CORA Cork Open Research Archive
Cartlann Taighde Oscailte Chorcaí

Title	Minimizing the driving distance in ride sharing systems
Authors	Armant, Vincent;Brown, Kenneth N.
Publication date	2014-11
Original Citation	Armant, V. and Brown, K. N. (2014) "Minimizing the driving distance in ride sharing systems", IEEE 26th International Conference on Tools with Artificial Intelligence (ICTAI), Limassol, Cyprus, 10-12 November. doi: 10.1109/ICTAI.2014.91
Type of publication	Conference item
Link to publisher's version	http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6984527&isnumber=6983902 - 10.1109/ICTAI.2014.91
Rights	© 2014 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works.
Download date	2026-11-07 12:30:20
Item downloaded from	https://hdl.handle.net/10468/2504



University College Cork, Ireland
Coláiste na hOllscoile Corcaigh

Minimizing the Driving Distance in Ride Sharing Systems

Vincent Armant and Kenneth N. Brown

***Abstract*—**

Reducing the number of cars driving on roads is an important objective for smart sustainable cities, for reducing emissions and improving traffic flow. To assist with this aim, ride-sharing systems match intending drivers with prospective passengers. The matching problem becomes more complex when drivers can pick-up and drop-off several passengers, both drivers and passengers have to travel within a time-window and are willing to switch roles. We present a mixed integer programming model for this switching rider problem, with the objective of minimizing the total distance driven by the population. We exhibit how the potential saving in kilometers increases as the driver flexibility and the density of the distribution of participants increases. Further, we show how breaking symmetries among the switchers improves performance, gaining over an order of magnitude speed up in solving time, and allowing approximately 50% more participants to be handled in the same computation time.

Reducing pollution and carbon emissions while maintaining the population mobility is an important objective for sustainable societies. Particular attention is being paid to reducing the total distance driven by motor vehicles, and to reducing the number of those vehicles on the roads in cities and on main through routes. One approach to achieving this that is becoming more popular is the development of ride-sharing schemes. In such schemes, a driver advertises an intended trip, while prospective passengers make requests for rides in vehicles between specific destinations. The agency then attempts to match passengers to drivers. Each successful match takes one vehicle off the road. Such schemes become more successful as the number of participants increases. In [7] this is described as a chicken-and-egg problem, where both drivers and passengers agree to participapte only if there are enough participants in the other role. [2] investigates an extension of this general problem where a set of shifters can take the role of either driver or passenger. In this context, a ride-sharing solution must specify a clear role for each shifter, and then match riders to drivers.

Since one of the aims is to reduce the total driven distance, the problem is one of optimisation, converting shifters with longer routes into passengers, and allocating rides for as many passengers as possible. The problem of computing an optimal matching to establish a ride-share plan is challenging [3]. It becomes harder as the number of shifters increases. Similarly, allowing drivers to pick up and drop off multiple passengers on the same journey increases the complexity.

Several reformulations of the ride-sharing problem to a combinatorial problem have been proposed. [8] extends the initial-commitment decision problem for tackling the ride-sharing as a collaborative planning problem. The experiments are conducted on a dataset of real-world trips, and show the efficiency in terms of saved miles, cost and time. That work considers a fixed number of pick-ups and drop-offs per driver and includes shifters in the model. In our experiments, we show that removing these limitations increases the efficiency of the system. [9] focusses on the satisfaction of user preferences to enhance the ride sharing user experience. Furthermore, their system is able to trade-off the minimization of vehicle kilometres travelled with the overall probability of successful ride-shares. However, only few dozen users are modelled in their experiments. [3] has developed an optimization based approach aimed at minimizing the total driving distance incurred by users and their individual travel cost. Their system is able to build an efficient ride share plan for a large number of users on short time scales. They also investigate the possibility for users to change role. However, drivers are only allowed to make a single pick-up and a single drop-off. In addition, their experiments, based on a simulation of ride-sharing announcements from metropolitan Atlanta, show good results when the role of each user is fixed. In this case the ride-sharing problem returns to a polynomial bipartite graph matching problem [6]. [5] proposes a multi-hop ride-sharing system where prospective riders can share a ride with several drivers to reach their destination. This approach, based on efficient techniques for shortest path finding, scales with the number of trip announcements. It does not model time windows or the possibility to change role.

In this study, we propose a mixed integer programming formulation of the ride-sharing problem with shifters, where drivers are allowed to multiple pick-ups and drop-offs and all users specify a time window. To tackle the combinatorial issue, we introduce two optimisation techniques to our model: linearization and symmetry breaking. We evaluate our methods on randomly generated problems created from real map data. We show that adding symmetry breaking constraints offers an order of magnitude improvement in runtime over the initial model. Furthermore, from optimising ride sharing, we show benefits for sustainability and we exhibit the impact on the metric of different concentrations of users in space and time.

