Final Documentation

Senior Design Project

Sam Ruggieri – Swarm Routing

Contents:

I. Goals

a. Introduction

My earliest ideas for this project began to arise about a year ago, as I first became interested in swarm intelligence algorithms. I wrote a fairly trivial program to visually demonstrate how an ant colony optimization algorithm works, using GDI+ and VB.NET. After having some success in using a swarm of modeled swarm entities to determine an optimized path between points in unobstructed two-space, I decided to try to apply the principles of that algorithm to a different, more practical environment.

Specifically, my thought to use this sort of mechanism to route traffic was pieced together while sitting in traffic on the Schuylkill Expressway. From my office to my apartment, there are 4 main choke-points, one of which must be traversed to get home. If I took 76 across the river, about once a week I would hit smooth traffic and be home in 20 minutes. However, most of the time I would become ensnarled in some significant traffic jam and it would take closer to 40 minutes. If I took the East Falls Bridge, I would be home fairly consistently in about 30 minutes. Once in a while, though, an accident would occur in the vicinity of the bridge, and it would take closer to 40 or 45 minutes. My other alternatives, the Strawberry Mansion Bridge and driving all the way up to Girard, were both fairly impractical, as they would usually take about as long as if I hit a traffic jam of normal severity on one of the previously mentioned routes. However, I have used both successfully, having heard on KYW about massive traffic jams.

The crux of the problem is incomplete information. KYW news radio is fairly good at reporting the traffic situation on the expressway, but they often overlook the other routes completely. Sometimes, too, they are completely wrong about the traffic on the expressway, occasionally reporting jams when the route is clear, and vice versa. If a mechanism could be employed to distribute comprehensive information about current traffic patterns on all routes, individual vehicles could autonomously decide which route to take. As vehicles decide on routes that avoid congestion, the severity of the congestion in the areas that are jammed is mitigated and mean compute times would decrease. My primary objective was to build a simulation that would demonstrate this.

b. Specific Objectives

At the beginning of this project, I formalized a set of goals in a requirements and design document. Every goal was met and many were exceeded. The specific requirements that were initially defined are listed below, along with an explanation of how each requirement was fulfilled.

---

1. Requirement Category: Visual Component

    a. Requirement: The visual environment in which action will take place will be a two-dimensional grid. This grid will consist of individual squares, the colors of which will define the properties of each square.

       Implementation: To represent the visual environment, I used .NET's DataGridView control. Sufficiently fast, simpler than building visual classes using GDI+, and having a robust set of accompanying functions, use of this form control simplified the project as a whole.

b. The grid must be able to accommodate the simulated movement of the vehicles that traverse it.

   Implementation: Through experimentation, a way to refresh the grid often enough to show all movement without causing the program to appreciably slow down was devised (see calls to the Refresh_Map routine).

2. Environment / Simulation Modeling

   a. The environment will model a simulated town with a specific set of rules.

      Implementation: The visual representation of the town (the colored grid) models the different types of areas and roads. The rules for traversing the road can be found in the Vehicle.Move routine.

   b. The simulation will model an evening rush hour scenario. In this scenario, all vehicles will originate from road squares adjacent to zones designated as workplaces and travel to road squares adjacent to zones designated as living areas.

      Implementation: The logic behind the Run_Button, whose caption reads “Non-Optimized Solution”, contains all parts of the traffic-jam simulation. The mechanisms for defining paths, once all the visual objects have been converted to graphs, can be found in the Vehicle.Move routine if it has received a value of False in the Optimized parameter.

      The logic behind the Run_Optimized_Button contains all parts of the more effective routing mechanism. The methods for defining paths, once all the visual objects have been converted to graphs, can be found in the Vehicle.Move routine if it has received a value of True in the Optimized parameter.

   c. The environment must model a workplace zone.

      Implementation: The DataGridView itself serves as a model for the zones. The workplace zones are shown in blue.

   d. The environment must model a living zone.

      Implementation: The DataGridView itself serves as a model for the zones. The living zones, referred to in other documentation as residential zones, are shown in green.

   e. The environment must model roads.

      Implementation: Roads are shown visually as black tiles on the DataGridView control. The rules pertaining to movement along roads are contained in the Vehicle.Move routine.

   f. The environment must model intersections.

      Implementation: Intersections occur visually where two separate roads intersect. The rules for negotiating intersections are contained in the Vehicle.Move routine.

g. Vehicles must be modeled.

   Implementation: The Vehicle class models vehicles.

h. There must be a simple mechanism for users to design environments in which the simulation can be conducted.

   Implementation: The map editor tool, a separate program from the simulation, allows users to draw maps in a visual environment.

