New Bounds for Maximizing Revenue in Online Dial-a-Ride

Ananya Christman[1][0000−0001−9445−1475], Christine Chung[2][0000−0003−3580−9275], Nicholas Jaczko[1], Tianzhi Li[1], Scott Westvold[1], Xinyue Xu[1], and David Yuen[3]

[1] Middlebury College, Middlebury, VT 05753, USA achristman@middlebury.edu
[2] Connecticut College, New London, CT 06320, USA cchung@conncoll.edu
[3] 92-1507 Punawainui St. Kapolei, HI 96707, USA yuen888@hawaii.edu

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 prove that no non-preemptive deterministic online algorithm for OLDARP can be guaranteed to earn more than half the revenue earned by OPT. We then investigate the SEGMENTED BEST PATH (SBP) algorithm of [8] for the general case of weighted graphs. The previously-established lower and upper bounds for the competitive ratio of SBP are 4 and 6, respectively, under reasonable assumptions about the input instance. We eliminate the gap by proving that the competitive ratio is 5 (under the same assumptions). We also prove that when revenues are uniform, SBP has competitive ratio 4. Finally, we provide a competitive analysis of SBP on complete bipartite graphs.

1 Introduction

In the On-Line 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, and the *release time* of the request, 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 the total travel time (i.e. makespan) to satisfy all requests, minimizing the average completion time (i.e. latency), 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. On-Line Dial-a-Ride Problems have many practical applications in settings where a vehicle is dispatched to satisfy requests involving pick-up and delivery of people or goods. Important examples include ambulance routing, transportation for the elderly and disabled,

taxi services 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 simply 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 [10] 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 [12] and [16]. Feuerstein and Stougie studied the problem for two different objectives: minimizing completion time and minimizing latency. For minimizing completion time, they showed that any deterministic algorithm must have competitive ratio of at least 2 regardless of the server capacity. They presented algorithms for the cases of finite and infinite capacity with competitive ratios of 2.5 and 2, respectively. For minimizing latency, they proved that any algorithm must have a competitive ratio of at least 3 and presented a 15-competitive algorithm on the real line when the server has infinite capacity. Ascheuer et al. [2] studied OLDARP with multiple servers with the goal of minimizing completion time and presented a 2-competitive algorithm. More recently, Birx et al. [5] studied OLDARP on the real line and presented a new upper bound of 2.67 for the SMARTSTART algorithm [2], which improves the previous bounds of 3.41 [14] and 2.94 [4]. For OLDARP on the real line, Bjelde et al. [6] present a preemptive algorithm with competitive ratio 2.41. The Online Traveling Salesperson Problem (OLTSP), introduced by Ausiello et al. [3] and also studied by Krumke [15], is a special case of OLDARP where for each request the source and destination are the same location. There are many studies of variants of OLDARP and OLTSP [3, 11, 13, 15] that differ from the variant that we study which we omit here due to space limitations.

In this paper, we study OLDARP where 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 [8]. More recently, it was shown that even the special case of the offline version with uniform revenues and uniform weights is NP-hard [1]. Christman and Forcier [9] presented a 2-competitive algorithm for OLDARP on graphs with uniform edge weights. Christman et al. [8] showed that if edge

weights may be arbitrarily large, then regardless of revenue values, no deterministic algorithm can be competitive. They therefore considered graphs where edge weights are bounded by a fixed fraction of the time limit, and gave a 6-competitive algorithm for this problem. Note that this 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.

1.2 Our results

In this work we begin with improved lower and upper bounds for the competitive ratio of the
SEGMENTED BEST PATH (SBP) algorithm that was presented in [8]. We study SBP because it has the best known competitive ratio for ROLDARP and is a relatively straightforward algorithm. In [8], it was shown 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. This assumption is imposed because, as we show in Lemma 1, *no* non-preememptive 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.