PROBLEM DESCRIPTION

Our intention is to model ride-sharing for daily commuting scenarios, and the aim is to match riders to drivers, satisfying car capacities and individual trip constraints, while minimizing the total driven distance. Drivers and riders have some flexibility in negotiating specific pick-up and drop-off locations, thus, rather than assume each user has a unique starting point and unique a destination, as is standard in dial-a-ride problems [4], trips are initially arranged based on a smaller set of standard locations, representing small towns, districts or main junctions in the road network. Many users may thus share the same origin or destination. We assume travel times between all pairs of locations are known, and we assume all drivers will use the same route between a given pair. Similarly, we assume time windows are based on standard intervals e.g. 15 minutes, and users will specify their latest arrival time at that granularity. A driver's trip offer is specified as a route from start to finish, a time window of earliest departure and latest arrival, and the number of available seats for passengers. A rider's trip request specifies a start location and destination and a time window. Some drivers (called shifters) are flexible, and are willing to be selected as riders. Shifters specify both a trip offer and a trip request. We assume a driver will definitely drive, regardless of whether or not match is made; a shifter will definitely travel, and thus will drive if not selected as a passenger and no rider is matched; a rider who is not offered a match may drive on their own, outside the system, with a given probability, and otherwise they will take public transport. Each driver that is allocated one or more riders must be given a departure time and an arrival time at each point on the route, such that the time gap between any pair of locations is not less than the known travel times. For a rider to be assigned to a driver, the rider's start and finish locations must be on the drivers route in the right order, and the driver's times must satisfy the rider's time window. Multiple riders can be allocated to the same driver, as long as the number of passengers travelling between any pair of locations on the route does not exceed the available number of seats. If a shifter is assigned as a passenger for another driver, then the shifter does not drive, and no rider can be assigned to the shifter. Similarly, if a rider is assigned as a passenger on the shifter's trip, then the shifter cannot be assigned as a passenger. A driver is served when at least one rider is assigned to the driver's trip; a rider is served when he or she is assigned to a trip. A solution to the ride sharing problem is then an assignment of a clear role to shifters (driver or rider) and a matching of drivers to riders that satisfies the above constraints. An optimal solution is one that minimizes the total driven distance - that is, the sum of the route distance for drivers, for shifters that are selected to be drivers, and for the fixed proportion of unserved riders.

Mixed Integer Programming Formulation

We model the ride-sharing participants as follows:

User notation:

- $D = \{d_i, 1 \leq i \leq m\}$ represents the finite set of possible drivers.
- $R = \{r_j, 0 \leq j \leq n\}$ represents the finite set of possible riders.
- $S = D \cap R$ represents the finite set of shifters (i.e., drivers or riders that are willing to change role).
- $U = D \cup R$ represents the finite set of users.

Location notation:

- $V = \{v_k, 1 \leq k \leq p\}$ represents the set of user locations.
- vs_u is the departure location of $u \in U$.
- ve_u is the arrival location of $u \in U$.
- π_d represents the preferred path (ordered list of locations) of $d \in D$.
- $pred_{\pi_d}(v)$ denotes the predecessor of v in the path π_d.
- $starts_{\pi_d}(v)$ denotes the set of riders that can start from $v \in \pi_d$.
- $ends_{\pi_d}(v)$ denotes the set of riders that can end at $v \in \pi_d$.

Time windows, distance and car capacity notation:

- et_u represents the earliest departure time of u.
- lt_u represents the latest arrival time of u.
- $st_{v,v'}$ denotes the minimum time from v to v'.
- $d_{v,v'}$ denotes the distance corresponding to the minimum time from v to v'.
- q_d represents the car capacity of $d \in D$.

Decision Variables

- $y_{d,r}$ represents a matching between a driver $d \in D$ and a rider $r \in R$, if $y_{d,r} = 1$ d and r share a ride, $y_{d,r} = 0$ otherwise.
- $t_{d,v}$ represents the departure time of the driver d from the location v.

Auxiliary Variables

- x_s represents the role of a shifter $s \in S$ s.t. $x_s = 1$ iff s is a driver, $x_s = 0$ otherwise. The value of x_s entirely depends on the ride sharing variables $y_{s,r}$ and $y_{d,s}$.
- z_r denotes a served rider $r \in R \setminus S$ s.t. $z_r = 1$ iff r shares a ride, $z_r = 0$ otherwise. The value of x_s entirely depends on the ride sharing variables $y_{d,r}, \forall d \in D$.
- $o_{d,v}$ denotes the car occupancy of driver $d \in D$ when leaving the location $v \in V$. It also depends on the ride sharing variables $y_{d,r}$, $\forall d \in D$.

We use the logical operators implication ($\Rightarrow$) and equivalence ($\Leftrightarrow$) to model some constraints. Note that both operators are provided in standard tools - e.g., CPLEX. In the next section, we will discuss how to linearize those constraints.

