UNIVERSITY OF BATH

User's Manual for VRP Spreadsheet Solver

Güneş Erdoğan
University of Bath

Download URL: http://verolog.deis.unibo.it/vrp-spreadsheet-solver
Video tutorial: www.youtube.com/watch?v=enCBp2lBn64

Developed by Dr. Güneş Erdoğan, 2015.
School of Management, University of Bath.
Member of CHI2.

TABLE OF CONTENTS

1. INTRODUCTION

The objective of this document is twofold. First and foremost, I aim to describe the functionality of the associated Microsoft Excel workbook, “VRP Spreadsheet Solver”. Secondly, I aim to introduce the professionals in the field of logistics to the existing literature on Vehicle Routing Problems (VRP), which has been built around their problems and yet they largely ignore. However, it would be unjust to blame them for this phenomenon. To the best of my knowledge, there is no standard data format for stating, solving, and visualizing VRPs. Without a standard, every VRP should be tackled on its own, with the results getting lost in the ocean of information that is the Internet. Why, then, nobody ever established such a standard?

On one hand, the standard software for small to medium scale quantitative analysis for businesses has been established as, arguably, Microsoft Excel. On the other hand, most academics develop solution algorithms in C++ and the resulting codes are not for the faint of heart. Distance and driving time data have to be retrieved from a Geographical Information Systems (GIS) database, which requires investment. The results of the algorithms are usually represented as a single value, the total cost, and it can only mean so much. It is not straightforward to manually find a solution of a VRP, much less so to compute its cost or to visualize it. Hence, constructing a tool for the data sources, solution algorithms, and visual representation of the results is a problem on its own.

I have noticed the need for this tool during VeRoLog 2013, an annual conference for EURO Working Group on Vehicle Routing and Logistics. In a panel that was being held at the conference, I asked about the possibility of building such a tool to the eminent names of research and practice in the field of logistics. I was immediately silenced by a terse answer: “It is impossible.” I am an engineer by training, and as Scott Adams insightfully stated in his book `The Dilbert Principle’: “The fastest way to get an engineer to solve a problem is to declare that the problem is unsolvable.” Thus I went down the rabbit hole to the Wonderland of Excel, Visual Basic for Applications (VBA), and public GIS.

Excel has been around for far too long, and has been used far too widely and frequently to be ignored. It is being taught at the management / business schools of many universities as of the time of this writing. However, as mathematicians despise engineers who dabble in mathematics, engineers belittle management people who dabble in quantitative methods and consequently their tool of choice. As an engineer teaching in a management school, I was ambivalent about the issue for some time. In the end, my practical side took over and I realised that the setup time required for solving a small to medium scale problem with Excel is far less than developing a full software solution. In my humble opinion, Excel is very flexible - within its limits. If you aim to do something with Excel and you find yourself unable to do so, you always have the option of asking an expert. Many discussion boards exist on the Internet, populated with altruistic people that will usually offer a solution within a day or two. In fact, most of the questions have been already asked and answered. If they cannot answer, the solution possibly does not exist, and you can only wish that Microsoft will extend the capabilities of Excel in the next version to handle your problem.

VBA is a programming language that is embedded within Excel (as well as other Microsoft Office products). Anyone with basic training in computer programming can write a *macro*, a small program in which you can utilise *for* and *while* loops, a feature that Excel spreadsheets lack. If you are using Excel, VBA is not too far away, it is just cleverly hidden. The Developer tab should be activated through the Excel options. Alternatively, pressing ALT+F11 opens up the VBA editor. On the upside, VBA gives you all the basic functionality and flexibility of a high level programming language, as well as access to all the data you are storing within the Excel workbook. On the downside, VBA cannot possibly compete with C++ in terms of efficiency. I did not find a formal benchmarking study, but the information I have gathered through a web search revealed that VBA is 3 times slower than C++ for basic arithmetic operations. My experience turned out to be much worse than that. There is a price to be paid for the flexibility, after all.

