**Rideshare:**

**A Web Application for Automated**

**Carpool Matching and Optimization**

**Submitted To**

**CheolHee Park**

**Dr. Constantine Caramanis**

**Prepared By**

**Yoni Ben-Meshulam**

**Garrett Cooper**

**Derrick Huhn**

**Patrick Lowry**

**EE464 Senior Design Project**

**Electrical and Computer Engineering Department**

**University of Texas at Austin**

**Spring 2008**

**CONTENTS**

**FIGURES vi**

**EXECUTIVE SUMMARY vii**

**1.0 INTRODUCTION 1**

**2.0 THE RIDESHARE PROBLEM 2**

**3.0 OUR SOLUTION 2**

**3.1 FRONT-END 3**

**3.2 DATABASE 3**

**3.3 CONTROLLER BACK-END 3**

**3.3.1 Controller Library 4**

**3.3.2 Optimizers 4**

**3.3.3 Algorithm Visualizer 4**

**3.3.4 Notifications 4**

**4.0 CHALLENGES AND IMPLEMENTATION 4**

**4.1 FRONT-END CHALLENGES 3**

**4.1.1 XHTML 4**

**4.1.2 Optimizers 4**

**4.2 CONTROLLER CHALLENGES 3**

**4.2.1 Remote Developmen 4**

**4.2.2 Build System 4**

**4.2.3 Third Party Libraries 4**

**4.3 OPTIMIZER CHALLENGES 3**

**4.3.1 Optimizer Framework 4**

**4.3.2 Implementing the Algorithms 4**

**5.0 TEST AND EVALUATION 5**

**5.1 TEST-DRIVEN DEVELOPMENT 3**

**5.2 DESIGN TESTING 3**

**5.3 PERFORMANCE TESTING 3**

**CONTENTS (Continued)**

**6.0 TIME AND COST CONSIDERATIONS 7**

**7.0 SAFETY AND ETHICAL ASPECTS 8**

**8.0 CONCLUSIONS AND RECOMMENDATIONS 8**

**REFERENCES 9**

**APPENDIX A – PSEUDO CODETITLE FOR OPTIMIZERSA-1**

**FIGURES**

3.1 Brute Force Algorithm6

3.2 Genetic Algorithm6

3.3 Bipartite Matching Algorithm6

3.4 Algorithm Visualizer6

4.1 Scoring Function6

4.2 Schedule Criteria6

4.3 Bipartite Matching Algorithm6

5.1 Score Comparison: Bipartite Matching vs. Genetic6

5.2 Time Comparison: Bipartite Matching vs. Genetic6

5.3 Logarithmic Time Comparison: Bipartite Matching vs. Genetic6

5.4 Run Time vs. Score: Bipartite Matching6

5.5 Run Time vs. Score: Genetic6

**EXECUTIVE SUMMARY**

[Text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text.

Text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text.]

*[Start your Executive Summary on a new page immediately after your table of contents (or list of figures or tables). The Executive Summary is part of the front matter of the report, and it should therefore have a lowercase Roman numeral page number. Set the heading in 12-point bold, all caps.*

*Note that the Executive Summary is the only section in the report that is typed in single space.* ***The length of the summary is limited to one page.***

*The purpose of the Executive Summary is to give your readers a quick overview of the contents of the final report. As such, it identifies the major ideas in the general order they appear in the report, so that the readers can go directly to a full discussion of these ideas if they desire.*

*Write the Executive Summary under the assumption that it may be the only part of the report some readers look at. Emphasize the important aspects of the problem, your design approach, and the specifics of your solution.*

*A good idea is to write your Executive Summary after you have completed the rest of your report, but do not sacrifice its quality. All too often, students write the Executive Summary at the last minute and give it insufficient attention.]*

**1.0 INTRODUCTION**

[Text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text.