Our objective is to minimize:

$$\sum_{s\in S} x_s.d_{vs_s,ve_s} + \alpha \sum_{r\in R\setminus S}(1-z_r).d_{vs_r,ve_r} \quad (1)$$

subject to:

$$\sum_{d\in D,d\neq r} y_{d,r} \leq 1, \quad \forall r \in R \quad (2)$$

$$(\sum_{d\in D,s\neq d} y_{d,s} = 1) \Leftrightarrow x_s = 0, \quad \forall s \in S \quad (3)$$

$$(\sum_{r\in R,s\neq r} y_{s,r} \geq 1) \Rightarrow x_s = 1, \quad \forall s \in S \quad (4)$$

$$(\sum_{d\in D,r\neq d} y_{d,r} = 1) \Leftrightarrow z_r = 1, \quad \forall s \in R \setminus S \quad (5)$$

$$o_{d,v} = o_{d,v'} + \Sigma y_{d,r} - \Sigma y_{d,r'}, \quad r\in starts_{\pi_d}(v) \quad r'\in ends_{\pi_d}(v) \quad \forall d \in D, \forall v \in V \quad (6)$$

$$y_{d,r} \Rightarrow et_r \leq t_{d,vs_r}, \quad \forall d \in D, \forall r \in R, d \neq r \quad (7)$$

$$y_{d,r} \Rightarrow t_{d,ve_r} \leq lt_r, \quad \forall d \in D, \forall r \in R, d \neq r \quad (8)$$

$$t_{d,v} + st_{v,v'} \leq t_{d,v'}, \forall d \in D, \forall v = pred_{\pi_d}(v') \quad (9)$$

$$y_{r,d} \in \{0,1\}, o_{d,v} \in [0,q_d], t_{d,v} \in [et_d, lt_d] \quad (10)$$

The aim is to minimize the total driven distance (1). The objective only represents the shifters and the unserved riders, since the pure drivers will contribute the same distance for all feasible solutions. α is the proportion of unserved riders that use their cars. The constraints (2) force each rider to be a passenger of at most one driver. A shifter assigned to one driver as a passenger is a rider, otherwise a driver (3). A shifter that is assigned a passenger, then it must be a driver (4). Note that this is an implication, and so a shifter with no passengers may still drive. A rider is served if he shares a trip with a driver (5). When a driver leaves a location its car occupancy is equal to the difference between picked up and dropped off passengers plus the car occupancy of the previously visited location (6). A driver leaves a passenger's location not before the passenger's earliest departure time (7). A driver visits the passenger's destination before the passenger's latest arrival time (8). The time spent between two consecutive locations on a path is not less than the minimum time to travel between the two locations (9). In the experiments, to reduce the size of the model, we only constrained drivers and rider for which the intersection of time-windows allow feasible rides.

Our first potential improvement consists of linearizing all the constraints in the previous model that contain logical operators. Consider constraints (3). Given two boolean variables or two constraints c_1 and c_2, $c_1 \Leftrightarrow c_2$ can be rewritten as $c_1 = c_2$. constraints (3) can be rewritten as the following formula:

$$\sum_{d\in D,s\neq d} y_{d,s} = 1 - x_s, \forall s \in S \quad (11)$$

Similarly, constraints (5) can be rewritten as:

$$\sum_{d\in D,r\neq d} y_{d,r} = z_r, \forall r \in R \setminus S \quad (12)$$

To linearize the set of constraints (4), we use the fact that given two boolean variables or two constraints c_1 and c_2, $c_1 \Rightarrow c_2$ can be rewritten as $c_1 \leq c_2$. Thus, constraints (4) can be rewritten introducing a reified variable v_s s.t.

$$v_s = \sum_{r\in R,s\neq r} y_{s,r} \geq 1 \quad \forall s \in S. \quad (13)$$

$$v_s \leq x_s, \quad \forall s \in S \quad (14)$$

Without the insertion of the new variable v_s, constraints (4) can also be rewritten using a big integer M greater than any variable upper bound.

$$\sum_{r\in R,s\neq r} y_{s,r} \leq M \times x_s \quad \forall s \in S \quad (15)$$

Using the above concepts, constraints (7) can be rewritten as follows:

$$(y_{d,r} \times et_r) \leq t_{d,vs_r}, \quad \forall d \in D, \forall r \in R, d \neq r \quad (16)$$

Similarly constraints (8) can be replaced by the following:

$$(M - t_{d,ve_r}) \geq (M - lt_r) \times y_{d,r}, \quad \forall d \in D, \forall r \in R, d \neq r \quad (17)$$

In the last constraint, replacing (8) with $(-t_{d,ve_r} \geq -lt_r \times y_{d,r})$ reformulates the implication when $y_{d,r} = 1$ but enforces $t_{d,ve_r} = 0$ when $y_{d,r} = 0$. To let t_{d,ve_r} vary when $y_{d,r} = 0$, we introduce M in both sided of the inequality. In the section Experiments, we compare the efficiency of the basic model and the linearized models for solving ride-sharing problems.

Breaking Symmetries between Possible Passengers

Breaking symmetries is a powerful technique that has been successfully applied for tackling the complexity issue of many challenging problems. A symmetry breaking constraint among a set of variables aims to filter the search space explored by an algorithm. It avoids the exploration of branches that lead to solutions for which it suffices to permute values to obtain a symmetrical assignment. For example, in our ride-sharing setting, consider two shifters s_1 and s_2 having the same trip announcement i.e., same departure, same arrival, same time window. The only case that can be filtered without obstructing the search of the optimal ride-sharing plan is when s_1 or s_2 have a distinct role. Indeed, for the assignment of shifters we have 3 possibilities.

