Operations Research Forum _####################_
https://doi.org/10.1007/s43069-021-00076-x

ORIGINAL RESEARCH

Improved Bounds for Revenue Maximization in Time-Limited Online Dial-a-Ride

Ananya D. Christman[1] · Christine Chung[2] · Nicholas Jaczko[1] · Tianzhi Li[1] · Scott Westvold[1] · Xinyue Xu[1] · David Yuen[3]

Received: 17 February 2021 / Accepted: 8 June 2021

Abstract

In the Online Dial-a-Ride Problem (OLDARP), a server travels to serve requests for rides. We consider a variant where each request specifies a source, destination, release time, and revenue that is earned for serving the request. The goal is to maximize the total revenue earned within a given time limit. We first prove that no non-preemptive deterministic online algorithm can be guaranteed to earn more than half the revenue earned by OPT. We then investigate the SEGMENTED BEST PATH (SBP) algorithm (of Christman et al. in Revenue maximization in online dial-a-ride 2017, [1]). The previously established lower and upper bounds for the competitive ratio of SBP are 4 and 6, respectively, under reassumptions about the input instance. We eliminate the gap by proving that the competitive ratio is 5 (under the same assumptions) and also prove that this bound is tight. When revenues are uniform, we prove that SBP has competitive ratio 4. Next we provide a competitive analysis of SBP on complete bipartite graphs. We then consider this problem on the uniform metric and revisit the BP algorithm (of Christman et al. in Revenue maximization in online dial-a-ride 2017, [1]); we provide an instance where the algorithm's competitive ratio is unbounded. We conclude with experimental results that suggest that SBP would be effective if applied in practice.

1 Introduction

In the Online Dial-a-Ride Problem (OLDARP), a server travels through a graph 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,

A preliminary conference paper containing a portion of these results appeared in [2]

✉ Ananya D. Christman
achristman@middlebury.edu

[1] Middlebury College, Middlebury VT 05753, USA

[2] Connecticut College, CT 06320 New London, USA

[3] 92-1507 Punawainui St, Kapolei, HI 96707, USA

and the *release time*, which is the earliest time the request may be served. Requests arrive over time; specifically, each arrives at its release time and the server must decide whether to serve the request and at what time, with the goal of meeting some optimality criterion. The server has a *capacity* that specifies the maximum number of requests it can serve at any time. Common optimality criteria include minimizing makespan (i.e., time the server has completed the last request), minimizing the average flow time (i.e., the difference in a request's completion and release times), or maximizing the number of served requests within a specified time limit. In many variants *preemption* is not allowed, so if the server begins to serve a request, it must do so until completion. Online 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 including Ride-for-Hire systems (such as Uber and Lyft), and courier services.

We study a variation of OLDARP 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 server has unit capacity and the goal for the server is to serve requests within the time limit so as to maximize the total priority. A request's priority may represent the importance of serving the request in settings such as courier services. 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, a request's priority may represent the revenue earned from serving the request. For the remainder of this paper, we will refer to the priority as "revenue," and to this variant of the problem as ROLDARP. Note that if revenues are uniform the problem is equivalent to maximizing the number of served requests.

1.1 Related Work

The Online Dial-a-Ride problem was introduced by Feuerstein and Stougie [3] 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 [4] and [5]. There are two versions of the problem: *closed* where the server must return to the origin after serving requests, and *open* where the server need not do so. Many objectives have been considered for these problems and the terminology varies in the related studies. For consistency, we will use the terminology introduced in [3] where *makespan* refers to the time the server has completed the last request (and returned to the origin in the closed version) and *latency* refers to the average *completion* time of requests (where completion time is the time at which the server arrives at the destination of the request). Feuerstein and Stougie studied the closed version for these two objectives. For minimizing makespan, 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 lower bounds of 3 and $1 + \sqrt{2}$ when the server has capacity 1 and greater than 1, respectively. They also presented a 15-competitive algorithm on the real line for infinite

capacity. Several studies have considered the open version with the objective of minimizing makespan, and we list a few here. Krumke et al. presented a 3.41-competitive algorithm for the general metric space [6]. More recently, Birx et al. [7] presented a new upper bound of 2.67 for the SMARTSTART algorithm of [8] for the real line, which improves their previous bound of 2.94 [9] (the full version of this work is [10]). Bjelde et al. [11] present a preemptive algorithm with competitive ratio 2.41 on the real line. The Online Traveling Salesperson Problem (OLTSP), introduced by Ausiello et al. [12] is a special case of OLDARP where for each request the source and destination are the same location. Krumke et al. [13] studied both OLDARP and OLTSP for the uniform metric space with the objective of minimizing the maximum *flow time*, that is 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.

Also related to the problem we study is the Online Prize Collecting Traveling Salesman Problem (PCTSP) [14], where the server visits a set of cities where cities arrive over time and each city has a given prize and penalty. Ausiello et al. [15] presented a 7/3-competitive algorithm for this problem when the goal is to collect a given quota of prizes of cities and return to the origin while minimizing the length of the tour plus the penalties of the cities not in the tour.

In this paper, we study a variation of OLDARP that has relevance to modern-day on-demand transportation systems. Each request has a revenue that is earned if the request is served and the goal is to maximize the total revenue earned within a specified time limit; the offline version of the problem was shown to be NP-hard in [1]. More recently, it was shown that even the special case of the offline version with uniform revenues and uniform weights is NP-hard [16]. Christman et al. [17] presented a 2-competitive algorithm for this problem on graphs with uniform edge weights. In [1], we showed that if edge weights may be arbitrarily large, then regardless of revenue values, no deterministic algorithm can be competitive. We therefore considered graphs where edge weights are bounded by a fixed fraction of the time limit, (a natural subclass of inputs since in real-world dial-a-ride systems, drivers would be unlikely to spend a large fraction of their day moving to or serving a single request) and gave a 6-competitive algorithm. To our knowledge, despite its relevance to modern transportation systems, aside from the work in [1], the revenue-maximizing time-limited version of OLDARP on weighted graphs that we investigate in this paper has not been previously studied.

1.2 Our Results

In this work we begin with a general lower bound for ROLDARP; specifically, we show that no non-preemptive deterministic online algorithm can be better than 2-competitive. We then study the SEGMENTED BEST PATH (SBP) algorithm ([1, 2]), which currently yields the best competitive ratio for ROLDARP; we improve this ratio and prove that our new ratio is tight. Specifically, in [1], we showed that SBP's competitive ratio has lower bound 4 and upper bound 6, provided that the edge weights are bounded by a fixed fraction of the time limit, i.e., T/f where T is the time limit and $1 < f < T$, and that the revenue earned by the optimal offline solution

(OPT) in the last $2T/f$ time units is bounded by a constant. The first assumption is imposed because, as shown in [1], *no* non-preemptive deterministic online algorithm can be guaranteed to be constant-competitive for this problem if edge-weights are unbounded. Note that this assumption is natural in real-world systems where drivers wish to limit ride durations to some fraction, $1/f$, of the length of their day. The second assumption is imposed because, as we show in Lemma 1, *no* non-preemptive deterministic online algorithm can be guaranteed to earn this revenue. We note that as T grows, the significance of the revenue earned by OPT in the last two time segments diminishes.

In this work, we close the gap between the upper and lower bounds of SBP by providing an instance where the lower bound is 5 (Section 3.1) and a proof for an upper bound of 5 (Section 3.2). We note that another interpretation of our result is that under a weakened-adversary model where OPT has two fewer time segments available, while SBP has the full time limit T, SBP is 5-competitive. We then investigate the problem for uniform revenues (so the objective is to maximize the total number of requests served), which is a useful variant for settings where all requests have equal priorities such as not-for-profit services that provide transportation to elderly and disabled passengers and courier services where deliveries are not prioritized. We prove that SBP earns at least 1/4 the revenue of OPT, minus an additive term linear in f, the number of time segments (Section 4), which is nearly tight due to the lower bound of 4 from [1].

Next we consider the problem for complete bipartite graphs; for these graphs every source is from the left-hand side and every destination is from the right-hand side (Section 5). These graphs model the scenario where only a subset of locations may be source nodes and a disjoint subset may be destinations, e.g., in the delivery of goods from commercial warehouses only the warehouses may be sources and only customer locations may be destinations. We refer to this problem as ROLDARP-B. We first show that if edge weights are not bounded by a minimum value, then ROLDARP reduces to ROLDARP-B. We therefore impose a minimum edge weight of kT/f for some constant k such that $0 < k \leq 1$ (note this reflects real-world settings where pickup and drop-off locations are likely to be at least some minimum distance away from each other). We show that if revenues are uniform, SBP has competitive ratio $\lceil 1/k \rceil$. Finally, we show that if revenues are nonuniform SBP has competitive ratio $\lceil 1/k \rceil$, provided that the revenue earned by OPT in the last $2T/f$ time units is bounded by a constant, which is again justified by Lemma 1 which says no non-preemptive deterministic algorithm can be guaranteed to earn any fraction of what is earned by OPT in the last $2T/f$ time units. Table 1 summarizes our results on SBP.

We then consider ROLDARP on the uniform metric space and revisit the BESTPATH (BP) algorithm of [1]. We provide an instance where BP has an unbounded ratio (a surprising result as this algorithm appears to be "smarter" than a 2-competitive algorithm proposed in [17]).

Finally, we include experimental results on SBP and BP. For SBP, we extend our experiments from [1] to study the algorithm in more realistic settings where request release times and source-destination pairs follow nonuniform distributions. These settings were informed by interviews we had with representatives from real-world Dial-a-Ride organizations [18–20]. The results on BP show that although the

Table 1 Bounds on the SBP algorithm for ROLDARP variants. †This upper bound assumes the optimal revenue of the last two time segments is bounded by a constant as justified by Lemma 1. ‡This upper bound assumes the number of time segments is constant. ¶This bound is nearly tight as shown by the lower bound instance in [1]. ‖This bound is tight as shown by Theorem 2. § k is a constant where $0 < k \leq 1$ and is the ratio between the minimum and maximum edge weights in the graph

Summary of our results on SBP's competitive ratio ρ

	Uniform Revenue	Nonuniform Revenue
Weighted graphs	$\rho = 4^{\dagger\ddagger\P}$ (Thrm. 4)	$\rho = 5^{\dagger\|\|}$ (Thrm. 3)
Weighted bipartite graphs	$\rho \leq \lceil 1/k \rceil^{\S}$ (Thrm. 6)	$\rho \leq \lceil 1/k \rceil^{\dagger\S}$ (Thrm. 7)

algorithm has an unbounded competitive ratio, it outperforms the 2-competitive algorithm from [17] on a simulated ROLDARP environment.

2 Preliminaries

The Revenue Online Dial-a-Ride Problem (ROLDARP) is formally defined as follows. 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. For every edge $(u, v) \in E$, there is a weight $w_{u,v} > 0$, which represents the distance with the server traveling at unit speed. Alternatively, an edge weight can be interpreted as the time it takes to traverse the edge.[1] 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 are 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 (or priority) earned by the server for serving the request. The server does not know about a request until its release time t. To serve a request, the server must move from its current location x to s, then from s to d. The total time for serving the request is equal to the length (i.e., travel time) of the path from x to s to d, and the earliest time a request may be released is at $t = 0$. The output is a schedule of requests, i.e. a subset of requests and the time at which to serve each. A request may not be served earlier than its release time and at most one request may be served at any given time. The server is non-preemptive so once the server decides to serve a request, it must do so until completion. As in real-world systems (and described in [12]), the server may proceed to the source of a request with the "intention" of serving it, but until it begins serving the request it may "change its mind" and choose to move from that source location to another request instead.

[1] We note that 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.

The goal for the server is to serve requests within the time limit so as to maximize the total earned revenue. (The server need not return to the origin and may move freely through the graph at any time, even if it is not traveling to serve a request.)

2.1 General Lower Bound

We first present a general lower bound for ROLDARP and show that *no* non-preemptive deterministic online algorithm can be better than 2-competitive with respect to the revenue earned by the optimal offline schedule (ignoring the last two time segments; see Lemma 1, below).

Theorem 1 *No non-preemptive deterministic online algorithm for ROLDARP can be guaranteed to earn more than half the revenue earned by an optimal offline schedule in the first $T - 2T/f$ time units. This is the case for both the uniform and nonuniform revenue variants of ROLDARP.*

Proof Here we prove the theorem for the nonuniform variant; for the proof of the uniform variant, please see Section 1 of Appendix. Consider the following instance with $f = 5$ (so there are 5 time segments of length T/f). For simplicity, we let $X = T/f \geq 3$ denote the length of a time segment and therefore the maximum distance between two locations, so $T = 5X$. All distances are X unless otherwise stated. Let OPT denote an optimal offline schedule, let ON denote a deterministic online algorithm, and let s_0 denote the origin, i.e., the location of ON and OPT at time 0. Let $d(u, v)$ denote the distance between locations u and v. Let the graph consist of $s_0, a_i, b_i, c_i, d_i, e_i$ for $i = 0, 1, 2, 3, 4$. These nodes are all distance X from each other except:

1. $d(b_i, c_i) = 1, d(c_i, d_i) = X - 1$ $(i = 0, 1, 2)$
2. $d(b_i, c_i) = X - 2, d(c_i, d_i) = 1$ $(i = 3, 4)$

At time X, ON is either at a node s or heading to a node s. Pick two of a_0, a_1, a_2 that are not s. Because the graph is symmetric in indices 0, 1, 2, without loss of generality we can suppose a_1, a_2 are not s. The significance of this choice is that ON cannot have driven toward either a_1 or a_2 starting at a time $t_1 < X$. The adversary releases requests $r_1 = (a_1, b_1, X, \epsilon)$ and $r_2 = (a_2, b_2, X, \epsilon)$. Note that if no other requests are released, ON cannot earn any revenue unless it heads toward a_1 or a_2 at some time.

1. Case: ON has not yet made a drive toward a_1 or a_2 before time $2X + 1$. The adversary releases $r_3 = (b_1, e_1, 2X + 1, \epsilon)$ at this time. There is $3X - 1$ time remaining and that is insufficient to serve more than one of the requests. Thus ON earns at most ϵ while OPT earns 2ϵ via s_0, a_1, b_1, e_1 with time $3X$.
2. Case: ON moves from some node toward a_1 or a_2 at time t_1 with $2X \leq t_1 < 2X + 1$. We may assume w.l.o.g. ON is heading toward a_1. The adversary releases $r_3 = (c_2, d_2, 2X + 1, \epsilon)$. When ON arrives at a_1 at time $t_1 + X \geq 3X$, there is at most

$2X$ time remaining. There is insufficient time for ON to serve more than one request. Thus ON earns at most ϵ while OPT earns 2ϵ in time $3X$ via s_0, a_2, b_2, c_2, d_2.