3. Traffic Jam Simulation

   a. For the traffic jam simulation, all vehicles must traverse a calculated shortest-path from their origin to their destination.

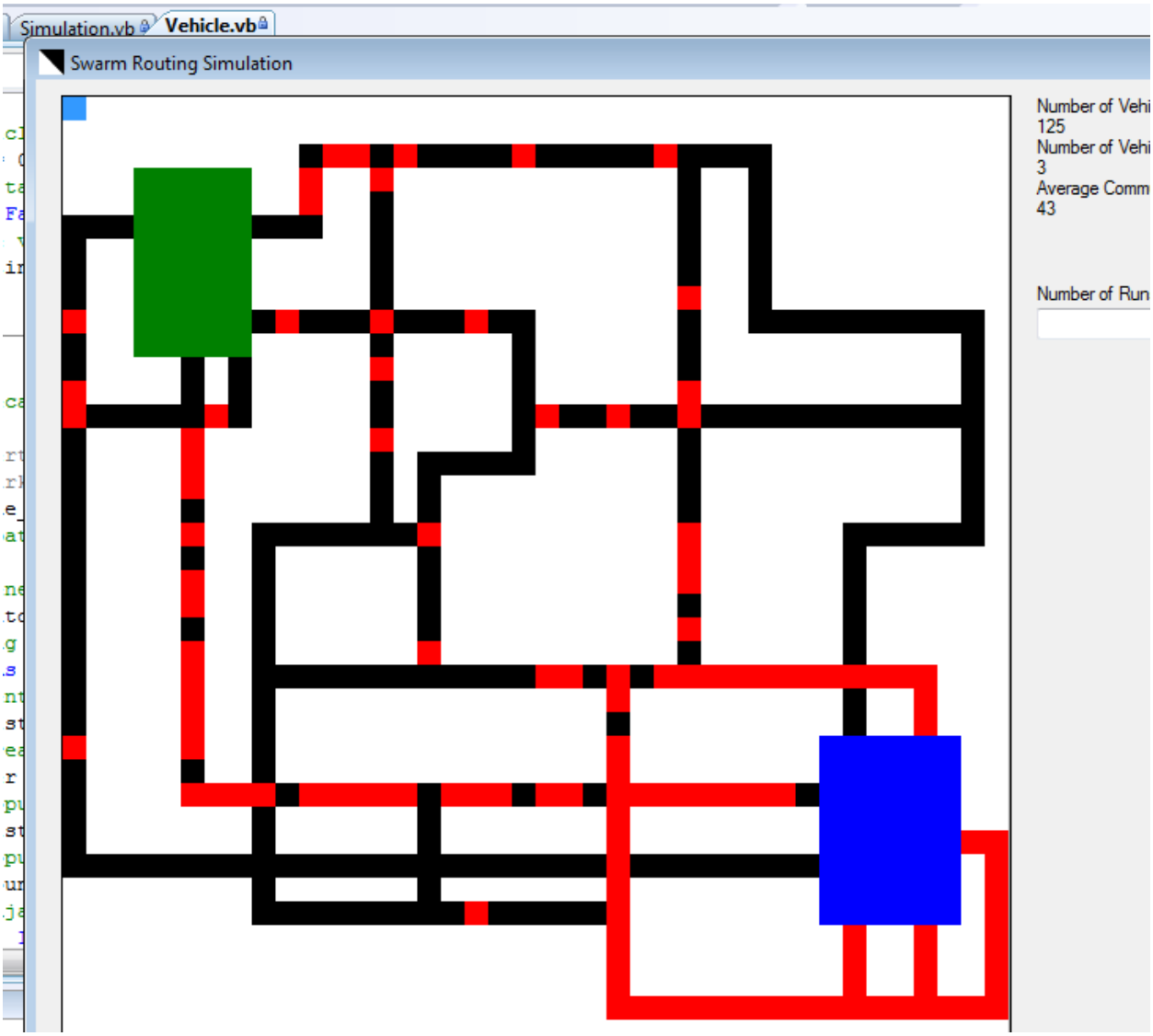
      Implementation: Logic for moving in the traffic-jam simulation can be found in the Vehicle.Move routine, given a value of False for the Optimized parameter.

      The constructor routine for the Graph object and the other subroutines that calls are used to convert the 2-dimensional grid into a graph of vertices and edges. Node, Vertex, Adjacent_Vertex, and Distance objects are used to model different portions of the graph in different situations.