Text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text]

*[The Introduction prepares the reader for the information that is to follow. You may want to write, or at least revise, your introduction after you have written the rest of the report.*

*The Introduction begins by giving the purpose of the report (what the final report is to accomplish), identifying its topic (your design project), and defining your design product solution (what it is that you designed). The important point is this: Get to the point immediately. You can then provide background information, including identification of the team members and general information about the sponsor. In addition, you should describe how your design project has contributed to engineering knowledge and practice. Finally, the Introduction provides the reader with a brief overview of the organization of the report.*

*The Introduction starts a new page after the Executive Summary. It does not need to be long to be effective. A length from a half page to a single full page will generally suffice.*

*The Introduction begins your first page of the body of the report. Although this page represents page 1 of the numbering system, do not place a page number on this page; only on page 2 and succeeding pages do the page numbers appear in the body text. Note that the first-level heading is centered on the page, set in all capital letters, and designated by a decimal section number (1.0). Skip a line space after the centered heading. The Introduction and all following text take 1-1/2 line spacing.]*

**2.0 DESIGN PROBLEM STATEMENT**

*[With line spacing set at 1-1/2, press Enter twice to skip a line after centered headings.]*

[Text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text.

Text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text]

*[This section gives a complete description of your project, including its goals, motivation,* *and requirements. This description should include an explicit statement of the design problem, problem specifications, the design parameters, and all constraints on problem solutions. Provide a general statement of your design approach to the problem and explain early decisions that influenced your approach. Keep the focus on the design path you followed; that is, do not go into details of the alternative designs you considered.*

*Note that this section is again designated by a first-level heading centered on the page. Skip a line (press the Enter key twice) to separate the heading from the previous text. Do not start a first-level section on a new page unless it is necessary. You may use subheadings to help organize information within the section, as follows.]*

**3.0 DESIGN PROBLEM SOLUTION**

*[With line spacing set at 1-1/2, press Enter twice to skip a line before and after centered headings.]*

[Text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text.]

*[Whereas in the previous section you described the design problem, in this section you describe the design solution. Describe your product (artifact, program, system, etc.) by explaining its underlying theory and its operating principles. Describe the design decisions leading up to your solution and how you implemented them. A brief discussion of alternative designs is appropriate in this section. Emphasize how your solution meets the various problem specifications, parameters, and constraints. Be as explicit in describing areas where your solution fails to meet specs as you are in describing areas where it does.*

*Again, this section may require subsections as previously described. Use figures and tables to support your explanation where necessary.*

*Note****:*** *A technically knowledgeable person should have enough information to understand and judge your solution and its implementation. An economic analysis should be included as part of the project solution. An economic analysis addresses, for instance, cost/benefit analysis.]*

**3.1 [A SECOND-LEVEL HEADING]** *[Do not skip a line after side headings.]*

**3.1 Front-End**

As mentioned previously, we decided to implement the front-end as a web-based PHP solution.  We needed to provide a means of distinguishing users from one another and allow them to set preferences, and create and view routes and assigned matches through a map interface.  The front-end needed to validate all data input by users - authenticated or not - and write this data the database which could later be processed by the optimizers.

We decided to implement an authentication-based structure, common among web-applications, where users authenticate themselves with a password they set upon creating their account.  For security reasons, we decided to never store the users' passwords in plain-text; instead we saved a digest of the password.  This policy addresses the possibility of identity theft if our server ever became compromised since an attacker would not be able to extract our users' passwords.

The map of web-pages can be seen in Figure 3.1. The number of pages is kept to a minimum so users are less likely to get lost while navigating the site.  As can be seen in the design layout screenshot of Figure 3.2, the interface is designed to be minimalistic and intuitive so that the users can focus on what is important.  We decided to go with Google Maps as opposed to Yahoo! Maps or MapQuest for aesthetic reasons and their simple yet effective API.

In addition to using traditional forms of notifications such as email to notify users of matches, we also integrated Really Simple Syndication (RSS) feeds into the user interface [14].  RSS feeds provide new ways of interaction by allowing users to subscribe to Rideshare content from within other websites or other applications such as iGoogle and Outlook, respectively.  This allows Rideshare's users to view their results in real time since the RSS feed would be generated dynamically using PHP.

