**Simulator Manual**

Trever Hallock

Tamlyn Harley

Matthew Stanley

Cody Zhang

CONTENTS

Purpose …………………………………………………………………………………………... 2

Getting Started …………………………………………………………………………………... 3

What Goes In ……………………………………………………………………………………. 4

*Cities …………………………………………………………………………………………………………. 4*

*Actions ……………………………………………………………………………………………………….. 5*

*Solutions ……………………………………………………………………………………………………... 6*

*Viability ……………………………………………………………………………………………………… 7*

*Translation ………………………………………………………………………………………………….. 7*

What Comes Out ………………………………………………………………………………… 8

Conclusion ………………………………………………………………………………………. 9

PURPOSE

The purpose of our simulator is to generate a random city / route to test for Sam’s Hauling, as well as provide a function that can take a solution and demonstrate its efficiency based on various metrics. The file is written using MATLAB, and this manual is intended to show how to use this program and deal with its output.

This program is a work in progress, and the team is eager to receive feedback! If anyone has any suggestions for improvement, or spots any errors, please feel free to either post this feedback to our Canvas discussion board, where we will work diligently to implement all of the changes needed or requested.

GETTING STARTED

To get started, one must simply go to our Canvas discussion page and download the .zip file that is located there. (This is constantly being updated, so make sure to get the latest version if changes are implemented). Then, if one does not have MATLAB installed on their computer, the computer lab has it on the workstations; additionally, MATLAB is sometimes offered free to students through the university.

Once downloaded, you will need to change the directory on MATLAB to reflect …\animated-dubstep-master\matlab, where our files are stored. This can be done through the MATLAB function 'cd' or by browsing through the current path on the left-hand panel of MATLAB.

Once in the correct directory, you should see several subfolders and a file called env.m. You must run this file for all of the subdirectories to be enabled, and this should be your first step; so, from the MATLAB command window, type “env” and it will enable everything for you. Thus, there are three easy steps:

1. **Download animated-dubstep-master.zip from Canvas, and unzip the files**
2. **Change the MATLAB path to reflect …\animated-dubstep-master\matlab**
3. **Run the “env” script in order to set the matlab environment**

This is all you need to get started simulating!

WHAT GOES IN

Now that you’ve installed the simulator, how does it work? Well, there are two main components to the simulator: Cities and solutions.

What are cities?

Cities are objects (if you don’t know this programming term, don’t worry, it isn’t necessary to understand) that store multiple pieces of information. The city stores all of the data about routes, stops, customer requests, landfills, staging areas, etc. It is an entire set of data that makes up the problem statement. The function generate\_city(R, L, Y, D) generates a city at random, based on these arguments:

**R is the number of customer requests to be generated**

**L is the number of landfills to be generated**

**Y is the number of staging areas, or yards, to be generated**

**D is the number of drivers that work in that city**

This function creates a random set of customer requests, landfills, staging areas, trucks, etc. for us to run a simulation on. Everything is random, so that you can create a diverse set cities to run simulations through. To get a visual overview of the city, simply use the display\_city function. So, in order to create a city and see what it looks like, follow these steps:

1. **From the MATLAB command window, type in c = generate\_city(R, L, Y, D) where you replace the arguments with numbers of your own**
2. **From the MATLAB command window, type display\_city(c)**

The visual output is not at all necessary to create a city, but it is helpful to understanding what was generated. *For those who understand object oriented programming, they can certainly create a city by hand manually, but there is a lot of data to be filled in.*

What are actions?

A city contains actions – also called stops – and the first thing anyone will notice with our simulator is that there are a *lot* of actions for any city. Why are there so many, and what are they?

An action is simply something a driver can do; that’s it. An action has five main parts to it:

**The operation –** this can be P (pickup), D (dropoff), R (replace), S (stage), U (unstage), or E (empty)

**The in-size** – this is the size of the dumpster that is being brought to the action or stop, as a numerical value between 0 and 5, where 0 represents no dumpster and the others represent the four sizes of containers we are dealing with

**The out-size** – this is the same thing as in-size, but it is what size of dumpster the driver is supposed to leave with

**The start**-**time** – this is the time, in seconds, when this action can start being performed; for example, if the start-time is 10,000, this means that 10,000 seconds must elapse during the simulation before this action can be performed.

**The stop-time** – this is the time, in seconds, when an action must be performed by; together with start-time, these model time window constraints.

