# The ridematching problem with time windows in dynamic ridesharing: A model and a genetic algorithm

Wesam Herbawi and Michael Weber
Institute of Media Informatics
University of Ulm
Ulm, Germany
Email:{wesam.herbawi, michael.weber}@uni-ulm.de

***Abstract*—The increasing ubiquity of mobile handheld devices paved the way for the dynamic ridesharing which could save travel cost and reduce the environmental pollution. The ridematching problem with time windows in dynamic ridesharing considers matching drivers and riders with similar routes (with drivers detour flexibility) and time schedules on short notice. This problem is hard to solve.**

**In this work, we model the ridematching problem with time windows in dynamic ridesharing as an optimization problem and propose a genetic algorithm to solve it. We consider minimizing the total travel distance and time of the drivers (vehicles) and the total travel time of the riders and maximizing the number of the matches. In addition, we provide datasets for the ridematching problem, derived from a real world travel survey for northeastern Illinois, to test the proposed algorithm. Experimentation results indicate that the idea of dynamic ridesharing is feasible and the proposed algorithm is able to solve the ridematching problem with time windows in reasonable time.**

## I. Introduction

The shared use of vehicle by its driver and one or more passengers (riders) is called ridesharing. The idea of ridesharing goes back to the 1940s during the World War II as a way to conserve resources for the war. Since then, the idea of ridesharing passed through many motivations and goals [4]. Recently, there is an increasing interest in ridesharing as response to the environmental pollution and climate change, traffic congestion and oil cost. The increasing ubiquity of mobile handheld devices also made the idea of ridesharing to be more appealing. It takes the idea of ridesharing from static to dynamic ridesharing. In dynamic ridesharing, the process of matching the riders and drivers to form ridesharing is done on very short notice [1].

In this study, we are interested in the ridematching problem with time windows (RMPTW) in dynamic ridesharing. An example RMPTW is shown in Figure 1. The problem consists of a set of drivers' offers (representing a set of vehicles) and riders' requests. Each participant (driver or rider) specifies a source and a destination location for her/his trip. In addition, each participant specifies an earliest departure time and a latest arrival time which we use to define a time window at each location as we show later. Drivers define maximum travel time and distance for their trips and they are willing to detour to take some riders. Riders define maximum travel time for their trips. The RMPTW is to assign riders to drivers and determine the timing and order of the pickup and delivery of riders. From now on, we will use the word vehicle instead of driver.

The RMPTW is an optimization problem and it is subject to multicriteria optimization. In this study we consider the following criteria to optimize (We use the word vehicle instead of driver):

$c_1$. The total distance of vehicles' trips (to be minimized)
$c_2$. The total time of vehicles' trips (to be minimized)
$c_3$. The total time of the trips of the matched riders' requests (to be minimized)
$c_4$. The number of matched (served) riders' requests (to be maximized)

Note that the trip's time might include some waiting time (A vehicle with possibly some riders is waiting to pick up a rider when time allows). Criterion $c_4$ is conflicting with the other three criteria. Also criteria $c_1$ and $c_2$ could be conflicting with criterion $c_3$ as vehicles might tend to take riders for longer time to reduce their own time and distance.

The RMPTW could be seen as a pickup and delivery problem with time windows (PDPTW) which is a hard optimization problem. Genetic algorithms are search metaheuristics that use techniques inspired by natural evolution to solve hard optimization and search problems. The idea of genetic algorithms showed success in solving many hard optimization problems and therefore we utilize it to solve our problem.

In this study we model the ridematching problem with time windows in dynamic ridesharing and propose a genetic algorithm for solving it. In addition we provide real world datasets for the RMPTW extracted from a travel and activity survey for northeastern Illinois conducted by Chicago Metropolitan Agency for Planning. The datasets are used to test the proposed genetic algorithm and could be used as benchmarks.

The rest of the paper is organized as follows. In Section II, we formally describe the problem and provide the mathematical model. We discuss some related work in Section III. The proposed genetic algorithm is explained in Section IV followed by the experimentation and results in Section V. Finally we outline the conclusions of the paper.

