An Intelligent and Fair GA Carpooling Scheduler as a Social Solution for Greener Transportation

Carl Michael Boukhater, Oussama Dakroub, Fayez Lahoud, Mariette Awad, Hassan Artail
Department of Electrical and Computer Engineering
American University of Beirut
Beirut, Lebanon
Emails: {cab14, ohd01, fal00, mariette.awad, hartail}@aub.edu.lb

***Abstract*—Although many carpooling systems have been proposed, most of them lack various levels of automation, functionality, practicality, and solution quality. While Genetic Algorithms (GAs) have been successfully adopted for solving combinatorial optimization problems, their use is still rare in carpooling problems. Motivated to propose a solution for the many to many carpooling scenario, we present in this paper a GA with a customized fitness function that searches for the solution with minimal travel distance, efficient ride matching, timely arrival, and maximum fairness. The computational results and simulations based on real user data show the merits of the proposed method and motivate follow on research.**

Keywords-carpooling; GA; intelligent transportation

I. INTRODUCTION

It is a common knowledge that passenger cars are way underutilized, where cars are often observed with one or two riders. In fact 78% of Americans drive alone to work [1]. Carpooling is an effective approach to exploit available transportation resources. Carpooling problems are classified as either a Daily Carpooling Problem (DCPP) or a Long-term Carpooling Problem (LTCPP) [2]. In the first class of problems, each day a set of users declare themselves as drivers for that day. The challenge is to efficiently assign passengers to these drivers to ensure the lowest cost routes and respect the users' schedules. The LTCPP is more challenging, as it is NP-Complete [3]. Users in this case can be drivers on certain days and passengers on others. It is up to the system to effectively and fairly assign roles for the different days of carpooling.

Carpooling presents many benefits. Financially, money is saved on gas, parking fees, and wear and tear on the vehicle. With an increase of carpooling, roads would become considerably less congested and parking spots more available. Moreover, carpooling is very eco-friendly. In fact, highway and off-highway vehicles contribute to roughly 80% of the total CO emissions [4]. Additionally, there are some social benefits to carpooling: less stress from daily driving and forming new friendships during ridesharing. Also, carpooling would enhance a sense of responsibility with those who tend to be late as they would become more accountable to the other commuters. Finally, leaving a car at home allows other family members to use it if need be. Despite all these benefits, carpoolers hardly constitute 15% out of all car passengers and drivers [4].

Carpooling scenarios can also be classified as many-to-one with multiple origins and one destination, one-to-many with one origin and multiple destinations, and many-to-many with multiple origins and destinations. For developing a practical system, this paper proposes for the long-term many-to-many carpooling problem a genetic algorithm (GA) with a customized fitness function. The objective is to minimize travel distance; and to offer an efficient ride matching, timely arrival of users, and fairness in driver selection. In the rest of this paper Section II presents related research, while Section III introduces the proposed model. Section IV contains computational and simulation results, whereas Section V concludes the work with future plans.

II. RELATED LITERATURE

One of the few research projects that have investigated the carpooling problem is [5] with a distributed algorithm that iteratively maps drivers with the earliest departure time to their destinations. It does not account for fairness, and gives sub-optimal assignments (maximum of picked up passengers being 80%). To start with, the authors in [6] subdivided the network of users into areas centered around a driver. A check is done on each passenger to see if a car with empty seats passes near him or her. This solution is fast, but it is globally optimal and thus it will not be fair when considering incremental driven distances. On the other hand, the Adaptive GA in [7] studied the LTCPP with a little knowledge about the search space. Drawback of this approach is the absence of individual fairness for near-optimal solutions. In a different scheme, a clustering ant colony algorithm in [8] simulated the behavior of ants searching for an objective. The algorithm reached near-optimal results when the preferences and attractiveness formulas were well selected despite the lack of the fairness component. The work in [9] proposed a Lagrangian relaxation method based on a network flow technique that models the drivers and passengers' routes and schedules. The solution is fast and efficient for small populations, but degraded quickly with large populations. Yet another approach is the one in [10], where two methods based on two integer-programming formulations are proposed while [11] present DCA Branch and Bound to globalize the obtained solution. And [12] presented a heuristic routing problem using the web, GIS, and SMS and assumed that car owners share rides to the same destination without accounting for users without cars or multiple destinations.