[Text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text]

*[Use subheadings as necessary to improve accessibility to the contents of your report. A subheading is aligned on the left and is set in all capital letters. The decimal section designator assures that each section and subsection in the report has a unique designation, which is useful for referencing one section from another.* ***Skip a line space before, but not after, a subheading (side heading).****]*

**3.2 [SECOND-LEVEL HEADING]**

[Text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text]

*[If you divide a section into subsections and sub-subsections, you must have at least two subheadings in each level of headings. This preserves the hierarchical organization of your report.]*

**3.2.1 [Third-Level Heading]**

[Text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text]

*[You can form a lower level subheading by adding another decimal to the section designator. To give further differentiation from higher level headings, set this third-level heading in bold and capitalize initial letters only. Again, skip a line before, but not after all second-and third-level headings.]*

*Organize your report to avoid fourth- and fifth-level headings. Too many sub- and sub-subheadings confuse the reader and reduce your ability to highlight the organization of the report.]*

**3.2.2 [Third-Level Heading]**

[Text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text]

*[Again, for any level of subheadings in a section, there must be at least two subheadings at that level.]*

**3.3.2 Optimizers**

Our design for the optimizers involved a modular approach.  We first created a framework containing all the functionality the optimizers need to execute but focused on making the functionality usable by all of the optimizers instead of specific to each.

Using this framework, we implemented three optimizers.  We required optimizers to accept an initial solution, i.e. a set of rideshares and unmatched riders, and return an optimized solution.  We designed the optimizers to accept rideshares that still had a vacancy in the car and potentially matched riders to the rideshare in later optimization sessions.  The optimized solution returned by the optimizer represents the solution with the highest score it had found while optimizing, i.e. best possible match of riders to drivers.

We now detail the functionality of each optimizer.  Pseudo-code for each of the optimizers, as well as the scoring function, can be found in Appendix #.

The first optimizer used the brute force approach.  This naive approach involves checking every possible solution in the search space and returning the highest scoring solution.  The time complexity for this optimizer is too large for it to return an optimized solution in a tangible amount of time so we did not use it in practice.  Figure 3.1 illustrates the brute force approach.

**Figure 3.1  Brute Force Algorithm**

The second optimizer used the genetic approach.  This approach uses the theory of natural selection.  It begins with an initial solution and spawns a population of other solutions based on mutations of the original.  Taking the best solutions from the population, we again introduce mutations and generate a new population.  This process is repeated until a predetermined amount of time has elapsed, or the solution scores have converged to an optimal score.  Figure # [FIXME] illustrates the functionality of the genetic algorithm.

**Figure 3.2  Genetic Algorithm**

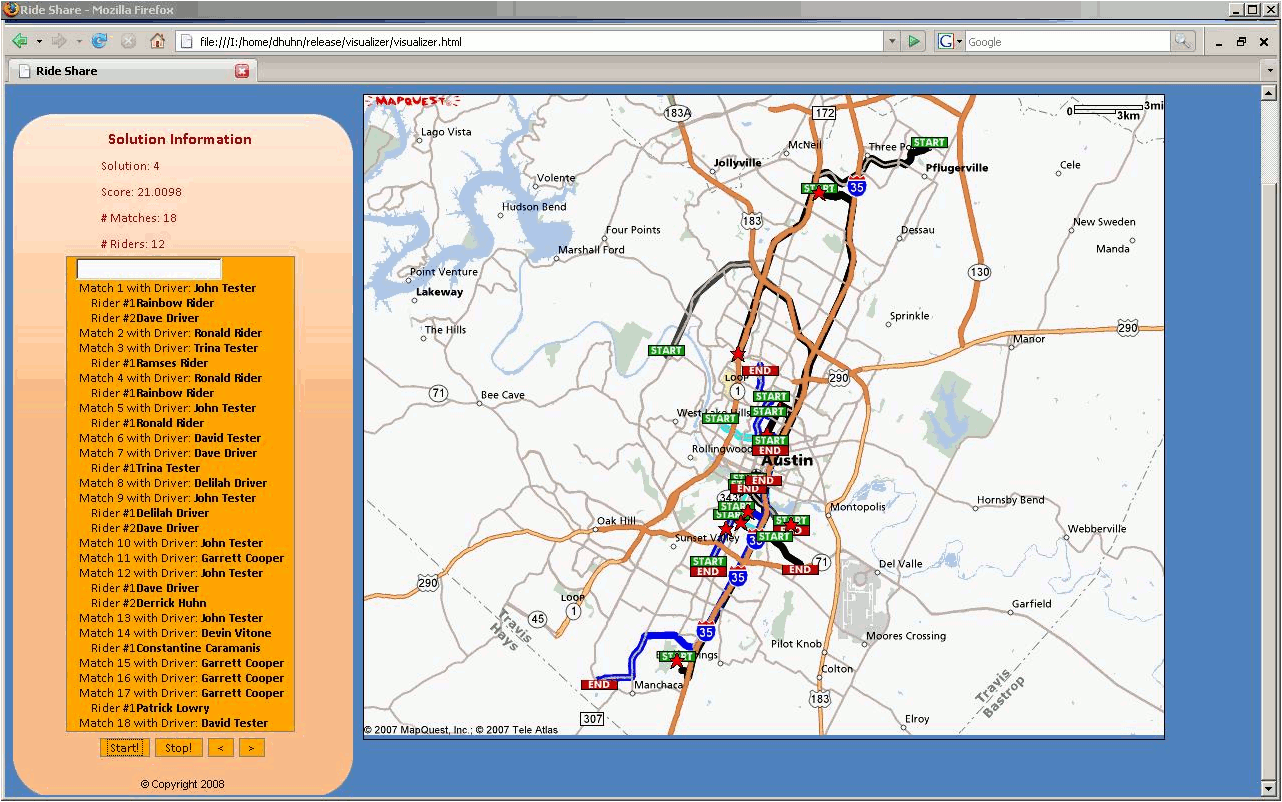
The third optimizer used an approach for maximal matching on bipartite graphs.  This approach views the problem as that of matching between two disjoint sets of points: rideshares and unmatched riders.  These two sets have a relationship of compatibility between them, namely, a rider's compatibility with the rideshare.  The algorithm for maximally matching the sets utilizes some properties of paths between these sets.  Figure # illustrates how it works.

