial optimization and an hard one (NP-Hard)[4].

The basic idea is that a salesperson needs to travel to a set of cities and come back to its original city in the shortest time (distance) possible. This corresponds to finding an Hamiltonian cycle of minimal distance.

There exists a number of variants of the classical statement such as the metric TSP, euclidean TSP and the asymmetric TSP.

The TSP has been studied extensively and many approaches are available to solve it ranging from heuristics to exact methods. A variant of the problem was even solved with DNA computing [1]. The problem can be formulated as an integer linear program so that classical mathematical programming tools can be used.

1.2 Elastic Net

The Elastic Net is a geometrical approach to the TSP [2]. It consists of having a ring of neurons initialized in the center of the cities which is then elongated (think of a rubber band) by minimizing the length of the band and the distance of the band points to cities for a set number of iterations. Eventually, the band stabilizes and a TSP solution is inferred by assigning cities to the nearest neuron until no city is left unassigned.

The implementation used here is quite naive but some details must be mentioned. First, every problem instance is transposed onto the unit square to avoid underflow errors (exponentiation of large negative numbers is involved).

Second, the update rule used here is the same as in [2], that is the parameter K is set to 0.2 and is decreased by 1% every 25 iterations until 2500 iterations are reached.
Also, an additional 2-opt descent optimization was performed on the Elastic Net solutions, in case some edges cross.

1.3 Evaluation

From the evaluation standpoint, two relatively easy measures are available: using known problems or generating new ones as in [2].

1.3.1 Baseline

To make the evaluation more interesting, a simple solver based on the generation and improvement (2-opt descent) of **5** random permutations was used as a baseline. A description of the 2-opt descent was given in homework 2, it is exactly the same algorithm used here.

1.3.2 TSPLIB instances

The first way of evaluating the quality of generated solutions is to use previously solved problem instances. To do so, the 2D euclidean TSP instances ("EUC_2D") from TSPLIB [6] are used. Interestingly, to avoid floating-point precision problems, the total distance are calculated by rounding the distances between cities to the nearest integer. The distance calculation was confirmed to work well with a total length of 221440 on the sequential traversal (from 1 to 442) of tsp442.

1.3.3 Random Problems

Another way to evaluate the performance of our solver is to simply generate random new instances of problems with fixed number of clients (in this case 50, 100, 200) distributed uniformly through the unit square.
For each problem size, the best distance over 5 different randomly generated instances is reported [2]. Obviously, this is done using floating-point distance calculation.

2 Experimental Results

2.1 TSPLIB Instances

First of all, the baseline method (monte carlo with 2-opt descent) and the Elastic Net were used to solve some of the 2D euclidean instances from TSPLIB. The instances provided with the homework weren't used as only the optimal distance was given, not the path (but it seems tsp51 corresponds to eil51, tsp76 to eil76, tsp100 to kroD100).

