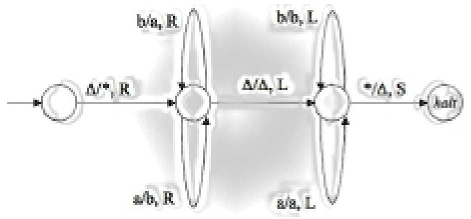
CS 312: Algorithm Design and Analysis



- Syllabus
- Schedule
- Assignments
- Projects
- Grades
- Slack
- Miscellaneous

Project #6: Group Project---Advanced Approaches to the Traveling Salesperson Problem

In this project, you will compare accuracy and the theoretical and empirical complexity of different algorithmic solutions to the traveling salesmen problem (TSP).

Objectives

- Wrestle with an NP-complete problem
- Implement a greedy algorithm for finding solutions to the TSP
- Implement an algorithm of your choice to get high accuracy "approximate" TSP solutions in "reasonable" time
- Develop your ability to conduct empirical analysis and understand resource trade-offs by comparing your algorithm, the greedy algorithm, and your branch and bound solution (project #5) for TSP
- Work effectively as a group

Teams

This will be a group project on which you will all work in teams of 3-4 students. Your team assignment can be found here. As a team you will hand in one report and also do one group presentation on the last day of class. Your team will all receive the same base grade for the project. To augment this, each team member will e-mail me a thoughtful and honest evaluation of the contributions of their team members, including themselves. For each individual, this evaluation should include a score from 1 to 10 indicating your evaluation of their work (10 meaning they were a valuable member of the team that made significant contributions to the project and were good to work with, 1 meaning they contributed nothing and were difficult to work with). If you would like, you may also include any clarifying comments, etc. (especially for low scores). If a person receives consistently low evaluations from peers, then their score will be proportionally decreased.

Framework

You will use the same framework used for the TSP Branch and Bound project, implementing methods for (at least) two additional solver techniques. Since this is a team project, it is not necessary for every member to extend their B&B project. Instead, choose one person's code and work on it collaboratively.

Your Algorithm

Your goal will be to implement an algorithm which can solve the largest possible city problems with the most possible accuracy (approximation) in "reasonable" time. You can use any algorithm you come up with. It is all right to look on the web or elsewhere to find algorithms, but you should write your own code. Note that since we are requiring "reasonable" runtimes, your algorithm will likely not find optimal length tours (unless you make a huge breakthrough!) You may not use branch and bound with different bounding functions as that would not give you as much of a new experience.

To help make the presentations a better educational experience, we will have no more than 3 groups doing the same (or very similar) algorithm. As soon as you know which algorithm you want to do, send me an e-mail. This will be first come first serve. If 3 groups choose a similar approach, I will immediately send out an e-mail stating that that approach is no longer available.

To Do

1. Implement a greedy algorithm which starts from an arbitrary city, picks the shortest path available to an unvisited city, then picks the shortest path from that city to an unvisited city, etc., until it returns to the origin city. Note that for Hard problems, it is possible to reach a city which has no paths to any remaining unvisited cities (don't forget to check for a path from the last city back to the first to complete the cycle). In such cases, there is no legal solution (unless you backtracked which you don't here). Your algorithm should iteratively try starting from each city and return the best solution found for any of them.
2. Implement your own algorithm of any type (except B&B) to solve the TSP. You will use the default random and your greedy algorithms as a baseline and see how much better you can do. The best algorithms will have a good combination of
    a. % improvement in solution quality (tour length) over random and greedy
    b. size of problem (# of cities) that can be handled

   You should also try to be somewhat creative in your approach, by either coming up with your own algorithm, or trying to extend or modify a published algorithm. Your grade will in part be based on these three components:
    a. % improvement over greedy
    b. the size of problems can it handle
    c. the creativity of your algorithm

   As long as you make a good effort in these areas you can get full credit. If you do your own creative algorithm which may not be as good as published versions, you will still get rewarded for your creative effort. There will also be some extra credit for those groups who get exceptional results or who demonstrate significant effort and creativity on their algorithm.

