AIM Project 2020 – Implementation of a HyFlex Compatible PWP Domain (Project Report Exercise Sheet)

The project coursework is comprised of two parts; a domain implementation (80%), and a project report (20%). The sections below outline different parts which should be discussed in the report, and we urge you to provide the same headings in the report as in this description for clarity. Your report should not exceed 2,000 words and should be no longer than 6 pages using 11pt sized fonts throughout. Any content after whichever comes first will be discarded upon marking.

1 (Short) Question and Example based Essay

You are first asked to describe some elements of your implementation, explaining the reasons why you have implemented them the way you have, and how they operate. For question 1.3, even if you have not implemented delta evaluation then you should still explain how you would have done it so that you can receive the marks for the respective question.

1.1 Random Initialisation

You were asked to use the permutation representation for representing solutions to the PWP problem instances. In this section, you should illustrate how the locations as defined in the "Square.pwp" instance file are mapped to locations within the solution representation and how you generated a random solution as used by the `initialiseSolution(int index)` method from the HyFlex API.

You should explain the following where s_0 is an initial solution of your choice, and $s_1 = [0,1,2,3,4,5]$

(1) Explain how **all the locations** in square.pwp are mapped on to the solution s_1.
(2) Draw out the solution for square.pwp given the solution representation s_1, labelling each location with their coordinates and location type.
(3) Explain briefly how you create a randomly initialised solution (s_0) with specific reference to the representation.
(4) Draw out the solution for square.pwp given the solution representation s_0 generated in part (3).

1.2 Inversion Mutation

You should explain how you have implemented the `apply(PWPSolutionInterface oSolution, double dDepthOfSearch, double dIntensityOfMutation)` method for **Inversion Mutation**, specifically how the solution **representation** is changed. **You should give a small example using 6 delivery locations and go through your implementation step-by-step showing how the representation is modified over each steps of your algorithm for two locations which have at least two other locations between them in the current tour** given an initial solution representation $s = [0,4,1,3,5,2]$.

1.3 Delta Evaluation for Adjacent Swap

You should explain how the objective value of a solution can be updated using delta evaluation when applying the adjacent swap heuristic given a problem instance with 6 **delivery locations** $(L_1, L_2, L_3, L_4, L_5, L_6)$ and where the distance between two locations can be represented as $C(L_x, L_y)$, and each current and candidate solution cost as $f(s_i)$ and $f(s_{i+1})$. You should do this for the following scenarios:

1. Adjacent swap of locations L_1, L_2.
2. Adjacent swap of locations L_3, L_4.
3. Adjacent swap of locations L_6, L_1.

2 Tramstops-85

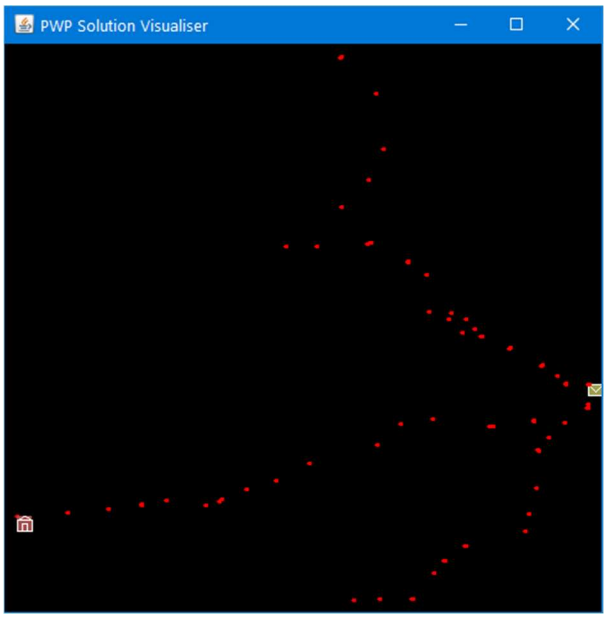

Figure 1 shows the positions of 85 locations of tram stops across Nottingham (not so scale). There are 3.946×10^{124} possible solutions to this problem; hence, no provably optimal solution has been found to it. Your task is to create a new hyper-heuristic which, given 1 minute of computational time, can find tours which on (mean) average, are shorter than the shortest route which can be found by the nearest-neighbour constructive heuristic method. **The mean average must be calculated over a minimum of 11 runs.**

When designing your hyper-heuristic, you should answer the following points:

(1) Briefly explain what you did to design your hyper-heuristic.
(2) Briefly explain why you have chosen the particular components for your hyper-heuristic:
    a. Heuristic selection strategy,
    b. Move acceptance method, and
    c. Set of low-level heuristics considered.
(3) Briefly explain how your hyper-heuristic works. Focusing on parts 2a and 2b.

