High Performance Computing: Progress Report

Instructed by: *Xiaoge Wang*

Due on: 2011.12.13

Tong Xiao, Jiaxin Mao, Zhen Ru CST92

Brief introduction

During the past two weeks, we had got a brief view of the Traveling Salesman Problem(TSP). We now clearly know the following issues:

1. The ways to find an exact solution.
2. The ways to find an approximate solution.
3. How to checkout a solution to be optimal?
4. Where are the test data?

Details will be described next.

Exact algorithms

1. Dynamic programming
   Suppose the tour start at city 1. Let $f(S, v)$ be the minimum length of the path, which started from city 1, visited all the cities in S by once, and finally arrived at city v. So we get

   $$f(S, v) = f(S - v, u) + dist(u, v), u \in S - v$$

   The answer should be

   $$min\{f(V, v) + dist(v, 1)\}, v \neq 1$$

   Here V is the set of all cities.
   The algorithm will solve the problem in time $O(n^2 2^n)$. But the memory is the limitation.

2. Cutting-Plane method
   It's an algorithm using linear programming. It was published by Dantzig, Fulkerson and Johnson. The idea is that we'd like to find

   $$min\{c^T x\}$$

   Here c^T denotes the cost vector, x denotes the vector of the selection edges.
   We can add some constraints to it. Every constraint can be expressed by $Ax \leq b$. For example, $A = I, b = (1, 1, \dots, 1)^T$, limits $x_i <= 1$. By analyzing the result, we can find more constraints. Thus we can finally get the optimal solution.
   The algorithm need to use simplex method to solve the linear programming problem. We don't know much on it so far.

Approximate algorithms

There are many different algorithms in this area. The most commonly used are genetic algorithm and simulated annealing algorithm. Both are easily to be paralleled. We will implement one of them and find out how better the solution could be compared to sequential algorithm under a same period of time.

Checkout a solution

There is a naive idea to get a quite good lower bound. First, we can draw a disk with a radius r centered at each city so that it does not consist of any other cities. Thus we get radius $r_1, r_2, \ldots, r_n$. The lower bound can be calculated as $2\sum_{i=1}^{n} r_i$. To make this bound better, we can change it into a linear programming problem by adding some constraints. Figure 1 shows an example.

There is still some space to be optimized. Given a set of connected disks, we can expand a region until

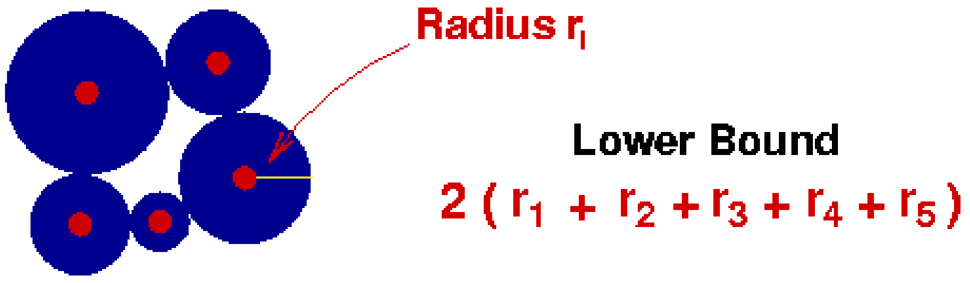
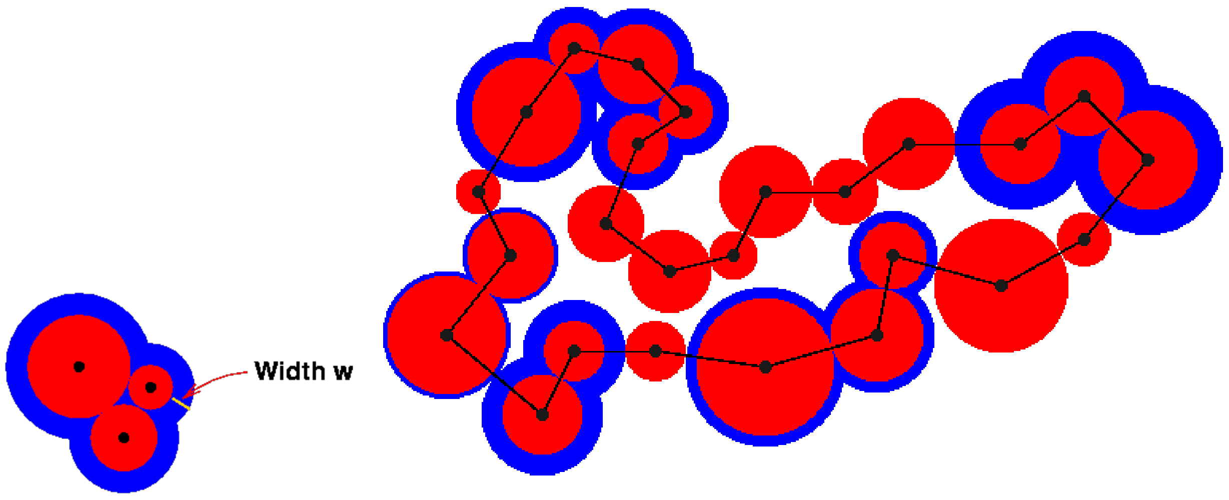


Figure 1: Lower bound

reach another region. Figure 2 shows this optimization clearly.