We then 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) and 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). This variant is useful 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 then 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 on general graphs reduces to ROLDARP-B. We therefore impose a minimum edge weight of kT/f for some constant k such that $0 < k \leq 1$. 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. (This assumption is justified by Lemma 1 which says no

	Competitive ratio ρ of SBP for ROLDARP	
	Uniform Revenue	**Nonuniform Revenue**
Weighted graphs	$\rho = 4^{\dagger\ddagger}$ ([8], [this work])	$\rho = 5^{\dagger}$ [this work]
Weighted bipartite graphs	$\rho \leq \lceil 1/k \rceil^{\S}$ [this work]	$\rho \leq \lceil 1/k \rceil^{\dagger\S}$ [this work]

Table 1: Bounds on the algorithm SBP for ROLDARP variants. † This upper bound assumes the optimal revenue of the last two time segments is bounded by a constant. ‡ This upper bound assumes the number of time segments is constant. § k is a constant where $0 < k \leq 1$ such that the minimum edge weight is kT/f where T is the time limit and $1 < f < T$.

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.

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 amount of time it takes to traverse (u, v). [4] 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 $t = 0$. For each request, the server must decide whether to serve the request and if so, at what time. A request may not be served earlier than its release time and at most one request may be served at any given time. Once the server decides to serve a request, it must do so until completion. 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.)

[4] 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.

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

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

The algorithm SEGMENTED BEST PATH (SBP) [8] starts by splitting the total time T into f segments each of length T/f (recall that f is fixed and $1 < f < T$). 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 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 [8] for full details).

It was observed in [8] that no deterministic online algorithm can be guaranteed to serve the requests served by OPT during the last time segment and the authors 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 $\text{rev}(\text{SBP}(t_j))$ and $\text{rev}(\text{OPT}(t_j))$ denote the revenue earned by SBP and OPT respectively during the j-th time segment. Then if $\text{rev}(\text{OPT}(t_f)) + \text{rev}(\text{OPT}(t_{f-1})) \leq c$ for some constant c, then $\sum_{j=1}^{f} \text{rev}(\text{OPT}(t_j)) \leq 6 \sum_{j=1}^{f} \text{rev}(\text{SBP}(t_j)) + c$. It was also shown in [8] that as T grows, the competitive ratio of SBP is at best 4 (again with the additive term equal to $\text{rev}(\text{OPT}(t_f)) + \text{rev}(\text{OPT}(t_{f-1}))$), resulting in a gap between the upper and lower bounds.

2.1 General lower bound

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

Theorem 1. *No non-preemptive deterministic online algorithm for OLDARP can be guaranteed to earn more than half the revenue earned by* OPT *in the first* $T - 2T/f$ *time units. This is the case whether revenues are uniform or nonuniform.*

Proof (Sketch). The adversary repeatedly releases requests such that depending on which request(s) the algorithm serves, other request(s) are released that the algorithm cannot serve in time. This scheme requires carefully constructed edge weights, release times, and revenues so that the optimal offline revenue is always twice that of any online algorithm. Please see the full version of the paper for details [7].

We now show that *no* non-preemptive deterministic online algorithm (e.g. SBP) can be competitive with the revenue earned by OPT in the last two segments of time. We note that this claim applies to the version of non-preemption where, as in real-world systems like Uber/Lyft, once the server decides to serve a request, it must move there and serve it to completion.

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 (). The adversary releases a request in the last two time segments and if the online algorithm chooses not to serve it, no other requests will be released. If the algorithm chooses to serve it, another batch of requests will be released elsewhere that the algorithm cannot serve in time. Please see the full version of the paper for details [7].

3 Nonuniform revenues

In this section we improve the lower and upper bounds for the competitive ratio of the SEGMENTED BEST PATH algorithm [8]. In particular, we eliminate the gap between the lower and upper bounds of 4 and 6, respectively, from [8], by providing an instance where the lower bound is 5 and a proof for an upper bound of 5. Note that throughout this section we assume the revenue earned by OPT in the last two time segments is bounded by some constant. We must impose this restriction on the OPT revenue of the last two time segments because, as we showed in Lemma 1, *no* non-preemptive deterministic online algorithm can be guaranteed to earn any constant fraction of this revenue.