The capabilities of public GIS have significantly increased in the past few years. As of the time of this writing, Bing Maps as well as Google Maps have the *autocomplete* feature, and you can find the address you are looking for within 10-15 keyboard / mouse clicks. Although there are many more, I will be using three functions of the public GIS. *Geocoding* is the function that converts and address into the corresponding Latitude / Longitude values. *Directions* is the function that returns the distance and driving time between two points in addition to the directions. Finally, *Static maps* is the function that returns image files, which are defined by their centre point, zoom level, and their size. The leading public GIS at this point are Bing Maps and Google Maps. Both offer a limited free service, and extended services for subscribers. Within this study, I will be using free services of Bing Maps. Please consult their Terms of Service (http://www.microsoft.com/maps/product/terms.html) before using the VRP Spreadsheet Solver.

At the end of my little trip to the Wonderland, I have realised that it is indeed possible to construct a unified platform for representing, solving, and visualising the results of VRPs. The accompanying Excel workbook is the result of my efforts to build it. My research focuses on exact and approximate combinatorial optimization algorithms, and I am not an expert of software engineering (or Excel or VBA or GIS for that matter), and an expert could possibly build a much better solution. The resulting workbook is, all in all, a *proof of concept* for what can be done. I hereby advise you to go for a professional software solution if you have to routinely solve VRPs that involve more than 200 locations. The price you pay for open source software is the time you need to invest into modifying it to suit your needs. The readers are advised to keep the points made above in their minds as they progress through the rest of this document.

2. HOW TO SOLVE A VEHICLE ROUTING PROBLEM

VRP Spreadsheet solver has been designed for simplicity above all. Using the menu item “VRP Spreadsheet Solver” in the tab “Add-ins” (for Macs, it is under the “Tools” menu), you may issue the commands in their increasing numerical index, filling in data to each worksheet as it is generated. This menu is automatically generated when the file is opened, and deleted when it is closed. If the menu is not available for some reason, you can run the macro *SetupMenuItems* to reset it. A step by step guide is given below. Note that GIS based functions are not available for Mac computers, due to the unavailability of certain VBA libraries.

- Enter all relevant data in the “VRP Solver Console” worksheet.
- Execute “1.1 Setup Locations Worksheet” and enter the names, beginning and the end of the time window, service time, pickup amount, delivery amount, and profit of each location.
- Optionally, you may enter the addresses and execute “1.2 Optional – Populate Lat / Lon using addresses”. You need a Bing Maps key and an active internet connection to use this function.
- Optionally, you may execute “1.3 Optional – Sort locations alphabetically” for easier access through the solution worksheet (to be setup later).
- Execute “2.1 Setup Distances Worksheet”, and fill in the distance and duration data, either manually or by executing “2.2 Optional – Populate Distances Worksheet”. You need a Bing Maps key and an active internet connection to populate the distances using Bing Maps.
- Execute “3. Setup Vehicles Worksheet” and fill in the vehicle type names, capacity, cost, distance limit, work start time, driving time limit, working time limit, and fleet size.
- Execute “4. Setup Solution Worksheet”.
- Optionally, you may execute “5. Optional – Setup Visualization Worksheet”. All coordinates for the locations must be input for this worksheet to be created. You need a Bing Maps key and an active internet connection to have a static map as the background.
- You may want to manually solve your VRP from this point on, by choosing locations from the drop down menus (the green cells under the columns “Location name”) in the solution worksheet. Alternatively, you can execute “6.1 Engage the VRP Spreadsheet Solver” and wait for the run to end. The longer time you allow the solver, the better the result.
- Check the solution and modify it to suit your objectives. Optionally, you can execute “6.2 Optional – Feasibility Check” function to see if the modified solution is still feasible.
- Optionally, you may invoke an external solver you have developed and compiled into a Dynamically Linked Library (DLL). This is an advanced feature, requires above average programming knowledge. More details of how to build this DLL is given in Section 5.12. Unfortunately, this option is not available for Macs.
- The “Send feedback / ask a question” command will start an e-mail addressed at me. All (positive) feedback is welcome. Please provide details of your problem when you are asking a question.
- The “About” command will display the version of the workbook, the web address to download the latest version, and my contact information. Please cite the software and this manual in all projects they have been used.

3. SCOPE AND LIMITATIONS

Among many others, VRP Spreadsheet Solver operates on the following assumptions:

- The number of customers is limited to 200. This limit can be increased by editing the code, but you are advised against it since the efficiency of VBA does not allow the VRP Spreadsheet Solver to handle instances of that size. If you have built an external solver, you may attempt to solve larger instances, but the other features will be quite slow (GIS functions, setting up worksheets, etc.)
- The vehicles incur a fixed cost if they execute a route (may be zero), and a cost per unit distance (may be zero as well), but they do not incur any other costs.