1) $x_{s_1} = 1$, $x_{s_2} = 1$, both shifters are drivers. This case cannot be filtered, since the optimal solution may require s_1 and $s2$ to drive.
2) $x_{s_1} = 0$, $x_{s_2} = 0$ both shifters are riders. Similarly, this case cannot be filtered, since the optimal solution may require s_1 and $s2$ to ride.

3) ($x_{s_1} = 0$ and $x_{s_2} = 1$) or ($x_{s_1} = 1$ and $x_{s_2} = 0$), one of the shifters is a rider and the other one is a driver. Since we assume that drivers follow the same route, each set of passengers picked up by s_1 in a solution where s_1 is driver will be able to be picked-up in a symmetric solution where s_2 is a driver. Similarly, since s_1 and s_2 share the same trip announcement, each driver sharing a ride with s_1, in a solution where $s1$ is a rider, will be able to share his ride with s_2 in a symmetric solution where s_2 is a rider.

More formally, to define the set of symmetry breaking constraints added to the basic ride-sharing model, we introduce the following notation:

Definition 1: Two users u_i and u_j share the same trip announcement $\alpha(u_i) = \alpha(u_j)$ if:

- $vs_{u_i} = vs_{u_j}$, u_i and u_j share the same departure.
- $ve_{u_i} = ve_{u_j}$, u_i and u_j share the same arrival location.
- $et_{u_i} = et_{u_j}$, u_i and u_j have the same earliest departure time.
- $lt_{u_i} = lt_{u_j}$, u_i and u_j have the same latest arrival time.
- if $u_i, u_j \in D^2$ then $\pi_{u_i} = \pi_{u_j}$

To avoid the computation of symmetrical solutions, we consider an ordered set of possible riders (resp. drivers) sharing the same announcement. The ordered sets of riders (resp. drivers) can be seen as a queue of riders where the rider with highest priority in the queue is served first. We denote by $A = \{a_1, ..., a_n\}$ the set of different announcements and $R_a = \{r \in R | a = \alpha(r)\}$ the ordered set of possible riders sharing the announcement a s.t. $R = \bigcup_{a \in A} R_a$. We partition R_a in two ordered sets: S_a, the set of shifters and, PR_a, the set of pure riders. S_a^i (resp. PR_a^i) denotes the i^{th} shifter in S_a (resp. PR_a). Using this notation we can formulate the first set of symmetry breaking constraints between shifter roles sharing the same trip announcement.

$$x_{S_a^i} \leq x_{S_a^{i+1}}, \quad \forall a, \forall i \in [1..|S_a|[\tag{18}$$

These last symmetry breaking constraints avoid the exploration of the search space where the shifter S_a^i drives (i.e., $x_{S_a^i} = 1$) and the shifter S_a^{i+1} is a rider $x_{S_a^{i+1}} = 0$. The remaining possibilities are explored. Similarly, if two (pure) riders $\rho_a^{pr}(i)$ and $\rho_a^{pr}(i)$ share the same trip announcement, we can avoid the exploration of solutions where one of them is served and the other is not.

$$z_{PR_a^i} \geq z_{PR_a^{i+1}}, \quad \forall a, \forall i \in [1..|PR_a|[\tag{19}$$

Note that, the shorter the symmetry breaking constraint, the more efficient is the filtering on the search space. Suppose we want to solve a problem with n variables. Without considering the problem encoding, when the size of the domain of each variable is d, for each non intersecting symmetry constraint, we potentially filter d^{n-2} paths from the maximal search space of size d^n if the symmetric-variables are in the top of the search tree. In practice, the problem encoding, the search heuristic, the bound updates, and the cuts discovery prevent us to explore all the potential search space. Nevertheless, adding a small number of short symmetry breaking constraints to a model can significantly speed up the search. We show in our experiments that this last remark remains valid in the case of ride-sharing problem.

Simulation of Ride-Sharing

To evaluate the efficiency of our approach, we generate sets of random instances of ride-sharing problems. Our instance generator takes as input the main roads parsed from the Open Street Map of Dublin area, a number of locations $|V|$, a number of users $|U|$, a percentage of shifters among the users $|S|/|U|$, and a rush-hour time-window between 6 and 12 am. V represents the l closest towns to Dublin that had a node tag *townhall* in the OSM file. User trips are chosen among these towns. The number of users represents the number of announcements received by the ride-sharing system i.e., driver or shifter trip-offers, rider trip requests. The percentage of shifters represents the proportion of users willing to change role. The rush-hour time window varies from 0 to 5 hours, the latest arrival of 80% of the user announcements are uniformly distributed within the rush-hour time window. The rush-hour time basically represents the arrival time at work in the morning. For each type of user, a generated announcement describes: (i) start and arrival locations, (ii) the earliest departure and the latest arrival at destination, (iii) the fastest path for a potential driver. To generate realistic trip announcements, we select and rank the $|V|$ towns surrounding Dublin having the greatest number of public transport stations tagged in the OSM file. For each announcement, the start location and destination of the trip is randomly chosen. We give higher priority to towns surrounded by the most public transport facilities. Implicitly, we consider that ride sharing is an attractive alternative to public transportation that tackles congestion issues in big agglomeration. Once destinations are chosen, we determine the size of user trip time windows to represent 130% of the minimum time from the user trip departure to the user trip destination. The user trip latest arrival is set up to at a particular time. To set up a user latest arrival, the generator split each hour in uniform intervals, then we calculate the departure time accordingly. For example, for a generated trip announcement from New Bridge to Dublin, the minimum time is 43 mins, we set up the time window size to 56 mins ($43 * 130\%$). If we consider users arrive at destination at any quarter and we set up the latest arrival to 10:15 am for the trip, the user departure is set up to 9:19 am.

