ESE 499 Capstone Design Project Final Report

Vehicle Matching for Student Volunteers: A Transportation Solution for the Gephardt Institute

Fall 2017

Submitted to Professors Trobaugh and Schaettler and the Department of Electrical and Systems Engineering

December 10, 2017

Olivia Beres (oliviaberes@wustl.edu), Candidate for BS in Systems Engineering
Savannah Rush (savannah.rush@wustl.edu), Candidate for BS in Systems Engineering
Kara Todd (karagtodd@wustl.edu), Candidate for BS in Systems Engineering

I. Abstract

Students at Washington University in St. Louis lack reliable and cost effective transportation to local volunteer organizations. Previous attempts to address this problem have been prohibitively expensive. Data about student volunteers' availability, common volunteer destinations, and students with cars was obtained from the Gephardt Institute for Civic and Community Engagement and student sign-ups. Our main design objectives were 1) to use this data to create a model of the number of student volunteers and students with cars that included their availability and desired locations and 2) to create a student rideshare program that matched student volunteers and students with cars. To achieve these objectives, we used integer programming with binary components to match volunteers and drivers. Constraints including volunteer and driver availability, car size, and location requests are considered. We implemented our optimization equations in both Excel and Matlab. Our matching solutions successfully provide rides for students at a lower cost than previous solutions.

Table of Contents

II. Introduction

Washington University in St. Louis students are engaged in civic action; in fact, "67 percent of undergraduate students participate in community service" (WUSTL Admissions). With 90% of WUSTL's 6,800 undergraduate students hailing from out of state and 65% coming from farther than 500 miles away, many students do not have cars in their new St. Louis home (WUSTL). Students struggle to find transportation from the Danforth Campus to local organizations throughout the community. Affordable and efficient transportation is imperative to their ability to continue their civic engagement.

The Gephardt Institute for Civic and Community Engagement at Washington University in St. Louis exists to cultivate "informed and actively engaged citizens" (Gephardt Institute). The Gephardt Institute works to remove barriers that prevent students from getting involved. Seeking to solve the problem of student transportation, they implemented a car rental program with Enterprise CarShare. The program allows undergraduate students to rent vehicles at no cost to themselves to drive to volunteer organizations. The car rental program costs the Gephardt Institute $5 an hour. If a student volunteered for 4 hours once a week at a location 30 minutes away, the student would drive to the location, the car would then sit idly for four hours, and then the student would drive home; though the car was only being driven for 1 hour, Gephardt paid $25 for all 5 hours. The CarShare program successfully provided the student with convenient and affordable transportation, but was financially taxing on the institution. *The lack of transportation for student volunteers that is convenient and financially viable for all parties is a problem at Washington University in St. Louis. The purpose of this project was to provide transportation for students at a lower cost than the current Gephardt program.*

In order to address the problem of volunteer transportation, we sought to create a matching system that pairs students with cars and students who want to volunteer. This would provide both transportation for student volunteers and an easy way to make an impact for students with cars. *Our design objective was to develop possible solutions that provide rides to student volunteers and cost less than the Enterprise CarShare program.* The process we used consisted of collecting data, forming a statistical model of volunteer behavior, creating equations to optimize matching, implementing our formulation in Excel and Matlab, assuring project feasibility, analyzing program results, and calculating costs. Our full process is outlined in the following sections.

Additionally, we conducted a literature review of scheduling programs and ride share programs. Ride sharing would not be unique to Washington University in St. Louis. Universities including The University of Utah, The University of Puget Sound, and Oregon State University have ride sharing programs. Using them as an example, we knew that ride sharing is legally viable. However, we had to determine if and how a ride sharing program would be feasible on a mainly residential, medium sized campus. Student sign-ups and a successful Matlab program indicate that our solution is indeed feasible on Washington University in St. Louis' campus.

Our solutions allow students to continue to do their important volunteer work throughout the St. Louis region. Our solutions are beneficial for student drivers, student volunteers, and local organizations and reduce costs for the Gephardt Institute.

III. Data

We acquired data on the previous use of the CarShare program by student volunteers from the Gephardt Institute. This data includes the date, time, trip distance, and cost for every CarShare reservation placed through Gephardt in the last fiscal year. Reservations by day are shown in Figure 1 and by month in Figure 2. Based on this data, we estimated the number of passengers seeking rides for any given day.

The constraints of our integer programs also required us to know the approximate number of students with cars on campus who would be interested in participating in a ride sharing program. To acquire this information, we asked for volunteers who would be willing to participate in a program where students with cars offer rides to students who wish to volunteer in the community. Via a form we sent to 140 students, 43 students offered to participate in the program by driving students to and from volunteering destinations.

We also acquired data on the common volunteering locations of WashU students in order to create a list of locations and their distances from campus. Working with the Gephardt Institute, we were able to conduct a survey of their community partners and student volunteers. After a brief analysis of the results, we identified the 16 most popular volunteering locations, which we use in our volunteer model.

All of this data and the analyses performed greatly influenced our volunteer model, which is described in detail in a subsequent section. In particular, we used data including passenger request times, number of student requests, and reservation length to formulate our volunteer model. Reservation length is shown in Table 1 and Figure 3.