- The travel distances and durations are fixed and known beforehand. Note that Bing Maps returns driving times for a car, which may be considerably shorter than that of a truck. You may need to modify the duration data accordingly.
- Every location other than the depot can be visited by at most one vehicle, at most once (no split deliveries). The visit must pick up all the supply and deliver all the demand, and partial service is not allowed. The supply, the demand, the profit obtained, and the amount of time a vehicle spends at a location are fixed and known beforehand.

4. STRUCTURE OF THE WORKSHEETS

VRP Spreadsheet Solver adopts an incremental flow of information, with subsets of data being kept in separate worksheets. Initially, the workbook only contains the worksheet named VRP Solver Console. The remaining worksheets, 1.Locations, 2.Distances, 3.Vehicles, 4.Solution, 5.Visualization, should be generated in the sequence denoted by their indices. The names of the worksheets are hardcoded within the VBA code, so you are strongly advised against renaming them. You are also advised against inserting or deleting cells (or columns or rows) in these worksheets. With a few exceptions, if a worksheet is modified then the worksheets with a larger index will need to be generated again, and the previous information will be permanently overwritten. If you would like to do a what-if analysis on the parameters, you are strongly advised to make copies of the worksheets before doing so.

The cells containing the data in the worksheets are colour-coded with the following scheme:

The cells with a black background are set by the worksheets and should not be modified.
The cells with a green background are parameters to be set by the user.
The cells with a yellow background are to be computed by the worksheets.
The cells with an orange background signal a warning.
The cells with a red background signal an error.

Note that some of the colour-coding features are not available for Excel 2007.

4.1 VRP Solver Console

This worksheet is central to the workbook, and **it should not be deleted**. If it is deleted, please run the macro *SetupConsoleWorksheet*, or close and reopen the workbook to generate it again. The parameters defined within the worksheet are described below.

Sequence: Instead of having a wizard interface, which is very easy to use but also very restrictive, the workbook numbers the worksheets in the order of progress. The parameters related to each worksheet are presented along with their sequence number. Please stick to the sequence unless you know what you are doing.

Bing Maps Key: Having a Bing Maps License is optional. You can still use the workbook without a Bing Maps License. A valid key is required for populating the Latitude / Longitude, the distances and duration, and for generating visualization of the locations and the routes on a map. You can

generate a free key at https://www.bingmapsportal.com/. Just copy and paste your key into cell C2 of the console worksheet.

Number of depots: Depots serve as starting and ending points for the vehicle routes.

Number of customers: This parameter does not include the depot(s), so it is the number of locations your vehicles are serving.

Distance / duration computation: This parameter describes how the distances should be populated, if they will be. The options are *Manual entry*, *Euclidian distances*, *Rounded Euclidian distances*, *Hamming (Manhattan) distances*, *Bird's flight distances*, and *Bing Maps driving distances*. The option *Manual entry* disables the distance population function. The option *Euclidian distances* computes the distance between points (x_1, y_1) and (x_2, y_2) as $d_{12} = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$, and the results of the formula are **not** in kilometres. The option *Rounded Euclidian distances* uses the Euclidian distance formula and rounds the result to the closest integer, and again the results of the formula are **not** in kilometres. The option *Hamming (Manhattan) distances* computes the distance between points (x_1, y_1) and (x_2, y_2) as $d_{12} = |x_1 - x_2| + |y_1 - y_2|$, and the results of the formula are **not** in kilometres. The option *Bird's flight distances* uses a spherical approximation for the surface of the Earth, and the results of the formula **are** in kilometres. This option is useful if you need routes for boats or aircraft instead of vehicles. The option *Bing Maps driving distances* uses the web service of Bing Maps, and the results of the formula **are** in kilometres.

Bing Maps route type: This parameter describes the type of route returned by Bing Maps. The options are *Shortest, Fastest*, and *Fastest - Real Time Traffic*. *Shortest* option will find the shortest path, which usually goes through city centres, is subject to strict speed limits, and often results in a very long duration. The recommended option is *Fastest*. The option *Fastest - Real Time Traffic* will give you estimates at the time the distances are populated, which may change drastically based on the traffic conditions at that instant.