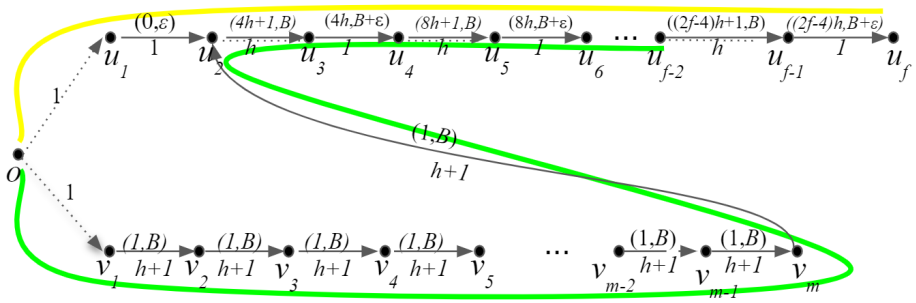

Fig. 1: An instance where OPT (whose path is shown in green below) earns $5-4/(f-2)$ times the revenue of SBP (shown in yellow above). In this instance, $T = 2hf$, and edges that represent requests are shown as solid edges. For each such edge the release time followed by revenue of the corresponding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

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 (Sketch). For the formal details, please refer to the proof of Theorem 2 in the full version [7]. Consider the instance depicted in Figure 1. Since $T = 2hf$ in this instance, h represents "half" the length of one time segment, so only one request of length $h + 1$ fits within a single time segment for SBP. The general idea of the instance is that while SBP is serving every other request across the top row of requests (since the other half across the top are not released until after SBP has already passed them by), OPT is serving the entire bottom row in one long chain, then also has time to serve the top row as one long chain.

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 [8], 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(OPT) and rev(SBP) denote the total revenue earned by OPT and SBP over all time segments t_j from $j = 1 \dots f$.

Theorem 3. *If the revenue earned by* OPT *in the last two time segments is bounded by some constant c, then* SBP *is 5-competitive, i.e., if* $\text{rev}(\text{OPT}(t_f)) + \text{rev}(\text{OPT}(t_{f-1})) \leq c$, *then* $\sum_{j=1}^{f} \text{rev}(\text{OPT}(t_j)) \leq 5 \sum_{j=1}^{f} \text{rev}(\text{SBP}(t_j)) + c$. *Note that 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 $\text{rev}(\text{OPT}) - \text{rev}(B)$, where $\text{rev}(B)$ 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. $\overline{\text{OPT}}_i = \arg\max\{\text{rev}(\overline{\text{OPT}}(t_{2i-1})), \text{rev}(\overline{\text{OPT}}(t_{2i}))\}$. 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 $\text{rev}(\overline{\text{OPT}_i})$ 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 $\text{rev}(\text{SBP}'_i)$ denote the revenue earned by SBP'_i. Let H denote the chronologically ordered set of time windows w where $\text{rev}(\overline{\text{OPT}}_w) > \text{rev}(\text{SBP}'_w)$, 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 $\text{rev}(C_j) \geq \text{rev}(C_j^*)$ 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: $\text{rev}(C_j) \geq \max\{\text{rev}(r^*), \text{rev}(C_j^* \setminus \{r^*\})\} \geq \frac{1}{2}\,\text{rev}(C_j^*)$. 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: $\text{rev}(\overline{\text{OPT}}_{h_j}) = \text{rev}(A_j) + \text{rev}(C_j^*) \leq \text{rev}(A_j) + 2 \cdot \text{rev}(C_j) = \text{rev}(\overline{\text{SBP}}_{h_j})$.

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 $\text{rev}(A_j) + \text{rev}(C_j) \geq \text{rev}(C_j^*)$, therefore we know either $\text{rev}(A_j) \geq \frac{1}{2}\,\text{rev}(C_j^*)$ or $\text{rev}(C_j) \geq \frac{1}{2}\,\text{rev}(C_j^*)$. 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 $\text{rev}(\overline{\text{OPT}}_{h_j}) \leq \text{rev}(\overline{\text{SBP}}_{h_j})$, 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: $\text{rev}(\overline{\text{OPT}}_{h_j}) = \text{rev}(A_j) + \text{rev}(C_j^*) \leq 2 \cdot \text{rev}(A_j) + \text{rev}(C_j) = \text{rev}(\overline{\text{SBP}}_{h_j})$.