2.1.1 Monte Carlo

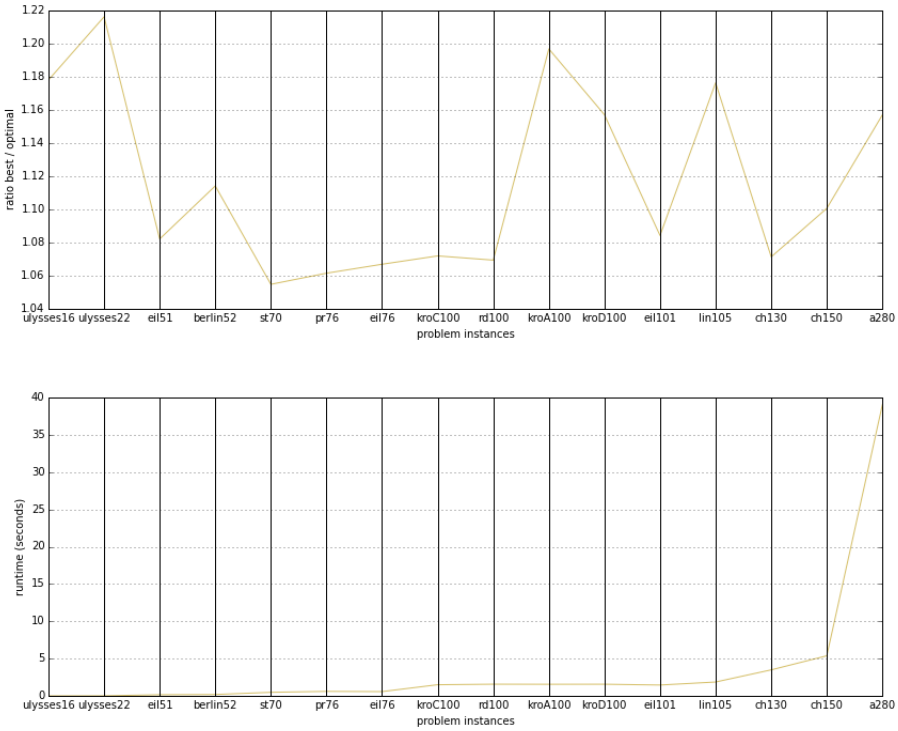


Figure 1: Distance ratio and runtime of obtained solution against optimal known solution. The problems are arranged by the number of clients to visit in increasing order from left to right. We can see that the quality of the solutions is quite good, especially for a baseline and the runtime isn't so bad (considering 5 different solutions were generated for each problem).

2.1.2 Elastic Net

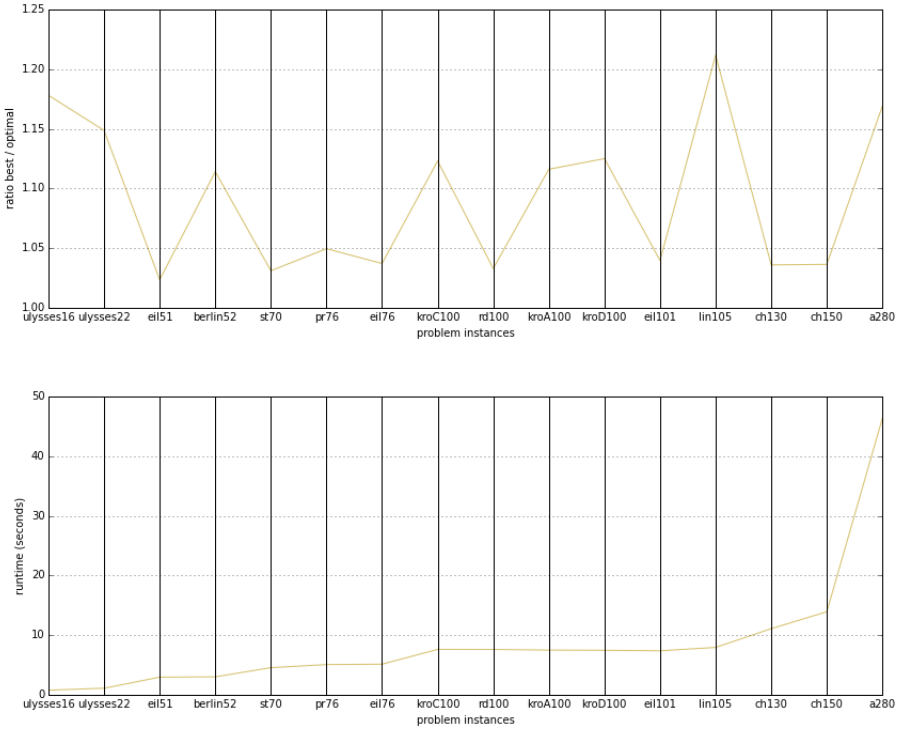


Figure 2: Distance ratio and runtime of obtained solution against optimal known solution. The problems are arranged by the number of clients to visit in increasing order from left to right. We can see that the number of clients doesn't seem to influence the relative performance. The results are quite good, going from 2% to 18% of the optimal values. Full solutions can be visualized in section 5.2.

