



# The shared-taxi problem: Formulation and solution methods



Hadi Hosni<sup>a</sup>, Joe Naoum-Sawaya<sup>b,\*</sup>, Hassan Artail<sup>a</sup>

<sup>a</sup> *Electrical and Computer Engineering Department, American University of Beirut, Beirut, Lebanon*

<sup>b</sup> *IBM Research – Ireland, IBM Technology Campus Building 3, Mulhuddart, Dublin 15, Ireland*

## ARTICLE INFO

### Article history:

Received 2 February 2014

Received in revised form 2 September 2014

Accepted 24 September 2014

### Keywords:

Shared taxi

Lagrangian relaxation

Integer programming

## ABSTRACT

With the rising fuel costs, ride sharing is becoming a common mode of transportation. Sharing taxis which has been prominent in several developing countries is also becoming common in several cities around the world. Sharing taxis presents several advantages as it minimizes vacant seats in cars thus reducing costs on taxi operators which results in significantly lower taxi fares for passengers. Besides the economical advantages, taxi sharing is highly important for reducing congestion on the roads and for minimizing the impact of transportation on the environment. In this paper, we formulate the problem of assigning passengers to taxis and computing the optimal routes of taxis as a mixed integer program. To solve the proposed model, we present a Lagrangian decomposition approach which exploits the structure of the problem leading to smaller problems that are solved separately. Furthermore, we propose two heuristics that are used to obtain good quality feasible solutions. The Lagrangian approach along with the heuristics are implemented and compared to solving the full problem using CPLEX. The computational results indicate the efficiency of the methodology in providing tighter bounds than CPLEX in shorter computational time.

© 2014 Elsevier Ltd. All rights reserved.

## 1. Introduction

Inefficiencies in transportation are nowadays causing several economical and environmental problems mainly due to the high levels of traffic congestion resulting in increasingly wasted resources and staggering amounts of pollution. Ride-sharing, which basically aims at minimizing the number of vacant seats in vehicles and therefore reducing the number of required vehicles, emerges as an essential practice to reduce traffic and fuel costs as well as to support the green initiatives that aim at improving the environment.

Several variants of the ride-sharing problem exist, most of which having to do with either developing efficient car pooling systems or optimizing the operation of public transportation. The recent survey of Furuhata et al. (2013) provided an extensive overview of the state of the art ride-sharing systems and discussed the existing challenges. Among the main challenges that have been identified are those that relate to service pricing and passengers matching. The survey of Agatz et al. (2012) focused on the recent research in the area of optimization models for ride-sharing particularly passenger matching and concluded that the field of transportation is far from mature and still has room for major contributions. A problem in which customers request rides from specific pickup locations to specific dropoff locations, while imposing certain requirements on the ride (such as the earliest pickup time and latest dropoff time), is known in the literature as the Dial-A-Ride Problem (DARP).

\* Corresponding author.

E-mail addresses: [hae23@aub.edu.lb](mailto:hae23@aub.edu.lb) (H. Hosni), [jnaoumsa@uwaterloo.ca](mailto:jnaoumsa@uwaterloo.ca) (J. Naoum-Sawaya), [hartail@aub.edu.lb](mailto:hartail@aub.edu.lb) (H. Artail).

Several variants of DARP exist including single-vehicle and multi-vehicle DARP as well as static and dynamic DARP. The dynamic version accommodates for changes that include users leaving and entering the system, and most importantly handles new customer requests well after the fleet of riders is spread and already traveling.

The shared-taxi problem falls under the multi-vehicle dynamic DARP, and is defined as follows. Taxi-seekers place a request by indicating pickup locations and their desired dropoff locations. They also indicate the earliest acceptable pickup time and the latest acceptable dropped off time as well as a maximum ride time. The fare that is paid by each passenger is based on the pickup zone and the dropoff zone. This form of taxi sharing has been the main form of public transportation in many developing countries. Very recently, such taxi-sharing methods became common in several cities such as New York, London, and Montreal ([Transport Canada, 2011](#)). The shared-taxi problem seeks to determine the optimal assignment of passengers to taxis as well as the optimal route for each taxi.

In this paper, the shared-taxi problem is formulated as a mixed integer program. A Lagrangian decomposition approach that exploits the structure of the problem is proposed as a solution methodology. We also propose a novel heuristic that finds good quality feasible solutions within limited computational time thus making it appropriate for real-time implementations. Extensive computational testing showed that the proposed Lagrangian decomposition outperforms solving the full mixed integer program while the proposed heuristics provide good quality feasible solutions.

The rest of the paper is organized as follows. Section 2 presents a literature review, while Section 3 describes the problem formulation. Section 4 on the other hand presents the Lagrangian decomposition approach, and Section 5 presents a heuristic for finding good feasible solutions. The proposed Lagrangian decomposition approach and heuristics are evaluated in Section 6. Finally, Section 7 concludes and illustrates future research directions.

## 2. Literature review

The DARP is a generalization of vehicle routing problems, like the Pick-up and Delivery Vehicle Routing Problem and the Vehicle Routing Problem with Time Windows, which have been studied in the literature ([Desrosiers et al., 1995](#); [Desaulniers et al., 2002](#)). In [Psaraftis \(1980\)](#) and [Psaraftis \(1983\)](#), a dynamic programming approach, in which the objective function takes into consideration both cost minimization and customer dissatisfaction minimization, was proposed and small instances of the single-vehicle static and dynamic DARP were solved. [Cordeau and Laporte \(2003\)](#) presented a tabu search heuristic for the multi-vehicle static DARP where ride seekers specify time windows on pickups and drop-offs, and drivers serve as many ride seekers as possible starting from a common point and arriving at a final common point. A similar problem where all ride seekers and ride providers are heading to the same destination (employees of the same company heading to office), was solved in [Baldacci et al. \(2004\)](#) using a column generation approach. By nature, such a problem is not dynamic since all inputs are known beforehand and the scheduling can happen as much as days earlier. In contrast, [Attanasio et al. \(2004\)](#) proposes parallel tabu search heuristics for the dynamic DARP where an initial static solution is first achieved and then the new passenger requests are inserted into the solution as they arrive. Other metaheuristics such as the genetic algorithm that is presented in [Jorgensen et al. \(2006\)](#), the variable neighborhood search algorithm of [Parragh et al. \(2010\)](#), the stochastic variable neighborhood search algorithm of [Schilde et al. \(2011\)](#), the granular tabu search algorithm of [Kirchler and Wolfler Calvo \(2013\)](#), and the multicriteria tabu search of [Paquette et al. \(2013\)](#) also proved to be successful in finding high quality solutions. It is worth noting that the approach of [Paquette et al. \(2013\)](#) considers a multicriteria objective which includes cost minimization in addition to quality of service measures that relate to waiting and travel times.

An exact approach for solving the multi-vehicle static DARP was presented in [Cordeau \(2006\)](#) where a branch-and-cut algorithm that uses valid inequalities that are derived from the dial-a-ride, the traveling salesman, the vehicle routing, and the pick-up and delivery problems is used to efficiently solve small to medium-size instances. Based on the 3-index formulation that was presented in [Cordeau \(2006\)](#), [Ropke et al. \(2007\)](#) proposed two new models based on a 2-index formulation and new families of valid inequalities are derived to strengthen the models.

In [Coslovich et al. \(2006\)](#), the dynamic aspect of the DARP is expressed when drivers which are en-route suddenly receive requests at known stops and must make a decision on whether or not to service these requests. The decision is based on whether or not an efficient insertion algorithm succeeds into inserting the request into the already computed route. [Parragh et al. \(2012\)](#) presented extensions of the 3-index of [Cordeau \(2006\)](#) and the set partitioning formulation of [Ropke et al. \(2007\)](#) to include heterogeneous fleet and passengers, where vehicles can have different capacities and passengers can request different modes of transportation. The effective solution approach integrates variable neighborhood search into a column generation algorithm, and high-quality solutions for realistic test instances are presented.

[Xiang et al. \(2006\)](#) and [Xiang et al. \(2008\)](#) presented fast heuristics for the multi-vehicle static and dynamic DARP based on Local search and several diversification and improvement strategies that improve initial solutions. The special application of dynamic DARP to transporting patients between several locations in a hospital campus was also discussed in [Beaudry et al. \(2010\)](#) which presented a new solution approach based on a two-phase heuristic that includes a tabu search algorithm. In [Agatz et al. \(2011\)](#), a dynamic carpooling problem is considered and a model based on rolling horizon is compared to a greedy heuristic using a simulation study based on travel demand data from metropolitan Atlanta. Most recently, [Berbeglia et al. \(2012\)](#) proposed a strategy in which requests are treated one by one upon arrival. A hybrid algorithm

consisting of a tabu search heuristic and a constraint programming algorithm running in parallel continuously optimizes the newly arrived requests. A request is accepted when either of the two algorithms identifies a feasible solution. In a recent survey, Agatz et al. (2012) argue that a centralized ride-share matching may not be computationally feasible and therefore research should focus on finding efficient ways of decomposing the problem. Furthermore, it was acknowledged that a geographic decomposition of the problem is likely to fail since origins and destinations could lie in different geographical divisions.

The main contributions of this paper are as follows. (a) This paper presents a general mixed integer programming model for the shared-taxi problem, which generalizes several aspects of the dynamic multi-vehicle DARP problem. Particularly, vehicles can be assumed to be at different locations, have different capacities, and can have passengers onboard. Onboard passengers along with their own constraints are fundamental to model a dynamic system where passengers enter and exit the system at different points in time. Additionally, the proposed models allow for potential passengers to be rejected and thus not assigned to a taxi if they are deemed unprofitable. The proposed model is hence an extension to the static dial-a-ride problem such as the ones discussed in Ropke et al. (2007) and Parragh et al. (2012). (b) In the spirit of the recommendations in the survey by Agatz et al. (2012), this paper presents a Lagrangian decomposition heuristic that decomposes the problem based on the underlying structure rather than the geographical divisions. (c) To alleviate the computational burden which is critical for a real time implementation a second heuristic that is based on the concept of incremental cost is also presented.

