Programming Assignment

The due date: 10:00am, Tuesday, 31th of October 2017.

This assignment is worth of 15% of the total course mark.

1 Introduction

The goal of this programming assignment is to make you familiar with research and development work, yet give you experience in solving practice-oriented optimisation problems and writing technical reports. It deals with a variant of the Vehicle Routing Problem, called the Capacitated Vehicle Routing Problem with two-dimensional loading constraints, or 2L-CVRP for short. The problem requires determining the optimal set of routes, for a homogenous fleet of vehicles, to serve a set of customers with known demands of items. The designed routes must originate and terminate at the central depot, and totally satisfy customer demand. Each customer must be visited once by one vehicle only. The total demand of the customer set, covered by a route, must not exceed the maximum carrying load of the assigned vehicle. The 2L-CVRP problem models the case when the demand of customers consists of two-dimensional rectangular and weighted items. The aim of the problem is to generate a set of routes that respect the aforementioned constraints and also guarantee the feasible loading/unloading of the items into/from the vehicles. It is a practical problem that finds its application in the field of transportation logistics. From the theoretical point of view, this is an $\mathcal{NP}$-hard combinatorial optimization problem; it is unlikely that large instances can be solved to optimality in short computational time. The problem has been widely studied by Zachariadis et al. (2009); Leung et al. (2013); Duhamel et al. (2011); Fuellerer et al. (2009); Gendreau et al. (2008); Dominguez et al. (2014) and other academics and practitioners.

Because solving the problem exactly is a hard task, you are asked to develop and implement an approximate heuristic algorithm. It should solve the problem quite well in reasonable time, though optimality may be not achieved. To solve the problem, you may either propose your own algorithmic approach or adopt one from the literature. To study the existing techniques, you may refer to the papers attached. You are allowed to work in a **team of two students** on this assignment, moreover, we encourage you to do this to improve your teamwork skills. At the same time, we expect you to work cooperatively with your partner so that you both study from each other and share your ideas rather than just simply split the work. Both members must be familiar with every part of the project and its code. Individual work is also allowed, but will be treated in the same way as a team work.

2 Formal Problem Statement

The 2L-CVRP problem can be defined as follows. Let $G = (V, E)$ be an undirected graph, where V is the vertex set containing the central depot and n customers, and

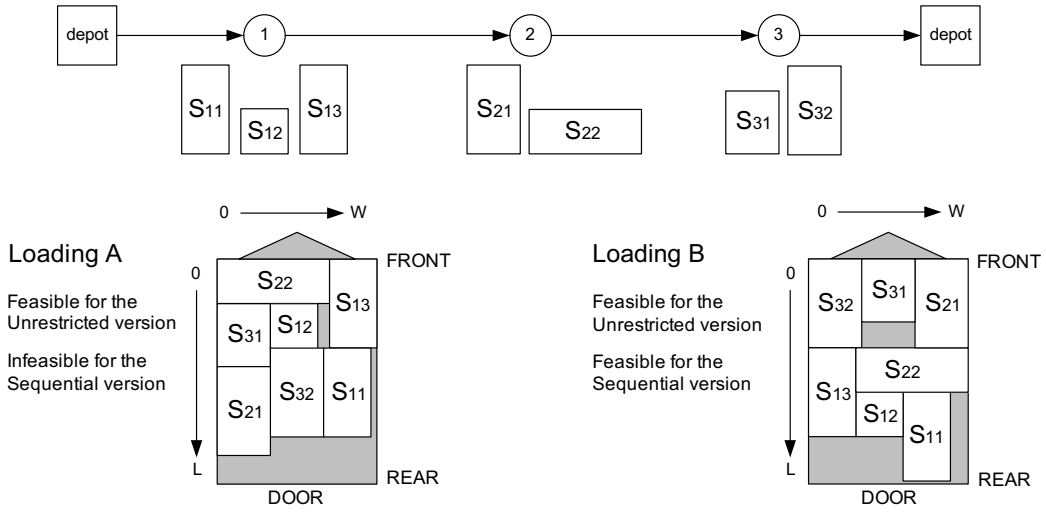

Figure 1: The *Unrestricted* and *Sequential* loadings.

