**Zeus VRPTW**

User Manual

*December 4, 2014*

**Slippery Rock University**

Eric McAlpine [ewm1001@sru.edu](mailto:ewm1001@sru.edu)

Marc Sensenich [mas1029@sru.edu](mailto:mas1029@sru.edu)

Aaron Weckerly [ajw1002@sru.edu](mailto:ajw1002@sru.edu)

**Table of Contents**

**1 Zeus VRPTW**

1.1 Setting up Zeus

1.2 Running Zeus

**2 Selection/Insertion Classes**

2.1 Selection Classes

*2.1.1 Closest Euclidean Distance*

*2.1.2 Smallest Polar Angle To Depot*

*2.1.3 Time Oriented Nearest Neighbor*

*2.1.4 Unrouted Customer with Earliest Deadline*

*2.1.5 Unrouted Customer with Furthest Distance From Depot*

2.2 Insertion Classes

*2.2.1 Linear Greedy Insert*

*2.2.2 Insertion Criterion I*

*2.2.3 Insertion Criterion II*

*2.2.4 Insertion Criterion III*

*2.2.5 Savings*

*2.2.6 Savings Time Oriented*

*2.2.7 Sweep Insert*

2.3 Switching Active Classes

**3 File I/O**

3.1 Input Files

*3.1.1 Location*

*3.1.2 Formatting*

3.2 Output Files

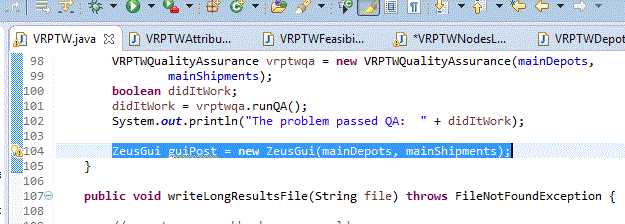
*3.2.1 Location*

*3.2.2 Formatting*

**1. Zeus VRPTW**

Zeus is a Java implementation that uses various heuristics to solve the vehicle routing problem with time windows. It utilizes a selection-insertion model to route different points amongst the routes.

**1.1 Setting up Zeus**

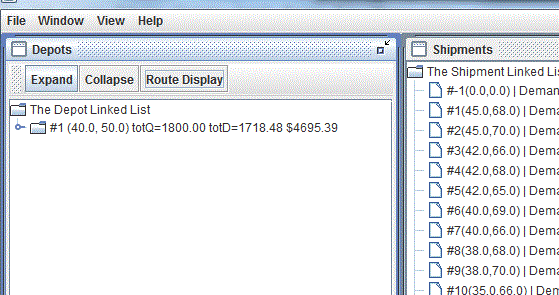
1. Copy the workspace file with all containing files to your own directory
2. Open the workspace with a Java IDE
3. Ensure that the GUI is enabled in the VRPTW.java file

\*Should contain *ZeusGui guiPost = new ZeusGui(mainDepots, mainShipments);*

1. Ensure data file structure is in place. Refer to page (file i/o)

**1.2 Running Zeus**

1. Choose the Selection Type you wish to use. Refer to section 2.1 for more information
2. Choose the Insertion Type you wish to use. Refer to section 2.2
3. Click Run
4. After a few moments, A window should launch looking similar to this:



\*This is an example of what Zeus looks like upon loading. Clicking Route Display will show what the routes would look like on a map.

1. **Selection/Insertion Classes**

Following the Selection-Insertion model, there are are two main functions of Zeus: choosing which customer to service next, and choosing how to service that customer. The selection classes are in charge of choosing which customer is to be routed next, while the insertion classes choose how to route the selected customer. These classes are all different from each other, but they all follow a central set of rules: The customers are all to be serviced once. The demand for each route cannot exceed the truck’s capacity. A truck must arrive at the customer before the end of the time window to service it.

**2.1 Selection Classes**

Zeus comes equipped with multiple selection classes that change the order in which the shipments are routed, which can alter the way the customers are chosen. Each class will choose every customer once, and no more than once.

**2.1.1 Closest Euclidean Distance**

Closest Euclidean Distance chooses customers based on their vicinity to the first depot defined in the input file. Vicinity is calculated using the standard distance formula using the coordinates for both the depot and each customer. For these data sets, there should only be one depot. This method does not account for time windows nor clustering.