2.2 Randomly Generated Instances

Here are the best results obtained for randomly generated problems of size 50, 100 and 200.

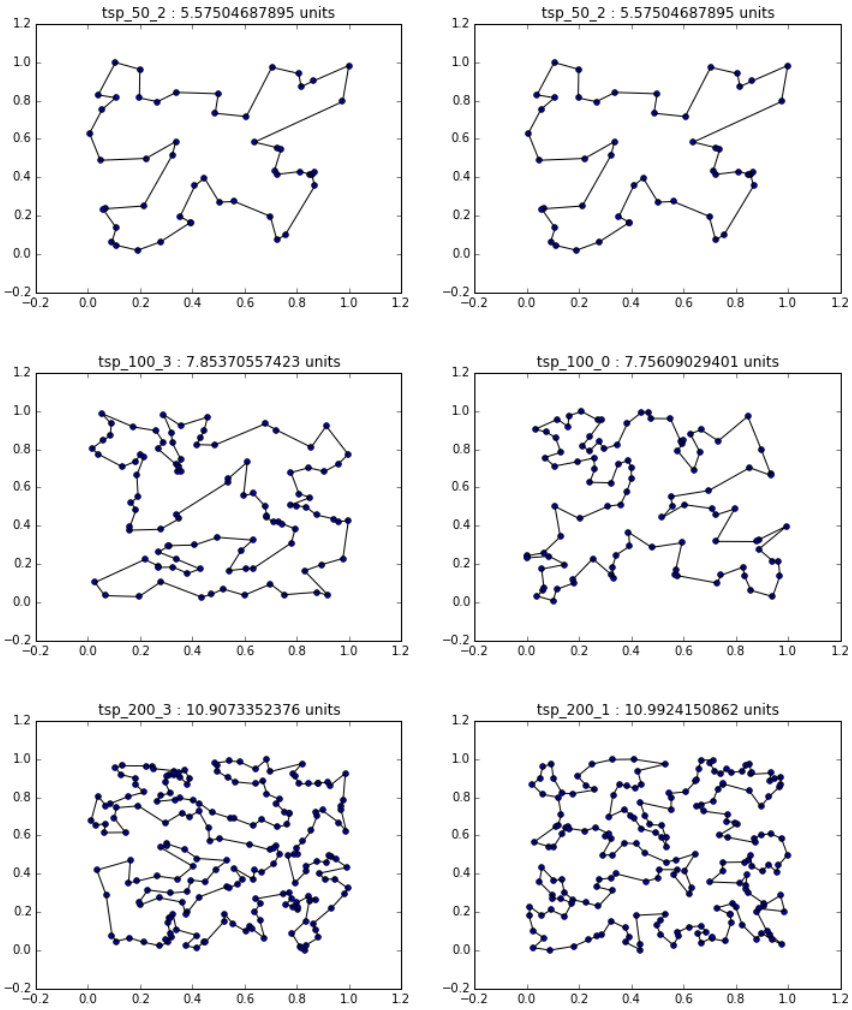


Figure 3: Best solutions to (five different) randomly generated 2D euclidean TSP instances of size 50, 100 and 200. On the left is the baseline and on the right the Elastic Net. Ironically, the baseline is better than the Elastic Net on some instances... Compared to the results from [5], the Elastic Net results are similar for the Elastic Net (5.62, 7.69 and 11.14) and still much better than the Peterson and Soderberg model results (6.61, 8.58, 12.66).

3 Discussion

3.1 Challenges

3.1.1 Implementation

The Elastic Net was implemented in a sequential fashion using Cython and Numpy as in the previous homework. All the code is in the IPython Notebook referred in section 5.1.