978-1-4799-2337-3/14/$31.00 ©2014 IEEE

One attempt to solve the LTCPP is based on the notion of savings functions is in [13]. A set of proposed automatic and heuristic data processing routines allowed efficient matching of rides and yielded more than 50 percent savings in car traveled distances when compared to having no carpooling system. On the other hand, the Multi-Matching System in [14] devised an optimized matching model to intelligently match the passengers and the riders based on Lagrangian relaxation with sub-gradient methods. It focused on the fairness aspect, by considering driving distances along with the frequency of being a driver when computing fairness. Finally, the research in [15] attempted to secure fairness when determining the set of drivers from a group of carpooling participants for a given day. In spite of accounting for fairness though, they did not consider the distance traveled or the participants' schedules.

Different from the discussed research, our work offers innovative contributions and improves the insufficiently addressed issues, such as fairness and preferences, which can affect the willingness of users to participate in carpooling. Moreover, we insure that all participants secure a ride and allow the participation of users who do not own cars.

III. PROPOSED MODEL

This section gives the definition and the analysis for the understanding of the LTCPP.

A. Problem Formulation

To ensure that all participants secure a ride, we assume that the number of cars on a certain day is enough to accommodate all the users who wish to car pool on that day. The model also necessitates that all participants provide their requested rides beforehand. This data includes the car capacity if the user owns a car, the origin and destination of the user for each trip, the desired departure and arrival times, and personal preferences if any. Preferences include desired number of users to ride with, smoking permission, and a blacklist. In addition to the aforementioned user data, the geographical coordinates of all the users and destinations, and the distance and travel time between every two nodes of the network are needed to run the model. Our model can be viewed as a graph $G = (PD \cup SR, A)$:

- PD is the set of potential drivers who own cars and can participate as drivers. Each potential driver $pd \in PD$ is associated with an origin and a destination.
- SR is the set of strictly riders who don't have cars and can only participate as passengers. Each strictly rider $sr \in SR$ is associated with an origin and a destination.
- $A = \{arc(i, j) \mid i, j \in (PD \cup SR)\}$ is the set of arcs connecting two different nodes from PD and SR. Each arc has a distance cost, and a time cost to be discussed later.

Defining a ride R of n users where $n \leq Q$, the vehicle's maximum capacity, each ride will have a driver pd and some passengers from the pool $PD \cup SR$. The driver starts the ride from his/her origin, picks up the passengers and drops them off to their destination(s), following the least cost path. The driver's and the passengers' departure and arrival times should be respected. The total cost for this ride will be the sum of the penalties incurred by every passenger in this ride as will be further clarified in the following section.

Given the users' requirement and constraints, the model aims to solve the LTCPP by minimizing the number of operating cars, assigning the role of driver or passenger for every pd while conserving individual fairness, associating passengers with each vehicle, and finding the least cost routes.

B. Objective Function

Assuming the following variables:

- OC_i: origin coordinates of user i
- DC_i: destination coordinates of user i
- $T_{D(i)}$: departure time window for user i range $[t_{LD(i)}, t_{HD(i)}]$
- $T_{A(i)}$: arrival time window for user i of range $[t_{LA(i)}, t_{HA(i)}]$
- C_i: fairness counter for user i ($i \in PD$), incremented by one if i drives, and decremented by one if i doesn't drive
- Q_i: user i's maximum car capacity

The evaluation of the quality of a ride R is based on the following three costs:

1) Distance Cost

- D_k: original distance driven by driver of car k
- W_j: waypoint j that car k has to pass through
- D_k': extra distance travelled by the driver when in pool k: $D_k' = \sum_{j \in k} d(W_j, W_{j+1}) - D_k$
- d_k: ratio representing extra distance driven by the driver of pool k

$$f(D) = d_k = \frac{D_k'}{D_k} \qquad (1)$$

2) Time Cost

- $T_{A(i)}'$: the time of arrival of passenger i at his destination $T_{A(i)}' = T_{D(i)} + \sum_{j \in k} d(W_j, W_{j+1}) \times V_{avg} \times \tau \times s$
- V_{avg} being an average speed depending on the road type
- τ is the traffic condition affected by time of day, weather conditions, and special events
- s is a security rounding factor to ensure timely arrival
- $f(T_{A(i)}')$: arrival time penalty for passenger i: $f(T_{A(i)}')=$

$$\begin{cases} 0 \; if \; T'_{A(i)} \in T_{A(i)} \\ Very\ high\ (1000) \; if \; T'_{A(i)} > t_{HA(i)} \\ 0.5 \; if \; t_{LA(i)} - 5min < T'_{A(i)} < t_{LA(i)} \end{cases} \qquad (2)$$

3) Fairness Cost

- $f(C_i)$: fairness penalty for user i:

$$f(C_i) = Constant^{(C_i + dk)} \qquad (3)$$

The fairness cost will account for the number of days and the amount of extra distance a user drives.