The defining characteristics of the Shared-Taxi Problem are:

- Customers yield a constant revenue that is based on the pickup and dropoff zones.
- Only the customers that deem profitable are serviced.
- All passengers are picked up from their corresponding pickup locations and dropped off at their corresponding dropoff locations.
- The ride time of customers, be it passengers onboard or customers waiting to be serviced, must never exceed the values specified by the customers.
- The pickup and dropoff must always happen within the specified time windows.
- The number of passenger on any vehicle may never exceed the maximum vehicle capacity.

Accordingly, a mixed integer programming formulation for the shared-taxi problem is presented next.

### 3. Problem formulation

The shared-taxi problem considered in this paper is as follows. Given a directed graph  $G = (V, A)$  where  $V$  is the set of vertices and  $A$  the set of arcs, the set  $V$  is partitioned as  $V = V_1 \cup V_2 \cup V_3 \cup V_4$ , where  $V_1$  is the subset of vertices associated with the current taxi locations where  $v(t)$  denotes the vertex associated with taxi  $t$ .  $V_2$  is the subset of vertices associated with the dropoff locations of the onboard passengers.  $V_3$  is the subset of vertices associated with the pickup locations of the seekers. We note the distinction between onboard passengers which denote the passengers that have already been picked up by the taxis and are on route to their respective destinations, and the seekers which denote the passengers that have requested a taxi and hence are associated with a pickup and a dropoff location.  $V_4$  is hence the subset of vertices associated with the dropoff locations of the seekers.

Let  $T$  be the set of all taxis,  $O$  be the set of all onboard passengers where  $t(p)$  denotes the taxi that is currently transporting onboard passenger  $p$ ,  $s(p)$  is the index of the pickup node of passenger  $p$ , and  $f(p)$  is the index of the dropoff node. Also let  $S$  be the set of all the seekers, with  $P = O \cup S$ . Each taxi  $t \in T$  has a maximum vehicle capacity  $W^t$ . Each seeker is associated with an earliest pickup time  $U_p^e$  from the corresponding pickup location  $s(p)$ . Seekers and onboard passengers also have a latest dropoff time  $U_p^l$  at their corresponding dropoff locations  $f(p)$  and a maximum trip duration  $\hat{U}_p$  that is specific for each passenger  $p \in P$ . The maximum trip duration ensures that passengers are not assigned to undesirable long routes. A taxi going from node  $i$  to node  $j$  spends  $\bar{U}_{ij}$  units of time and incurs a cost  $C_{ij}$ . Each serviced customer  $p \in P$  yields a revenue  $R_p$  which is usually based on the pickup and dropoff zones. The objective is to maximize the total profit while respecting the pickup and dropoff times as well as maximum ride times. The decision variables are

$$x_{ij}^{pt} = \begin{cases} 1 & \text{if passenger } p \text{ traverses arc } (i,j) \text{ onboard of taxi } t, \\ 0 & \text{otherwise.} \end{cases}$$

$$y_{ij}^t = \begin{cases} 1 & \text{if taxi } t \text{ traverses arc } (i,j), \\ 0 & \text{otherwise.} \end{cases}$$

$$d_{pt} = \begin{cases} 1 & \text{if passenger } p \text{ is picked up by taxi } t, \\ 0 & \text{otherwise.} \end{cases}$$

$u_{it}$  is a non-negative variable representing the time at which vertex  $i$  is reached by taxi  $t$ .

The shared-taxi problem is then formulated as follows:

$$[\text{MVP}] : \max \sum_{p \in S} R_p \sum_{t \in T} d_{pt} - \sum_{i \in V} \sum_{j \in V} C_{ij} \sum_{t \in T} y_{ij}^t \quad (1)$$

$$\text{s.t.} \sum_{j \in V} x_{v(t(p))j}^{pt(p)} = 1, \quad \forall p \in O, \quad (2)$$

$$\sum_{j \in V} x_{jf(p)}^{pt(p)} = 1, \quad \forall p \in O, \quad (3)$$

$$\sum_{j \in V} x_{js(p)}^{pt} = \sum_{j \in V} x_{s(p)j}^{pt} - d_{pt}, \quad \forall p \in S, \quad \forall t \in T, \quad (4)$$

$$\sum_{j \in V} x_{jf(p)}^{pt} = \sum_{j \in V} x_{f(p)j}^{pt} + d_{pt}, \quad \forall p \in S, \quad \forall t \in T, \quad (5)$$

$$\sum_{j \in V} x_{ji}^{pt} = \sum_{j \in V} x_{ij}^{pt}, \quad \forall p \in P, \quad \forall t \in T, \quad \forall i \in V - V_1 - \{s(p)\} - \{f(p)\}, \quad (6)$$

$$\sum_{p \in P} x_{ij}^{pt} \leq W^t y_{ij}^t, \quad \forall t \in T, \quad \forall i \in V, \quad \forall j \in V, \quad (7)$$

$$\sum_{t \in T} d_{pt} \leq 1, \quad \forall p \in S, \quad (8)$$

$$\sum_{j \in V} y_{ji}^t \leq 1, \quad \forall t \in T, \quad \forall i \in V, \quad (9)$$

$$\sum_{j \in V} y_{ij}^t \leq 1, \quad \forall t \in T, \quad \forall i \in V, \quad (10)$$

$$\sum_{j \in V} y_{ij}^t \leq \sum_{j \in V} y_{ji}^t, \quad \forall t \in T, \quad \forall i \in \{V_3 \cup V_4\}, \quad (11)$$

$$u_{ij} - u_{ti} \geq \bar{U}_{ij} - M(1 - y_{ij}^t), \quad \forall t \in T, \quad \forall i \in V, \quad \forall j \in V, \quad (12)$$

$$u_{ts(p)} \geq U_{s(p)}^e d_{pt}, \quad \forall p \in S, \quad \forall t \in T, \quad (13)$$

$$u_{tf(p)} \leq U_{f(p)}^l d_{pt}, \quad \forall p \in P, \quad \forall t \in T, \quad (14)$$

$$u_{tf(p)} - u_{ts(p)} \leq \hat{U}_p + (1 - d_{pt})M, \quad \forall p \in P, \quad \forall t \in T, \quad (15)$$

$$x_{ij}^{pt} \in \{0, 1\}, y_{ij}^t \in \{0, 1\}, d_{pt} \in \{0, 1\}, u_{ti} \geq 0, \quad \forall i \in V, \quad \forall j \in V, \quad \forall p \in P, \quad \forall t \in T. \quad (16)$$

The objective function (1) maximizes the total profit. Constraints (2) ensure that onboard passengers leave their current location while Constraints (3) ensure that onboard passengers arrive to their corresponding destinations. Similarly, Constraints (4) and (5) ensure that seekers are picked-up from their pickup locations and dropped off at their corresponding destinations. Constraints (6) are for the conservation of flow. Constraints (7) ensure that the taxis' capacities are not violated while Constraints (8) ensure that each passenger is picked up by at most one taxi. We note that Constraints (8) do not force all the passengers to be picked up and thus some passengers might be rejected if they are deemed not profitable. Constraints (9) and (10) ensure that a taxi cannot reach a certain destination from two different sources simultaneously or head to two different destinations at once. Constraints (11) force the path continuity for the taxis. Constraints (12) define the time at which each location is reached by each taxi while Constraints (13) and (14) ensure that customers can only be picked up after the specified earliest pickup time and dropped off before the specified latest dropoff time. Constraints (15) ensures that customers do not spend more time on the road than the maximum ride time. Finally Constraints (16) enforce the binary and non-negativity conditions.

### 3.1. Comparing [MVP] to alternative formulations

As discussed in Section 2, mixed integer programming formulations have been proposed for the dial-a-ride-problem which resembles to the shared taxi problem that is of interest in this paper. Thus in order to evaluate formulation [MVP], we compare solving [MVP] to the DARP formulation which has been proposed in Cordeau (2006). We note that DARP as presented in Cordeau (2006) contains less binary variables as compared to [MVP]. However, it makes the following assumptions that are not included in [MVP].

- It requires that all seekers to be served.
- It minimizes the total route cost.
- It assumes that there are no onboard passengers.

Thus problem [MVP] is an extension of the DARP problem that is presented in Cordeau (2006) and as discussed next, the following restrictions are included in order to make the two problems equivalent for comparison.

- Constraints (8) are set to equality constraints  $\sum_{t \in T} d_{pt} = 1, \forall p \in S$  thus forcing all seekers to be served.
- Parameter  $R$  in the objective function (1) is set to 0 thus making the objective function equivalent to minimizing the total route cost.
- No onboard passengers are considered in the experimental results that are discussed next.
- The maximum route time constraint is ignored in the DARP model since it is not included in [MVP].

We thus compare solving [MVP] and the DARP formulation using CPLEX with default settings by using 12 instances of varying sizes. The instances are generated using the a2–16 instance that is considered in Cordeau (2006) which contains two taxis and 16 seekers. Particularly, each instance a2- $n$  is generated by considering the first  $n$  seekers of the a2–16 instance with all the corresponding data that include pickup and dropoff coordinates and the desired time windows. We note that the DARP model that is solved does not include the valid inequalities that are also discussed in Cordeau (2006) and can be generated within a branch-and-cut algorithm. A time limit of 2 h is set for all instances and the best lower and upper bounds (LB, UB) are reported for the instances that are not solved within the time limit. The results that are reported in Table 1 show that the [MVP] formulation achieved better results as compared to the DARP formulation on all instances except a2–5. Instances a2–6, a2–7, and a2–8 are solved to optimality using [MVP] in significantly less computational time than using the DARP model. Furthermore, a2–9 which could not be solved to optimality using the DARP model, was solved using [MVP]. For the remaining instances which were not solved to optimality, we notice that [MVP] obtained better bounds than DARP. We observed from the computational experiments that CPLEX is able to find a feasible solution earlier in the branch-and-bound tree when [MVP] model is used as compared to DARP. Finding feasible solutions earlier contributes to the solution of the problem in shorter amount of time.

In the following section, we detail the application of Lagrangian decomposition to [MVP].