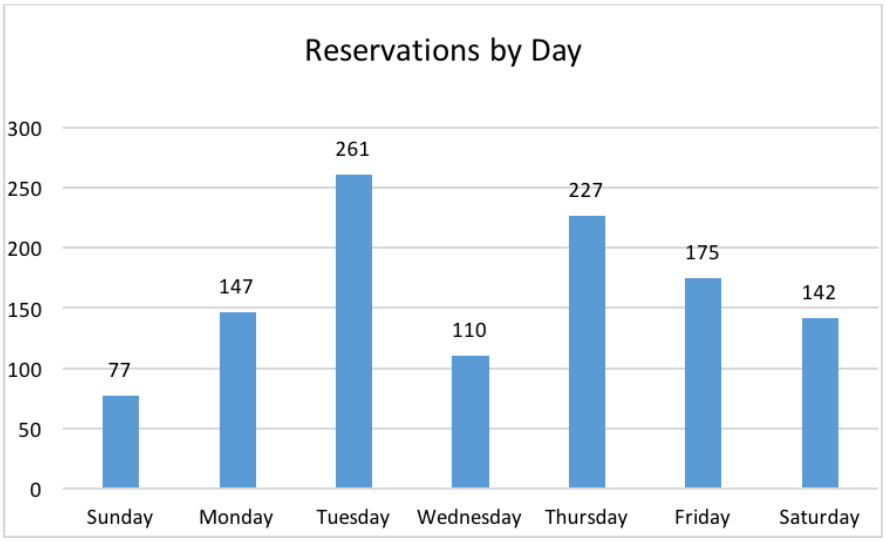

Figure 1: Distribution of CarShare reservations when grouped by day of the week

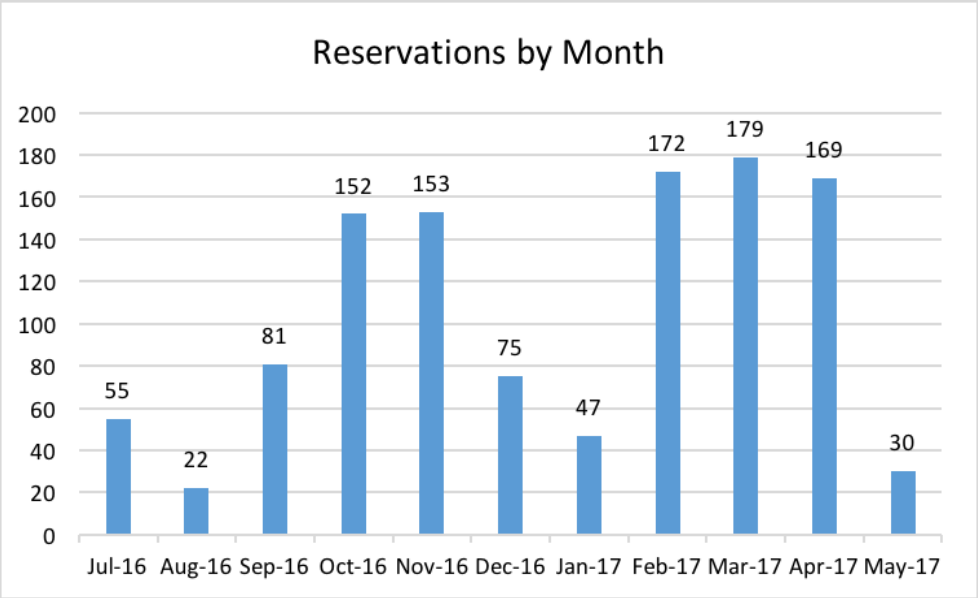

Figure 2: Distribution of CarShare reservations when grouped by month

Reservation Length	
Mean	3.43836111
Median	3
Mode	3
Standard Deviation	2.01795671
Range	19
Minimum	1.00
Maximum	20.00
Count	1149

Table 1: Descriptive statistics on CarShare reservation length

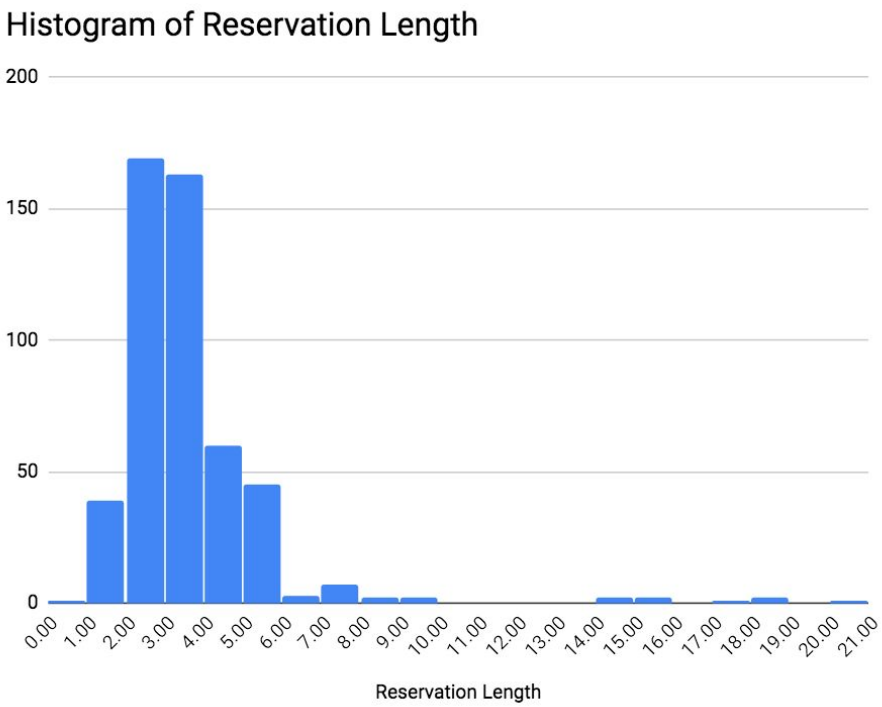

Figure 3: Histogram showing that most CarShare reservation lengths are between 2 to 4 hours

IV. Volunteer Model

In order to create an effective solution, we first wanted to model the volunteering situation on campus. Using a combination of data from the Gephardt institute, student surveys, student sign ups, and our knowledge as Washington University in St. Louis students, we created models for volunteering behavior.