3. Case: ON moves from some node toward a_1 or a_2 at time t_1 with $X < t_1 < 2X$. We may assume w.l.o.g. ON is heading toward a_1. Then the adversary releases request $r_3 = (b_2, e_2, 2X, \epsilon)$. After ON arrives at a_1 at time $t_1 + X > 2X$, there is strictly less than $3X$ time left and it is impossible for ON to serve two requests. Thus ON earns at most ϵ while OPT earns 2ϵ in time $3X$ via s_0, a_2, b_2, e_2.
4. Case: ON moves from some node at time $t_1 = X$ to either a_1 or a_2. We may assume w.l.o.g. ON moves to a_1 and arrives there at time $2X$. Then the adversary releases the requests: $r_3 = (a_3, b_3, X + 1, 1)$, $r_4 = (a_4, b_4, X + 1, 1)$.

   a) Case: ON has not moved toward any of b_1, a_3, a_4 before time $3X - 1$. So ON is at least X away from these three nodes. Because the graph is symmetric in indices 3, 4 we may suppose w.l.o.g. that ON is not at or heading toward c_3 at time $3X - 1$ (otherwise ON is not at or heading toward c_4). Then release $r_5 = (c_3, d_3, 3X - 1, 1)$. With time $2X + 1$ remaining, it is not possible for ON to serve two of the requests r_3, r_4, r_5 because it takes at least time $X - 2 + X + X + 1 = 3X - 1$ but the time remaining is $2X + 1 < 3X - 1$ (since $X > 3$). The most ON can possibly earn is $1 + \epsilon$ by serving one of r_3, r_4, r_5 along with one of r_1, r_2.
   b) Case: ON heads toward b_1 at time $2X \le t_2 < 3X - 1$. Then release $r_5 = (c_3, d_3, 3X - 1, 1)$. Then ON will be at b_1 at time $t_2 + X \ge 3X$. With at most $2X$ time remaining, it is clear that ON can serve at most one more request. Again, the most ON can earn is $1 + \epsilon$.
   c) Case: ON heads toward a_3 or a_4 at time $2X \le t_2 < 3X - 1$. We may assume w.l.o.g. that ON heads toward a_4. Then release $r_5 = (c_3, d_3, 3X - 1, 1)$. Then ON will be at a_4 at time $t_2 + X \ge 3X$. With at most $2X$ time remaining, it is impossible for ON to serve more than one request.

   In all subcases of Case 4, ON earns at most revenue $1 + \epsilon$. On the other hand, OPT serves r_3 and r_5 by traversing s_0,a_3, waiting time 1, and then traversing b_3, c_3, d_3, in total time $3X$ earning revenue 2, so $\frac{\text{OPT}}{\text{ON}} \ge 2/(1 + \epsilon)$.

We now show that *no* non-preemptive deterministic online algorithm can be competitive with the revenue earned by OPT in the last two time segments.

Lemma 1 *No non-preemptive deterministic online algorithm can be guaranteed to earn any fraction of the revenue earned by OPT in the last $2T/f$ time units. This is the case whether revenues are uniform or nonuniform.*

Proof Note that it is sufficient to construct an instance with uniform revenue that proves the claim. Let $dist(u_1, u_2)$ denote the distance between two locations u_1, u_2. Consider the following instance for $f \ge 3$ for some non-preemptive deterministic algorithm ALG.

Let k be a positive integer. Let the graph consist of nodes s_1, s_2, d, and $a_{i,j}$ ($i = 1, 2$ and $j = 0, 1, \ldots, k$). These nodes are all distance T/f from each other except $dist(a_{i,j}, a_{i,j'}) = |j - j'|T/(2kf)$. Let any of these nodes be the origin.

There are no requests released before time $T - 2T/f$. At time $T - 2T/f$, ALG is either at some node, call it b, or heading toward b.

Pick one of s_1, s_2 that is not b. Without loss of generality, let $s_1 \neq b$. Release the request $(s_1, d, T - 2T/f, 1)$. There are three cases.

1. Case: ALG is heading toward b at time $T - 2T/f$. Then it must arrive at b at a time greater than $T - 2T/f$. Then there is insufficient time to travel to s_1 and serve the request since doing so would take a total of $2T/f$ additional time. (It takes time at least T/f to travel to s_1 because s_1 is T/f distance from all other nodes.) No further requests are released in this case. So ALG earns zero while an OPT path goes from the origin to s_1 and waits there to serve the request during the time interval $[T - 2T/f, T - T/f]$, earning 1.
2. Case: ALG is at node b at time $T - 2T/f$, and does not immediately make an empty drive toward s_1. Then ALG can not serve this request by the same reasoning as in Case 1 in that it takes at least time T/f to move to s_1 along any path. No other requests are released in this case. As in Case 1, ALG earns zero while an OPT path earns 1 during the time interval $[T - 2T/f, T - T/f]$.
3. Case: ALG is at node b at time $T - 2T/f$ and immediately makes an empty drive toward s_1. Choose an $i \in \{1, 2\}$ such that $s_1 \notin \{a_{i,j} : j = 0, \ldots, k\}$. W.l.o.g. suppose $i = 1$ is chosen. Then the adversary releases the requests $(a_{1,j-1}, a_{1,j}, T - 1.5T/f, 1)$ for $j = 1, \ldots, k$. Then ALG arrives at s_1 at time $T - T/f$. Since moving from s_1 to any $a_{1,j}$ requires T/f, then there is insufficient time for ALG to serve any of these new requests that start at $a_{1,j-1}$. Thus ALG can earn at most 1 whereas an OPT path goes from the origin to $a_{1,0}$ at $T - 3T/f$ and waits there to serve along the path $a_{1,0}, \ldots, a_{1,k}$ during time interval $[T - 1.5T/f, T - T/f]$, earning revenue k.

In all three cases, ALG cannot earn any fraction of the revenue that an OPT path earns in the second to last time segment.

3 Nonuniform Revenues and Nonuniform Metric

We now consider ROLDARP with nonuniform revenues and nonuniform edge weights. In [1], we showed that if edge weights may be arbitrarily large, then no deterministic algorithm can be competitive. We therefore considered graphs where edge weights are bounded by a fixed fraction of the time limit, i.e., edge weights are at most T/f for some $1 < f < T$, which is a natural subclass of inputs since in real-world dial-a-ride systems, drivers would be unlikely to spend a large fraction of their day moving to or serving a single request. We presented an algorithm, SEGMENTED BEST PATH (SBP), that starts by splitting the total time T into f segments each of length T/f (please see Algorithm 1). At the start of a time segment, the server determines the *max-revenue-request-set*, i.e., the maximum revenue set of unserved

requests that can be served within one time segment, and moves to the source of the first request in this set. During the next time segment, it serves the requests in this set. It continues this way, alternating between moving to the source of the first request in the max-revenue-request-set during one time segment, and serving this request-set in the next time segment. To find the max-revenue-request-set, the algorithm maintains a directed auxiliary graph, G', to keep track of unserved requests (an edge between two vertices u,v represents a request with source u and destination v). It finds all paths of length at most T/f between every pair of nodes in G' and returns the path that yields the maximum total revenue (please refer to [1] for full details). Finding all paths of length at most T/f in G' requires enumeration of all paths in G' and the number of possible paths is exponential in the size of G', which is determined directly by the number of unserved requests in the current time segment. However, in many real world settings, the size of G' will be small relative to the size of G and in settings where T/f is small, the run time is further minimized. Therefore it should be feasible to execute the algorithm efficiently in many real-world settings.

Algorithm 1: Algorithm SEGMENTED BEST PATH (SBP). Input is complete graph G with time limit T and maximum edge weight T/f for $1 < f < T$

Let $t_1, t_2, \ldots t_f$ be the time segments ending at times $T/f, 2T/f, \ldots, T$, resp.
Let $i = 1$.
if f is odd **then**
    At t_1, do nothing. Increment $i = 2$.
end if
while $i < f$ **do**
    At the start of t_i, find the *max-revenue-request-set*, R.
    if R is non-empty **then**
        Move to the source location of the first request in R.
        At the start of t_{i+1}, serve request-set R.
    else
        Remain idle for t_i and t_{i+1}
    end if
    Let $i = i + 2$.
end while

We reconsider SBP since it has the current best upper bound for ROLDARP. Specifically, in [1] we proved that SBP is 6-competitive barring an additive factor equal to the revenue earned by OPT during the last two time segments. More formally, let $rev(\text{SBP}(t_j))$ and $rev(\text{OPT}(t_j))$ denote the revenue earned by SBP and OPT respectively during the j-th time segment. Then if $rev(\text{OPT}(t_f)) + rev(\text{OPT}(t_{f-1})) \leq c$, for some constant c, then $\sum_{j=1}^{f} rev(\text{OPT}(t_j)) \leq 6 \sum_{j=1}^{f} rev(\text{SBP}(t_j)) + c$. In [1], we also showed that as T grows, the competitive ratio of SBP is at best 4 (again with the additive term equal to $rev(\text{OPT}(t_f)) + rev(\text{OPT}(t_{f-1}))$), resulting in a gap between the upper and lower bounds.

In this section we improve the lower and upper bounds for the competitive ratio of the SEGMENTED BEST PATH algorithm [1, 2]. In particular, we eliminate the gap between the lower and upper bounds of 4 and 6, respectively, from [1], by

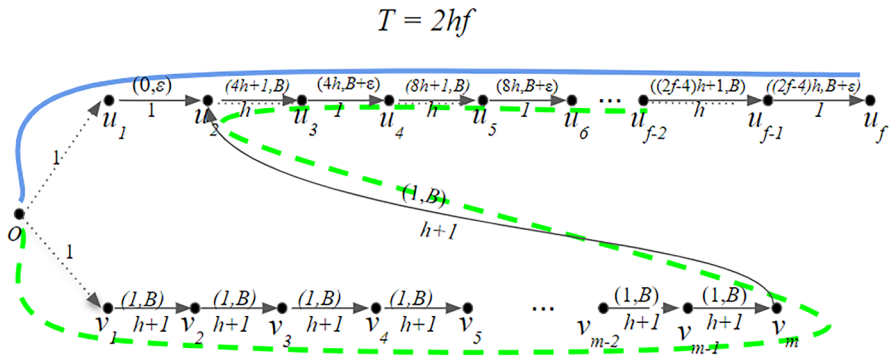

Fig. 1 An instance where OPT (whose path is shown in dashed green below) earns $5 - 4/(f-2)$ times the revenue of SBP (shown in solid blue above). In this instance, $T = 2hf$, and edges that represent requests are shown as solid edges. For each such edge the release time followed by revenue of the corresponding request is shown in parenthesis above the edge. The weight of an edge is shown below the edge. Dotted edges represent empty moves for SBP

providing an instance where the lower bound is 5 and a proof for an upper bound of 5. We study SBP because it has the best known competitive ratio for ROLDARP. Note that throughout this section we assume the revenue earned by OPT in the last two time segments is bounded by some constant since, as we showed in Lemma 1, *no* non-preemptive deterministic online algorithm can be guaranteed to earn any constant fraction of this revenue.

3.1 Lower Bound on SBP

Theorem 2 *If the revenue earned by OPT in the last two time segments is bounded by some constant, and SBP is γ-competitive, then $\gamma \geq 5$.*

Proof Consider the instance depicted in Figure 1. Fix $f > 2$ to be an even integer and fix $h > 1$ to be an integer. Set $T = 2fh$, so that the time segment length is $T/f = 2h > 1$. The distances in our instance will be either 1, h, or $1 + h$. Let $0 < \epsilon < 1$ be vanishingly small and let $B > 0$.

Let o be the origin, with other points in the graph being u_i for $i = 1, 2, \ldots, f$ and v_i for $i = 1, 2, \ldots, m$, where m will be determined below.

The idea is that SBP will take the path $o, u_1, \ldots, u_f$ in time T serving a single request of revenue $B + \epsilon$ every other time segment as prescribed by the algorithm. Meanwhile, discounting the revenue earned in the last two time segments, OPT will take the path $o, v_1, \ldots, v_m, u_2, ..., u_{f-2}$ in time $T - 2T/f = T - 4h$. The distances are shown below each edge in the figure: $d(o, u_1) = 1$, $d(u_i, u_{i+1}) = 1$ for i odd, $d(u_i, u_{i+1}) = h$ for i even, $d(o, v_1) = 1$, $d(v_i, v_{i+1}) = h + 1$, for $i = 1, \ldots, m - 1$, $d(v_m, u_2) = h + 1$, and all other distances (not shown) are h.

The requests are depicted as directed edges in the figure. They are: $(u_1, u_2, 0, \epsilon)$, $(u_{2i+1}, u_{2i+2}, 4ih, B+\epsilon)$ for $i = 1, \ldots, f/2-1$, $(u_{2i}, u_{2i+1}, 4ih+1, B)$ for $i = 1, \ldots, f/2-1$, $(v_i, v_{i+1}, 1, B)$ for $i = 1, ..., m-1$, and $(v_m, u_2, 1, B)$.

Note that SBP will take the path $o, u_1, ..., u_f$:

(1) At time $t = 0$, SBP will choose drive (o, u_1) followed by request (u_1, u_2) because that is all that is available.
(2) For $k = 1, \ldots, f/2-1$, at time $t = 4hk$ (the start time of a pair of time segments of length $2h = T/f$ each), SBP is at vertex u_{2k}. The available requests (that have not yet been served) are: $(u_{2i}, u_{2i+1}, 4ih+1, B)$ for $i = 1, ..., k-1$, $(u_{2k+1}, u_{2k+2}, 4kh, B+\epsilon)$, along with $(v_i, v_{i+1}, 1, B)$ for $i = 1, ..., m-1$, and $(v_m, u_2, 1, B)$. Note that none of the requests of revenue B along the top path arrive in time for SBP to serve more than a single request at a time. Further, since we are looking for a request set that has path length at most $2h$, we cannot put together a path of length at most $2h$ that has two or more of these requests (since an edge from either u_{2k+1} or u_{2k+2} to any other vertex that is the source or destination of a request has weight h by design). Thus a maximum revenue set chosen by SBP using a path of length at most $2h$ has only one request. And a maximum revenue request would clearly be the request $(u_{2k+1}, u_{2k+2}, 4kh, B+\epsilon)$. Thus SBP would drive from u_{2k} to u_{2k+1} at time $t = 4kh$ and serve the request (u_{2k+1}, u_{2k+2}) at time $t = 4kh + 2h$. And at time $t = 4(k+1)h$, SBP would be at vertex u_{2k+2}.

Thus SBP earns revenue $\epsilon + (f/2-1)(B+\epsilon) = B(f-2)/2 + f \cdot \epsilon/2$.

We now consider OPT, and let OPT′ denote the OPT path without the last two time segments. That is, OPT′ earns only the revenue earned up until time $T - 2T/f = 2fh - 4h = (2f-4)h$. So OPT′ is the path $o, v_1, \ldots, v_m, u_2, \ldots, u_{f-2}$. Note we stop at node u_{f-2} because the requests from u_{f-2} to u_{f-1} and u_{f-1} to u_f are not yet released before time $(2f-4)h$.