Average vehicle speed: The results of the distance computation are divided by the value of this parameter to determine the travel time. Note that this value is not used when the distance population method is *Bing Maps*, which provides the estimated travel time of a car based on the speed restrictions on the path. You may want to modify the travel duration if your vehicles are considerably slower.

Number of vehicle types: For the purposes of this workbook, the differentiating factors between two vehicle types are the fixed cost per trip, the variable cost per trip, the distance limit, the work start time, the driving and working time limits, and the carrying capacity. If your fleet includes two vehicles of different model and make, yet with the same values for these parameters, then the vehicles can be considered to be of the same type. Most of the studies in the literature assume that all vehicles are of the same type, and problems involving multiple types of vehicles are called Heterogeneous VRPs (HVRP).

Vehicles must return to the depot? : The option *Yes* forces all vehicles to return to the depot at the end of their service. The option *No* relaxes this constraint, and is useful when the vehicles are outsourced and billed only for the service rather than the return, or when the vehicles

should follow the same trip in the same direction when return. The problems corresponding to the *No* option are called Open VRP in the literature.

Time window type: "Hard" time windows mean that a solution in which a vehicle visits a customer location outside the given time window is infeasible. With the "Soft" time windows, such a solution is undesirable but feasible.

Working time includes waiting time? : Waiting time occurs when a vehicle arrives at a location before the start of its time window. If you define "work" as driving and servicing, the waiting time does not count as "work". However, if you define "work" as time spent out of the depot, the waiting time does count as "work".

Visualization background: The options are *Bing Maps* and *Blank*. If *Bing Maps* is selected, the workbook will download the appropriate map containing the coordinates of the locations, and use it as the background image of the scatter chart depicting the routes.

Location labels: The options are *Blank*, *Location IDs*, and *Location names*. If *Location IDs* is selected, the ID number of the location will be displayed on top of to the location on the map. If *Location names* is selected, the name of the location will be displayed on top of the location on the map.

Warm start: If "Yes" is selected, the solution algorithm will attempt to use the existing solution on the solution worksheet as a starting point.

Show updates on the status bar: If "Yes" is selected, the solution algorithm will display the iteration number and the best solution value obtained in the bottom left of the Excel window. This takes a fraction of a second for every time it is displayed (i.e. every iteration), and may steal the CPU time from the solver. It is better suited for larges instances where fewer iterations will be performed, and the user has to wait.

CPU time limit: This parameter denotes the runtime limit of the VRP Spreadsheet Solver. As a general rule, a longer run gives a higher probability of finding a good solution. The VRP Spreadsheet Solver will not stop before completing the first iteration, which may be longer than the time limit provided. The solver can be interrupted by pressing and holding down the ESC key for a few seconds (sometimes you need to repeatedly press it, one of the joys of using Microsoft products).

4.2 1.Locations

The columns in this worksheet are explained below.

Location ID: This column is automatically generated, and must not be deleted or altered.

Name: The entries in this column must be unique. You may want to use the addresses in this column as well as the *Address* column.

Address: Please make sure that the address you provide is precise enough. For example the coordinates of "Cambridge" may resolve as those of "Cambridge, US**"** rather than those of "Cambridge, UK".

Latitude (y) : The letter in the parenthesis refers to the axis on which these values correspond to.

Longitude (x): The letter in the parenthesis refers to the axis on which these values correspond to.

Time window start: The beginning of the time window of visit for each location. Must be between 00:00 and 23:59.

Time window end: The end of the time window of visit for each location. Must be between 00:00 and 23:59.

Must be visited? : The options are *Must be visited*, *May be visited*, and *Don't visit*. The option *May be visited* leaves the choice to the solver, which decides based on the total profit. The option *Don't visit* simply takes the location out of consideration without altering the rest of the data. Note that for the first location (with ID 0) the value is *Depot*, which is generated by the system. The VRPs involving locations that may or may not be visited are called "Profit Collecting VRPs" in general. If a limit exists on the driving distance or time, they may be referred to as "Orienteering Problems" as well.

Service time: The amount of time a vehicle spends during a visit. The dwell time counts towards the working time limit, and should be less than or equal to the difference of the TW start and TW end.