$E = \{(i, j) : i, j \in N\}$ is the edge set. With each edge $(i, j) \in E$ is associated a distance d_{ij} from vertex i to vertex j. In the central depot, a fleet of q homogenous vehicles is available. Every vehicle has a maximum carrying load equal to C and a rectangular loading surface of length L and width W. The demand of each customer i, $i = 1, \ldots, n$, consists of a set S_i of m_i rectangular items of total weight B_i. Each item $s_{ai} \in S_i$, $a = 1, \ldots, m_i$, has length l_{ai} and width w_{ai}. The items must be placed on the loading surfaces without being rotated: their l- and w-edges must be parallel to the L- and W-edges of the vehicle surfaces, respectively. This constraint models the practical cases of automated, fixed orientation palette loading. The 2L-CVRP aims at determining the set of routes minimizing the total distance and satisfying the following constraints:

- the size of the generated route set must not exceed the number of available vehicles q (though less than q vehicles can be used in total);
- every route starts and ends at the central depot and employs one vehicle;
- the demand of every customer is totally covered;
- each customer must be visited only once;
- the total weight of all items demanded by the set of customers covered by a route must not exceed the capacity C;
- there must be a non-overlapping loading of all items demanded by the set of customers covered by a route into the $L \times W$ loading surface of the vehicle.

The 2L-CVRP problem has two versions concerning possible loading: the *unrestricted* 2L-CVRP and the *sequential* 2L-CVRP. The unrestricted version is modelled by the aforementioned six constraints, while for the sequential version of the problem an additional constraint, namely *Sequence Constraint* is imposed: the loading of the items must ensure that whenever a customer i is visited, all items in the set S_i can be unloaded by employing a sequence of straight movements (one per item) parallel to the length dimension of the vehicle surface. In other words, no item of customer j, visited after customer i, can be placed between items of customer i and the rear part (loading door) of the vehicle.

The sequence constraint arises in practice, when it is not feasible to move items inside the vehicle, due to their weight or fragility. In these cases, the unloading of every item must be accomplished without any repositioning of other items. Figure 1 compares the two versions of loading. In particular, it depicts a route visiting three customers. Customer 1 demands items s_{11}, s_{12} and s_{13}, customer 2 demands items s_{21} and s_{22}, and customer 3 demands items s_{31} and s_{32}. Let $(0, 0)$ correspond to the front left corner of the loading surface. The position of an item is expressed as the coordinate pair (x_{ai}, y_{ai}), where x_{ai} denotes the W-axis position and y_{ai} denotes the L-axis position of the left front corner of item s_{ai} of customer i. In *Loading A*, for example, item s_{22} is placed at $(0, 0)$. It is a feasible unrestricted packing, but it violates the constraint posed for the sequential version of the problem. To be precise, item s_{32} obstructs the unloading of items s_{12} and s_{22}, while item s_{31} obstructs the unloading of item s_{22}. On the other hand, *Loading B* is a feasible loading for both the unrestricted and the sequential versions of the problem. Items s_{11}, s_{12} and s_{13} are the first to be unloaded, followed by s_{21} and s_{22}, and finally s_{31} and s_{32}.

3 Specification

To succeed in this assignment, you must complete both of its parts: the programming part and the report. Lack of any part will cause zero scores awarded for this assignment. The first part asks you to develop an algorithm which solves the problem approximately with regard to the unrestricted and the sequential loading. Then, you need to implement and test your algorithm to ensure that solutions it returns are feasible and correct. Your program will be checked manually later on. In the second part, you need to evaluate the performance of your algorithm in terms of its quality and speed by running computational experiments on benchmark instances. For this purpose, 180 instances divided into 5 classes are attached. There are small and large instances in order to test your approach for different input sizes. Your next step is to compare your solutions with some state-of-the-art results. Using the comparison, you should be able to draw conclusions on the efficiency of your approach, decide on which test instances seem to be easier or harder to solve and why, and discuss on how your approach can be potentially improved. Finally, you must write a technical report. Though you are allowed to adopt the design of your algorithm by considering an example from the literature, you must report findings in your own language. Therefore, you may not straightly copy your algorithm's description from any paper or material to the report. You must present your best solution for each of the test instances. The feasibility of your solutions will be later checked automatically by the web-submission system. Ensure that they are correct, otherwise you may loose scores. To estimate the size of the report, set its font size to 12 pt and line spacing to 1. The report must be submitted in the PDF format.