Class Formatting: ClosestEuclideanDistance()

**2.1.2 Smallest Polar Angle to Depot**

Smallest Polar Angle simulates taking a line, originating from the depot and projecting it due East, or positive x direction, and sweeping it counterclockwise around the space with the customers in it. The customers are selected to be routed in the order that the selection line would encounter them. This method also does not account for one customer’s relative spatial or time orientation to any other customer, but simply selects based on the coordinates of each point.

Class Formatting: SmallestPolarAngleToDepot()

**2.1.3 Time Oriented Nearest Neighbor**

Time Oriented Nearest Neighbor selects customers based on their vicinity to the previously selected customer, starting with the depot. This class accounts for time windows by using it as a third dimension of vicinity. If the x and y coordinates are a plane, the time window dimension can be thought of as depth added to that plane. The distance between the points are calculated by using a three dimensional distance formula using the time window as well as the standard x and y coordinates.

Class Formatting: TimeOrientedNearestNeighbor()

**2.1.4 Unrouted Customer with Earliest Deadline**

Unrouted Customer with Earliest Deadline sorts the customers based on the end of their time window. It then selects based on this order, starting with the customer with the time window that closes the earliest. This class ignores all other information about the customers, such as demand, coordinates, and the beginning of the time window.

Class Formatting: UnroutedCustomerWithEarliestDeadline()

**2.1.5 Unrouted Customer with Furthest Distance to Depot**

Unrouted Customer with Furthest Distance to Depot sorts all the customers based on the simple two-dimensional distance between each customer and the depot. The data sets should contain only one depot. It then selects customers in order, starting with the customers that have the largest distance between to the depot. This class ignores all time window information.

Class Formatting: UnroutedCustomerWithFurthestDistanceToDepot()

**2.2 Insertion Classes**

The insertion classes are different methods of inserting a given customer into a route. The customers are chosen by the selection classes and are given to the insertion classes one-at-a-time. Each insertion class checks during routing that each modified route still meets the constraints defined earlier.

**2.2.1 Linear Greedy Insert**

The Linear Greedy Insert Class places the selected customer’s shipment into the trucks in order, creating a new truck if necessary, and stopping once it has found a truck that when the shipment is inserted, it meets the constraints listed. It will try to insert the shipment into the truck’s route in such a way that minimizes the cost increase, without rearranging the pre-existing order of the truck’s route.

Class Formatting: LinearGreedyInsertShipment()

**2.2.2 Insertion Criterion I**

Insertion Criterion I inserts the customers into routes based on criteria based on the distance between the customer to be inserted and the existing customers at different points along the route. This insert class does have direct access to the list of customers instead of strictly being passed one customer at a time by the selection class. The selection class will still dictate which customer is to be routed next.

Class Formatting: InsertionCriterionI(mainShipments)

**2.2.3 Insertion Criterion II**

Insertion Criterion II inserts the customers into their routes based on a set of criteria which is based upon the overall distance between the customers in the route and the customer to be inserted. This class has access to the list of customers but still obeys the selection-insertion model.

Class Formatting: InsertionCriterionII(mainShipments)

**2.2.4 Insertion Criterion III**

Insertion Criterion III inserts the customers into a feasible route based upon specific criteria derived from the overall increase time and distance between the customers already in the route and the customer to be inserted. Having access to the list of customers, this class still obeys the selection-insertion model and the rules listed above.

Class Formatting: InsertionCriterionIII(mainShipments)

**2.2.5 Savings**

The Savings class is supposed to insert every customer shipment into its own route, but this version merely simulates it by using a Linear Greedy Insert that also compares the cost increase to the cost of creating a new route. This cuts down on waiting time from the Linear Greedy Insert.

Class Formatting: SavingsInsertHeuristic()

**2.2.6 Savings Time Oriented**

The Savings Time Oriented is an adaptation of the Savings heuristic that accounts for time window information when calculating cost. This approach ensures that waiting time cannot comprise too large an amount of the cost increase.