Pickup amount: Denotes the amount to be picked up. The unit of the amount is unspecified, but must be the same for that of the capacity of the vehicles. Note that the items being picked up and delivered are assumed to be different types of items (e.g. full and empty gas cans), and are not substitutes of each other.

Delivery amount: Denotes the amount to be delivered. The unit of the amount is unspecified, but must be the same for that of the capacity of the vehicles. Note that the items being picked up and delivered are assumed to be different types of items (e.g. full and empty gas cans), and are not substitutes of each other.

Profit: The amount of monetary gain associated with visiting a location. The unit of the profit is unspecified, but must be the same for this parameter, the fixed cost per trip, and the cost per unit distance for the vehicles.

4.3 2.Distances

The columns in this worksheet are explained below.

From: The origin for the arc.

To: The destination for the arc.

Distance: The unit of the distance is kilometres if it is computed using *Bird flight distances* or *Bing Maps driving distances*. There are no units for using the Euclidian distance formula or the Manhattan distances.

Duration: Computed by dividing the distance by the average vehicle speed, unless the option *Bing Maps* is selected. As mentioned above, the Bing Maps option returns the driving time for a car rather than a truck, which you may want to modify.

4.4 3.Vehicles

The columns in this worksheet are explained below.

Vehicle type ID: This column is automatically generated and should not be deleted.

Type name: To better distinguish the types of vehicles.

Capacity: VRP Spreadsheet Solver supports only one dimension of capacity. This may be weight, volume, number of passengers, or whatever is relevant to your problem.

Fixed cost per trip: This cost is incurred if a vehicle of this type performs a trip. This parameter can be used for determining the fleet mix or "buy / hire" decisions.

Cost per unit distance: This is the cost incurred per unit distance this type of vehicle traverses.

Distance limit: This parameter denotes the maximum amount of distance a vehicle can be driven.

Work start time: This parameter denotes the time at which the vehicles depart from the depot.

Driving time limit: This parameter denotes the maximum amount of time a vehicle can be driven.

Working time limit: This parameter denotes the maximum amount of time a driver can work in a day. The dwell times count towards working time. The waiting times may or may not count towards working time, based on the option you choose.

Number of vehicles: The number of vehicles of the given type. Must be a nonnegative integer.

Note that for multiple depots, you need to enter the data for Vehicle type, Capacity, Fixed cost per trip, Cost per unit distance, and Distance limit of a vehicle type only for the first depot. These data will be replicated for the other depots. You may manually alter the replicated data if you need to.

4.5 4.Solution

For each vehicle a set of columns detailing its route will be generated. You may scroll right to see the routes of other vehicles. Column A also contains the *List of detected infeasibilities*, below the stops of the first vehicle. The columns in this worksheet are explained below.

Stop count: Number of stops the vehicle makes, including the return to the depot if it does.

Location name: This column is composed of cells with drop down menus, populated with the names of the locations rather than the indices.

Distance travelled: Cumulative distance traversed by the vehicle.

Driving time: Cumulative driving time of the vehicle.

Arrival time: Arrival time of the vehicle at the stop.

Departure time: Departure time of the vehicle from the stop.

Working time: The sum of the driving times and service times, up to and including the stop.

Profit collected: Cumulative amount of profit collected by the vehicle.

Load: Total amount (picked up and to be delivered) on board, at the time of departure.

4.6 5.Visualization

This worksheet is optional, and if generated it contains a scatter chart showing the locations and the routes of the vehicles. The chart can be formatted as any other Excel chart. The current design adopts a "less ink" approach. You can add axes and gridlines, or remove labels as you see fit.

5. FUNCTIONS

5.1 Optional – Reset the workbook

This is a quick way of deleting the data worksheets and renewing the console worksheet. Be careful, it is irreversible. Corresponds to the macro *ResetWorkbook*.

5.2 Setup Locations Worksheet

Corresponds to the macro *SetupLocationsWorksheet*.

5.3 Optional - Populate Lat/Lon using addresses

An active internet connection and a Bing Maps key are required for this function. Please make sure that the limitations of your key match with the size of your problem instance. An instance with n customers will require $n + 1$ queries to populate the whole set of coordinates. Corresponds to the macro *PopulateLatitudeAndLongitude*.

