CS4234: Optimisation Algorithms Mini Project

October 22 – November 12, 2020

V1.4: Steven Halim

Preliminaries

This mini project spans 3 weeks from Thursday of Week 10 to Thursday of Week 13. It worth 15% of the course weightage (3x of the normal PS). It focuses mostly on things that are difficult, or perhaps impossible, to be examined on paper for just 2 hours, which is the experimentation and empirical analysis of Stochastic Local Search (SLS) techniques on real (NP-)hard COPs.

To reduce your own project workload and my own grading workload in the busiest last month of the semester, I reduce the number of student submissions (up to 44) by a factor of 4 by making you all form a Project Group[1] of size 4. Note that 44 % 4 == 0, so there are exactly 11 project groups with exactly 4 students.

This mini project is therefore a 3-weeks, group-project of size 4 students/group, Open Internet assignment.

This semester, 8 project groups self choose themselves whereas the last 3 project groups were 'randomly grouped'.

To-Do

Use *any* techniques that you have learned so far in CS4234 (or will learn in the last few weeks of CS4234, or any hacking techniques, or any technique that you can find in the Internet - read only mode as usual...) to *get the best possible results* on two (NP-)hard Combinatorial Optimization Problems (COPs) + 1 'gradient descent'-type problem (the 'easiest' among the three). As you will work in a team of 4, set one of you to clear problem C with Week 10 lecture technique quickly, then split the workload by having 2 members do one COP + the other 2 members do the other COP and share the successful (or unsuccessful) techniques that have been tried, or all 4 members can attack both COPs, and simply present the best solutions (and combined experimentation reports) found by any team member at the end.

The first COP for the fifth AY is still the same as the first four AYs: TRAVELLING-SALESMAN-PROBLEM (No-Repeat version, i.e., NR-TSP, not 100% metric but almost...) that is available at Kattis. The second COP is a 'now-no-longer-that-new' task at Kattis that is prepared by a CS4234 alumni: Ranald Lam Yun Shao. For the first COP, Steven has lots of research data from his own PhD days: 2004-2009 and seniors works in 2016-2019. For the second COP, Steven has the 2018-2019 data.

COP 1: `Travelling-Salesman-Problem` (TSP) (7%)

Assoc Prof Per Austrin from KTH Royal Institute of Technology, Sweden, whom I have met several times in ICPC World Finals and helped as (external) technical committee for my ICPC Singapore 2015, has set up this problem: `https://nus.kattis.com/problems/tsp` back in 2006 (14 years ago) which is very suitable for our CS4234 training purposes. If you read the problem description carefully, it is clearly the No Repeat TSP that we discussed back in CS4234 Lecture 4.

However, it is not perfectly Metric TSP because although we use Euclidean distances, we have to *round them to nearest integers*, i.e., we may have a triangle with side lengths of $a = 1.4$, $b = 1.4$, $c = 2.6$ (which satisfies triangle inequality as $a + b \geq c$) but if the side lengths are rounded to nearest integers, we may have $a = 1$, $b = 1$, and $c = 3$ (which does *not* satisfy triangle inequality anymore as $a+b < c$). This round to nearest

[1] This `Project-Grouping` (`Min-Clique-Cover`/`Min-Exact-Cover`) is an NP-hard optimization problem as we have seen in last and this semester's midterm test.

integer operation is called as the *nint(x)* operation in the famous (to TSP researchers) TSP Benchmark Library called: TSPLIB: `http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/` that Steven used during his PhD days. More detailed documentation of TSPLIB format, especially the EUC_2D format, can be found in: `http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/tsp95.pdf`. Albeit it is not perfectly Metric TSP by definition, running 2- (or the harder to implement 1.5- (or the 1.5-e recently), if you choose to do so) approximation algorithm for Metric TSP should still give a reasonable result.

Now `https://nus.kattis.com/problems/tsp` provides a convenient platform for Steven to have CS4234 students experiment on various SLS techniques on TSP. We will hear the first few preliminary techniques during Lecture 9 on Week 10 that will be expanded further in the subsequent lectures. You are given only 2 second per (secret) TSP instance. There are 50 TSP instances of up to $N \leq 1000$ points (vertices). You are given a certain scoring function that measures your tour quality of each TSP instance against its (known) optimal value (maybe after running the optimal algorithm for hours).

These are the benchmarks:

- As of 18 October 2016, getting 42.0/50 points already put you in the top 10.
  Steven's PhD code from $\approx$ 11 years ago was at $\approx$ 41.5/50 in 2016 (I don't bother optimizing any more).
- As of 24 October 2017, you need 45.7/50 to be in top 10,
- As of 22 October 2018, you need 47.1/50 to be the top in the world (for this problem), set by one CS4234 senior Tan Jun An,
- As of 23 October 2019, you need 48.2/50 to be the top in the world (for this problem), set by another CS4234 senior Bay Wei Heng.
- As of 22 October 2020, you need 48.8/50 to be the top in the world (for this problem), set by another CS4234 senior Alvin Yan.

Can you be close to these benchmark results or even do better (be the next new number 1)?
See `https://open.kattis.com/problems/tsp/statistics` for the latest live statistics.