Our primary goal was to model the behavior of students who used the CarShare fund through the Gephardt Institute. Noting both the volunteering rates and the locations students were traveling to, we decided we actually needed two submodels to accurately model the behavior of CarShare fund users. Different non-profits are open on Sundays and fewer students volunteer on that day, as compared to Monday through Saturday. Excited by the potential of our solution, we also wanted to create a larger model that would capture the behavior of more students volunteering than those currently using the Gephardt CarShare fund. Building on our CarShare models, we created an expanded volunteer model.

For all three of our models we needed to determine the number of passengers (P), the number of drivers (D), the locations (T), the passenger and driver availabilities, and the volunteering duration. The following charts compare the models and describe our reasoning for selecting the values.

Number of Passengers (P)

Model	Value	Reasoning
Monday-Saturday	20	The average amount of requests for days Mon-Sat was 177. These numbers are for 11 months (44 weeks). Therefore an average day Mon-Sat had 4 car requests. Assume each CarShare was full (5 passengers). Average car requests * 5 passengers per request = 20 passengers
Sunday	9	The amount of requests for Sundays was 77. These numbers are for 11 months (44 weeks). Therefore an average Sunday had 1.75 car requests. Assume each CarShare was full (5 passengers). Average car requests * 5 passengers per request = 8.75 passengers
Expanded	132	Knowing that 67% of WUSTL students volunteer, there are 4,748 students that volunteer. While this many students volunteer, we will only consider weekly volunteers in our model. Based on the number of students involved in weekly volunteering groups, we estimate that 790 students volunteer weekly. Using the CarShare students as a representative group, most students volunteer Monday-Saturday with each day having a similar number of volunteers. Therefore, the number of students in our model will be 790/6 which is 132 students.

Table 2: Reasoning for Number of Passengers in Models

Number of Drivers (D)

Model	Value	Reasoning
Monday-Saturday	10	Our volunteer sign up resulted in more than 40 volunteer drivers. Knowing we need fewer drivers than signed-up and there are 16 weeks in a semester, we will have 10 drivers, so each driver may volunteer 2-3 times.
Sunday	4	Knowing student availability is less on Sunday, we believe we could get 10% of the drivers who signed up to help each Sunday.
Expanded	40	Used a conservative estimate based on the 43 students who signed up.

Table 3: Reasoning for Number of Drivers in Models

Locations (T)

Model	Value	Reasoning
Monday-Saturday	4	According to the Gephardt CarShare fund data, an average day Mon-Sat had 4 car requests. Assume each car went to one unique location and passengers with the same availabilities had the same location requests because in the Gephardt model they carpooled. Our model therefore has 4 locations.
Sunday	2	An average day Sunday had 1.75 car requests. Our model will assume 2 locations. Assumptions from our Monday-Saturday model hold.
Expanded	7	WUSTL student group data indicates that students volunteer with 16 different community organizations. Ignoring the two locations closest to campus and easily access by Metro, our expanded model will consider 14 locations. Knowing that most locations are visited multiple times a week, but not every day, each day will have 7 locations. Location requests will be randomly assigned using a random number generator.

Table 4: Reasoning for Number of Locations in Models

Passenger Availability

Model	**Reasoning**
Monday-Saturday	In the Gephardt data, 18% of students wanted to leave at 8am and 12% of students wanted to leave at 2pm. The rest of the times were fairly evenly distributed from 9am-5pm (inclusive, with the exception of 2pm). Our model assumes 18% of passengers are free at 8am, 12% are free at 2pm, and 8.75% are free the other hours.
Sunday	For Sundays, it is important to note that in the Gephardt data there are a few requested times for trips that were overnight. Ignoring those overnight trips, we see all normal trips are between 11am-6pm (inclusive). 18% of students wanted to leave at 12pm, 15% wanted to leave at 2pm, and 27% of students wanted to leave at 4pm. Because the average number of car requests was less than 2, we know the students on any given Sunday only left at 2 times. Therefore we will assume 5 students left at 4 pm and 4 students left at 12pm.
Expanded	We used the Monday-Saturday model on volunteer availability (that was grounded in Gephardt data) as representative for an expanded model of volunteer availability.

Table 5: Reasoning for Passenger Availability in Models

Driver Availability

Model	**Reasoning**
Monday-Saturday	We used the Monday-Saturday model on volunteer availability (that was grounded in Gephardt data) as representative for drivers' availability.
Sunday	We will use the CarShare passengers' availability on Sundays as representative for average student availability. Therefore we will have drivers available at 12, 1, 3, and 4pm.
Expanded	We used the Monday-Saturday model on volunteer availability (that was grounded in Gephardt data) as representative for an expanded model of drivers' availability.

Table 6: Reasoning for Driver Availability in Models

Volunteering Duration

Model	Value	Reasoning
Monday-Saturday	3	As shown in Table 1, the average length of trip in the Gephardt data rounded to whole hours is 3. The majority of lengths were in an hour of this. Multi-day service trips and full day service excursions, though found in our CarShare data, will not be in our model. We assume all durations to be 3 hours.
Sunday	3	Used Monday-Saturday Model (grounded in Gephard data) as representative.
Expanded	3	Used Monday-Saturday Model (grounded in Gephard data) as representative.

Table 7: Reasoning for Volunteering Duration in Models

Zoning

As an alternate problem formulation we will consider zoning the locations because some of the 16 most popular locations are near one another. Figure # shows these popular locations and zones. The two locations that are left out of the model due to their proximity to campus and metro stations are indicated by purple Ms on the map. The three zones are represented by green, pink, and gray perimeters.