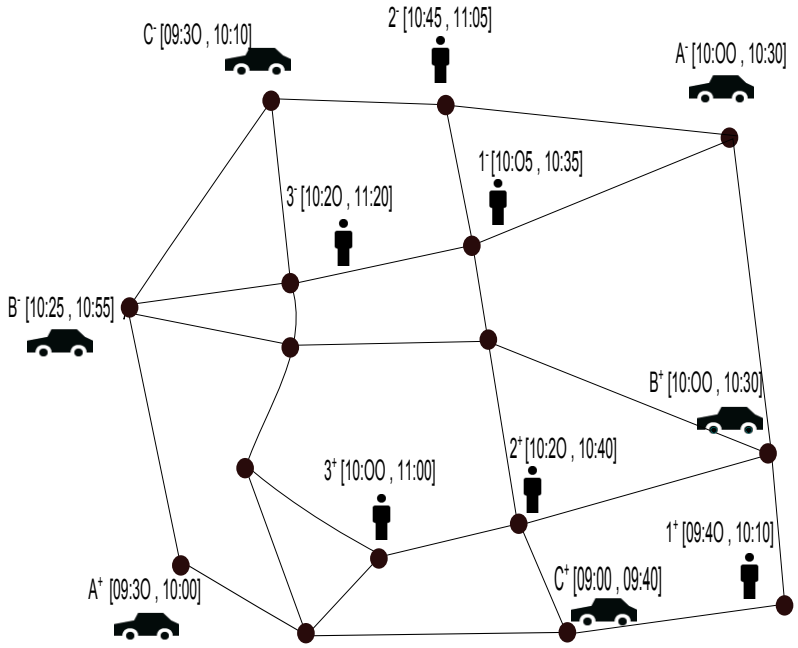


Fig. 1. Ridematching problem with time windows. + and - denote sources and destinations respectively

## II. Formal problem definition and the mathematical model

The ride matching problem with time windows is similar to the PDPTW especially the dial-a-ride problem (DAR). Therefore the formulated mathematical model is very similar to the DAR problem model in [8] which in turn is an extension to the pickup and delivery problem model in [13] and [3]. The main difference between the RMPTW problem and the DAR problem is that, in RMPTW the vehicles are not free to move everywhere\when and limited by the vehicles' sources and destinations (with possible detour flexibility) and time windows. We adapt the model of DAR problem for the RMPTW and modify the definition of the objective function and add additional constraints on it (Constraints 12, 13 and 14 in the model).

The problem consists of a set $R = \{1, 2, ..., n\}$ of $n$ riders' requests and a set $V = \{2n + 1, 2n + 2, ..., 2n + v\}$ of $v$ vehicles representing drivers' offers, $R \cap V = \varnothing$. For each rider's request $i \in R$, we have a pickup point $i$, a delivery point $i + n$, demand $dm_i$ defining the numbers of persons to be delivered from point $i$ to point $i + n$, earliest departure time $ED_i$ from point $i$, latest arrival time $LA_i$ to point $i + n$ and constants $AT_i$ and $BT_i$ to define the rider's maximum travel time $MTT_i$. Let $t_{i,j}$ be the direct travel time between points $i$ and $j$. We compute $MTT_i$ as done in [7]: $MTT_i = AT_i + BT_i \cdot t_{i,i+n}$. For each pickup point $i$ and delivery point $i + n$, we compute a time window as follows: [$a_i$,$b_i$] = [$ED_i$, $LA_i - t_{i,i+n}$] denoting the earliest and latest pickup time and [$a_{i+n}$,$b_{i+n}$ ] = [$ED_i + t_{i,i+n}$, $LA_i$ ] denoting the earliest and latest delivery time. Figure 2 shows the calculation of the time windows.

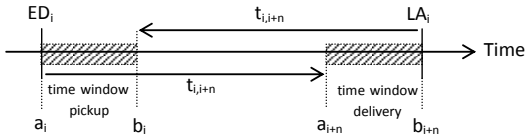


Fig. 2. Time window calculation at pickup point $i$ and delivery point $i + n$ for request $i$

For each vehicle $k \in V$ we have a source point $k$, destination point $k + v$, a maximum capacity $C^k$, earliest departure time $ED_k$ from point $k$, latest arrival time $LA_k$ to point $k + v$, and constants $AT_k$, $BT_k$, $AD_k$ and $BD_k$ to define the vehicle's maximum travel time $MTT_k$ and distance $MTD_k$. The calculation of the time windows for the points $k$ and $k + v$ ([$a_k$, $b_k$] and [$a_{k+v}$, $b_{k+v}$]) that denote the earliest and latest departure time and the earliest and latest arrival time of vehicle $k$ respectively is the same as for riders points. The calculation of the maximum vehicle's travel time $MTT_k$ is the same as for riders. Let $d_{i,j}$ denote the direct travel distance between points $i$ and $j$, then the maximum vehicle's travel distance is $MTD_k = AD_k + BD_k \cdot d_{k,k+v}$.

Let $P = \{1, ..., n\}$ denote the set of pickup points and $D = \{n + 1, ...2n\}$ denote the set of delivery points. The set of all pickup and delivery points is $N = P \cup D$ and the set of all points including the vehicles' sources and destinations is $A = N \cup \{k, k + v\}\ \forall k \in V$. Let $l_i = dm_i$ and $l_{i+n} = -dm_i$ represent the load change at points $i$ and $i+n$ respectively. The load after servicing point $i$ is $L_i^k$. Let $T_i^k$ denote the service starting time at point $i$ by vehicle $k$. It denotes the pickup and delivery times for riders and the departure and arrival times for vehicles (drivers). The service time at point $i \in N$ (time for pickup or delivery) is $s_i$ and $\forall i \in A \setminus N$, $s_i = 0$. A binary decision variable $x_{i,j}^k$ is set to 1 if vehicle $k$ services point $i$ and

travels directly to point $j$ to service it. it is set to 0 otherwise. The multicriteria objective function that minimizes the total distance and time of the vehicles' trips and the total time of the trips of the matched riders' requests and maximizes the number of matched riders' requests $r$ is provided in Equation 1. The weights $\alpha$, $\beta$, $\gamma$ and $\delta$ define the relative importance of the different components.

