Maximizing Revenues for On-Line Dial-a-Ride [⋆]

Ananya Christman[1], William Forcier[2], Aayam Poudel[3]

[1] Middlebury College, Middlebury VT 05753
achristman@middlebury.edu,
[2] Abbott Laboratories, Lake Forest IL 60045
william.forcier@abbott.com
[3] Middlebury College, Middlebury VT 05753
apoudel@middlebury.edu

Abstract. *We consider the On-Line Dial-a-Ride Problem, where a server fulfills requests that arrive over time. Each request has a source, destination, and release time. We study a variation of this problem where each request also has a revenue that the server earns for fulfilling the request. The goal is to serve requests within a time limit while maximizing the total revenue. We first prove that no deterministic online algorithm can be competitive unless the input graph is complete and edge weights are unit. We therefore focus on these graphs and present a 2-competitive algorithm for this problem. We also consider two variations of this problem: (1) the input graph is complete bipartite and (2) there is a single node that is the source for every request, and present a 1-competitive algorithm for the former and an optimal algorithm for the latter. We also provide experimental results for the complete and complete bipartite graphs. Our simulations support our theoretical findings and demonstrate that our algorithms perform well under settings that reflect realistic Dial-a-Ride systems.*

Keywords: online algorithms, dial-a-ride, competitive analysis, graphs

1 Introduction

In the original Dial-a-Ride Problem (DARP), a server travels in some metric space to serve requests for rides. Each request specifies a *source*, which is the pick-up (or start) location of the ride, a *destination*, which is the delivery (or end) location, and a release time which is the earliest time the request may be served. The server serves a request by traveling from the source to the destination on or after the release time. The server may have a prescribed capacity for serving one or more requests at a time. Common optimality criteria for DARP include minimizing the total distance traveled, or minimizing the total travel time (i.e. makespan), minimizing the average completion time (i.e. latency), or maximizing the number of requests served within a specified time limit. DARP generalizes both classical job scheduling problems and classical routing problems like the Traveling Salesperson Problem. For most basic variants, DARP is known to be NP-hard [6, 8, 10, 18].

In the On-Line Dial-a-Ride Problem (OLDARP), the requests arrive over time (each request arrives at its release time) and time is discretized. At each time unit, the server considers the available requests so far and decides which to service. In many cases preemption is not allowed, so if the server decides to serve a request, it must do so until completion.

On-Line Dial-a-Ride Problems have many practical applications in settings where a vehicle is dispatched to satisfy requests involving pick-up and delivery of people or goods. Important examples include ambulance routing, transportation for the elderly and disabled, taxi services (such as Uber, Lyft, and Via), and courier services. A less obvious but equally important application involves programming intelligent robots for IED detection and dismantlement.

The OLDARP variant we consider is one where each request has an accompanying priority value, which may also be thought of as a revenue earned from serving that request. We refer to this problem as ROLDARP (Revenue OLDARP). The goal is to maximize total priority of requests served (or revenue earned). To our knowledge, despite having a strong relevance to modern-day OLDARP systems, prior to the work in [7], this objective has not previously been studied. There are also many relevant variants

[⋆] A preliminary version of this work appeared in the proceedings of *Conference of Combinatorial Optimization and Application, 2014* of OLDARP with this objective that have yet to be studied and analyzed. We discuss more details and related work on this problem in Section 2.

For ROLDARP, the metric space is modeled by a graph of nodes and weighted edges, where edge weights represent the travel times between nodes. We present a 2-competitive algorithm to solve this problem for complete graphs where edge weights are unit. While our algorithm applies to only these graphs, we show that no competitive algorithm exists if we relax either of these graph properties. Specifically, we prove that no deterministic online algorithm can be competitive for complete graphs with varying edge weights nor for non-complete graphs. We also consider a variation of this problem where the input graph is complete bipartite with unit-weight edges, and where every source is from the left-hand side and every destination is from the right-hand side. For these graphs, we present a 1-competitive algorithm. We analyze our algorithms based on their *competitive ratio*, i.e. the worst-case ratio between the revenue earned by the algorithm and the revenue earned by an optimal offline algorithm that knows all of the requests in advance. Although the competitive ratios of both algorithms include an additive constant equal to the last request served by the optimal algorithm, we show that no online algorithm can avoid this constant. Finally, we consider a version of ROLDARP where there is a single node that is the source of all requests and present an optimal algorithm for this problem.

The remainder of this paper is organized as follows. In Section 2 we describe several works related to our problem. In Section 3 we formally define ROLDARP and describe the variations: V-ROLDARP (where edge weights are varying), complete-bipartite ROLDARP, and single-source ROLDARP. In Section 4 we prove that no deterministic online algorithm can be competitive for V-ROLDARP. In Section 5 we present a 2-competitive algorithm for ROLDARP. In Section 6.1 we present a 1-competitive algorithm for complete bipartite ROLDARP and in Section 6.2 we present an optimal algorithm for single-source ROLDARP. In Section 7 we present experimental results of our ROLDARP algorithm on various examples. Finally in Section 8 we summarize our results and discuss some possible extensions.

2 Related Work

The Online Dial-a-Ride problem was introduced by Feuerstein and Stougie [22] and several variations of the problem have been studied since. For a comprehensive survey on these and many other problems in the general area of *vehicle routing* see [13]. The authors of [22] studied the problem for two different objectives. One was to minimize the time to serve all requests and return to the origin (also known as *completion time*); the other was to minimize the average time it takes to complete the requests (also known as *latency*). For minimizing completion time, they showed that any deterministic algorithm must have competitive ratio of at least 2 regardless of the server capacity. They presented algorithms for the cases of finite and infinite capacity with competitive ratios of 2.5 and 2, respectively. For minimizing latency, they proved that any algorithm must have a competitive ratio of at least 3. They presented a 15-competitive algorithm for this problem on the real line in which the server has infinite capacity.

Ascheuer et al. [1] studied minimizing total completion time for OLDARP with multiple servers and capacity constraints and presented a 2-competitive algorithm for this problem. Jaillet and Wagner [12] considered a version of OLDARP where each request consists of one or more locations, and precedence and capacity requirements for the locations. To serve a request the server must visit each location, while satisfying the requirements. The authors provided a nonpolynomial-time 2-competitive algorithm for minimizing completion time.

The Online Traveling Salesperson Problem (OLTSP), introduced by Ausiello [2], is a special case of OLDARP where for each request the source and destination are the same location. In [2] the server starts at an origin and must serve all requests while minimizing the overall completion time. The authors studied this problem on the Euclidean space and proved that no online algorithm can be better than 2-competitive. They gave a 2.5-competitive non-polynomial time online algorithm and a 3-competitive polynomial time algorithm to solve this problem. Krumke [15] studied both OLDARP and OLTSP for the uniform metric space. Their objective was to minimize the maximum *flow time*, that is the amount of time it takes for the server to start serving a request after it has been released. They proved that no competitive algorithm exists for OLDARP and gave a 2-competitive algorithm to solve OLTSP.

