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.

1.2 Elastic Net

The TSP has been studied extensively and many approaches are available to solve it ranging from heuristics to exact methods. 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.

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 rubber band stabilizes and a traversal is extracted by assigning cities to the nearest neuron until no city is left unassigned.

The implementation used here is quite naive and some specifics must be documented. First, every problem instance is transposed onto the unit square to avoid underflow errors. 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.

Also, an additional 2-opt descent optimization was performed on the solutions.

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 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.

1.3.2 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 Elastic Net was used to solve some of the 2D euclidean instances from TSPLIB.

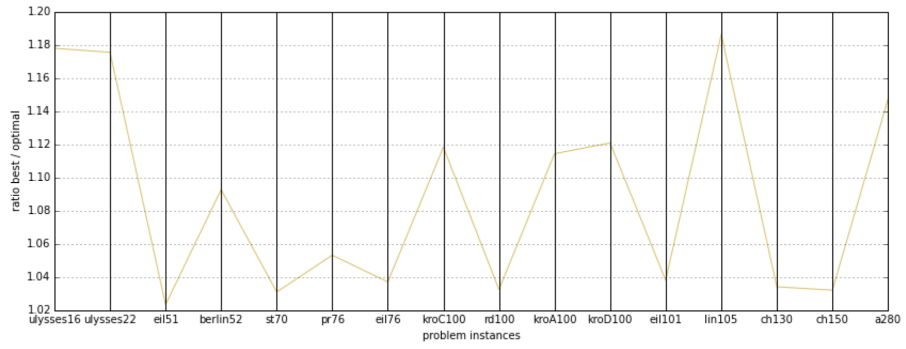


Figure 1: Distance ratio of obtained solution against optimal known solution. The problems are arranged by the number of clients to visit in increasing order from left to right. We can see that the number of clients doesn't seem to influence the relative performance. Full solutions can be visualized in section 5.2.

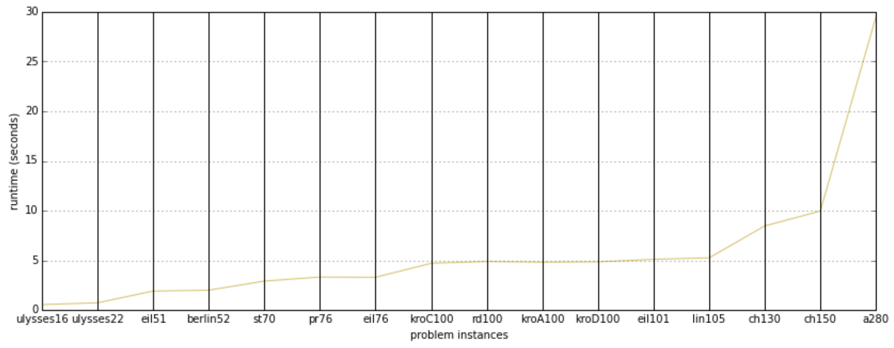


Figure 2: Runtime measured in seconds for the different problem instances. The problems are arranged by the number of clients to visit in increasing order from left to right. We can see that the runtime grows with the number of clients (as expected).