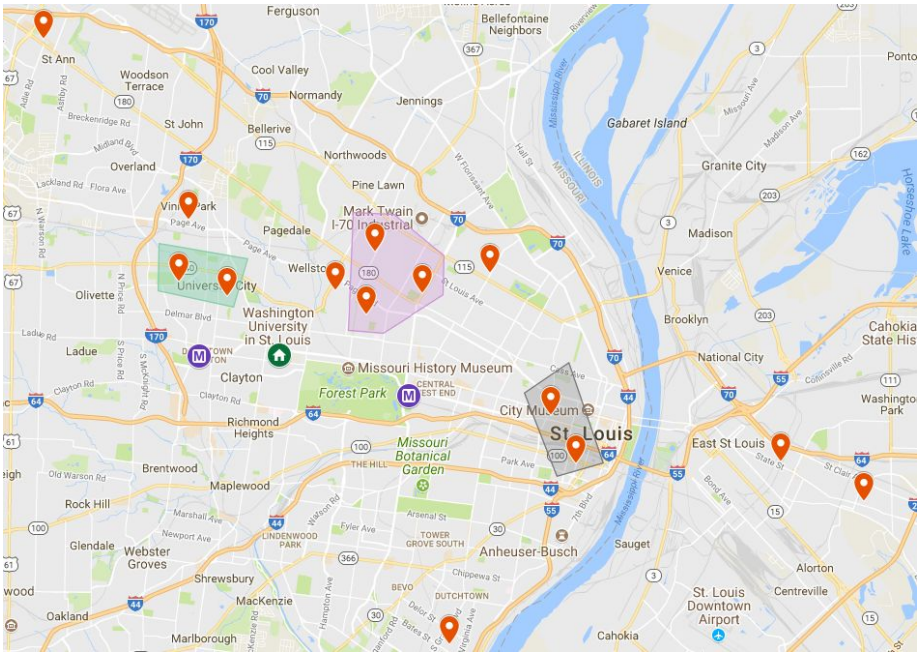

Figure 4: Map Showing Popular Volunteer Locations and Zones

V. Integer Program Formulation

General Integer Program

We accomplished our design objectives by creating an integer program which matchedes student volunteers to a driver for a given day. We had basic knowledge about integer programming from Washington University in St. Louis's Operations Research course. We supplemented this knowledge with more specific information from Fu-Shiung Hsieh of Chaoyang University of Technology (Hsieh). Our solutions are designed to be implemented at a small to mid-size university, like Washington University in St. Louis. As such, there will be a single on-campus pick-up location for all rides. For our solutions, we define the following notations:

- P: denotes the set of all passengers in the system. A single passenger is represented by p ($p \in P$) .
- D: denotes the set of all drivers in the system. A single driver is represented by d ($d \in D$).
- M is a large negative constant (-10 in our program), such that it would not be part of any optimal solution
- t_p : denotes the destination request of a passenger, where T_p is the set of all requests. $T_p = \{ t_{p_1}\ t_{p_2}, \ldots,\ t_{p_n} \}$. Each destination t_p is identified by a unique, non negative integer (i.e., location A is denoted by the integer 1, location B is denoted by 2, and so on).
- r_t : denotes the distance from WashU campus to destination t
- s_{pt} : denotes the number of hours spent volunteering by passenger p at destination t
- $a_{d,j}$ and $b_{p,j}$: denote time availabilities of drivers and passengers, respectively. These values are binary and are assigned using the following:
    - $a_{d,j} = 1$ *if driver d is available at hour j* ($j = 1,\ 2, 3, ...,\ 24$)
      $=0$, otherwise
    - $b_{p,j} = 1$ *if passenger p requests to volunteer at hour j* ($j = 1,\ 2, 3, ...,\ 24$)
      $=0$, otherwise

The decision variables included in the program are as follows:

$x_{pdt} = 1,$ *if passenger p is assigned to driver d going to destination t*
$=0$, otherwise

$y_{dt} = 1,$ *if driver d is traveling to destination t*
$=0$, otherwise

The objective of the ride sharing system is to assign passengers to cars such that as many requests of passengers are fulfilled as possible. The integer program is as follows:

Max $\Sigma\, x_{pdt}$
s.t.

Driver must be free at passenger's selected times (hour j and hour j+$s_{p,t}$, where $s_{p,t}$ is the passenger's requested volunteering duration):

$x_{pdt} b_{p,j} - a_{d,j} y_{dt} \leq 0, \quad \forall\, p \in P,\ d \in D,\ t \in T$

$x_{pdt} b_{p,j+s_{p,t}} - a_{d,j+s_{p,t}} y_{dt} \leq 0, \quad \forall\, p \in P,\ d \in D,\ t \in T$

Each passenger can be assigned to a car only once: $\sum_{d \in D} x_{pdt} \leq 1, \ \forall p \in P$

Driver can only be assigned to one location: $\sum_{t \in T} y_{dt} \leq 1, \ \forall d \in D$

Passengers must be assigned to chosen location: $for\ all\ t \neq t_p,\ x_{pdt} = 0 \quad \forall p \in P$

$for\ all\ t = t_p,\ x_{pdt} = 1 \quad \forall p \in P$

Ensure that if a passenger is assigned to a driver, the driver must also be assigned:

$x_{pdt} + M\, y_{dt} \leq 0 \ \ \forall\, p \in P,\ d \in D,\ t \in T$

Assume that the maximum car capacity is four persons: $\sum_{p \in P} x_{pdt} \leq 4, \ \forall d \in D$

Alternate Solutions

As an alternate solution, the above constraint about car capacity was modified to incorporate car-specific capacities.

Another alternative approach we implemented was a zoning technique for volunteering locations. For all destinations T, we assigned each individual destination to a zone. With this approach, a driver may drop-off/pick-up at more than one destination, but we ensure that drivers travel to only one zone. The new notation is as follows:

Z_p denotes the zone request of all passengers, where $Z_p = \{z_{p1}, z_{p2}, \ldots, z_{pn}\}$ and z_{pn} denotes a zone that encompasses the desired location of a passenger p.