**Figure 3.3  Bipartite Graph Algorithm**

**3.3.3 Algorithm Visualizer**

Although we were able to verify the correctness of optimizers through unit tests and simulated test cases, the Rideshare team also developed a tool for visually analyzing the results produced by the different optimizers.  Even though the algorithm visualizer was initially proposed for debugging purposes, it later became an aesthetic feature that exhibited the core functionality of the back-end of the application.

The algorithm visualizer worked in concert with the optimizers as it took 'snapshots' of the intermediate solutions generated during the different stages of optimization.  As each optimizer performed calculations, a 'camera' embedded in fixed stages of the algorithms would capture the current state of the solution and save it for later use. After an optimizer completed working on a set of rider and driver submissions, all of the 'snapshots' would then be viewable through the algorithm visualizer after images were created using the MapQuest Image API.  Designing and the algorithm visualizer required us to evaluate different options for implementation.  The language, framework, and tools to use were not obvious from the requirements of the visualizer so we initially considered Java and Ruby as the language for implementation and Swing and GTK as the GUI tool kits.  Eventually, the team decided to use XHTML and JavaScript since Patrick had already established a framework and style that could be easily adapted for the visualizer.

Generating the match information in a usable format for the visualizer was accomplished through meta-programming by generating JavaScript data using a C++ program.  This C++ program translated the information in the solution 'snapshots' into JavaScript and also generated images of the solutions by invoking the image creation features of the MapQuest library.  Once the JavaScript and images were generated, the XHTML page would then execute the JavaScript and an animation of the different solutions would be displayed.  A screen shot of the algorithm visualizer can be seen in Figure 3.4.

**Figure 3.4  Algorithm Visualizer**

**4.0 DESIGN IMPLEMENTATION**

*[With line spacing set at 1-1/2, press Enter twice to skip a line after centered headings.]*

[Text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text]

[Text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text]

[*This section describes the challenges you met and overcame while building your prototype. Explain what modifications you had to make to the original design, and describe any innovations in technique, materials, or design practice that you performed during the prototype construction phase.*

*Sometimes student teams submit reports that contain an excellent analysis of the situation but inadequately explain all of the implications of the chosen solution. Sponsors tend to look for* results *and are appreciative of a well-documented solution. This section describes detailed design work beyond concept generation (that is, the embodiment of the design in a working prototype), and it should refer to appendices that contain parts lists, drawings, prototype construction information, and anything else that might be useful. Summarize the analysis and modeling that you used to predict the performance of your solution. Detailed derivations and calculations should be presented in the appendices. Use good judgment in placing important information (major drawings, analysis, or test data) in the text and less important supporting documentation (parts lists, detail drawings, or raw data tables) in the appendices.]*

4.3 Optimizer Challenges

Framework

In order to implement the optimizers, we first needed a common framework that the optimizers would build upon.  In this section we discuss two of the more important methods.

