CpSc 2120: Algorithms and Data Structures

Instructor: Dr. Brian Dean — Fall 2014
Webpage: `http://www.cs.clemson.edu/~bcdean/` — MWF 9:05-9:55
Handout 13: Homework #3 — Vickery 100

1 Iterative Refinement and Relaxation/Rounding

This homework will give you some familiarity with the design and implementation of optimization methods based on iterative refinement, as well as with the concept of "relaxing" a hard optimization problem to yield an easier problem, whose solution can then be "rounded" to obtain a high-quality solution of the original problem.

The problem we will consider for this assignment is a type of *linear arrangement* problem. Just as with the traveling salesman problem you have studied in lab, your goal will be to find a good ordering of n items, numbered $0 \ldots n-1$ (in the input we consider, $n = 936$). If you look at the input file

`/group/course/cpsc212/f14/hw3/wires.txt`

you will find a list of $m = 2664$ pairs of numbers. Each pair (i, j) tells you that items i and j are considered "connected". In computer science terminology, this is typically called a *graph* or *network*, with n nodes and m edges, each edge connecting a pair of nodes. Depending on the application, a graph can represent many different things. For example, it could represent a social network, with each node being a person and each edge (connection) representing a pair of friends. In another application, the nodes could represent electrical components on a circuit board, and the edges could represent wires connecting pairs of components.

The goal of this problem is to order the n nodes so that the sum of squared lengths of all edges is minimized. More precisely, we want to assign each node i to a position x_i on the number line so as to minimize

$$\sum (x_i - x_j)^2,$$

where the sum is taken over all edges (i, j). The positions $x_0 \ldots x_{n-1}$ of the n nodes must be some permutation of $\{0, 1, \ldots, n-1\}$. That is, the nodes are to be mapped to positions $0, 1, \ldots, n-1$ on a number line in some optimal order.

This problem has several applications. In the context of a social network, it asks us to order the n individuals so that friends tend to be located near each-other in the ordering (if two friends i and j are far away from each-other, the term $(x_i - x_j)^2$ in the objective would be large, adding a substantial penalty to the total objective). If our n nodes represent electrical components, this problem asks us to arrange these components in a line so as to minimize the total squared lengths of all wires connecting the components, which leads to an efficient circuit layout. Figure 1 shows an example of a small problem instance on just 4 nodes, along with two possible orderings; the first

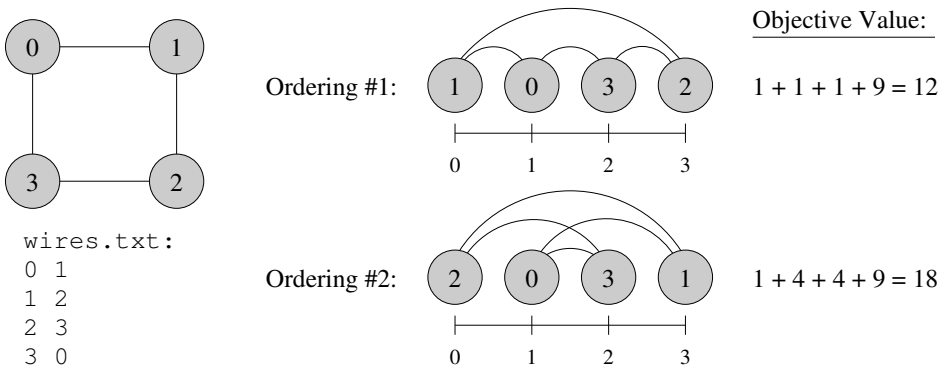

Figure 1: Example instance with 4 nodes and 4 edges, with two possible orderings, one better than the other.

has objective value $1^2 + 1^2 + 1^2 + 3^2 = 12$, which is better than the second, with objective value $1^2 + 2^2 + 2^2 + 3^2 = 18$.

Since the problem above is known to be NP-hard, it is not known how to compute an optimal solution in an efficient manner (for the input file given, we know the objective value of an optimal solution is slightly less than 1 million, but we do not know the exact optimal solution). It will be interesting to see the quality of solutions you, as a class, mange to compute.

2 First Approach: Iterative Refinement

Your first goal is to generate a good solution using iterative refinement. Much as with the TSP lab, you will start with an arbitrary (random) ordering of the n nodes, which you will continually refine until it becomes locally optimal.

How exactly you refine your solution is left slightly open-ended on purpose, to allow you to experiment with different ideas (this is often a key part of the process of developing an effective solution based on iterative refinement). For example, in each refinement step, you could try all neighboring solutions obtained by:

1. Swapping a pair of adjacent nodes in the ordering,
2. Swapping a pair of non-adjacent nodes in the ordering, or
3. Reversing part of the ordering (as with the TSP lab).