We also adapted the constraint ensuring that all passengers travel to the same destination:

Passengers must be assigned to chosen zone: $for\ all\ z \neq z_p, x_{pdt} = 0 \quad \forall p \in P$

$for\ all\ z = z_p,\ x_{pdt} = 1 \quad \forall p \in P$

VI. Cost Analysis

We set out to calculate the costs for our program and the program as it would be currently calculated by the Gephardt Institute in order to measure the financial benefits of our solutions.

Cost of Current Gephardt System

To calculate the cost using the current Gephardt system, the following formula was used:

$$Cost = (number\ of\ drivers\ assigned + number\ of\ passengers\ not\ assigned) * (3\ hour\ volunteering\ duration) * \$5\ ,$$

where $5 is the current CarShare hourly rate. This formula assumes that all passengers without a ride assignment will reserve their own CarShare and that the drivers assigned will also reserve a CarShare. We assume all volunteers will reserve the vehicles for 3 hours.

Cost of Our Solutions

The cost of our solution assumes that the drivers assigned using the program will take 2 round-trip rides to their assigned volunteering destination for a total of four times the distance from campus to the location. The mileage is for each driver is multiplied by the IRS standard reimbursement rate.

From the total costs of the assigned drivers, we then add the cost of the passengers who were unassigned. We assume that each of these unassigned passengers will reserve their own CarShare and will all have a volunteering duration of 3 hours. We then arrive at the total cost using the formula:

$$Cost = \sum (miles\ to\ location * 4) * (\$.535)\ ,\ \text{for each driver assigned}$$

$$+ [(number\ of\ passengers\ not\ assigned) * (3\ hour\ volunteering\ duration) * (\$5)]$$

VII. Implementations

Excel Implementation

We first implemented a small scenario in Excel, with only four passengers and two drivers over a four-hour time horizon. All of the decision variables and coefficients were coded into an Excel spreadsheet and constraints were incorporated using the Solver add-in. After several iterations, Solver's GRG Nonlinear solving method determined an optimal solution.

This exercise helped us refine our constraints and decision variables. One of the key decisions that occurred as a result of the Excel implementation was adding a third dimension, location, to our passenger decision variables. We also identified simpler, more easily scalable ways to write many of the constraints.

However, the Excel implementation was incredibly limited. As we scale the number of passengers and drivers, the number of constraints grows quickly. The number of constraints is related to the number of decision variables by the equation below, where P is the number of

passengers, D is the number of drivers, T is the number of destinations, and h is the length of the time frame in which volunteers requested rides.

$$Number\ of\ Constraints\ =\ (h+2)PDT\ +\ P\ +\ D\ +\ DT$$

For example, even a small problem with 5 passengers, 2 drivers, 2 locations, and an 8 hour day has 211 constraints. Due to Excel Solver's limit of 100 constraints, it quickly became infeasible to implement our solutions in the program. Accordingly, once we had refined our general integer programming solution, we turned to Matlab to implement larger programs.

Matlab Implementation

The tradeoff when moving to Matlab was an increase in problem-solving capacity at the expense of usability for non-engineers. To account for this, we set out to create a Matlab program that requires limited input from the user. In our program, the user must enter the number of passengers (P), drivers (D), and locations (T).

We assumed that the user would still use a more familiar program like Excel to record passenger and driver availabilities. The following table represents a scaled-down version of an availability spreadsheet that could be imported into our Matlab program, with the availabilities of 4 passengers and 2 drivers shown over 4 hours as an example, where a 1 indicates availability at that hour:

		Passenger 1	**Passenger 2**	**Passenger 3**	**Passenger 4**	**...**	**Passenger P**	**Driver 1**	**Driver 2**	**…**	**Driver D**
	Desired Location	A	A	B	B			-	-		
Availability	**Hour 1**	1		1				1			
	Hour 2		1		1				1		
	Hour 3	1		1				1			
	Hour 4		1		1				1		
	...										
	Hour N										

Table 8 : Example of an availability spreadsheet

Our program first creates the matrices that are used in the `intlinprog()` function, which provides an integer solution to a linear program. As stated, our initial Excel implementation was useful in developing logical methods to create matrices in the Matlab program. The following table describes the components of the matrices for each constraint that made up the A and b matrices used in the final function. The script for the Matlab program can be found in the Appendix.

Constraint	**Subpart Name**	**A Components**	**b Components**
Driver must be free at passenger's selected times	`avail`	Diagonal matrices of 1's and 0's corresponding to availabilities at each hour	Vector of 1's of length P*D*T
Each passenger can be assigned to a car only once	`passOnce`	Block diagonal of vectors of 1's of length D*T for each P	Vector of 1's of length P
Driver can only be assigned to one location	`driveOnce`	Block diagonal of vectors of 1's of length T for each D	Vector of 1's of length D
Passengers must be assigned to chosen location	`passToLoc`	Identity matrix for passengers	Vector with 1's corresponding to passenger destination
Ensure that if a passenger is assigned to a driver, the driver must also be assigned	`driverGo`	Identity matrix for passengers, along with a big N block for drivers	Vector of 0's of length P*D*T
Assume that the maximum car capacity is four persons	`fourPerson`	Identity matrix of size D*T for each passenger	Vector of 4's of length D*T