The only real implementation detail that was problematic was to avoid exponentiating big negative numbers, that is I initially forgot to transpose the TSP instances in the unit square...

3.1.2 Parameter Sensitivity

Unsurprisingly, the algorithm is quite sensitive to the parameters used. The basic parameters were the following.

- $M = 2.5\text{N}$
- radius of initial net = 0.01
- $\alpha = 0.2$
- $\beta = 2.0$
- $K = 0.2$
- K is decreased 1% every 25 iterations
- lowest $K = 0.01$
- 2500 iterations

The Elastic Net is quite sensitive to most of these parameters. It is certainly very sensitive to the ratio of the two forces involved in the update of the network (α, β and K), as it drives the net towards a minima.

I didn't play too much with the parameters, but it seems that choosing the decay of K is quite important while the number of iterations doesn't make much of a difference as long as the net has converged a bit. An additional effort could have been made to use convergence as a stopping criteria, but that would imply a risk of prematurely stopping.

3.2 Performance Analysis

First of all, the baseline method was surprisingly good. The major computational cost comes from the 2-opt descent which is $O(n^2)$ on the size of the problem instance, but it still manages to finish really fast. Its speed and quality of solutions was much better than expected (figs. 1 and 3).

The Elastic Net algorithm was also very fast and should be much faster given a parallel implementation. Also, since the choice of parameters is important, it isn't clear that a bigger number of iterations with slower decay of K is better (the opposite was actually observed on some instances).
The performance on TSPLIB instances is generally better than the baseline (fig. 2) and comes close to optimal for some instances. The quality of the solutions for randomly generated instances (fig. 3) is also good and very similar to the expected values from [5].

4 Conclusion

The Elastic Net approach to the TSP is an interesting idea with some nice theoretical guarantees [3]. It seems generally reliable and efficient and I find it more intuitive than the self-organizing map.

I am still surprised that the baseline method yielded such good results given its simplicity, but I believe that it doesn't have the potential to scale as well as the Elastic Net should.

References

[1] LM Adleman. Molecular computation of solutions to combinatorial problems. *Science*, 266, 1994.

[2] R Durbin and D Willshaw. An analogue approach to the traveling salesman problem using an elastic net method. *Nature*, 326(16), 1987.

[3] Richard Durbin, Richard Szeliski, and Alan Yuille. An analysis of the elastic net approach to the traveling salesman problem. *Neural Comput.*, 1(3):348–358, September 1989.

[4] C H Papadimitriou. The euclidean tsp is np-complete. *Theoretical Computer Science*, 4, 1977.

[5] J-Y Potvin. The traveling salesman problem: A neural network perspective. *ORSA Journal on Computing*, 5:328–348, 1993.

[6] G. Reinelt. Tsplib– a traveling salesman problem library. *ORSA J. Comput.*, 3:376–384, 1991.

5 Supplementary Materials

5.1 User Guide

The following section should help with verification of the results and repeatability.

The language used is a mix of python and cython, an optimising compiler that allows static typing and generates C code.

5.1.1 Working Environment

All computational results obtained in this work should be repeatable given a suitable python environment. The particular dependencies of this work are the Cython, Numpy, IPython and Seaborn along with standard python environment.

The following python environment was used:

```
CPython 2.7.9
ipython 2.2.0

numpy 1.9.2
cython 0.21
ipython 2.2.0

compiler   : GCC 4.4.7 20120313 (Red Hat 4.4.7-1)
system     : Linux
release    : 3.13.0-46-generic
machine    : x86_64
processor  : x86_64
CPU cores  : 4
interpreter: 64bit
```

As of now, my personal recommendation is to use the excellent Anaconda python distribution from Continuum Analytics.

5.1.2 Running Computational Results

All computational results and figures are contained in the form of IPython Notebooks with the hope of allowing repeatability and reproducibility.