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

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

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

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

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

$$\sum_{j=1}^{f} \text{rev}(\overline{\text{OPT}}(t_j)) \leq 2 \sum_{i=1}^{\lceil f/2 \rceil} \text{rev}(\overline{\text{OPT}}_i)$$
$$\leq 4 \sum_{j=1}^{f} \text{rev}(\text{SBP}(t_j)) + \text{rev}(\overline{\text{OPT}}(t_{f-1})) + \text{rev}(\overline{\text{OPT}}(t_f)). \quad (3)$$

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 $\text{rev}(B) \leq \text{rev}(\text{SBP}')$, so $\text{rev}(B) \leq \text{rev}(\text{SBP})$. By the definition of OPT, $\text{OPT} = \overline{\text{OPT}} + B$, so

$$\sum_{j=1}^{f} \text{rev}(\text{OPT}(t_j)) = \text{rev}(B) + \sum_{j=1}^{f} \text{rev}(\overline{\text{OPT}}(t_j)) \quad (4)$$

And by combining (3)-(4) with the fact that $\text{rev}(B) \leq \text{rev}(\text{SBP})$, we have $\sum_{j=1}^{f} \text{rev}(\text{OPT}(t_j))$
$\leq \sum_{j=1}^{f} \text{rev}(\text{SBP}(t_j)) + 4 \sum_{j=1}^{f} \text{rev}(\text{SBP}(t_j)) + \text{rev}(\overline{\text{OPT}}(t_{f-1})) + \text{rev}(\overline{\text{OPT}}(t_f))$
$\leq 5 \sum_{j=1}^{f} \text{rev}(\text{SBP}(t_j)) + \text{rev}(\overline{\text{OPT}}(t_{f-1})) + \text{rev}(\overline{\text{OPT}}(t_f))$.

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 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 [8] immediately establishes a lower bound instance for SBP that has a ratio of 4. 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 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. $S^*_i = \arg\max\{\text{rev}(\text{OPT}(t_{2i-1}), \text{rev}(\text{OPT}(t_{2i}))\}$ where $\text{rev}(\text{OPT}(t_j))$ 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. $J^*_i = \arg\min\{\text{rev}(\text{OPT}(t_{2i-1}), \text{rev}(\text{OPT}(t_{2i}))\}$.

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 lemma whose proof has been deferred to the full version of the paper [7]. It states that at any given time window, the cumulative requests of OPT that were earlier served by SBP are no more than the number that have been served by SBP but not yet by OPT.

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

We are now ready to prove our 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* $\text{rev}(\text{OPT}(t_f)) + \text{rev}(\text{OPT}(t_{f-1})) \leq c$, *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. (So if f is also bounded by some constant, then* SBP *is 4-competitive). I.e.,* $\sum_{j=1}^{f} \text{rev}(\text{OPT}(t_j)) \leq 4 \sum_{j=1}^{f} \text{rev}(\text{SBP}(t_j)) + 2\lceil f/2 \rceil + c$.

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., $\text{rev}(U) = |U|$. Consider each window i where $\text{rev}(S^*_i) > \text{rev}(S'_i)$. 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