5.4 Optional - Sort locations alphabetically

The drop down menus of the solution worksheet are populated by the list of locations. To have alphabetically ordered drop down menus, you need to sort the locations. Corresponds to the macro *SortLocations*.

5.5 Setup Distances Worksheet

Corresponds to the macro *SetupDistancesWorksheet*.

5.6 Optional - Populate Distances Worksheet

An active internet connection and a Bing Maps Key are required for this function. Please make sure that the limitations of your key match with the size of your problem instance. An instance

with *n* customers will require approximately *n* * (*n* + 1) / 24 queries to populate the whole set of distances. Corresponds to the macro *PopulateDistances*.

5.7 Setup Vehicles Worksheet

Corresponds to the macro *SetupVehiclesWorksheet*.

5.8 Setup Solution Worksheet

Corresponds to the macro *SetupSolutionWorksheet*.

5.9 Optional - Setup Visualization Worksheet

Corresponds to the macro *SetupVisualizationWorksheet*.

5.10 Engage VRP Spreadsheet Solver

Corresponds to the macro *VRP_Solver*.

5.11 Optional – Feasibility Check

This function is supplied for checking the feasibility of the data and the solution after manual alterations. Corresponds to the macro *FeasibilityCheckDataAndSolution*.

5.12 Optional – Engage external solver

Using the .NET platform, you can build a DLL and invoke your own solution algorithm from VRP Spreadsheet Solver. Although this can be done in any language within the platform and any suitable compiler, I will only describe how it can be done using Visual Studio C++.

You first need a "def" file (e.g. vrp_spreadsheet_solver_def.def), which should read:

```
LIBRARY vrp_spreadsheet_solver
EXPORTS
   vrp_spreadsheet_solver
```

Your source file (e.g. vrp_spreadsheet_solver.cpp) should contain two "`struct`"s, one containing the instance data, the other containing the solver options. The `struct` definitions (including the `#pragma` statements) and the function definition are given below:

```
#include <stdio.h>
#include <stdlib.h>

#pragma pack(4)

struct instance_data
{
    bool open_vrp;
    double penalty;
    bool soft_time_windows;
    bool waiting_is_working;
    long num_depots;
    long num_customers;
    long num_locations;
    long num_vehicle_types;
};
```

```
struct solver_option_data
{
    double CPU_time_limit;
    double LNS_minimum_removal_rate;
    double LNS_maximum_removal_rate;
    long LNS_candidate_list_size;
    bool warm_start;
};

#pragma pack()

double __stdcall vrp_spreadsheet_solver(

instance_data &instance,
solver_option_data &solver_options,
long *time_window_start,
long *time_window_end,
long *mandatory,
long *service_time,
double *pickup_amount,
double *delivery_amount,
double *profit,
double *distances,
long *durations,
long *from_location,
long *to_location,
double *capacity,
double *fixed_cost,
double *variable_cost,
double *distance_limit,
long *work_start_time,
long *driving_time_limit,
long *working_time_limit,
long *number_available,
long *vertex_index,
long *origin_depot,
long *vehicle_type,
long *vehicle_index
)
{
   // your code

   return 0;
}
```

All array indices start from 0. The arrays `time_window_start`, `time_window_end`, `mandatory`, `service_time`, `pickup_amount`, `delivery_amount`, and `profit` are of size `instance.num_locations` and indexed based on the Location ID column of the locations worksheet, and denote the values of the corresponding entries in that worksheet.

The arrays `distances`, `durations`, `from_location`, and `to_location` are of size `instance.num_locations` * `instance.num_locations`. The distance from the location with index `from_location[i]` to the location with index `to_location[i]` is equal to `distances[i]` and the associated driving duration is `durations[i]`.

The arrays `capacity`, `fixed_cost`, `variable_cost`, `distance_limit`, `work_start_time`, `driving_time_limit`, `working_time_limit` and `number_available` are of size `instance.num_depots * instance.num_vehicle_types`. The value of `fixed_cost[i * instance.num_vehicle_types + j]` corresponds to the fixed cost of using a vehicle type with index *j* located at the depot with index *i*. The rest of these arrays can be interpreted similarly.