$$min\ \alpha \sum_{k \in V} \sum_{i,j \in A} x_{i,j}^k d_{i,j} + \beta \sum_{k \in V} (T_{k+v}^k - T_k^k) + \gamma \sum_{k \in V} \sum_{i \in P} (T_{i+n}^k - s_i - T_i^k) + \delta(n - r) \quad (1)$$

subject to:

$$\sum_{k \in V} \sum_{j \in P \cup \{k+v\}} x_{k,j}^k = v \quad (2)$$

$$\sum_{k \in V} \sum_{i \in D \cup \{k\}} x_{i,k+v}^k = v \quad (3)$$

$$\sum_{k \in V} \sum_{j \in A} x_{i,j}^k \leq 1, \forall i \in N \quad (4)$$

$$\sum_{j \in N} x_{i,j}^k - \sum_{j \in N} x_{j,i+n}^k = 0, \forall k \in V, i \in P \quad (5)$$

$$x_{i,j}^k (T_i^k + s_i + t_{i,j} - T_j^k) \leq 0, \forall k \in V, i, j \in A \quad (6)$$

$$a_i \leq T_i^k \leq b_i, \forall k \in V, i \in A \quad (7)$$

$$T_i^k + s_i + t_{i,i+n} \leq T_{i+n}^k, \forall k \in V, i \in P \quad (8)$$

$$x_{i,j}^k (L_i^k + l_j - L_j^k) = 0, \forall k \in V, i, j \in A \quad (9)$$

$$l_i \leq L_i^k \leq C^k, \forall k \in V, i \in P \quad (10)$$

$$L_k^k = L_{k+v}^k = 0, \forall k \in V \quad (11)$$

$$\sum_{i,j \in A} x_{i,j}^k d_{i,j} \leq MTD_k, \forall k \in V \quad (12)$$

$$(T_{k+v}^k - T_k^k) \leq MTT_k, \forall k \in V \quad (13)$$

$$(T_{i+n}^k - s_i - T_i^k) \leq MTT_i, \forall i \in P \quad (14)$$

$$x_{i,j}^k \in \{0, 1\}, \forall i, j \in A \quad (15)$$

Constraints (2) and (3) ensure that the number of vehicles leaving their sources is equal to the number of vehicles arriving to their destinations. Constraint (4) ensures that a rider's request is matched at most once and constraint (5) ensures that both the pickup and delivery of the rider are performed by the same vehicle. Constraint (6) ensures that the minimal travel time between two consecutive points is not violated and constraint (7) ensures that the time windows are not violated. Constraint (8) is a precedence constraint where a pickup point has to be visited before its corresponding delivery point. Constraint (9) ensures that if vehicle $k$ has a load $L_i^k$ after servicing point $i$, it must have a load of $l_j + L_i^k$ after servicing point $j$ and therefore it ensures that the number of loaded riders at a pickup point should be equal to the number of unloaded riders at its corresponding delivery point.

Constraint (10) ensures that the vehicles' capacities are not exceeded and the load of the vehicle is at least $l_i$ after servicing a pickup point $i$. Constraint (11) ensures that vehicles depart without having riders onboard and arrive to their destinations without having riders onboard (Note that riders and vehicles might share the same geographic sources and destinations, however we consider their sources and destinations to be distinct points). Constraints (12) and (13) ensure that the maximum distance and time, respectively, for each vehicle's trip is within a defined value. Constraint (14) ensures that the time of the trip of each rider is within his defined value.

## III. Related work

In a previous work [6] [5], we have proposed an evolutionary multiobjective route planning algorithm for the dynamic multihop ridesharing problem. The idea was to find the set of best route plans for a single rider's request where a route plan might include sharing the ride with multiple drivers. The riders' routes were supposed to be known and fixed. In this work, we extend our previous work to cover the more practical configuration to match a set of riders (not only one) with a set of drivers with time windows and drivers' detour possibility.

Agatz et al. [2] considered a similar matching problem in dynamic ridesharing. They have proposed optimization-based techniques for solving the matching problem. Their results indicate that the use of sophisticated optimization methods improves the performance of ridesharing systems. However, in their problem definition, a driver can make only one pickup and one delivery. While this assumption makes the problem easier to solve, it prevents the driver from taking some riders even if they are on his exact way. In this work, we consider the possibility of making more than one pickup and delivery where we could simply put a constraint on the number of such operations.