4) Preferences Cost

- P_i: set of preferences of user i with their relative priority, e.g., (Smoking, 1), (Only 3 passengers, 3).

- Cost (preference) = $\begin{cases} 0 \; if \; met \\ priority \; if \; unmet \end{cases}$

$$f(P) = \sum_{i \in k} \sum_{j \in Pi} cost(j) \qquad (4)$$

With a smaller sum of the four costs, the quality of the ride R will be higher. The objective function (OF) then becomes:

$$OF = f(D) + \sum_{i \in k} f(T(i)) + f(Ci) + f(P) \quad (5)$$

IV. Genetic Algorithm for Car Pooling

A. Overview

GAs are adaptive search algorithms based on natural selection and survival of the fittest. Fig. 1 describes a standard GA workflow. We modified GA to improve convergence, with the selection of an initial solution that roughly chooses the early population instead of a randomly initialized one as well as a parallel implementation of GA.

B. Initial Solution

The main components of this algorithm are the arrival and departure times of the user and the extra distance driven. The initial solution produces a set of solutions with a varied number of drivers for each solution. Initially, the drivers are randomly selected from *PD*. To assign passengers to vehicles, the algorithm performs three checks. The first one checks for the existence of at least one empty seat in the driver's car. The second one evaluates the ratio of the extra distance to be driven if the pickup takes place to the original distance to be driven without pickup. This ratio must be less than or equal to the average of the ratios computed for all pairs of drivers and unassigned passengers. The third one verifies if the pickup affects the driver's desired arrival time. If the last check isn't met, the offset from the driver's arrival time is evaluated as it must be smaller or equal to the average of the offsets computed for all pairs of drivers and unassigned passengers.

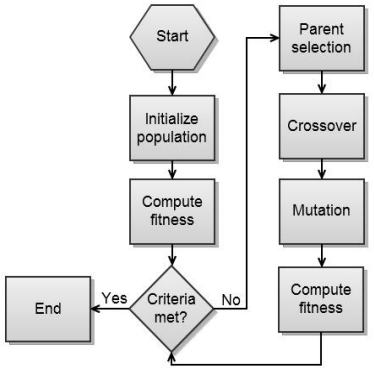

Fig. 1. Genetic Algorithm workflow

The pseudo-code shown in Fig. 2 uses the following variables and functions:

- *RAU*: set of unassigned users
- *Drivers*: set of assigned drivers
- *getRandomDriver()*: returns a random user from *PD* after removing it from the set
- *distanceRatio(Driver,User)*: returns the ratio of the distance it takes for Driver to pick up User and drop him/her off at his/her destination to the distance from Driver origin to destination
- *timeDifference(Driver,User)*: return the extra time it takes for Driver to pick up User and drop him/her off at his/her destination
- *globalDistanceRatio(Drivers,RAU)*: the average of the distanceRatio() between each User in Driver with all the Users of RAU
- *globalTimeRatio(Drivers,RAU)*: the average of the timeDifference() between each User in Driver with all the Users of RAU
- *pickUpWithinTimeWindow(Driver,User)*: return true if Driver can pick up User and stay within his/her desired time window
- *condition1*: driver has remaining car capacity > 0 && distanceRatio(Driver,User) <= globalDistanceRatio && pickUpWithinTimeWindow(Driver,User)
- *condition2*: driver has remaining car capacity > 0 && distanceRatio(Driver,User) <= globalDistanceRatio && timeDifference(User,Driver) <= globalTimeRatio
- *condition3*: driver has remaining car capacity > 0 && distanceRatio(Driver,User) <= globalDistanceRatio

```
1.  Solution ← null
2.  For i = 0, …, numberOfDrivers -1
3.     Driver ← getRandomDriver()
4.     Chromosome ← Chromosome with first
              gene set to Driver and rest is empty
5.     Solution.add(Chromosome)
6.     Drivers.add(Driver)
7.  End For
8.  RAU ← PD +SR
9.  While RAU is not empty
10.    globalDistanceRatio ←
          getGlobalDistanceRatio(Drivers,RAU)
11.    globalTimeRatio← getGlobalTimeRatio(
         Drivers,RAU)
12.    addedUsertoSolutions ← false
13.    For each Driver in Drivers
14.       For each User in RAU
15.          If condition1 is satisfied
16.            Solution(index of Drives in
                    Drivers).add(User)
17.            addedUsertoSolutions ←true
18.          End if
19.       End For
20.       If addedUsertoSolutions is false
21.          For each User in RAU
22.             If condition2 is satisfied
23.               Solution(index of Drives in
                        Drivers).add(User)
24.               addedUsertoSolutions ←true
25.             End if
26.          End For
27.       End if
28.       If addedUsertoSolutions is false
29.         For each User in RAU
30.            If condition3 is satisfied
31.                Solution(index of Drives in
                          Drivers).add(User)
32.                addedUsertoSolutions ←true
33.             End if
34.         End For
35.       End if
36.    End For
37. End While
    Return Solution
```