Code of conduct for this Mini Project (applicable to both problems): Steven is aware that by simply Goggling 'Kattis tsp' will yield interesting hits of past public results, like this public GitHub repository `https://github.com/estan/tsp` (I already tried resubmitting this verbatim, it currently get $\approx$ 36/50) that may give you some initial local search ideas and working :O implementation ideas with reasonably high scores. As it is impossible for Steven to police this, I will let you peruse those *current* publicly available material in your quest to get as close as possible to 50/50. The rule of NOT posting your own project to public domain throughout this semester (and beyond - so that your code is not copied by juniors) as with our other PSes remains, basically you cannot share code beyond your group mates now and in the future. Obviously anyone who can get >48.8 with legal means will (likely) get full marks for this task.

The scoring scheme for this COP 1: TSP is therefore:

1. Your team best score at Kattis (maximum 50) when ranked against the other 7 teams in CS4234 (3.5%, with the best group (hopefully within top 10 @ Kattis) taking all 3.5% and the next ranked group gets 0.2% less, i.e., 3.3% and so on (as we will only have exactly 11 teams, the last one will get at least 1.5 for this component). Each group has to show briefly the nus.kattis submission id (Steven can verify) that generates that best score for verification purpose (and to see if you copy too much (or all?) from the few available public resources :(...). Steven will have access to all your Kattis submissions at nus.kattis and all those plagiarism flag alerts, so never submit anyone else code verbatim (or with trivial modifications) :O... Note that submissions to the open version `https://open.kattis.com/problems/tsp` will **NOT** be tracked this time but will be 'roughly counted' for the next check below.
2. A penalty of 0.01% per submission **AFTER the first 30 free submissions**, i.e., submitting up to 130-30 free = 100 more times after the free quota (the reason: to avoid crashing Kattis server, it is being

used by other Universities too) will get you a 1% score deduction, submitting 380 times essentially cause your team to have 0 point this task. This counting is shared among all 4 team members and include both open.kattis AND nus.kattis submissions. The spirit of the rule that can be modified depending on the 'creativity' of this year's teams: Team canNOT write a script to 'test' the judge. Please separate 'training data' (that you can generate yourself) with 'test data' at Kattis server.

3. Your group presentation about your experimentations on TSP @ Kattis on Thursday, 12 November 2020 (2.5%, categorical grading: average: 2%, good: 2.5%, bad: 1.5%). Steven is more interested in your learning/experimentation journey on various SLS techniques on attacking the TSP rather than just the final result above. Therefore, you have to also record negative results and should be able to give some explanation on why such (failed) ideas did not work.

4. Your no-more than 2 pages of report about TSP experimentations (the written from of your presentation above), for Steven to re-read after your presentation to give final grade (1%, categorical grading: OK: 1% or poor: 0.5%).

COP 2: `Min-Weight-Vertex-Cover` (MWVC) (7%)

For the second COP this S1 AY20/21, Steven and Ranald have ported the Min-Weight-Vertex-Cover task that was previously on Mooshak to Kattis. The format is set to be identical with the TSP version.

These are the benchmarks:

- As of 14 November 2018, Ranald's best code (the best performing code as of last AY) scored 25.5/40 for this problem setup (the setup was slightly different that time).
- As of 23 October 2019, you need 35.1/40 to be the top in the world (for this problem setup) as Ranald has found some more improvements after he completed CS4234R in S2 AY 2018/2019.
- As of 22 October 2020, you need 37.1/40 to be the top in the world (for this problem setup), set by one CS4234 senior Mathis Chenuet.

The scoring scheme for this COP 2: MWVC is therefore:

1. Same as /tsp: Your team best score at Kattis (maximum 40 for this /mwvc task) when ranked against the other 10 teams in CS4234 (3.5%, with the best group (many should be within top 10 @ Kattis as it is new) taking all 3.5% and the next ranked group gets 0.2% less, i.e., 3.3% and so on (as we have exactly 11 teams, the last one will get at least 1.5 for this component).
2. Same as /tsp: A penalty of 0.01% per submission **AFTER the first 30 free submissions**.
3. Your group presentation about your experimentations on MWVC problem on Thursday, 12 November 2020 (2.5%, categorical grading: average: 2%, good: 2.5%, bad: 1.5%). Steven is more interested in your learning/experimentation journey on various SLS techniques on attacking the MWVC problem rather than just the final result above. Therefore as with TSP report, you should also record negative results and provide explanations.
4. Your no-more than 2 pages of report about MWVC problem experimentations (the written from of your presentation above), for Steven to re-read after your presentation to give final grade (1%, categorical grading: OK: 1% or poor: 0.5%).

Task 3: `Gradient-Descent`-type Problem (1%)

This is a recap of what we discussed on Week 10 lecture: 'gradient descent' (somewhat similar to the 'hill climbing') type problem. You can use this 'simple' task to practice on this technique before potentially using similar technique for /tsp or /mwvc above.

Postlude

Steven has more ideas to expand this kind of mini projects further for future AYs. Interested students can apply for 1 MC CS4234R in S2 AY 2020/21 to work with Steven to add one more new NP-hard COP (likely to replace the ageing /tsp task that may be just used as in-lecture demo in the future).