$\text{rev}(X_i)+\text{rev}(Y_i) \geq \text{rev}(Y_i^*)$. 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
   $$\text{rev}(X_i) + \text{rev}(Y_i) \geq \text{rev}(r^*) \tag{5}$$
   since otherwise SBP$'$ would have chosen to serve r^*. We also know
   $$\text{rev}(X_i) + \text{rev}(Y_i) \geq \text{rev}(Y_i^* \backslash \{r^*\}) \tag{6}$$
   since otherwise SBP$'$ would have chosen to serve $Y_i^* \backslash \{r^*\}$.
   From (5), we have $|X_i| + |Y_i| \geq 1$ and from (6), 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 $\text{rev}(X_i) + \text{rev}(Y_i) \geq \text{rev}(Y_i^*)$ 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 $\text{rev}(A_i) + \text{rev}(X_i) + \text{rev}(Y_i) \geq \text{rev}(A_i) + \text{rev}(Y_i^*) - \text{rev}(r^*)$ which means $\text{rev}(X_i) + \text{rev}(Y_i) \geq \text{rev}(Y_i^*) - \text{rev}(r^*)$ 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 $m = \lceil f/2 \rceil - 1$,

$$\sum_{i=1}^{m} (|Y_i^*| - |X_i|) \leq m + \sum_{i=1}^{m} |Y_i|. \tag{7}$$

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 $\text{rev}(\overline{\text{SBP}}) \geq \sum_{i=1}^{m} \text{rev}(S_i^*)$.

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

By Lemma 2 and equation (7), we can say $\text{rev}(\text{OPT}') - \text{rev}(\text{SBP}') = \sum_{i=1}^{m} |Y_i^*| - \sum_{i=1}^{m} |X_i| + \sum_{i=1}^{m} |X_i^*| - \sum_{i=1}^{m} |Y_i| \leq \sum_{i=1}^{m} |Y_i^*| - \sum_{i=1}^{m} |X_i| \leq m + \sum_{i=1}^{m} |Y_i|$. 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,

$$\mathrm{rev}(\overline{\mathrm{SBP}}) = \mathrm{rev}(\mathrm{SBP}') + m + \sum_{i=1}^{m} |Y_i| \geq \mathrm{rev}(\mathrm{OPT}'). \tag{8}$$

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

$$\mathrm{rev}(\overline{\mathrm{SBP}}) = \mathrm{rev}(\mathrm{SBP}') + m + \sum_{i=1}^{m} |Y_i| \leq 2\,\mathrm{rev}(\mathrm{SBP}') + m. \tag{9}$$

Combining (8) and (9), we get $\mathrm{rev}(\mathrm{OPT}') \leq 2\,\mathrm{rev}(\mathrm{SBP}') + m$, which means

$$\sum_{i=1}^{m} \mathrm{rev}(S_i^*) \leq 2 \sum_{i=1}^{m} \mathrm{rev}(S_i') + m. \tag{10}$$

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, $\sum_{j=1}^{2m} \mathrm{rev}(S^*(t_j)) \leq 2 \sum_{i=1}^{m} \mathrm{rev}(S_i^*)$, which, combined with (10), gives us

$$\sum_{j=1}^{2m} \mathrm{rev}(S^*(t_j)) \leq 4 \sum_{i=1}^{m} \mathrm{rev}(S_i') + 2m. \tag{11}$$

We assumed that the total revenue of requests served in the last two time segments by OPT is bounded by c. From (11), we get

$$\sum_{j=1}^{f} \mathrm{rev}(S^*(t_j)) \leq \sum_{j=1}^{2m} \mathrm{rev}(S^*(t_j)) + \mathrm{rev}(S^*(t_{f-1})) + \mathrm{rev}(S^*(t_f)) \leq 4 \sum_{i=1}^{m} \mathrm{rev}(S_i') + 2m + c. \tag{12}$$

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 (12), we have $\sum_{j=1}^{f} \mathrm{rev}(S^*(t_j)) \leq 4 \sum_{j=1}^{f} \mathrm{rev}(S(t_j)) + 2m + c$.

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 maybe 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). Due strictly to space limitations, most proofs of theorems in this section are deferred to the full version of the paper [7].

We refer to this problem as ROLDARP-B and the offline version as RDARP-B. We first show that if edge weights of the bipartite graph are not bounded by a minimum value, then the offline version of ROLDARP on general graphs, which

we refer to as RDARP, reduces to RDARP-B. Since RDARP has been show in [8, 1] to be NP-hard (even if revenues are uniform), this means RDARP-B is NP-hard as well.

