The Benefits of Meeting Points in Ride-sharing Systems

Mitja Stiglic[1], Niels Agatz[2], Martin Savelsbergh[3] and Mirko Gradisar[1]

[1]Faculty of Economics, University of Ljubljana, Ljubljana, Slovenia

[2]Rotterdam School of Management, Erasmus University Rotterdam, Rotterdam, Netherlands

[3]H. Milton Stewart School of Industrial and Systems Engineering, Georgia Institute of Technology, Atlanta, Georgia, USA

February 19, 2015

Abstract

We investigate the potential benefits of introducing meeting points in a ride-sharing system. With meeting points, riders can be picked up and dropped off either at their origin and destination or at a meeting point that is within a certain distance from their origin or destination. The increased flexibility results in additional feasible matches between drivers and riders, and allows a driver to be matched with multiple riders without increasing the number of stops the driver needs to make. We design and implement an algorithm that optimally matches drivers and riders in large-scale ride-sharing systems with meeting points. We perform an extensive simulation study to assess the benefits of meeting points. The results demonstrate that meeting points can significantly increase the number of matched participants as well as the system-wide driving distance savings in a ride-sharing system.

1 Introduction

In ride-sharing, individuals with matching itineraries and schedules share a ride in a personal vehicle. The driver and rider(s) typically share the associated costs (e.g. fuel, tolls, parking fees) so that each benefits from the shared ride. Additionally, drivers may save time because

they are able to use high-occupancy vehicle lanes reserved for the exclusive use of vehicles with two or more occupants, while riders may appreciate that they do not need to drive or even own a vehicle.

Ride-sharing can significantly reduce the number of cars needed to satisfy the mobility needs of participants and, thus, reduce congestion and other externalities related to heavy traffic when people rely on individual transportation to satisfy their mobility needs. It will, at the same time, also reduce the need for parking space, which is becoming an increasingly scarce and expensive commodity in most urban areas. (Congestion and parking are interrelated as searching for parking space prolongs driving time and can thus contribute to congestion.) Challenges related to high congestion and limited parking space arise in a myriad of urban areas around the world. In the USA, for instance, urban congestion is an acute problem with far-reaching consequences. It is estimated that the cost of extra time and fuel in 498 urban areas in the USA in 2011 alone was roughly $121 billion. Congestion in the USA is expected to grow in the foreseeable future in spite of the planned measures to curb it [Schrank et al., 2012]. In this context, ride-sharing appears as an interesting possibility since it may result in significant effects without large investments.

Ride-sharing services on the market range from simple online bulletin boards to complex systems that can be accessed through web and mobile applications offering automated matching, routing and payment (see Furuhata et al. [2013] for an overview). In this paper, we focus on systems that offer automated matching of drivers and riders within an urban area. An example of a provider offering such a service is Flinc (`https://flinc.org`). The service provider is receiving a large number of ride-share offers and requests from its users. Riders looking for ride-share opportunities need to be matched with drivers that are offering rides and the resulting trips need to be scheduled. Time windows and other restriction imposed by the system or the users need to be respected.

In ride-sharing, each driver has a specific itinerary and is willing to pick-up and drop-off riders en route. To accommodate the riders, the driver has to make a detour and make extra stops. The length of the detour and the number of extra stops depend on the driver's willingness to extend his trip time. This distinguishes genuine ride-sharing from services in which the drivers act as de facto taxicab drivers, e.g., Uber (`https://www.uber.com`). The

level of service in such systems may be higher due to the flexibility of the drivers, but this comes at a higher cost to the rider compared to genuine ride-sharing. With the exception of shared taxi services, such services also do not necessarily reduce congestion.

Limited flexibility in drivers' itineraries and schedules is a major challenge in ride-sharing. It may result in many drivers and riders not finding a match. In the simulations performed by Agatz et al. [2011], approximately 15 to 40 percent of riders and drivers remained unmatched (depending on the setting of the simulation). The simulations also showed that the ratio of matched participants predominantly depends on the distribution density of announced trips in space and time. Settings with very low density (e.g., recently launched ride-sharing services, off-peak hours, rural areas) suffer from the so-called chicken-and-egg problem [Furuhata et al., 2013], where demand for trips is not sufficient to attract sufficient supply and vice-versa. Such a situation may lead to stagnation or implosion in the number of users. To overcome such a situation the ride-sharing system has to be designed well and must employ an effective matching algorithm, so as to ensure that the largest possible number of participants is matched and the system has satisfied users. Only users that have been successfully matched and have had a positive experience can be expected to continue to use the service and promote the ride-sharing service to others. Thus, a high matching rate is a critical success factor for a ride-sharing service.

That being said, ride-sharing systems also have to minimize the effort and inconvenience for the participants. One way to achieve this is to restrict the number of riders per trip to at most one rider. In a single rider match, at most one pickup and drop off take place during a driver's trip. This minimizes the inconvenience of the driver and also makes it easy to divide the trip costs between rider and driver.

In this paper, we investigate benefits of introducing meeting points to take advantage of any flexibility on the part of the riders. Meeting points allow the construction of routes with smaller detours, while maintaining a satisfactory level of service for the riders. Riders may be picked up and dropped off at meeting points that are within an acceptable distance from their origin or destination. (A pick up or drop off can, of course, still take place at the rider's origin and destination as well.) By exploiting the rider flexibility, more matches may be found. Furthermore, meeting points allow a driver to be matched with multiple riders

without increasing the number of stops on the driver's trip.

Consider the example depicted in Figure 1 with driver $d1$ and rider $r1$ and two meeting points $m1$ and $m2$, where the number above an arc represents the time it takes to travel between the nodes, and where the driver is willing to accept an increase in trip time of at most five minutes. Without the use of meeting points, a match between $d1$ and $r1$ is not

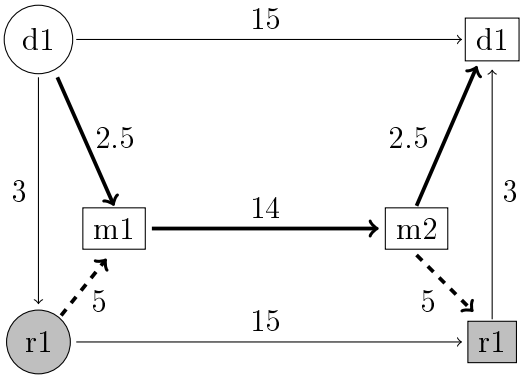

Figure 1: Rider (grey) and Driver (white) traveling from Origin (circle) to Destination (square) via Meeting Points

feasible because the required increase in trip time (6 min) exceeds the driver's limit. If, however, the rider is willing to walk 5 minutes to and from a meeting point, a feasible match between $d1$ and $r1$ is possible, because $d1$ has to make a smaller detour. (The rider's trip will be 9 minutes longer than if he drove by himself, but he will loose no time finding a parking space and he will not be using his own car.)

Note that the savings in driving distance in the example above is about 37% (where the savings in driving distance is obtained by comparing the driving distance when both participants drive by themselves to the driving distance when they are matched, i.e., 30 versus 19 in the example above). It is customary to consider a match distance feasible if there is a positive driving distance savings.

Meeting points can also result in more matches because they allow a driver to be matched with multiple riders without extra stops. Consider the example depicted in Figure 2 with driver $d1$ and riders $r1$ and $r2$ and two meeting points $m1$ and $m2$, where the number above an arc represents the distance between the nodes. (As before, the dashed lines represent walking of riders to and from meeting points.) None of the matches between $d1$ and $r1$ and $d1$ and $r2$ (with or without a pickup at $m1$ and/or a drop-off at $m2$) leads to positive savings

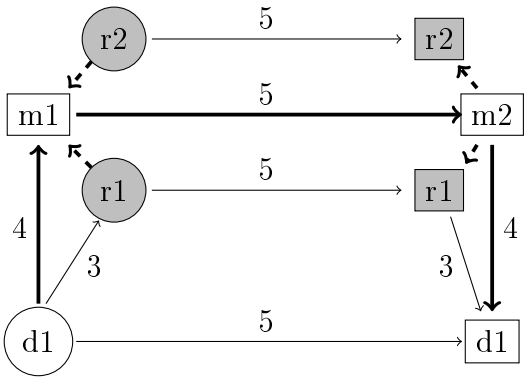

Figure 2: Riders (grey) and Driver (white) traveling from Origin (circle) to Destination (square) via Meeting Points

in driving distance. However, a multi-rider match between driver $d1$ and riders $r1$ and $r2$ (with a pickup at $m1$ and a drop-off at $m2$) does lead to positive driving distance savings (15 versus 13).

In the setting we consider in this paper, a driver can be matched with multiple riders, as long as the capacity of his vehicle is not exceeded, and the riders are picked up at the same meeting point at the same time and dropped off at the same meeting point (at the same time). Allowing only one pickup and one drop-off point per shared ride ensures that the trips are easy to execute and minimize the inconvenience for the driver; additional stops and detours increase the inconvenience for participants and the risk of complications arising during execution. Multi-rider matches may have other, harder to quantify, benefits: waiting for a ride and sharing a ride as a group may increase the feeling of safety and social cohesion and might thereby improve the image of ride-sharing.

Meeting points are an integral component of some existing ride-sharing systems, e.g., slugging or casual carpooling, where passengers form (slug) lines at specific locations and wait for rides (the incentive to pick up riders is typically that it allows drivers to use faster HOV lanes and/or share the cost of tolls), and long-distance ride-sharing, which tends to be scheduled in advance and has less restrictive requirements regarding meeting place and time. The locations that can be used for meeting points varies by region or country.

For instance, in Slovenia, bus stops and gas stations are commonly used as meeting points. However, the use of bus stops may be perceived as unsafe and inappropriate in many parts of the US (or may even be illegal). It is conceivable that fast food restaurants

or coffee shops can act as meeting points in the US, because they are frequented regularly by commuters. Park & ride facilities and entrances to well-known institutions/buildings are additional options.