We needed a general scoring function for all optimizers, allowing us to evaluate and compare the different optimizers.  Each optimizer returns a solution, which includes a set of rideshares (a rideshare is taken to mean a driver with zero or more riders) and a set of unmatched riders.  We used two criteria for scoring a given solution: the primary criteria is the number of users matched and the secondary criteria is the inverse of the total route distance.  The total route distance is the sum of the route lengths over all drivers, taking into account deviations due to carrying riders. This scoring function has interesting properties such as a tendency to score higher solutions that spread out riders across drivers, as drivers without riders are not counted as matched and total route length is effectively a tie-breaker of solutions when the same number of people matched.

Figure 1 [FIXME] details the equation the scoring function uses.  The integer part of the score corresponds to the primary criteria, and the fraction corresponds to the secondary criteria for scoring.



**Figure 4.1  Scoring Function**

Optimizers would also need to be able to check for compatibility between a rideshare and an unmatched rider.  We took a long time developing the compatibility check and were forced to redesign several aspects of it throughout the semester.  In order to assert that a rider is compatible with a rideshare, we check that all of the following criteria are met:

1. The rider's age, gender, and rating comply with the age, gender, and rating preferences of all members of a rideshare.
2. All members of a rideshare's ages, genders, and ratings comply with the rider's age and gender preferences.
3. The maximum capacity of the vehicle is not exceeded.
4. The rider's schedule is within the time windows of the driver's schedule.
5. The rider's beginning and ending locations are within a reasonable distance from the driver's beginning and ending locations, or along the way.
6. The route length of the resulting rideshare would not cause the driver to deviate more than a reasonable distance out of his or her way.
7. The schedules of all members of a rideshare, as well as the rider's schedule would all be met if the rider is added to the rideshare.

The first three criteria are simple comparisons and require nothing more than a simple series of "if...then" statements to implement.  The fourth and fifth criteria are essentially common sense criteria, i.e. heuristics.  The role of these criteria is to minimize the use of the sixth and seventh criteria which have much larger time costs than the rest of the criteria combined.  The reason for this is that they both require us to first find the optimal order for going from the driver's origin, through all of the rideshare members' locations, and ending at the driver's destination.&nbsp; This route should also include the prospective rider.  
  
In order to compute the optimal route, a simple brute force method is used, which requires that we check all possibilities.   The number of possibilities can be shown to be the factorial of the vehicle's capacity.  Since the capacity of a vehicle for riders generally does not exceed four, the time required can be considered on the order of O(4!)=O(24)=O(1).  In big-O terms, that is sufficient to consider it constant time for purposes of the optimizers.  
  
Once the optimal route is computed, the order of points and time and distance required to reach each point is known.  This allows criteria number six to be checked by comparing the total route length with the driver's initial route length.  If it is within a factor of 1.3 of the driver's initial route, then that criteria is met.   
  
Finally, making sure that everyone's schedule is met involves two stages.  The first stage involves finding the narrowest time window for the driver to leave.  The second stage is adding to this time window the time required to get to each of the members' locations in the order of the optimal route.  At each location, we check that the time window is within the corresponding user's schedule.

Figure 4.2 illustrates the algorithm for asserting the schedule criteria is met. We narrow the driver’s leave window until we reach a worst case leave window for the driver. We then add to this leave window the time necessary to get from point to point, each time verifying that the users’ time windows do not conflict with the arrival window for that point.

**Figure 4.2  Schedule Criteria**

Optimizers

Brute force and genetic optimizers proved to be very simple to implement, once we had the framework ready.  The framework methods included most of the logic used by both optimizers, making implementation a straight-forward task.  
  
The bipartite matching optimizer was a different story.  We needed to use several graph algorithms, and planned on using one of several C++ graph libraries, including Boost, and LEDA [16][17].  When we tried using each of these libraries, we met with lots of problems.  Eventually, we abandoned our original plan, and decided to augment our existing data structures to provide an interface for the bipartite matching optimizer.  Using this interface, we implemented the necessary graph algorithms, which could then be used for bipartite matching.  This alone probably set us back about two weeks on all of the optimizers.