OPT′ takes $1 + m \cdot (h+1)$ time to get to u_2 and takes time $(1+h) \cdot (f/2-2)$ to go from u_2 to u_{f-2}. The total is $1 + m(h+1) + (1+h) \cdot (f/2-2)$.

The largest m for which OPT′ is completed before time $(2f-4)h$ is

$$m = \lfloor (3hf - 4h - f + 2)/(2(h+1)) \rfloor.$$

Observe that for this value of m, we have $1 + m(h+1) + (f/2-2) \cdot (1+h) \leq (2f-4)h$ as needed.

Clearly, OPT′ can serve all the requests on the path $v_1, ..., v_m, u_2$ because these requests were all released at time $t = 1$. Now, for each $k = 1, ..., f/2-2$, OPT′ arrives at vertex u_{2k} at time $\tau_k = 1 + m(h+1) + (1+h)(k-1)$. By Lemma 5 in Section 2 of the Appendix, $\tau_k \geq 4kh + 1$ for each $k = 1, \ldots, f/2-2$. Therefore the requests $(u_{2k}, u_{2k+1}, 4kh+1, B)$ and $(u_{2k+1}, u_{2k+2}, 4kh, B+\epsilon)$ are released on or before τ_k, allowing OPT′ to serve these two requests when it reaches u_{2k} at time τ_k. Therefore all the drives starting at v_1 are revenue generating requests for OPT′.

Now, OPT′ has revenue mB from v_1 up to u_2 and revenue $(2B+\epsilon)(f-4)/2$ from u_2 to u_{f-2}. Let $rev(\text{alg})$ denote the total revenue earned by a schedule, ALG. Then we can write $rev(\text{OPT}') = mB + (2B+\epsilon)(f-4)/2 = mB + Bf - 4B + \epsilon(f-4)/2 = \lfloor (3hf - 4h - f + 2)/(2(h+1)) \rfloor B + fB - 4B + \epsilon(f-4)/2$.

The ratio is thus

$$\frac{rev(\text{OPT}')}{rev(\text{SBP})} = \frac{\lfloor (3hf - 4h - f + 2)/(2(h+1)) \rfloor B + fB - 4B + \epsilon(f-4)/2}{B \cdot (f-2)/2 + f \cdot \epsilon/2}.$$

Taking the limit as ϵ approaches 0,

$$\begin{aligned}\lim_{\epsilon \to 0} \frac{rev(\text{OPT}')}{rev(\text{SBP})} &= \frac{\lfloor (3hf - 4h - f + 2)/(2(h+1)) \rfloor B + fB - 4B}{B \cdot (f-2)/2} \\ &= \frac{\lfloor (3hf - 4h - f + 2)/(2(h+1)) \rfloor + f - 4}{(f-2)/2}.\end{aligned}$$

Next we will take the limit as T approaches infinity, which is the same as taking h to infinity, because f is fixed. First, note the inside of the floor can be rewritten as

$$(3hf - 4h - f + 2)/(2(h+1)) = 3f/2 - 2 - (2f-3)/(h+1).$$

Note $3f/2 - 2$ is an integer since f is even, so when $h + 1 > 2f - 3$, we have $0 < (2f-3)/(h+1) < 1$. Thus $\lfloor 3f/2 - 2 - (2f-3)/(h+1) \rfloor = 3f/2 - 2 - 1 = 3f/2 - 3$ when $h > 2f - 4$. Then $\lim_{T \to \infty} \lfloor 3f/2 - 2 - (2f-3)/(h+1) \rfloor = 3f/2 - 3$. Therefore

$$\lim_{T \to \infty} \lim_{\epsilon \to 0} \frac{rev(\text{OPT}')}{rev(\text{SBP})} = \frac{3f/2 - 3 + f - 4}{(f-2)/2} = \frac{5f/2 - 7}{(f-2)/2} = \frac{(5f-14)}{(f-2)} = 5 - 4/(f-2).$$

Summarizing, for a fixed $f > 2$, this instance gives a lower bound of $5 - 4/(f-2)$ as ϵ approaches 0 and T approaches infinity.

3.2 Upper Bound on SBP

We now show that SBP is 5-competitive by creating a modified, hypothetical SBP schedule that has additional copies of requests. First, we note that SBP loses a factor of 2 due to the fact that it serves requests during only every other time segment.

Then, we lose another factor of two to cover requests in OPT that overlap between time segments. Finally, by adding at most one more copy of the requests served by SBP to make up for requests that SBP "incorrectly" serves prior to when they are served by OPT, we end up with 5 copies of SBP being sufficient for bounding the total revenue of OPT.

Note that while this proof uses some of the techniques of the proof of the 6-competitive upper bound in [1], it reduces the competitive ratio from 6 to 5 by cleverly extracting the set of requests that SBP serves prior to OPT before making the additional copies.

Let $rev(\text{OPT})$ and $rev(\text{OPT})$ denote the total revenue earned by OPT and SBP, respectively, over all time segments t_j from $j = 1 \ldots f$.

Theorem 3 *If the revenue earned by OPT in the last two time segments is bounded by some constant c (a necessary assumption due to Lemma 1), then SBP is 5-competitive, i.e., if* $rev(\text{OPT}(t_f)) + rev(\text{OPT}(t_{f-1})) \leq c$, *then* $\sum_{j=1}^{f} rev(\text{OPT}(t_j)) \leq 5 \sum_{j=1}^{f} rev(\text{SBP}(t_j)) + c$.

Another interpretation of this result is that under a resource augmentation model where SBP has two more time segments available than OPT, SBP is 5-competitive.

Proof We analyze the revenue earned by SBP by considering the time segments in pairs (recall that the length of a time segment is *T/f* for some $1 < f < T$). We refer to each pair of consecutive time segments as a time window, so if there are f time segments, there are $\lceil f/2 \rceil$ time windows. Note that the last time window may have only one time segment.

For notational convenience we consider a modified version of the SBP schedule, that we refer to as SBP′, which serves exactly the same set of requests as SBP, but does so one time window earlier. Specifically, if SBP serves a set of requests during time window $i \geq 2$, SBP′ serves this set during time window $i - 1$ (so SBP′ ignores the set served by SBP in window 1). We note that the schedule of requests served by SBP′ may be infeasible, and that it will earn at most the amount of revenue earned by SBP.

Let B_i denote the set of requests served by OPT in window i that SBP′ already served *before* in some window $j < i$. And let B be the set of all requests that have already been served by SBP′ in a previous window by the time they are served in the OPT schedule. Formally, $B = \bigcup_{i=2}^{\lceil f/2 \rceil} B_i$. Consider a schedule $\overline{\text{OPT}}$ that contains all of the requests in the OPT schedule minus the requests in B. So $\overline{\text{OPT}}$ earns total revenue $rev(\text{OPT}) - rev(B)$, where $rev(B)$ denotes the total revenue of the set B.

Let $\overline{\text{OPT}}(t_j)$ denote the set of requests served by $\overline{\text{OPT}}$ in time segment t_j. Let $\overline{\text{OPT}}_i$ denote the set of requests served by $\overline{\text{OPT}}$ in the time segment of window i with greater revenue, i.e., $\overline{\text{OPT}}_i = \arg\max\{rev(\overline{\text{OPT}}(t_{2i-1})), rev(\overline{\text{OPT}}(t_{2i}))\}$. Note this set may include a request that was started in the prior time segment, as long as it was completed in the time segment of $\overline{\text{OPT}}_i$. Let $rev(\overline{\text{OPT}}_i)$ denote the revenue earned in $\overline{\text{OPT}}_i$.

Let SBP'_i denote the set of requests served by SBP′ in window i and let $rev(\text{SBP}'_i)$ denote the revenue earned by SBP'_i. Let H denote the chronologically ordered set of time windows w where $rev(\overline{\text{OPT}}_w) > rev(\text{OPT}'_w)$, and let h_j denote the jth time window in H. We refer to each window of H as a window with a "hole," in reference to the fact that SBP′ does not earn as much revenue as $\overline{\text{OPT}}$ in these windows. In each window h_j there is some amount of revenue that $\overline{\text{OPT}}$ earns that SBP′ does not. In particular, there must be a set of requests that $\overline{\text{OPT}}$ serves in window h_j that SBP′ does not serve in h_j. Note that this set must be available for SBP′ in h_j since $\overline{\text{OPT}}$ does not include the set B.

Let $\overline{\text{OPT}}_{h_j} = A_j \cup C_j^*$, where A_j is the subset of requests served by both $\overline{\text{OPT}}$ and SBP′ in h_j and C_j^* is the subset of $\overline{\text{OPT}}$ requests available for SBP′ to serve in h_j but SBP′

chooses not to serve. Let us refer to the set of requests served by SBP′ in h_j as $\text{SBP}'_{h_j} = A_j \cup C_j$ for some set of requests C_j. Note that if $\overline{\text{OPT}}_{h_j} = A_j \cup C_j^*$ can be executed within a single time segment, then $rev(C_j) \geq rev(C_j^*)$ by the greediness of SBP′. However, since h_j is a hole we know that the set $\overline{\text{OPT}}_{h_j}$ cannot be served within one time segment.

Our plan is to build an infeasible schedule $\overline{\text{SBP}}$ that will be similar to SBP′ but contain additional "copies" of some requests such that no windows of $\overline{\text{SBP}}$ contain holes. We first initialize $\overline{\text{SBP}}$ to have the same schedule of requests as SBP′. We then add additional requests to h_j for each $j = 1 \ldots |H|$, based on $\overline{\text{OPT}}_{h_j}$.

Consider one such window with a hole h_j, and let k be the index of the time segment corresponding to $\overline{\text{OPT}}_{h_j}$. We know $\overline{\text{OPT}}$ must have begun serving a request of $\overline{\text{OPT}}_{h_j}$ in time segment t_{k-1} and completed this request in time segment t_k. Let us use r^* to denote this request that "straddles" the two time segments.

After the initialization of $\overline{\text{SBP}} = \text{SBP}'$, recall that the set of requests served by $\overline{\text{SBP}}$ in h_j is $\overline{\text{SBP}}_{h_j} = A_j \cup C_j$ for some set of requests C_j. We add to $\overline{\text{SBP}}$ a copy of a set of requests. There are two sub-cases depending on whether $r^* \in C_j^*$ or not.

Case $r^* \in C_j^*$. In this case, by the greediness of SBP, and the fact that both r^* alone and $C_j^* \setminus \{r^*\}$ can separately be completed within a single time segment, we have: $rev(C_j) \geq \max\{rev(r^*), rev(C_j^* \setminus \{r^*\})\} \geq \frac{1}{2} rev(C_j^*)$.

We then add a copy of the set C_j to the $\overline{\text{SBP}}$ schedule, so there are two copies of C_j in h_j. Note that for $\overline{\text{SBP}}$, h_j will no longer be a hole since:

$rev(\overline{\text{OPT}}_{h_j}) = rev(A_j) + rev(C_j^*) \leq rev(A_j) + 2 \cdot rev(C_j) = rev(\overline{\text{SBP}}_{h_j})$.

Case $r^* \notin C_j^*$. In this case C_j^* can be served within one time segment but SBP′ chooses to serve $A_j \cup C_j$ instead. So we have $rev(A_j) + rev(C_j) \geq rev(C_j^*)$, therefore we know either $rev(A_j) \geq \frac{1}{2} rev(C_j^*)$ or $rev(C_j) \geq \frac{1}{2} rev(C_j^*)$. In the latter case, we can do as we did in the first case above and add a copy of the set C_j to the $\overline{\text{SBP}}$ schedule in window h_j, to get $rev(\overline{\text{OPT}}_{h_j}) \leq rev(\overline{\text{SBP}}_{h_j})$, as above. In the former case, we instead add a copy of A_j to the $\overline{\text{SBP}}$ schedule in window h_j. Then again, for $\overline{\text{SBP}}$, h_j will no longer be a hole, since this time:

$rev(\overline{\text{OPT}}_{h_j}) = rev(A_j) + rev(C_j^*) \leq 2 \cdot rev(A_j) + rev(C_j) = rev(\overline{\text{SBP}}_{h_j})$.

Note that for all windows $w \notin H$ that are not holes, we already have $rev(\overline{\text{SBP}}_w) \geq rev(\overline{\text{OPT}}_w)$. So we have

$$\sum_{i=1}^{\lceil f/2 \rceil - 1} rev(\overline{\text{OPT}}_i) \leq \sum_{i=1}^{\lceil f/2 \rceil - 1} rev(\overline{\text{SBP}}_i) \leq 2 \sum_{i=1}^{\lceil f/2 \rceil - 1} rev(\text{SBP}'_i) \tag{1}$$

where the second inequality is because $\overline{\text{SBP}}$ contains no more than two instances of every request in SBP′.

Combining (1) with the fact that SBP′ earns at most what SBP does yields

$$\sum_{i=1}^{\lceil f/2 \rceil} rev(\overline{\text{OPT}}_i) \leq 2 \sum_{i=1}^{\lceil f/2 \rceil} rev(\text{SBP}_i) + rev(\overline{\text{OPT}}(t_{f-1})) + rev(\overline{\text{OPT}}(t_f)). \tag{2}$$

Since SBP serves in only one of two time segments per window, we have $\sum_{i=1}^{\lceil f/2 \rceil} rev(\text{SBP}_i) = \sum_{j=1}^{f} rev(\text{SBP}(t_j))$.
Hence, by the definition of $\overline{\text{OPT}}$, and by (2) we can say

$$\sum_{j=1}^{f} rev(\overline{\text{OPT}}(t_j)) \leq 2 \sum_{i=1}^{\lceil f/2 \rceil} rev(\overline{\text{OPT}}_i) \tag{3}$$

$$\leq 4 \sum_{j=1}^{f} rev(\text{SBP}(t_j)) + rev(\overline{\text{OPT}}(t_{f-1})) + rev(\overline{\text{OPT}}(t_f)). \tag{4}$$

Now we must add in any request in B, such that OPT serves the request in a time window after SBP′ serves that request. By definition of B (as the set of all requests that have been served by SBP′ in a previous window) B may contain at most the same set of requests served by SBP′. Therefore $rev(B) \leq rev(\text{SBP}')$, so $rev(B) \leq rev(\text{SBP})$.

By the definition of OPT, $\text{OPT} = \overline{\text{OPT}} + B$, so

$$\sum_{j=1}^{f} rev(\text{OPT}(t_j)) = rev(B) + \sum_{j=1}^{f} rev(\overline{\text{OPT}}(t_j)). \tag{5}$$

And by combining (3)-(5) with the fact that $rev(B) \leq rev(\text{SBP})$, we have:

$$\begin{aligned} \sum_{j=1}^{f} rev(\text{OPT}(t_j)) \leq & \sum_{j=1}^{f} rev(\text{SBP}(t_j)) + 4 \sum_{j=1}^{f} rev(\text{SBP}(t_j)) \\ & + rev(\overline{\text{OPT}}(t_{f-1})) + rev(\overline{\text{OPT}}(t_f)) \\ \leq & 5 \sum_{j=1}^{f} rev(\text{SBP}(t_j)) + rev(\overline{\text{OPT}}(t_{f-1})) \\ & + rev(\overline{\text{OPT}}(t_f)). \end{aligned}$$

4 Uniform Revenues

We now consider the setting where revenues are uniform among all requests, so the goal is to maximize the total number of requests served. This variant is useful for settings where all requests have equal priorities, for example for not-for-profit services that provide transportation to elderly and disabled passengers.