In this paper, we discuss the design and implementation of an algorithm that optimally matches drivers and riders (based on an extension of the traditional bipartite matching formulation) in large-scale ride-sharing systems with meeting points. We perform an extensive simulation study (based on real-world traffic patterns) to assess the benefits of meeting points. The results demonstrate that meeting points can significantly increase the number of matched participants as well as the system-wide driving distance savings.

The paper is organized as follows. In Section 2, we provide an overview of related literature and explain how we build upon it. In Section 3, we introduce notation and a mathematical model of the ride-share optimization problem with meeting points. In Section 4, we detail the solution approach we have developed for this optimization problem. In Section 5, we motivate and discuss the simulation study we have conducted and we present and analyze its results. Finally, in Section 6, we summarize the key findings and suggest directions for future research.

2 Literature

Ride-sharing is receiving more and more attention from the transportation optimization community. Agatz et al. [2012] and Furuhata et al. [2013] provide an overview and different classifications of the various types of ride-sharing systems encountered in practice. Important dimensions include the dynamics of the system and the number of riders and drivers that are involved in a ride-share match. The advance of internet-enabled mobile technology makes it possible to consider more dynamic ride-sharing systems in which riders and drivers announce non-recurring trips on short notice [Agatz et al., 2011, Amey, 2011].

Agatz et al. [2011] represent the single rider, single driver ride-share matching problem by a max-weight bipartite matching problem. They explore different approaches to match drivers and riders in real-time and investigate the impact of different service characteristics of the system. Their study shows that the success of a ride-sharing system strongly depends

on the participation density, e.g., the number of participants per square mile, and that a minimum participation density is required to ensure a stable system (in which participants do not leave the system because they repeatedly fail to find a match). Wang et al. [2014] extend this analysis by investigating the trade-off between matchings that are optimal for the system as a whole and matchings that are optimal for each of the participants in the system. They introduce the concept of stable matches in the ride-sharing setting. Lee and Savelsbergh [2014] consider the employment of a small number of dedicated drivers to serve riders that would otherwise remain unmatched. The aim is to guarantee a certain service level (i.e., fraction of riders that is matched) thereby ensuring a stable system.

Another way to increase the number of riders that find a match is to allow riders to transfer between different drivers, i.e., allowing a rider to travel with more than one driver to reach his destination [Agatz et al., 2012]. Herbawi and Weber [2011] consider a multi-hop ride-sharing problem in which drivers do not deviate from their routes and time schedules. As such, the drivers' ride-share offers form the transportation network for the rider, who has to find a route that minimizes costs, time, and number of transfers. Drews and Luxen [2013] extend this work by also allowing reasonable detours and time deviations for the drivers. While rider transfers might be acceptable to a driver, they are inconvenient for a rider as they may involve waiting times between rides and they increase the risk of anything going wrong during execution.

In contrast to the existing work, we explicitly consider a setting in which riders are willing to walk to and from meeting points to facilitate easy pick up and drop off. We are aware of only one paper that considers meeting points in a related context. Kaan and Olinick [2013] consider vanpooling, in which up to 15 people share a van to travel to a common location. The commuters in the vanpool drive to a park-and-ride location and then ride together to a final location. The authors consider the problem of assigning commuters and vans to park-and-ride locations, and present a mixed integer programming formulation and several heuristics for its solution. The main difference with our setting is that, in the end, the vans provide scheduled transportation. The setting is also simpler in that all riders travel to one common final location.

The use of pickup locations is not unique to the ride-share setting. It is prevalent in

the school bus transportation in which students in urban areas are assumed to walk to a bus stop from their homes to take the bus to school. The selection of bus stops and the assignment of students to bus stops is a subproblem in the school bus routing problem that is related to our work. While several papers address the school bus routing problem, only few papers explicitly consider the selection of bus stops [Park and Kim, 2010]. Some recent papers have integrated the selection of bus stops with the bus route generation using both exact [Riera-Ledesma and Salazar-González, 2013] and heuristic methods [Schittekat et al., 2013].

Ride-sharing, especially when incorporating meeting points, requires the coordination of rider and driver itineraries. This is related to the area of routing problems with synchronization constraints. This line of research deals primarily with vehicle routing problems in which more than one vehicle may be required to fulfill certain tasks. For a recent review, see Drexl [2012]. In general, vehicle routing problems with synchronization constraints are difficult to solve so heuristics are most commonly used, see for example Goel and Meisel [2013] and Meisel and Kopfer [2014].

The ride-sharing setting we consider has a simple routing structure, because we allow the drivers to make only one pickup and one drop-off. The number of feasible driver-rider matches is also relatively small due to capacity and time constraints. As a consequence, we can enumerate the feasible routes and represent the problem of optimally routing drivers as a matching problem. This allows us to use an exact approach to solve even large instances of the ride-share problem to optimality.

3 Problem Definition

We are provided with a set of trip announcements S. With each trip announcement $s \in S$ are associated, an origin location o_s and a destination location d_s as well as an earliest departure time e_s and a latest arrival time l_s. We assume the departure times of participants are somewhat flexible so that the difference $l_s - e_s$ is greater than the travel time from origin to destination. The set of announcements S can be partitioned into $D \subset S$, the set of trip announcements by the drivers, and $R \subset S$, the set of trip announcements by the riders.

Each driver $i \in D$ also specifies a maximum trip duration T_i, which implies the extra time the driver has available to accommodate a ride-share, and a vehicle capacity C_i, which gives the maximum number of people the driver's vehicle can accommodate. Each rider $j \in R$ also specifies a maximum distance d_j^{max} that he is willing to walk to and from a meeting point. (For presentational convenience, we will sometimes also use o_i and d_i to indicate the origin and destination of a driver i and o_j and d_j to indicate the origin and destination of a rider j.)

We denote the distance from location i to j with d_{ij} and the travel time between the two locations by t_{ij}. Furthermore, we denote the set of meeting point locations that can be reached by at least one rider by M. The set of feasible pickup meeting points for rider j is $M_j^p := \{k \in M \mid d_{ko_j} \leq m_j\}$, and the set of feasible drop-off meeting points for rider j is $M_j^d := \{k \in M \mid d_{kd_j} \leq m_j\}$. We introduce the concept of a *meeting point arc* a to denote a combination of a pickup point and a drop-off point. The set of feasible meeting point arcs for rider j is $A_j := \{(k,l) \mid k \in o_j \cup M_j^p,\ l \in d_j \cup M_j^d\}$. Thus, each rider j can be picked up at his origin o_j or a meeting point in M_j^p and dropped off at his destination d_j or a meeting point in M_j^d. Let $A = \bigcup_{j \in R} A_j$. Finally, we denote the service time at each meeting point $m \in M$ by τ_m, i.e., the time needed to get into and out of the vehicle at a pick up or drop-off meeting point.

3.1 Definition of a Feasible Match

A match is defined as a combination of driver $i \in D$, a set of riders $J \subset R$, and a meeting point arc $a \in A$. Hence, it can be defined by a triplet (i, J, a). Note that since we do not allow more than one pick-up and one drop-off in a match, in a feasible match (i, J, a), we must have $a \in \bigcap_{j \in J} A_j$. Furthermore, a feasible match implies a unique route for the driver and for every rider. A feasible match (i, J, a) must also have $|J| + 1 \leqslant C_i$ and must satisfy the time constraints of the participants. A match is time feasible if it is possible for all participants to traverse the meeting point arc a at the same time, while respecting the earliest departure times from their origins and the latest arrival times at their destinations and, for the driver, the maximum ride time.

In order to check the time feasibility of a match (i, J, a), with $a = (k, l)$, we construct an

implied time window at k for each participant in the match. We denote the implied time window for a participant p (either i or $j \in J$) at k by $[e_p^k, l_p^k]$, where $e_p^k = e_p + t_{o_p k}$ and $l_p^k = l_p - (\tau_k + t_{kl} + \tau_l + t_{ld_p})$. To check the time feasibility of the match, the intersection of the implied time windows has to be non-empty, which implies that we must have

$$\max\left(\max_{j\in J} e_j^k, e_i^k\right) \leq \min\left(\min_{j\in J} l_j^k, l_i^k\right) \tag{1}$$

When the above inequality holds, $\max(\max_{j\in J} e_j^k, e_i^k)$ is the earliest time, and $\min(\min_{j\in J} l_j^k, l_i^k)$ is the latest time, at which the shared ride can depart from meeting point k. The maximum ride time for the driver is satisfied, if

$$t_{o_i k} + \tau_k + t_{kl} + \tau_l + t_{ld_i} \leq T_i. \tag{2}$$

A match between driver i and riders in J on meeting point arc $a = (k, l)$ has an associated driving distance savings of $\sigma_{(i,J,a)}$:

$$\sigma_{(i,J,(k,l))} = d_{o_i d_i} - \left(d_{o_i k} + d_{kl} + d_{ld_i}\right) + \sum_{j\in J} \left(d_{o_j d_j} - (d_{o_j k} + d_{ld_j})\right). \tag{3}$$

A match (i, J, a) is considered distance feasible when $\sigma_{(i,J,a)} > 0$. Note the walking distances are taken into account in (3) to break ties when two or more arcs have similar savings.

3.2 Matching Problem

The single rider, single driver ride-share matching problem can naturally be formulated as a maximum weight bipartite matching problem (Agatz et al. [2011]). We extend this formulation to the single driver, multiple rider ride-share matching problem. We note that the formulation introduced below can represent a variety of ride-share matching problems in which a driver can be matched with multiple riders, because the identification of feasible matches and the associated routing is handled in a subproblem. By accommodating single driver, multiple riders matches the formulation becomes a maximum weight bipartite matching problem with side constraints, which, in theory, is no longer solvable in polynomial time,

but still solves extremely fast in practice. We note too that maximizing system-wide driving distance savings does not guarantee that a maximum number of participants is matched. Consider, for example, the situation depicted in Figure 3 with drivers $d1$ and $d2$ and riders $r1$ and $r2$ and two meeting points $m1$ and $m2$, where the number above an arc represents the distance between the nodes.