Exercise 1 *Writing the abstract (worth of 15%)*

You must start your report with an abstract. This should give you skills in explaining key ideas briefly. First, you need to outline your algorithm on a conceptual level and talk about the applied techniques. It can be worth to mention your approaches to handle the loading component. You then need to summarily explain the idea behind the generation

of the route set. It should be done on a high level so that readers get an idea what you do, but still need to read the rest for particular details. Therefore, reporting algorithmic strategies and the way how you construct solutions (iteratively, sequentially, randomly are some keywords) is helpful. You may start with your loading strategy, discussing it in 3-4 sentences, then spend another 3-4 sentences to cover your whole approach. You should finally mention the results of computational experiments along with the main outcomes of your project. Here, you should briefly describe the performance of your algorithm compared to existing approaches from the literature. In total, the abstract should not exceed a half-page in size. It should be concise and comprehensive.

Writing an abstract is an art. Indeed, it could be easier to start with a very raw draft and return back to it when all the subsequent sections of the report are finished.

Exercise 2 *Addressing the unrestricted packing component (worth of 10%)*

In this section, you need to explain your strategy to resolve the unrestricted loading problem. You may find that the two-stage problem-solving strategy is commonly applied for the 2L-CVRP in research literature. This assumes solving the routing part first, then checking its feasibility regarding the loading constraints. Specifically, the first stage allocates a number of vehicles, assigns customers to the vehicles, and orders the customers to build the routes. The second stage is usually implemented in the form of a subroutine, which solves the loading decision problem for each route accepting the corresponding items as an input. If loading is feasible for every route, then the whole solution is valid. Thus, it is very likely that your approach to the 2L-CVRP will follow the same way. Therefore, your task is to develop the subroutine, which you think is to be efficient, then describe it in the report. In fact, the subroutine should solve the decision version of the so-called Two-Dimensional Bin Packing Problem, which has various heuristic solutions you may find in the internet.

To explain your algorithm for the loading component in a proper way, you should use a pseudocode or a diagram accompanied with a text description. You need to write 1-1.5 pages in total. You may feel it rather difficult to start your report and introduce your algorithm. If so, have a look at the papers attached first. Many of them explain similar loading (packing) strategies, and you may adopt their style and notation. Again, your description should be concise yet significant for a reader to understand and reproduce your algorithm. You may also discuss the complexity of your subroutine as a function of given items and figure out potential methods to speed up its work.

Exercise 3 *Addressing the sequential packing component (worth of 10%)*

Similarly to the previous exercise, you need to detail your strategy for the sequential loading problem.

Exercise 4 *Solving the 2L-CVRP problem (worth of 20%)*

Here, you need to detail your approach for the 2L-CVRP. The structure of your algorithm should dictate how to do this. For example, you may consecutively describe all the steps first, then specify every step. If the approach starts with an initial or a partial solution,

you should mention this fact as well as show how such solution is to be generated. When the approach is iterative, you need to clearly explain the iterative step and specify the conditions which terminate its work. When the approach is recursive, explain its recursive step and note the base case. If the algorithm is randomised, explain all the potential actions it may undertake. You also must address all the control parameters, if any are utilized in the approach. The way how the algorithm constructs solutions must be comprehensive. Again, pseudocodes, diagrams, and schemes are to be used to augment the text description. You need to write at most two pages.

Exercise 5 *Running computational experiments (worth of 25%)*

This is to be the last section of your report which requires to evaluate the performance of your algorithm for the 2L-CVRP. Your task is to apply your algorithm to every instance of the provided benchmark suit. If your algorithm is deterministic, i.e. it always returns the same result for a fixed input, you may compute each instance only once. Alternatively, your approach may be randomised, thus returning different results for a same input. In this case, you have to run your algorithm for each instance multiple times; 10 times should be enough assuming that computations might be costly. For each instance, you then have to report the best, the worst, and the average objective values that have been obtained. Proceed similarly to calculate the best, the worst, and the average running times.