#### 4. Solution methodology

##### 4.1. Lagrangian relaxation

Upon applying Lagrangian relaxation to (8) with multipliers  $\lambda_p$ , we get the following subproblem:

$$[SP] : Z_{SP} = \max \sum_{p \in S} (R_p - \lambda_p) \sum_{t \in T} d_{pt} - \sum_{i \in V} \sum_{j \in V} C_{ij} \sum_{t \in T} y_{ij}^t \left[ + \sum_{p \in S} \lambda_p \right]$$

s.t. (2)–(7), (9)–(16),

which decomposes into  $|T|$  single taxi problems that can be solved in parallel:

$$[TPC] : z_{TPC}^t = \max \sum_{p \in S} (R_p - \lambda_p) d_{pt} - \sum_{i \in V} \sum_{j \in V} C_{ij} x_{ij}^t \tag{17}$$

$$\text{s.t. } \sum_{j \in V} x_{v(t(p))j}^{pt(p)} = 1, \quad \forall p \in O, \tag{18}$$

$$\sum_{j \in V} x_{jf(p)}^{pt(p)} = 1, \quad \forall p \in O, \tag{19}$$

$$\sum_{j \in V} x_{js(p)}^{pt} = \sum_{j \in V} x_{s(p)j}^{pt} - d_{pt}, \quad \forall p \in S, \tag{20}$$

$$\sum_{j \in V} x_{jf(p)}^{pt} = \sum_{j \in V} x_{f(p)j}^{pt} + d_{pt}, \quad \forall p \in S, \tag{21}$$

$$\sum_{j \in V} x_{ji}^{pt} = \sum_{j \in V} x_{ij}^{pt}, \quad \forall p \in P, \quad \forall i \in V - V_1 - \{s(p)\} - \{f(p)\}, \tag{22}$$

$$\sum_{p \in P} x_{ij}^{pt} \leq W^t y_{ij}^t, \quad \forall i \in V, \quad \forall j \in V, \tag{23}$$

$$\sum_{j \in V} y_{ji}^t \leq 1, \quad \forall i \in V, \tag{24}$$

$$\sum_{j \in V} y_{ij}^t \leq 1, \quad \forall i \in V, \tag{25}$$

$$\sum_{j \in V} y_{ij}^t \leq \sum_{j \in V} y_{ji}^t, \quad \forall i \in \{V_3 \cup V_4\}, \tag{26}$$

$$u_{ij} - u_{ti} \geq \bar{U}_{ij} - M(1 - y_{ij}^t), \quad \forall i \in V, \quad \forall j \in V, \tag{27}$$

$$u_{ts(p)} \geq U_{s(p)}^e d_{pt}, \quad \forall p \in S, \tag{28}$$

**Table 1**  
Comparison between DARP and [MVP].

Instance	Cordeau (2006) DARP			[MVP]		
	Solution	Gap	CPU	Solution	Gap	CPU
a2-5	41.77	0%	0.08	41.77	0%	0.40
a2-6	81.60	0%	15.80	81.60	0%	4.35
a2-7	88.98	0%	387.44	88.97	0%	243.49
a2-8	102.29	0%	3652.38	102.29	0%	626.58
a2-9	(94.93, 104.23)	9%	>7200	104.23	0%	2041.39
a2-10	(91.62, 111.33)	18%	>7200	(103.02, 111.18)	8%	>7200
a2-11	(91.51, 114.04)	19%	>7200	(95.16, 115.51)	18%	>7200
a2-12	(87.15, -)	-	>7200	(104.32, 126.50)	18%	>7200
a2-13	(98.49, -)	-	>7200	(110.85, 134.82)	18%	>7200
a2-14	(99.63, -)	-	>7200	(108.02, 169.57)	36%	>7200
a2-15	(103.45, -)	-	>7200	(111.78, 201.33)	44%	>7200
a2-16	(103.32, -)	-	>7200	(118.11, -)	-	>7200

-: No valid bound found within the time limit of 7200 s.

$$u_{tf(p)} \leq U_{f(p)}^l d_{pt}, \quad \forall p \in P, \tag{29}$$

$$u_{tf(p)} - u_{ts(p)} \leq \widehat{U}_p + (1 - d_{pt})M, \quad \forall p \in P, \tag{30}$$

$$x_{ij}^{pt} \in \{0, 1\}, y_{ij}^t \in \{0, 1\}, d_{pt} \in \{0, 1\}, u_{ti} \geq 0, \quad \forall i \in V, \quad \forall j \in V, \quad \forall p \in P. \tag{31}$$

The solution of [SP] can be thought of as finding the maximum-profit vector:

$$\begin{bmatrix} (x_{ij}^{pt})^h \\ (y_{ij}^t)^h \\ (d_{pt})^h \\ (u_{ti})^h \end{bmatrix}_{h=1, \dots, H}$$

that satisfies constraints (18)–(31). Hence, [TPC] can be written as

$$Z_{TPC} = \max_{h=1, \dots, H} \left\{ \sum_{p \in S} (R_p - \lambda_p) \sum_{t \in T} (d_{pt})^h - \sum_{i \in V} \sum_{j \in V} C_{ij} \sum_{t \in T} (y_{ij}^t)^h \right\},$$

where  $H$  is the index set of feasible solutions to [TPC], which is finite as the feasible region of [TPC] is finite.

The best Lagrangian upper bound is

$$\min_{\lambda} \left\{ \sum_{p \in S} \lambda_p + \sum_{t \in T} z_{TPC}^t \right\}.$$

Defining

$$\theta_t = z_{TPC}^t$$

leads to the Lagrangian Master Problem:

$$\begin{aligned} \text{[MP]} : \min & \sum_{p \in S} \lambda_p + \sum_{t \in T} \theta_t \\ \text{s.t.} & \theta^t + \lambda_p \sum_{t \in T} (d_{pt})^h \geq \sum_{p \in S} R_p \sum_{t \in T} (d_{pt})^h - \sum_{i \in V} \sum_{j \in V} C_{ij} \sum_{t \in T} (y_{ij}^t)^h, \quad t \in T, \quad \forall h \in H, \\ & \lambda_p \geq 0 \quad \forall p \in S. \end{aligned}$$

The calculation of the Lagrangian bound is done iteratively where a relaxed version of [MP] is solved. The subproblem [TPC] corresponding to the Lagrange multipliers  $\lambda$  is then solved, leading to a new cut in [MP]. Because we are maximizing profit, the optimal solution of [TPC] provides an upper bound:  $\sum_{p \in S} \lambda_p + \sum_{t \in T} z_{TPC}^t$  on the optimal Lagrangian bound. Being a relaxed version, the optimal solution of [MP] provides a lower bound. The Lagrangian upper bound is reached and the iterative algorithm is stopped when the updated lower and the upper bounds coincide.

#### 4.2. Constructing a feasible solution

As discussed earlier, a Lagrangian decomposition is applied by relaxing Constraints (8). Thus, the subproblem solution  $\{\bar{x}_{ij}^{pt}, \bar{y}_{ij}^t, \bar{d}_{pt}^h, \bar{u}_{ti}^h\}$  is often infeasible to [MVP] as it might violate Constraints (8), i.e. the solution contains customers that are

served by more than one taxi. As typically done in the literature where a Lagrangian decomposition is used (Elhedhli and Xiaolong Hu, 2005; Elhedhli et al., 2011; Ghaddar et al., 2014), a feasible solution is constructed by slightly modifying the subproblem solution  $\{\bar{x}_{ij}^{pt}, \bar{y}_{ij}^t, \bar{d}_{pt}^h, \bar{u}_{ti}^h\}$ . We hence present an efficient heuristic to decide which of the multiple taxis should be assigned to each of the passengers that have been assigned multiple taxis. The core idea behind the heuristic is to evaluate all the possible taxi assignments and choose the taxi that results in a minimum cost. A pseudo-code description of the heuristic is presented below.

*Initialization:* Let  $L_s$  be the list of seekers that have been assigned multiple taxis and  $LT_s$  be the list of taxis that have been assigned to seeker  $s$ .  $TC_t$  is the total cost for taxi  $t$  according to the current solution  $\{\bar{x}_{ij}^{pt}, \bar{y}_{ij}^t, \bar{d}_{pt}^h, \bar{u}_{ti}^h\}$ . Let  $NC_{t(s)}$  be the new total cost for taxi  $t$  after eliminating  $s$  from its route and  $AC_{t(s)}$  be the additional cost that is incurred by taxi  $t$  if  $s$  is added to its route.

1. For  $s \in L_s$ 
    - 1.1 Set  $minAC$  to  $\infty$  and  $bestTaxi$  to null
    - 1.2 For  $t \in L_{t(s)}$ 
      - Calculate  $TC_t$
      - Calculate  $NC_{t(s)}$
      - Calculate  $AC_{t(s)} = TC_t - NC_{t(s)}$
      - If  $AC_{t(s)} < minAC$ , set  $minAC$  to  $AC_{t(s)}$  and  $bestTaxi$  to  $t$
    - 1.3 Add  $s$  to the route of taxi  $bestTaxi$  and remove it from the route of all other taxis.
- End For.

Since it accounts for onboard passengers, the proposed Lagrangian decomposition approach is applicable to the dynamic setting by updating the status of each taxi and resolving the problem every time a new passenger arrives. However, the main challenge is due to the required computational time that may prevent its applicability in a real time implementation. In the following section, we present a more advanced heuristic based on the concept of incremental costs. Furthermore, the proposed incremental cost heuristic accounts for customer requests being initiated at different points in time and hence is also applicable to dynamic systems. Furthermore since the Lagrangian decomposition yields upper and lower bounds, the solutions of the Lagrangian decomposition approach serve as a benchmark to assess the quality of the Incremental Cost Heuristic that is proposed in the following section.

## 5. Heuristic for dynamic system

As presented earlier, the location of the pool of seekers is assumed to be known beforehand and hence the taxi routing is performed accordingly. A typical approach for including the dynamic nature of the system where seekers appear in real time is to update the problem data and re-optimize the problem whenever a new request arrives. Although such an approach is possible by fixing the partial routes that have been already completed in problem [MVP] and re-optimizing the resulting problem, an approach that takes into account the dynamic nature of the problem is typically preferred mainly to reduce the computational burden which is critical for a real time implementation.