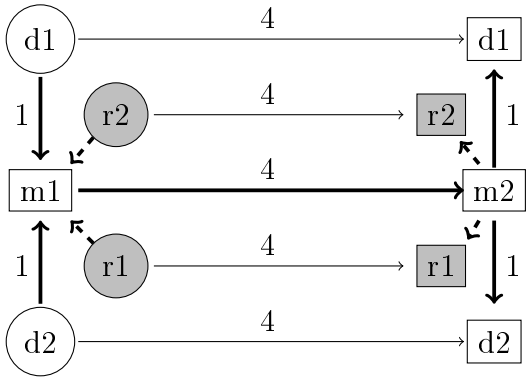

Figure 3: Riders (grey) and Drivers (white) traveling from Origin (circle) to Destination (square) via Meeting Points

The maximum driving distance savings is achieved when either $d1$ or $d2$ is matched with both $r1$ and $r2$ (and, thus, one of the drivers will not be matched). By matching driver $d1$ with rider 1 and driver $d2$ with rider $r2$, all system participants are matched, but with lower driving distance savings (6 vs 4).

As in Agatz et al. [2011], we create a node for each driver $i \in D$ and each rider $j \in R$ and an edge connecting node i and j if there is a feasible match between driver i and rider j. In addition, we introduce nodes that represent a set of riders J, e.g., a pair of riders, a triple of riders, etc., and introduce an edge connecting driver $i \in D$ and set of riders J, if there is a feasible match between driver i and the set of riders in J. Each edge e has two weights associated with it: number of participants in the match ν_e, and maximum driving distance savings σ_e. Note that a particular combination of a driver and a set of riders may have more than one feasible match because there may exist more than one feasible meeting point arc. However, we are clearly only interested in the one with the highest driving distance savings.

Let E represent the set of all edges in the bipartite graph and let the binary decision variable x_e for edge $e \in E$ indicate whether the edge is in an optimal matching ($x_e = 1$) or not ($x_e = 0$). Furthermore, let E_i and E_j represent the set of edges in E associated

with driver i and rider j, respectively. Then, the single driver, multiple riders ride-share matching problem with the objective of maximizing the number of matched participants can be formulated as the following integer program:

$$\max z_1 = \sum_{e \in E} \nu_e x_e \tag{4}$$

subject to

$$\sum_{e \in E_i} x_e \leq 1 \quad \forall i \in D, \tag{5}$$

$$\sum_{e \in E_j} x_e \leq 1 \quad \forall j \in R, \tag{6}$$

$$x_e \in \{0, 1\} \quad \forall e \in E. \tag{7}$$

Objective function (4) maximizes the number of matched participants. Constraints (5) and (6) assure that each driver and each rider is only included in at most one match in an optimal matching, respectively.

To obtain a matching that maximizes the driving distance savings, the objective should be replaced by

$$\max z_2 = \sum_{e \in E} \sigma_e x_e. \tag{8}$$

Since both objectives, i.e., maximizing the number of matches and maximizing the driving distance savings, are relevant in the ride-sharing context, we take both objectives into account in a hierarchical fashion, where we consider z_1 as the primary objective and z_2 as the secondary objective. We first solve (4) subject to (5) - (7). Let z_1^* be the number of matched participants. We then solve (8) subject to (5) - (7) plus the additional constraint $\sum_{e \in E} \nu_e x_e \geq z_1^*$.

Finally, we observe that it is possible to extend the model with a set of participants with flexible roles F similar to Agatz et al. [2011]. The nodes corresponding to flexible participants may appear on either side of the bipartition, but can never be connected with an edge. The

model can be extended by introducing sets E_f representing edges in E associated with flexible participants and adding another set of constraints $\sum_{e \in E_f} x_e \leq 1, \forall f \in F$.

4 Solution Approach

When the number of participants and of meeting points is large, it can become computationally prohibitive to determine the time and cost feasible single matches (especially since multi-rider matches have to be considered as well). Therefore, we have implemented this component of the solution approach carefully and efficiently. For expository purposes, we assume that the locations are in a Euclidean plane, that distances are Euclidean, and that traveling (either walking or driving) occurs at a constant speed. However, most of these assumptions can relatively easily be relaxed so as to cover more realistic settings.

4.1 Determining Feasible Meeting Points for a Rider

We store the set of meeting points in a $k-d$ tree (Bentley [1990]). $K-d$ trees support Euclidean distance nearest neighbor search, n nearest neighbors search, and fixed-radius near neighbor search in logarithmic time. We use the $k-d$ tree to efficiently find, for each rider j, the meeting points within a radius d_j^{max} from the rider's origin o_j and the rider's destination d_j.

4.2 Determining Time and Cost Feasible Matches

Our approach for determining time and cost feasible matches critically depends on the following observation.

Observation 1. *A match between a driver i and a set of riders $J \subseteq R$ with $|J| \geq 2$ is time feasible if and only if the match between driver i and subset of riders $J' \subseteq J$ is time feasible for all $J' \subseteq J$.*

Hence, for a match of one driver and two riders to be time feasible, the match of the driver with each of these two riders must be time feasible as well. Similarly, for a match

of one driver and three riders to be time feasible, the match of the driver with each of the possible pairs of riders must be time feasible as well. And so forth.

It is not necessarily the case that in a distance feasible match between one driver and two riders, the matches between the driver and the individual riders are distance feasible as well (recall Figure 2). In fact, one of the major benefits of meeting points is that this does not have to be the case.

4.2.1 A Basic Algorithm

The basic algorithm considers drivers one by one and finds all time and cost feasible matches for that driver. A straightforward enumeration algorithm with run time complexity $O(nmk)$ finds all feasible single-rider matches, where n is the number of drivers, m is the number of riders, and k is the average number of feasible meeting point arcs per rider.

It follows from Observation 1 that only riders that could feasibly be matched with a driver have to be considered when constructing matches with two riders for that driver. Furthermore, we only need to start from pairs of riders that could feasibly be matched with a driver when constructing matches with three riders for that driver, and so forth. This realization is important because, typically, a driver can be feasibly matched with only a small fraction of the riders. Furthermore, not all pairs of feasible single-rider matches result in feasible two-rider matches, etc. This greatly reduces the number of combinations of riders that have to be considered when determining all feasible matches for a driver.

We thus construct feasible matches for a driver i recursively. We first find all feasible matches involving only one rider, then find all matches involving two riders, etc., up to the available capacity C_i. If a time and distance feasible match is found, an edge e is added to the ride-share matching problem with associated coefficients σ_e and ν_e.

4.2.2 A Refined Algorithm

The basic algorithm presented in the last section is asymptotically optimal, because all matches can theoretical be feasible. However, significantly better practical performance can be achieved with a refined algorithm that exploits the structure of the problem and the

characteristics of an instance. The refined algorithm uses the same recursive approach, but introduces a number of refinements aimed at reducing the number of driver, rider pairs that are (fully) evaluated. We provide an outline of the algorithm in Appendix A.

Rider time windows are stored in a memory structure, which allows us to find the riders with time windows that overlap with the time window of a driver in sub-linear time. We have further enhanced this refinement by developing a method that reduces the size of the time windows stored in the memory structure by considering the minimum required overlap in the time window of a rider and a driver (related to the time a driver and a rider will spend travelling together in a feasible match). The details are given in Appendix B.

Next, we use the locations of the origin and destination and the time window of a driver and a rider to recognize that there cannot be a feasible match without considering meeting point arcs explicitly. The idea is similar to the logic employed to determine the feasibility of a match (i, J, a), but rather than using a meeting point arc, we calculate an implied time window considering origin and destination information only - not the actual pickup and drop-off points. The implied time window for the driver is calculated assuming that the rider is picked up and dropped off on the boundary of his walking range, i.e., on the two circles around o_j and d_j in Figure 4.

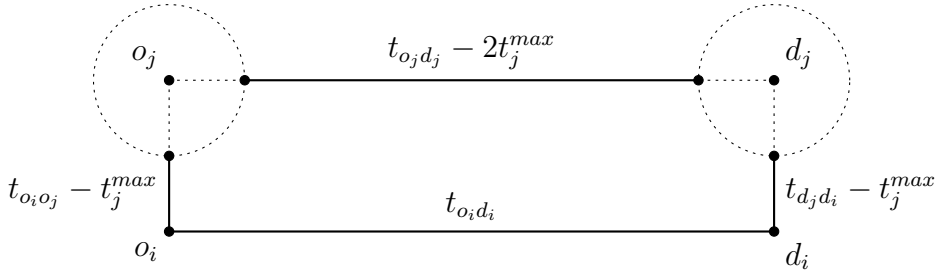

Figure 4: Detecting infeasibility of a match between driver i and rider j without considering meeting point arcs

Furthermore, we assume that the rider travels to the boundary of his walking range at driving speed. If there is no feasible match under that assumption, then there is no feasible match when the rider is walking.

Let t_j^{max} denote the time needed to drive distance d_j^{max}, which is the longest distance a rider is willing to walk to and from a meeting point. Driver i cannot pick up rider j before

$e'_i = e_i + (t_{o_i o_j} - t_j^{max})$ and he cannot pick up rider j after $l'_i = l_i - t_{o_j d_j} - t_{d_i d_j} + 3t_j^{max}$ if he wishes to arrive to d_i in time. We can assume rider j cannot be picked up before $e'_j = e_j + t_j^{max}$ and after $l'_j = l_j - t_{o_j d_j} + t_j^{max}$. If $\max(e'_j, e'_i) > \min(l'_j, l'_i)$, then there cannot be a feasible match between driver i and rider j. From Figure 4, it is also clear that there cannot be a feasible match between driver i and rider j if $(t_{o_i o_j} - t_j^{max}) + (t_{o_j d_j} - 2t_j^{max}) + (t_{d_j d_i} - t_j^{max}) > T_i$.

Only when the two checks above indicate that there may be a feasible match between a driver i and a rider j, we examine the matches of driver i and rider j for each meeting point arc (k, l) where $k \in M_j^p$ and $l \in M_j^d$. If a time and distance feasible match is found, an edge e is added to the ride-share matching problem with associated coefficients σ_e and ν_e.