In their work Jorgensen et al. [10], they have proposed a genetic algorithm for solving the dial-a-ride problem. The problem is very similar to the ridematching problem with the latter having more constraints specified by the drivers (vehicles are not free to move everywhere/when). The additional constraints might reduce the search space but also might make its structure harder. Their genetic algorithm is based on the classical cluster-first, route-second approach. They cluster the customers using a genetic algorithm and route using a heuristic algorithm. The problem in this approach is that the result of the routing defines the quality of the clustering where the heuristic routing is subject to stuck at local optima (especially with the

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| $A^+$ | $5^+$ | $5^-$ | $4^+$ | $7^+$ | $4^-$ | $7^-$ | $A^-$ | | |
| $B^+$ | $1^+$ | $3^+$ | $3^-$ | $1^-$ | $B^-$ | | | | |
| $C^+$ | $8^+$ | $2^+$ | $8^-$ | $2^-$ | $C^-$ | | | | |
| $D^+$ | $6^+$ | $6^-$ | $D^-$ | | | | | | |
| $E^+$ | $10^+$ | $10^-$ | $9^+$ | $11^+$ | $9^-$ | $11^-$ | $12^+$ | $12^-$ | $E^-$ |

Fig. 3. Solution representation. Five Vehicles (A-E) and twelve matched riders' requests (1-12). + - denote the pickup and delivery points of riders' requests and the sources and destinations of the vehicles

presence of time windows) as are all heuristic algorithms. In our work, the proposed genetic algorithm performs both the clustering and routing tasks with the penalty of requiring repair operations. This helps avoid the problem of local optima.

Kamar and Horvitz [9] developed computational methods for guiding collaboration that demonstrate how shared plans can be created in ridesharing. They consider the problem as a set cover problem where each participant has his own car and the problem is to group them and select a driver for each group to minimize the operational cost. They consider a maximum group size limited by the capacity of the vehicle and they did not take time windows into consideration.

## IV. Genetic Algorithm

In this section, we describe the different components of the proposed genetic algorithm. The proposed algorithm is a generational genetic algorithm gGA that follows the general skeleton of gGAs.

### A. Solution Representation

A solution is represented as a schedule of $v$ routes, one route for each vehicle. Each route starts with the source point of the vehicle followed by a list of pickup and delivery points and ends with the destination point of the vehicle. The pickup and delivery points are inserted satisfying the constraints defined in the model. A solution may not match all riders' requests and therefore we keep a list of all non-matched requests. Figure 3 shows an example solution representation matching 5 vehicles (A - E) and 12 riders' requests (1 - 12).

### B. Solution Initialization

An initial solution is created as follows. We randomly select a route to initialize among the $v$ routes. The first two points to be inserted in the route are the source and destination points of the vehicle. Initially the two points are scheduled at their earliest time $a_i$ (to take advantage of the maximum possible push forward as below). Then we continuously select a random pair of pickup and delivery points (from a set of feasible points for the current vehicle (route) taking the maximum vehicle's travel time and distance into consideration) and try to insert them in the route. The process repeats for all routes.

During the initialization process, we adopt the Solomon's insertion method [12]. Given a partially created route $route = \{p_1, p_2, ..., p_u\}$ for vehicle $k$, the service time at $p_i$ is $T^k_{p_i} = max\{a_{p_i}, T^k_{p_{i-1}} + s_{p_{i-1}} + t_{p_{i-1},p_i}\}$. If the vehicle arrives too early at point $p_i$, then it should wait before servicing $p_i$ and the waiting time at $p_i$ is $w_{p_i} = max\{0, a_{p_i} - T^k_{p_{i-1}} - s_{p_{i-1}} - t_{p_{i-1},p_i}\}$.

Now we discuss the necessary conditions for time feasibility when inserting point $p$ between the points $p_{i-1}$ and $p_i$, $1 < i \leq u$. Let $newT^k_{p_i}$ denote the new service time by vehicle $k$ for point $p_i$ after inserting the new point $p$ before $p_i$. Assuming that the triangular inequality holds both on time and distance between the points, then the insertion of $p$ will result in a time push forward (PF) in the route at $p_i$ such that $PF_{p_i} = newT^k_{p_i} - T^k_{p_i} \geq 0$. It will also result in a time push forward at the points after $p_i$ such that $PF_{p_{j+1}} = max\{0, PF_{p_j} - w_{p_{j+1}}\}$, $i \leq j < u$. After the time push forward, the time window and/or the maximum travel time constraints of some point $p_j$, $i \leq j \leq u$, might be violated. We sequentially check these two time feasibility constraints until we reach a point $p_j$ with $PF_j = 0$ or its time window or maximum travel time constraint is violated or in the worst case until $j = u$.

In addition to time feasibility, we check the driver's maximum distance constraint after inserting the new point $p$. The distance should be less than $MTD_k$. For the precedence constraint, we always insert the pickup point before it corresponding delivery point and we search for a feasible insertion position for the delivery point in the route to the right of its pickup point.

### C. Crossover Operator

We have defined a single point crossover operator. Given two parent solutions, we randomly select a crossover position (route index) and we make the crossover by exchanging the routes among the two parents starting from the crossover position upwards as in Figure 4. Note that the resulting children of the crossover do not violate any of the defined constraints except constraints (4) and (5) ( a riders' request could be matched twice with different vehicles). Example violation points are shaded in the child solutions. In addition to the constraints violation, some pairs of pickup and delivery points that exist in the parents might disappear in the children because of the crossover (e.g. points $10^+$ and $10^-$ in child 1).