(a) Expand region (b) An example

Figure 2: Optimization of lower bound

Test data

There is a website for TSP data sets. For each data set, it tells the location of every city. The format of a 16 cities data set is shown below.

```
NAME: ulysses16.tsp
TYPE: TSP
COMMENT: Odyssey of Ulysses (Groetschel/Padberg)
DIMENSION: 16
EDGE_WEIGHT_TYPE: GEO
DISPLAY_DATA_TYPE: COORD_DISPLAY
NODE_COORD_SECTION
 1 38.24 20.42
 2 39.57 26.15
 3 40.56 25.32
 4 36.26 23.12
 5 33.48 10.54
 6 37.56 12.19
 7 38.42 13.11
 8 37.52 20.44
 9 41.23 9.10
 10 41.17 13.05
 11 36.08 -5.21
 12 38.47 15.13
 13 38.15 15.35
 14 37.51 15.17
 15 35.49 14.32
 16 39.36 19.56
 EOF
```

The number of cities ranges from 16 to 85900 on this website. And there gives exact solutions to some specific data sets. There are some interesting data sets with graphical view. Such as Mona-Lisa's TSP and Chinese TSP which are shown as figure 3. However, the solution is not optimal.

Future works

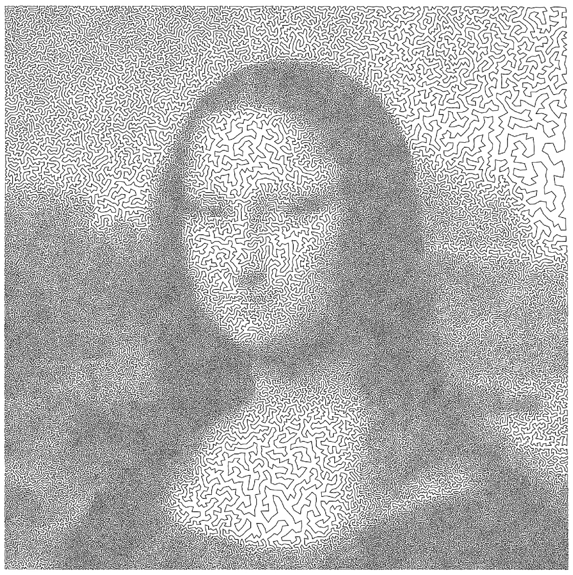
Till now, every one of us has got a general idea of TSP. We did the works together so that we can understand the algorithms more quickly. In the next two weeks, we'll focus on implementing such algorithms and testing for the performance. The major tasks are

Zhen Ru	Implement parallel GA and see how better the solution can be compared to the sequential one under a same period of time.
Jiaxin Mao, Tong Xiao	Implement parallel cutting-plane algorithm to solve TSP with quite many cities, e.g. $n >= 100$.

References

1. http://www.tsp.gatech.edu
2. http://comopt.ifi.uni-heidelberg.de/software/TSPLIB9
3. http://en.wikipedia.org/wiki/Travelling_salesman_problem

References continued on next page...


(a) Mona-Lisa, 100k points

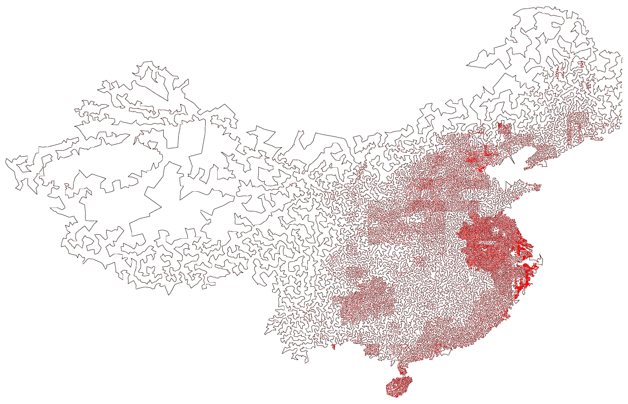

(b) Chinese Map, 71,009 cities

Figure 3: Graphical views of some data sets

4. Parallel Heuristics for TSP on MapReduce *Siddhartha Jain, Matthew Mallozzi*