Experiments

To solve the problem instances and compare the different MIP formulations, we use CPLEX [1]. More precisely, we parametrize the solver to run a variant of Branch And Cut

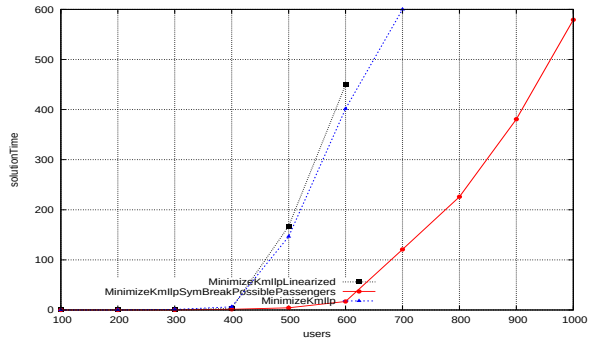
Algorithm with a dynamic search heuristic. In the solving process, CPLEX interleaves the solving of continuous relaxation, the addition of cuts to remove infeasible regions from the search space and the branching on fractional-valued variables which results in two new sub-problems with restrictive bounds on the branching variables. In our experiments, we notice that configuring CPLEX with a dynamic search heuristic gives better performance that the default Branch And Cut algorithm. In this study, we use this solver as a black box for comparing the efficiency of our different models. We run our experiment on a machine with 2 Processors of 2.66Ghz, 12Gb of memory and, 8 cores. Each point drawn in the following figure represents the average over 20 runs. We impose a time limit of 10 minutes. For parameters that are not being varied, we use the following values:

- number of locations: 15
- number of users: 600
- percent of shifters: 80%
- rush-hour time window size: 2 hours
- percent of pure Rider 10%
- percent of pure Driver 10%
- user time window size 1.30% minimum time from departure to destination.

For each set of experiments, we vary one of the first four parameters above.

Scalability and solving time

When evaluating the scalability of the different encodings, we gradually increase the size of the problem. We stop the evaluation of an encoding after 10 successive executions without verifying the optimal solution within the time limit.



cm

Figure 1: Number of users vs CPU time (in sec)

In Figure 1, we plot the solving time of the different models as we increase the number of users. For up to 400 users, the optimal ride-plan is found almost instantaneously. For 500 hundred users, the basic MIP model and the linearized version quickly increase to 150 seconds, and some executions hit the time limit (Figure 2). The symmetry breaking model solves all instances, averaging less than 5 seconds, representing an improvement by a factor of over 30. As the number of users increases further, the first two models repeatedly hit the time limit, while the symmetry breaking model increases more gracefully, requiring less than 240 seconds on average for up to 800 users.

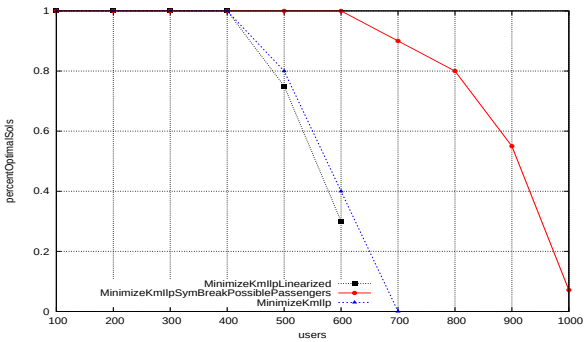


Figure 2: Number of Users vs Percent of Optimal Solutions Found in 10 mins

In Figure 3, we vary the proportion of shifters in the population (size = 600). As that proportion increases to 0.8, the first two models rapidly increase in solving time, and start to hit the time limit. The symmetry breaking model is able to solve to optimality all problems where all users are potential shifters, with an average solving time of less than 200 seconds.

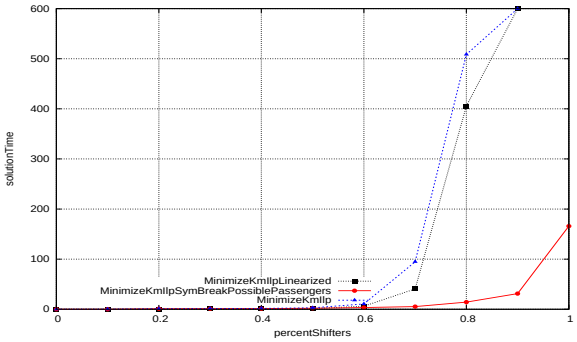


Figure 3: Percent of Shifters vs CPU time (in sec)

In Figures 4 and 5, we measure the impact of two other problem parameters on solving time (for the symmetry breaking model). In 4 we vary the length of the rush hour for a fixed number of users over the half-day. If the rush hour is 5 hours long, then there is essentially no rush hour, and we have a uniform distribution of the number of participants over time; if the window is 1 hour long, then the rush hour is concentrated, and we have a high number of participants all

wanting to travel at the same time. We see that concentrated rush hours significantly increase solving time, due to the increase in the space of possible matches.