The proof strategy is to carefully consider the requests served by SBP in each window and track how they differ from that of OPT. The final result is achieved

through a clever accounting of the differences between the two schedules, and bounding the revenue of the requests that are "missing" from SBP.

We now show that OPT earns at most 4 times the revenue of SBP in this setting if we assume the revenue earned by OPT in the last two time segments is bounded by a constant, and allow SBP an additive bonus of f. Note that even when revenues are uniform, *no* non-preemptive deterministic online algorithm can earn the revenue earned by OPT in the last two time segments (see Lemma 1). We note that the lower bound instance of Theorem 2 can be modified to become a uniform-revenue instance that has ratio $5 - 14/f$. We further note that the lower bound instance provided in [1] immediately establishes a lower bound instance for SBP that has a ratio of 4, so the upper bound we now present is nearly tight. We begin with several definitions and lemmas.

As in the proof of Theorem 3, we consider a modified version of the SBP schedule, that we refer to as SBP′, which serves exactly the same set of requests as SBP, but does so one time window earlier. For all windows $i = 1, 2, ..., m$, where $m = \lceil f/2 \rceil - 1$, we let S'_i denote the set of requests served by SBP′ in window i and S^*_i denote the set of requests served by OPT during the time segment of window i with greater revenue, i.e., $S^*_i = arg\,max\{rev(\text{OPT}(t_{2i-1}), rev(\text{OPT}(t_{2i}))\}$ where $rev(\text{OPT}(t_j))$ denotes the revenue earned by OPT in time segment t_j. We define a new set J^*_i as the set of requests served by OPT during the time segment of window i with less revenue, i.e., $J^*_i = arg\,min\{rev(\text{OPT}(t_{2i-1}), rev(\text{OPT}(t_{2i}))\}$.

Let $S^*_i = A_i \cup X^*_i \cup Y^*_i$, and $S'_i = A_i \cup X_i \cup Y_i$, where:
(1) A_i is the set of requests that appear in both S^*_i and S'_i; (2)
X^*_i is the set of requests that appear in S'_w for some $w = 1, 2, ..., i-1$. Note there is only one possible w for each individual request $r \in X^*_i$, because each request can be served only once; (3)
Y^*_i is the set of requests such that no request from Y^*_i appears in S'_w for any $w = 1, 2, ..., i-1, i$; (4) X_i is the set of requests that appear in S^*_w for some $w = 1, 2, ..., i-1$. Note there is only one possible w for each individual request $r \in X_i$, because each request can be served only once; (5) Y_i is the set of requests such that no request from Y_i appears in S^*_w for any $w = 1, 2, ..., i-1, i$.

Note that elements in Y_i can appear in a previous J^*_w for any $w = 1, 2, ..., i-1, i$ or in a future S^*_v or J^*_v for any $v = i+1, i+2, ..., m$, or may not appear in any other sets. Also note that since each request can be served at most once, we have: $A_1 \cap X^*_1 \cap Y^*_1 \cap A_2 \cap X^*_2 \cap Y^*_2 \cap ... \cap A_m \cap X^*_m \cap Y^*_m = \emptyset$ and $A_1 \cap X_1 \cap Y_1 \cap A_2 \cap X_2 \cap Y_2 \cap ... \cap A_m \cap X_m \cap Y_m = \emptyset$.

Given the above definitions, we have the following lemmas.

Lemma 2 *All requests $r \in X^*_i$ must satisfy that $r \in Y_w$ for some $w = 1, 2, ..., i-1$, and there is only one possible value of w.*

Proof By definition, each request of X^*_i must appear in S'_w for some $w = 1, 2, ..., i-1$, and there is only one possible value of w. Let r be a request of X^*_i. We know that r

must appear in either A_w, or X_w, or Y_w. However, r cannot appear in A_w, since otherwise r would have been served in S_w^*, where $w < i$, which is a contradiction since r is served in S_i^*. Similarly, r cannot appear in X_w, since otherwise r would have been served in S_v^* for some $v = 1, 2, ..., w-1$, where $v < w < i$, which is a contradiction since we know r is served in S_i^*. By elimination, r must be a request of Y_w.

Lemma 3 $X_1^* \cup X_2^* \cup ... \cup X_i^* \subseteq Y_1 \cup Y_2 \cup ... \cup Y_{i-1}$ *for all* $i = 2, 3, ..., m$.

Proof We prove this by induction. For the base case, by Lemma 2, X_1^* must be a subset of Y_0, where Y_0 is the empty set; X_2^* must be a subset of Y_1. Therefore, $X_1^* \cup X_2^* = \emptyset \cup X_2^* \subseteq Y_1$. For the inductive case, assume

$$X_1^* \cup X_2^* \cup ... \cup X_k^* \subseteq Y_1 \cup Y_2 \cup ... \cup Y_{k-1}. \tag{6}$$

Consider X_{k+1}^* and Y_k. By Lemma 2, elements of X_{k+1}^* can come from only two sources: Y_k and $Y_1 \cup Y_2 \cup ...Y_{k-1}$. Therefore,

$$X_{k+1}^* \subseteq Y_1 \cup Y_2 \cup ... \cup Y_{k-1} \cup Y_k. \tag{7}$$

Combining (6) and (7), we have

$$X_1^* \cup X_2^* \cup ... \cup X_k^* \cup X_{k+1}^* \subseteq Y_1 \cup Y_2 \cup ... \cup Y_{k-1} \cup Y_k.$$

Lemma 4 *For all* $i = 1, 2, ..., m$ *we have* $\sum_{j=1}^{i} |X_j^*| \leq \sum_{j=1}^{i} |Y_i|$.

Proof The size of a set A must be at least the size of any subset of A so from Lemma 3, we have for all $i = 1, 2, ..., m$,

$$|X_1^* \cup X_2^* \cup ...X_i^*| \leq |Y_1 \cup Y_2 \cup ...Y_{i-1}|.$$

Recall we observed above that $X_1^* \cap X_2^* \cap ... \cap X_i^* = \emptyset$ and $Y_1 \cap Y_2 \cap ... \cap Y_{i-1} = \emptyset$. Hence, we can rewrite the inequality as

$$|X_1^*| + |X_2^*| + ... + |X_i^*| \leq |Y_1| + |Y_2| + ... + |Y_{i-1}|.$$

By adding $|Y_i|$ on the right-hand-side, we have proven the claim.

We are now ready to prove the main theorem of this section.

Theorem 4 *If the revenue earned by* OPT *in the last two time segments is bounded by some constant* c, *i.e. if* $rev(\text{OPT}(t_f)) + rev(\text{OPT}(t_{f-1})) \leq c$ (*a necessary assumption due to Lemma 1*), *then* SBP *earns at least* 1/4 *the revenue of* OPT, *minus an additive term linear in* f, *where* T/f *is the length of one time segment. I.e.,* $\sum_{j=1}^{f} rev(\text{OPT}(t_j)) \leq 4 \sum_{j=1}^{f} rev(\text{SBP}(t_j)) + 2\lceil f/2 \rceil + c$. *So if* f *is also bounded by some constant, then* SBP *is* 4*-competitive*.

Proof Note that since revenues are uniform, the revenue of a request-set U is equal to the size of the set U, i.e., $rev(U) = |U|$. Consider each window i where

$rev(S_i^*) > rev(S_i')$. Note that the set S_i^* may not fit within a single time segment. We consider two cases based on S_i^*.

1. The set S_i^* can be served within one time segment. Note that within $S_i^* = A_i \cup X_i^* \cup Y_i^*$, X_i^* is not available for SBP′ to serve because SBP′ has served the requests in X_i^* prior to window i. Among requests that are available to SBP′, SBP′ greedily chooses to serve the maximum revenue set that can be served within one time segment. Therefore, we have $rev(X_i) + rev(Y_i) \geq rev(Y_i^*)$. Since revenues are uniform, we also have $|X_i| + |Y_i| \geq |Y_i^*|$. If this is not the case, then SBP′ would have chosen to serve Y_i^* instead of $X_i \cup Y_i$ since it is feasible for SBP′ to do so because the entire S_i^* can be served within one time segment.
2. The set S_i^* cannot be served within one time segment. This means there must be one request in S_i^* that OPT started serving in the previous time segment. We refer to this straddling request as r^*. There are three sub-cases based on where r^* appears.

(a) If $r^* \in Y_i^*$, then due to the greediness of SBP′, we know that

$$rev(X_i) + rev(Y_i) \geq rev(r^*) \tag{8}$$

since otherwise SBP′ would have chosen to serve r^*. We also know

$$rev(X_i) + rev(Y_i) \geq rev(Y_i^* \backslash \{r^*\}) \tag{9}$$

since otherwise SBP′ would have chosen to serve $Y_i^* \backslash \{r^*\}$. From (8), we have $|X_i| + |Y_i| \geq 1$ and from (9), we have $|X_i| + |Y_i| \geq |Y_i^*| - 1$.

(b) If $r^* \in X_i^*$, then r^* is not available to SBP′ and only A_i, X_i, Y_i, and Y_i^* are available to SBP′. Therefore we know that $rev(X_i) + rev(Y_i) \geq rev(Y_i^*)$ since otherwise, by its greediness, SBP′ would have chosen to serve A_i and Y_i^* instead of A_i, X_i and Y_i, because A_i and Y_i^* can be served within one time segment. Therefore, we have $|X_i| + |Y_i| \geq |Y_i^*|$.

(c) $r^* \in A_i$. Then r^* is served by both OPT and SBP′. We know that $A_i \cup Y_i^* \backslash \{r^*\}$ can be served within one time segment since r^* is the only request that causes S_i^* to straddle between two time segments. Again by the greediness of SBP′, we have $rev(A_i) + rev(X_i) + rev(Y_i) \geq rev(A_i) + rev(Y_i^*) - rev(r^*)$ which means $rev(X_i) + rev(Y_i) \geq rev(Y_i^*) - rev(r^*)$ and $|X_i| + |Y_i| \geq |Y_i^*| - 1$.

Therefore, for all cases, for window i, we have $|X_i| + |Y_i| \geq |Y_i^*| - 1$, which means $|Y_i^*| - |X_i| \leq 1 + |Y_i|$, and with $m = \lfloor f/2 \rfloor - 1$,

$$\sum_{i=1}^{m} (|Y_i^*| - |X_i|) \leq m + \sum_{i=1}^{m} |Y_i|. \tag{10}$$

Now we will build an infeasible schedule $\overline{\text{SBP}}$ that will be similar to SBP′ but contain additional "copies" of some requests such that no windows of $\overline{\text{SBP}}$ contain holes, i.e., such that $rev(\overline{\text{SBP}}) \geq \sum_{i=1}^{m} rev(S_i^*)$.

We define a modified OPT schedule which we refer to as OPT′ such that OPT′ $= \cup_{i=1}^{m} S_i^*$ and observe that $rev(\text{OPT}') = \sum_{i=1}^{m} |A_i| + \sum_{i=1}^{m} |X_i^*| + \sum_{i=1}^{m} |Y_i^*|$, while $rev(\text{SBP}') = \sum_{i=1}^{m} |A_i| + \sum_{i=1}^{m} |X_i| + \sum_{i=1}^{m} |Y_i|$.

By Lemma 4 and Equation (10), we can say $rev(\text{OPT}') - rev(\text{SBP}') = \sum_{i=1}^{m} |Y_i^*| - \sum_{i=1}^{m} |X_i| + \sum_{i=1}^{m} |X_i^*| - \sum_{i=1}^{m} |Y_i| \leq \sum_{i=1}^{m} |Y_i^*| - \sum_{i=1}^{m} |X_i| \leq m + \sum_{i=1}^{m} |Y_i|$.

This tells us that to form an $\overline{\text{SBP}}$ whose revenue is at least that of OPT′, we must "compensate" SBP′ by adding to it at most copies of all requests in the set Y_i for all $i = 1, 2, ..., m$, plus m "dummy requests." In other words,

$$rev(\overline{\text{SBP}}) = rev(\text{SBP}') + m + \sum_{i=1}^{m} |Y_i| \geq rev(\text{OPT}'). \tag{11}$$

We know the total revenue of all Y_i can not exceed the total revenue of SBP′, hence we have

$$rev(\overline{\text{SBP}}) = rev(\text{SBP}') + m + \sum_{i=1}^{m} |Y_i| \leq 2rev(\text{SBP}') + m. \tag{12}$$

Combining (11) and (12), we get $rev(\text{OPT}') \leq 2rev(\text{SBP}') + m$, which means

$$\sum_{i=1}^{m} rev(S_i^*) \leq 2 \sum_{i=1}^{m} rev(S_i') + m. \tag{13}$$

Recall that S_i^* is the set of requests served by OPT during the time segment of window i with greater revenue. In other words, $\sum_{j=1}^{2m} rev(S^*(t_j)) \leq 2 \sum_{i=1}^{m} rev(S_i^*)$, which, combined with (13), gives us

$$\sum_{j=1}^{2m} rev(S^*(t_j)) \leq 4 \sum_{i=1}^{m} rev(S_i') + 2m. \tag{14}$$

Since we may assume that the total revenue of requests served in the last two time segments by OPT is bounded by c, from (14) we get

$$\sum_{j=1}^{f} rev(S^*(t_j)) \leq \sum_{j=1}^{2m} rev(S^*(t_j)) + rev(S^*(t_{f-1})) + rev(S^*(t_f)) \tag{15}$$

$$\leq 4 \sum_{i=1}^{m} rev(S_i') + 2m + c. \tag{16}$$

We also know that the total revenue of requests served by SBP′ during the first m windows is less than or equal to the total revenue of SBP. Therefore, from (15), we have $\sum_{j=1}^{f} rev(S^*(t_j)) \leq 4 \sum_{j=1}^{f} rev(S(t_j)) + 2m + c$.

5 Bipartite Graphs

In this section, we consider ROLDARP for complete bipartite graphs $G = (V = V_1 \cup V_2, E)$, where only nodes in V_1 may be source nodes and only nodes in V_2 may be destination nodes. One node is designated as the origin and there is an edge from this node to every node in V_1 (so the origin is a node in V_2).

We refer to this problem as ROLDARP-B and the offline version as RDARP-B. We refer to the offline version of ROLDARP with integer edge weights as RDARP-$\mathbb{Z}$, and show that RDARP-$\mathbb{Z}$ reduces to RDARP-B, and the uniform-revenue version of RDARP-$\mathbb{Z}$ reduces to uniform-revenue RDARP-B. Since RDARP-$\mathbb{Z}$ and uniform revenue RDARP-$\mathbb{Z}$ are NP-hard [16], then RDARP-B and uniform revenue RDARP-B are also NP-hard.