Fig. 2. Initial solution for a predefined number of drivers

C. Population Encoding

Each chromosome as shown in Fig.3, represents a ride and the genes the car occupants. The set of all chromosomes belonging to a thread constitutes a solution. Within a thread and after every evolution, the genes are swapped and/or mutated to obtain better rides. The number of drivers is bound to the number of chromosomes in the population. The passengers as listed in Fig.3, are not in the order of pickup. Instead the path is computed using the origin and destination coordinates of each user in the chromosome, and fitness evaluations are based on these calculations.

Fig. 3. Sample chromosome

D. Fitness and Operators

Since the fitness of a chromosome should reflect the quality of the ride, it should thus include distance, travel time and fairness metrics. Thus, the fitness function (FF) is:

$$FF = \begin{cases} 50 \; if \; solo \; driver \\ 100 - \; \mathrm{f(D)} + \sum_{i \in k} \mathrm{f(T(i))} + \mathrm{f(Ci)} + \mathrm{f(P)} \; otherwise \end{cases} \quad (6)$$

The FF attributes different weights to the four costs:

1) f(D) ranges between 0 and 100; any value greater than 100 is clamped.

2) f(T(i)) has a value of 0 for a solution in the time window, and a value of 100 for exceeding the time limit.

3) f(C_i) is an exponent of a constant. The greater the number of days a user has to drive, the faster this function grows.

4) f(P) is a small integer reflecting the priority that a user attributes to his preferences.

GA standard operators have also been modified and one was created to suit the requirements of the car-pooling model:

1) Crossover: given 2 chromosomes, 2 random indexes are computed to determine the crossover positions. The chromosomes are swapped only if the resulting 2 chromosomes are better than their parents while a check tests if the first position is still a driver.

2) Mutation: The mutation is a swap between two different chromosomes because a user cannot be completely removed from the solution. The mutation is kept and the population is updated only if the sum of the new fitness from the two new chromosomes is greater than their parents'.

3) DriverSwitch: Since the driver in the chromosome might not be the best driver to take the passengers, this operator randomly switches the driver with another potential driver from the genes. Only if the resulting chromosome turns out to be better than the previous one, changes are saved.

V. Simulation Results

In order to test the proposed model on real world scenarios, we performed multiple tests using real user data consisting of 119 students wishing to car pool to four different university campuses in Lebanon. To build the model and the solution algorithm, the Java computer language was used. Since the preferences cost function is still not finalized, it was disregarded in the tests for the time being. The tests were performed on an Intel Core i7-3632QM CPU @ 2.20GHz and 6 GB of RAM in the environment of Microsoft Windows 8.

A. Advantage of the Initial Solution Algorithm

This test is run with the full population of 119 users and four destinations. Case A runs the proposed model as described in this paper. Case B runs the GA without the proposed initial solution algorithm, i.e. using a standard GA encoding. The results are summarized in Table I where:

- Fitness: as in (6), but without the preferences component.
- Time Penalty: reflects if users arrived according to their desired schedules. A value of 0 indicates that all users of the corresponding ride arrived as desired. A value between 0 and 100 indicates that one or more users arrived earlier than desired. A value of 100 or above means one or more users arrived late.
- Increase in distance (%): is the % of extra distance incurred by the driver when picking up other passengers.
- Fairness: fairness score as calculated in (3).
- # Used cars: the number of operating cars
- # Solo drivers: the number of cars occupied by one driver

The results confirm the usefulness of adding the proposed initial solution: all metrics scores are higher for case A as compared to case B. Also, as shown in Fig. 4, case A converges faster than case B. The average fitness value of 90.4 is reached in 480 iterations for case A, versus 818 for case B.