2.2 Randomly Generated Instances

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

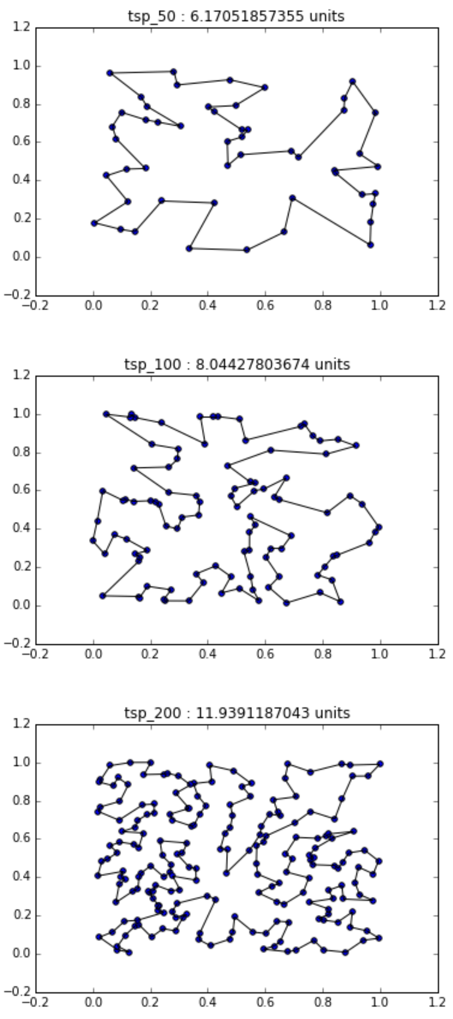


Figure 3: Best solutions to (five different) randomly generated 2D euclidean TSP instances of size 50, 100 and 200. Compared to the results from [5], the results are slightly worse than the ones for the Elastic Net but still much better than the Peterson and Soderberg model results.

3 Discussion

3.1 Implementation details

The Elastic Net was implemented in a sequential fashion.

3.2 Challenges

One needs to avoid exponentiating big negative numbers.

3.3 Performance

3.3.1 Time

As expected, the algorithm is quite fast and could be much faster given a parallel implementation.

3.3.2 Quality

The performance is a bit lower than expected but the results are quite sensitive to parameters.

4 Conclusion

The elastic net approach to the TSP is an interesting idea with some nice theoretical guarantees [3].

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
seaborn 0.5.1

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.

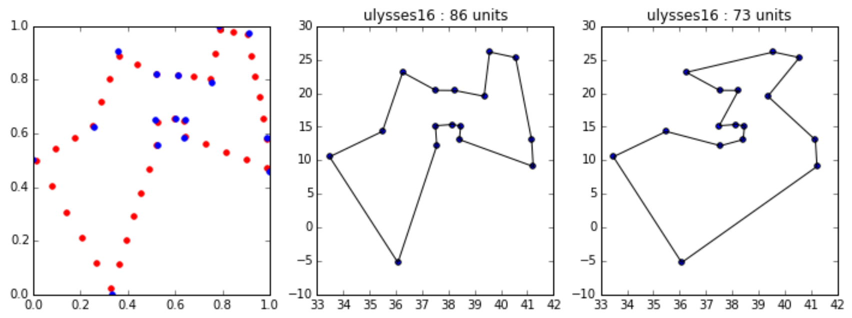
5.1.2 Running Computational Results

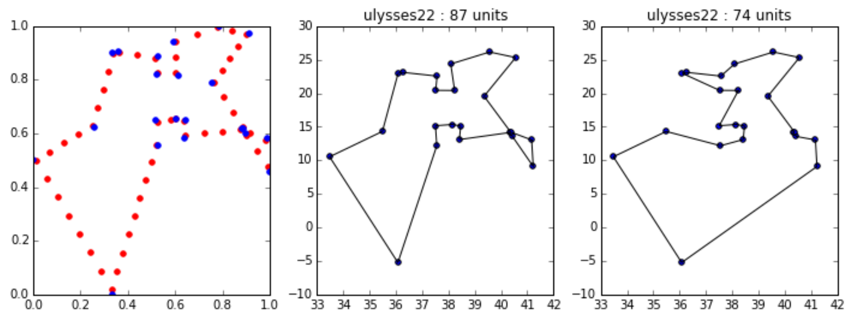
All computational results and figures are contained in the form of IPython Notebooks with the hope of allowing repeatability and reproducibility.
If IPython is available on the computer, an IPython Notebook service can be launched from command line using the following call.
This allows viewing and recomputation of the results.

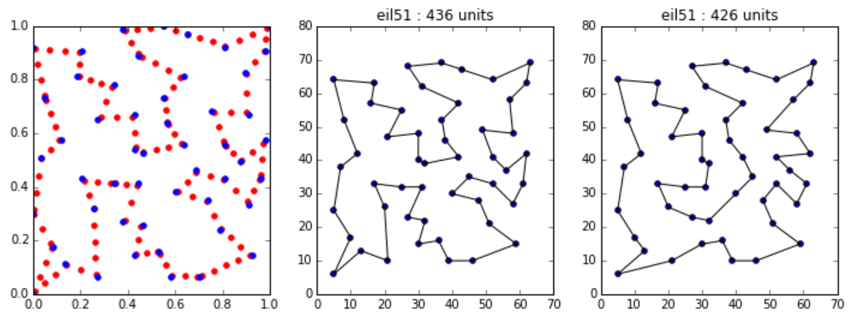
Alternatively, the IPython notebooks can be viewed online if it is reachable from a url using the nbviewer tool. This allows viewing a static version of an IPython Notebook.