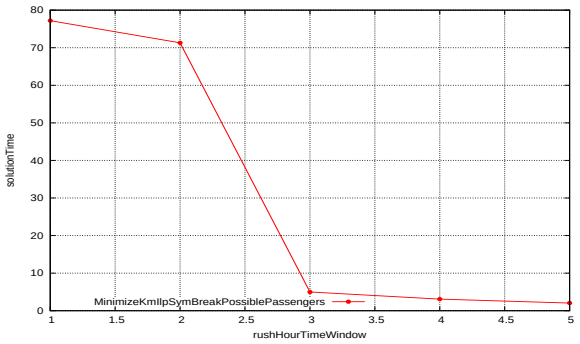


Figure 4: rush-hour vs CPU time (in sec)

In Figure 5, we vary the number of locations, again for a fixed number of users in the population. We start by including the 5 locations closest to Dublin, gradually extending the area at each step. As the number of locations increases (without increasing the number of users), there are fewer options for ride sharing, and so solving time is faster. The spike at 15 locations is an artifact of our underlying data - the locations ranked 11 to 15 are almost all along the same road corridor, and so there is a large increase in the combinatorial options.

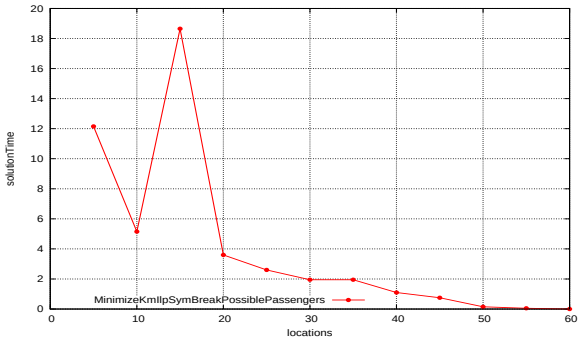


Figure 5: Number of Location Vs CPU time (in sec)

In Figure 6, we analyse the number of constraints generated by an encoding with symmetry breaking when the number of intervals per hour increases for 800 users. We recall that the number of intervals per hour corresponds to the number of possible arrival times in one hour. For example, when the number of intervals per hour is 4, we consider that the latest arrival time of a user can be at any quarter. The number of constraints starts approximately at 60000 for 1 interval per hour and increases linearly with a factor of 1.6. The increasing number of constraints is explained by the increasing number of overlapping time windows of users. It is easy to understand that if the average time size is less than one hour and if user latest arrivals are set up every hour, there will be no ride sharing possibility and then no constraints between users arriving at 9:00 and the user arriving at 10:00. When the time between the different latest arrivals decreases, the possibility of ride sharing between users increases accordingly.

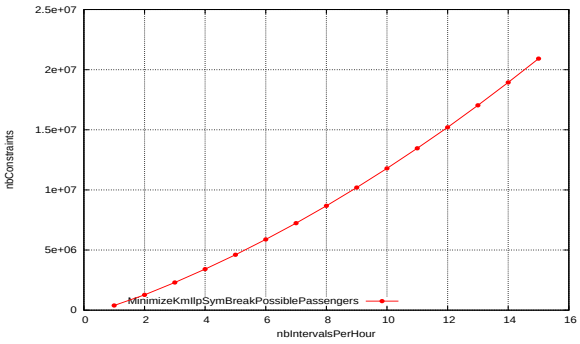


Figure 6: Number of Intervals Per Hour Vs nb of Constraint

In Figure 7 we measure the percentage of symmetry constraints among all the constraints. The percentage of symmetries is extremely low even for a unique possibility of latest arrival per hour (0.01%), and gradually decreases with the number of intervals. Surprisingly, since the percentage of symmetries is very low, but as expected it reaches a peak when the the intervals between two successive arrival times is larger.

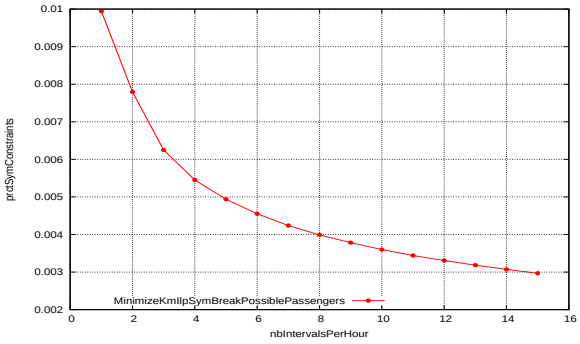


Figure 7: Number of Intervals Per Hour Vs Percentage Symmetries

In Figure 8 we compare the solving time of two models when the number of intervals varies and the instances are solved before the time limit. The model without symmetry breaking shows a peak when the size of the intervals are larger and there exist more similar trips between users. Both of the models perform a fast solving for instances with small intervals per hour.