TABLE I. Impact of Initial Solution

Metrics	Cases	
	A (with initial solution)	***B (without initial solution)***
Min/Max/Avg Fitness	50/97.7/90.4	50/98.6/89.3
Min/Max/Avg Increase in Distance (%)	0/18/3.97	0/12.8/4.1
Min/Max/Avg Time Penalty	0/9/3.57	0/11/4.46
Min/Max/Avg Fairness	0/1.47/0.75	0/1.47/0.80
# Used Cars	35	35
# Solo Drivers	1	1

B. Different fitness functions

The purpose here is to investigate the impact of these three costs on the quality of the car-pooling solution. The test results are summarized in table II. In case A, the fitness function accounts for the distance, time, and fairness costs, while disregarding the preferences cost; in case B, for the distance cost only; and in case C, for the time cost only. It can be noted that in case B, a time penalty of 101 is seen. This means that in at least one ride, a certain number of users arrived late to their destination(s). In case C, an increase in distance of 4497% is observed. This means that at least one driver is driving almost 45 times the distance s/he would have driven if no passengers were picked up.

TABLE II. IMPACT OF DIFFERENT FITNESS FUNCTIONS

Metrics	Cases		
	A (3 costs)	*B (distance cost)*	*C (time cost)*
Min/Max/Avg Complete Fitness	50/97.7/90.3	-14/98.7/86.3	-9645/82/-740
Min/Max/Avg Edited Fitness	50/97.7/90.3	87.3/99.9/95	50/100/94.5
Min/Max/Avg Increase in Distance (%)	0/18/3.97	0/12.7/4.7	15.9/4497/644
Min/Max/Avg Time Penalty	0/9/3.57	0/101/8	0/2/0.25
Min/Max/Avg Fairness	0/1.47/0.75	0/1.47/0.96	0/5247/189
# Used Cars	35	31	28
# Solo Drivers	1	0	3

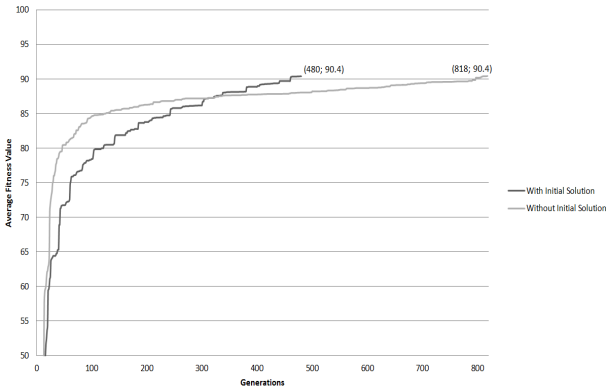

Fig. 4. Average fitness value versus number of generations for cases A and B

C. Sparse versus Clustered Distribution of Users

To study the effect of geographical distribution, a third test was done with a very sparse population and a highly clustered one. Table III shows the results, where case A is for a population made of 8 clusters of 20 users each, and case B. Both scenarios produced very good results: the average fitness of case A is 89.4 and that of case B is 82.1. The increase in driven distance does not exceed 10% and no late arrivals occurred. Another advantage of the proposed approach is that it allows for a global solution (Fig. 5), which represents rides from case A. The nodes and path illustrations were done using Google Maps JavaScript API v3 [15]. In the figure, driver A picks up passengers B, C, D, and E and continues to the first destination F where s/he drops off one or more passengers, then to the final destination G. All the picked up passengers are from the same cluster, which alludes to a local solution.

TABLE III. TEST RESULTS FOR DIFFERENT POPULATION DISTRIBUTIONS

Metrics	Cases	
	A (clustered)	*B (sparse)*
Min/Max/Avg Fitness	0.56/99/89.4	50/98.8/82.1
Min/Max/Avg Increase in Distance (%)	0/23/6.4	1.4/24.1/9.4
Min/Max/Avg Time Penalty	0/13/2.31	0/12/4.1
Min/Max/Avg Fairness	0/1.5/0.78	0/1.5/0.87
# Used Cars	41	41
# Solo Drivers	1	1

VI. Conclusion

In this paper, a model is proposed to solve the LTCPP, where the presented approach is based on a customized GA coupled with a structured initial solution. Preliminary tests have been performed to assess the efficiency of the model in different scenarios. Finalizing user preferences and implementing a user rating system will be the subject of future work. Future research will also involve enhancements to the algorithm to allow sudden change of schedules.