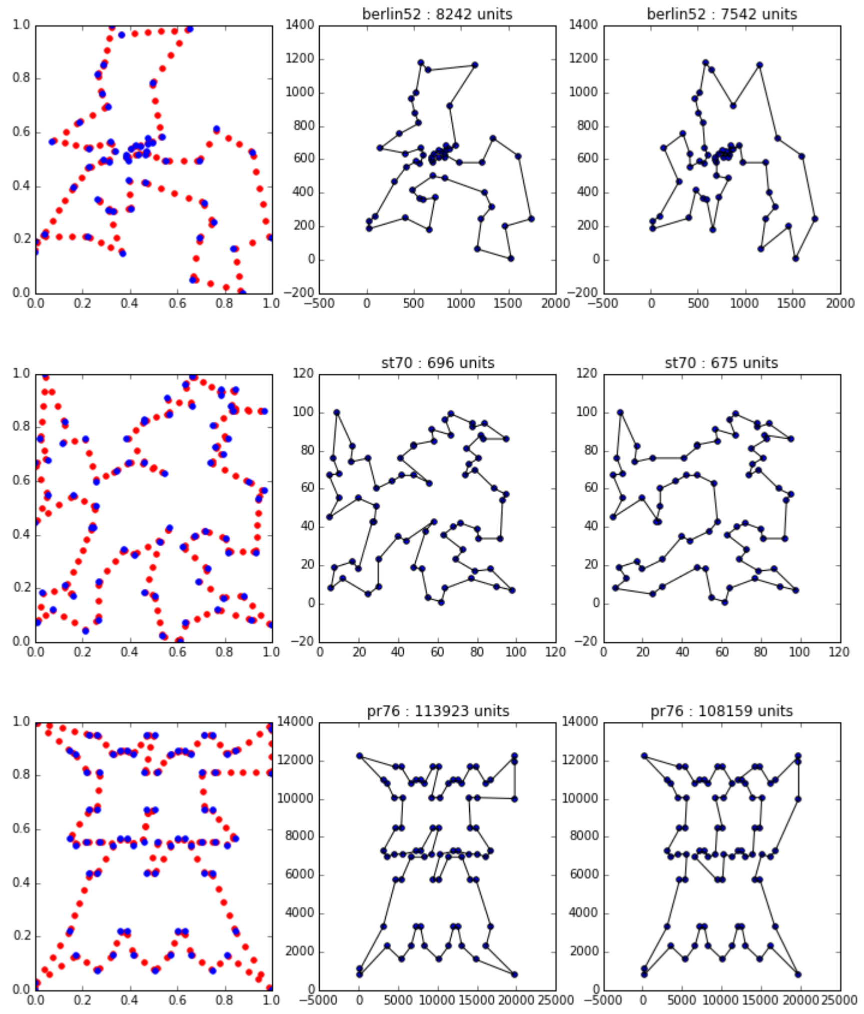
5.2 TSPLIB Solutions

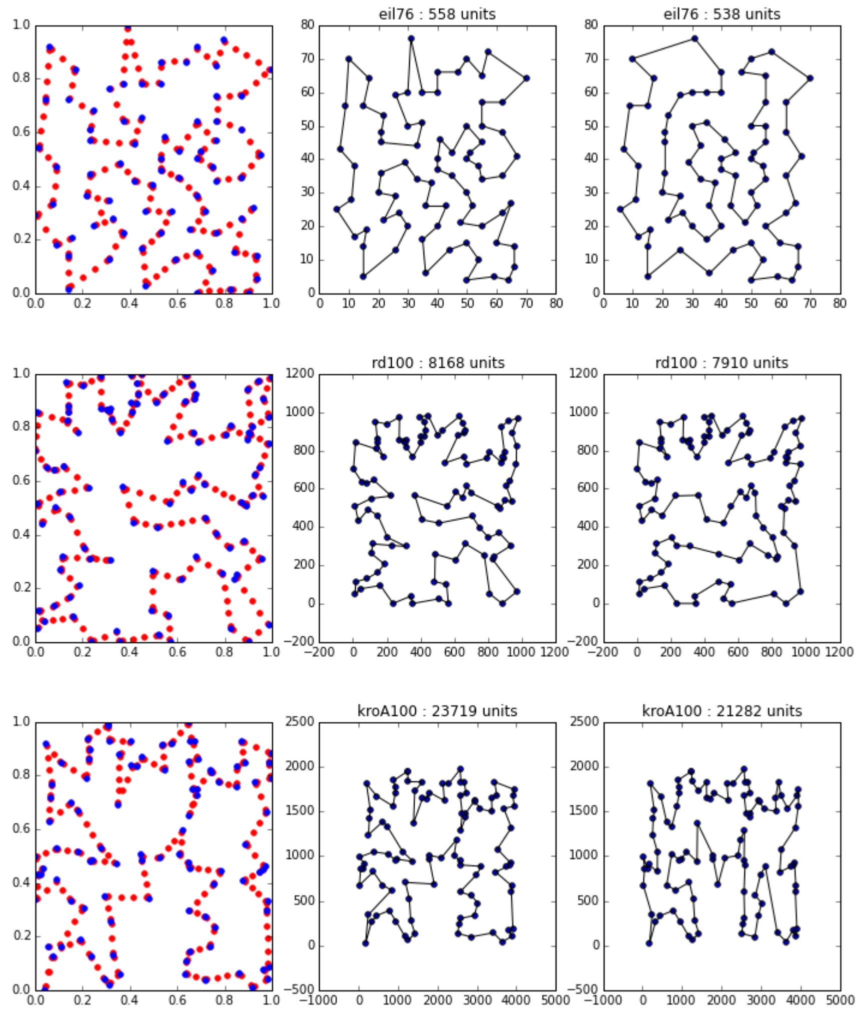
For each of the figures below, the Elastic Net, the corresponding path and the best path obtained from TSPLIB are arranged from left to right.