      The Determine_Shortest_Path routine of the Vehicle class is used to determine the shortest path between the vehicle's source and destination using a list of vertices and edges derived from a Graph object.

   b. Traffic jams should occur, since only one vehicle at a time will be permitted to traverse an intersection. Traffic jams must be perceptible to the user by instances of vehicles (at least 2) stopped at intersections. This stoppage will elongate cumulative commute times.

      Implementation: This goal's success was gauged visually. The screenshot below shows a traffic jam occurring in a non-optimized run:



One can note the areas where vehicles are jammed (not moving), especially in the bottom-right corner.

c. The cumulative amount of time of each vehicle's commute must be maintained. An average commute time must be calculated.

   Implementation: Average commute times are calculated using global variables that are modified by individual vehicles. These times are reported to the user on the labels on the right of the form, and are saved to a CrLf delimited text file for looping runs.

4. Optimized Swarm Routing

   a. For the optimized swarm routing, each vehicle must autonomously make decisions to determine ideal routes. The vehicles should gather information about current traffic situations (such as backups around intersections) and use that information to determine routes that may be better than the shortest path.

Implementation: In many ways, the fundamental routing mechanisms are similar to those employed in the non-optimized logic. However, the presence of vehicles along routes is accounted for, and route lengths are augmented by a constant value for the presence of each additional vehicle. This allows vehicles to avoid congestion.

b. Again, vehicle commute times must be recorded and averaged. The ultimate criterion for determining the success of the project is obtaining better average commute times for the swarm routing simulations than for the traffic jam (shortest-path-only) simulations.

   Implementation: Average commute times are calculated in the same fashion as in the non-optimized solution. Lower average commute times were observed using the optimized method, and the results were statistically verified using a two-sample t-test.

II. Approach

a. Object Oriented Design / Documentation

To model the environment, I converted what can be easily drawn and understood by a user (the 2-dimensional grid control) into objects that represent components of graph theory. This dichotomy in the code in terms of what can easily be drawn and comprehended by the user and the mathematical components to which I applied the algorithms resulted in an aesthetically appealing application with robust and readily understandable code.

Careful thought was put into the design of both components of the application. Testing was done with the DataGridView control to make sure it was suitable for all purposes, and the friends to whom I showed it said that it was easily understandable. The planning dedicated to the object model helped to make the code more comprehensible, allowing me to walk away from the project for a week and, upon returning, to have little difficulty understanding what I had done previously. I took notes as I was developing and left ample comments in the code, such that I could read the comments only to follow the flow of the program, without having to take apart the code itself.

On the whole, the time spent in planning and documenting the application saved a great deal of frustration in the development phase. In previous semesters, I had only written shorter programs for assignments, and hadn't understood the benefits of design and documentation. In my professional experience, I had at first enjoyed working in an environment where virtually no time was dedicated to design or documentation. However, after two years, I've come to understand that the reason my shop spends 50%+ of its time fixing problems with existing code is largely because serious planning and useful documentation are not part of our process.

Unlike previous academic projects, this and my work with Dr. Shi from last semester spanned several thousand lines of code instead of several hundred. Even on projects of this relatively small size, a great deal can be gained from planning and writing *useful* documentation. While I still do think that some software engineering practices are too burdensome, I would say that my appreciation for the importance of planning and documentation has increased significantly in the past year since 338, both as a result of academic projects and working on a piece of software professionally that is 500,000 lines long but has virtually no documentation.

Finally, although I had initially liked programming in C much more than Java and other OOP languages, I've come to like object-oriented programming more and more. The abstraction that object-oriented programming allows truly can make solving complex problems much easier. I found it much less taxing mentally to consider the problems of routing in terms of nodes, distance objects (which are effectively weighted edges), and vehicles than to think of it in terms

of code and instructions.

b. Implementation

As noted previously, implementation proved much easier as a result of the careful planning and documentation that was generated as I went along. I did run into a few problems regarding some unforeseen consequences of my initial rule set (such as two vehicles going down the same road in opposite directions, which could never get around one another initially). However, I feel as though some of those problems may have been difficult to envision prior to implementation. The object model allowed for me to work around those issues without making changes to the fundamental structure of the system, and I was able to address those issues with relatively few lines of code.

Working within the object model, I was able to stay on schedule throughout the semester, and even to get ahead by the end to add a few extra features that were not part of the initial design. I feel as though the combination of a reasonable plan and a strong design allowed me to reach all my objectives without inordinate frustration.

III. Results

The final results of the simulation were highly satisfying. The first successful objective completed was the implementation of the map editor. The tool provided a useful and simple way to draw the environments in which the simulation could be run. The drawing was intuitive, the data storage was simple, and it proved a highly functional tool.