**5.0 TEST AND EVALUATION**

[Text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text.]

*[This section describes the test and evaluation plan that you devised and conducted. Record your results and observations with great care for accuracy. Summarize your results, and show your data. Compare your results against the design specifications. Where do you fall short and why?]*

[Text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text as shown in Figure 1. Text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text.]

**Figure 1. [Caption of Figure]**

[text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text.]

*[Note that a figure is placed as soon as possible after its first reference in the text, usually at the end of the paragraph in which it is mentioned. If the figure is too large to fit on the same page, then continue the text to the bottom of the page and place the figure at the top of the next page (and tell the reader it is on the next page).*

*Notice that the figure caption is typed in bold and initial caps. Center the caption below the figure and skip a line space between the top of the figure and the preceding text and between the caption and following text.]*

[Text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text **as shown in Table 1.** Text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text.]

**Table 1. [Title of Table]**

|  |  |  |
| --- | --- | --- |
| **Column 1** | **Column 2** | **Column 3** |
| **Category 1** | **345** | **58.99** |
| **Category 2** | **156** | **56.44** |
| **Category 3** | **567** | **24.33** |

[Text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text.]

*[A table, like a figure, is placed as soon as possible after its first reference in the text, usually at the end of the paragraph in which it is mentioned. If the table is too large to fit on the same page, then continue the text to the bottom of the page and place the table at the top of the next page (and tell the reader it is on the next page).*

*Notice that the table title is typed in bold and initial caps. Center the title above the figure and skip a line space between the preceding text and the title and between the bottom of the table and the following text.]*

Rideshare was primarily developed using the Test Driven Development paradigm.  For new every feature we implemented, we also wrote a corresponding unit test to verify the feature worked correctly.  Subsequently, we executed all other unit tests to ensure that the new feature didn't break existing code.  We believe this design pattern was very useful since we were able to quickly detect bugs and design problems.  Each team member also owned specific modules of code so when a unit test failed, the owner was notified and the issue was addressed.  We had over one hundred unique assertions that ranged from testing the functionality of copy constructors to verifying the behavior of the MapQuest libraries.   
  
For design testing, we created several unique corner cases designed to test the optimizers and verify correct behavior.  Each of these cases tested a different function: one simple case put three identical riders in the path of the same driver who only had the capacity to pick up two of them.  This simple cast tested the logic in the canAddRider function pertaining to the capacity of drivers and was used to verify that a driver could not be assigned more riders than it had spaces available.  Other tests verified the distance logic such as whether the driver would pick up the closer of two riders, or how far the driver would go out of his way to pick up a rider.  As we tested each of these cases, we were able to appreciate having multiple optimizers since a few of the bugs we fixed in one optimizer were found quickly as a direct result of comparing results from another optimizer using the same inputs.  For the bipartite matching optimizer, we even had a test case specifically designed to verify the algorithm was finding augmenting paths correctly.  
  
In order to accurately simulate real Rideshare users, we acquired actual user data from a shipping website called uShip [18].  Though the data they gave us was all from Texas, it was all across the state.  Initially, we put all of the routes in the database under Garrett's account, but we felt this did not adequately simulate real users.  We wrote a program augment this data, listing the routes under different test users, each with individual preferences.  We randomized whether the route would be a driver or a rider and what capacity the driver would have.  Each route also had a stochastic probability of including specific preferences.  Once we established the functionality of the optimizers using the corner cases, we used sixteen test cases from the enhanced uShip data to test the time and score performance of the algorithms.  The data has several properties which are important to consider when analyzing the resulting data.

The first property observed in the results was that the data sets increase with the number of users. This property ensures that we can measure horizontally over the test cases with a standard measure of the number of users.  In addition, all of the test cases contain the same distribution of drivers and riders, which is roughly one driver for every two riders.  Finally, it is important to note that each test case is independent of other test cases i.e. the users in one test do not necessarily include the users from another test.

With these facts in mind, we now take a look at the test results.  We will not discuss the performance of the brute force optimizer in this section, because already at the third test case, it fails to complete execution within a reasonable time frame.  If we were to graph the results with the other optimizers, brute force's would be off the charts.