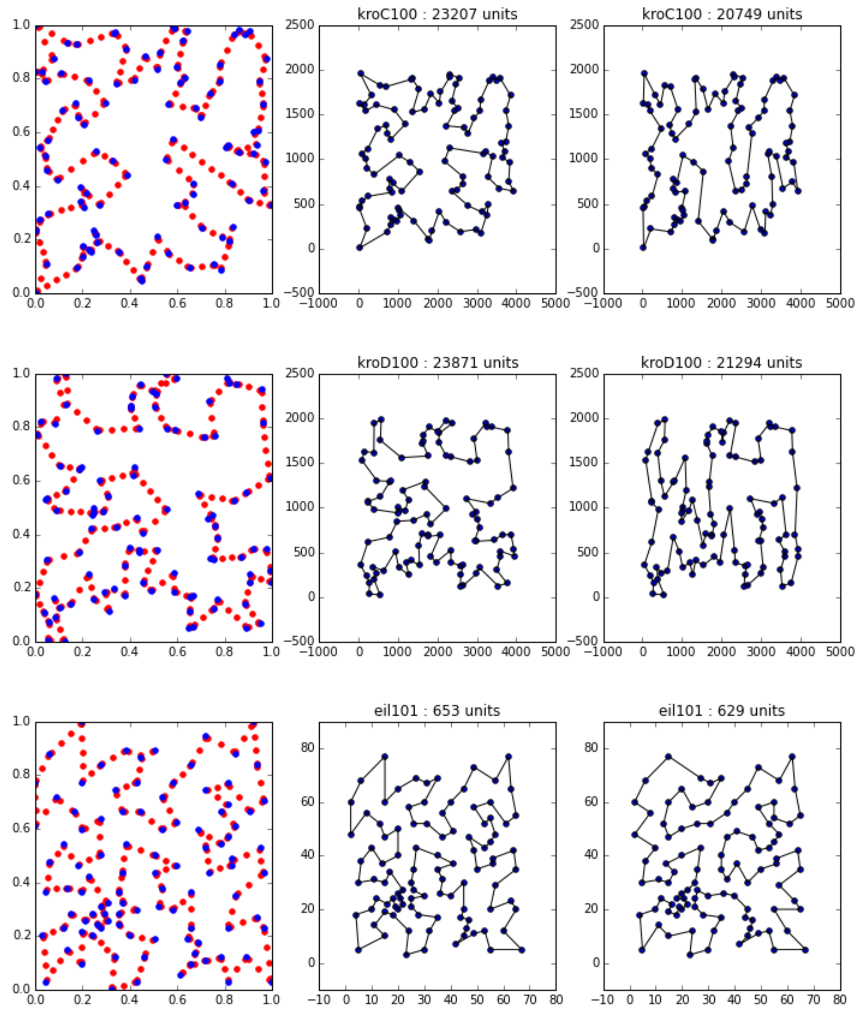


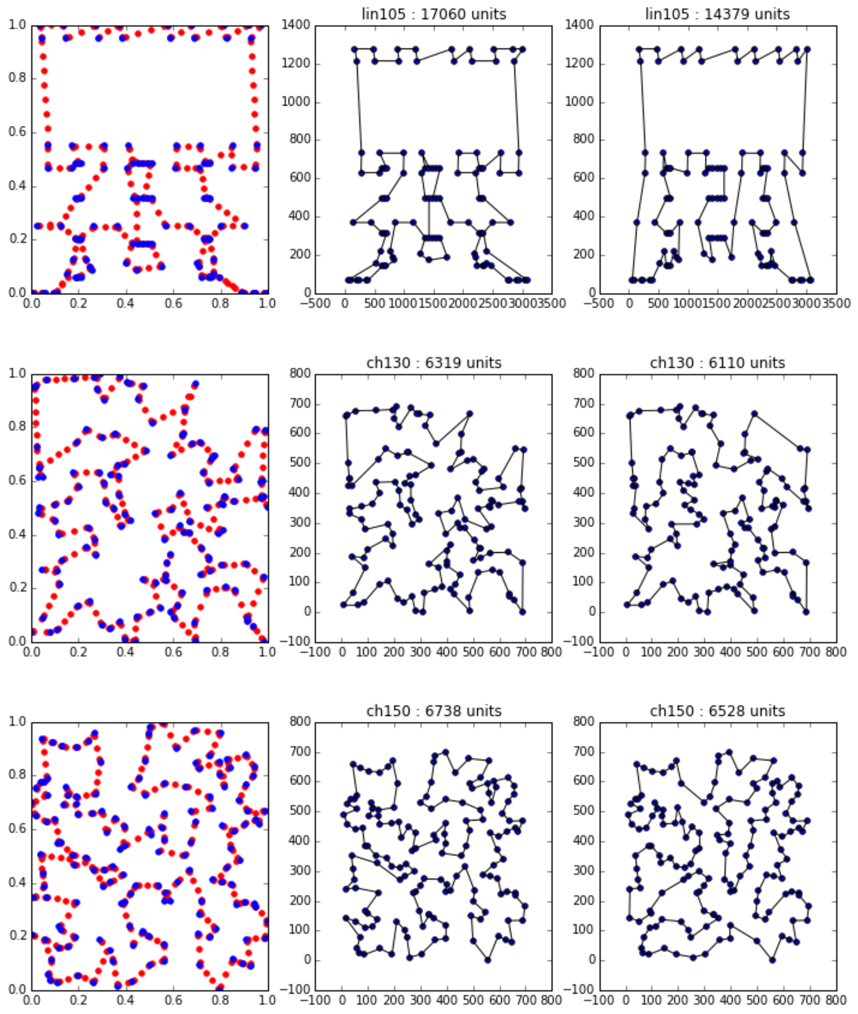





berlin52 : 8242 units
berlin52 : 7542 units
st70 : 696 units
st70 : 675 units
pr76 : 113923 units
pr76 : 108159 units


eil76 : 558 units
eil76 : 538 units
rd100 : 8168 units
rd100 : 7910 units
kroA100 : 23719 units
kroA100 : 21282 units


kroC100 : 23207 units
kroC100 : 20749 units
kroD100 : 23871 units
kroD100 : 21294 units
eil101 : 653 units
eil101 : 629 units


lin105 : 17060 units
lin105 : 14379 units
ch130 : 6319 units
ch130 : 6110 units
ch150 : 6738 units
ch150 : 6528 units