After the crossover operation, we apply a repair operation for the children. The repair operation removes all the pickup and delivery points in the routes after the crossover position whenever they exist in any route before the crossover position. Besides the repair operation, we also try to insert the pickup and delivery points of the riders' requests that are not matched in the resulting children after each crossover in a process similar to the one during solutions initialization.

### D. Mutation Operators

We have defined five different mutation operators. All of them are on the route level (gene level). A mutation operation that violates any of the constraints is rejected.

The first mutation operator is the push backward. In this operator, we randomly select a point $p_i$ from the route and define the push backward ($PB_i$) value as a random percentage of the time difference between the service time of the selected point and its earliest service time: $PB_i = random(T^k_{p_i} - a_{p_i})$.

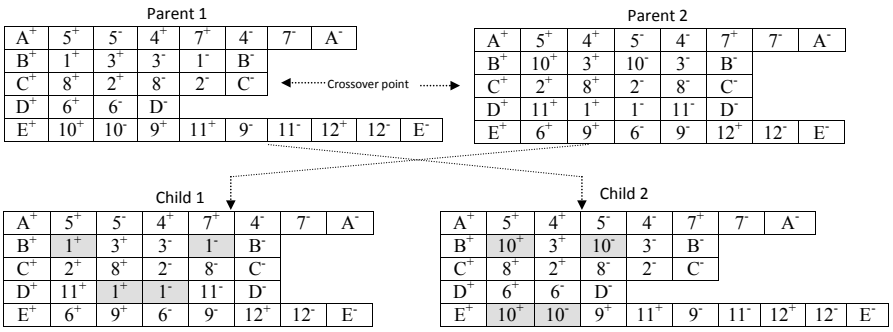


Fig. 4. Single point crossover operator. Shaded points violate constraints (4) and (5)

We also push backward all the points after $p_i$ in the route: $PB_{p_{j+1}} = min\{PB_j, T^k_{p_{j+1}} - a_{p_{j+1}}\}$, $i \leq j < u$. We stop the push backward operation when we reach a point $j$ with $PB_j = 0$ or when $j = u - 1$

The second mutation operator is the push forward (very similar to initialization push forward). In this operator, we randomly select a point $p_i$ from the route and define the push forward ($PF_i$) value as a random percentage of the time difference between the latest service time of the selected point and its service time: $PB_i = random(b_{p_i} - T^k_{p_i})$. We also push forward all the points after $p_i$ in the route: $PF_{p_{j+1}} = max\{0, PF_{p_j} - w_{p_{j+1}}\}$, $i \leq j < u$. We stop the push forward operation when we reach a point $j$ with $PF_j = 0$ or when $j = u - 1$.

The third mutation operator is the remove-insert operator. We randomly select a pair of pickup and delivery points to delete from the route (and mark their request as non-matched). After the delete operation, we perform a push backward operation in the positions of the deleted points. By the end of the delete operations, we try to insert as much pairs of pickup and delivery points as possible from the points of the non-matched requests.

The fourth mutation operator is the transfer mutation. In this operator, we randomly select a pair of pickup and delivery points from the route and try to insert them in another route. The last mutation operator is the swap mutation operator. In this operator, we swap a randomly selected point $p_i$ in the route and its neighbor point $p_{i+1}$.

We noticed that the push forward and push backward mutation operators are the most important among the mutation operators. However, still the other three mutation operators participate in the quality of the solution.

## V. Experimentation and results

In this section, we describe the proposed real world datasets for ridematching and the experimentation methodology and discuss the experimentation results.

### A. Experimentation data

To test the behavior of the proposed algorithm for solving the RMPTW, We have utilized a travel and activity survey for northeastern Illinois conducted by Chicago Metropolitan Agency for Planning CMAP [1]. Data collection took place between January 2007 and February 2008 and covered a total of 10,552 households participant. A trip in the study is defined as a pair of source and destination positions (using latitude and longitude) with departure and arrival timestamps. We have aggregated trips from six days of the survey. This resulted in a total of 698 trips among them 469 travel with their own vehicles and the rest rely on other forms of transportation.

We have engineered a time window for each trip. 250 participants among the 469 with their own vehicles have been selected to be as drivers in our experiment and the rest 448 participants as riders. The selection of the 250 drivers makes the difference between our two datasets. In the first dataset (RMPTW698_L), we have selected the drivers with the longest travel distance while in the second dataset (RMPTW698_R) we have selected them randomly. The total drivers direct travel distance and time for RMPTW698_L are $4602.6$km and $4717$min respectively and for RMPTW698_R are $2451.9$km and $2565$min respectively. We consider a vehicle speed of $60$km/h. The distance between two points is measured using the Haversin formula [11] and the travel time between them is the ceil of the computed travel time (always integer).