In Figure 5.1, we compare the score results for the bipartite matching and genetic optimizers.  The horizontal axis shows each of the test cases 1-16, and the vertical axis shows the score.  For the genetic optimizer, we observed that the score increased linearly.  On the other hand, the score of the bipartite optimizer has a more irregular pattern which changes for each test case.  Last, we noticed that the ratio of the bipartite matching score to the genetic score increased, i.e. the bipartite algorithm might be “catching up” with the genetic algorithm with respect to score. &nbsp;

From these results, we concluded that the algorithms perform differently on the same data, and the advantage of using the genetic optimizer for larger test cases is clear.  Furthermore, it would be of interest to use larger test cases and determine if these patterns continue or change with the size of the test case.

**Figure 5.1 Score Comparison: Bipartite Matching vs. Genetic**

Figures 2 and 3 show the number of users verses time for the bipartite matching and genetic optimizers.  Figure 2 shows the time in minutes, whereas Figure 3 shows time on a logarithmic scale.

In Figure 2, we see that the time complexities are somewhat similar for both optimizers. Each of them has several rises and falls in the time.  Some of the points seem to show dips in both graphs but not all of the points are correlated.  In addition, it looks like the run time increases rapidly over the first test cases, but slows down and even reverses over the larger test cases.  We would need to test further with more data points and larger test cases in order to find a curve for the time complexity, but we can already tell it’s not growing too rapidly.

**Figure 5.2  Time Comparison: Bipartite Matching vs. Genetic**

In Figure 3, we take a look at the same data from Figure 2, though we evaluate it on a logarithmic scale.  The point of this evaluation is to compare the time complexity of the optimizers to that of an exponential curve.  In a logarithmic graph, a linearly increasing graph represents exponential growth.

Sure enough, the first test cases cause what appears to be exponential growth in the amount of time taken, but the larger test cases seem to break this pattern, and follow a flat, fairly steady curve.  This is good news, and confirms that our optimizers do run in polynomial time.  Again, we would like to add larger test cases to confirm this pattern.

**Figure 5.3  Logarithmic Time Comparison: Bipartite Matching vs. Genetic**

In Figures 4 and 5, we compare the final score with the run time for each algorithm. The motivation for this graph comes from the fact that the previous graphs suggested that the time is correlated to something other than the number of users.  If each point had the absolute optimal solution, we would expect to see a linear curve for the score.

Bipartite matching seems to have dips in both scales around the same test cases.  This implies that the time to complete bipartite matching is strongly correlated to the final score it produces.  This result might be due to the fact that bipartite matching runs time consuming compatibility checks more frequently when it has compatible users.

When we take a look at the genetic optimizer, Figure 5, we see what looks like a near perfect linear increase in the score.  Therefore, it seems, the genetic optimizer’s time complexity is not correlated with the score it produces.  Further, the genetic optimizer seems to be producing a curve which reflects our expectation of a linear increase of the score with the number of users.

**Figure 5.4  Run Time vs. Score: Bipartite Matching**

**Figure 5.5  Run Time vs. Score: Genetic**

The results from Figures 1-5 tell us a lot about the performance and functionality of our optimizers.  We were especially surprised with the irregular increase in the time required by both algorithms, and the linear increase in the score for the genetic optimizer.  In addition, we established that our optimizers operate in polynomial time.

The end result, though, is more questions than answers.  Can we expect these trends to continue? Will the behavior of the optimizers behave differently with test cases in the thousands of users?  If we had time, we could generate more results to test our conclusions.

**6.0 TIME AND COST CONSIDERATIONS**

[Text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text.]

*[State whether the project met time and budget constraints. Describe any special problems you encountered that led to schedule or cost overruns and your response to those problems.]*

**7.0 SAFETY AND ETHICAL ASPECTS OF DESIGN**

[Text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text.]

*[Assess the safety aspects of your project and how your project will address the larger ethical issues of professional engineering practice. Such issues may include danger to human life and limb, possibility of environmental damage, life-cycle recycling issues, and warning notices required for the user. What safety factors were designed into the project? How will your project serve humanity and the public interest?]*

**8.0 CONCLUSIONS AND RECOMMENDATIONS**

[Text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text.]