Theorem 5. *The problem RDARP is poly-time reducible to RDARP-B. Also, RDARP with uniform revenues is poly-time reducible to RDARP-B with uniform revenues.*

Proof (Sketch). The idea of the reduction is to split each node into two nodes connected by an edge in the bipartite graph with a distance of ϵ. Then we turn each edge in the original graph into two edges in the bipartite graph. Please see the full version for details [7].

5.1 Uniform revenue bipartite

We show that for bipartite graph instances, 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* $\text{rev}(\text{OPT}) \leq \lceil 1/k \rceil \cdot \text{rev}(\text{SBP}) + \lceil 1/k \rceil$.

Proof (Sketch). The proof idea is akin to that of Theorem 7 below. Please see the full version of the paper for details [7].

5.2 Nonuniform revenue bipartite

In this section we show that even if revenues are nonuniform, we can still 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. Recall that we refer to each pair of consecutive time segments as a time window. Note that *no* non-preemptive deterministic online algorithm can be competitive with any fraction of the revenue earned by OPT in the last $2T/f$ time units (i.e. Lemma 1 also holds for ROLDARP-B with nonuniform revenues). Due space limitations, please refer to the full version of this work [7] for the proof of the following theorem.

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

References

1. Anthony, B., Boyd, S., Birnbaum, R., Christman, A., Chung, C., Davis, P., Dhimar, J.: 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 (2019)
2. Ascheuer, N., Krumke, S.O., Rambau, J.: Online dial-a-ride problems: Minimizing the completion time. In: Annual Symposium on Theoretical Aspects of Computer Science. pp. 639–650. Springer (2000)
3. Ausiello, G., Feuerstein, E., Leonardi, S., Stougie, L., Talamo, M.: Algorithms for the on-line travelling salesman 1. Algorithmica **29**(4), 560–581 (2001)
4. Birx, A., Disser, Y.: Tight analysis of the smartstart algorithm for online dial-a-ride on the line. In: 36th International Symposium on Theoretical Aspects of Computer Science (2019)
5. Birx, A., Disser, Y., Schewior, K.: 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 (2019)
6. Bjelde, A., Disser, Y., Hackfeld, J., Hansknecht, C., Lipmann, M., Meißner, J., Schewior, K., Schlöter, M., Stougie, L.: Tight bounds for online tsp on the line. In: Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms. pp. 994–1005. Society for Industrial and Applied Mathematics (2017)
7. Christman, A., Chung, C., Jaczko, N., Li, T., Westvold, S., Xu, X., Yuen, D.: New bounds for maximizing revenue in online dial-a-ride. arXiv preprint arXiv:1912.06300 (2020)
8. Christman, A., Chung, C., Jaczko, N., Milan, M., Vasilchenko, A., Westvold, S.: 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 (2017)
9. Christman, A., Forcier, W.: Maximizing revenues for on-line dial-a-ride. In: International Conference on Combinatorial Optimization and Applications. pp. 522–534. Springer (2014)
10. Feuerstein, E., Stougie, L.: On-line single-server dial-a-ride problems. Theoretical Computer Science **268**(1), 91–105 (2001)
11. Jaillet, P., Wagner, M.R.: Generalized online routing: New competitive ratios, **56**(3), 745–757 (2008)
12. Jaillet, P., Wagner, M.R.: Online vehicle routing problems: A survey. The Vehicle Routing Problem: Latest Advances and New Challenges pp. 221–237 (2008)
13. Jawgal, V.A., Muralidhara, V., Srinivasan, P.: Online travelling salesman problem on a circle. In: International Conference on Theory and Applications of Models of Computation. pp. 325–336. Springer (2019)
14. Krumke, S.O.: Online optimization: Competitive analysis and beyond (2002)
15. Krumke, S.O., de Paepe, W.E., Poensgen, D., Lipmann, M., Marchetti-Spaccamela, A., Stougie, L.: 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 (2005)
16. Molenbruch, Y., Braekers, K., Caris, A.: Typology and literature review for dial-a-ride problems. Annals of Operations Research **259**(1-2), 295–325 (2017)