### B. Experiment settings

The parameters' settings (defined in Sec. II) of the problem instances are $AT_i = 0$ and $BT_i = 1.3$ for all riders' requests and vehicles (drivers' offers), $C^k = 5$, $AD_i = 0$ and $BD_i = 1.3$ for all vehicles and $s_i = 0$ and $dm_i = 1$ for all riders' requests. The criteria weights are: $\alpha = 0.7$ and $\beta = \gamma = \delta = 0.1$.

The parameter settings of the genetic algorithm are: population size = $100$, number of generations = $100$, crossover probability = $1.0$ and mutation probability =$0.4$.

### C. Results discussion

First we provide the results of a single run of the genetic algorithm on the instance RM698_L and we visualize the value of the objective function and its components through different generations. Then we provide the statistical results of 30 independent runs of the algorithm on the two problem instances. To avoid scaling problems, the values of the different criteria

[1] http://www.cmap.illinois.gov/travel-tracker-survey.

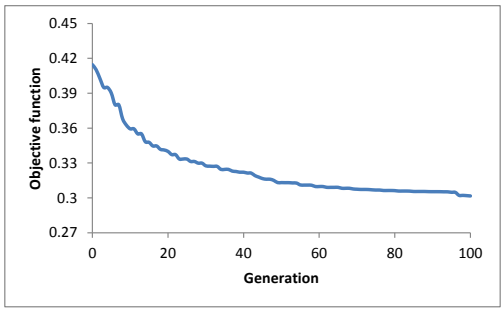


Fig. 5. The multicriteria objective function at different generations

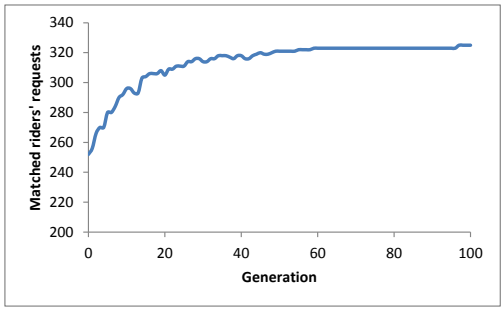


Fig. 6. The number of matched riders' requests at different generations

are normalized before computing the value of the objective function. Note that the direct travel time and distance and the maximum allowed travel time and distance can provide us with upper and lower bounds for the total travel time and distance. The number of the riders' requests gives us an upper bound for the number of matched riders' requests.

Figure 5 shows the value of the objective function at different generations (fitness of the best individual). We see that the proposed algorithm is able to minimize the multicriteria objective function. In the following, we look at each criterion in the objective function through generations.

Figure 6 shows the number of matched riders' requests at different generations. We notice that the general trend is that the number of matched riders' requests increases with generations as this criterion has the highest weight. Figure 7 shows the total distance of the vehicles' trips at different generations. We notice a sawteeth behavior in the curve of

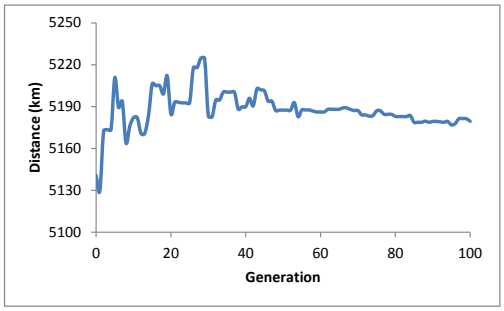


Fig. 7. The total distance of the vehicles' trips at different generations

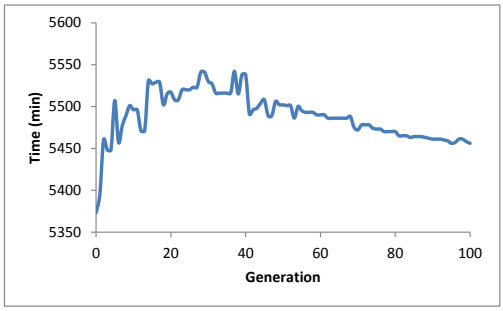


Fig. 8. The total time of the vehicles' trips at different generations

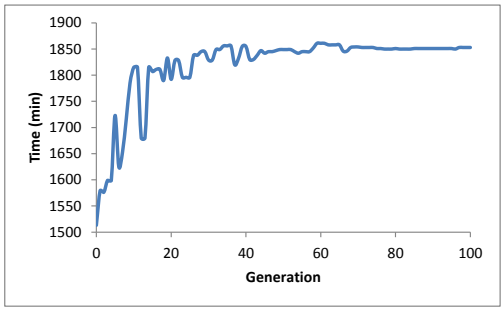


Fig. 9. The total time of the trips of the matched riders' requests at different generations

this criterion which is explained as follows: By the increase of the generations, new riders' requests are matched resulting in increasing the total distance of vehicles' trips. The algorithm then minimizes the distance of the vehicles' trips that is required to serve the matched riders' requests which opens a room for matching new riders' requests. Matching new riders' requests increases the distance of the vehicles' trips again.