Table 9: Breakdown of Matlab program

After creating the matrices and solving the linear program, we created the script so that Matlab prints out the solution in an easily understandable format that states which driver each passenger is assigned to. The program also calculates the cost of reimbursing these drivers for their mileage at the IRS rate and compares this to what it would cost the Gephardt Institute to cover the price of a CarShare for these students. A sample of the print out is shown below in figure #.

```
Passenger 1 is matched to Driver 2
Passenger 2 is matched to Driver 7
Passenger 3 is matched to Driver 1
Passenger 4 is matched to Driver 2
Total cost is $385.848
Cost with only CarShare would be $465
17.0219 percent savings
```

Figure 5: Sample Matlab Print Out

VIII. Results

Cost Comparison

Model	Cost of Status Quo (using current Gephardt model), $	Cost of Proposed Solution, $	Percentage Savings
Monday-Saturday	180.00	125.13	30.43%
Sunday	45.00	34.26	23.87%
Weekly Totals	**255.00**	**159.39**	**37.50%**

Table 10: The cost comparison using the Gephardt CarShare model as currently instituted versus our proposed solution

Scenario		Cost of Status Quo (using current Gephardt model), $	General Solution		Alternative Solution (with zoning)	
Passengers	Drivers		Cost, $	Savings	Cost, $	Savings
50	20	420.00	285.64	31.99%	206.34	37.47%
60	20	465.00	362.52	22.04%	241.43	26.84%
70	20	540.00	434.53	19.53%	304.00	22.05%
75	20	615.00	509.53	17.15%	385.85	17.02%

Table 11: The cost comparison as the scenarios are scaled

Solution Statistics

Scenario		General Solution		Alternative Solution (with zoning)	
Passengers	**Drivers**	**Percentage of Passengers Assigned**	**Percentage of Drivers Assigned**	**Percentage of Passengers Assigned**	**Percentage of Drivers Assigned**
50	20	84.0%	80.0%	96.0%	90.0%
60	20	81.7%	95.0%	96.67%	95.0%
70	20	77.1%	100.0%	91.43%	100.0%
75	20	72.0%	100.0%	85.33%	100.0%

Table 12: Solution statistics of our general and alternative solutions as implemented in Matlab

Scenario (P=50, D=20)	Percentage of Passengers Assigned	Percentage of Drivers Assigned	Cost, $	Savings
Car size 4	84.0%	80.0%	206.34	37.47%
Car size varied 1-6	92.0%	90.0%	255.42	22.60%

Table 13: Solution statistics for alternate solution accounting for car size variation

IX. Discussion

Our Excel solution provides a user friendly workspace. We chose to start the coding of our constraints in Excel because it is a very visual program. Our Excel solution was helpful for us in thinking about presentation of results and refining our constraints. When we run the solver it places the answer (1 for yes, 0 for no) directly below the variable. The Gephardt Institute hires community engagement specialists who often have no engineering background. Their employees are familiar with the program Excel. As a result, we believe they could learn to use our Excel solution quickly and put it to good use.

Although Excel is fairly user friendly for non-engineers, it lacks the ability to handle large problems. Our example problem that we coded into the solver is very small: it has 4 volunteers. Washington University in St. Louis is lucky to have a large volunteer community. An implementable solution may need to handle dozens of volunteers and locations. Excel quickly overloads storage and run time when we increase the size of our problem. To grow our program, we needed a program with more computational power.

In addition to the growth limitation, our Excel solution requires tediously inserting driver and passenger availability. Other programs like Matlab provide the opportunity to reuse components like diagonal matrices, zero matrices, and ones matrices. The amount of hand coding

required in Excel is frustrating as an engineering student, though it may be easy to grasp for non-engineers.

To address the limitations of our Excel solution, we authored another solution in Matlab. We purposefully wrote our constraints (described above in the Integer Program Formulation section) to be mathematically specific and software generic; our constraints may be implemented in Excel and Matlab.

Our Matlab implementation uses mixed-integer linear programming (MILP). We implemented our Matlab solution using `intlinprog()` and binary variables. Our Matlab solution provides more computing ability and the ability to reuse matrix parts without typing them again.

In order to make our Matlab solution as user friendly as possible, we created the program to only require 4 inputs from a user: the number of passengers (P), the number of drivers (D), the locations (T), and a spreadsheet of student availabilities. The program is thus scalable by P, D, and T. Although the coding structure of Matlab may be more intimidating to non-technical users than Excel, we believe our comments in the code and simple four-input structure make our Matlab program widely usable.

This user friendly program was able to provide rides for the students represented by our CarShare models. Not only does our solution work for the Gephardt Institute's needs, it would save them over 30% in transportation expenses.

Seeking to go above the Gephardt Institute's needs, we ran our program on our large expanded model. Unfortunately, the matrices get so large that the program requires more RAM than is available on most WUSTL computers, including the computers in the engineering laboratories, which have 8GB. Realizing that our optimistically large model was too large, we scaled the program up until it would not work. As shown in the results section, as the program scales to our maximum of 75 volunteers, we use more of our drivers and fewer students get rides. More powerful computers and more drivers would be required to serve more students, though our current program does meet the Gephardt Institute's needs. Additionally, we were able to scale to 3.5 times the needs of the Gephardt Institute.

In addition to scaling the number of drivers, passengers, and locations in our Matlab solution, we considered additional and alternative solutions like different sized cars and zoned drop offs as described in the Integer Program Formulation section. Our program was able to accommodate different sized cars, for example a sports car or minivan. The effects of this accomodation on successful passenger matching and cost savings vary depending on driver availability (i.e. If a driver with a minivan is only available when one passenger requests to volunteer, the different car size will have no effect).

Our zoned solution provides particularly exciting results. By having one driver go to a few nearby locations, we can serve over 85% of students in even our largest model. This alternative solution also lowers the overall cost for the Gephardt Institute even more than our

general solution. We recommend utilizing the zoning technique when the program is implemented.