Ausiello et al. [4] studied a variation of OLTSP where each request has both a penalty and a weight. The goal was to collect a specified quota of weights by satisfying a sufficient number of requests, while minimizing the total service time plus the penalties of rejected requests. They gave a 7/3-competitive algorithm for this problem on a general graph and proved lower and upper bounds of 1.89 and 2, respectively, on the real halfline. Jaillet and Wu [11] considered a variation of OLTSP where each request also has a penalty (incurred if the request is rejected). The goal was to minimize the time to serve all accepted requests plus the sum of the penalties of rejected requests. They gave a 2-competitive algorithm to solve the problem on the real line and a 2.28-competitive algorithm on a general metric space. Wen et al. [25] model customers' waiting psychology into the online TSP. To do so, they associate with each request a disclosure time, which is the amount of time a request is known to the server prior to its release time. Their objective was to serve as many requests as possible. Surprisingly, they find that even with this advanced information, no deterministic algorithm can achieve a constant competitive ratio on the general metric space but present a 2-competitive algorithm for the unit metric space. Finally, Liao et al. [16] study a variation of the Online Canadian Traveller Problem (CTP). In this problem, the server must find the shortest route from a source to a destination under uncertain conditions such as road blockages. They consider a variation where the server must find the shortest tour that visits a set of locations and returns to the origin, and present a k-competitive ratio, where k is the maximum number of blockages.

Within the last five years many heuristic approaches have emerged as means for tackling practical variations of Dial-a-Ride problems and here we describe just a handful. Schilde et al. [20] consider a hybrid version of DARP where some requests are static, i.e. known in advance, while others are dynamic, i.e. arrive online, and finally, some are stochastic in that they arrive based on a conditional probability related to previous requests. Azi et al. [5] employ a decision policy that applies an adaptive large neighborhood search heuristic to generate multiple possible scenarios of future requests. Kergosien et al. [14] present a tabu search heuristic, inspired by the research of [9], for transporting patients between care units. Lorini et al. [17] demonstrate that using preemption, i.e. allowing the vehicle to divert from its current route to accommodate new requests, yields more efficient routing of customer requests. In each of these studies, the proposed algorithms were shown to be effective via experimental simulations.

On the other hand, provably competitive results on OLDARP have been sparse within the last five years; in fact we are unaware of any such work, other than [7], on the revenue maximizing OLDARP problem we have described above. To our knowledge, this is also the first work that presents experimental results for OLDARP with the goal of maximizing total revenue within a time limit.

3 Problem Statement

We study a variation of the Online-Dial-a-Ride Problem where in addition to the source, destination and release time, each request also has a priority, and there is a time limit within which requests must be served. The goal for the server is to serve requests within the time limit so as to maximize the total priority. In settings such as courier services, a request's priority may simply represent the importance of serving the request. In more time-sensitive settings such as ambulance routing, the priority may represent the urgency of a request. In profit-based settings, such as taxi and ride-sharing services, the priority may represent the revenue earned from serving the request.

Formally, the input is an undirected complete graph $G = (V, E)$ where V is the set of vertices (or nodes) and $E = \{(u, v) : u, v \in V, u \neq v\}$ is the set of edges. (We note that in fact any simple, undirected, connected, weighted graph is allowed as input, with the simple pre-processing step of adding an edge wherever one is not present whose weight is the length of the shortest path between its two endpoints. We further note that the input can be regarded as a metric space if the weights on the edges are expected to satisfy the triangle-inequality.) For every edge $(u, v) \in E$, there is a weight $w_{u,v} > 0$. One node in the graph, o, is designated as the origin and is where the server is initially located (i.e. at time 0).

The input also includes a time limit T and a sequence of requests, σ, that is dynamically issued to the server. Each request is of the form (s, d, t, p) where s is the source node, d is the destination, t is the time the request is released, and p is the revenue value (or priority) earned by the server for serving the request. To serve a request, the server must move from its current location x to s, then from s to d (or simply from s to d if the server is already at s). The total time is equal to the length of the path from x to d (or s to d if the server is already at s). We assume the earliest time a request may be released is at $t = 0$.

For each request, the server must decide whether to serve the request and if so, at what time. A request may not be served earlier than its release time and at most one request may be served at any given time (i.e. the server capacity is one). Once the server starts serving a request, it must serve the request until completion (i.e. preemption is not allowed). The goal for the server is to serve requests within the time limit so as to maximize the total earned revenue.

For OLDARP, an algorithm $\textsc{on}$ is considered *online* if it learns about a request only at its release time, whereas an algorithm is considered offline if it is aware of all requests at time 0 (i.e. the earliest time). We let $\textsc{opt}$ denote an optimal offline algorithm, as in, an algorithm that given any input will serve the requests that return the maximum possible revenue for that input. Given an input graph G, a sequence $\sigma = r_1, \ldots r_m$ of requests, we denote $\textsc{on}(G, \sigma)$ and $\textsc{opt}(G, \sigma)$ as the total revenue earned by $\textsc{on}$ and $\textsc{opt}$, respectively, from σ on G. We say that $\textsc{on}$ is *c-competitive* if there exists $c \geq 1, b \geq 0$ such that for all σ:

$$\textsc{opt}(G, \sigma) \leq c \cdot \textsc{on}(G, \sigma) + b \tag{1}$$

We consider several variations of ROLDARP which we summarize below:

- original ROLDARP - The graph is a complete undirected graph where every edge has unit weight. Each request has a source, destination, release date, and revenue that is earned for serving the request. There is a global time limit before which requests must be served. The goal is to maximize the total revenue earned within the time limit.
- V-ROLDARP - Edges in the graph have varying weights.
- complete-bipartite ROLDARP - The graph is an undirected complete-bipartite graph where every source is from the left-hand side and every destination is from the right-hand side of the graph.
- single-source ROLDARP - One vertex is the source for every request.

4 Non-Competitiveness of V-ROLDARP

We first consider V-ROLDARP. The input to this problem is a complete undirected graph G of $n \geq 2$ nodes where for every edge (u, v) there is a weight $w_{u,v} > 0$, and there are at least two distinct edges (u, v) and (x, y) where $w_{u,v} \neq w_{x,y}$. In this section, we prove that no deterministic online algorithm can be competitive for V-ROLDARP. Note that any connected graph can be converted to a complete graph such that the pairwise distance between nodes of both graphs is equivalent. Specifically, for two non-adjacent nodes i and j of a non-complete graph, we can create an edge (i, j) with weight equal to the distance between i and j. Therefore this proof also holds for connected non-complete graphs.