Of course, these are just suggestions – you certainly don't need to try all of them, and you are encouraged to try other ideas as well. Note that (i) involves checking only $O(n)$ neighboring solutions, while (ii) and (iii) involve checking $O(n^2)$ neighboring solutions, so with $n = 936$ the latter two approaches will take substantially more time (although on the other hand, they may identify improvements that the simpler approach (i) would miss). Also note that if you swap two nodes, you can quickly compute the resulting change in objective value by only looking at the edges connected to those two nodes and seeing how much these change in length (since all other edges remain the same length); this can help speed up your search tremendously.

As with the TSP lab, you might want to consider whether you want to move immediately to a better neighboring solution when it is found, or whether you want to search the entire neighborhood and only then move to the best neighboring solution. In the first case, you may want to be cautious if you only search for neighboring solutions, say from left to right in the ordering, since this may cause your swaps to be concentrated only on the left side of the ordering (since you will likely discover a good swap before scanning very far to the right); this might lead to slow convergence.

You should continue to refine your solution until it becomes locally optimal (where there are no neighboring solutions that are better). Your code should print out the final ordering and its objective value – the total squared length of all its edges. For full credit, you must obtain an ordering of objective value less than 1 million with at most 10 minutes of running time.

If desired, you can restart your algorithm from several randomly chosen initial orderings (as with the TSP lab). However, note that it may already take a few minutes to refine each ordering, so you may not have much time to try too many orderings.

3 Second Approach: Relaxation and Rounding

Our original problem asks us to assign each node i to a position x_i (where the x_i's are restricted to be a permutation of $\{0, 1, \ldots, n-1\}$) so as to minimize $\sum(x_i - x_j)^2$ over all edges (i, j). This problem is unfortunately NP-hard, but if we relax the constraint that our nodes need to be assigned to integer locations in the set $\{0, 1, \ldots, n-1\}$, the problem becomes efficiently solvable, leading to another interesting method for generating good solutions via iterative refinement.

Suppose we guess that node a should be first in our ordering ($x_a = 0$) and node b should be last ($x_b = n - 1$). For the remaining nodes, suppose that we want to assign them to locations between 0 and $n - 1$ which no longer need to be integers. That is, we might assign node i to a location $x_i = 3.71$. By removing the constraint that our nodes need to be embedded at integer positions, this makes the problem of minimizing $\sum(x_i - x_j)^2$ *much* easier. By taking the gradient of the objective function and setting it to zero, we get a linear system (if you know multi-variable calculus, you are encouraged to give this a try, although it is not a necessary part of this assignment). By examining the linear system, we find that in any optimal solution, the position x_i of node i must be the *average* of the positions of i's neighbors in the network (the nodes connected to i).

The observation above gives us a simple iterative refinement method for optimally solving the relaxed problem: start with $x_a = 0$ and $x_b = n - 1$, and x_i set arbitrarily in the range $0 \ldots n - 1$ for all other nodes i. Then, we repeatedly replace the position x_i of each node i with the average of the positions of its neighbors (the only two nodes we do not update are a and b, which stay fixed at the endpoints of the ordering). Over sufficiently many iterations, the positions of the nodes will converge so as to minimize $\sum(x_i - x_j)^2$. You can test for convergence by looking at how much the positions of the nodes move in each iteration, stopping when they cease to move very much.

After solving the relaxed problem, we have a solution in which nodes are assigned to non-integer positions like $x_i = 3.71$. We can use this as a guide for obtaining a good solution for the original problem, however, by simply sorting the nodes by their positions, and assigning them to locations $0, 1, 2, \ldots, n - 1$ in this order. By "rounding" our solution in this fashion, we will typically obtain a reasonably good solution to the original problem.

The only question remaining is how to choose the endpoints a and b. You could try running the approach above for all $O(n^2)$ possible choices of a and b, but this might take too long, so instead

you may want to run the approach above on a smaller number of random choices for a and b, taking the best final solution you get. For full credit, your code should produce an ordering of objective value less than 1 million in less than 10 minutes of compute time on the lab machines. Please print out the final ordering of your nodes, along with the objective value of this ordering.

4 Submission and Grading

You should submit two programs for this homework, the first using iterative refinement and the second using the relaxation and rounding approach (which is also based on iterative refinement, of course). Each program should produce output in less than 10 minutes when run on the lab computers (we will only run them for up to 10 minutes during grading).

If you wish, feel welcome to submit a text file containing the best ordering you can obtain for the input file given in this assignment – irrespective of the technique used or the running time spent. This is only for fun, to see who in the class manages to come up with the best solution overall. You will earn no points for this part, only the respect and admiration of your instructor.

Please name your submissions for the two parts of this assignment `part1.cpp` and `part2.cpp`.

Final submissions are due by 11:59pm on the evening of Wednesday, November 5. No late submissions will be accepted.