*[Your conclusion section is not only the culmination of your paper but also of your entire project. The conclusion recapitulates your achievements, findings, results, and any limitations or shortcomings in your compliance with specifications. The section presents a frank evaluation of how well the final solution meets the project requirements and parameters, as well as any ways in which it exceeds the project requirements.*

*In this section, list any recommendations for further studies or investigations for improving the design.*

*This section (along with the Executive Summary) is the most frequent source of grade reduction. Demonstrate your knowledge. At this point, you should be the expert on your solution. Do not be afraid to make recommendations not directly related to your specific solution. Give your view of the “big picture.”]*

**REFERENCES**

[1] [First source reference in IEEE format. First source reference in IEEE format. First source reference in IEEE format. First source reference in IEEE format.]

[2] [Second source reference in IEEE format. Second source reference in IEEE format. Second source reference in IEEE format. Second source reference in IEEE format.]

*[Etc.]*

*[Note that the brackets around each number (unlike the other brackets in this mockup) are required in a list of references. Each listed reference should be single spaced, with 1-1/2 space between references. This section is a listing of all references of source material. (See Item 14 of the General Specifications for Final Reports.) List the references in the same numbered order that they are cited in the text.* ***Note that this section does not have a decimal number; yet, it is a first-level heading.****]*

*[This page is the title sheet for the first appendix. All appendices must have a title sheet, with the title centered horizontally approximately a third of the way down the page. Be sure to give each appendix a descriptive title. No appendix can appear that has not been referenced in the text, and all appendices must be listed in the table of contents. This page receives the page number A-1, and the numbering system continues with A-2, A-3, etc. on the following pages.]*

**APPENDIX A – [TITLE OF FIRST APPENDIX]**

**APPENDIX A – [TITLE OF FIRST APPENDIX]**

Text, figures, tables, or chart. Text, figures, tables, or chart. Text, figures, tables, or chart. Text, figures, tables, or chart. Text, figures, tables, or chart. Text, figures, tables, or chart. Text, figures, tables, or chart. Text, figures, tables, or chart. Text, figures, tables, or chart. Text, figures, tables, or chart. Text, figures, tables, or chart. Text, figures, tables, or chart. Text, figures, tables, or chart. Text, figures, tables, or chart. Text, figures, tables, or chart. Text, figures, tables, or chart. Text, figures, tables, or chart. Text, figures, tables, or chart. Text, figures, tables, or chart. Text, figures, tables, or chart. Text, figures, tables, or chart. Text, figures, tables, or chart. Text, figures, tables, or chart. Text, figures, tables, or chart. Text, figures, tables, or chart. Text, figures, tables, or chart.v Text, figures, tables, or chart. Text, figures, tables, or chart. Text, figures, tables, or chart.**]**

*[Repeat the appendix letter and title on the first page after the title sheet. Place an en dash between the appendix letter and title. There are no other requirements for the appendix. Use the space in this appendix for figures, tables, coding technical details, photographs, sketches, or any other type of data that is too extensive or distracting to place in the main text, even though it is relevant and necessary information.*

*[Same instructions as for Appendix A. This page is the title sheet for the second appendix. This page receives the page number B-1, and the numbering system continues with B-2, B-3, etc. on following pages.]*

**APPENDIX B – [TITLE OF SECOND APPENDIX]**

**APPENDIX B – [TITLE OF SECOND APPENDIX]**

[Text, figures, tables, or chart. Text, figures, tables, or chart. Text, figures, tables, or chart. Text, figures, tables, or chart. Text, figures, tables, or chart. Text, figures, tables, or chart. Text, figures, tables, or chart. Text, figures, tables, or chart. Text, figures, tables, or chart. Text, figures, tables, or chart. Text, figures, tables, or chart. Text, figures, tables, or chart. Text, figures, tables, or chart. Text, figures, tables, or chart. Text, figures, tables, or chart. Text, figures, tables, or chart. Text, figures, tables, or chart. Text, figures, tables, or chart. Text, figures, tables, or chart. Text, figures, tables, or chart. Text, figures, tables, or chart. Text, figures, tables, or chart. Text, figures, tables, or chart. Text, figures, tables, or chart. Text, figures, tables, or chart. Text, figures, tables, or chart.v Text, figures, tables, or chart. Text, figures, tables, or chart. Text, figures, tables, or chart.**]**

*[Same instructions as those for the first appendix.]*