In Section 5 we consider ROLDARP and provide an algorithm that is 2-competitive when the additive constant b from Equation (1) is the revenue of the last request served by $\textsc{opt}$, v_{last}. In this section, we prove that no algorithm can be competitive for V-ROLDARP for any $b \leq v_{last}$[§]). In particular, we first show that there is no value b independent of the input such that a deterministic online algorithm can be c-competitive. We then show that even if we allow b to depend on the input, no deterministic online algorithm can be c-competitive with $b \leq v_{last}$. Specifically, we show that an adversary can issue a request sequence σ such that for any $\alpha \geq 1, b \leq v_{last}$, $\alpha \cdot \textsc{on}(G, \sigma) + b < \textsc{opt}(G, \sigma)$ where G is any input graph as described above.

Theorem 1. *No deterministic online algorithm can be competitive for V-ROLDARP.*

Proof. We first show that there is no value b independent of the input such that a deterministic online algorithm can be competitive. In other words, the additive constant b in Equation (1) must depend on the input (in particular, the last request served by $\textsc{opt}$, v_{last}). Let $T \geq 2$; for all time units before $T - 1$, the adversary will release no requests. At time $T - 1$ a deterministic online algorithm must be at some node u of G. The adversary can release a request $(w, u, T - 1, v_{last})$, so the request sequence σ consists only of this request. This request cannot be served by the online algorithm, but it can be served by the optimal algorithm, so the online algorithm earns 0 while the optimal algorithm earns v_{last}. Since we can set $v_{last} = b + 1$, for any b, $\textsc{opt}(G, \sigma) > \textsc{on}(G, \sigma) + b$. This also shows that no deterministic online algorithm can be competitive when $b < v_{last}$.

[§] Note that since v_{last} can be at most the maximum allowed revenue, this proof shows non-competitiveness for b equal to all possible revenue values.

We now show that even if we allow an additive constant of $b = v_{last}$, still no online algorithm can be competitive. Specifically we show that an adversary can issue a request sequence σ such that for any $\alpha \geq 1, b \leq v_{last}$, $\alpha \cdot \text{ON}(G, \sigma) + b < \text{OPT}(G, \sigma)$.

Let G denote a complete graph with three nodes s, x, and y, where s is the origin, $w_{s,x} = c$ for any $c \geq 3$, $w_{s,y} = 1$ and $w_{x,y} > c$, and let $T \geq 2c$ denote the time limit. The adversary will release two requests: $(s, x, T - 2c, v)$ and $(x, s, T - c, v)$ for $v > 0$. There are two cases for ON:

Case 1: ON serves neither of these requests.
Since there is enough time for OPT to serve both requests, $\text{OPT}(G, \sigma) = 2v$ while $\text{ON}(G, \sigma) = 0$. If $b = v_{last}$ we have $\alpha \cdot 0 + v_{last} = v \leq 2v$ for any α, so ON is not competitive.

Case 2: ON serves at least one of these requests.
ON earns revenue at most $2v$. Suppose ON starts serving a request at time t. Then the adversary will release two requests $(s, y, t + 1, v^*)$ and $(y, s, t + 1, v^*)$ where $v^* = \alpha \cdot 2v + 1$. ON will not have enough time to serve either of these requests but OPT will serve both of these requests and earn revenue $2v^* = v^* + v_{last} = \alpha \cdot 2v + 1 + v_{last}$. For $b = v_{last}$ we have $\alpha \cdot 2v + v_{last} \leq \alpha \cdot 2v + 1 + v_{last}$ for any α, so ON is not competitive.

5 ROLDARP on a Complete Graph with Unit Edges

In this section, we provide an online algorithm, Greatest Revenue First (GRF) that is 2-competitive for ROLDARP. Specifically, for input graph G, given request sequence σ, if $\text{OPT}(G, \sigma)$ denotes the optimal revenue earned from σ, $\text{GRF}(G, \sigma)$ denotes the amount of revenue earned by GRF from σ, and v_{last} denotes the last revenue earned by OPT, we show:

$$\text{OPT}(G, \sigma) \leq 2 \cdot \text{GRF}(G, \sigma) + v_{last} \qquad (2)$$

We first show that for any graph G with $n \geq 2$ nodes and a time limit $T \geq 2$ no online algorithm can avoid the v_{last} additive constant of Equation (2). In particular an adversary can generate a request sequence such that no online algorithm can serve the last request of the sequence.

At time $T - 1$ any online algorithm must be at some node in G. Consider two arbitrary nodes u and v. If the algorithm is at u the adversary can release a request from v to u. If the algorithm is not at u the adversary can release a request from u to v. In either case the online algorithm cannot satisfy the request while an optimal algorithm can.

Algorithm 1 describes the GRF algorithm.[†] The main idea is that for every time unit for which there is some unserved request, GRF either moves to the source location of the request with the highest revenue or serves a previously issued request with the highest revenue. However, when the time limit is odd, the algorithm first waits for one time unit to allow more requests to arrive, so it can make a more informed decision about which requests to serve.

Theorem 2. *Greatest Revenue First is 2-competitive.*

Proof. We now prove Equation 2 to show that GRF is 2-competitive. We assume without loss of generality, that a request is issued at every time unit[‡]. To prove Equation 2, we consider another algorithm MAX. We define MAX such that at every time unit except $T - 1$, MAX serves the request with the greatest revenue regardless of the source node of the request. At time $T - 1$, MAX does nothing. Note that MAX may not coincide with the request set of a feasible algorithm. In other words, given the input graph, request sequence, and time limit, MAX may fulfill a set of requests that *no* algorithm can fulfill. For example, suppose the origin is some node o. Suppose at time 0, a request with maximal revenue is released with

[†] We note that there are at least two enhancements that can improve the performance of GRF without improving the competitive ratio: (1) In steps 2 and 7, instead of simply moving to the request that earns the greatest revenue, the algorithm can serve a request if there is one available while performing this move. (2) In steps 3 and 8, the algorithm can check if a request with higher revenue has been released since the previous step and if so, serve this request instead of r.

[‡] If at any time t, there is no request issued, we can generate a "dummy" request of the form $(s, d, t, 0)$, where s and d are nodes in the input graph, since neither GRF nor any optimal algorithm would accept this request.