Theorem 5 *RDARP-$\mathbb{Z}$ is polynomial-time reducible to RDARP-B, and the uniform-revenue version of RDARP-$\mathbb{Z}$ is polynomial-time reducible to uniform-revenue RDARP-B.*

Proof Consider an instance of RDARP-$\mathbb{Z}$ with time limit T (the proof below holds for both the uniform and nonuniform revenue variants). Because edge weights are integers, we can assume for simplicity that T is an integer. For every node u in the RDARP-$\mathbb{Z}$ graph, make two nodes u_1 and u_2 for the bipartite graph such that $u_1 \in V_1$ (the source side) and $u_2 \in V_2$ (the destination side), then set the weights $w(u_1, u_2) = w(u_2, u_1) = \epsilon = \frac{1}{2T}$.

For every edge (u, v) in the RDARP-$\mathbb{Z}$ graph with weight $w(u, v)$, make two edges in the bipartite graph: one from $u_1 \in V_1$ to $v_2 \in V_2$ and a second from $u_2 \in V_2$ to $v_1 \in V_1$, both with weight $w(u, v)$. Finally, for each request from source s to destination d in the RDARP-$\mathbb{Z}$ instance, create an equivalent-revenue request in the RDARP-B instance from $s_1 \in V_1$ to $d_2 \in V_2$. Set the time limit in the RDARP-B instance to be $T + \frac{1}{2}$. The additional $\frac{1}{2}$ is to accommodate time needed to travel the ϵ-weight edges.

We complete the proof by showing that a schedule σ with revenue R exists in the RDARP-$\mathbb{Z}$ instance if and only if a schedule σ' with revenue R exists in the RDARP-B instance.

(1) Let the path $P = (a, b, c, \ldots, z)$ be the path followed in the execution of the schedule σ in the RDARP-$\mathbb{Z}$ instance. The equivalent path in the RDARP-B instance would be $P' = (a_2, a_1, b_2, b_1, c_2, c_1, \ldots, z_2)$. Clearly P' has the same revenue as P. Notice that because P can have at most T edges, and since the number of edges in P' of the form (u_2, v_1) is exactly the number of edges in the path P, then P' takes time at most $T + T\epsilon = T + \frac{1}{2}$. So P' gives our desired schedule σ'.

(2) Conversely, suppose schedule σ' uses a path P'. Then P' must alternate from one side of the bipartition to the other. Construct P from P' by simply mapping each u_1 or u_2 of P' to u. Clearly, P earns the same revenue as P'. Any moves in P' along ϵ -weight edges would result in consecutive repeated vertices in P and therefore represent the server staying put in P. Since P' takes time at most $T + \frac{1}{2}$, path P will also

take time at most $T + \frac{1}{2}$. Since edge weights in the RDARP-$\mathbb{Z}$ instance are integers, P takes time at most $\lfloor T + \frac{1}{2} \rfloor = T$.

Corollary 1 *RDARP-B is NP-hard. This is the case for both the uniform and nonuniform revenue variants.*

Proof The proof is a direct result of Theorem 5 and the NP-hardness of uniform revenue RDARP-$\mathbb{Z}$ [16].

Corollary 2 *RDARP-$\mathbb{Z}$ is poly-time reducible to RDARP-B-$\mathbb{Z}$ (the bipartite version with integer weights). Thus RDARP-B-$\mathbb{Z}$ is NP-hard. This is the case for both the uniform and nonuniform revenue variants.*

Proof In the proof to Theorem 5, modify the RDARP-B instance to an RDARP-B-$\mathbb{Z}$ instance as follows. Scale all the edge weights along with the time limit by the factor $2T$. Then this modified instance has a schedule of revenue R if and only if the unmodified instance has a schedule of revenue R. Note that the new weights are all integers. This is a polynomial time reduction because the size of the graph is the same and the new time limit $2T^2$ is a polynomial function of the previous time limit T.

5.1 Uniform Revenue Bipartite

Since the reduction of Theorem 5 relied on edge weights being arbitrarily small, we now impose a minimum edge weight of kT/f for some constant k such that $0 < k \leq 1$. Note these graphs reflect real-world settings where pickup and drop-off locations are likely to be at least some minimum distance away from each other. We now show that if revenues are uniform, we can guarantee that SBP earns a fraction of OPT equal to the ratio between the minimum and maximum edge-length.

Theorem 6 *For any instance of ROLDARP-B where the revenues are uniform for all requests, if edge weights are upper and lower bounded by T/f and kT/f, respectively, for some constant* $0 < k \leq 1$, *then* $rev(\text{OPT}) \leq \lceil 1/k \rceil \cdot rev(\text{SBP}) + \lceil 1/k \rceil$.

Proof We define a *k-unit* as a length of time of duration kT/f. As before, we refer to each pair of consecutive time segments as a time window. We say a k-unit belongs to time window i if it ends within time window i. Note that a k-unit may straddle two windows by starting at one window and ending in the next.

If $2/k$ is an integer, then there are strictly $\frac{2T/f}{kT/f} = 2/k$ k-units in one time window. Note that this is true even if the window contains a straddling k-unit, since this k-unit will force another one to straddle into the next time window.

If $2/k$ is not an integer, then there are $\lfloor \frac{2T/f}{kT/f} \rfloor = \lfloor 2/k \rfloor$ non-straddling k-units within one window. If the window contains a straddling k-unit, the number of k-units will be $\lfloor 2/k \rfloor + 1 = \lceil 2/k \rceil$. Among those $\lceil 2/k \rceil$ k-units, at most $\lceil \lceil 2/k \rceil / 2 \rceil = \lceil 1/k \rceil$ of

them can be used in an OPT schedule to serve requests since no algorithm can serve two or more requests consecutively without a move in between. Therefore, whether or not $2/k$ is an integer, the maximum number of requests that can be served in each window is $\lceil 1/k \rceil$.

Let $\mu = \lfloor f/2 \rfloor$ denote the total number of windows. There are two cases based on the performance of SBP:

1. SBP serves at least 1 request per window. In this case, SBP serves at least μ requests, and OPT serves at most $\mu \cdot \lceil 1/k \rceil$. Therefore, $\frac{rev(\text{OPT})}{rev(\text{SBP})} \leq \frac{\mu \lceil 1/k \rceil}{\mu} \leq \lceil 1/k \rceil$.
2. There exists at least one window where SBP serves no requests. We refer to such a window as an "empty window." Consider the last empty window that occurred within the entire time limit, and denote this window as w. Let τ denote the start time of window w. We analyze the requests served (1) before, (2) during, and (3) after w:

   - Before window w: Since SBP serves nothing during window w, we know that all requests released before time τ have been served by SBP. Let b denote this number of requests. So before τ, OPT could have served at most b requests.
   - During window w: OPT can serve at most $\lceil 1/k \rceil$ requests and SBP serves no requests.
   - After window w: Suppose there are x windows after w (excluding window w). By definition, window w is the last window where SBP serves no requests, hence during the x windows afterward, SBP serves at least one request per window. On the other hand, OPT serves at most $\lceil 1/k \rceil$ requests per window.

   We know that $\lceil 1/k \rceil \geq 1$. Therefore we have $rev(\text{OPT}) \leq b + \lceil 1/k \rceil + x \cdot \lceil 1/k \rceil$ and $rev(\text{SBP}) \geq b + 0 + x$. Hence

$$rev(\text{OPT}) \leq \lceil 1/k \rceil \cdot rev(\text{SBP}) + \lceil 1/k \rceil.$$

5.2 Nonuniform Revenue Bipartite

In this section we show that even when revenues are nonuniform, we can guarantee that SBP earns a fraction of OPT equal to the ratio between the minimum and maximum edge-length, minus the revenue earned by OPT in the last window (i.e. the last two time segments). Note that *no* non-preemptive deterministic online algorithm can be competitive with any fraction of the revenue earned by OPT in the last time window (i.e., Lemma 1 also holds for ROLDARP-B with nonuniform revenues).

Theorem 7 *For any instance of ROLDARP-B where the revenues of requests are nonuniform, if edge weights are upper and lower bounded by T/f and kT/f, respectively, for some constant* $0 < k \leq 1$, *and if the revenue earned by* OPT *in the last time window is bounded by some constant c* (*a necessary assumption due to Lemma 1*), *then* $rev(\text{OPT}) \leq \lceil 1/k \rceil \cdot rev(\text{SBP}) + c$.

Proof Again, we refer to each pair of consecutive time segments as a time window. We consider a hypothetical schedule which we refer to as SBP′ that proceeds as follows.

In the first time window, SBP′ does nothing.

In the i^{th} window ($2 \le i \le \lceil f/2 \rceil$), SBP′ serves exactly one request: the maximum revenue request served by OPT in the $(i-1)^{th}$ window.

(In Lemmas 6 and 7 of Appendix 3 we show that the revenue earned by SBP′ is no greater than the revenue earned by SBP.)

Let Q_i, Q'_i, and Q^*_i denote the sets of requests served by SBP, SBP′, and OPT, respectively, in window i. There are two cases based on the performance of SBP.

Case 1: SBP serves at least one request per window.

Again let $\mu = \lceil f/2 \rceil$ denote the total number of time windows. Let $r = \lceil 1/k \rceil$. We know from Theorem 6 that OPT can serve at most r requests per window. We assume without loss of generality that OPT serves exactly r requests per window and let $\rho_i = \rho_{i,1}, \rho_{i,2}, ..., \rho_{i,r}$ denote the r revenues earned by OPT in window i. Consider the first window of OPT and the second window of SBP′. In the first window, OPT earns revenues $\rho_1 = \rho_{1,1}, \rho_{1,2}, ..., \rho_{1,r}$. In the second window, SBP′ serves the maximum revenue request from ρ_1. Therefore, $rev(Q^*_1) = \sum_{k=1}^{r} \rho_{1,k} \le r \cdot \max\{\rho_1\}$ and $rev(Q'_2) = \max\{\rho_1\}$. So we have $rev(Q^*_1) \le r \cdot rev(Q'_2)$. Similarly, we have $rev(Q^*_i) \le r \cdot rev(Q'_{i+1})$ for all $i = 1, 2, ..., \mu - 1$. Summing up for all $i = 1, 2, ..., \mu - 1$, we know

$$\sum_{i=1}^{\mu-1} rev(Q^*_i) \le r \sum_{i=2}^{\mu} rev(Q'_i). \tag{17}$$

From Lemmas 6 and 7 of Appendix 3 we know the right-hand-side of (17) is no more than the total revenue earned by SBP during all μ windows.

Specifically, Lemma 6 defines a another subroutine, called SBP″, that, for every window, serves the highest revenue available request and shows why $rev(\text{SBP}'') \ge rev(\text{SBP}')$. Lemma 7 shows why $rev(\text{SBP}) \ge rev(\text{SBP}'')$, so we know $rev(\text{SBP}) \ge rev(\text{SBP}')$.

$\sum_{i=1}^{\mu-1} rev(Q^*_i) \le r \sum_{i=1}^{\mu} rev(Q_i)$. Since the revenue earned by OPT in the last (i.e., μ^{th}) window is bounded by a constant c, $\sum_{i=1}^{\mu} rev(Q^*_i) \le r \sum_{i=1}^{\mu} rev(Q_i) + c$. In other words, $rev(\text{OPT}) \le r \cdot rev(\text{SBP}) + c = \lceil 1/k \rceil \cdot rev(\text{OPT}) + c$.

Case 2: There may be empty windows (i.e., windows where SBP serves nothing). Let w denote the last empty window that occurred during the entire time limit and let τ denote the start time of window w. We analyze the requests served before, during, and after w.

- Before window w: since SBP serves nothing during window w, we know that all requests released before time τ have been served by SBP. Let b denote the total revenue of these requests. We know that before τ, OPT could have earned revenue at most b.
- During window w: OPT earns revenue $\rho_{w,1}, \rho_{w,2}, ..., \rho_{w,r}$ and SBP earns nothing.

- After window w: now we proceed by running SBP$'$ which serves the maximum revenue request served in the previous window in the OPT schedule. Similar to (17), we have $rev(Q_i^*) \le r \cdot rev(Q'_{i+1})$ for all $i = w, w+1, ..., \mu - 1$. Summing up for all $i = w, w+1, ..., \mu - 1$ yields $\sum_{i=w}^{\mu-1} rev(Q_i^*) \le r \sum_{i=w+1}^{\mu} rev(Q'_i)$. From Lemma 6 (in the Appendix) we know that $r \sum_{i=w+1}^{\mu} rev(Q'_i)$ is no more than the total revenue earned by SBP during all windows after window w, therefore $\sum_{i=w}^{\mu-1} rev(Q_i^*) \le r \sum_{i=w+1}^{\mu} rev(Q_i)$. Since the revenue earned by OPT in the last (ie. μ^{th}) window is bounded by a constant c, we have

$$\sum_{i=w}^{\mu} rev(Q_i^*) \le r \sum_{i=w+1}^{\mu} rev(Q_i) + c. \tag{18}$$

So:

$$rev(\text{OPT}) = \sum_{i=1}^{\mu} rev(Q_i^*) \le b + \sum_{i=w}^{\mu} rev(Q_i^*) \le b + r \sum_{i=w+1}^{\mu} rev(Q_i) + c \tag{19}$$

$$rev(\text{SBP}) = \sum_{i=1}^{\mu} rev(Q_i) = b + 0 + \sum_{i=w+1}^{\mu} rev(Q_i). \tag{20}$$

Combining 19 and 20 we have:

$$\frac{rev(\text{OPT}) - c}{rev(\text{SBP})} \le \frac{b + r \sum_{i=w+1}^{\mu} rev(Q_i)}{b + \sum_{i=w+1}^{\mu} rev(Q_i)} \le \frac{rb + r \sum_{i=w+1}^{\mu} rev(Q_i)}{b + \sum_{i=w+1}^{\mu} rev(Q_i)} = r. \tag{21}$$

Which means $rev(\text{OPT}) \le \lceil 1/k \rceil \cdot rev(\text{SBP}) + c$.

6 Nonuniform revenues and uniform metric

In this section we consider ROLDARP for nonuniform revenues and uniform weight, a useful variant for settings where requests take roughly the same amount of time to serve (e.g., in urban areas). The problem was first studied in [17], where it was shown that no deterministic algorithm can earn the revenue that an optimal offline schedule earns in the last time unit. In [17] the simple greedy GREATEST REVENUE FIRST (GRF) algorithm was presented for this variant and shown to have competitive ratio 2, provided that the revenue earned by the optimal offline schedule in the last time unit is bounded by a constant. This version of the problem was then studied in [1] where we presented the BEST PATH (BP) algorithm as a less naive, more exhaustive greedy algorithm for the problem. At every time unit, BP finds and serves the request-path with the highest *score*, or revenue per time unit. Specifically, the *score*(P) of a request-path P is the total revenue of the requests in P divided by the time it takes to complete P, including the time it takes to move from the current server location to the start of P.