Figure 8 shows the total time of the vehicles' trips. This curve is very similar to the curve of the total distance of the vehicles' trips and the sawteeth behavior is explained in the same way. Optimizing the time of the vehicles' trips opens a new room for matching new riders' requests which increases the time of the vehicles' trips again. The time of the vehicle's trip depends on the distance of the trip, service time of the riders' requests (at pickup and delivery points) and potential waiting time at some of the pickup points. While the service time is constant, the waiting time changes for different configurations. This means that it is not necessary that the time of the drivers' trips always increases/decreases with the increase/decrease of their distance. We might increase the length of some route and reduce the waiting time at some of its points. We notice also from Figures 6 - 9 that after near about 40 generations, the algorithm mostly optimizes the criteria other than the number of matched riders' requests as it becomes harder to find new matches.

Figure 9 shows the total time of the trips of the matched riders' requests. The value of this criterion depends on the number of matched riders' requests, travel distance to serve these requests and the waiting time at particular pickup points of the matched requests. Note that for this criterion, the

TABLE I
MEANS AND STANDARD DEVIATIONS OF THE OBJECTIVE FUNCTION AND THE FOUR CRITERIA

| Instance | Objective function | No. matched riders' requests | Vehicles' Travel Distance | Vehicles' Travel Time | Riders' Travel Time |
|---|---|---|---|---|---|
| RM698_L | $0.295_{\sigma=0.0070}$ | $326.067_{\sigma=3.855}$ | $5158.488_{\sigma=21.662}$ | $5438.733_{\sigma=22.044}$ | $1798.033_{\sigma=38.815}$ |
| RM698_R | $0.482_{\sigma=0.0030}$ | $187.867_{\sigma=2.68}$ | $2679.89_{\sigma=13.82}$ | $2853.967_{\sigma=15.096}$ | $1014.733_{\sigma=41.151}$ |

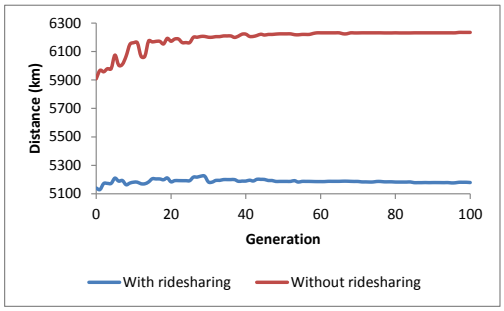


Fig. 10. The total distance of the trips of the vehicles and the matched riders' requests with and without ridesharing

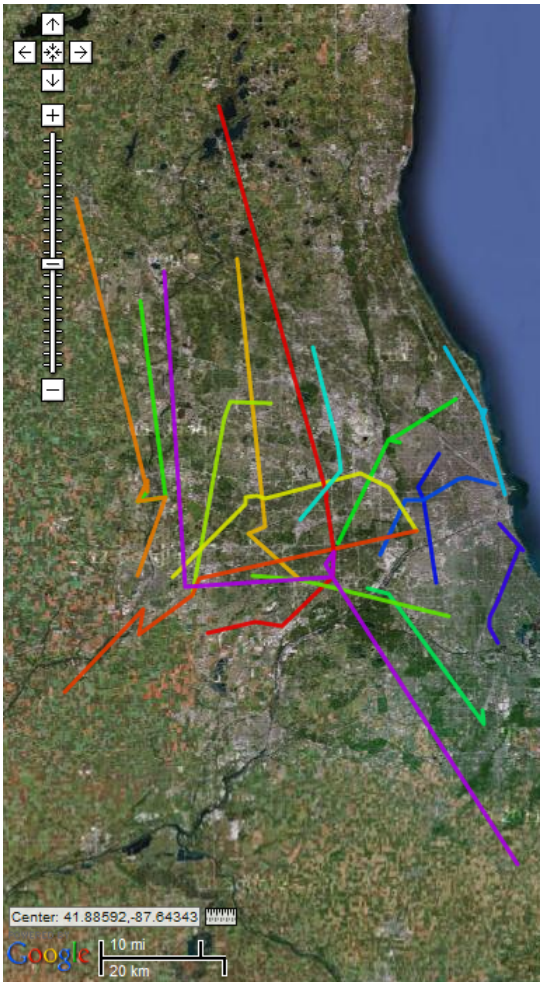


Fig. 11. Visualization of some vehicles' trips. The used GPX visualizer is: http://www.gpsvisualizer.com.

vehicle's waiting time at particular point is multiplied by the number of riders in the vehicle.

Figure 10 shows the total distance of the trips with and without ridesharing. The curve with ridesharing is the same as the curve of the total distance of the vehicles' trips in Figure 7. The curve without ridesharing shows the total distance of the trips of the vehicles and the matched riders' requests as if each one travels alone. The figure shows that the ridesharing could save a significant amount of travel distance. We notice also that the distance saving generally increases with generations (till generation $\approx 40$ in the Figure) as new riders' requests are matched and trips' distances are minimized.

To see how the trips of the vehicles look like, we have created a GPX file for some of the vehicles' trips and used a GPX visualizer to visualize it on Google maps. The visualization result is provided in Figure 11.