Algorithm 1 Algorithm GRF. Input is complete graph G and time limit T.

```
1: if $T$ is even then
2:     At every even time, determine which released request earns the greatest revenue and move to the source location of this request. Denote this request as $r$. If no unserved requests exist, do nothing until the next even time.
3:     At every odd time, serve request $r$ (if it exists) from the previous step.
4: end if
5: if $T$ is odd then
6:     At time 0, do nothing.
7:     At every odd time, determine which released request earns the greatest revenue and move to the source location of this request. Denote this request as $r$. If no unserved requests exist, do nothing until the next odd time.
8:     At every even time, serve request $r$ (if it exists) from the previous step.
9: end if
```

source $s_0 \neq o$ and destination d_0. No algorithm can serve this request in the time slot from 0 to 1, but we assume that MAX does. Thus, by the construction of MAX, for any sequence of requests σ, the following equation holds:

$$\text{MAX}(G,\sigma) \geq \text{OPT}(G,\sigma) - v_{last} \tag{3}$$

We will show that:

$$2 \cdot \text{GRF}(G,\sigma) \geq \text{MAX}(G,\sigma) \tag{4}$$

A proof of (4) will immediately prove (2).

We now prove Equation (4). For this proof we use the terminology "algorithm A *serves* (or *has served*) request r at time t" to indicate that A *begins* serving r at time t and *completes* serving r at time $t+1$.

Let $r_0, r_1, r_2, \dots r_{T-2}$ denote the requests served by MAX at times $0, 1, 2, \dots T-2$ earning revenues $v_0, v_1, v_2, \dots v_{T-2}$. We consider two cases based on the parity of T.

Case 1: T is even (Table 1 shows an example with $T = 6$).
At $t = 0$, GRF and MAX determine that v_0 is the greatest revenue. At $t = 0$ MAX fulfills v_0 and at $t = 1$ GRF fulfills v_0.

We now show that for every odd time $t \neq 1$, if MAX serves r_{t-1} and r_{t-2} at times $t-1$ and $t-2$, earning total revenue $v_{t-1} + v_{t-2}$, then at time t GRF earns revenue at least $\max\{v_{t-1}, v_{t-2}\}$. Note that when T is even GRF serves requests only at odd times. We show that at time $t-1$, when GRF decides which request to serve at time t, both r_{t-1}, r_{t-2} are available requests (so GRF will serve the one with the higher revenue).

Consider r_{t-1} and r_{t-2}. Since MAX has served them at $t-1$ and $t-2$, they must have been released by times $t-1$ and $t-2$, respectively. The only way that they would not be available requests for GRF at time $t-1$ is if GRF has already served them. However, this is not possible because if GRF serves a request at some time τ, it must have been released by time $\tau - 1$, and therefore MAX must serve this request by time $\tau - 1$ at the latest.

Case 2: T is odd.
Omit the first paragraph and replace odd with even and even with odd in the proof for Case 1.

Now, let r'_t denote the request served by GRF at time t and let v'_t denote the revenue of r'_t. Then (assuming $v_t = 0$ for $t < 0$), for all times t in which GRF earns revenue:

$$v'_t \geq \max\{v_{t-1}, v_{t-2}\} \tag{5}$$
$$2 \cdot v'_t \geq \quad v_{t-1} + v_{t-2} \tag{6}$$

Since Equation (6) holds for all $r'_t \in \sigma$ served by GRF, we have:

$$2 \cdot \text{GRF}(G,\sigma) \geq \text{MAX}(G,\sigma) \tag{7}$$

Then from (3), we have

$$2 \cdot \text{GRF}(G, \sigma) \geq \text{OPT}(G, \sigma) - v_{last} \tag{8}$$
$$2 \cdot \text{GRF}(G, \sigma) + v_{last} \geq \text{OPT}(G, \sigma) \tag{9}$$

Table 1: An example of the revenues earned by GRF and MAX for $T = 6$ (GRF revenues given w.l.o.g.). GRF earns total revenue $v_{\text{GRF}} = v_0 + v_1 + v_3$ and MAX earns total revenue $v_{\text{MAX}} = v_0 + v_1 + v_2 + v_3 + v_4$. Since $v_1 \geq v_2$ and $v_3 \geq v_4$, we have $2 \cdot v_{\text{GRF}} \geq v_{\text{MAX}}$.

t	GRF	MAX
0	0	v_0
1	v_0	v_1
2	0	v_2
3	$v_1 = max(v_1, v_2)$	v_3
4	0	v_4
5	$v_3 = max(v_2, v_3, v_4)$	0

6 Variants of ROLDARP

6.1 Complete Bipartite ROLDARP

In this section, we consider ROLDARP for complete bipartite graphs, specifically where if V_1 and V_2 denote the two sets of nodes, then every source is in V_1 and every destination is in V_2. We prove that a modified version of Greatest Revenue First, BGRF (Bipartite Greatest Revenue First) is 1-competitive for this version of ROLDARP (see Algorithm 2).

Algorithm 2 Algorithm BGRF. Input is a complete bipartite graph G and time limit T.

```
1: if $T$ is even then
2:     At time 0, do nothing.
3:     At time 1, move to any destination node (i.e. any node in $V_2$). Do nothing if already at a destination node.
4:     At every even time, determine which released request earns the greatest revenue and move to the source location of this request. Denote this request as $r$. If no unserved requests exist, do nothing until the next even time.
5:     At every odd time, serve request $r$ (if it exists) from the previous step.
6: end if
7: if $T$ is odd then
8:     At time 0, move to any destination node (i.e. any node in $V_2$). Do nothing if already at a destination node.
9:     At every odd time, determine which released request earns the greatest revenue and move to the source location of this request. Denote this request as $r$. If no unserved requests exist, do nothing until the next odd time.
10:    At every even time, serve request $r$ (if it exists) from the previous step.
11: end if
```

Proposition 1. *Algorithm* BGRF *is 1-competitive for Complete Bipartite ROLDARP.*

Proof. We will show that given request sequence σ and input graph G, if $\text{OPT}(G, \sigma)$ denotes the optimal revenue earned from σ and $\text{BGRF}(G, \sigma)$ denotes the amount of revenue earned by BGRF from σ, then:

$$\text{OPT}(G, \sigma) \leq \text{BGRF}(G, \sigma) + v_{last} \tag{10}$$

where v_{last} is the revenue of the last request served by OPT.

We now prove Equation (10). We consider two cases based on the parity of T:

Case 1: T is odd.
As in Section 5, we consider another algorithm MAX such that for any optimal algorithm OPT, MAX serves all except the last request served by OPT, but in a different order. Specifically, for $t < T-1$, when OPT serves a request at time t, MAX serves the request with the greatest revenue that has been released by time $t+1$ that OPT eventually serves. In other words, out of all the requests released by $t+1$ that OPT serves, MAX serves the one with the greatest revenue. Note that for any request sequence σ:

$$\text{MAX}(G,\sigma) = \text{OPT}(G,\sigma) - v_{last} \tag{11}$$

where v_{last} is the last revenue earned by OPT.

We show that every time MAX earns a revenue, BGRF earns at least as much revenue two time units later. We assume without loss of generality that there are enough requests such that at every time unit, MAX and BGRF have a request to serve. Note that any algorithm requires at least two time units to serve each request after the first request — the first time unit for moving to the source of the request and the second time unit for moving to the destination. Therefore MAX and BGRF serve requests at every other time unit. Specifically, MAX serves at every even t for $t < T-1$ and BGRF serves at every even t except $t = 0$.