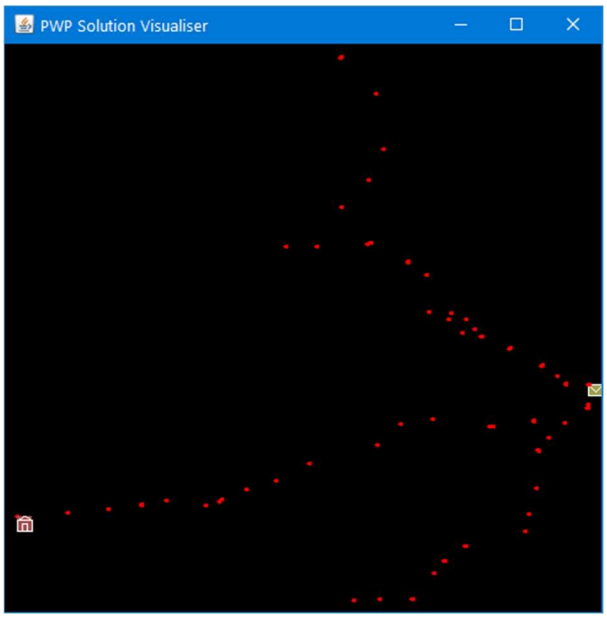

Figure 1 - 85 locations of tram stops in Nottingham with no known provably optimal tour.

The nearest-neighbour algorithm is a constructive heuristic which starts at a random location, and iteratively adds the location which is nearest to the previously added location until all locations are present in the tour. To conduct our experiments, all locations were chosen exactly once forming a total of 83 "trials". The best solution found was of length $5.587318932\times 10^{-1}$ to 10 significant figures, and has the tour as shown in Figure 2. The results of all 81 trials are available in a spreadsheet which can be found on Moodle entitled "Tramstops85NN_Results.xlsx".

You should report the hyper-heuristic used, the average tour length over multiple trials, the length of the best tour that you have found, and the best solution to the problem into the spreadsheet entitled "Tramstops85.xlsx" to at least 10 significant figures. The solution can be printed using the utility method `SolutionPrinter.printSolution(List<Location> routeLocations);` where the solution is passed as an ordered list of `Location` Objects, and you should be able to copy and paste it into the relevant field in the spreadsheet.

Figure 2 - Best route for tramstops-85 found using the Nearest-Neighbour constructive heuristic with tour length 5.587318932× 10^{-1} (10 s.f.).

3 HYPER-HEURISTIC COMPARISON

3.1 RANKING COMPARISON

After completing the implementation part, you should evaluate the hyper-heuristic used to solve the PWP problem instance Tramstops-85 in the previous section and compare its results to the existing hyper-heuristic. Note that you will also need to evaluate the supplied SR_IE_HH to enable a fair comparison of both algorithms.

To compare both hyper-heuristics, you will be evaluating them on three PWP instances of very different sizes. These instances are:

1. Carparks-40 (forming the "small" sized instance),
2. Tramstops-85 (from question 2 forming the "medium" instance), and
3. Trafficsignals-446 (forming the "large" sized instance).

You should create a test framework (hint: look at how it's done in the lab exercises!) which evaluates each of the hyper-heuristics by performing 11 trials per instance for each of the three above instances with a time limit of 1 minute. Note that if you are using a mac, as the benchmark tool does not work, you should just use 1 minute.

You should compare both hyper-heuristics using a *pairwise ranking* method. This entails ranking each hyper-heuristic on a per-trial basis. **Note that this means that the trials should be paired; i.e. the initial solution for the same trial should be the same, but for different trials they should be different as determined by a seeded pseudorandom number generator.** In this case, there are only two hyper-heuristics so the best method will receive a rank and score of **0** whereas the worst method for each trial will receive a rank and score of **1**. The mean rank of each hyper-heuristic should then be calculated over all trials and instances and reported to **three significant figures**.

A spreadsheet is provided for you to report the results of your experimentation and is partially filled in for you; you will only need to fill in the cells of the spreadsheet that appear yellow. This question should be answered in the "Ranking Comparison" sheet within Comparison.xlsx.

Note that for SR_IE_HH, the heuristic set should be comprised of **mutation** and **local search** only. For your own hyper-heuristic, you may also use **crossover** heuristics if you have designed a hyper-heuristic that is multi-point based.

3.2 STATISTICAL COMPARISON (HINT – LOOK BACK AT THE LAB EXERCISES!)

You should use an appropriate statistical test at a 95% confidence interval to compare the performance of SR_IE_HH and your hyper-heuristic to test whether there is any evidence that they perform statistically significantly different from one another. You should repeat this test for each of the three problem instances, giving the results for each, concluding with whether any of the two hyper-heuristics tested performs significantly different from one another for solving different PWP problem instances and/or give any limitations of the statistical analysis that you are asked to use which may limit your conclusion(s).

You can find a list of online statistical tests here https://www.socscistatistics.com/tests/. You should report which test you have used (and which tool if you used a different software package), as well as the configuration of the test to answer your hypothesis.

A spreadsheet is provided for you to report the results of your experimentation and is partially filled in for you; you will only need to fill in the cells of the spreadsheet that appear yellow. This question should be answered in the "Statistical Comparison" sheet within Comparison.xlsx.

4 DEADLINE AND MARKS

The deadline for providing your report is **11/05/2020 – 15:00.**

You should submit a single PDF version of your report which should not exceed 2,000 words and not exceed 6 pages in length, along with the spreadsheet "`Tramstops85.xlsx`", and the hyper-heuristic comparison spreadsheet "`Comparison.xlsx`".