The survey of Berbeglia et al. (2010) summarizes the recent developments in dynamic vehicle routing. The traditional approach of updating the problem data and re-optimizing was first used in Psaraftis (1980). Specialized heuristics for the dynamic problem were then discussed in Madsen et al. (1995) which presented an insertion based heuristic for a real-life problem from the Copenhagen Fire-Fighting Service. Parallel tabu search heuristics were also presented in Attanasio et al. (2004) while Coslovich et al. (2006) presented a two-phase insertion based algorithm for the dynamic problem with time windows. A two-phase algorithm is also discussed in Beaudry et al. (2010) where an insertion based algorithm is first applied followed by a tabu search heuristic in a second phase. Extensions of the dynamic case to include stochastic service and travel times were discussed in Xiang et al. (2008). In Schilde et al. (2011), four different metaheuristics were introduced for the dynamic stochastic dial-a-ride problem. The paper also showed that taking into account stochastic information provides better routing solutions. Most recently, Berbeglia et al. (2012) presented a new approach which combines tabu search and constraint programming to benefit from the advantage of tabu search of easily inserting new requests into the current solution while proving the infeasibility of some problems through constraint programming.

In this section, we present the incremental cost heuristic (ICH) that is motivated by the Lagrangian approach which decomposes the problem into  $t$  subproblems that are solved independently. The core idea is to solve a cost minimization problem for each taxi  $t$  whenever a new request arrives in order to calculate the additional cost, i.e. the incremental cost, that will be incurred by each taxi if the new request is included in its route. Such an approach offers two main advantages: First, the new request is assigned to the taxi that has the lowest incremental cost if the that cost does not exceed the expected return. Second, in the latter case, from a practical point of view a service charge quote that exceeds the incremental cost can be offered to the passenger rather than rejecting the passenger's request. We note that the typical approach is to consider the incoming requests on a first-come first-serve basis however it might be beneficial to queue the requests and process them in a different order. Such an approach can be considered as a hybrid between dynamic and static systems and is worth studying in a future research.

### 5.1. Calculating the incremental cost

As discussed, the incremental cost approach calculates for each arriving passenger the additional cost that is incurred by each taxi if it is to be included in its route. Hence calculating the incremental cost for a particular taxi is equivalent to finding the minimum cost route that includes the additional passenger. Hence this problem consists of one taxi and multiple passengers where the passengers consist of a set  $O$  of customers that are already onboard the taxi, the passengers that have been assigned to that taxi however not yet picked up, and a set  $S$  of newly arriving passengers. Since in general the number of passengers not yet picked up and the number of newly arriving passengers can be arbitrary large, an enumerative approach that evaluates the order at which passengers are picked and dropped off is not practical and thus formulating the assignment as an optimization problem is more appropriate. We note that the onboard passengers have dropoff points only while each of the remaining passengers, i.e. the seekers, have a pickup location and a dropoff location. Furthermore, similar to problem [MVP], we assume that every passenger  $p$  has a maximum trip duration  $\hat{U}_p$  time, an earliest pickup time  $U_p^e$  (if not onboard), and a latest dropoff time  $U_p^l$ . Like before, given a directed graph  $G(V, A)$  where  $V$  is the set of vertices and  $A$  the set of arcs, the set  $V$  is partitioned as  $V = V_1 \cup V_2 \cup V_3 \cup V_4$  where  $V_1 = \{0\}$  is the vertex associated with the current taxi location.  $V_2$  is the subset of vertices associated with the drop-off locations of the onboard passengers.  $V_3$  is the subset of vertices associated with the pickup locations of the seekers and  $V_4$  is the subset of vertices associated with their dropoff locations. The decision variables are as follows

$$x_{ij}^p = \begin{cases} 1 & \text{if passenger } p \text{ traverses arc } (i,j), \\ 0 & \text{otherwise.} \end{cases}$$

$$y_{ij} = \begin{cases} 1 & \text{if the taxi traverses arc } (i,j), \\ 0 & \text{otherwise.} \end{cases}$$

$u_{it}$  is a non-negative variable representing the time at which vertex  $i$  is reached by the taxi.

The incremental cost problem is hence formulated as follows

$$[\text{SVP}] : \min \sum_{i \in V} \sum_{j \in V} C_{ij} y_{ij} \quad (32)$$

$$\text{s.t.} \sum_{j \in V} x_{0j}^p = 1, \quad p \in O, \quad (33)$$

$$\sum_{j \in V} x_{jf(p)}^p = 1, \quad p \in O, \quad (34)$$

$$\sum_{j \in V} x_{js(p)}^p = \sum_{j \in V} x_{s(p)j}^p - 1, \quad \forall p \in S, \quad (35)$$

$$\sum_{j \in V} x_{jf(p)}^p = \sum_{j \in V} x_{f(p)j}^p + 1, \quad \forall p \in S, \quad (36)$$

$$\sum_{j \in V} x_{ji}^p = \sum_{j \in V} x_{ij}^p, \quad \forall p \in P, \quad \forall i \in V - V_1 - \{s(p)\} - \{f(p)\}, \quad (37)$$

$$\sum_{p \in P} x_{ij}^p \leq W y_{ij}, \quad \forall i \in V, \quad \forall j \in V, \quad (38)$$

$$\sum_{i \in V} y_{ij} = 1, \quad j \in V_3, \quad (39)$$

$$\sum_{j \in V} y_{ji} \leq 1, \quad i \in \{V - 0\}, \quad (40)$$

$$\sum_{j \in V} y_{ij} \leq 1, \quad i \in V, \quad (41)$$

$$u_j - u_i \geq \bar{U}_{ij} - M(1 - y_{ij}), \quad i \in V, j \in V, \quad (42)$$

$$u_i \geq U_i^e, \quad i \in V_3, \quad (43)$$

$$u_j \leq U_j^l, \quad j \in \{V_2 \cup V_4\}, \quad (44)$$

$$u_{f(p)} - u_{s(p)} \leq \hat{U}_p, \quad \forall p \in P, \quad (45)$$

$$x_{ij}^c \in \{0, 1\}, y_{ij} \in \{0, 1\}, u_i \geq 0. \quad (46)$$

The objective function (32) minimizes the route cost. Constraints (33) ensure that onboard passengers leave their current location while constraints (34) that they are dropped-off at their corresponding destinations. Constraints (35)–(37) are the conservation of flow constraints. Constraint (38) ensure that the capacity of the taxi is not violated. Constraints (39) ensure that every seeker is picked-up by exactly a single taxi. Constraints (40)–(41) ensure that the taxi can reach and leave every node at most once. Constraints (42) define the time at which each location is reached and eliminates subtours while

(44)–(45) enforce the earliest pickup time, the latest dropoff time, and the maximum ride time constraints. Finally Constraints (46) enforce the binary and non-negativity conditions.

Given the route cost  $C_{old}$  for taxi  $t$  before assigning the newly arriving seeker and the minimum cost  $C_{new} = \sum_{i \in V} \sum_{j \in V} C_{ij} y_{ij}^*$  for the new route if the seeker is assigned, the incremental cost for taxi  $t$  is hence  $IC = C_{new} - C_{old}$ . As such, each seeker is assigned to the taxi with the minimum incremental cost. In the case of a fixed taxi fare system, the newly arriving seeker may be either rejected if the minimum incremental cost exceeds the taxi fare or asked to pay an extra charge to make the service profitable. The extra charge is thus only imposed on the arriving seeker and not on the other passengers that have been already assigned to a taxi. The new fare is still expected to be significantly lower than the typical fare for a non shared taxi.

### 5.1.1. Detecting infeasibility

Solving problem [SVP] repeatedly is the main time consuming step of the incremental cost heuristic. Furthermore since problem [SVP] is solved for all the taxis, the computational results showed that in a number of cases problem [SVP] is infeasible and hence the computational performance can be improved by detecting the infeasible cases before solving the problem. This section describes simple conditions that have been implemented in ICH to improve the performance.

The core reason behind the infeasible cases of [SVP] is due to the fact that forcing the taxi to serve a customer might violate the time window constraints resulting in an infeasible [SVP] problem. As such, to detect infeasible instances, we make the reasonable assumption that the direct trip time  $\bar{U}_{ij}$  between vertex  $i$  to vertex  $j$  is less than all other paths connecting these two vertices (i.e. the direct trip is the fastest way to travel). Hence, an infeasible problem occurs when there exists a passenger where the direct route exceeds the latest dropoff time. Therefore given a passenger  $p$ , if

$$\bar{U}_{0s(p)} + \bar{U}_{s(p)f(p)} > U_p^l, \quad (47)$$

or

$$\bar{U}_{0s(p)} + \bar{U}_{s(p)f(p)} > \hat{U}_p, \quad (48)$$

then the problem is infeasible. Conditions (47) and (48) state that if the travel time associated with the path that links the current taxi position to the pickup location of passenger  $p$  and then to the dropoff location of  $p$  exceeds either the latest dropoff time or the maximum route time, then the problem is infeasible.

Conditions (47) and (48) can be further tightened when considering onboard passengers and hence additional conditions are generated. This is based on finding an onboard passenger  $n$  that must be dropped off before picking up any of the other passengers. An onboard passenger  $n$  must be dropped off immediately if  $U_{0s(n)}^l < \bar{U}_{0j} + \bar{U}_{js(n)}$ ,  $\forall j \in V - \{0, s(n)\}$  or  $\hat{U}_n < \bar{U}_{0j} + \bar{U}_{js(n)}$ ,  $\forall j \in V - \{0, s(n)\}$  which states that the only way to dropoff passenger  $n$  within the time window constraints is to go from the current node to the destination node  $s(n)$  directly. Hence, given a passenger  $p$ , if

$$\bar{U}_{0s(n)} + \bar{U}_{s(n)s(p)} + \bar{U}_{s(p)f(p)} > U_p^l, \quad (49)$$

or

$$\bar{U}_{0s(n)} + \bar{U}_{s(n)s(p)} + \bar{U}_{s(p)f(p)} > \hat{U}_p, \quad (50)$$

then problem [SVP] is infeasible. Conditions (49) and (50) state that if the travel time associated with servicing passenger  $p$  after passenger  $n$  (which must be served first) violates the time window constraints of  $p$  then [SVP] is infeasible.