The last refinement is based on the following observation.

Observation 2. *A match between a driver and a set of riders can only be feasible if the driver and the riders have at least one meeting point arc in common.*

In the basic algorithm, we store, for each feasible match of k riders, all time feasible meeting point arcs, i.e., not only the time feasible meeting point arc that resulted in the maximum driving distance savings. These meeting points arcs are used to construct matches with $k + 1$ riders. Observation 2 shows that only meeting point arcs that are time feasible for at least $k + 1$ riders are relevant. Hence, to construct matches with $k + 1$ riders, we iterate over the meeting point arcs with feasible matches involving k riders, rather than over the feasible matches with k riders, and construct all feasible matches with $k + 1$ riders on a meeting point arc, using the riders that are part of k-rider matches on that particular meeting point arc.

5 A Computational Study

In this section, we report the results of an extensive computational study conducted to assess the benefits of the introduction of meeting points in different ride-sharing environments.

5.1 Generation of Ride-share Data Sets

Similar to Agatz (2011), we use the travel demand model for the metropolitan Atlanta region, developed by the Atlanta Regional Commission, as the basis for generating daily vehicle trips between different travel analysis zones (TAZs) within the region. For a subset of TAZs within the city of Atlanta, we generate five random streams of trips as follows. Each TAZ is a possible origin and a possible destination for a trip. For each origin-destination pair, we calculate an expected number of daily trip announcements by multiplying the average number of single-occupancy home-based work vehicle trips with a fixed percentage of vehicle-trips that we assume might consider participating in a ride-sharing system. Then for each pair, we determine the number of actual trip announcements using a Poisson random variable with expected value equal to the computed expected number of trips. For each trip announcement, we generate the origin and destination points within a fixed radius of 1.1 mile around the center of the travel analysis zone based on a uniform distribution. Each trip announcement is equally likely to be a rider announcement or a driver announcement. The minimum distance of a ride-share trip is 4 miles, i.e. we only consider trips between origins and destinations that are at least 4 miles apart. For each TAZ, we also randomly generate 4 meeting points around its center within a fixed radius of 1.1 mile.

Trip timing information is not available from the travel demand model. Therefore, we create the time windows for each announcement as follows. For each trip, we draw the earliest departure time from a normal distribution with mean 7:30 a.m. and standard deviation of half an hour to model a typical travel peak and calculate the earliest arrival time by adding the direct travel time to the earliest departure time. Subsequently, we calculate the latest departure (arrival) time by adding fixed time flexibility to the earliest departure (arrival) time. We assume the fixed time flexibility to be 30 minutes for all participants. The difference between the latest arrival time and earliest departure time is hence equal to the sum of the direct travel time from origin to destination and the fixed time flexibility. (Note that this means that we are investigating a morning commute.)

The travel distances between all points are computed using the haversine formula with a 30% uplift. To compute travel times, we assume a driving speed of 15 miles per hour (we are

considering an urban area). For each driver $i \in D$, we define a limit on the total duration of his trip $T_i = t_{o_i,d_i} + min(4 + c_{flex} \cdot t_{o_i,d_i}, 20)$. Coefficient c_{flex} is the driver flexibility parameter - our assumption about how the willingness of the drivers to make detours depends on their original trip duration (fixed at 0.25 in base case - see below). The maximum trip duration for driver i is thus defined as his original trip duration plus an additional time that positively depends on his original trip duration. We assume that each driver that wishes to participate in ridesharing is always ready to extend his trip by at least 4 minutes, which is the time associated with one pick-up and one drop-off operation. We also assume that drivers are not willing to extend their original trips by more than 20 minutes, irrespective of their original trip length.

We assume a walking speed of 4 feet per second (LaPlante and Kaese, 2007). The maximum walking distance for the rider to or from a meeting point is 0.5 miles, which correspond to 11 minutes of walking at this speed. In addition, we impose the constraint that the total walk time cannot exceed the total time in a ride-share trip for a rider. In other words, the time that is spent walking in a trip must not exceed the time that is spent in the vehicle. This constraint can be manipulated by adjusting the rider flexibility parameter (assumed 1.0) which is the maximum ratio of the travel time to and from a meeting point to the time spent in the shared ride. (This additional restriction is enforced when searching for feasible meeting point arcs in the $k-d$ tree for each rider $j \in R$.)

A rider may be picked up at a meeting point or at his origin and dropped off at a meeting point or at his destination. A match involving two or more riders always starts and ends at a meeting point. Irrespective of the pickup or drop-off location, we always assume a service time per stop of 2 minutes. Each driver has a capacity of 3 spare seats. We limit ourselves to matches with no more than three riders, since this is the number of free seats in a typical sedan if the driver is driving alone. Also, back benches in most personal vehicles typically cannot accommodate three adults without compromising comfort.

The characteristics of the base case instances are summarized in Table 1.

Table 1: Characteristics of the base case instances.

Trip pattern:	suburb to center
Avg. number of participants:	2849.4
Avg. number of drivers:	1425.8
Avg. number of riders:	1423.6
Avg. trip distance for driver:	7.58 mi
Avg. trip distance for rider:	7.64 mi
Avg. trip duration for driver:	30.34 min
Avg. trip duration for rider:	30.56 min
Max. distance to a meeting point:	0.5 mi
Travel (walk) speed to/from meeting point:	4 ft/s
Max. walk time to meeting point:	11 min
Driving speed:	15 mi/h
Rider flexibility parameter:	1.0
Driver flexibility parameter:	0.25
Maximum flexibility of driver:	20 min
Vehicle capacity:	3 seats

5.2 Performance

Both the algorithm for generating feasible matches and the simulation framework are implemented in Python 2.7. CPLEX 12.6 is used for solving matching problems.

The base case instances solve in less than 150 seconds on a quad-core i5-3360M machine with 4GB of RAM. CPLEX solves the two integer programs (recall that we employ hierarchical optimization) in a few seconds in all settings; virtually all the time for these instances is spent generating feasible matches. Instances with increased rider flexibility (Section 5.5) and increased participant density (Section 5.6) take more time, up to 10 minutes in a few cases, because the number of feasible matches increases.

These run times suggest that the algorithm is appropriate for use in practice. The instances used in our computational study represent trip announcements accumulated over several hours. In practice, in a dynamic setting, instances with a much smaller set of driver and rider announcements have to be solved at any one time. Furthermore, instead of having to generate a set of matches from scratch for each optimization, the existing set of matches has to be updated given any new information that has become available (e.g., matches involving certain riders and drivers have to be deleted and new matches involving riders and drivers that have just announced their trips have to be generated). This will be a matter of

seconds rather than minutes. Performance wise, we expect to see similar results in a dynamic setting (albeit somewhat worse). For an environment without meeting points, Wang et al. [2014] have shown that the gap between a dynamic rolling horizon solution and a static benchmark is quite small. The gap will likely increase somewhat in an environment with meeting points, because some of the matches have to be committed to earlier (i.e., at the time that the rider has to start walking towards the meeting point).

5.3 Experiments

The main aim of this research is to analyze and quantify the benefits that meeting points can bring to a ride-sharing system. The solution approach that has been implemented provides a good basis for this, because it not only provides an optimal set of matches (for different objectives), but also furnishes the set of all feasible matches. We use the instance data and the set of feasible matches to compute and evaluate a number of metrics that provide insight into the quality of the optimal matching. In all the experiments, we either use the base case setting or a setting in which one of the characteristics is changed in order to assess the sensitivity of an optimal matching to this characteristic.

We evaluate and compare solutions using the following metrics: (1) the *matching rate for participants*, i.e., the fraction of participants that are matched, (2) the *matching rate for drivers*, i.e., the fraction of drivers that are matched, (3) the *matching rate for riders*, i.e., the fraction of riders that are matched, (4) the *mileage savings*, i.e., the relative mileage savings – system-wide vehicle-miles savings as a fraction of system-wide vehicle-miles when all participants drive alone, (5) the *driver trip time increase*, i.e., the average relative increase in the trip time of a driver – driver trip time increase as a fraction of original trip time, (6) the *rider trip time increase*, i.e., the average relative increase in the trip time of a rider – rider trip time increase as a fraction of original trip time, and (7) the *walking time*, i.e., the average walking time for a matched rider with a match that involves at least one meeting point.

5.4 Benefits of Meeting Points

As mentioned above, this research focuses on analyzing and quantifying the benefits of meeting points in a ride-sharing system. In Table 2, we compare the solution for the base case setting without meeting points to the solutions for the base case settings with 1,2 and 4 meetings points per TAZ, averaged over 5 randomly generated instances. To gain further insight, we also report statistics for two additional settings: in the first setting (labeled 4*), there are 4 meeting points per TAZ, but only single rider – single driver matches are allowed, and in the second setting (labeled 4**), there are 4 meeting points per TAZ, but only rider – driver matches using the closest meeting point to a rider's origin and destination are allowed. This reflects a setting in which the riders specify a particular meeting point upfront.

Table 2: Results for different numbers of meeting points and types of matches.

	0	**1**	**2**	**4**	**4***	**4****
System:						
Matching rate (%)	68.00	71.14	72.90	**74.83**	74.13	69.71
Mileage savings (%)	27.39	28.36	28.93	**29.63**	29.24	27.59
Drivers:						
Matching rate (%)	67.96	70.93	72.45	**74.08**	74.08	69.65
Trip time increase (%)	25.45	25.98	26.31	**26.41**	26.19	25.77
Riders:						
Matching rate (%)	68.11	71.43	73.43	**75.65**	74.26	69.84
Trip time increase (%)	13.09	19.27	22.74	**26.54**	16.43	16.42
Walk time (min:sec)	-	8:06	8:28	**8:56**	8:45	5:06

We see that the introduction of meeting points results in a substantial increase in the number of participants matched (our primary objective) as well as in the mileage savings (our secondary objective). The matching rate increases by 6.8% when there are 4 meeting points per TAZ. The matching rate increase is slightly larger for riders than for drivers, because of matches involving more than one rider. The average trip time for matched drivers increases less than one percent (from 25.45% to 26.41%), but, as expected, the average trip time for matched riders increases noticeably, by slightly more than 12 percent (from 13.09% to 26.54%). This increase is due to the walking that is required for certain riders to or/and from a meeting point; on average the total walking time is between 8 and 9 minutes, which corresponds to a distance of about 0.4 miles. Riders with a match involving a pickup meeting

point need to plan and execute their trips more carefully so as to ensure that they arrive at the meeting point in time. This may be considered an inconvenience, but, on the other hand, the service level (in terms of the chance of being matched) improves significantly. The results also suggest that most of the benefits can be achieved with single rider – single driver matches (4*) and that it is essential to consider all meeting points within range of a rider's origin or destination (4**).

