CS4234: Optimisation Algorithms Tutorial 10

SLS Part 3; Final Preparation 2

V1.7: Steven Halim *November 6, 2023*

Preliminaries

By now, we have seen about almost three weeks worth of SLS ideas. By Lecture 10, you will realize that when one is presented with a new (NP-)hard COP Z, or even COP A again but with different constraints (e.g., the classic TSP/MWVC but with very limited run time limit), one cannot simply take 'any favorite' SLS algorithm from a book/lecture note/research paper/one's own experience and apply that SLS algorithm verbatim with 'default parameters' on problem Z and hopes to get a good result out of the box. In this tutorial, we will apply what we have learned in our Mini Project experimentations so far. This is the last moment to discuss Mini Project ideas freely before we freeze the results for this semester on Thursday, 09 November 2023, 11.59am.

Discussion Points

Q1: In Lecture 10, Prof Halim discussed that various (still NP-hard) TSP instances have different properties that can be 'attacked differently'. So, please investigate Metric-No-Repeat TSP test case patterns. See `http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/` and perhaps try to visualize (or Google :O) the visualizations of some of these well known TSP benchmark test cases. What are the (initial) conclusions and/or local search ideas that you have after seeing all those?

Q2: In slide 8 of `https://www.comp.nus.edu.sg/~stevenha/cs4234/lectures/10.SLS-DTP.pdf`, Steven has outlined the list of potential parameters (type-1 of SLS DTP), components (type-2 of SLS DTP), and search strategies (type-3 of SLS DTP) of Tabu Search (TS) Meta-heuristic. Now please do the same for Iterated Local Search (ILS) Meta-heuristic. You can refer to `https://www.comp.nus.edu.sg/~stevenha/cs4234/lectures/09.Meta-heuristics.pdf` for the **bold red text** parts of ILS or other resources to give a more complete view.

Q1: Past paper (AY2019/20) hidden MCQs (up to Lecture 10):

1. Which part of SLS algorithm engineering process is usually the most time consuming?
   (a) Picking the correct SLS algorithm
   (b) Picking the correct components of the chosen SLS algorithm
   (c) Implementing the SLS algorithm into a working program
   (d) Debugging the SLS algorithm
   (e) Fine tuning the SLS algorithm
2. Context: Mini-Project 1: Traveling-Salesman-Problem (TSP) of up to 1000 vertices with limited 2s runtime (single thread; or $2/k$ second if we have k threads). Which of the following SLS/heuristic ideas has the highest chance to work well in your opinion? You can assume that 2-edges-exchange (2-opt) Neighborhood relation is used for all 5 options below.

(a) Steepest Descent with random restart upon hitting any Local Optima

(b) Simulated Annealing with aggressive cooling function

(c) Iterated Local Search that only accepts the better of the two Local Optima after each perturbation+subsidiary local search steps

(d) Evolutionary/Memetic Algorithm with a medium-size population

(e) Tabu Search algorithm with high Tabu Tenure to encourage exploration of search space

3. Context: Mini-Project 2: MIN-WEIGHTED-VERTEX-COVER (MWVC) of up to 4000 vertices and 600 000 edges with limited 2s runtime (single thread; or $2/k$ second if we have k threads). Which statement is likely incorrect?

   (a) We can use add vertex to vertex cover and remove vertex from vertex cover local moves

   (b) We need to favor intensification to get a good performing SLS for MWVC

   (c) We can use gain/loss scoring heuristic to help the SLS picks the next local move

   (d) We need to take care of potential *infeasible* solutions that are not a vertex cover

   (e) If the graph is dense, we can complement the input graph and optimizing for MAX-CLIQUE or choose to optimize for MAX-INDEPENDENT-SET instead

Remarks Before we end our CS4234 tutorial sessions this semester, let's take a class photo as momento.