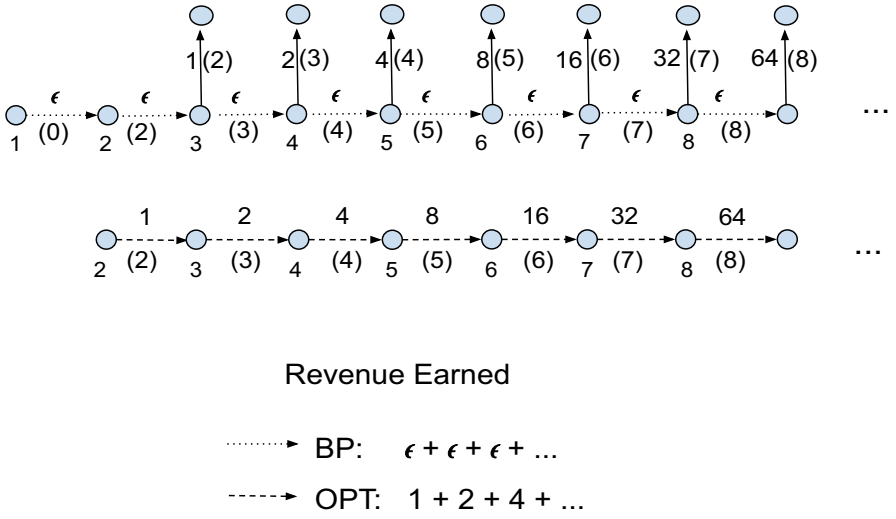

Fig. 2 An instance for which BP's competitive ratio is unbounded. Requests served by BP are shown on top (dotted edges) while the requests served by OPT are shown below. The number below each node indicates the time at which the corresponding algorithm reaches that node. The value above (or left of) each edge indicates the revenue of the corresponding request. The value in parentheses below (or right of) each request is the release time of that request. BP serves the first two requests with revenue ϵ, then only the first request of each chain of length 2

We demonstrate that although BP seems to perform better on average than the simpler GRF algorithm (see Section 7), unlike GRF's competitive ratio of 2, the competitive ratio of BP is unbounded.

In the instance shown in Fig. 2, OPT serves a "chain" of requests, i.e., with no moves in between requests. Specifically, at time $t = 1$, OPT (omnisciently) moves to the source of a request that arrives at $t = 2$ and then serves it, completing it at $t = 3$; it then serves the next request from $t = 3$ to $t = 4$ and so on until completing the last request at time T. In general, for $m = 2, \ldots, T-1$ OPT serves a request with revenue 2^{m-2}. On the other hand, at $t = 0$, BP moves to the request with revenue $\epsilon > 0$, since it is the only released request, and serves it from $t = 1$ to $t = 2$. At $t = 2$, it finds a path that continues from its current location containing two consecutive requests with revenues ϵ and 1, which has a score of $(\epsilon + 1)/2 = 1/2 + \epsilon/2$. Since this path has higher score than the path OPT is serving which has score 1/2, BP begins to serve this path. However after serving the first request and earning revenue ϵ, it then finds another path of two consecutive requests that have the highest score of $(\epsilon + 2)/2 = 1 + \epsilon/2$ and begins serving this path, again earning revenue ϵ. Meanwhile, the path served by OPT during this time, and any other available path, has score at most 1. BP continues this way, serving only the first request of each path it finds and earning revenue ϵ every time. So we have:

$$rev(\text{OPT}) = \sum_{m=2}^{T-1} 2^{m-2} = 2^{T-2} - 1 \tag{22}$$

and:

$$rev(\text{BP}) = \sum_{m=1}^{T-1} \epsilon = (T-1)\epsilon. \tag{23}$$

So as ϵ approaches 0, $rev(\text{OPT})/rev(\text{BP})$ approaches infinity.

7 Experimental Results

In this section we provide experimental results on SBP and BP.

7.1 Experimental Results for SBP

To evaluate the performance of the SBP algorithm, we simulated three realistic Online Dial-a-Ride systems. Our experimental settings were informed by interviews with representatives from real-world Dial-a-Ride organizations ([18–20]) and reflected three Dial-a-Ride environments: rural, suburban, and urban. The problem inputs varied based on the environment type. In [1], we evaluated SBP experimentally under the same three environments where the source, destination, and release times of requests were uniformly distributed. To simulate a more realistic environment, we now assume nonuniform distributions that were informed by the real-world Dial-a-Ride systems mentioned above.

We now describe the three environments more specifically. In each environment, a time unit is 10 minutes, so there are 6 time units per hour. The graph is complete and contains 50 nodes where five of these nodes are designated "hot-spots", i.e., they are 10 times more likely to be chosen as a source or destination (but the same node cannot be both the source and destination of a request). The origin is randomly chosen from the set of nodes. The release times of requests are non-integral values distributed from $t = 0$ to $t = T$ as follows. For each environment, certain hours of the day are designated as "rush-hours". During these hours, requests are 4 times more likely to be released than during non-rush-hour times. The rush-hours are as follows: for Rural and Suburban: 8-9am, 12-1pm, and 4-5pm; for Urban: 7-9am, 12-1pm, and 5-7pm. Request priorities are integral values uniformly distributed from 1 to m where m is the number of requests (we tested $m = 25, 50, 75$, and 100). The time limits and maximum edge weights for each environment are as follows:

1. Rural: Time spans 9 hours (from 8:00am to 5:00pm), so $T = 54$. The maximum distance between locations is 60 minutes (i.e., 6 time units), so $T/f = 6$ and $f = 9$. Edge weights are chosen uniformly at random from the interval $[1, T/f]$.
2. Suburban: Time spans 9 hours (from 8:00am to 5:00pm), so $T = 54$. The maximum distance between locations is 45 minutes (i.e., 4.5 time units), so $T/f = 4.5$ and $f = 12$. Edge weights are chosen uniformly at random from the interval $[1, T/f]$.

3. Urban: Time spans 13 hours (from 6:00am to 7:00pm), so $T = 78$. The maximum distance between locations is 20 minutes (i.e., 2 time units), so $T/f = 2$ and $f = 39$. Edge weights are chosen uniformly at random from the interval $[0.5, T/f]$.

Since R-DARP is NP-hard to solve optimally ([1]), we do not compare SBP to the optimal schedule. Additionally, since, to our knowledge, SBP is the only algorithm that has been published for this variant of OLDARP, we compare SBP to an offline version of itself, as well as to a basic greedy algorithm similar to one proposed in [12]. Like SBP, the offline-SBP algorithm serves requests in time segments - i.e., it uses one time segment to determine and move to the maximum-revenue-request-set and the next time segment to serve this set. However, unlike SBP, this offline algorithm learns of all the requests at the start of time ($t = 0$) and is therefore able to make a more informed decision about which requests to serve. In contrast, the greedy algorithm does not break time into segments and instead simply moves to and serves the outstanding request whose revenue is highest (similar to the GRF algorithm of [17]). One significant advantage that the greedy algorithm has is that, unlike SBP and the offline algorithm, it may not spend a time segment moving after each time segment it spends serving. In other words, if after serving a set of requests (or a single request) for T/f time, the greedy algorithm finds itself at a node from which another request originates, it can continue on to serve this request, rather than wait T/f time to do so. This is especially advantageous since source and destinations are nonuniformly generated, so a node that is a destination of a request is more likely to be a source of another request and the greedy algorithm will serve these two requests contiguously.

Figures 3, 4, and 5 show the results of the experimental simulations. The graphs show that for all three settings SBP is competitive (colloquially speaking) with the

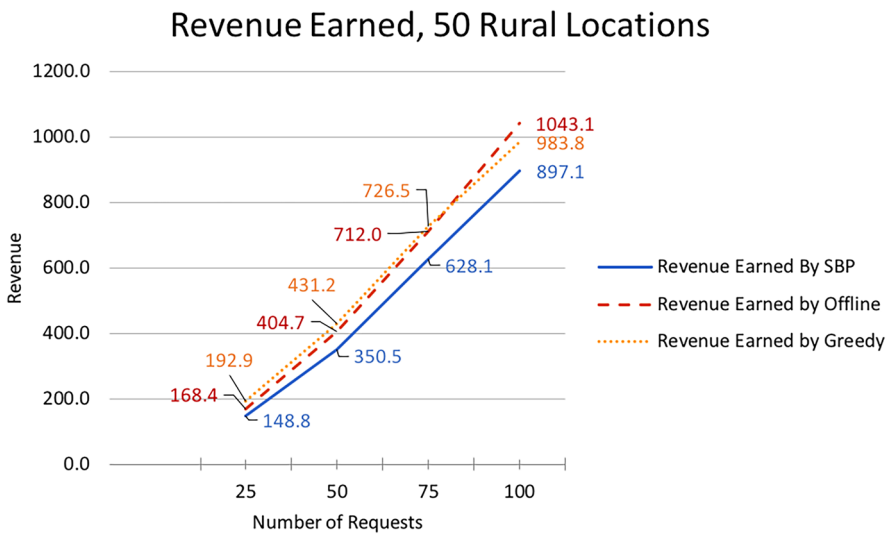

Fig. 3 Revenue earned for rural setting

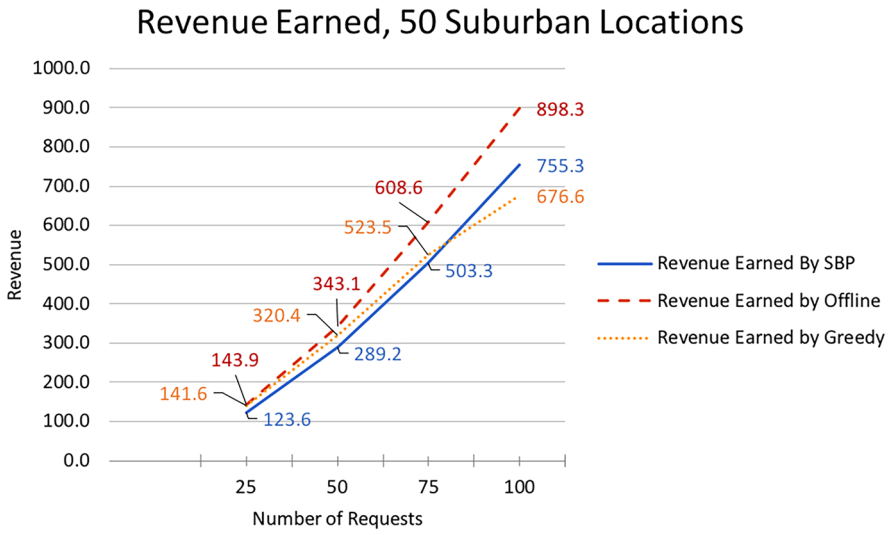

Fig. 4 Revenue earned for suburban setting

offline and greedy algorithms. Specifically, in the rural setting, SBP earns approximately 86-88% of the revenue earned by the offline algorithm and 77-91% of the revenue earned by the greedy algorithm. Furthermore, as the number of requests increases from 25 to 100, SBP's performance over the greedy algorithm also increases.

In the suburban setting, SBP earns approximately 82-85% of the revenue earned by the offline algorithm. For $m = 25, 50$, and 75 SBP earns 87-96% of the revenue earned by the greedy algorithm and for $m = 100$, SBP earns 11% more than

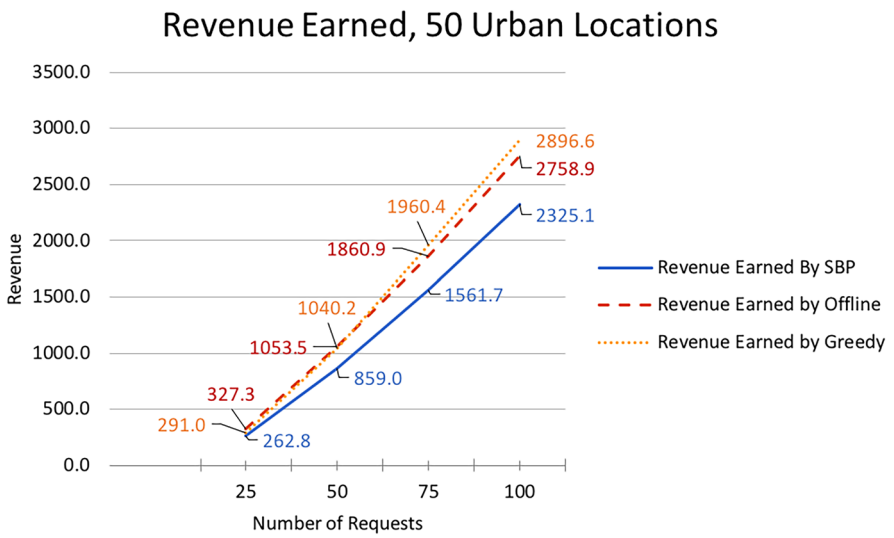

Fig. 5 Revenue earned for urban setting

the greedy algorithm. As in the rural setting, as the number of requests increases from 25 to 100, SBP's performance over the greedy algorithm increases.

Finally, in the urban setting, SBP earns approximately 80-84% of the revenue earned by the offline algorithm and 80-90% of the revenue earned by the greedy algorithm.

We note that while the greedy algorithm often outperforms SBP, it has an unbounded competitive ratio, while SBP has a constant-competitive worst-case guarantee. Specifically, the greedy algorithm will choose the highest revenue request regardless of the time it takes to serve this request; let p_{max} and T/f denote the revenue and time for this request. In the meantime, the optimal schedule may serve k requests where each has revenue slightly less than p_{max} and takes time $T/(fk)$. As k increases, the ratio of the revenue earned by the greedy algorithm to that earned by OPT approaches infinity.

Overall, these simulation results suggest that SBP would be effective in real-world on-demand dial-a-ride systems.

7.2 Experimental Results for BP

To evaluate the performance of the BP algorithm (see Section 6), we simulated an Online Dial-a-Ride system on a graph with uniform edge weights. The graph is complete and contains 50 nodes. The time limit $T = 24$ and release times of requests are integral values uniformly distributed from $t = 0$ to $t = T$. Request priorities are integral values uniformly distributed from 1 to m where m is the number of requests (we tested $m = 25, 50, 75, 100$, and 125). Recall that a request's priority represents the urgency of the request (i.e., transport to a pharmacy is more urgent than to a shopping mall).