Figure 5 shows the number of single, double, and triple rider matches in the optimal solution for different numbers of meeting points. We see that the number of participants

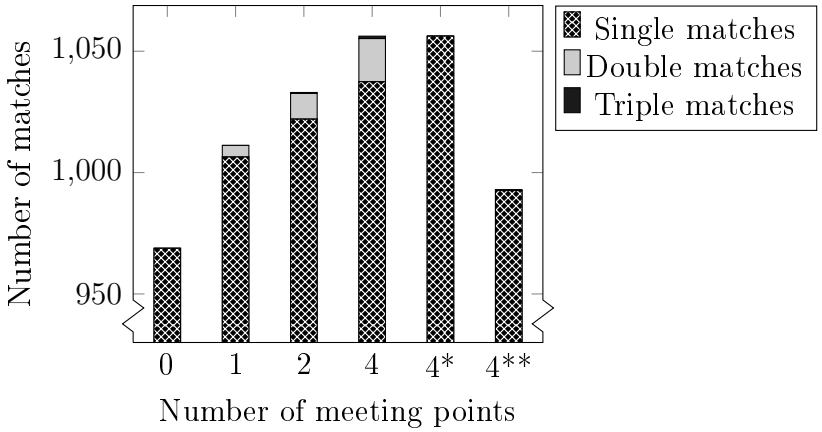

Figure 5: Number of single, double, and triple matches for different numbers of meeting points.

in matches with two or three riders is quite small, 2.5% and 0.2%, respectively, of the total number of matched participants when there are 4 meeting points per TAZ. This suggests that the primary benefit of the introduction of meeting points is an increase in the number of single rider – single driver matching opportunities (rather than being able to create multiple rider – single driver matches). However, to some extent, this result may be a consequence of our choice of objective hierarchy: maximize the number of matched participants followed by maximizing the mileage savings. When the number of drivers and riders in the system is roughly the same (as in our base case instances), it is more desirable to have single rider – single driver matches. That is, if it is possible to match two riders with the same driver, but it is also possible to match the two riders with different drivers, then the latter option is preferred as it results in four matched participants while the former results in only three matched participants. We take a closer look at the impact of the choice of objective hierarchy

in Section 5.7.

Next, we examine the use of meeting points in more detail. In Figure 6, we show how many of the matches in the optimal solution use two meeting points, only a pick-up meeting point, only a drop-off meeting point, or no meeting points at all. As expected, the fraction

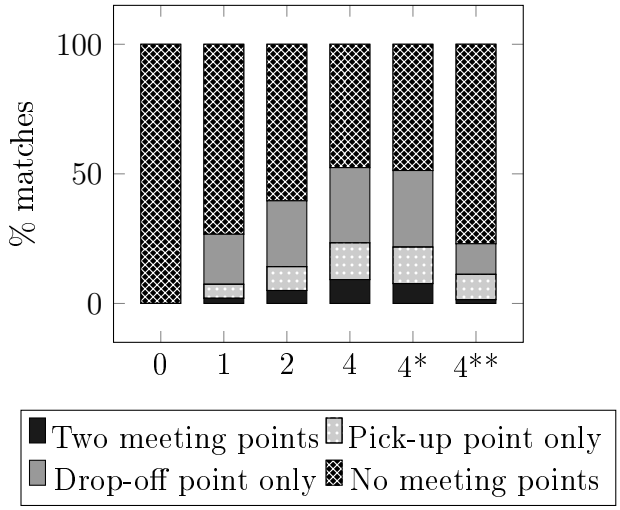

Figure 6: Use of meeting points in matches for different numbers of meeting points.

of matches involving meeting points increases as the number of meeting points per TAZ increases. The fact that the fraction of matches that use only a drop-off point is much larger than the fraction of matches that use only a pickup point is a consequence of the fact that the instances represent trips during a morning commute with destinations mostly in the center of Metro Atlanta, which has a higher concentration of TAZs (each covering a smaller geographic area) and consequently a higher concentration of meeting points.

Table 3 provides further information regarding the matches in an optimal solution. Specifically, we report the fraction of matches in the optimal solution that did not involve a meeting point, the fraction of matches in the optimal solution for which the mileage savings are higher because of the use of meeting points, and the fraction of matches in the optimal solution that would have been infeasible if it were not for the use of meeting points. For the latter set, we also identify the reason(s) that the use of meeting points resulted in a feasible match, i.e., the driver detour would have been infeasible without the use of meeting points, the participants' time windows would have been incompatible without the use of meeting points, the distance savings would have been negative without the use of meeting points.

Table 3: Charateristics of the matchings in the optimal solution in terms of their use of meeting points for different numbers of meeting points.

	0	**1**	**2**	**4**	**4***	**4****
No meeting points used (%)	100.00	73.31	60.38	**47.57**	48.72	76.98
Higher mileage savings (%)	-	23.26	35.13	**47.77**	47.45	20.17
Feasible because of meeting points (%)	-	12.75	19.40	**25.30**	24.75	7.47
- Detour became feasible (%)	-	11.89	18.14	**23.41**	22.75	6.77
- Time windows became feasible (%)	-	1.58	2.17	**3.20**	3.23	0.84
- Mileage savings became positive (%)	-	-	-	-	-	-

Note that a match can be counted in several categories, e.g., a match that is feasible because of the use of meeting points, could have been detour infeasible and time window infeasible. Note that all matches involving multiple riders are (by definition) feasible because of the use of meeting points and, for simplicity, all such matches are considered to have resulted in higher mileage savings.

We observe that when there are 4 meeting points per TAZ, the fraction of matches in the optimal solution that do not use meeting points is a little less than 50% and the fraction of matchings that would have been infeasible without meeting points is a little more than 25%. Furthermore, the use of meeting points makes matches feasible predominantly because it allows a smaller detour for the driver (only in a few cases, it makes rider and driver time windows compatible).

The fact that the fraction of matches in the optimal solution that do not use meeting points is close to 50% suggests that a more careful selection of meeting point locations may result in larger mileage savings. (Recall that meeting points have been selected randomly within a TAZ in these instances.)

Finally, in Table 4, we take a look at the number of additional feasible matching options generated by the introduction of meeting points. We show the number of riders (or pairs of riders or triples of riders) with at least one feasible match and the total number of feasible matches. Without meeting points, approximately 90.6% of the riders have at least one feasible match. With meeting points, this fraction increases to approximately 92.5%. We see too that as the number of meeting points increases, the number of feasible matches grows steadily. There are about 27.5% more feasible matches for riders when there are

Table 4: Analysis of the number of feasible matches for different numbers of meeting points.

	0	**1**	**2**	**4**	**4***	**4****
Single riders with feasible match	1290.6	1302.6	1308.4	**1316.0**	1316.0	1302.0
Number of single rider matches	20253.0	22940.6	24297.0	**25836.0**	25836.0	21565.0
Pairs of riders with feasible match	-	9.8	28.6	**53.4**	-	0.6
Number of rider pair matches	-	145.4	481.2	**994.6**	-	9.0
Triples of riders with feasible match	-	-	0.8	**2.4**	-	-
Number of triple rider matches	-	-	10.8	**28.6**	-	-

4 meeting points per TAZ. Not surprisingly, the increases are even more pronounced for matches involving pairs and triples of riders. This demonstrates that to increase the number of multi-rider matches, it will be critical to have a large number of carefully located meeting points.

5.5 Impact of Time Flexibility

In this section, we study the impact of the time flexibility of the participants on the performance of the system and the benefits of meeting points. We separately vary the time flexibility of the drivers and the riders.

In the base case, we consider a driver time flexibility of 25% of the original trip time ($c_{flex} = 0.25$). This time flexibility c_{flex} refers to the maximum extra trip time the drivers are willing to accept to serve one or more riders. Furthermore, all participants are assumed to have 30 minutes of flexibility in their trip departure time. To assess the impact of the time flexibility on the performance of a ride-sharing system, we evaluate the system performance when the time flexibility is lower, i.e., $c_{flex} = 0.15$, and when the time flexibility is higher, i.e., $c_{flex} = 0.35$. We note that extra trip time for drivers (and extra trip time for riders) always includes the service time incurred at a pick-up and a drop-off location. The results of these experiments are found in Table 5.

We see that the willingness of drivers to accept a larger extra trip time has a substantial effect on the matching rate and the mileage savings. We also see that the negative impact of a decrease in time flexibility is larger than the positive impact of an increase, which suggests that there will be diminishing returns from increasing time flexibility. We observe too that

Table 5: Effects of driver time flexibility.

	$c_{flex} = 0.15$		$c_{flex} = 0.25$		$c_{flex} = 0.35$	
	0	**4**	**0**	**4**	**0**	**4**
System:						
Matching rate (%)	56.68	64.96	68.00	74.83	75.41	**82.11**
Mileage savings (%)	23.70	26.65	27.39	29.63	29.23	**30.89**
Drivers:						
Matching rate (%)	56.66	64.27	67.96	74.08	75.36	**81.19**
Trip time increase (%)	19.35	20.66	25.45	26.41	30.65	**32.31**
Riders:						
Matching rate (%)	56.77	65.71	68.11	75.65	75.54	**83.11**
Trip time increase (%)	13.09	25.01	13.09	26.54	13.09	**27.91**
Walk time (min:sec)	-	8:52	-	8:56	-	**9:08**

the benefit of meeting points is negatively correlated with the time flexibility of the drivers. That is, the difference in participant matching rates is highest for the most constrained case (8.28%) and smallest for the least constrained case (6.70%). This points to the fact that meeting points are most valuable when drivers are reluctant to add extra time to their trip (e.g., on their way to work in the morning).