Class Formatting: SavingsTimeOriented()

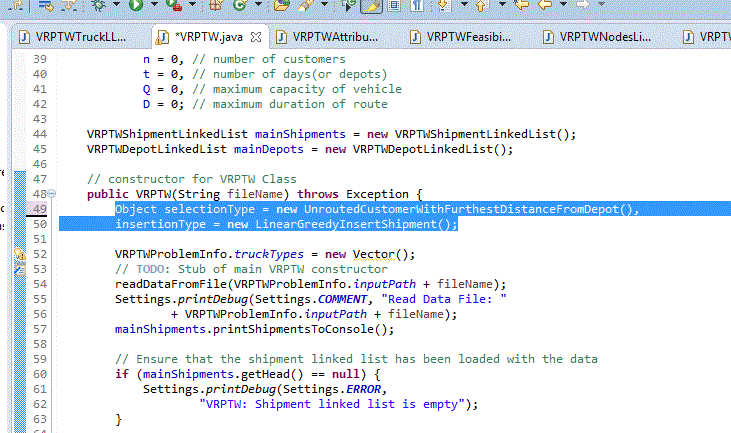
**2.2.7 Sweep Insert**

The Sweep Insert tries to place the selected customer at the end of a route and advances it through the order until it finds a feasible route. If no feasible order is found, the next route is loaded and the customer is put the process again, until a feasible route is found. Inserting into its own shipment is a last resort, but will always work, as is the case with every insertion class.

Class Formatting: SweepInsert()

**2.3 Switching Active Classes**

1. Open VRPTW.java
2. Near the top, find the line containing: *selectionType=* ***new***



1. Replace class definition with desired class formatting as defined above
   1. For selection class, use the line containing: ***s****electionType*

Ex. selectionType = new ClosestEuclideanDistance();

* 1. For insertion class, use the line containing: *insertionType*

Ex. insertionType = new LinearGreedyInsert();

1. **File I/O**

**3.1 Input Files**

**3.1.1 Location**

All input data files will be read from the *ExcelDataFiles* folder found in the workspace folder under the file path: *Zeus\data\vrptw\*

Inside the *ExcelDataFiles* folder, contains folders representing the different problem sets which can be specified in the VRPTW.java file. Each data set is to be given its own excel file within its problem set folder.

**3.1.2 Formatting**

The formatting for the input files abides by the following rules:

1. All data is to be on the first sheet
2. The first row contains the labels for the data fields for the depot.
3. No units are to be included in the data files
4. The second row contains the data corresponding to the depot in the order:
   1. Cell 1: X coordinate
   2. Cell2: Y coordinate
   3. Cell 3: Nodes/Customers to be routed by this depot (the only depot)
   4. Cell 4: Depot index #
   5. Cell 5: Maximum Capacity allowed for each truck
   6. Cell 6: Maximum Travel time allowed for each truck
5. The third row contains the labels for the data fields for the customers
6. The remaining rows contain data for the customers, one per row
   1. Cell 1: X coordinate
   2. Cell 2: Y coordinate
   3. Cell 3: Demand of the shipment
   4. Cell 4: Node identification number
   5. Cell 5: Beginning of time window
   6. Cell 6: End of the time window
   7. Cell 7: time required to service customer

**3.2 Output Files**

**3.2.1 Location**

The output files will be located in the *Results* folder located in the */Zeus/data/vrptw/* directory. Each data set will be given its own long and short results file. The long results file contains all the data about the finished routes, whereas the short data file contains a summary of the trucks.

**3.2.2 Formatting**

All output files will obey the following rules:

1. All results/output files will be in an excel spreadsheet (.xlsx file extension)
2. The first row contains the labels for the depot information
3. The second row contains the depot information in the following order
   1. Empty
   2. Total Demand handled for all the trucks
   3. Total Distance covered by all the trucks
   4. Time Spent in transit for the trucks
   5. Total Cost of all the trucks
   6. Maximum Travel Time allowed for each truck
   7. Total number of Trucks used
4. The third row begins truck information
   1. For Short results files: each truck on its own line
   2. For Long results files: each truck is followed by each customer in its route
   3. Customer data is in route delivery order and contains all input data as stated in 3.1.2
   4. Customers also contain distance from previous customer