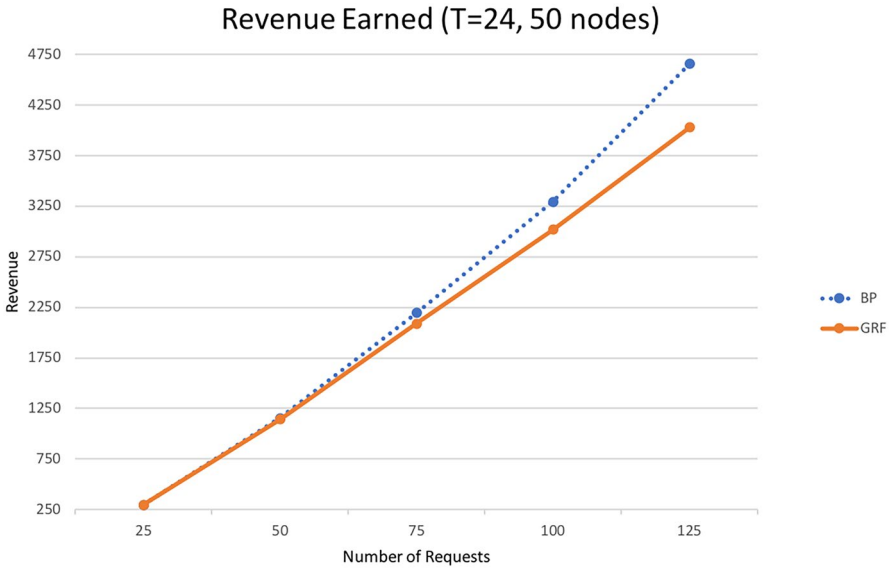

Fig. 6 Revenue earned for BP vs. GRF with 50 nodes and $T = 24$

Figure 6 shows that for all values of m, BP performs as well as or better than GRF. The figure suggests that although BP has an unbounded competitive ratio (while GRF's ratio is 2), the algorithm's approach to choosing paths that are more likely to yield higher total revenue may yield improved performance in practice.

8 Conclusions

We studied the Revenue-On-Line-Dial-a-Ride Problem (ROLDARP), provided improved upper and lower bounds for the SEGMENTED BEST PATH (SBP) algorithm, and proved that our new upper bound is tight. We also applied the algorithm to the uniform revenue variant and proved an upper bound of 4, given that the revenue earned by OPT in the last two time segments is bounded by a constant (justified by Lemma 1) and the number of time segments is bounded by a constant. This bound is nearly tight due to the lower bound instance provided in [1]. We then applied SBP to complete bipartite graphs, a problem we refer to as ROLDARP-B, and showed that ROLDARP on general graphs reduces to ROLDARP-B. Since the reduction relies on edge weights being arbitrarily small, we impose a minimum edge weight and prove that when revenues are nonuniform, SBP has competitive ratio $\lceil 1/k \rceil$, where k is the ratio between the minimum and maximum edge weights in the graph (and the revenue earned by OPT in the last two time segments is bounded by a constant). When revenues are uniform, we show that SBP is strictly $\lceil 1/k \rceil$-competitive (so the additive term of the nonuniform revenue variant can now be dropped). Finally, we provide experimental results that suggest that SBP would perform well in real-world on-demand dial-a-ride systems. Open problems include finding online algorithms that achieve or come closer to the general lower bound of 2 or proving that the general lower bound is more than 2.

Appendix

In this section we provide supplementary proofs for our results in the main sections.

Proof of general lower bound for uniform revenues

Theorem 8 *Even if revenues are uniform, no non-preemptive deterministic online algorithm for ROLDARP can be guaranteed to earn more than half the revenue earned by* OPT *in the first* $T - 2T/f$ *time units.*

Proof Consider the following instance with $f = 5$ (so there are 5 time segments of length T/f). For simplicity, we let $X = T/f$ denote the length of a time segment and therefore the maximum distance between two locations, so $T = 5X$. We let the uniform revenue be 1. Fix a positive integer k and let $\delta = X/(4k)$. Let $d(u, v)$ denote the distance between locations u and v.

The graph will consist of nodes $s_0, a_i, b_i, z_i, c_j, d_j, e_j^{(m)}, f_j^{(m)}$, $(0 \le i \le 3,\ 0 \le j \le k,$ $1 \le m \le 4k-1)$, with distances

- $d(a_2, a_3) = d(b_2, b_3) = d(z_2, z_3) = \delta$,
- $d(a_3, a_4) = d(b_3, b_4) = d(z_3, z_4) = X - \delta$,
- $d(c_i, c_j) = d(d_i, d_j) = |i-j|X/k$
- $d(e_i^{(m)}, e_j^{(m)}) = d(f_i^{(m)}, f_j^{(m)}) = |i-j|(X - m\delta)/k$,
- $d(c_k, e_0) = d(d_k, f_0) = (m-1)\delta$.

All distances are X unless otherwise stated above.

Let OPT denote an optimal offline schedule, let ON denote a deterministic online algorithm and let s_0 denote the origin, i.e., the location of ON and OPT at time 0. At time X let s be the node that ON is at or moving toward. Pick two of a_1, b_1, z_1 that are not s. Because the graph is symmetric in the a's, b's, and z's, we may w.l.o.g. pick a_1, b_1. Then ON cannot have been moving toward a_1 or b_1 at a time $t_1 < X$. The adversary releases requests $r_1 = (a_1, a_2, X, 1)$ and $r_2 = (b_1, b_2, X, 1)$. We do a proof by cases that depend on when ON headed toward a_1 or b_1.

1. Case: ON does not heads toward a_1 or b_1 before time $2X + \delta$. Then the adversary releases $r_3 = (b_2, b_0, 2X + \delta, 1)$. At time $2X + \delta$, ON has only $3X - \delta$ time left, which is insufficient to serve more than one request. So ON earns at most revenue 1 while OPT earns 2 in time $3X$ via s_0, b_1, b_2, b_0.
2. Case: ON moves from some node at time $2X \le t_1 < 2X + \delta$ to either a_1 or b_1, which we can say is a_1 without loss of generality. The adversary releases request $r_3 = (b_3, b_4, 2X + \delta, 1)$. Then ON will be at a_1 at time $t_1 + X \ge 3X$. So ON can serve at most one request because there is at most time $2X$ remaining. On the other hand, OPT can earn revenue 2 in time $3X$ via the path s_0, b_1, b_2, b_3, b_4.
3. Case: ON moves from some node at time $X < t_1 < 2X$ toward either a_1 or b_1, which we can assume w.l.o.g. to be a_1. The adversary releases request $r_3 = (b_2, b_0, 2X, 1)$. Then ON arrives at a_1 at time $t_1 + X > 2X$. There is less than time $3X$ remaining and so it is impossible to earn more than revenue 1, while OPT earns 2 in time $3X$ via s_0, b_1, b_2, b_0.
4. Case: ON moves from some node at time $t_1 = X$ toward either a_1 or b_1, which we can assume w.l.o.g. to be a_1. Then the adversary releases the requests: $r_i' = (c_{i-1}, c_i, X + \delta, 1)$, $r_i'' = (d_{i-1}, d_i, X + \delta, 1)$, for $i = 1, \ldots, k$. So ON arrives at a_1 at time $2X$.
   a. Case: ON does not head toward any of a_2, c_i, d_i before time $3X - \delta$. At time $3X - \delta$, call s the node that ON is at or heading toward. Because simultaneously exchanging $f_i^{(m)}$ with $e_i^{(m)}$ and c_i with d_i is a graph isomorphism, we can w.l.o.g. assume this s is not one of the $f_i^{(m)}$. The adversary releases $\bar{r}_i = (f_{i-1}^{(4k-1)}, f_i^{(4k-1)}, 3X - \delta, 1)$ for $i = 1, \ldots, k$. Note the chain of $\bar{r}_1, \ldots, \bar{r}_k$ has total length δ and $d(d_k, f_0^{(4k-1)}) = X - 2\delta$. There is time $2X + \delta$ remaining. That is insufficient time to serve more than $k + 1$ requests. On the other hand, OPT can earn $2k$ in time $3X$ via $s_0, d_0, (\text{pause time}\delta), d_1, \ldots, d_k, f_0^{(4k-1)}, \ldots, f_k^{(4k-1)}$.

b. Case: ON heads toward a_2 at time t_2 with $2X \le t_2 < 3X - \delta$. The adversary releases $\bar{r}_i = (f_{i-1}^{(4k-1)}, f_i^{(4k-1)}, 3X - \delta, 1)$ for $i = 1, \ldots, k$. Then ON arrives at a_2 at time $t_2 + X \ge 3X$. There is time at most $2X$ remaining; then it is not possible to serve more than k additional requests. So ON earns at most $1 + k$, while OPT can earn $2k$ in time $3X$ via s_0, d_0, (pause time δ), $d_1, \ldots, d_k, f_0^{(4k-1)}, \ldots, f_k^{(4k-1)}$.

c. Case: ON heads toward some c_{i_0} or d_{i_0} at time t_2 with $2X \le t_2 < 3X - \delta$. We can assume w.l.o.g. ON is heading toward some c_{i_0}. Let $1 \le m \le 4k - 1$ such that $2X + (m-1)\delta \le t_2 < 2X + m\delta$. The adversary releases $\bar{r}_i = (f_{i-1}^{(m)}, f_i^{(m)}, 2X + m\delta, 1)$ for $i = 1, \ldots, k$. Then ON arrives at c_{i_0} at time $t_1 + X \ge 3X + (m-1)\delta$. There is at most $2X - (m-1)\delta$ time remaining. Note that the chain $\bar{r}_1, \ldots, \bar{r}_k$ has total length $X - m\delta$, and thus it takes at least time $2X - m\delta$ to serve this chain from c_{i_0}. Thus ON can serve at most k requests. On the other hand, OPT can earn $2k$ in time $3X$ via s_0, d_0, (pause timeδ), $d_1, \ldots, d_k, f_0^{(m)}, \ldots, f_k^{(m)}$.

In all subcases of Case 4, ON earns at most revenue $1 + k$ while OPT earns $2k$. So $\frac{\text{OPT}}{\text{ON}} \ge 2k/(1+k)$.

Proof of Lower Bound for SBP with Nonuniform Revenues

The following lemma is used for the proof of the SBP lower bound (Theorem 2).

Lemma 5 *$\tau_k \ge 4kh + 1$ for $k = 1 \ldots f/2 - 2$.*

Proof Recall that for each $k = 1 \ldots f/2 - 2$, OPT' arrives at vertex u_{2k} at time $\tau_k = 1 + m(h+1) + (1+h)(k-1)$. By definition of $m = \lfloor (3hf - 4h - f + 2)/(2(h+1)) \rfloor$, we know

$$m \ge (3hf - 4h - f + 2)/(2(h+1)) - 1.$$

So

$$m(h+1) \ge (3hf - 4h - f + 2)/2 - (h+1).$$

Then

$$\tau_k \ge 1 + (3hf - 4h - f + 2)/2 - (h+1) + (1+h)(k-1).$$

We then rewrite as

$$\begin{aligned}
\tau_k \ge & 1 + (3f/2 - 2)h - f/2 + 1 - h - 1 + k - 1 + h(k-1) - 4kh + 4kh \\
\ge & 1 + 4kh + (3f/2 - 2 + k - 1 - 1 - 4k)h - f/2 - 1 + k \\
\ge & 1 + 4kh + (3f/2 - 4 - 3k)h - f/2 - 1 + k \\
\ge & 1 + 4kh + (3f/2 - 4 - 3k)h - f/2 + 4/3 + k - 1 - 4/3 \\
\ge & 1 + 4kh + (3f/2 - 4 - 3k)(h - 1/3) - 1 - 4/3.
\end{aligned}$$

Since $k \leq f/2-2$, then $3k \leq 3f/2-6$. Then $3f/2-6-3k \geq 0$. Hence $3f/2-$ $-4-3k \geq 2$. So

$$\begin{aligned}(3f/2-4-3k)(h-1/3)-1-4/3 \geq & 2(h-1/3)-2(h-1)-4/3\\ \geq & 2h-2/3-2h+2-4/3 \geq 0.\end{aligned}$$

Thus we have shown $\tau_k \geq 1+4kh$.

Proofs for Bipartite Graphs

The proof of Theorem 7 (in Section 5.2) relies on the fact that $rev(\text{SBP}) \geq rev(\text{SBP}')$ which we now prove using the following two lemmas. Recall that within the i^{th} window SBP′ serves exactly one request: the maximum revenue request served by OPT during the $(i-1)^{th}$ window. We now create another subroutine, called SBP″, that, for every window, serves the highest revenue available request. Lemma 6 shows why $rev(\text{SBP}'') \geq rev(\text{SBP}')$, and Lemma 7 shows why $rev(\text{SBP}) \geq rev(\text{SBP}'')$, so we know $rev(\text{SBP}) \geq rev(\text{SBP}')$.

Lemma 6 *Let O' and O'' denote the set of revenues of the $\mu = \lceil f/2 \rceil$ requests served by SBP′ and SBP″ by the end of window μ, sorted in descending order. Let $O'[z]$ and $O''[z]$ denote the z^{th} element of O' and O'', respectively, where $z = 1, 2, ..., \mu$. Then:*

$$rev(\text{SBP}'') = \sum_{z=1}^{\mu} O''[z] \geq \sum_{z=1}^{\mu} O'[z] = rev(\text{SBP}'). \tag{24}$$

Proof We proceed by strong induction. Recall Q_i' is the set of revenues of all requests served by SBP′ during window i. Let Q_i'' and Q_i^* denote the set of revenues of all requests served by SBP″ and OPT, respectively, during window i.

Base case. $z = 1$. We know that $O'[1] = \max\{Q_1^* \cup Q_2^* \cup ... \cup Q_{\mu-1}^*\}$.

Consider SBP″ during the last window; there are two cases: 1. SBP″ has served a request with revenue equal to or larger than $O'[1] = \max\{Q_1^* \cup Q_2^* \cup ... \cup Q_{\mu-1}^*\}$; 2. SBP″ has not served such a request.

In the first case, no matter which request SBP″ serves in the last window, $O''[1] \geq O'[1]$.

In the second case, SBP″ will choose one available request with the maximum revenue to serve in the last window, so Q_μ'' will have revenue either equal to $\max\{Q_1^* \cup Q_2^* \cup ... \cup Q_{\mu-1}^*\}$ or larger than $\max\{Q_1^* \cup Q_2^* \cup ... \cup Q_{\mu-1}^*\}$. Thus, when SBP″ is done, $O''[1] \geq O'[1]$.

Inductive case. Suppose $O''[z] \geq O'[z]$ is true for $z = 1, 2, ..., l$. Consider $z = l+1$. We will show by contradiction that $O''[l+1] \geq O'[l+1]$. Suppose $O''[l+1] < O'[l+1]$.

By the definition of O'', we know $O''[1], O''[2], ..., O''[l]$ are the l largest revenues served by SBP″, and

$$O''[1] \geq O''[2] \geq ... \geq O''[l]. \tag{25}$$

From the inductive hypothesis,

$$O''[l] \geq O'[l]. \tag{26}$$

Given the ordered nature of O', we have

$$O'[l] \geq O'[l+1]. \tag{27}$$

Given $O''[l+1] < O'[l+1]$, (26), and (27), we have

$$O''[l] \geq O'[l] \geq O'[l+1] > O''[l+1]. \tag{28}$$

The general approach of this proof is that, if the request that corresponds to the revenue $O'[l+1]$ is not the request that corresponds to any of $O''[1], O''[2], ..., O''[l]$, then the $(l+1)^{th}$ largest request selected by $_{SBP}''$ would have been $O'[l+1]$ instead of $O''[l+1]$, since $O''[l+1] < O'[l+1]$. This is a contradiction. Now we must affirm that the request corresponding to $O'[l+1]$ does not correspond to any of $O''[1], O''[2], ..., O''[l]$. To verify this precondition, we consider the two possible cases of (28).