Having interfaced the simulation program with the output of the map editor, I next implemented the traffic jam simulation. In this, vehicles all travel the shortest path in terms of distance, without accounting for traffic patterns. As shown in the screenshot above (section I.3.b), traffic jams were generated as anticipated. The result is highly pleasing visually, as once can see the motionless vehicles that back up around the key intersections, just as one experiences when driving at rush hour.

The truly satisfying moment in this project, however, was when the optimized solution began to work properly. After a bit of fiddling around with mechanisms to account for traffic, I decided upon a simple technique to increase the weight of edges in the graph based on the number of vehicles between two vertices. When I first saw vehicles travelling on “back roads”, or paths that weren’t numerically the shortest, I knew I was on the verge of success. After experimentally determining constants by which I could multiple numbers of vehicles to augment edge weights for different maps, I felt as though results were consistently and significantly lower for the optimized mechanism than its non-optimized counterpart. Seeing vehicles autonomously select superior routes that were not necessarily shorter in terms of distance was the best part of the entire project.

However, to verify my results more thoroughly, I added a mechanism to run the simulation in both its non-optimized and optimized forms hundreds of times overnight. Results of these runs would then be written to a text file, which could then be parsed for statistical analysis. I only had to run the simulation a few times to get results which were deemed statistically significant. My brother, who majors in mathematics, was able to use a statistical analysis program he has to develop a confidence interval from which the results could be viewed. After some initial confusion regarding for what I was interested in testing, we concluded that a one-way ANOVA test would yield the most meaningful results. The results of this analysis indicated that the two data sets almost certainly came from different populations, and that the mean times in seconds of the optimized runs are significantly lower, on average, than their non-optimized counterparts. The test's results and an accompanying explanation of their meaning can be seen below.

4/16/2009 11:27:55 AM

One-way ANOVA: Mean_Time_Taken(Seconds) versus Run_Type

```
Source     DF      SS      MS      F      P

Run_Type    1  339.44  339.44  87.18  0.000

Error      43  167.42    3.89

Total      44  506.86

S = 1.973   R-Sq = 66.97%   R-Sq(adj) = 66.20%

                           Individual 99% CIs For Mean Based on Pooled StDev

Level   N    Mean  StDev     +---------+---------+---------+---------

0      23  78.626  2.104                                  (----*-----)

1      22  73.132  1.826     (-----*----)

                             +---------+---------+---------+---------

                           72.0      74.0      76.0      78.0

Pooled StDev = 1.973

Tukey 99% Simultaneous Confidence Intervals

All Pairwise Comparisons among Levels of Run_Type

Individual confidence level = 99.00%

Run_Type = 0 subtracted from:


Run_Type   Lower  Center   Upper  --------+---------+---------+---------+-

1         -7.080  -5.494  -3.908  (-----*-----)
```