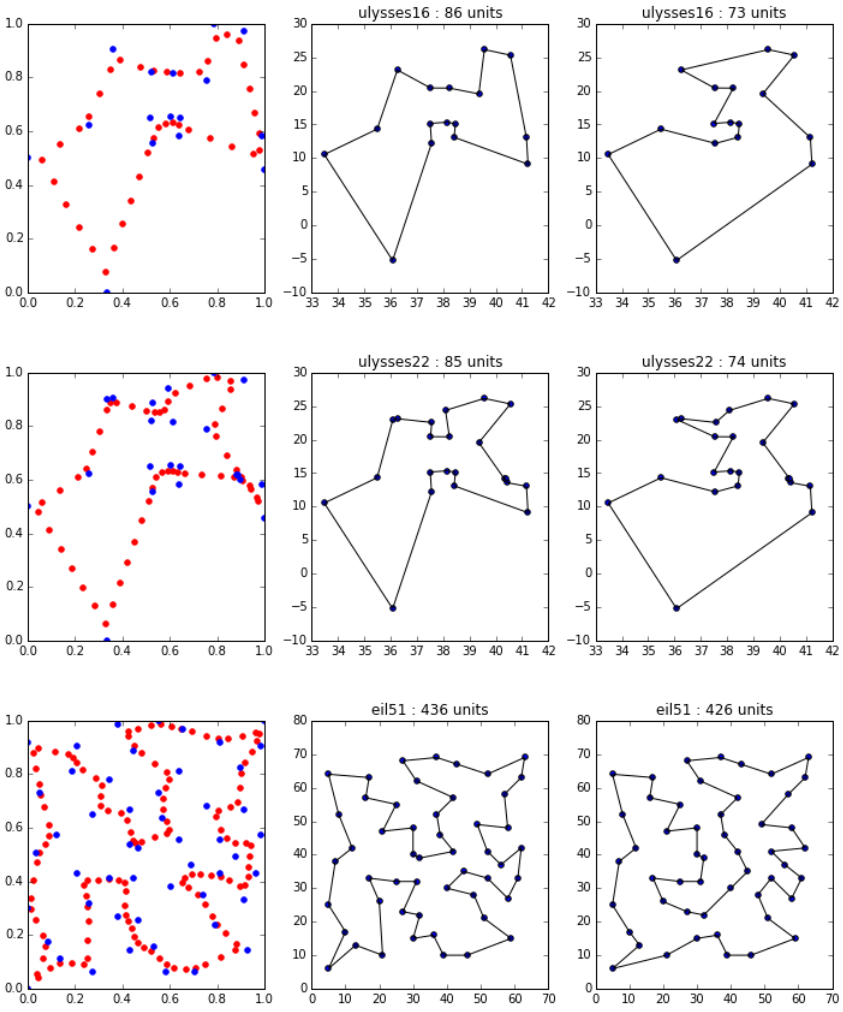
If IPython is available on the computer, an IPython Notebook service can be launched from command line by typing "ipython notebook" from the proper directory.

