Maximizing Revenues for On-Line Dial-a-Ride

Ananya Christman[1(✉)] and William Forcier[2]

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

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.

Keywords: Online algorithms · Dial-a-ride · Competitive analysis · Graphs

1 Introduction

In the On-Line Dial-a-Ride Problem (OLDARP), a server travels in some metric space to serve requests for rides. The server has a *capacity* that specifies the maximum number of requests it can serve at any time. The server starts at a designated location of the space, the *origin*, and moves along the space to serve requests. Requests arrive dynamically and 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 the release time of the request, which is the earliest time the request may be served. For each request, the server must decide whether to serve the request and at what time, with the goal of meeting some optimality criterion. 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 (or multiple vehicles) 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, and courier services.

Z. Zhang et al. (Eds.): COCOA 2014, LNCS 8881, pp. 522–534, 2014.
DOI: 10.1007/978-3-319-12691-3_38

In the version of OLDARP that we consider, there is a global time limit such that requests must be served before this time, and each request has an associated revenue, which is the amount earned by the server for serving the request. In the context of the applications described above, the revenue can represent the priority levels of the requests. We assume that the server can serve at most one request at a time (in other words the server has unit capacity). Such is the case for ambulance routing and many taxi services. The goal is to serve requests within the time limit so as to maximize the total revenue. We assume preemption is not allowed, therefore serving a particular request may prevent the server from serving a higher revenue request that arrives later. We refer to this problem as ROLDARP (Revenue On-Line Dial-A-Ride Problem) and describe it formally in Sect. 3.

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 Sect. 2 we describe several works related to our problem. In Sect. 3 we formally define ROLDARP and describe the variations: V-ROLDARP (where edge weights are varying), complete-bipartite ROLDARP, and single-source ROLDARP. In Sect. 4 we prove that no deterministic online algorithm can be competitive for V-ROLDARP. In Sect. 5 we present a 2-competitive algorithm for ROLDARP. In Sect. 6.1 we present a 1-competitive algorithm for complete bipartite ROLDARP and in Sect. 6.2 we present an optimal algorithm for single-source ROLDARP. Finally in Sect. 7 we summarize our results and discuss some possible extensions.

2 Related Work

Several variations of the On-Line Dial-a-Ride Problem have been studied in the past. The authors of [10] studied the problem for the unit metric space with two different objectives. One is to minimize the time to serve all requests and

return to the origin (also known as *completion time*); the other is to minimize the average completion time of 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.

The authors of [1] studied minimizing total completion time for OLDARP with multiple servers and capacity constraints and present a 2-competitive algorithm for this problem. For the version of the problem with one server and no capacity constraints, they presented a $1 + \sqrt{(1 + 8\rho)}/2$-competitive algorithm, where ρ is the approximation ratio of a related offline problem.

The work in [9] considered a modified version of OLDARP where at the release time of a request only the source location is revealed. The destination location is revealed only when the server arrives at the source. Such a setting is appropriate for applications such as elevator scheduling or ride scheduling for taxis. The authors proved that for minimizing completion time, when preemption is allowed (i.e. the server is allowed to halt a ride at any time and possibly proceed with it later) any deterministic algorithm must have competitive ratio of at least 3. They also gave a 3-competitive algorithm to solve this problem.

The work in [7] 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 non-polynomial 2-competitive algorithm for minimizing completion time.

The authors of [2] considered a version of the problem where each request consists of a single location and a release time. This is referred to as the On-Line Traveling Salesman Problem (OLTSP). 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.

The authors of [3] also aimed to minimize completion times for OLTSP but considered an asymmetric network where the distance from one point to another may differ in the inverse direction. They considered two versions of the problem: *homing*, where the server is required to return to the origin after completing the requests, and *nomadic*, where there is no such requirement. They presented a non-polynomial $(3 + \sqrt{5})/2$-competitive algorithm for the homing version and proved that this is the best possible. For the nomadic version, they proved that no general online competitive algorithm can exist for this problem; instead the competitive ratio for any online algorithm must depend on the asymmetry of the space.

The work in [5] considers a version of OLTSP where edge costs change over time. They prove upper and lower bounds on the competitive ratio and provide a competitive ratio that is based on the minimum and maximum edge costs. However, their competitive algorithms require a solution to a related NP-hard problem.

The authors of [8] studied both OLDARP and OLTSP for the uniform metric space. Their objective is to minimize the maximum *flow time*, the difference between a request's release and service times. They proved that no competitive algorithm exists for OLDARP and gave a 2-competitive algorithm to solve OLTSP.

More recently, the authors of [6] considered a variation of OLTSP where each request also has a penalty (incurred if the request is rejected). The goal is 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.

The authors of [4] studied a variation of OLTSP where each request has both a penalty and a weight. The goal is 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.

To our knowledge, this is the first work that studies OLDARP with the goal of maximizing total revenue within a time limit. The related works of [1,7–10] study OLDARP, however none consider requests with revenues. Although in the work of [4], requests have revenues (their paper refers to them as "weights"), each request has only one location, whereas requests for our problem consist of both a source and a destination.