Note that additional infeasibility conditions can be determined by considering additional complicated combinations of routes however we restricted the implementation to the stated conditions.

## 6. Computational results

This sections presents extensive computational results assessing the performance of the proposed algorithms. The proposed Lagrangian decomposition algorithm and heuristics were coded in C using CPLEX 12.3 callable libraries. The computations are carried out on a Lenovo C30 workstation with 2 GHz CPU. To assess the performance of the proposed approaches, we compare the results against solving problem [MVP] in CPLEX with default solver settings. To evaluate the computational performance, we first present results on a set of instances that are generated randomly. Then to clearly show the applicability of the proposed methods in practice, we conduct a simulation that mimics a real life scenario and involves 20 taxis for a two hours operation over the city of Manhattan.

### 6.1. Static vs dynamic system

As highlighted throughout the paper, we distinguish between the static and the dynamic shared taxi system. The static system assumes that all the data regarding the taxis (taxi locations and capacity) and the passengers (pickup locations and time windows) are all available and the shared taxi problem is solved once to find the optimal assignment. In the dynamic setting, taxi seekers arrive in real time and thus the shared taxi problem is solved every time a new seeker arrives. While not taking into account the computational time requirements for practical implementation, the Lagrangian decomposition

approach is used in the dynamic framework by resolving the shared taxi problem each time a new seeker arrives. Similarly, ICH is applied in the static framework by processing the seeker requests in the order that they are listed. We demonstrate both setups in the computational experiments that are presented next. Computational results discussing application of the Lagrangian decomposition and ICH in the static framework are presented in Section 6.2. Finally, Section 6.3 discusses the application of Lagrangian decomposition and ICH in the dynamic setting.

## 6.2. Static framework

Six sets of instances with varying characteristics were generated. The instances are inspired from the data that was used for benchmarking in Cordeau (2006). The sets have varying number of taxis, seekers, and onboard passengers. The coordinates of the taxi locations as well as the pick-up and drop-off locations were randomly generated in the square  $[-10,10] \times [-10,10]$  according to a uniform distribution. The travel time  $\bar{U}_{ij}$  for arc  $(i,j)$  is assumed to be the Euclidean distance between the two coordinates and the cost is assumed to be  $C_{ij} = 0.1 * \bar{U}_{ij}$ . We also assume a revenue per passenger  $R_p = 2$  and a vehicle capacity of 3. The instances are divided into two classes: Sets A, B, and C are instances with relaxed time windows while Sets D, E, and F are instances with tight time windows. Assuming the starting time is  $t = 0$ , the earliest pickup times  $U_p^e \quad \forall p \in S$  for the seekers are randomly generated from the uniform distribution  $\mathcal{U}(0, 5)$ . The latest dropoff time  $U_p^l$  for every passenger  $p \in P$  is set to  $U_p^l = U_p^e + \bar{U}_{s(p)f(p)} + \mathcal{U}(5, 30)$  for the instances with relaxed time windows and  $U_p^l = U_p^e + \bar{U}_{s(p)f(p)} + \mathcal{U}(0, 15)$  for the instances with tight time windows where  $\bar{U}_{s(p)f(p)}$  is the travel time from the current location of passenger  $p$  to the corresponding dropoff location. The name of each problem indicates the size of the considered problem. For instance,  $p.n.m.k.t.i$  denotes a problem with  $n$  taxis,  $m$  onboard passengers in each taxi, and  $k$  seekers. Also  $t$  is set to  $R$  to indicate relaxed time windows and  $T$  to indicate tight time windows. For each problem size, we generate two instances and denote by  $i = 1$  the first instance and  $i = 2$  the second.

### 6.2.1. Results

Extensive computational testing is conducted on instances with various sizes. The results are displayed in Tables 2 and 3 and are summarized in Table 4. The following information is reported:

Name :	Name of the instance.
CPLEX solution :	(LB, UB) are the lower and upper bounds for the best solution found by CPLEX.
CPLEX gap :	Percentage gap for the CPLEX solution.
CPU time :	Total computational time in seconds.
LR bound :	Lagrangian bound which gives an upper bound on the optimal objective.
LR CPU time :	Total computational time in seconds for the Lagrangian decomposition.
Simple heuristic solution :	Feasible solution found by the simple heuristic which gives a lower bound on the optimal objective.
Simple heuristic gap :	Percentage gap between the feasible solution found by the simple heuristic and the Lagrangian bound.
ICH Heuristic solution :	Feasible solution found by the ICH heuristic which gives a lower bound on the optimal objective.
ICH Heuristic gap :	Percentage gap between the feasible solution found by ICH and the Lagrangian bound.
ICH CPU time :	Total computational time in seconds for ICH.

The results show that CPLEX is only capable of solving small instances to optimality within the timelimit of 2 h. Furthermore, as the problems get larger and harder CPLEX fails to find a feasible solution. These are the instances that are marked by (–) for the solution gap which cannot be calculated since CPLEX failed to find a lower bound (feasible solution). Solving the sub-problems repeatedly constitutes the main computationally demanding part of the Lagrangian decomposition algorithm and hence as expected for the small instances, CPLEX performs better than the Lagrangian decomposition. Within the timelimit of 2 h, CPLEX found better upper bounds than the Lagrangian decomposition algorithm for the small instances however for the larger instances the Lagrangian bound is significantly stronger. Furthermore, the Lagrangian decomposition approach provides an upper bound in a shorter computational time than CPLEX.

As typically observed in the literature, the Lagrangian decomposition typically yields a solution that is not feasible to the original problem. Hence to find lower bounds, two heuristics are proposed. Based on the Lagrangian solution, the simple heuristic that was described in Section 4.2 found a feasible solution for all the instances. We note that the computational time for the simple heuristic is not shown in the tables since the computational time was less than 0.1 s for all the instances. The feasible solution that is found by the simple heuristic was optimal for the small instances ( $p\_2\_1\_2\_(R,T)$ ,  $p\_4\_1\_4\_(R,T)$ ,