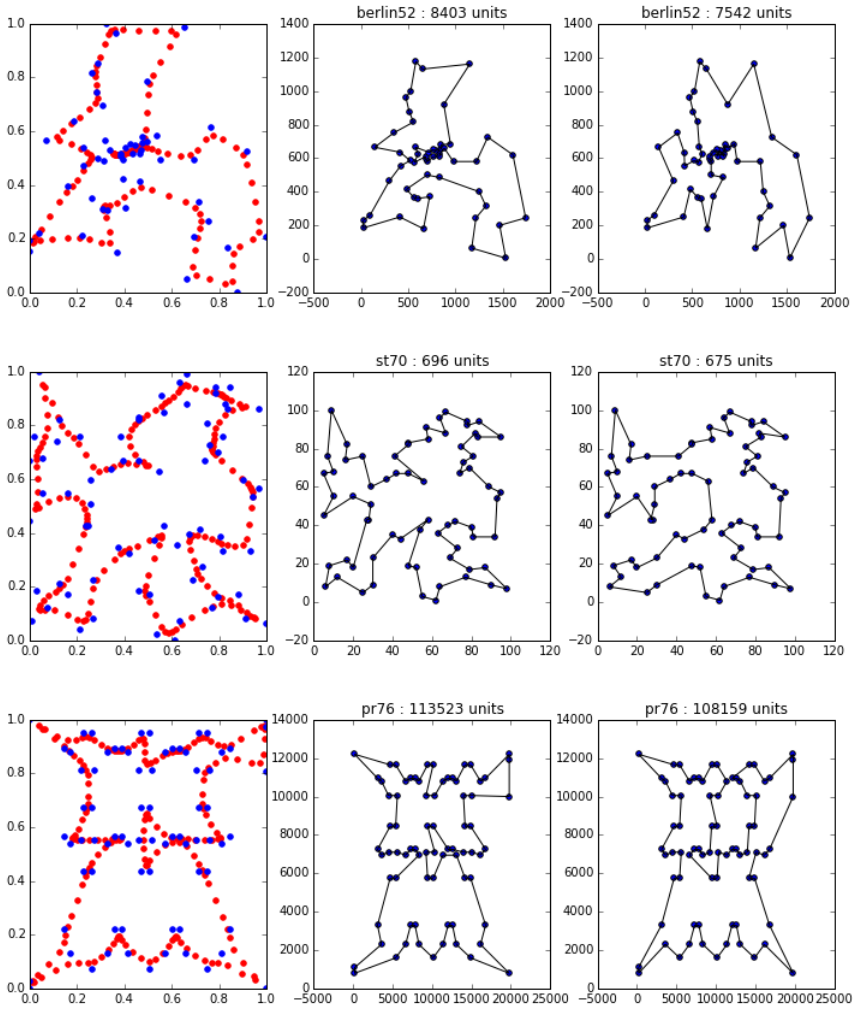
Alternatively, the IPython notebooks can be viewed online if it is reachable from a url using the nbviewer tool. This allows viewing a static version of an IPython Notebook.