X. Conclusions

Our Matlab program can provide more affordable transportation for students currently using the Gephardt Institute's CarShare Fund. Furthermore, our results from our expanded volunteer model indicate it could provide affordable transportation to more students. As a result, an additional possible benefit that appeals to the Gephardt Institute is the opportunity to get more students involved in their programs and create a sense of community with students who have already been volunteering. Students who have previously been deterred from getting more involved in community service due to the large time commitment will have a less time-intensive opportunity to contribute to, and learn about, these organizations from their fellow students.

Future iterations of our solutions could consider increasing the number of passengers and drivers the Matlab code can consider before crashing; however, the current code does meet the current needs. Our contacts at the Gephardt Institute have continued to express enthusiasm for the potential savings our solutions could offer. Moving forward, we anticipate some delays in implementation of our solution as the necessary legal work is done; however, we expect ample interest and participation from students for the program to be effective once it is available.

XI. Deliverables

Our project has resulted in the following deliverables for the Gephardt Institute:

- A model that describes the behavior of student volunteers that use the current char share program as two submodels for Monday-Saturday and Sunday
- A model of student volunteer and student driver availability for the Gephardt Institute to grow into
- Optimization equations to match student volunteers with student drivers
- An Excel solution that demonstrates the implementation of the above equations
- A scalable Matlab solution that assigns student volunteers to a driver
- An alternative solution that considers drivers with different sized cars
- An alternative solution that groups volunteering locations by zone to provide more students with rides

XII. Division of Responsibilities

We followed our original timeline fairly closely, with some adjustments based on unexpected challenges. Our original timeline can be found in the Appendix. Savannah took charge of the initial Excel implementation and formulation of the general linear program. Olivia used the data we collected from the Gephardt Institute and students to create the models we used in our solutions. Kara took the lead on developing the Matlab program. We all helped to

troubleshoot when problems arose in these tasks and worked collaboratively to improve each part.

XIII. References

Gephardt Institute. "About Us." *Gephardt Institute for Civic and Community Engagement*, WUSTL, 2017, gephardtinstitute.wustl.edu/about-us/.

Hsieh, Fu-Shiung. "Car Pooling Based on Trajectories of Drivers and Requirements of Passengers ." *IEEE Xplore* , IEEE, 2017, ieeexplore.ieee.org/stamp/stamp.jsp?arnumber.

Office of Undergraduate Admissions. "Undergraduate Admissions." *Office of Undergraduate Admissions*, WUSTL, 2017, admissions.wustl.edu/.

Oregon State University. "Rideshare." *Student Leadership & Involvement*, Oregon State University , 8 July 2014, sli.oregonstate.edu/ssi/please-review/operations/transportation-options/rideshare-0.

The University of Puget Sound. "Ridesharing." *Carpools, Vanpools & Ridesharing*, The University of Puget Sound, 2017, www.pugetsound.edu/academics/academic-resources/sound-policy-institute/student-research/city/transportation/ridesharing/.

The University of Utah. "Rideshare." *Sustainability*, The University of Utah, 2017, sustainability.utah.edu/engagement/programs/rideshare/.

WUSTL. "University Facts." *Washington University in St. Louis*, WUSTL, 2017, wustl.edu/about/university-facts/.

"2017 Standard Mileage Rates for Business, Medical and Moving Announced." *2017 Standard Mileage Rates for Business and Medical and Moving Announced | Internal Revenue Service*,www.irs.gov/newsroom/2017-standard-mileage-rates-for-business-and-medical-and-moving-announced. Accessed 23 Sept. 2017.