**Table 2**  
Computational results for instances with relaxed time windows.

Set	Name	CPLEX			Lagrangian decomposition		Simple heuristic		ICH heuristic		
		Solution	Gap	CPU time	LR bound	CPU time	Solution	Gap	Solution	Gap	CPU time
A	p_2_1_2_R_1	(2.0, 2.0)	0%	0.22	2	0.1	2	0%	1.7	15%	0.1
	p_2_1_2_R_2	(4.9, 4.9)	0%	0.02	4.9	0.1	4.9	0%	4.4	10%	0.1
	p_4_1_4_R_1	(7.6, 7.6)	0%	6.6	7.6	1.9	7.6	0%	7.6	0%	0.2
	p_4_1_4_R_2	(7.7, 7.7)	0%	41.62	7.7	12.7	7.7	0%	7.7	0%	0.2
	p_6_1_6_R_1	(11.2, 11.2)	0%	5295.9	11.2	1367	11.2	0%	9.4	16%	0.3
	p_6_1_6_R_2	(12.1, 12.1)	0%	402.37	12.1	131.6	12.1	0%	12.1	0%	0.4
	p_8_1_8_R_1	(17.8, 19.3)	8%	>7200	18.2	7157.5	17.8	2%	17.5	4%	11.3
	p_8_1_8_R_2	(17.3, 18.8)	8%	>7200	21.4	197.2	15.8	26%	17.3	19%	10.7
	p_10_1_10_R_1	(20.4, 22.5)	9%	>7200	29.2	253.7	19.6	33%	20.2	31%	3.2
	p_10_1_10_R_2	(20, 23)	13%	>7200	28.3	602.5	19.2	32%	19.6	31%	3.8
	p_12_1_12_R_1	(25.2, 28.1)	11%	>7200	37.2	1441.4	25.9	30%	24.3	35%	7.8
	p_12_1_12_R_2	(26.3, 28.4)	8%	>7200	34.3	1015.2	24.6	28%	25.9	24%	10.2
	p_14_1_14_R_1	(31.3, 35.9)	13%	>7200	40.9	2425.7	30.3	26%	31.2	24%	21.7
	p_14_1_14_R_2	(17.4, 30.7)	43%	>7200	84.2	2644.7	22.9	73%	25.1	70%	25.9
	p_16_1_16_R_1	(34.4, 38.3)	10%	>7200	65	>7200	25.6	61%	35.4	46%	34.7
	p_16_1_16_R_2	(33.5, 40)	16%	>7200	48.6	>7200	36.4	25%	34.1	30%	33.8
	p_18_1_18_R_1	(-, 43.5)	-	>7200	145.2	>7200	32.2	78%	38.2	74%	13.3
	p_18_1_18_R_2	(-, 43.6)	-	>7200	136.2	>7200	32.6	76%	38.2	72%	57.3
	p_20_1_20_R_1	(-, 49.9)	-	>7200	175.1	>7200	32.1	82%	44.3	75%	50.5
	p_20_1_20_R_2	(-, 54.6)	-	>7200	201.2	>7200	30.5	85%	48.9	76%	60.7
Average			27%					33%		33%	17.3
B	p_2_1_4_R_1	(3.3, 3.3)	0%	44.6	3.5	6.4	3.3	6%	3.3	6%	0
	p_2_1_4_R_2	(3.8, 3.8)	0%	1.83	3.8	4	3.8	0%	3.8	0%	0.1
	p_4_1_8_R_1	(12.6, 14.3)	12%	>7200	21	59.3	8.4	60%	10.8	49%	13.6
	p_4_1_8_R_2	(10.2, 12.7)	20%	>7200	16.9	63.2	3.2	81%	9.2	46%	12.9
	p_6_1_12_R_1	(16.6, 21.4)	22%	>7200	31.7	425.7	14.5	54%	14.1	56%	3.1
	p_6_1_12_R_2	(19.7, 22.6)	13%	>7200	40.9	624.9	13.2	68%	18	56%	59.4
	p_8_1_16_R_1	(21.5, 30.5)	30%	>7200	64.2	2138.2	13.1	80%	22.5	65%	29.3
	p_8_1_16_R_2	(21.8, 31.8)	31%	>7200	70.1	3565.2	20	71%	20.7	70%	31.2
	p_10_1_20_R_1	(17.6, 38.5)	54%	>7200	93.8	4215.8	8.9	91%	27.4	71%	47.7
	p_10_1_20_R_2	(17.4, 38.4)	55%	>7200	103.9	5480.9	12.1	88%	29.2	72%	43.6
	p_12_1_24_R_1	(-, 49.1)	-	>7200	147.3	>7200	18	88%	34.9	76%	130.5
	p_12_1_24_R_2	(-, 52)	-	>7200	171.7	6584.8	12.4	93%	41.7	76%	119.7
	p_14_1_28_R_1	(-, 812.0)	-	>7200	201.3	>7200	16.8	92%	41.5	79%	170.6
	p_14_1_28_R_2	(-, 812)	-	>7200	208.7	>7200	14	93%	45	78%	145.5
	p_16_1_32_R_1	(-, 1056.0)	-	>7200	194.9	>7200	14.4	93%	50.8	74%	213.1
	p_16_1_32_R_2	(-, 1056)	-	>7200	156.5	>7200	15.8	90%	46.7	70%	219.7
	p_18_1_36_R_1	(-, 1332.0)	-	>7200	164.4	>7200	17.8	89%	54.9	67%	117.1
	p_18_1_36_R_2	(-, 1332)	-	>7200	119.7	>7200	19.3	84%	64.3	46%	886.5
	p_20_1_40_R_1	(-, 1640.0)	-	>7200	136.1	>7200	15.7	88%	69.3	49%	294.6
	p_20_1_40_R_2	(-, 1640)	-	>7200	130.3	>7200	21.2	84%	69.5	47%	310.3
Average			62%					75%		58%	142.4
C	p_2_1_6_R_1	(4.0, 4.0)	0%	4409.2	9.7	12.3	2	79%	4	59%	0.1
	p_2_1_6_R_2	(3, 4.1)	27%	>7200	7.1	10.4	1.7	76%	3	58%	0.1
	p_4_1_12_R_1	(12.0, 19.2)	38%	>7200	28.5	175.4	8	72%	8.3	71%	9.5
	p_4_1_12_R_2	(11.3, 19.3)	42%	>7200	27.8	220.5	4.9	82%	9.7	65%	20.2
	p_6_1_18_R_1	(24.3, 32.3)	25%	>7200	59.3	3577.4	9.5	84%	20.7	65%	177
	p_6_1_18_R_2	(18.8, 33.2)	43%	>7200	62.6	3281.6	11.9	81%	20.2	68%	74.9
	p_8_1_24_R_1	(-, 44.8)	-	>7200	99.1	>7200	11.2	89%	32.1	68%	132.8
	p_8_1_24_R_2	(-, 45.1)	-	>7200	98.1	>7200	11.8	88%	23.6	76%	83.1
	p_10_1_30_R_1	(-, 60.8)	-	>7200	151.8	>7200	11.5	92%	27.8	82%	165
	p_10_1_30_R_2	(-, 620)	-	>7200	164	>7200	11.3	93%	32.1	80%	343.8
	p_12_1_36_R_1	(-, 888.0)	-	>7200	113.9	>7200	11.8	90%	41.2	64%	219.3
	p_12_1_36_R_2	(-, 888)	-	>7200	106.9	>7200	12.2	89%	45.9	57%	247.4
	p_14_1_42_R_1	(-, 1204.0)	-	>7200	129.6	>7200	12	91%	59.5	54%	1181.6
	p_14_1_42_R_2	(-, 1204)	-	>7200	109.9	>7200	12	89%	49	55%	463.9
	p_16_1_48_R_1	(-, 1568.0)	-	>7200	80	>7200	17.9	78%	56.1	30%	691.3
	p_16_1_48_R_2	(-, 1568)	-	>7200	80	>7200	11.7	85%	46.8	42%	180.9
	p_18_1_54_R_1	(-, 1980.0)	-	>7200	90	>7200	18.1	80%	69.4	23%	1039.3
	p_18_1_54_R_2	(-, 1980)	-	>7200	90	>7200	15.1	83%	75.2	16%	770.7
	p_20_1_60_R_1	(-, 2440.0)	-	>7200	100	>7200	25.3	75%	76.6	23%	3303.6
	p_20_1_60_R_2	(-, 2440)	-	>7200	100	>7200	20.3	80%	83.4	17%	652.3
Average			79%					84%		54%	488.8

-: No valid lower bound found within the time limit of 7200 s. Gap assumed to be 100% when calculating the average.

**Table 3**  
Computational results for instances with tight time windows.

Set	Name	CPLEX			Lagrangian decomposition		Simple heuristic		ICH heuristic		
		Solution	Gap	CPU time	LR bound	CPU time	Solution	Gap	Solution	Gap	CPU time
D	p_2_1_2_T_1	(4.6, 4.6)	0%	0	4.6	0.1	4.6	0%	4.6	0%	0.1
	p_2_1_2_T_2	(2.2, 2.2)	0%	3	2.2	0.1	2.2	0%	2.2	0%	0
	p_4_1_4_T_1	(8.2, 8.2)	0%	1.7	8.2	0.5	8.2	0%	8.2	0%	0.1
	p_4_1_4_T_2	(3.6, 3.6)	0%	17.4	3.6	0.1	3.6	0%	3.6	0%	0
	p_6_1_6_T_1	(9.4, 12.2)	23%	>7200	9.4	27.7	9.4	0%	9.4	0%	0.1
	p_6_1_6_T_2	(8.3, 11.5)	28%	>7200	8.3	15.3	8.3	0%	8.3	0%	0.1
	p_8_1_8_T_1	(14.0, 17.8)	21%	>7200	19.8	81.2	13.6	31%	12.4	37%	0.3
	p_8_1_8_T_2	(12.4, 18.1)	31%	>7200	17.3	90.3	10.5	39%	11.1	36%	0.2
	p_10_1_10_T_1	(20.4, 24.8)	18%	>7200	27.7	331.1	20.4	26%	18.8	32%	0.5
	p_10_1_10_T_2	(18.4, 23.4)	21%	>7200	25.8	212.9	16.7	35%	18.7	28%	0.4
	p_12_1_12_T_1	(20.7, 27.7)	25%	>7200	29.6	381.8	20.8	30%	20.4	31%	0.5
	p_12_1_12_T_2	(23.6, 29.7)	20%	>7200	28.9	339.3	21.5	26%	23.9	17%	0.5
	p_14_1_14_T_1	(29.0, 39.1)	26%	>7200	37.2	315.2	31.1	16%	30.6	18%	3.6
	p_14_1_14_T_2	(29, 39.1)	26%	>7200	37.2	315.5	31.1	16%	30.6	22%	3.6
	p_16_1_16_T_1	(23.5, 42.0)	44%	>7200	36.3	1743.8	28	23%	30.7	15%	1.1
	p_16_1_16_T_2	(17.5, 37.7)	53%	>7200	34	3879.2	18.6	45%	24.2	29%	1.7
	p_18_1_18_T_1	(-, 48.2)	-	>7200	47.9	6103.8	36.2	24%	38.4	20%	11.6
	p_18_1_18_T_2	(-, 49.6)	-	>7200	147.2	>7200	33.9	77%	38.4	74%	1.4
	p_20_1_20_T_1	(-, 53.4)	-	>7200	169.3	>7200	36.4	78%	39.3	77%	1.4
	p_20_1_20_T_2	(-, 48.7)	-	>7200	176.2	>7200	27.8	84%	35	80%	7.1
Average			37%				28%		26%	1.7	
E	p_2_1_4_T_1	(1.2, 1.2)	0%	2.7	1.2	0.9	1.2	0%	1.2	0%	0.1
	p_2_1_4_T_2	(3.8, 3.8)	0%	14.67	3.8	2.6	3.8	0%	3.7	3%	0.1
	p_4_1_8_T_1	(2.4, 12.6)	81%	>7200	18.8	50.6	2.4	87%	2.4	87%	0.1
	p_4_1_8_T_2	(6.8, 13.9)	51%	>7200	18	42.5	5.4	70%	6.3	65%	0.2
	p_6_1_12_T_1	(9.4, 23.5)	60%	>7200	42.5	328.9	8.5	80%	9.2	78%	6.2
	p_6_1_12_T_2	(13.4, 23)	42%	>7200	37.7	443.7	8.4	78%	13.5	64%	12.9
	p_8_1_16_T_1	(10.4, 33.5)	69%	>7200	68.4	1568.9	10.4	85%	15.8	77%	8.3
	p_8_1_16_T_2	(12.8, 31.7)	60%	>7200	67.7	1283.5	12.5	82%	17.3	74%	4.2
	p_10_1_20_T_1	(-, 41.2)	-	>7200	104.4	4354.8	13.7	87%	17.5	83%	7.1
	p_10_1_20_T_2	(-, 39.7)	-	>7200	102.1	4145.3	13.4	87%	23.9	77%	11.7
	p_12_1_24_T_1	(-, 49.3)	-	>7200	136.2	>7200	9.8	93%	21.4	84%	1.2
	p_12_1_24_T_2	(-, 50.9)	-	>7200	141.4	>7200	19.3	86%	21.4	85%	1.5
	p_14_1_28_T_1	(-, 812.0)	-	>7200	210.2	>7200	10.9	95%	38.9	81%	35.5
	p_14_1_28_T_2	(-, 812)	-	>7200	211	>7200	23.1	89%	28.3	87%	7.1
	p_16_1_32_T_1	(-, 1056.0)	-	>7200	246.6	>7200	19.8	92%	33.6	86%	2.5
	p_16_1_32_T_2	(-, 1056)	-	>7200	202.6	>7200	16.9	92%	37.7	81%	25
	p_18_1_36_T_1	(-, 1332.0)	-	>7200	120.3	>7200	17.7	85%	45.9	62%	25.2
	p_18_1_36_T_2	(-, 1332)	-	>7200	102.1	>7200	10.8	89%	36.9	64%	21.9
	p_20_1_40_T_1	(-, 1640.0)	-	>7200	181	>7200	20.1	89%	43.6	76%	3.8
	p_20_1_40_T_2	(-, 1640)	-	>7200	131.7	>7200	17.5	87%	40.3	69%	20.5
Average			78%				78%		69%	9.8	
F	p_2_1_6_T_1	(2.1, 2.1)	0%	1457.3	9.8	7.2	2.1	79%	2.1	79%	0.1
	p_2_1_6_T_2	(0.2, 5.5)	97%	>7200	9	19.5	0.2	98%	0.2	98%	0
	p_4_1_12_T_1	(5.7, 18.4)	69%	>7200	5.7	265.7	5.7	0%	4.5	21%	0.2
	p_4_1_12_T_2	(8.2, 18.6)	56%	>7200	26.1	347.4	4.3	84%	7.3	72%	0.3
	p_6_1_18_T_1	(11.9, 32.6)	64%	>7200	60.6	3783.9	6.6	89%	14.9	75%	0.5
	p_6_1_18_T_2	(10.9, 31.7)	66%	>7200	52.6	2557.5	8.6	84%	12.1	77%	0.8
	p_8_1_24_T_1	(-, 45.8)	-	>7200	107.8	>7200	10.5	90%	21.9	80%	10.5
	p_8_1_24_T_2	(-, 46.4)	-	>7200	107.7	>7200	8.4	92%	21.5	80%	16.3
	p_10_1_30_T_1	(-, 620.0)	-	>7200	153.4	>7200	11	93%	24.3	84%	14.6
	p_10_1_30_T_2	(-, 620)	-	>7200	172.5	>7200	10.6	94%	24.7	86%	26.2
	p_12_1_36_T_1	(-, 888.0)	-	>7200	103.7	>7200	9.2	91%	22.3	78%	5.3
	p_12_1_36_T_2	(-, 888)	-	>7200	125.2	>7200	11.3	91%	28.7	77%	32.9
	p_14_1_42_T_1	(-, 1204.0)	-	>7200	107.2	>7200	12.3	89%	35.3	67%	79.6
	p_14_1_42_T_2	(-, 1204)	-	>7200	126.6	>7200	10.7	92%	32.7	74%	9.5
	p_16_1_48_T_1	(-, 1568.0)	-	>7200	80	>7200	15.9	80%	37.4	53%	85.9
	p_16_1_48_T_2	(-, 1568)	-	>7200	80	>7200	18.8	77%	48.3	40%	19.8
	p_18_1_54_T_1	(-, 1980.0)	-	>7200	90	>7200	17.8	80%	50.2	44%	56.1
	p_18_1_54_T_2	(-, 1980)	-	>7200	90	>7200	18	80%	45.2	50%	82.8
	p_20_1_60_T_1	(-, 2440.0)	-	>7200	100	>7200	20.4	80%	50.1	50%	63
	p_20_1_60_T_2	(-, 2440)	-	>7200	100	>7200	16.6	83%	59.3	41%	169
Average			88%				82%		66%	33.7	