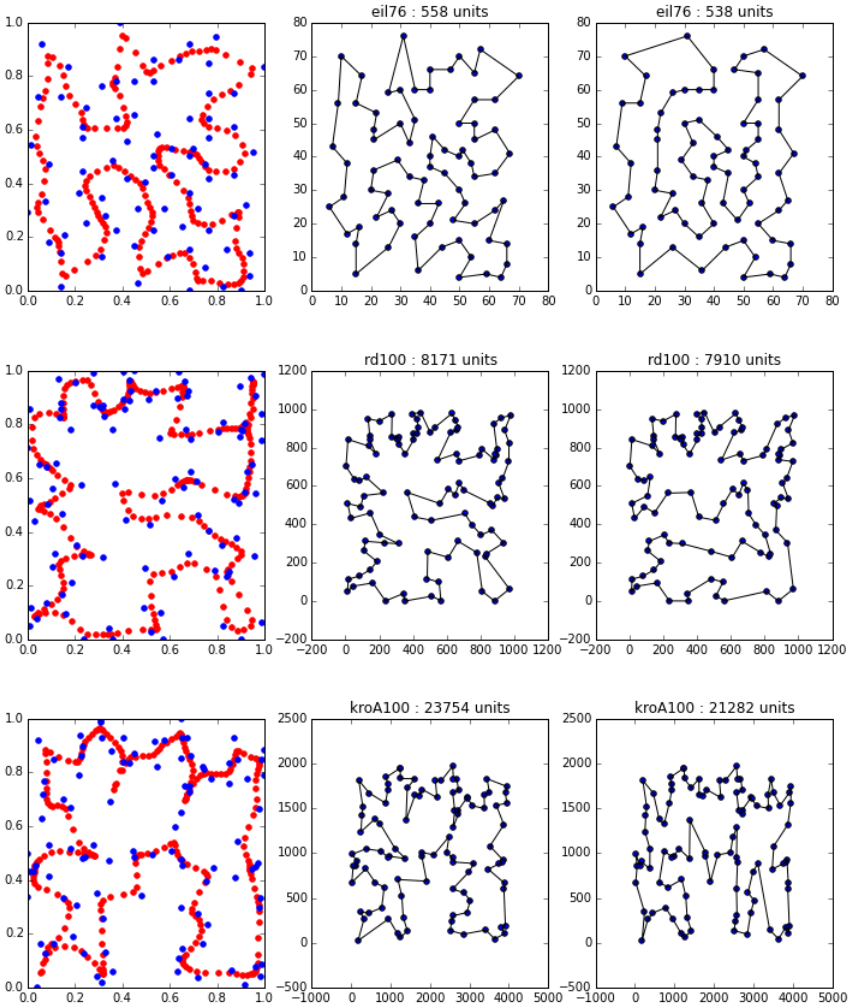
5.2 TSPLIB Solutions

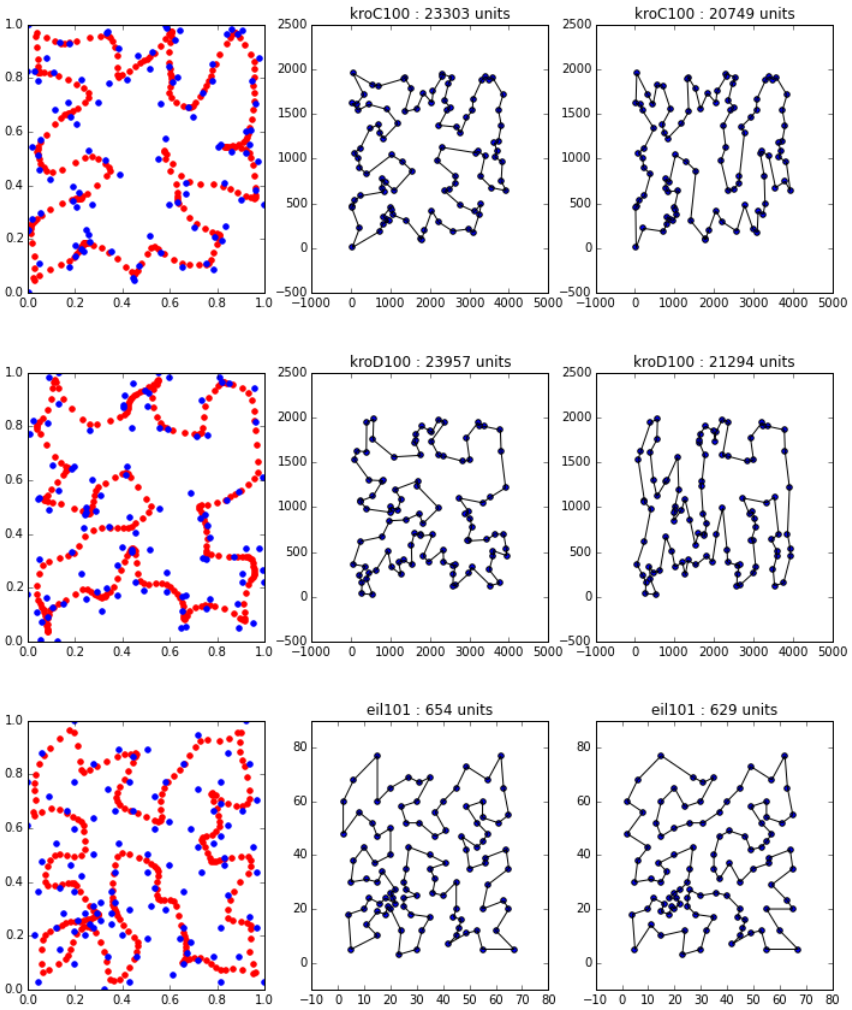
For each of the figures below, the Elastic Net, the corresponding path and the best path obtained from TSPLIB are arranged from left to right.