3 Problem Statement

In the basic form of the *On-Line Dial-a-Ride Problem* (OLDARP), a network server receives requests dynamically and each request has a source, destination, and release time. The server starts at a predefined origin location and serves a request by picking up at the source and delivering at the destination. The server can serve only one request at a time and cannot serve a request prior to its release time. We consider a version of the problem where there is a given time limit by which the server must serve all requests and every request has a revenue that the server earns for fulfilling the request. The goal is to serve requests within the time limit so as to maximize the total revenue.

We study competitive algorithms for this problem and use standard terminology from competitive analysis. An algorithm 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 OPT denote the optimal offline algorithm, as in the algorithm that given any input will earn the greatest revenue of any other algorithm on that input.

Given an input graph G, a sequence $\sigma = r_1, \ldots r_m$ of requests and an algorithm ALG, we denote $\text{ALG}(G, \sigma)$ as the total revenue earned by ALG from σ on G. We say that ON is *c-competitive* if there exists $c > 0, b \geq 0$ such that for all σ:

$$\text{OPT}(G, \sigma) \leq c \cdot \text{ON}(G, \sigma) + b \tag{1}$$

The input to the problem 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. For every edge $(u, v) \in E$, there is a weight $w_{u,v} > 0$. If for every edge $w_{u,v} = 1$ (i.e. the graph represents the unit metric space), we refer to the problem as ROLDARP. If edge weights are varying, we refer to the problem as V-ROLDARP. 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, r) where s is the source node, d is the destination, t is the time the request is released, and r is the revenue 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. The total time is equal to the length of the path from x to d. 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. 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. As a preprocessing step, we can remove any edge (u, v) such that $w_{u,v} > T$, since no algorithm (either online or offline) can use this edge to serve a request.

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 Sect. 5 we consider ROLDARP and provide an algorithm that is 2-competitive when the additive constant b from Eq. (1) is the revenue of the last request served by OPT, v_{last}. In this section, we prove that no algorithm can be competitive for V-ROLDARP for any $b \leq v_{last}$[1]). 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 $\alpha \geq 1, b \leq v_{last}$, $\alpha \cdot \text{ON}(G, \sigma) + b < \text{OPT}(G, \sigma)$ 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 Eq. (1) must depend on the input (in particular, the last request served by 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, $\text{OPT}(G, \sigma) > \text{ON}(G, \sigma) + b$. This also shows that no deterministic online algorithm can be competitive when $b < v_{last}$.

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.

[1] 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.

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:

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

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 Eq. (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 1.1 describes the GRF algorithm.[2] 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.

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

Proof. We now prove Eq. 2 to show that GRF is 2-competitive. We assume without loss of generality, that a request is issued at every time unit[3]. To prove Eq. 2, we consider another algorithm MAX. We define MAX such that at every time unit

[2] 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.

[3] 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 1.1. Algorithm GRF. Input is complete graph G and time limit T.

if T is even **then**
  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.
  At every odd time, serve request r (if it exists) from the previous step.
end if
if T is odd **then**
  At time 0, do nothing.
  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.
  At every even time, serve request r (if it exists) from the previous step.
end if

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 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 Eq. (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 $r_0, r_1, r_2, \ldots r_{T-2}$ denote the requests served by MAX at times $0, 1, 2, \ldots T-2$ earning revenues $v_0, v_1, v_2, \ldots v_{T-2}$. 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 Eq. (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

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

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

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 1.2).

Algorithm 1.2. 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 Eq. 10. We consider two cases based on the parity of T:

Case 1: T is odd.
As in Sect. 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 Eq. 11 proves Eq. 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 1.3).

Algorithm 1.3. 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**

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.

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 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.

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. Ascheuer, N., Krumke, S.O., Rambau, J.: Online dial-a-ride problems: minimizing the completion time. In: Reichel, H., Tison, S. (eds.) STACS 2000. LNCS, vol. 1770, pp. 639–650. Springer, Heidelberg (2000)
2. Ausiello, G., Feuerstein, E., Leonardi, S., Stougie, L., Talamo, M.: Algorithms for the on-line traveling salesman. Algorithmica **29**(4), 560–581 (2001)
3. Ausiello, G., Bonifaci, V., Laura, L.: The on-line asymmetric traveling salesman problem. J. Discrete Algorithms **6**(2), 290–298 (2008)
4. Ausiello, G., Bonifaci, V., Laura, L.: The on-line prize-collecting traveling salesman problem. Inf. Process. Lett. **107**(6), 199–204 (2008)
5. Broden, B., Hammar, M., Nilsson, B.: Online and offline algorithms for the time-dependent TSP with time zones. Algorithmica **39**(4), 299–319 (2004)
6. Jaillet, P., Lu, X.: Online traveling salesman problems with flexibility. Networks **58**, 137–146 (2011)
7. Jaillet, P., Wagner, M.: Generalized online routing: new competitive ratios, resource augmentation and asymptotic analyses. Oper. Res. **56**(3), 745–757 (2008)
8. Krumke, S.: On minimizing the maximum flow time in the online dial-a-ride problem. Networks **44**, 41–46 (2004)
9. Lipmann, M., Lu, X., de Paepe, W.E., Sitters, R.A., Stougie, L.: On-line dial-a-ride problems under restricted information model. Algorithmica **40**, 319–329 (2004)
10. Stougie, L., Feuerstein, E.: On-line single-server dial-a-ride problems. Theor. Comput. Sci. **268**(1), 91–105 (2001)