Assume at some time t MAX serves r^* and earns revenue v^*. We show by contradiction that r^* must be an available request for BGRF to serve at $t+2$. Since r^* must have been released by $t+1$, it must be available for BGRF to serve at $t+2$ unless BGRF has already served it prior to $t+1$. Suppose that BGRF served it prior to $t+1$ at some time t'. Then r^* must have been the highest revenue request released by t'. But this implies that MAX would have served r^* by t' which is a contradiction since MAX serves r^* at time $t \geq t'$.

Thus by every odd time $t = 1, 3, 5, \ldots T-2$, MAX has served one more request than BGRF and each request BGRF serves earns at least as much revenue as the request served by MAX two time units earlier. Finally, at $T-1$, BGRF will serve the last request that MAX serves. So for any request sequence σ, $\text{BGRF}(G,\sigma) \geq \text{MAX}(G,\sigma)$. Combining this with Equation (11) proves Equation (10).

Case 2: T is even.
A few modifications to the odd case will prove the even case. For the even case, we consider a modified version of OPT, $\overline{\text{OPT}}$, which we describe below.

First note that any optimal algorithm requires one time slot to serve the first request and two time slots to serve every additional request, so in total, an odd number of time slots. Therefore, if T is even, one time slot will serve no purpose so the algorithm can simply wait for more requests to be released during this time slot. The optimal choice is to use the first time slot (i.e. $t = 0$ to $t = 1$) to wait as this allows the maximum number of requests to be released from which the algorithm can choose. Therefore, any optimal algorithm that does not wait during the first time slot can be converted to an equivalent algorithm that does. Specifically if OPT is an optimal algorithm that waits after the first time slot, we can convert OPT to an equivalent algorithm $\overline{\text{OPT}}$ by shifting the wait to the first time slot and then performing the remaining moves of OPT in the same order as OPT. Now both $\overline{\text{OPT}}$ and BGRF wait during the first time slot. As in the odd case, we consider an algorithm MAX that serves the same requests as $\overline{\text{OPT}}$ (instead of OPT). Now we can apply the proof for the odd case by simply replacing odd for even and even for odd.

6.2 Single-Source ROLDARP

In this section, we consider a modified version of ROLDARP where there is a single source node, S, which is the source of every request and within unit distance of every other node in the graph. We present an algorithm SGRF (Single Greatest Revenue First) that is optimal for this problem (see Algorithm 3).

Proposition 2. *Algorithm* SGRF *is equivalent to an optimal offline solution for Single-Source ROLDARP.*

Proof. We can assume without loss of generality, that the origin is S: since *any* algorithm will need to move to S to serve any request, an instance of the problem where the origin is not S is equivalent to an instance where the origin is S and T is decremented by 1.

We consider an optimal offline algorithm OPT and prove by way of contradiction that the set of requests served by SGRF earns as much revenue as the set of requests served by OPT.

Algorithm 3 Algorithm SGRF. Input is graph G, time limit T, and the source node S.

```
1: if $T$ is even then
2:     At every odd time, serve the request with the greatest revenue.
3:     At every even time, move to $S$.
4: end if
5: if $T$ is odd then
6:     At every even time, serve the request with the greatest revenue.
7:     At every odd time, move to $S$.
8: end if
```

Let R denote the set of requests served by OPT, where r_i denotes a request with revenue v_i. We consider a set R^* that is a rearrangement of the requests from R. Let r_j and r_k denote two requests served by OPT where both r_j and r_k were released by some time t. Then in R^*, r_j and r_k will be ordered such that the request with the higher revenue appears first, i.e. r_j followed by r_k if $v_j > v_k$ or r_k followed by r_j if $v_k \geq v_j$. In other words, R^* is the set of requests of R such that if any request in R can be swapped with another request with a higher revenue, then this swap occurs in R^*.

Note that since any algorithm must move back to S after completing a request, serving any request takes exactly two time units, so rearranging the requests in R as described will not affect the overall time required. Therefore every request in R also appears in R^* so the two sets earn equal amounts of revenue.

Now, suppose, by contradiction, that in R^* there is some request r with revenue v served by OPT at time t but not served by SGRF.

Case 1: T is even.
Suppose t is odd. Since at every odd time, SGRF serves the request with the highest revenue, at time t SGRF must earn revenue at least v. If t is even, then there are at least 2 more time units remaining (i.e. $T \geq t+2$), so SGRF can serve r from $t+1$ to $t+2$; OPT would not be able to serve another request during this time slot as it would need to use this time to move back to s.

Case 2: T is odd.
Simply replace odd with even and even with odd in the proof for Case 1.

7 Experimental Analysis

7.1 Original ROLDARP

To evaluate the performance of our ROLDARP algorithm, we simulated a realistic Online Dial-a-Ride system. Our experimental settings were based on data we retrieved from real-world Dial-a-Ride systems ([19, 21, 23]). We first consider a uniform setting where requests are issued with equal probability throughout the day and every location is equally likely to be chosen as a source or destination. We then consider a more realistic setting where there are peak time periods, when requests are more likely to be released, and certain locations are are more likely than others to be chosen as a source or destination. Specifically, we consider the following settings:

1. Requests are released uniformly over time and locations are chosen to be the source or destination with uniform probability
2. There are peak times but locations are chosen to be the source or destination with uniform probability
3. Requests are released uniformly over time but locations are chosen to be the source or destination with non-uniform probability
4. There are peak times and locations are chosen to be the source or destination with non-uniform probability

The specifics of our experimental settings are as follows. The graph is complete and contains 50 nodes and unit edge weights, so the travel time between every pair of nodes is equal to a time unit. We consider both even and odd time limits. For the case with even time limit, a time unit is 10 minutes and the time spans 18 hours, from 6:00am to 12:00am, so the time limit $T = 108$. For the case with odd time limit, a time unit is 12 minutes and the time spans 17 hours, from 6:00am to 11:00pm, so the time limit $T = 85$. For both cases, we consider the four settings described above. In setting (1), the completely uniform setting, 1-5 requests are issued during every time unit and every location in the graph is equally likely to be chosen as either the source or destination of a request (such that the same location is not chosen as *both* the source and destination). In setting (2), the hours between 7:00am-9:00am, 12:00pm-1:00pm, and 5:00pm-7:00pm are considered peak hours. During these times, 10-15 requests are issued every time unit. In setting (3), there are no peaks hours, but a small subset of the locations (5 out of 50) have a 10% chance of being chosen as a source or destination while all other nodes have a 1.1% chance of being chosen. Finally in setting (4), there are peak hours and nodes are chosen with this probability distribution. In all four settings the request revenues are uniformly distributed from \$5 to \$20.