In the next set of experiments, we vary the time flexibility of the riders. In particular, we vary the travel speed and the travel range of the riders, i.e., the time it takes a rider to reach a meeting point and the distance a rider is willing to travel to reach a meeting point. Such an increase may be possible if riders use other modes of transportation instead of walking to get to a meeting point, e.g. using a (folding) bike, public transport, riding with a member of their household, etc. We increase the speed from the speed of walking (4 ft/s) to the speed of a cyclist (12 ft/s), and we increase the allowable range for the meeting points from 0.5 (base case) to 0.75 miles. Note that we maintain the assumption that the total travel time to and from a meeting point cannot exceed the time spent in the shared ride. We report selected results for this experiment in Table 6.

We see that the willingness to consider more distant meeting points combined with the ability to get to a meeting point faster than by walking can greatly increase system performance. The matching rates are much higher and also the number of feasible double and triple matches increases significantly. It is important to observe that only increasing the walking range results in improved system performance.

Table 6: Effects of rider time flexibility.

Travel speed to meeting point	-	Low		High	
Maximum distance to meeting point (mi)	-	0.5	0.75	0.5	0.75
System:					
Matching rate (%)	68.00	74.83	79.84	76.17	**83.84**
Mileage savings (%)	27.39	29.63	31.32	30.01	**32.14**
Drivers:					
Matching rate (%)	67.96	74.08	77.72	75.14	**81.5**
Trip time increase (%)	25.45	26.41	28.02	26.85	**27.33**
Riders:					
Matching rate (%)	68.11	75.65	82.02	77.26	**86.22**
Trip time increase (%)	13.09	26.54	38.78	16.88	**21.27**
Trip time to/from m. point (min:sec)	-	8:56	13:13	3:11	**5:56**
Trip distance to/from m. point (mi)	-	0.40	0.59	0.43	**0.70**

If we examine the structure of the optimal matchings in the most flexible scenario in more detail, we find that 74.47% of the matches use meeting points, compared to 52.4% in the base case (see Figure 6). Also, we find that 45.11% of these matches would be detour-infeasible without the meeting points, compared to 23.41% in the base case (see Table 3).

These findings stress the importance of encouraging riders to consider more distant meeting points and of encouraging drivers to accept longer detours. A ride-sharing service may investigate the benefits of incentive payments to riders and drivers that are willing to be more flexible as a way to increase the matching rate and the mileage savings.

5.6 Effect of Trip Patterns and Density

In this section, we study the effect of the number of participants in the system and their trip patterns on the system performance. Table 7 gives an overview of the characteristics of the instances that we generated for this purpose. First, we consider a setting with twice as many participants than in the base case (denote by $2:2$). We also consider a setting with twice as many riders but the same number of drivers as in the base case (denoted by $1:2$). This represents an environment in which the pool of ride-share participants is skewed towards the riders, who have more to gain from participating. To study the effect of a different trip patterns, we create a set of instances in which participants travel along a narrow South-

Table 7: Characteristics of instances with different trip patterns and densities.

	drivers : riders			
	1 : 1	2 : 2	1 : 2	$1 : 1^c$
Trip pattern	default	default	default	corridor
Avg. number of participants	2849.4	5578.6	4272.4	2594.4
Avg. number of drivers	1425.8	2777.8	1425.8	1295.4
Avg. number of riders	1423.6	2800.8	2846.6	1299.0
Avg. trip distance for driver (mi)	7.58	7.60	7.58	9.36
Avg. trip distance for rider (mi)	7.64	7.62	7.64	9.35
Avg. trip duration for driver (min)	30.34	30.39	30.34	37.43
Avg. trip duration for rider (min)	30.56	30.47	30.56	37.38

North corridor in the Atlanta region. While in the base case (denoted by default) the area is shaped like a square with trips originating in suburban areas and heading towards the urban center, the corridor instances represent trips that occur in a narrow rectangle. To allow for a fair comparison, the geographic area covered in the five corridor instances is roughly the same as in the base case, and, similarly, the number of trips, TAZ locations, and meeting points is roughly the same as in the base case (this setting is denoted by $1 : 1^c$). Table 8 presents the results for the different experiments.

Table 8: Effects of trip patterns and density.

	drivers : riders							
	1 : 1		2 : 2		1 : 2		$1 : 1^c$	
	0	**4**	**0**	**4**	**0**	**4**	**0**	**4**
System:								
Matching rate (%)	68.00	74.83	75.02	82.91	52.40	59.75	72.16	78.26
Mileage savings (%)	27.39	29.63	31.06	33.65	21.83	25.42	31.35	33.26
Drivers:								
Matching rate (%)	67.96	74.08	75.34	81.57	78.49	84.22	72.31	77.51
Trip time increase (%)	25.45	26.41	25.75	27.43	25.45	26.13	22.56	23.46
Riders:								
Matching rate (%)	68.11	75.65	74.73	84.26	39.35	47.52	72.23	79.21
Trip time increase (%)	13.09	26.54	13.13	28.16	13.09	29.22	10.70	21.54
Walk time (min:sec)	-	8:56	-	9:19	-	9:41	-	9:00

As expected, we see that the matching rate increases with the number of participants. More surprising is the fact that the relative advantage of the use of meeting points in terms of the overall matching rate also seems to increase slightly with the density. A potential

explanation for this is that opportunities for matches with multiple riders increases.

With twice as many riders than drivers in the system, we see that 47.52% of the riders are matched, which is almost best possible (50%) if we ignore the possibility of double and triple matches. The number of double and triple matches has increased compared to the base case, but it is still relatively small. This suggests that the full potential of the meeting points cannot be achieved unless the meeting point locations are carefully chosen and both riders and drivers have sufficient time flexibility.

For the corridor instances, we see that both the matching rate and the mileage savings are approximately 4% higher than for the default instances, and that the benefits of the meeting points are similar. The same holds for the trip time increase for drivers and the walking distance for riders.

5.7 The Impact of Objective Hierarchies

All the results discussed so far were obtained using the objective hierarchy in which the number of matches (z_1) is maximized first followed by maximizing the mileage savings (z_2). We observed in Section 5.4 that this objective hierarchy tends to favor solutions involving single matches. In this section, we compare the system performance for three natural objective hierarchies. The first (*Hierarachy 1*) is the default one, and maximizes participant matches followed by mileage savings, the second (*Hierarachy 2*) maximizes mileage savings followed by participant matches, and the third (*Hierarachy 3*) maximizes rider matches followed by mileage savings. The latter may be more desirable than the default hierarchy, in which the primary objective is maximizing participant matches, because unmatched riders may not necessarily have the option of using their own car to perform their trip. The results for the five base case instances and 4 meeting points per TAZ can be found in Table 9.

We see that the difference for all but one of the system performance metrics for the three objective hierarchies is less than one percentage point. The exception is the matching rate for the drivers, which for the default objective hierarchy (*Hierarchy 1*) is 1.42% larger than for the objective hierarchy in which the primary focus is on maximizing mileage savings (*Hierarchy 2*). This reflects the structural difference in the optimal matchings: there are almost

Table 9: Results for different objective hierarchies.

	Hierarchy 1 (Matches – Savings)	*Hierarchy 2 (Savings – Matches)*	*Hierarchy 3 (R. Matches – Savings)*
System:			
Matching rate (%)	**74.83**	73.88	74.36
Mileage savings (%)	29.63	**29.79**	29.74
Drivers:			
Matching rate (%)	**74.08**	72.66	73.14
Trip time increase (%)	**26.41**	25.58	26.09
Riders:			
Matching rate (%)	**75.65**	75.17	**75.65**
Trip time increase (%)	26.54	26.01	26.41
Walk time (min:sec)	8:56	8:51	8:57
Matching:			
Num. of single matches	**1037.4**	1003.8	1011.0
Num. of double matches	17.8	**30.6**	30.4
Num. of triple matches	1.0	**1.4**	**1.4**

twice as many double matches when the primary objective is to maximize mileage savings. Interestingly, the matching rate for riders does not increase when maximizing the number of matched riders is taken as the primary objective (rather than maximizing the number of match participants). Triple matches are still rare in all solutions. From a ride-sharing service provider's perspective, the default objective hierarchy is likely to be preferred, as their revenue is linked to the participant matching rate. However, from a societal perspective, the alternative objective hierarchy in which the number of riders matched is the primary objective is probably preferable, as it strikes a better balance between rider mobility and mileage savings (which are linked to congestion and emissions).

Since the number of matched participants is an integer, it is relatively easy to compute the Pareto frontier that characterizes the trade-off between the number of matched participants and the mileage savings. Figure 7 depicts the Pareto frontier for the third of the five base case instances (the frontier for the other base case instances look similar). As already indicated by the small differences in the values of the performance metrics in Table 9, the two objectives are well aligned. This is also reflected in the small number of points that constitute the Pareto frontier.

Figure 8 provides more detail regarding the change in the structure of the optimal match-

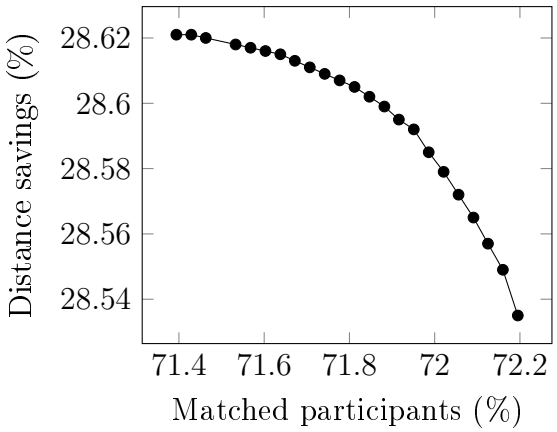

Figure 7: Pareto frontier for the third base case instance.

ings as we move from a solution obtained with the default objective hierarchy to a solution obtained with the alternative objective hierarchy in which the primary focus is on mileage savings. We see that single rider – single driver matches are replaced by matches involving two or three riders. As a consequence, the number of drivers with matches decreases. Surprisingly, the number of riders with matches also decreases, which indicates that in many situations riders are "competing" for the same drivers.