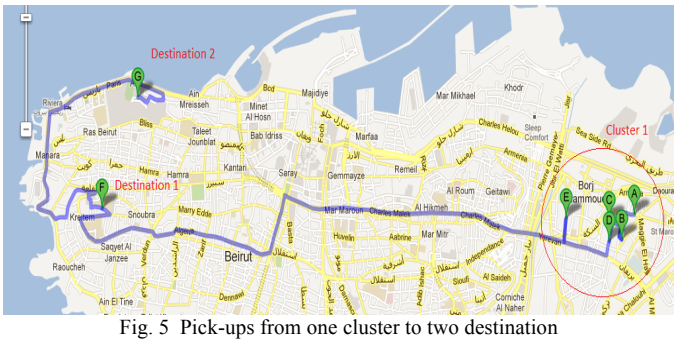

Fig. 5 Pick-ups from one cluster to two destination

REFERENCES

[1] G. Eason, B. Noble, and I. Sneddon, "On certain integrals of Lipschitz-Hankel type involving products of Bessel functions," Phil. Trans. Roy. Soc. London, vol. A247, pp. 529–551, April 1955.

[2] Varrentrapp, K., Maniezzo, V., Stutzle, T.: The Long Term Car Pooling Problem: On the Soundness of the Problem Formulation and Proof of NP-completeness. Technische Universitat Darmstadt (2002)

[3] Maniezzo, V., Carbonaro, A., Hildmann, H.: An ANTS heuristic for the long-term car pooling problem. In: New Optimization Techniques in Engineering, pp. 411–430 (2004)

[4] Son, Ta Anh, Le Thi Hoai An, Pham Dinh Tao, and Djamel Khadraoui. "A Distributed Algorithm Solving Multiobjective Dynamic Car Pooling Problem." *Int'l Conf. on Computer & Information Science.* 2012.

[5] Sghair, Manel, Hayfa Zgaya, Slim Hammandi, and Christian Tahon. "A Distributed Dijkstra's Algorithm For The Implementation Of A Real Time Carpooling Service With An Optimized Aspect On Siblings." *IEEE Annual Conf. on Intelligent Transportation Systems.* Madeira Island, Portugal, 2010.

[6] Guo, Yuhan - Goncalves, Gilles - Hsu, Tienté. *A Multi-agent Based Self-adaptive Genertic Algorithm for the Long-term Car Pooling Problem.* Springer Science Business Media B.V. 2012, 2011.

[7] Guo, Yuhan, Gilles Goncalves, and Tienté Hsu. "A Clustering Ant Colony Algorithm for the Long-term Car Pooling Problem." *Int'l Conf. on swarm interlligence.* Lille, France, 2011.

[8] Yan, Shangyao, Chun-Ying Chen, and Yu-Fang Lin. "A Model With a Heuristic Algorithm for Solving the Long-term Many-to-Many Car Pooling Problem." *IEEE TRANSACTIONS ON INTELLIGENT TRANSPORTATION SYSTEMS*, 2011.

[9] Baldacci R., Maniezzo V. and Mingozzi A.: An Exact Method for the Car Pooling Problem Based on Lagrangean Column Generation. Oper. Res. 52(3) (June 2004) 422-439

[10] Ta Anh Son, Le T hi Hoai An, Gerald Arnould, Djamel Khadraoui and Pharn Dinh Tao: Solving Car Pooling Problem using DCA, Global Information Infrastructure Symposium, Danang Aug. 2011, pp.1-6.

[11] Calvo R. W., Luigi F. L., Haastrup P. and Maniezzo V: A distributed geographic information system for the daily car pooling problem, Computers and Operations Research, 31 (2004) 2263-2278

[12] E. Ferrari, R. Manzini. The car pooling problem: heuristic algorithms based on savings functions. Journal of Advanced Transportation, 37(3):243-272, 2003.

[13] Y. Lin. Matching Model and Heuristic Algorithm for Fairness in the Car pool Problem. PhD Thesis, National Central University, Taiwan, 2009.

[14] Williams, & Fagin. (1983). A Fair Carpool Scheduling Algorithm. IBM Journal of Research, 133-139.

[15] Google maps api family. (n.d.). Retrieved from h https://developers.google.com/map