We compare our algorithm, GRF (see Algorithm 1), to the "ideal" revenue earning – that is, the amount of revenue that would be earned if a maximum revenue request (\$20) was served at every time unit. We note that even an optimal offline algorithm may not be able to earn the ideal amount, if for example, there were some time units where there were no maximum revenue requests to be served.

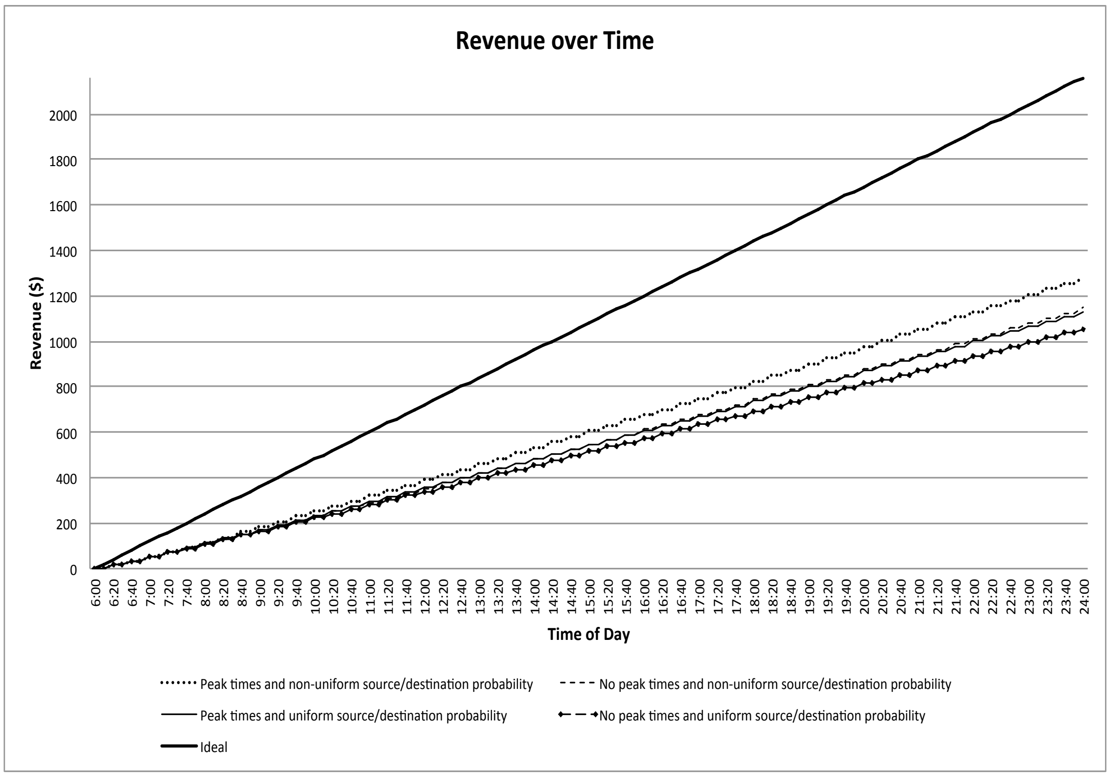

Fig. 1: Revenue earned over time for even time limit.

Figures 1 and 2 show the results of the experimental simulations. The graphs show that for both even and odd time limit, compared to this idealistic algorithm (that performs better than an optimal algorithm), GRF is 2-competitive (with b set to the revenue of the last request). Specifically, for the even case, the ideal revenue is 2160, while GRF earns on average 1056, 1131, 1148, and 1282, respectively, for settings 1-4 described above. For the odd case, the ideal revenue is 1700, while GRF earns on average 817, 872, 881, and 980, respectively, for settings 1-4. The graphs also show that for both even and odd time limits, GRF performs its best under the most realistic setting, i.e. when there are peak times and locations are chosen to be the source or destination with non-uniform probability (setting 4), earning almost an average of 60% of the ideal revenue (specifically, 59.4% when the time limit is even and 57.6% when the time limit is odd). The reason for this improved performance is straightforward: there are more requests released earlier in the day and many pairs of requests such that the destination of one request is the source of the next request. Therefore GRF is able to serve more requests consecutively without having to spend time moving from the destination of one request to the source of the next.

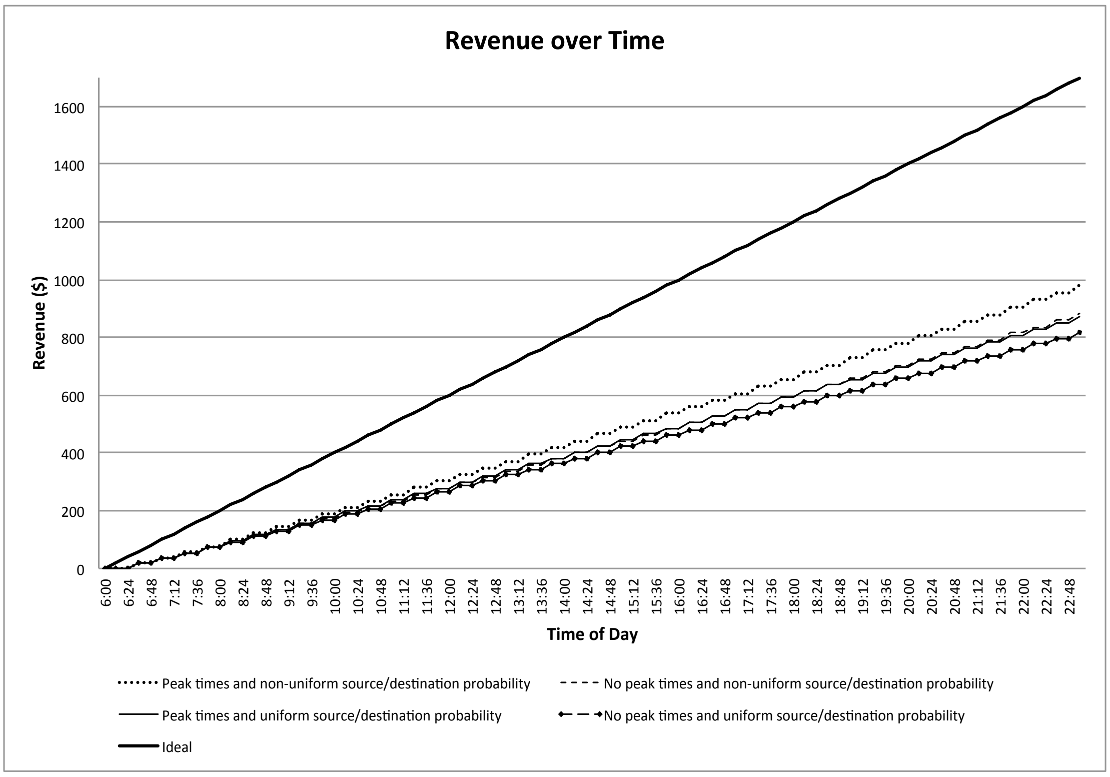

Fig. 2: Revenue earned over time for odd time limit.