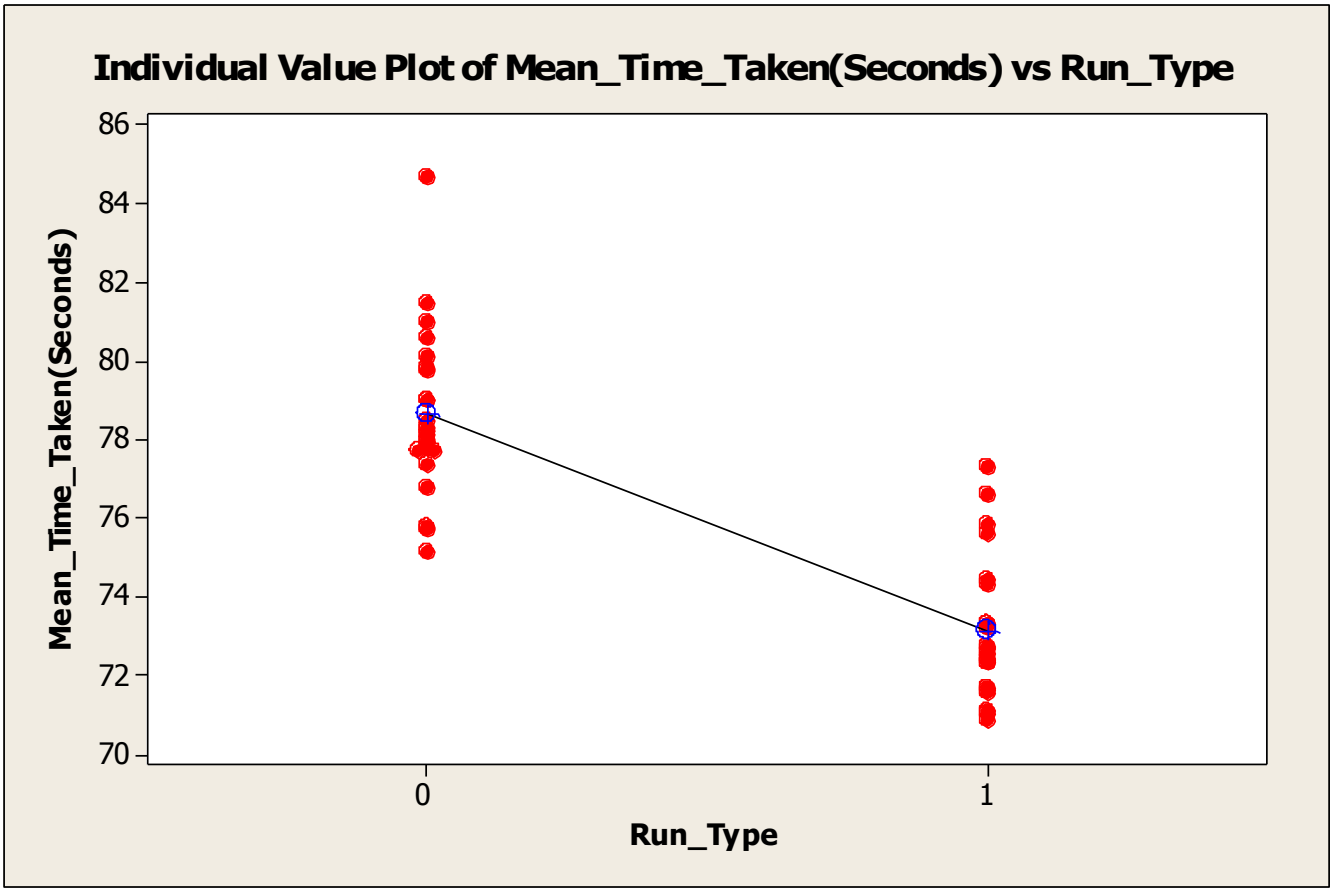
```
--------+---------+---------+---------+-
      -5.0      -2.5       0.0       2.5
```

H_0: Run Type 0 and Run Type 1 come from the same population of data.

H_1: Run Type 0 and Run Type 1 do not come from the same population of data.

Conclusion: Reject H_0.

Meaning: At 99% confidence, there is sufficient data to conclude that there is a significant difference in the mean number of seconds taken between Run Type One and Run Type Zero. Furthermore, at 99% confidence there is sufficient evidence to conclude that the the mean amount of time Run Type One takes to complete its task is significantly lower than that of Run Type Zero. Type Zero is expected to take, on average, 5.494 seconds longer than Type One.



Graphical Display of Mean Time Taken vs. Run type.

IV. Conclusions

On the whole, I am extremely satisfied with the results of this semester's work. I feel as though it was a good opportunity to apply some of the skills I have acquired and to produce a fun and interesting piece of software. This has been one of my most positive experiences in the CS program, and I feel as though I was able to learn a substantial amount, both about myself as a developer and the technical aspects of algorithms that interest me.

Finally, I am pleased with the results, both from a visual and numerical standpoint. Generating the desired results is the most important aspect of any undertaking, and I am glad that I was able to produce what I intended. There were a few times, especially when dealing with some of the unforeseen rule problems, that I wasn't entirely sure whether the project could be completed successfully. However, falling back on the original design and working within the framework I had initially devised allowed me to overcome the challenges and produce software with which I can be happy.

V. Future Work

Although this project is reaching its conclusion, I remain thoroughly interested in practical applications of specific artificial intelligence algorithms. As far as this particular program is concerned, I would like to perform more overnight runs on a wider variety of maps with different features, to get a feel for the relationship between optimized constants for edge weight augmentation and the number and severity of chokepoints in a map. My feeling from having briefly experimented with different maps is that, the more severe the bottlenecks, the higher the constants should be for edge weight augmentation. While I cannot statistically verify that now, I am confident that, if I had more time, I could devise and implement a mechanism for doing so.

Also, while I have enjoyed my time as an insurance systems developer, I am happy to have received an outstanding job offer from DMI, a company that programs artificial intelligence systems under DARPA's supervision. After having interviewed with them last week, I suspect that I will have the opportunity to combine my interests in parallel computing and artificial intelligence at their organization. I have one more interview next week with the air and missile defense group at the Applied Physics Laboratory of Johns Hopkins University, and, were I to get that position and access to their graduate program, I sincerely hope that I could further study these fields of interest.