XIV. Appendix

Matlab Script

The following script is an abbreviated version of the script used to solve the Monday-Saturday CarShare model. Repetitive segments have been eliminated to improve readability. This is denoted by `[...]`.

```
%Uses CarShare Model of Mon-Sat
%20 Passengers, 10 Drivers, and 4 Loc's over 14 hours
%Imports availability chart from Excel

%Number of passengers
P = 20;

%Number of drivers
D = 10;

%Number of locations
T = 4;

%Import availability spreadsheet
spreadsheet = xlsread('Model', 'Carshare M-S','C5:AF18');
destRequests = xlsread('Model','Large','C4:V4');

%A
%LHS

%Availabilities constraint
%Passenger availability by hour

h1pvector = zeros(1, D*T*P);
for i = 1:P
    for j = D*T*(i-1)+1:D*T*i
        h1pvector(1, j) = spreadsheet(1,i);
    end
end
h1p = diag(h1pvector);

[...]

%Combined passenger availability
passAvail = [h1p;h2p;h3p;h4p;h5p;h6p;h7p;h8p;h9p;h10p;h11p;h12p;h13p;h14p];

%Driver availability by hour
h1dvector = zeros(1, D*T);
for i = 1:D
    for j = T*(i-1)+1:T*i
        h1dvector(1, j) = spreadsheet(1,P+i);
```

```
    end
end
h1d = diag(-1.*h1dvector);

[...]


%Combined driver availability
driverAvail =
[repmat(h1d,P,1);repmat(h2d,P,1);repmat(h3d,P,1);repmat(h4d,P,1);repmat(h5d,P,
1);repmat(h6d,P,1);repmat(h7d,P,1);repmat(h8d,P,1);repmat(h9d,P,1);repmat(h10d
,P,1);repmat(h11d,P,1);repmat(h12d,P,1); repmat(h13d,P,1);repmat(h14d,P,1)];

%Complete availability matrix
avail = [passAvail driverAvail];

%Each passenger to a car only once
onesDT = ones(1,D*T); %repeated P times
passOnce = [blkdiag(onesDT, onesDT, onesDT, onesDT, onesDT, onesDT, onesDT,
onesDT, onesDT, onesDT, onesDT, onesDT, onesDT, onesDT, onesDT, onesDT,
onesDT, onesDT, onesDT, onesDT) zeros(P,D*T)];

%Drivers only drive once
onesT = ones(1,T); %repeated D times
driveOnce = [zeros(D,D*T*P) blkdiag(onesT, onesT, onesT, onesT, onesT, onesT,
onesT, onesT, onesT, onesT)];

%Passenger to desired location
passToLoc = [eye(D*T*P) zeros(D*T*P, D*T)];

%Ensure that driver is going if passenger is assigned
bigNblock = diag(repmat(-10,1,D*T));
bigN = repmat(bigNblock,P,1);
driverGo = [eye(D*T*P) bigN];

%Each car only has 4 people
fourPerson = [repmat(eye(D*T),1,P) zeros(D*T)];

%Combine LHS
A = [avail;passOnce; driveOnce; passToLoc; driverGo; fourPerson];

%B
%RHS
%All column vectors

%Availability
availB = zeros(D*T*P*14,1); %14 is the number of hours
```

```
%Each person to a car only once
passOnceB = ones(P,1);

%Drivers only drive once
driveOnceB = ones(D,1);

%Passenger to desired location
locs = eye(T);
pass1B = repmat(locs(:,p1Dest(1)),D,1);
pass2B = repmat(locs(:,p2Dest(1)),D,1);
pass3B = repmat(locs(:,p3Dest(1)),D,1);
pass4B = repmat(locs(:,p4Dest(1)),D,1);
pass5B = repmat(locs(:,p5Dest(1)),D,1);
pass6B = repmat(locs(:,p6Dest(1)),D,1);
pass7B = repmat(locs(:,p7Dest(1)),D,1);
pass8B = repmat(locs(:,p8Dest(1)),D,1);
pass9B = repmat(locs(:,p9Dest(1)),D,1);
pass10B = repmat(locs(:,p10Dest(1)),D,1);
pass11B = repmat(locs(:,p11Dest(1)),D,1);
pass12B = repmat(locs(:,p12Dest(1)),D,1);
pass13B = repmat(locs(:,p13Dest(1)),D,1);
pass14B = repmat(locs(:,p14Dest(1)),D,1);
pass15B = repmat(locs(:,p15Dest(1)),D,1);
pass16B = repmat(locs(:,p16Dest(1)),D,1);
pass17B = repmat(locs(:,p17Dest(1)),D,1);
pass18B = repmat(locs(:,p18Dest(1)),D,1);
pass19B = repmat(locs(:,p19Dest(1)),D,1);
pass20B = repmat(locs(:,p20Dest(1)),D,1);

passToLocB = [pass1B; pass2B; pass3B; pass4B; pass5B; pass6B; pass7B; pass8B;
pass9B; pass10B; pass11B; pass12B; pass13B; pass14B; pass15B; pass16B;
pass17B; pass18B; pass19B; pass20B];

%Ensure that driver is going if passenger is assigned
driverGoB = zeros(P*D*T,1);

%Each car only has 4 people
fourPersonB = repmat(4,D*T,1);

%Combine RHS
b = [availB; passOnceB; driveOnceB; passToLocB; driverGoB; fourPersonB];


%Make Binary
%# of decision var's = P*D*T + D*T
%Upper bound
ub = ones((P*D*T)+(D*T),1);
%Lower Bound
```

```
lb = zeros((P*D*T)+(D*T),1);

%Objective function
%sum of xs
%# of -1 as # of x's
%followed by # of zeros as # of ys
%row matrix has length of decision variables
f = [-ones(1,(D*T*P)) zeros(1,(D*T))];

%integer constraints
%all of our constaints are integers
%length is number of decision variables
intcon = 1:((D*T*P)+(D*T));

%Solve
 [x,fval] = intlinprog(f,intcon,A,b,[],[],lb,ub);


%Determine who is matched
 match = zeros(P,1);

 for i=1:P
     if ceil(find(x(D*T*(i-1)+1:D*T*i))/T)
         match(i) = ceil(find(x(D*T*(i-1)+1:D*T*i))/T);
     else match(i) = 0;
     end
 end

%Cost analysis
%Determine driver assignments
totalDist = 0;
assignment = zeros(1,D);
for d=1:D
    passRep = find(match==d,1);
    if(passRep)
        assignment(d) = destRequests(passRep);
    else
        assignment(d) = 0;
    end
end
driversGoing = nnz(assignment);

%Determine distances driven
for i=1:D
    if assignment(i)~=0
        totalDist = totalDist + distances(assignment(i));
    end
end
```

```
totalCost = totalDist*4*.535+(P+fval)*3*5;


carShare = (driversGoing+P+fval)*3*5;
savings = 100*(carShare-totalCost)/carShare;

%Print out in a clear format
for i=1:P
     fprintf('Passenger %i is matched to Driver %d \n', i, match(i))
end

fprintf('Total cost is $%g \n', totalCost)
fprintf('Cost with only CarShare would be $%g \n', carShare)
fprintf('%g percent savings \n', savings)
```

Project Timeline

		Month	September				October				November				
		Week of	3-Sep	10-Sep	17-Sep	24-Sep	1-Oct	8-Oct	15-Oct	22-Oct	29-Oct	5-Nov	12-Nov	19-Nov	26-Nov
Task	Responsibility														
Acquire data from the Gephardt Institute	Kara														
Perform literature review	All														
Write formal project proposal	All														
Clean the data	Savannah														
Perform descriptive statistics	All														
Determine matching criteria	Olivia														
Develop Excel program	Savannah														
Improve integer programming solutions	All														
Create formal model	Olivia														
Finalize initial Matlab Solution	Kara														
Iterate on the solutions	All														
Analyze effectiveness and cost	All														
Determine next steps for implementation	All														
Create PowerPoint and rehearse presentation	All														
Create project website and refine formal report	All														