Table I shows the means and standard deviations of the results of 30 independent runs on the two problem instances. For the instance RMPTW698_L, the algorithm was able to match 326 riders' requests in average (73% of the riders' requests) with 1.12% of the vehicles' direct travel distance and 1.15% of the vehicles' direct travel time. The algorithm was able to match 187 riders' requests in average (42% of the riders' requests) for the instance RMPTW698_R with 1.09% of the vehicles' direct travel distance and 1.11% of the vehicles' direct travel time. The number of matched riders' requests on the instance RMPTW698_R is near about half of the number of the matched riders' requests on the problem instance RMPTW698_L. This result is very clear as in instance RMPTW698_L, the vehicles travel longer distances and supposed to serve larger number of riders' requests. In average, it takes the algorithm near about 2 minutes of execution.

## VI. CONCLUSION AND OUTLOOK

In this work, we have modeled the ridematching problem with time windows in dynamic ridesharing and provided a genetic algorithm to solve it. In addition, we have engineered datasets for testing the behavior of the proposed algorithm that are extracted from real world data. We conclude that the idea of dynamic ridesharing is feasible and a reasonable number of riders' requests could be matched. We conclude also that the proposed genetic algorithm is able to solve the ridematching problem and to optimize the multicriteria objective function defined for it.

There are many directions for extending this work. One direction is to try different metaheuristics to solve the same problem. Another direction is to extend the algorithm to be dynamic itself and to do the matching on the fly. While we consider the problem of dynamic ridesharing, our approach is still static where we take each time a static snapshot of riders'

requests and drivers' offers to solve the ridematching problem. An interesting direction is to extend the problem definition to solve the ridematching problem with time windows in dynamic multihop ridesharing where riders can share rides with multiple drivers. On the application level, we might move to a more realistic scenario. This implies adding more constraints such as limiting the number of pickup and delivery operations for each driver, putting a maximum detour distance per rider's request and considering realistic travel times and routes using Google APIs.

For the reproducibility of the results, the problem instances and the source code will be available on the authors' website.

## ACKNOWLEDGMENT

This work was partially supported by the German Academic Exchange Service (DAAD), Grant no. A/08/99166. Special thanks for Thomas Geier for his insightful suggestions. The authors would like to thank the anonymous reviewers for their valuable comments.

## REFERENCES

[1] N. Agatz, A. Erera, M. Savelsbergh, and X. Wang, "Sustainable passenger transportation: Dynamic ride-sharing," Erasmus Research Inst. of Management (ERIM), Erasmus Uni., Rotterdam, Tech. Rep., 2010.

[2] N. A. Agatz, A. L. Erera, M. W. Savelsbergh, and X. Wang, "Dynamic ride-sharing: A simulation study in metro atlanta," *Transportation Research Part B: Methodological*, vol. 45, no. 9, pp. 1450 – 1464, 2011.

[3] J. Baugh, G. Kakivaya, and J. Stone, "Intractability of the dial-a-ride problem and a multiobjective solution using simulated annealing," *Engineering Optimization*, vol. 30, no. 2, pp. 91–123, 1998.

[4] N. Chan and S. Shaheen, "Ridesharing in north america: Past, present, and future," *Transportation Research Board Annual Meeting*, 2011.

[5] W. Herbawi and M. Weber, "Ant colony vs. genetic multiobjective route planning in dynamic multi-hop ridesharing," in *Tools with Artificial Intelligence (ICTAI), 2011 IEEE Conference*, 2011.

[6] W. Herbawi and M. Weber, "Comparison of multiobjective evolutionary algorithms for solving the multiobjective route planning in dynamic multi-hop ridesharing," in *Evolutionary Computation (CEC), 2011 IEEE Congress on*, june 2011, pp. 2099 –2106.

[7] J. Jaw, A. Odoni, H. Psaraftis, and N. Wilson, "A heuristic algorithm for the multi-vehicle advance request dial-a-ride problem with time windows," *Transportation Research Part B: Methodological*, vol. 20, no. 3, pp. 243 – 257, 1986.

[8] R. Jorgensen, "Dial-a-ride," Ph.D. dissertation, Technical University of Denmark, 2002.

[9] E. Kamar and E. Horvitz, "Collaboration and shared plans in the open world: studies of ridesharing," in *International Joint Conference on Artificial intelligence (IJCAI), 2009*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2009, pp. 187–194.

[10] J. L. R. Jorgensen and K. Bergvinsdottir, "Solving the dial-a-ride problem using genetic algorithms," *Journal of the Operational Research Society*, vol. 58, p. 13211331, 2007.

[11] R. W. Sinnott, "Virtues of the haversine," *Sky and Telescope, Vol.68:2, P.158, 1984*, vol. 62, no. 2, p. 158, 1984.

[12] M. Solomon, "Algorithms for the vehicle routing and scheduling problems with time window constraints," *Operations Research*, vol. 35, pp. 254–265, 1987.

[13] P. Toth and D. Vigo, Eds., *The vehicle routing problem*. Siam, 2002.