-: No valid lower bound found within the time limit of 7200 s. Gap assumed to be 100% when calculating the average.

**Table 4**  
Computational results summary.

Set	Time windows	Passengers ratio	CPLEX		Lagrangian decomposition		ICH	
			Problems solved within timelimit	Average GAP	Problems solved within timelimit	Average GAP	Problems solved within timelimit	Average GAP
A	Relaxed	Low	6/20	27%	14/20	33%	20/20	33%
B	Relaxed	Medium	2/20	62%	11/20	75%	20/20	58%
C	Relaxed	High	1/20	79%	6/20	84%	20/20	54%
D	Tight	Low	4/20	37%	17/20	28%	20/20	26%
E	Tight	Medium	2/20	78%	10/20	78%	20/20	69%
F	Tight	High	1/20	88%	6/20	82%	20/20	66%

**Table 5**  
Simulation 1 – results summary.

	Lagrangian decomposition	ICH
Average computational time (sec.)	86.83	1.68
Minimum computational time (sec.)	0.01	0.02
Maximum computational time (sec.)	1219.58	9.57
Number of rejected passengers	0	3
Number of taxis used	18	8
Total profit	71.5	63.8
Average % deviation from direct route	20%	26%
Minimum % deviation from direct route	0%	0%
Maximum % deviation from direct route	75%	87%

**Table 6**  
Simulation 1 – taxi occupancy rates.

Taxi	Lagrangian decomposition					ICH				
	0	1	2	3	4	0	1	2	3	4
1	98%	2%	0%	0%	0%	6%	28%	41%	21%	4%
2	72%	11%	8%	9%	0%	23%	37%	33%	7%	0%
3	80%	5%	11%	3%	0%	53%	21%	19%	7%	0%
4	96%	4%	0%	0%	0%	68%	12%	7%	7%	6%
5	99%	1%	0%	0%	0%	100%	0%	0%	0%	0%
6	80%	5%	10%	5%	0%	89%	11%	0%	0%	0%
7	73%	12%	9%	6%	0%	100%	0%	0%	0%	0%
8	70%	11%	9%	7%	3%	100%	0%	0%	0%	0%
9	97%	3%	0%	0%	0%	94%	6%	0%	0%	0%
10	99%	1%	0%	0%	0%	99%	1%	0%	0%	0%
11	92%	8%	0%	0%	0%	100%	0%	0%	0%	0%
12	96%	4%	0%	0%	0%	100%	0%	0%	0%	0%
13	100%	0%	0%	0%	0%	100%	0%	0%	0%	0%
14	91%	9%	0%	0%	0%	100%	0%	0%	0%	0%
15	87%	7%	5%	0%	0%	82%	3%	7%	4%	5%
16	70%	11%	9%	7%	3%	100%	0%	0%	0%	0%
17	77%	11%	3%	6%	3%	100%	0%	0%	0%	0%
18	97%	3%	0%	0%	0%	100%	0%	0%	0%	0%
19	100%	0%	0%	0%	0%	100%	0%	0%	0%	0%
20	93%	7%	0%	0%	0%	100%	0%	0%	0%	0%
Avg	88%	6%	3%	2%	0%	86%	6%	5%	2%	1%
Min	70%	0%	0%	0%	0%	6%	0%	0%	0%	0%
Max	100%	12%	11%	9%	3%	100%	37%	41%	21%	6%

p\_6\_1\_6\_(R,T), p\_2\_1\_4\_R\_2, p\_2\_1\_4\_T, p\_4\_1\_12\_T\_1). Assuming that all the passengers requests are available at the initial period, the incremental cost heuristic is also used to compute feasible solutions. Although ICH has been proposed for the dynamic setting, it can be used for the case where all calls are present at the initial period by processing the calls in the order that they are listed. The incremental cost heuristic is more computationally expensive than the simple heuristic however it tends to find better quality solutions. For the tested instances, ICH found feasible solutions that have on average 12% better objective function value at the expense of consuming an average of 115 s of computational time. Finally, the results show

**Table 7**  
Simulation 2 – results summary.

	ICH
Average computational time (sec.)	3.44
Minimum computational time (sec.)	0.07
Maximum computational time (sec.)	46.37
Number of rejected passengers	5
Number of taxis used	31
Total profit	222.2
Average % deviation from direct route	6%
Minimum % deviation from direct route	0%
Maximum % deviation from direct route	43%

**Table 8**  
Simulation 2 – taxi occupancy rates.

Taxi	ICH % of time taxi occupancy is					Taxi	ICH % of time taxi occupancy is				
	0	1	2	3	4		0	1	2	3	4
1	35%	34%	27%	3%	0%	26	100%	0%	0%	0%	0%
2	53%	27%	16%	3%	0%	27	100%	0%	0%	0%	0%
3	44%	32%	22%	3%	0%	28	75%	19%	6%	0%	0%
4	50%	37%	13%	0%	0%	29	100%	0%	0%	0%	0%
5	46%	35%	14%	5%	0%	30	100%	0%	0%	0%	0%
6	46%	24%	16%	9%	5%	31	100%	0%	0%	0%	0%
7	64%	25%	6%	5%	0%	32	93%	7%	0%	0%	0%
8	64%	26%	10%	0%	0%	33	97%	3%	0%	0%	0%
9	67%	25%	8%	0%	0%	34	100%	0%	0%	0%	0%
10	81%	19%	0%	0%	0%	35	100%	0%	0%	0%	0%
11	71%	18%	9%	3%	0%	36	100%	0%	0%	0%	0%
12	76%	21%	3%	0%	0%	37	98%	2%	0%	0%	0%
13	76%	16%	5%	3%	0%	38	100%	0%	0%	0%	0%
14	84%	12%	4%	0%	0%	39	100%	0%	0%	0%	0%
15	89%	7%	4%	0%	0%	40	100%	0%	0%	0%	0%
16	94%	6%	0%	0%	0%	41	100%	0%	0%	0%	0%
17	81%	17%	2%	0%	0%	42	100%	0%	0%	0%	0%
18	82%	14%	4%	0%	0%	43	88%	7%	4%	1%	0%
19	96%	4%	0%	0%	0%	44	100%	0%	0%	0%	0%
20	95%	5%	0%	0%	0%	45	100%	0%	0%	0%	0%
21	97%	3%	0%	0%	0%	46	98%	2%	0%	0%	0%
22	85%	9%	4%	1%	0%	47	100%	0%	0%	0%	0%
23	100%	0%	0%	0%	0%	48	100%	0%	0%	0%	0%
24	84%	11%	5%	0%	0%	49	96%	4%	0%	0%	0%
25	92%	6%	2%	0%	0%	50	100%	0%	0%	0%	0%
						Avg	86%	10%	4%	1%	0%
						Min	35%	0%	0%	0%	0%
						Max	100%	37%	27%	9%	5%

that the incremental cost heuristic is more computationally expensive on the instances with relaxed time windows compared to the ones with tight time window. This is due to the effect of the infeasibility detection that was detailed in Section 5.1.1. For the instances with tight time windows, it is more common for problem [SVP] to be infeasible and thus due to the infeasibility detection, considerable computational time is saved.

In the following section, we discuss the implementation and evaluate the performance of ICH in a dynamic setting through a simulation that mimics a real life scenario.