7.2 Complete Bipartite ROLDARP

We also simulated our algorithm for complete bipartite graphs (we omitted simulations for single source graphs since our algorithm was optimal for this case). We use the same experimental settings as in 7.1 with the following exceptions: (1) 5 out of 50 nodes are designated as source nodes while all other nodes are destinations nodes, (2) the graph is complete bipartite so edges exist only between every pair of nodes such that one node is a source and the other is a destination; no edges exist between source nodes and no edges exist between destination nodes. We consider a setting similar to (2) in 7.1 where the hours between 7:00am-9:00am, 12:00pm-1:00pm, and 5:00pm-7:00pm are considered peak hours (we also tested our algorithm under a setting with no peak times and found that it yielded similar results). During these times, 10-15 requests are issued every time unit whereas 1-5 requests are issued during all other times. Each source and destination is equally likely to be chosen for a request. We compare our algorithm, BGRF (see Algorithm 2), to the optimal algorithm – that is, the algorithm that alternates between serving the highest revenue request available and moving to the source of the highest revenue request available.

Figures 3 and 4 show the results of the experimental simulations. The graphs confirm that our algorithm BGRF is 1-competitive (with b set to the revenue of the last request). Specifically, for the even case, the optimal revenue is on average 1057, while BGRF earns on average 1042. For the odd case, the ideal revenue is on average 841, while GRF earns on average 825. In both cases, BGRF earns over 98% of the optimal revenue.

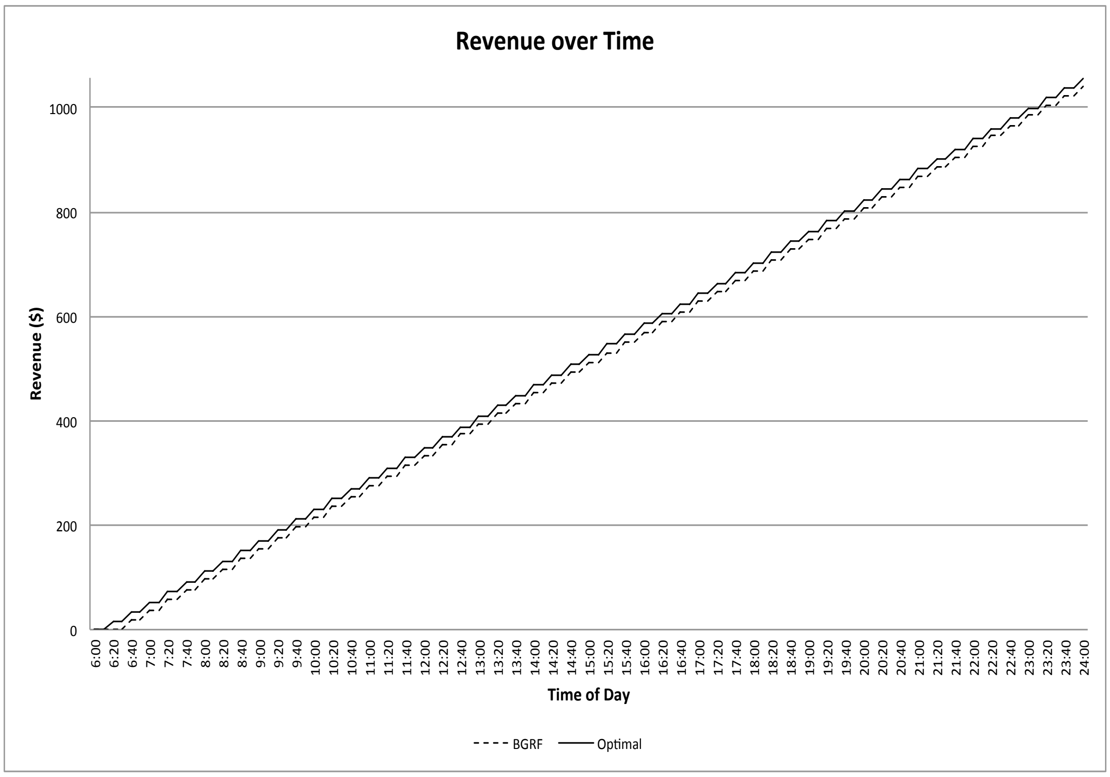

Fig. 3: Revenue earned over time for even time limit for a complete bipartite graph.

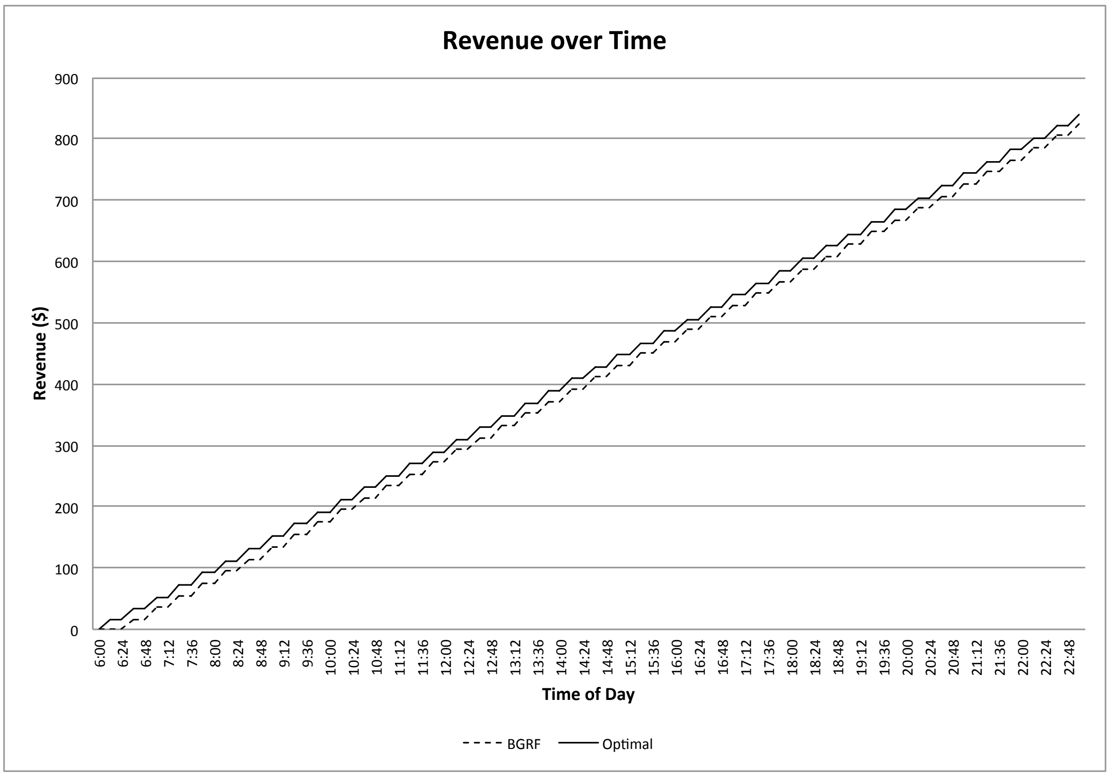

Fig. 4: Revenue earned over time for odd time limit for a complete bipartite graph.