Different areas have different kinds of actions:

**Staging areas** have 8 actions each, as shown in this table:

|  |  |  |  |
| --- | --- | --- | --- |
| STAGING AREA | Operation | In-size | Out-size |
| ACTION 1 | STAGE | 1 | 0 |
| ACTION 2 | STAGE | 2 | 0 |
| ACTION 3 | STAGE | 3 | 0 |
| ACTION 4 | STAGE | 4 | 0 |
| ACTION 5 | UNSTAGE | 0 | 1 |
| ACTION 6 | UNSTAGE | 0 | 2 |
| ACTION 7 | UNSTAGE | 0 | 3 |
| ACTION 8 | UNSTAGE | 0 | 4 |

This is because, for example, if you want to unstage (pickup) a size 3 dumpster, you must have an empty truck; thus, you would pick the seventh action, which means your in-size is 0 and your out-size is 3. *Why this is needed will become clear once a solution is given.*

Similarly, landfills have four actions each, one for each kind of dumpster you’ll be bringing there to empty. Customer requests only have one action each, as they will always have a predetermined in-size and out-size.

Now that we’ve covered cities, let’s look at solutions:

What are solutions?

A solution is a matrix; each row is a driver, and each corresponding column entry is what that driver will do in order. A negative one means that the driver does nothing (it is the end of his route). Here is an example:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| EXAMPLE SOLUTION MATRIX | | | | | | |
| 2 | 4 | 19 | 22 | -1 | -1 | -1 |
| 10 | 17 | 19 | 44 | 11 | 13 | 5 |
| 6 | 19 | -1 | -1 | -1 | -1 | -1 |

How is this interpreted? Well, as we said, each row is a driver; thus, this city has three drivers. Driver one performs action 2, then 4, then 19, then 22, and then he is done for the day. Similarly, driver three performs actions 6 and 19, in that order, then finishes his day. Driver two is the busiest, obviously, but his actions are just as easily read.

As you can see, it was necessary to encode what kinds of dumpsters the driver was dropping off, picking up, etc. with the in-size and out-size, so that this matrix would be feasible. **This encoding of the solution into a matrix is why there are so many actions associated with staging areas and landfills.** As an example, imagine that stop 19 is a staging area. Because of how the actions are structured, given any city, we will know exactly what kind of action the driver is performing at that staging area. Actions 18-25 may all be at that same staging area, but they all mean different things.

Viability

Some solutions will contain errors that the simulator will recognize as non-feasible solutions. It will return the variable feasible = false if this is the case. Many things, too many to cover, will cause it to error out; but, for example, if a driver visits a landfill followed by another landfill, this makes no sense and it will error. Or, if a driver visits a staging area to get a size 1 dumpster, but that staging area has none of those in inventory.

The simulator will continue to run to the best of its ability when dealing with a non-viable solution, but sometimes it will terminate. The errors it finds will be displayed in the main window of MATLAB.

Our suite of functions also comes with a generate\_rand\_solution function that takes a city as an argument; however, beware, this random solution is rarely viable.

Translation

This function takes the output from the user interface team (who are working with Excel) and translates it into our city structure in MATLAB. It also finds the coordinateness of the addresses, so that you can plot the city. This will be a work in progress until the UI team has finalized their output format.

WHAT COMES OUT

The simulate function takes in a city and a solution matrix as arguments, and outputs these metrics:

**Feasible** – this returns false if the solution encountered an error that makes it a non-viable solution; it returns true otherwise

**Times** – this returns, as a vector, all of the times it took each driver to complete his assigned route, based on the solution matrix given

**Distances –** this returns, as a vector, all of the distances traveled by each driver based on his assigned route

**Number Serviced** – this tells you how many of your customer requests were actually completed

**Fees** – this shows you how many fees you accrued on this route, based on the costs of landfills

**Inventories**– This lets you see how many dumpsters remain at each staging area at the end of the day

To run the simulator, simply create a city, your solution matrix, and from the main window of MATLAB call simulate(city, solution) with the variable names appropriately replaced.

CONCLUSION

I hope this gives everyone a good understanding of what our software can do; if you have any questions, feel free to contact our team through Canvas. Our team is comprised of the members:

Trever Hallock

Tamlyn Harley

Matthew Stanley

Cody Zhang

Thank you for your time, and have fun simulating! And remember – we’re here to help you!