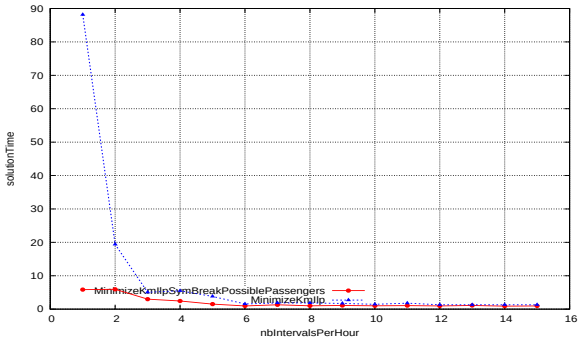


Figure 8: Number of Intervals Per Hour Vs CPU time (in sec)

Sustainability and efficiency of ride sharing

We now measure the quality of the ride sharing solutions, in terms of kilometres saved and the proportion of users being served, as we vary the problem parameters. Note that serving as many users as possible is an important outcome, to encourage continued participation in the ride-sharing scheme. In Figures 9 and 10, we measure the quality

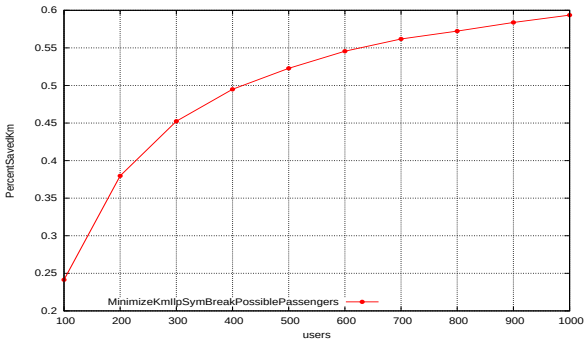


Figure 9: Nb users vs Percent Saved Km

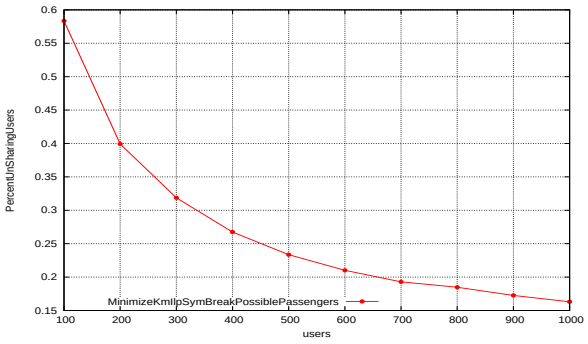


Figure 10: Nb users vs Percent Unserved User

of the Ride-Share plan when varying the number of users. For small population (size 100), we save on average, 25% of the driving distance. This rises steadily, beginning to stabilize around 60% as we increase to a population of 1000. In Figure 10, the proportion of unserved users as the inverse trend - for small populations, fewer than half the participants receive a match, while for the large populations, over 80% of participants receive a match.

In Figures 11 and 12, we vary the number of locations (as for Figure 5). We observe a roughly linear trend - the more heavily concentrated the population is, the more kilometres can be saved and the more participants receive a match.

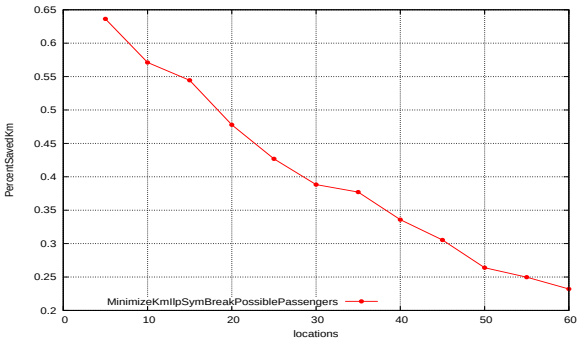


Figure 11: Nb locations vs Percent Saved Km

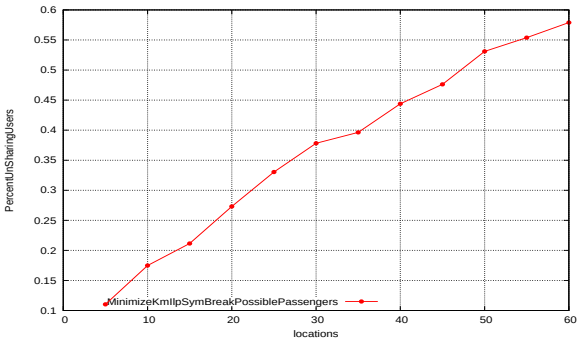


Figure 12: Nb locations vs Percent Unserved User

In Figure 13 , we vary the proportion of shifters. As expected, the more flexibility we have to assign participants to either role, the more kilometres we save and the more participants receive a match.

Finally, in Figures 14 and 15, we again vary the length of the rush hour for a fixed population, and again we observe that a more concentrated rush hour (and thus more participants seeking to travel at the same time) leads to greater savings in driving distance and to higher number of matched participants.