The arrays `vertex_index`, `origin_depot`, `vehicle_type`, and `vehicle_index` are of size `instance.num_customers`. These arrays are to be used for describing the solution to the problem instance. The data in these arrays should be **in the same order they appear in the solution**. Consider an instance that consists of two depots, two vehicle types with two available at each depot (8 vehicles in total), and 9 customers. Note that the depots are indexed as 0 and 1, and the customers are indexed from 2 to 10. Assume that the solution found consists of four routes:

1) First vehicle of type 0 located in depot 0, visiting customers with indices 3, 2, 4
2) Second vehicle of type 0 located in depot 0, visiting customers with indices 5, 6
3) First vehicle of type 1, located in depot 0, visiting customers with indices 8, 7
3) First vehicle of type 1, located in depot 1, visiting customers with indices 9, 10.

This solution would be represented as

```
vertex_index = [3, 2, 4, 5, 6, 8, 7, 9, 10]
origin_depot = [0, 0, 0, 0, 0, 0, 0, 1, 1]
vehicle_type = [0, 0, 0, 0, 0, 1, 1, 1, 1]
vehicle_index = [0, 0, 0, 1, 1, 0, 0, 0, 0]
```

If `solver_options.warm_start` is `true`, the function call will pass these arrays containing the solution in the solution worksheet, in the same format.

The DLL should be named *vrp_spreadsheet_solver.dll*. It will be accessible by the VRP Spreadsheet Solver if you place the DLL in the same directory as the workbook. You should incorporate the time limit within your code. Always save your file before engaging an external solver in order not to lose data.

6. SOLUTION ALGORITHM

The field of VRP research is ripe with many solution algorithms. The best known heuristic algorithm is arguably the Clarke and Wright (1964) heuristic. Many *metaheuristic* algorithms have been proposed in the last decade, including but not limited to Large Neighbourhood Search (LNS), Iterated Local Search, and Genetic Algorithms. The details of the literature would be beyond the scope of this document, and I do not claim that a single algorithm can successfully solve all variants of the VRP. Let it suffice to state that a variant of the LNS algorithm (Shaw, 1998; Pisinger and Ropke, 2007) is implemented within the VRP Spreadsheet Solver. An outline of the algorithm is given below.

Step 1 (Initialization): Initialize the incumbent solution, the best known solution, and the iteration counter *k* = 1. Read the solution on the solution worksheet into the incumbent solution if a "warm start" is required. Set α_1 = *LNS minimum removal rate*, α_2 = *LNS maximum removal rate*, and β = *LNS candidate list size*.

Step 2 (Stopping condition): If the time limit is exceeded, stop and report the best known solution.

Step 3 (Break): Randomly select and remove $\alpha_1 + U[0, 1] * (\alpha_2-\alpha_1)$ percent of the locations from the incumbent solution.

Step 4 (Repair): Randomly choose and insert a location among the best β candidate locations for insertion, until no more vertices can be inserted.

Step 5 (Polishing): Select and apply the best among the operators of *vertex relocation*, *vertex swap*, and 2-*opt*, until no further improvement is possible.

Step 6 (Best solution update): If the incumbent solution is feasible and better than the best known solution, update the best known solution. Increment *k* and go to Step 2.

7. Conclusion

Despite its shortcomings, I hope that the VRP Spreadsheet Solver will be used as a minor decision support system for small and medium enterprises as well as for teaching at undergraduate and postgraduate levels. I am almost sure that academics will disagree with some design choices, but they can be corrected. I am also quite certain that practitioners will find features that do not completely fit their needs, but it can provide you starting points that lead to an actual solution. Please send bug reports, suggestions, and (not-too-harsh) comments to G.Erdogan@bath.ac.uk.

REFERENCES:

S. Adams. *The Dilbert Principle*. Boxtree Ltd., 1997.

G. Clarke and J.W. Wright. Scheduling of Vehicles from a Central Depot to a Number of Delivery Points. Operations Research, 12:568-581, 1964.

D. Pisinger and S. Ropke. A general heuristic for vehicle routing problems. Computers & Operations Research, 34:2403–2435, 2007.

P. Shaw. Using constraint programming and local search methods to solve vehicle routing problems. In Proceedings of the 4th International Conference on Principles and Practice of Constraint Programming, pages 417–431, Springer, New York, 1998.