Report

(due ~~as a single hardcopy at the beginning of~~ electronically (slack or email) by midnight on the last day of class)

1. [10] Correct implementation of the greedy TSP algorithm. Brief discussion and complexity of the algorithm.
2. [40] Correct implementation of your own TSP algorithm including a discussion of the algorithm, why you chose it, and its pros and cons. Give proper attribution (references) for ideas you find externally. Clearly point out which ideas came from other sources, and which ideas are original contributions that you made. You do not need to include source code in your hardcopy, nor e-mail your source. Also include a discussion of the theoretical big-O complexity of your algorithm, and discuss how the empirical complexity matched that. Include screenshot examples of typical results for both algorithms (greedy and yours).
3. [35] Include a table with columns (see below) for each of your TSP algorithms including the Branch and Bound TSP algorithm you implemented in Project 5 (use one of your individual project 5 implementations as a representative version of B&B; however, do not set your time limit at 30 seconds as you previously did). For the random algorithm, use the default algorithm provided with the original framework (this acts as baseline for the greedy algorithm). You can play with all three levels of problem difficulty (Easy,

Normal, Hard) during testing, but just use the Hard level for all of your reporting. For the greedy algorithm report the % improvement over random [calculate this as 1 - (greedy_cost/random_cost)]. This will help calibrate and make sure your greedy algorithm is implemented correctly. For B&B and your own algorithm, report the % improvement over the greedy algorithm (this should be a positive number representing how much better your tours are as a percentage of the greedy tours). You should create a table just like the one below and with the city sizes shown below (15, 30, 60, 100, 200), so that we can easily compare with each other's work. Round average tour lengths to the nearest integer. Round time and % improvement to two significant digits beyond the decimal. Your results for each cell should be the average of some number (at least 5) of runs with different random seeds for that number of cities. Do not try to find a particular set of seeds on which your implementation does best. They should be randomly chosen runs. You will fill in average time (seconds) and average tour length for the different numbers of cities. If an algorithm takes more than "reasonable time" (more than about 10 minutes or so) to solve for that number of cities, then just fill in the cells for that algorithm with "TB" (Too Big). You can also include any other information you feel is interesting. The numbers in the example table below are typical numbers from past projects (yours will vary, but this allows you to sanity check your results).

	Greedy		**Random**		**Branch and Bound**			**Own Algorithm**		
# Cities	**Time (sec)**	**Path Length**	**Path Length**	**% Improve**	**Time (sec)**	**Path Length**	**% Improve**	**Time (sec)**	**Path Length**	**% Improve**
15	0.00021	4100	7743	-88.85%	3.26	3527	13.97%	1.47	3649	11.00%
30										
60										
100										
200	0.11	15798	100761	-537.81%	TB	TB	TB	165.01	14244	9.84%
...										

To complement the required five rows shown in the table above, you should also add to your table some additional rows (different numbers of cities). In particular go as large as you can before getting "Too Big." The last row of your table should be the largest number of cities which can be done in a reasonable time limit. Discuss and analyze the results in your table.

4. [5] Your write-up should look sharp and follow the form of a brief conference paper with abstract, introduction, algorithm explanation, complexity, analysis of results, and future work. A couple examples of excellent work and reports from previous semesters can be found here, here, here, and here.

Oral Presentation

5. [10] Each group will give an oral presentation of their project on the last day of class, consisting of a short slide presentation (4 minutes). Do not take time in the presentation to discuss the TSP problem or its basic greedy and Branch & Bound solutions, as everyone will already be familiar with these. Instead, the presentation should
   a. introduce and explain your algorithm (along with this briefly review why you chose this algorithm, how much is original while giving appropriate attribution, any challenges/detours along the way, and the pros and cons of your approach)
   b. present and discuss the theoretical big-O complexity of your algorithm
   c. show and discuss your version of the table above
   d. (briefly) describe what next steps you would try if you had more time to work on the project

   Your presentation should contain *no more* than one slide per topic point above, should be professional, and will need to be practiced to make sure it fits the time constraint (given the short amount of time allotted, it is probably wise to select a single group spokesman for the presentation).