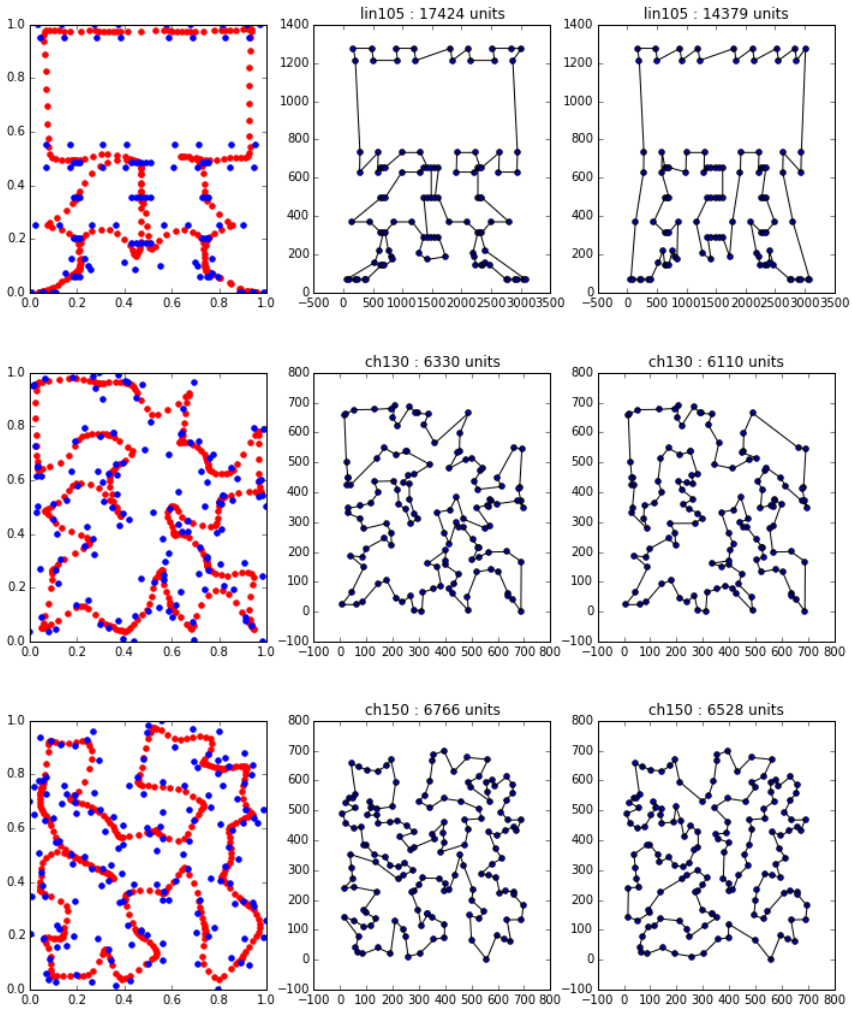
For more details, all parameters used are specified in the Homework3.ipynb IPython Notebook.




berlin52 : 8403 units
berlin52 : 7542 units
st70 : 696 units
st70 : 675 units
pr76 : 113523 units
pr76 : 108159 units


eil76 : 558 units
eil76 : 538 units
rd100 : 8171 units
rd100 : 7910 units
kroA100 : 23754 units
kroA100 : 21282 units


kroC100 : 23303 units
kroC100 : 20749 units
kroD100 : 23957 units
kroD100 : 21294 units
eil101 : 654 units
eil101 : 629 units


lin105 : 17424 units
lin105 : 14379 units
ch130 : 6330 units
ch130 : 6110 units
ch150 : 6766 units
ch150 : 6528 units