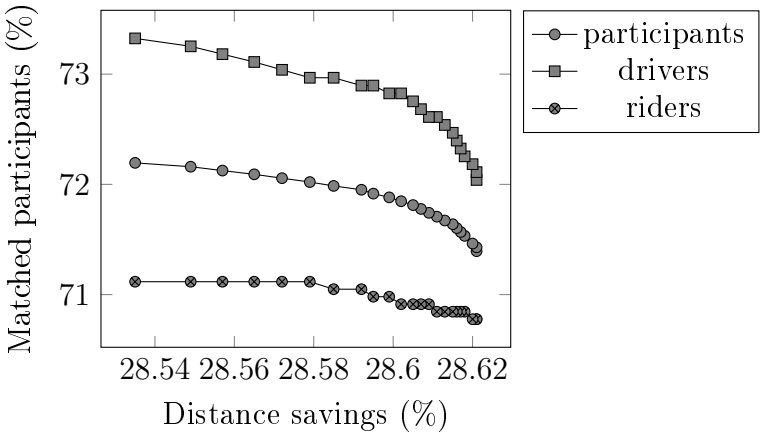

Figure 8: Matching rates for Pareto efficient points for the third base case instance.

6 Concluding Remarks

In this study, we have shown that the introduction of meeting points in a ride-sharing system can substantially improve a number of critical performance metrics, i.e., percentage

of matched riders, percentage of matched participants, and mileage savings. The price that has to be paid to achieve these performance increases is minor: riders may have to walk a short distance and may have to plan their time more carefully so as to ensure that they arrive on time at the meeting point where they are to be picked up (it is unlikely that drivers will be willing to wait for a rider at a pickup point for more than a minute or two). Even though the number of possible matches increases significantly with the introduction of meeting points, our computational experiments have demonstrated that all feasible matches can be generated efficiently with a carefully designed and implemented algorithm.

The observed increases in performance of a ride-sharing system resulting from the introduction of meeting points may even be greater when the meeting points are chosen carefully based on observed travel patterns. This is an interesting opportunity for further research.

As expected, driver and, especially, rider flexibility strongly impact the performance of a ride-sharing system. This points to two additional and interrelated future research directions: (1) how to stimulate (and reward) riders to increase their flexibility and be willing to use more distant meeting points, and, similarly, how to stimulate (and reward) drivers to increase their flexibility and be willing to make longer detours, and (2) how to (better) integrate ride-sharing systems with other available transportation systems, e.g., bike-sharing systems, so as to ensure that riders can reach meeting points that are further away fast and easy.

References

N. A. H. Agatz, A. Erera, M. P. W. Savelsbergh, and X. Wang. Dynamic ride-sharing: A simulation study in metro atlanta. *Transportation Research Part B*, 45(9):1450–1464, 2011.

N. A. H. Agatz, A. Erera, M. P. W. Savelsbergh, and X. Wang. Optimization for dynamic ride-sharing: A review. *European Journal of Operational Research*, 223(2):295–303, 2012.

A. Amey. Proposed Methodology for Estimating Rideshare Viability Within an Organization: Application to the MIT Community. *Transportation Research Board Annual Meeting 2011*, Paper 11-2585, 2011.

Jon Louis Bentley. K-d trees for semidynamic point sets. In *Proceedings of the Sixth Annual Symposium on Computational Geometry*, SCG '90, pages 187–197, New York, NY, USA, 1990. ACM. ISBN 0-89791-362-0.

Florian Drews and Dennis Luxen. Multi-hop ride sharing. In *Proceedings of the Sixth Annual Symposium on Combinatorial Search*, pages 71–79. AAAI Press, 2013.

Michael Drexl. Synchronization in vehicle routing-a survey of vrps with multiple synchronization constraints. *Transportation Science*, 46(3):297–316, 2012.

M. Furuhata, M. Dessouky, F. Ordonez, M.E. Brunet, X. Wang, and S. Koenig. Ridesharing: The state-of-the-art and future directions. *Transportation Research Part B: Methodological*, 57(0):28 – 46, 2013. ISSN 0191-2615.

Asvin Goel and Frank Meisel. Workforce routing and scheduling for electricity network maintenance with downtime minimization. *European Journal of Operational Research*, 231(1):210–228, 2013.

Wesam Herbawi and Michael Weber. Evolutionary multiobjective route planning in dynamic multi-hop ridesharing. In *Evolutionary Computation in Combinatorial Optimization*, pages 84–95. Springer, 2011.

Levent Kaan and Eli V Olinick. The vanpool assignment problem: Optimization models and solution algorithms. *Computers and Industrial Engineering*, 66(1):24–40, 2013.

A. Lee and M. P. W. Savelsbergh. Dynamic ridesharing: Is there a role for dedicated drivers? *Working paper*, 2014.

Frank Meisel and Herbert Kopfer. Synchronized routing of active and passive means of transport. *OR spectrum*, 36(2):297–322, 2014.

J. Park and B. Kim. The school bus routing problem: A review. *European Journal of Operational Research*, 202(2):311–319, 2010.

Jorge Riera-Ledesma and Juan José Salazar-González. A column generation approach for a school bus routing problem with resource constraints. *Computers and Operations Research*, 40(2):566–583, 2013.

Patrick Schittekat, Joris Kinable, Kenneth Sorensen, Marc Sevaux, Frits Spieksma, and Johan Springael. A metaheuristic for the school bus routing problem with bus stop selection. *European Journal of Operational Research*, 229(2):518–528, 2013.

D. Schrank, B. Eisele, and T. Lomax. Tti's 2012 urban mobility report. Technical report, Texas A&M Transportation Institute, December 2012.

X. Wang, N. A. H. Agatz, and A. Erera. Stable matching for dynamic ride-sharing systems. *Working paper*, 2014.

Appendix A

Algorithm 1: Refined Feasible Match Generation

```
build k − d tree with meeting points ;
build interval container with rider time windows ;
for each rider do
    query k − d tree and store feasible meeting point arcs ;
end
for each driver do
    query interval container to obtain compatible riders ;
    for each compatible rider do
        for each rider meeting point arc do
            if match driver, rider, meeting point arc is time feasible then
                store meeting point arc ;
                compute driving distance savings ;
                if driving distance savings > best match driving distance savings then
                    update best match driving distance savings ;
                    update best match ;
                end
            end
        end
        if best match driving distance savings > 0 then
            append best match to match list ;
            append rider to feasible rider list ;
        end
    end
    if number of feasible riders > 1 then
        for k = 2, . . . , C_i − 1 do
            Retrieve meeting point arcs ;
            Remove meeting point arcs that are feasible for less than k riders ;
            ... ;
            if driving distance savings > 0 then
                append match to match list ;
            end
        end
    end
end
return match list ;
```

Appendix B

Interval trees are designed to efficiently find intervals that overlap. It takes $O(log(m))$ time to find an overlapping interval in a balanced interval tree, where m is the number of intervals in the tree. Unfortunately, in the ride-sharing setting, as many as $1/2$ of the rider time windows may overlap with a driver time window on average. Hence, if we denote the expected ratio of overlapping rider intervals by r_{exp}, we cannot expect the performance to be better than $O(r_{exp} \cdot log(m))$ with a standard interval tree. In practice, the performance of an interval tree might even be inferior to linear search due to issues with locality of reference.

A query in a standard interval tree returns all overlapping intervals – including those with a very small overlap. However, the time window of a driver and a rider have to overlap by at least the time needed to complete the shared part of their trip.

Consider Figure 9. Let t_j^{max} denote the time needed to drive distance d_j^{max}, the longest distance rider j is willing to walk to and from a meeting point. Thus, the minimum time the

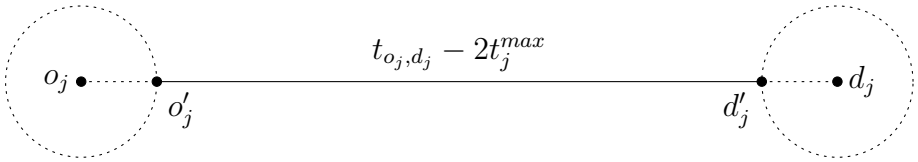

Figure 9: Minimum shared ride time for rider j

rider and driver will share is $t_j^{min} = t_{o_j,d_j} - 2t_j^{max}$. Therefore, we redefine the time window for rider j to be $[e_j + t_j^{max}, l_j - t_j^{max}]$.

Now consider Figure 10. It shows three possible ways in which the time window of a rider j can overlap with the time window of a driver i. The interval of the rider is stored in the memory structure, and the three intervals related to drivers i, i', and i'' represent three possible queries. The overlap can be from the right of interval $[e_j, l_j]$, as with $[e_{i'}, l_{i'}]$, it can be from the left of interval $[e_j, l_j]$, as with $[e_{i''}, l_{i''}]$, or it can span the entire interval $[e_j, l_j]$, as with $[e_i, l_i]$.

Using this simple observation, we see that if the overlap in the query has to be at least t_j^{min}, then we can shorten the rider intervals $[e_j, l_j]$ in our memory structure by t_j^{min}. We can shorten the rider intervals from the left *and* from the right. That is, we shorten the intervals by adding t_j^{min} to the lower end and by subtracting t_j^{min} from the upper end, i.e.