Compare your results with the state-of-the-art. You can find results of other approaches reported in the tables of the attached research papers. Your task is to compare with at least one another algorithm. You should then form a table where the objective values of your approach and of those you compare with are reported. You also need to publish the relative gaps between the objective values of your solutions and the best known results. The gap can be calculated as follows: $100 \times \frac{ALG - ALG^*}{ALG^*}\%$, where ALG represents an algorithm you evaluate and ALG^* is the smallest objective value across the set of all algorithms you compare with (including your one). You are not expected to beat the existing results (though it would be fantastic), but you should strive to achieve the best possible solutions.

Analyse the performance of your approach. Try to identify classes of the instances where it seems superior and think about factors that affect its behaviour. Build a series of diagrams which illustrate the dependence of solution quality on the problem size and class. Similarly, draw diagrams to express the increase of the running time as the problem size increases. Find out whether your approach runs faster for a certain class of the instances. Finally, summarise your findings and discuss the pros and cons of your algorithm that you may think about.

Exercise 6 *Testing your solutions (worth of 20%)*

Submit your best solution as a text file for each of the test instances. The format of the text file and submission instructions are provided below. In addition, submit your source code. Your code will be manually checked for whether it performs the same algorithmic approach as you report. Therefore, provide some high level comments for your methods and classes along with comments for important parts of your code. You will not get any

marks for style and commenting, but we should be able to verify that your approach matches your code. Your solution files will be tested automatically by a marking script to verify their feasibility.

4 Input File Format

2L-CVRP_test_instances.zip is a zip archive file available on the CANVAS web-page of the assignment. It stores 180 test instances for the 2L-CVRP problem as text files of the following structure.

```
Instance: ***
Class: ***
   *** --- number of customers (excluding the depot)
   *** --- number of vehicles
   *** --- number of items
Capacity - length - width of vehicles
   ***    ***    ***
Node - x - y - total weight of the demand
  0    ***    ***    ***
  1    ***    ***    ***
 ...
Node - number of items - l - w for each item
  0    0
  1    ***    ***    ***    ***    ***
 ...
```

Each instance file is a series of records structured as lines of text. There are five different sections to specify the name of an instance, its class, the total number of vehicles, customers and items, parameters of the vehicles, coordinates of the customers, and features of the items in demand. The parts of each line are separated by one or more space characters.

Instance section declares the name of an instance.

Class section determines the problem's class, an integer number between 1 and 5. The class characterises the way of generation of the item set. In Class 1, with each customer is associated a single item of width and length equal to 1. The problems of Class 1 are in fact pure Capacitated Vehicle Routing Problem instances, as every customer sequence is feasible in terms of the loading constraints of the problem. They are used to test the algorithmic effectiveness in terms of the routing aspect of the problem. In Classes 2-5, for each customer i, a set of m_i items has been generated. Each item is classified into one of the three shape categories, namely vertical, homogeneous and horizontal, with equal probability. The dimensions (width and length) of an item are uniformly distributed into the ranges determined by this items shape category. To get more details on the instance classes, you are referred to (Gendreau et al., 2008). Each class consists of 36 instances containing from 15 to 255 customers and from 15 to 786 items. The rest of the Class section details the number of customers (excluding the central depot), the number of vehicles, and the total number of items.

Capacity section describes the capacity C, length L, and width W of the vehicles. L and W of the loading surface are equal to 40 and 20 across all the test instances, respectively. There is a guarantee that the provided fleet is enough to transport all the items to customers for both the *unrestricted* and the *sequential* versions of the problem. That is, you are given enough vehicles of sufficient capacity and size.

Node (x, y, demand) section provides coordinates for each of the customers (and the depot) along with the total weight of demanded items. The distance between any two vertices is an Euclidean distance (no rounding).

Node (number of items, l, w) is the last section which specifies the number of items for a specific customer along with the length and the width of each item associated with it. The first record refers to the depot, therefore has no items.