Case 1. If $O''[l] > O'[l]$ or $O'[l] > O'[l+1]$, then given (26) and (27), the consequence of this case will be $O[l] > O'[l+1]$. More generally, taking into account (25) and (28), we have

$$O''[1] \geq O''[2] \geq ... \geq O''[l] > O'[l+1] > O''[l+1], \tag{29}$$

which tells us that revenue $O'[l+1]$ does not correspond to any of $O''[1], O''[2], ..., O''[l]$. Therefore $O'[l+1]$ should have been the $(l+1)^{th}$ largest value of the set O'' instead of $O''[l+1]$, which is a contradiction.

Case 2a. If $O''[l] = O'[l] = O'[l+1] > O''[l+1]$ and the request that corresponds to $O'[l+1]$ never corresponds to any revenue among $O''[1], O''[2], ..., O''[l]$, then $O'[l+1]$ should also have been the $(l+1)^{th}$ largest value of the set O'' instead of $O''[l+1]$, which is a contradiction.

Case 2b. If $O''[l] = O'[l] = O'[l+1] > O''[l+1]$, and the request that corresponds to $O'[l+1]$ also corresponds to $O''[g]$ for some $1 \leq g \leq l$, which indicates that

$$O''[g] = O''[g+1] = ... = O''[l] > O''[l+1]. \tag{30}$$

From the ordered nature of O' and the inductive hypothesis, we know

$$O''[g] \geq O'[g] \geq O'[g+1] \geq ... \geq O'[l+1]. \tag{31}$$

Combined with the fact that $O''[g] = O'[l+1]$, it must be the case that

$$O''[g] = O'[g] = O'[g+1] = ... = O'[l+1]. \tag{32}$$

Given that there could exist elements in O'' prior to $O''[g]$ that are equal to $O'[l+1]$, we let $O''[x]$ be the first element in O'' that is equal to $O'[l+1]$, where $1 \le x \le g$. Similar to (31), we have

$$O''[x] \ge O'[x] \ge O'[x+1] \ge ... \ge O'[l+1]. \tag{33}$$

Combining (33) and $O[x] = O'[l+1]$, we know that

$$O''[x] = O'[x] = O'[x+1] = ... = O'[l+1]. \tag{34}$$

Now we know that in O'', there are $(l-x+1)$ elements that have revenue $O'[l+1]$, while in O', there are $((l+1)-x+1) = (l-x+2)$ elements with revenue $O'[l+1]$. The set O' has one more element with revenue $O'[l+1]$ than the set O''. This is a contradiction because then SBP$''$ would have chosen the extra request in O' that has revenue $O'[l+1]$ to serve instead of directly serving $O''[l+1]$.

Lemma 7 *Let* SBP$''[i]$ *and* SBP$[i]$ *denote the* i^{th} *window served by* SBP$''$ *and SBP respectively. Define* $U = \{1, 2, \dots, \mu\}$ *as the set of all possible indices of windows. Define* A_j *and* B_j *as any two subsets of U that satisfy the following criteria*:

1. *Both* A_j *and* B_j *are of size j and are in increasing order.*
2. $A_j[m] \le B_j[m]$ *for all* $m = 1, 2, \dots, j$, *where* $A_j[m]$ *is the* m^{th} *element of* A_j *and* $B_j[m]$ *is the* m^{th} *element of* B_j.
3. *For requests served in* $\{$SBP$''[A_j[1]]$, SBP$''[A_j[2]] \dots$ SBP$''[A_j[j]]\}$, *if they are ever served by SBP, then they are served only in* $\{$SBP$[B_j[1]]$, SBP$[B_j[2]] \dots$ SBP$[B_j[j]]\}$.

Then, for all such possible A_j and B_j, we have

$$\sum_{m=1}^{j} rev(\text{SBP}''[A_j[m]]) \le \sum_{m=1}^{j} rev(\text{SBP}[B_j[m]]).$$

Proof (by induction) **Base case.** When $j = 1$, we assume $A_1[1] \le B_1[1]$ and that if the request served by SBP$''[A_1[1]]$ is served by SBP, it is served only in SBP$[B_1[1]]$. This implies that the request served in SBP$''[A_1[1]]$ is available to SBP$[B_1[1]]$, so given the greedy nature of SBP, $rev(\text{SBP}[B_1[1]]) \ge rev(\text{SBP}''[A_1[1]])$. From (24) we can say

$$\sum_{m=1}^{1} rev(\text{SBP}''[A_1[m]]) \le \sum_{m=1}^{1} rev(\text{SBP}[B_1[m]]).$$

Inductive Case. We assume that: $\sum_{m=1}^{j} rev(\text{SBP}''[A_j[m]]) \le \sum_{m=1}^{j} rev(\text{SBP}[B_j[m]])$ for all j where $1 \le j \le k$. Then we want to prove for all possible A_{k+1} and B_{k+1},

$$\sum_{m=1}^{k+1} rev(\text{SBP}''[A_{k+1}[m]]) \leq \sum_{m=1}^{k+1} rev(\text{SBP}[B_{k+1}[m]]). \tag{35}$$

There are two cases:

1. $rev(\text{SBP}''[A_{k+1}[k+1]]) \leq rev(\text{SBP}[B_{k+1}[k+1]])$
2. $rev(\text{SBP}''[A_{k+1}[k+1]]) > rev(\text{SBP}[B_{k+1}[k+1]])$

If the first case is true, then combining it with the inductive hypothesis, (35) is clearly true.

If the second case is true, we denote r as the request served in window $\text{SBP}''[A_{k+1}[k+1]]$. Then the equation of case 2 can be rewritten as

$$rev(\text{SBP}[B_{k+1}[k+1]]) < rev(r). \tag{36}$$

Eq. (36) implies that r is not available to $\text{SBP}[B_{k+1}[k+1]]$, because otherwise in window $B_{k+1}[k+1]$ SBP would have served some request(s) whose total revenue is at least the revenue of r. Suppose it is at window $\text{SBP}[B_{k+1}[h]]$ that SBP serves r where $1 \leq h < k+1$.

Define new subsets A_k and C_k where $A_k = A_{k+1} \backslash \{A_{k+1}[k+1]\}$ and $C_k = B_{k+1} \backslash \{B_{k+1}[h]\}$. It is evident that both A_k and C_k are of size k. Define $A_k[m]$ as the m^{th} element of the newly defined A_k, and $C_k[m]$ as the m^{th} element of the newly defined C_k.

Here we declare a shortcut of notation. For any set V and integers $s \leq t$, $V[s:t] = \{V[s], V[s+1], \ldots, V[t-1], V[t]\}$.

One observation is that for each element $C_k[m]$ of C_k, we have $C_k[m] \geq A_k[m]$. This is because each element of C_k will be minimized when $C_k[1] = B_{k+1}[1]$, $C_k[2] = B_{k+1}[2], \ldots, C_k[k] = B_{k+1}[k]$ respectively. Since we know $B_{k+1}[m] \geq A_{k+1}[m]$, we have $C_k[m] \geq A_k[m]$. There are two sub-cases:

1. If $\text{SBP}[B_{k+1}[h]]$ does not contain any of the requests of $\text{SBP}''[A_k[1:k]]$: Then A_k and C_k satisfy the three criteria listed in the lemma, so according to the inductive hypothesis,

$$\sum_{m=1}^{k} rev(\text{SBP}''[A_k[m]]) \leq \sum_{m=1}^{k} rev(\text{SBP}[C_k[m]]). \tag{37}$$

Adding $rev(r)$ on both sides of (37), we have

$$\sum_{m=1}^{k+1} rev(\text{SBP}''[A_{k+1}[m]]) \leq \sum_{m=1}^{k} rev(\text{SBP}[C_k[m]]) + rev(r) \tag{38}$$

$$\leq \sum_{m=1}^{k+1} rev(\text{SBP}[B_{k+1}[m]]). \tag{39}$$

Note the second $\leq$ sign is valid since all the requests corresponding to $\sum_{m=1}^{k} rev(\text{SBP}[C_k[m]]) + rev(r)$ are served in SBP (and SBP may serve additional requests as well).

2. If $\text{SBP}[B_{k+1}[h]]$ contains any of the requests of $\text{SBP}''[A_k[1:k]]$: Then A_k and C_k violate criterion (3) (since $B_{k+1}[h]$ is not in C_k) so the inductive hypothesis cannot be applied directly. Suppose there are n such requests where $n \leq k$, and denote the total revenue of those n requests as N. Then we define $A_{k-n} = A_k \backslash \{\text{indices of windows where the } n \text{ requests reside in SBP"}\}$. We also shrink C_k to C_{k-n} by removing n windows that do not contain any of the requests of A_{k-n}. Then we can follow the same reasoning above to deduce $C_{k-n}[m] \geq A_{k-n}[m]$ for all $1 \leq m \leq k-n$. So according to the inductive hypothesis,

$$\sum_{m=1}^{k-n} rev(\text{SBP}''[A_{k-n}[m]]) \leq \sum_{m=1}^{k-n} rev(\text{SBP}[C_{k-n}[m]]). \tag{40}$$

Adding N and $rev(r)$ on both sides, we have

$$\sum_{m=1}^{k+1} rev(\text{SBP}''[A_{k+1}[m]]) \leq \sum_{m=1}^{k-n} rev(\text{SBP}[C_{k-n}[m]]) + N + rev(r) \tag{41}$$

$$\leq \sum_{m=1}^{k+1} rev(\text{SBP}[B_{k+1}[m]]). \tag{42}$$

Note the second $\leq$ sign of (41) is valid since all the requests corresponding to $\sum_{m=1}^{k-n} rev(\text{SBP}[C_{k-n}[m]]) + N + rev(r)$ are served in SBP (and SBP may serve additional requests as well).

Finally, to prove $rev(\text{SBP}) \geq rev(\text{SBP}'')$, we let $j = \mu$, $A_\mu = \{1, 2, \ldots, \mu\}$, $B_\mu = \{1, 2, \ldots, \mu\}$. This satisfies (1) A_μ and B_μ are in increasing order, (2) $A_\mu[m] \leq B_\mu[m]$ for all $m = 1, 2, \ldots, \mu$, and (3) for requests served in $\{\text{SBP}''[A_\mu[1]], \text{SBP}''[A_\mu[2]] \ldots \text{SBP}''[A_\mu[\mu]]\}$, if they are ever served by SBP, are served only in windows $\{\text{SBP}[B_\mu[1]], \text{SBP}[B_\mu[2]] \ldots \text{SBP}[B_\mu[\mu]]\}$. Therefore, $rev(\text{SBP}) \geq rev(\text{SBP}'')$ is simply a specific case of $\sum_{m=1}^{j} rev(\text{SBP}''[A_j[m]]) \leq \sum_{m=1}^{j} rev(\text{SBP}[B_j[m]])$ where $j = \mu$.

Funding Information This study was not funded by a grant.

Data Availability The datasets generated and/or analyzed during the current study are available from the corresponding author on reasonable request.

Code Availability The code used to generate results for this study are available from the corresponding author on reasonable request.

Declarations

Conflict of Interest On behalf of all authors, the corresponding author states that there is no conflict of interest.

References

1. Christman A, Chung C, Jaczko N, Milan M, Vasilchenko A, Westvold S (2017) Revenue maximization in online dial-a-ride. In: 17th Workshop on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems (ATMOS 2017). Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik
2. Christman A, Chung C, Jaczko N, Li T, Westvold S, Xu X, Yuen D (2020) New bounds for maximizing revenue in online dial-a-ride. In: 31st International Workshop on on Combinatorial Algorithms (IWOCA 2020)
3. Feuerstein E, Stougie L (2001) On-line single-server dial-a-ride problems. Theor Comput Sci 268(1):91–105
4. Jaillet P, Wagner MR (2008) Online vehicle routing problems: A survey. The Vehicle Routing Problem: Latest Advances and New Challenges. pp. 221–237
5. Molenbruch Y, Braekers K, Caris A (2017) Typology and literature review for dial-a-ride problems. Ann Oper Res 259(1–2):295–325
6. Krumke SO (2002) Online optimization: Competitive analysis and beyond
7. Birx A, Disser Y, Schewior K (2019) Improved bounds for open online dial-a-ride on the line. In: Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2019). Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik
8. Ascheuer N, Krumke SO, Rambau J (2000) Online dial-a-ride problems: Minimizing the completion time. In: Annual Symposium on Theoretical Aspects of Computer Science. pp. 639–650. Springer
9. Birx A, Disser Y (2019) Tight analysis of the smartstart algorithm for online dial-a-ride on the line. In: 36th International Symposium on Theoretical Aspects of Computer Science
10. Birx A, Disser Y (2020) Tight analysis of the smartstart algorithm for online dial-a-ride on the line. SIAM J Discrete Math 34(2):1409–1443
11. Bjelde A, Hackfeld J, Disser Y, Hansknecht C, Lipmann M, Meißner J, SchlÖter M, Schewior K, Stougie L (2020) Tight bounds for online tsp on the line. ACM Transactions on Algorithms (TALG) 17(1):1–58
12. Ausiello G, Feuerstein E, Leonardi S, Stougie L, Talamo M (2001) Algorithms for the on-line travelling salesman. Algorithmica 29(4):560–581
13. Krumke SO, de Paepe WE, Poensgen D, Lipmann M, Marchetti-Spaccamela A, Stougie L (2005) On minimizing the maximum flow time in the online dial-a-ride problem. In: International Workshop on Approximation and Online Algorithms. pp. 258–269. Springer
14. Balas E (1989) The prize collecting traveling salesman problem. Networks 19(6):621–636
15. Ausiello G, Bonifaci V, Laura L (2008) The online prize-collecting traveling salesman problem. Inf Process Lett 107(6):199–204
16. Anthony B, Boyd S, Birnbaum R, Christman A, Chung C, Davis P, Dhimar J, Yuen D (2019) Maximizing the number of rides served for dial-a-ride. In: 19th Workshop on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems (ATMOS 2019). Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik
17. Christman A, Forcier W, Poudel A (2018) From theory to practice: maximizing revenues for on-line dial-a-ride. J Comb Optim 35(2):512–529
18. Corporation S. Dial-a-ride, http://stagecoach-rides.org/dial-a-ride/
19. Council M. Transit link: Dial-a-ride small bus service, https://metrocouncil.org/Transportation/Services/Transit-Link.aspx
20. City of Plymouth M. Plymouth metrolink dial-a-ride, http://www.plymouthmn.gov/departments/administrative-services-/transit/plymouth-metrolink-dial-a-ride