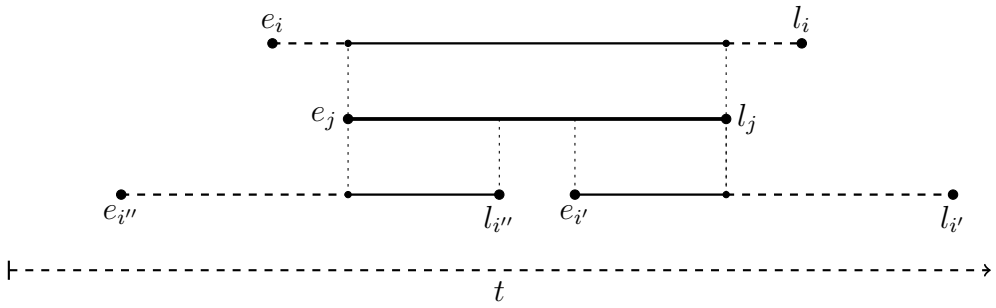

Figure 10: Overlaps of the time interval of rider j with queries corresponding to three drivers i,i',i''

$[e'_j, l'_j] = [e_j + t_{o_j,d_j} - t_j^{max}, l_j - t_{o_j,d_j} + t_j^{max}]$. This may result in: (1) $e'_j < l'_j$ or (2) $e'_j \geq l'_j$. For each of these two cases, we analyze the three possible types of driver queries.

Figure 11 depicts Case (1): the intervals of drivers i, i', and i'', represent all possible types of overlap. Queries in an interval tree with reduced rider intervals for drivers i' and

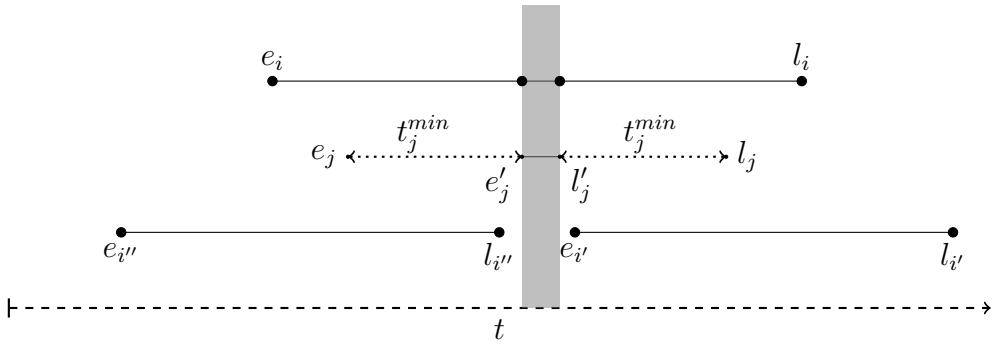

Figure 11: Shortening the time interval of rider j and overlaps with queries corresponding to drivers i, i', and i''

i'' will not return rider j as a compatible rider. The query for driver i on the other hand, will return rider j, since the overlap is greater than t_j^{min}. Thus, using reduced intervals in a standard interval tree produces the desired results.

Let us now consider Case (2). If $e'_j \geq l'_j$ (as in Figure 12), we have an inverted interval $[l'_j, e'_j]$. In such a situation, there is sufficient overlap only if interval $[l'_j, e'_j]$ is a sub-interval of interval $[e_i, l_i]$. The intervals of drivers i' and i'' have some overlap with the inverted interval $[l'_j, e'_j]$, but only the interval of driver i has sufficient overlap.

Queries using a standard interval tree with reduced time windows will return all time feasible riders for a driver, but, unfortunately, may also return some time infeasible riders

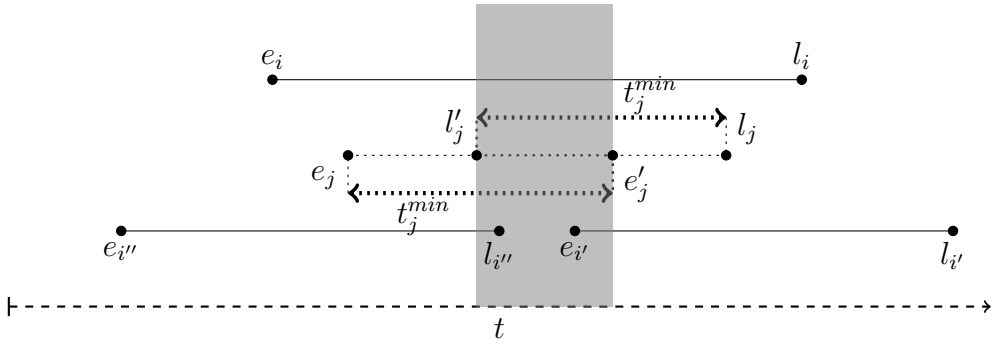

Figure 12: Inverted time interval of rider j and overlaps with queries corresponding to drivers i, i', and i''

(i.e., the driver's interval is too small to accommodate the rider). This disadvantage is outweighed by the fact that the intervals in the interval tree are much smaller and, thus, the queries are much faster. However, the number of riders returned can be quite large. Below, we propose an alternative approach that uses sorted lists.

We maintain a list of intervals, each with a lower end l and an upper end u. We first sort the intervals in non-decreasing order of their lower end. We then recursively compute an auxiliary upper end u^* for each interval in the sorted list. The auxiliary upper end is defined as $u_i^* = max(u_{i-1}^*, u_i)$ for an interval in position i in the list (for all positions $i \geq 1$) and $u_0^* = u_0$ for the interval in position 0. Intervals for which $u = u^*$ are super-intervals – they are not a sub-interval of any other interval in the list. Intervals for which $u < u^*$ are sub-intervals – they are a subset of at least one interval in the list.

If we are in Case (1), we find all intervals that potentially overlap with $[t_{low}, t_{high}]$ by finding the position of the leftmost interval in the sorted list with $t_{u*} \geq t_{low}$ and the position of the rightmost interval in the sorted list with $t_l > t_{high}$ using binary search. The two positions define the sublist we want. For Case (2), the query is even simpler: we first find the position of the leftmost interval in the sorted list with $t_l \geq t_{low}$ and the position of the rightmost interval in the sorted list with $t_l > t_{high}$. These positions define the sublist that contains all potential sub-intervals of $[t_{low}, t_{high}]$. We add the two resulting lists together and further refine them.

The speed of the search may be further improved by breaking up the sorted list into sublists. For each super-interval, we create a sublist containing all its sub-intervals. We

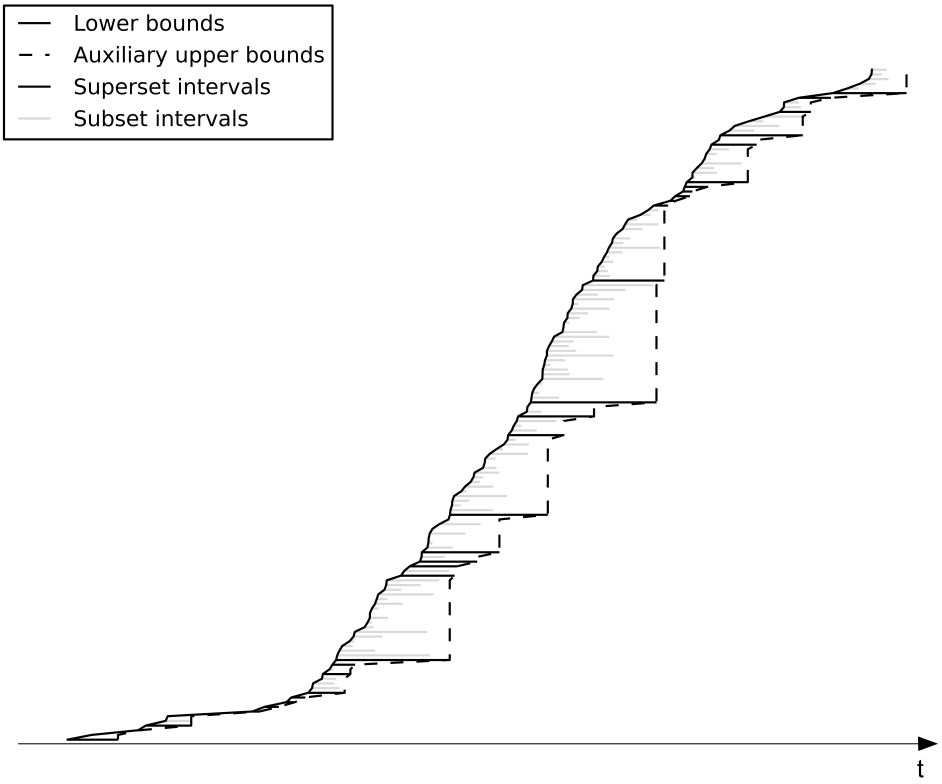

Figure 13: Interval list

only keep the super-intervals in the original list. Each sublist may be further broken down to a desired level of granularity (i.e., minimum number of intervals in a sublist). We know that all the intervals in a sublist of an interval are within its bounds, so we may use this information to explore only the relevant sublists. This approach can improve the search in large data sets, especially when super-intervals are large compared to sub-intervals. We managed to improve the query times by implementing this approach, but the run-times were only approximately $5 - 10$ % faster than with a single sorted list. We did not investigate this further, since the performance of the solution algorithm is not our major focus at this point in time.

ERIM Report Series *Research in Management*	
ERIM Report Series reference number	ERS–2015–003–LIS
Date of publication	2015–02–19
Version	19–02–2015
Number of pages	42
Persistent URL for paper	http://hdl.handle.net/1765/77595
Email address corresponding author	nagatz@rsm.nl
Address	Erasmus Research Institute of Management (ERIM) RSM Erasmus University / Erasmus School of Economics Erasmus University Rotterdam PO Box 1738 3000 DR Rotterdam, The Netherlands Phone: +31104081182 Fax: +31104089640 Email: info@erim.eur.nl Internet: http://www.erim.eur.nl
Availability	The ERIM Report Series is distributed through the following platforms: RePub, the EUR institutional repository Social Science Research Network (SSRN) Research Papers in Economics (RePEc)
Classifications	The electronic versions of the papers in the ERIM Report Series contain bibliographic metadata from the following classification systems: Library of Congress Classification (LCC) Journal of Economic Literature (JEL) ACM Computing Classification System Inspec Classification Scheme (ICS)