5 Solution File Format

Your algorithm must produce solution files of the following structure.

```
Instance: ***
Class: ***
Objective value: ***.**
Number of vehicles: ***
Routes:
0 index_i *** index_j 0
0 index_k *** *** index_l 0
...
Packing:
Node - number of items - h - w for each item
1 m_1 x_11 y_11 x_12 y_12 *** ***
2 m_2 x_21 y_21 x_22 y_22 ***
...
```

Instance and **Class** records must contain the name and the problem's class as defined by the original input file, respectively. **Objective value** reports the cost of the solution, which is a floating-point number rounded to two decimal places. **Number of vehicles** records how many vehicles have been utilized in your solution in fact. **Routes** section describes each of the routes computed by your algorithm. Each record must start and end with 0; that is, every vehicle must depart from the depot and return back to it. All other integers of the record correspond to customers assigned to the route. They must appear in the order as visited by the vehicle. For example, the second integer represents the city visited right after departure from the depot. All integers of the record must be space delimited. Note that the total number of records here must be equal to the number of vehicles used by the solution. **Packing** section considers nodes (customers) in ascending order starting from the first one. Each record specifies the index of a node, the number of requested items, and the coordinates of each item as computed by your algorithm. The coordinates must appear with respect to the front left corner of a vehicle which sets the $(0, 0)$ point as depicted in Figure 1.

6 Submission instructions for the report and programming code

You must submit your report in the PDF format and name the file as *report.pdf*. On the first page of the report, you must give a title and mention the names and IDs of the authors. If the report remains unnamed, the submission **will not be marked**. You must also submit all your *.cpp files* and a *makefile* named as *make_assignment* that compiles your program. Finally, submit the best solution found for each of the 180 test instances of the benchmark suit. The name of a solution file must consist of the name of the original instance followed by "*_U.sol*" postfix if the loading is unrestricted, and by "*_S.sol*" if the loading is sequential. For example, if the input file name is "*2l_cvrp0205.txt*", then *2l_cvrp0205_U.sol* is to be a solution for the problem with unrestricted loading. In total, exactly 360 solution files are expected. Check what you submit.

To submit your files, type the following command, all on one line (replacing `aXXXXXXX` with your username):

```
svn mkdir --parents -m "PSSD"
https://version-control.adelaide.edu.au/svn/aXXXXXXX/2017/s2/pssd/assignment
```

Then, check out this directory and add your files:

```
svn co https://version-control.adelaide.edu.au/svn/aXXXXXXX/2017/s2/pssd/assignment
cd assignment
svn add report.pdf
svn add *.cpp
svn add make_assignment
svn add *.sol
svn commit -m "assignment solution"
```

Next, go to the web submission system at:

https://cs.adelaide.edu.au/services/websubmission/

Navigate to *2017*, *Semester 2, Adelaide*, *Problem Solving and Software Development*, then *Assignment*. Click *Make a New Submission for This Assignment* and indicate that you agree to the declaration. The script will then check whether your code compiles. You can make as many resubmissions as you like.

References

Dominguez, O., Juan, A. A., & Faulin, J. (2014). A biased-randomized algorithm for the two-dimensional vehicle routing problem with and without item rotations. *International Transactions in Operational Research*, *21*, 375–398.

Duhamel, C., Lacomme, P., Quilliot, A., & Toussaint, H. (2011). A multi-start evolutionary local search for the two-dimensional loading capacitated vehicle routing problem. *Computers & Operations Research*, *38*, 617 – 640.

Fuellerer, G., Doerner, K. F., Hartl, R. F., & Iori, M. (2009). Ant colony optimization for the two-dimensional loading vehicle routing problem. *Computers & Operations Research*, *36*, 655 – 673.

Gendreau, M., Iori, M., Laporte, G., & Martello, S. (2008). A tabu search heuristic for the vehicle routing problem with two-dimensional loading constraints. *Networks*, *51*, 4–18.

Leung, S. C., Zhang, Z., Zhang, D., Hua, X., & Lim, M. K. (2013). A meta-heuristic algorithm for heterogeneous fleet vehicle routing problems with two-dimensional loading constraints. *European Journal of Operational Research*, *225*, 199 – 210.

Zachariadis, E. E., Tarantilis, C. D., & Kiranoudis, C. T. (2009). A guided tabu search for the vehicle routing problem with two-dimensional loading constraints. *European Journal of Operational Research*, *195*, 729 – 743.