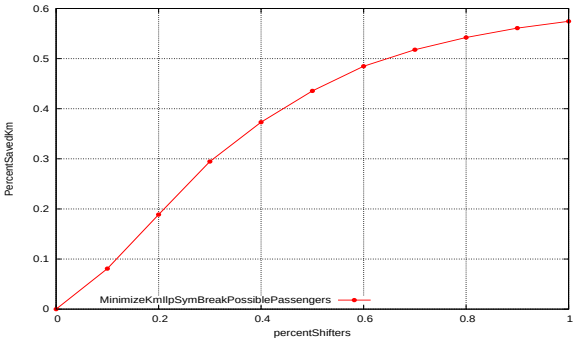


Figure 13: Percent of Shifters vs Percent Saved Km

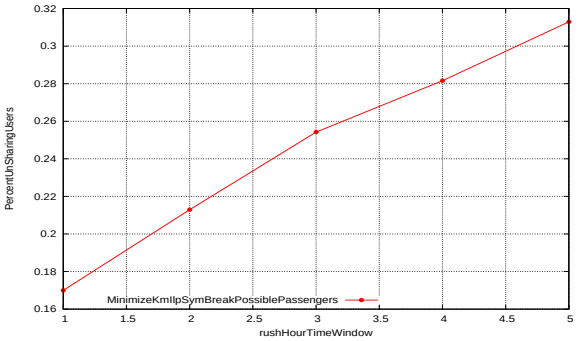


Figure 15: Rush-Hour vs Percent Unserved User

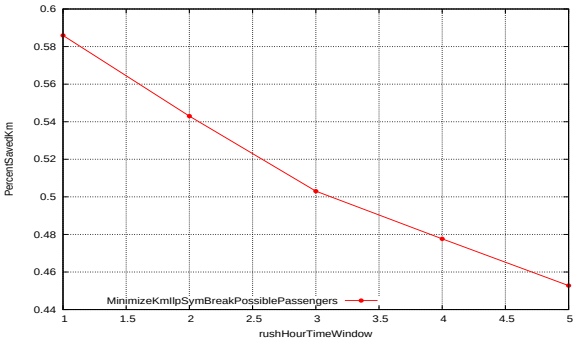


Figure 14: Rush-Hour vs Percent Saved Km

Conclusion and Perspectives

Reducing the total driving distance and the number of cars on the road is an important societal objective. Ride sharing systems are an increasingly popular way of achieving these objectives. The associated ride-sharing problem is one of matching prospective passengers to drivers, to ensure trip constraints are satisfied, while minimizing the total driven distance. The complexity of the problem increases when participants have time windows, and where some drivers are shifters who are willing to change role to become a passenger. We present three mixed integer programming models for this problem of flexible ride sharing systems. The third model uses symmetry breaking on the shifters to reduce the search space. We show that the symmetry breaking model outperforms the other two, achieving an order of magnitude speed up, and allowing 50% more participants to be handled in the same computation time. We also demonstrate the influence of different problem parameters on the societal objective, showing, as expected, that a higher proportion of shifters and a higher concentration of users allows us to save significantly more kilometres. Future work will focus on problems where drivers are willing to change routes, and where participant preferences over possible matches must be included in the objective.

References

[1] Ibm ilog cplex optimization studio v12.6, 2014.

[2] Niels Agatz, Alan Erera, Martin Savelsbergh, and Xing Wang. Optimization for dynamic ride-sharing: A review. *European Journal of Operational Research*, 223(2):295 – 303, 2012.

[3] Niels A.H. Agatz, Alan L. Erera, Martin W.P. Savelsbergh, and Xing Wang. Dynamic ride-sharing: A simulation study in metro atlanta. *Transportation Research Part B: Methodological*, 45(9):1450 – 1464, 2011.

[4] Jean-Franois Cordeau and Gilbert Laporte. The dial-a-ride problem (darp): Variants, modeling issues and algorithms. *Quarterly Journal of the Belgian, French and Italian Operations Research Societies*, 1(2):89–101, 2003.

[5] Florian Drews and Dennis Luxen. Multi-hop ride sharing. In *Proceedings of the Sixth Annual Symposium on Combinatorial Search, SOCS 2013, Leavenworth, Washington, USA, July 11-13, 2013*, 2013.

[6] Jack Edmonds and Ellis L. Johnson. Matching: a well-solved class of integer linear programs. In *in: Combinatorial structures and their applications (eds Gordon and Breach*, pages 89–92, 1970.

[7] Masabumi Furuhata, Maged Dessouky, Fernando Ordóñez, Marc-Etienne Brunet, Xiaoqing Wang, and Sven Koenig. Ridesharing: The state-of-the-art and future directions. *Transportation Research Part B: Methodological*, 57(C):28–46, 2013.

[8] Ece Kamar and Eric Horvitz. Collaboration and shared plans in the open world: Studies of ridesharing. In *Proceedings of the 21st International Jont Conference on Artifical Intelligence*, IJCAI'09, pages 187–194, San Francisco, CA, USA, 2009. Morgan Kaufmann Publishers Inc.

[9] Alexander Kleiner, Bernhard Nebel, and Vittorio Amos Ziparo. A mechanism for dynamic ride sharing based on parallel auctions. In *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence - Volume Volume One*, IJCAI'11, pages 266–272. AAAI Press, 2011.