### 6.3. Dynamic framework

In order to assess the applicability of the proposed algorithms in a dynamic setting, we construct two simulation runs that mimic real life scenarios. Particularly, we use the real street network of Manhattan and conduct two hours simulation runs.

In the first simulation, we consider a set of 20 taxis and 60 passengers with arrival times uniformly distributed over the two hours simulation time window with randomly generated origin and destination locations. The maximum capacity for each taxi is set to 4 passengers and the maximum ride time for every seeker is set to include a 20 min maximum deviation from the direct route travel time. With the same data, two runs are conducted. In the first run, ICH is used for assignment while in the second run the Lagrangian approach is used in order to obtain the optimal assignment. Given the results of the assignment algorithm, the taxis are then routed accordingly.

A summary of the results is shown in [Table 5](#). ICH requires an average computational time of 1.68 s with a minimum and a maximum of 0.02 and 9.57 s respectively. Alternatively, the Lagrangian approach requires an average of 86.83 s with a minimum of 0.01 and 1219.58 s respectively. With the ICH approach, 8 taxis are used achieving a total profit of 63.8 while 3 passengers are rejected as they were deemed unprofitable. With the Lagrangian approach, 18 taxis are used achieving a total profit of 71.5 while none of the passengers is rejected. [Table 6](#) indicates the number of passengers that are sharing taxis at the same time. The ICH heuristic resulted in more sharing compared to the Lagrangian approach while the total profit is lower.

To further evaluate the performance of the proposed algorithms, we conduct a second simulation that involves a larger set of participants. While the number of taxis and potential passengers can potentially be significantly larger, it is justifiable to limit them for practical computational considerations. In practice, it is typical to limit the set of taxis to those that are within the area of interest based on their proximity to the seekers. Accordingly in this simulation, we consider a set of 50 taxis. Moreover, guided by the survey results by [Schaller Consulting \(2006\)](#) which indicate that for every taxi an average of 4 potential customers over two hours are present in the area of Manhattan, we consider 200 passengers with randomly generated pickup and dropoff locations. In this simulation, due to the additional number of taxis and passengers, the Lagrangian decomposition approach fails in solving the majority of problems within the 10 min computational time limit that we set, we thus limit the discussion and the presentation of the results to the ICH approach which has been specifically introduced for the dynamic case. In the results that are summarized in [Tables 7 and 8](#), we notice that ICH is computationally efficient in solving the instances in less than 50 s of computational time and with an average of 3.44 s, thus justifying its fit to be used in real time dynamic settings. Given the 200 seekers, 5 were rejected as they were deemed unprofitable, and the deviation from the shortest route was only 6% on average.

While both the Lagrangian approach and the ICH algorithm can be used to make taxi assignments in practice, the simulation results show that ICH is applicable in a dynamic setting since good quality solutions are computed in relatively short computational time. As detailed in [Section 5](#), ICH is inspired from the Lagrangian approach and is designed for the dynamic case. On the other hand the Lagrangian approach provides higher profit at the expense of additional computational time.

## 7. Conclusion

This paper presented a new mixed integer programming formulation for the shared taxi problem. A solution approach based on Lagrangian decomposition which exploits the structure of the problem is also proposed. Furthermore, two heuristics are also presented to find good quality feasible solutions. The numerical results demonstrate the quality of the proposed solution methodology which outperformed CPLEX for a number of tested instances.

Solving the Lagrangian subproblems remains as the major computational burden for the proposed approach. In future research work, we aim to focus on the subproblem and propose efficient methods for its solution hence improving the overall performance of the proposed Lagrangian approach.

## Acknowledgments

We greatly thank two anonymous referees for their valuable feedback and comments that helped improve the content of the paper.

## References

- Agatz, Niels, Erera, Alan, Savelsbergh, Martin, Wang, Xing, 2011. Dynamic ride-sharing: a simulation study in metro Atlanta. *Transportation Research Part B* 45, 1450–1464.
- Agatz, Niels, Erera, Alan, Savelsbergh, Martin, Wang, Xing, 2012. Optimization for dynamic ride-sharing: a review. *European Journal of Operational Research* 223 (2), 295–303.
- Attanasio, Andrea, Cordeau, Jean-François, Ghiani, Gianpaolo, Laporte, Gilbert, 2004. Parallel Tabu search heuristics for the dynamic multi-vehicle dial-a-ride problem. *Parallel Computing* 30 (3), 377–387.
- Baldacci, R., Maniezzo, V., Mingozzi, A., 2004. An exact method for the car pooling problem based on Lagrangian column generation. *Operations Research* 52 (3), 422–439.
- Beaudry, Alexandre, Laporte, Gilbert, Melo, Teresa, Nickel, Stefan, 2010. Dynamic transportation of patients in hospitals. *OR Spectrum* 32 (1), 77–107.
- Berbeglia, Gerardo, Cordeau, Jean-François, Laporte, Gilbert, 2010. Dynamic pickup and delivery problems. *European Journal of Operational Research* 202 (1), 8–15.
- Berbeglia, Gerardo, Cordeau, Jean-François, Laporte, Gilbert, 2012. A hybrid tabu search and constraint programming algorithm for the dynamic dial-a-ride problem. *INFORMS Journal on Computing* 24 (3), 343–355.
- Cordeau, Jean-François, 2006. A branch-and-cut algorithm for the dial-a-ride problem. *Operations Research* 54 (3), 573–586.
- Cordeau, Jean-François, Laporte, Gilbert, 2003. A Tabu search heuristic for the static multi-vehicle dial-a-ride problem. *Transportation Research Part B* 37 (6), 579–594.
- Coslovich, Luca, Pesenti, Raffaele, Ukovich, Walter, 2006. A two-phase insertion technique of unexpected customers for a dynamic dial-a-ride problem. *European Journal of Operational Research* 175 (3), 1605–1615.
- Desaulniers, Guy, Desrosiers, Jacques, Erdmann, Andreas, Solomon, Marius M., Soumis, François, 2002. VRP with pickup and delivery. *The Vehicle Routing Problem* 9, 225–242.
- Desrosiers, Jacques, Dumas, Yvan, Solomon, Marius M., Soumis, François, 1995. Time constrained routing and scheduling. *Handbooks in Operations Research and Management Science* 8, 35–139.
- Elhedhli, Samir, Xiaolong Hu, Frank, 2005. Hub-and-spoke network design with congestion. *Computers and Operations Research* 32 (6), 1615–1632.

- Elhedhli, Samir, Li, Lingzi, Gzara, Mariem, Naoum-Sawaya, Joe, 2011. A branch-and-price algorithm for the bin packing problem with conflicts. *INFORMS Journal on Computing* 23 (3), 404–415.
- Furuhata, Masabumi, Dessouky, Maged, Ordóñez, Fernando, Brunet, Marc-Etienne, Wang, Xiaoqing, Koenig, Sven, 2013. Ridesharing: the state-of-the-art and future directions. *Transportation Research Part B*, 57 28–46.
- Ghaddar, Bissan, Naoum-Sawaya, Joe, Kishimoto, Akihiro, Taheri, Nicole, Eck, Bradley, 2014. A Lagrangian decomposition approach for the pump scheduling problem in water networks. *European Journal of Operational Research*. <http://dx.doi.org/10.1016/j.ejor.2014.08.033> (in press).
- Jorgensen, R.M., Larsen, Jesper, Berg Bergvinsdottir, Kristin, 2006. Solving the dial-a-ride problem using genetic algorithms. *Journal of the Operational Research Society* 58, 1321–1331.
- Kirchler, Dominik, Wölfler Calvo, Roberto, 2013. Granular Tabu search algorithm for the dial-a-ride problem. *Transportation Research Part B*, 56 120–135.
- Madsen, Oli B.G., Ravn, Hans F., Moberg Rygaard, Jens, 1995. A heuristic algorithm for a dial-a-ride problem with time windows, multiple capacities, and multiple objectives. *Annals of operations Research* 60 (1), 193–208.
- Paquette, Julie, Cordeau, Jean-François, Laporte, Gilbert, Pascoal, Marta, 2013. Combining multicriteria analysis and tabu search for dial-a-ride problems. *Transportation Research Part B*, 52 1–16.
- Parragh, Sophie N., Doerner, Karl F., Hartl, Richard F., 2010. Variable neighborhood search for the dial-a-ride problem. *Computers and Operations Research*, 37 1129–1138.
- Parragh, Sophie N., Cordeau, Jean-François, Doerner, Karl F., Hartl, Richard F., 2012. Models and algorithms for the heterogeneous dial-a-ride problem with driver-related constraints. *OR Spectrum* 34 (3), 593–633.
- Psaraftis, Harilaos N., 1980. A dynamic programming solution to the single vehicle many-to-many immediate request dial-a-ride problem. *Transportation Science* 14 (2), 130–154.
- Psaraftis, Harilaos N., 1983. An exact algorithm for the single vehicle many-to-many dial-a-ride problem with time windows. *Transportation Science* 17 (3), 351–357.
- Ropke, Stefan, Cordeau, Jean-François, Laporte, Gilbert, 2007. Models and branch-and-cut algorithms for pickup and delivery problems with time windows. *Networks* 49 (4), 258–272.
- Schaller Consulting, 2006. *The New York City Taxicab Fact Book*. Technical Report, Schaller Consulting.
- Schilde, Michael, Doerner, Karl F., Hartl, Richard F., 2011. Metaheuristics for the dynamic stochastic dial-a-ride problem with expected return transports. *Computers and operations research* 38 (12), 1719–1730.
- Transport Canada, 2011. *Taxi-share Programs in Canada and Abroad*. Technical Report TP 15151 E.
- Xiang, Zhihai, Chu, Chengbin, Chen, Haoxun, 2006. A fast heuristic for solving a large-scale static dial-a-ride problem under complex constraints. *European Journal of Operational Research* 174 (2), 1117–1139.
- Xiang, Zhihai, Chu, Chengbin, Chen, Haoxun, 2008. The study of a dynamic dial-a-ride problem under time-dependent and stochastic environments. *European Journal of Operational Research* 185 (2), 534–551.