8 Conclusion

We studied ROLDARP and variations of this problem. For ROLDARP we proved that deterministic competitive algorithms do not exist for complete graphs with varying edge weights nor for non-complete graphs. For complete graphs with unit edge weights, we presented a 2-competitive algorithm to solve this problem. For ROLDARP on complete bipartite graphs, we presented a 1-competitive algorithm. Finally, for ROLDARP on graphs with a single source vertex, we presented an optimal online algorithm. We also simulated our algorithms for complete graphs and complete bipartite graphs under experimental settings that were based on data we retrieved from real-world Dial-a-Ride systems. Our simulations support our theoretical findings and demonstrate that our algorithms perform well under settings that reflect realistic Dial-a-Ride settings.

Obviously improving the competitive ratio for original ROLDARP or proving a lower bound of 2 would be interesting extensions of our work. Some other open problems involve incorporating modifications that would make the problem more reflective of realistic settings. For example, we can consider a server that can serve multiple requests at a time or a setting with multiple servers. We can also associate with each request a penalty that is incurred if the request is not served; one goal would be to earn a specified amount of revenue while minimizing the penalties of unserved requests.

References

1. N. Ascheuer, S. Krumke, and J. Rambau, “Online Dial-a-Ride problems: minimizing the completion time,” in *Proceedings of the 17th International Symposium on Theoretical Aspects of Computer Science, Lecture Notes in Computer Science*, vol. 1770, pp. 639-650, 2000.
2. G. Ausiello, E. Feuerstein, S. Leonardi, L. Stougie, and M. Talamo, “Algorithms for the On-Line Traveling Salesman,” in *Algorithmica*, vol. 29, issue 4, pp. 560-581, 2001.
3. G. Ausiello, V. Bonifaci, and L. Laura, “The On-Line Prize-Collecting Traveling Salesman Problem,” in *Information Processing Letters*, vol. 107, issue 6, pp. 199-204, 2008.
4. N. Azi, M. Gendreau, and J. Potvin, “A dynamic vehicle routing problem with multiple delivery routes,” in *Annals of Operations Research,* vol. 199, issue 1, pp. 103 - 112, 2012.
5. A. Blum, P. Chalasani, D. Coppersmith, B. Pulleyblank, P. Raghavan, and M. Sudan, “The minimum latency problem,” in *Proceedings of the 26th annual ACM Symposium on Theory of Computing,* pp. 163-171, 1994.
6. A. Christman and W. Forcier, “Maximizing revenues for On-line Dial-a-Ride,” in *Combinatorial Optimization and Applications*, pp. 522-534, 2014.
7. G.N. Frederickson, M.S. Hecht, and C.E. Kim, “Approximation algorithms for some routing problems”, in *Journal of Computing*, vol. 7, pp. 178193, 1978.
8. M. Gendreau, A. Hertz, and G. Laporte, “A tabu search heuristic for the vehicle routing problem,” in *Management Science*, vol. 40, issue 10, pp. 1276-1290, 1994.
9. D.J. Guan, “Routing a vehicle of capacity greater than one,” in *Discrete Applied Mathematics*, vol. 81, issue 1, pp. 41-57, 1998.
10. P. Jaillet and X. Lu, “Online Traveling Salesman problems with flexibility,” in *Networks*, vol. 58, pp. 137-146, 2011.
11. P. Jaillet and M. Wagner, “Generalized online routing: new competitive ratios, resource augmentation and asymptotic analyses,” in *Operations Research*, vol. 56, issue 3, pp. 745-757, 2008.
12. P. Jaillet and M. Wagner, “Online vehicle routing problems: A survey,” in *The Vehicle Routing Problem: latest advances and new challenges*, pp. 221-237, 2008.
13. Y. Kergosien, C. Lente, D. Piton, and J-C. Billauta, ”A tabu search heuristic for the dynamic transportation of patients between care units,” *European Journal of Operational Research*, vol. 214, no. 2, pp. 442-452, 2011.
14. S. Krumke, “On minimizing the maximum flow time in the Online Dial-a-Ride Problem,” in *Networks*, vol. 44, pp. 41-46, 2004.
15. C. Liao and Y. Huang,“The covering Canadian traveller problem,” in *Theoretical Computer Science*, vol. 530, pp. 80-88, 2014.
16. S. Lorini, J-Y. Potvin, and N. Zufferey, “Online vehicle routing and scheduling with dynamic travel times,” in *Computers and Operations Research*, vol. 38, issue 7, pp. 1086-1090, 2011.
17. W. de Paepe, J. Lenstra, J. Sgall, A. Sitters, and L. Stougie. “Computer-aided complexity classification of dial-a-ride problems,” in *INFORMS Journal on Computing 16*, no. 2 pp. 120-132, 2004.
18. City of Plymoth Minnesota, “Plymouth Metrolink Dial-A-Ride,” http://www.plymouthmn.gov/departments/administrative-services-/transit/plymouth-metrolink-dial-a-ride.
19. M. Schilde, K. Doerner, and R. Harti, “Metaheuristics for the dynamic stochastic dial-a-ride problem with expected return transports,” in *Computers and Operations Research*, vol. 38, no. 12, pp. 1719-1730, 2011.

20. Stagecoach Corporation, "Dial-A-Ride," http://stagecoach-rides.org/dial-a-ride/.
21. L. Stougie and E. Feuerstein, "On-Line single-server Dial-a-Ride problems," in *Theoretical Computer Science*, vol. 268, issue 1, pp. 91-105, 2001.
22. Metropolitan Council, "Transit Link: Dial-a-ride small bus service," https://metrocouncil.org/Transportation/Services/Transit-Link.aspx.
23. University of Washington, "Dial-A-Ride," http://www.washington.edu/facilities/transportation/uwshuttles/dar.
24. X. Wen, Y. Xu, and H. Zhang, "Online traveling salesman problem with deadline and advanced information," in *Computers & Industrial Engineering*, vol. 63, no. 4, pp. 1048 - 1053, 2012.