On the Power of Greedy Approaches: Online Car Sharing with two servers on Two Locations, with varying booking times and paired requests

l.j.l.j.hoofs

June 14, 2022

Contents

1 Introduction

This is a Computing Science Master's Thesis about the *online car-sharing* problem. This document contains the following: An explanation of the online car-sharing problem, an analysis of work done on this problem, a glimpse into the work on closely related problems, a clear indication of the scope of our thesis, a compilation of techniques used in prior research on the topic, and finally, our contributions to the field.

The car-sharing problem is named after its basis in car-sharing services. Such services provide cars to people when and where they ask for them. These people then use the car to get from the location they asked the car from, to the location they want to go, where they leave the car for someone else to pick-up. Such rent-able cars are helpful for, for example, tourists who arrived by airplane, and do not have their own car available to them. Another advantage of rent-able cars is that high usage of such cars could reduce the amount of privately owned cars. Therefore, this problem is not just of theoretical, but also of practical interest.

The problem is also more widely applicable than just car-sharing services, such as Uber. Another application could be a river crossing that operates on demand, ferrying people from one side to the other. Bike sharing systems have become more popular as of late, as well. It is therefore common practice to refer to servers instead of cars. Though the basic car-sharing problem might oversimplify some of the practical problems it is trying to solve, any good solution needs a solid foundation. And even then, the solutions to the basic problem might provide some helpful insight that would help improve these systems, be that for the company or the customer.

Intuitively, the car-sharing problem consists of a series of requests from customers to be moved from one location to another at a certain time. It is up to the car-sharing company to decide whether or not to accept each of these requests. If a request is accepted, the company must provide a server at the first location at the requested time, that proceeds to move the customer to the new location. This action provides the company with a certain profit. The goal of the car-sharing problem is often to maximize that profit, though there are variations that instead implicitly try to maximize the amount of served customers.

2 Definitions and Problems

In this chapter, we formalize the definition of the general car sharing problem. The general car sharing problem is a very broad problem. Many different parameters influence the results algorithms can achieve on this problem. That is the reason why we have decided to limit our scope in this thesis. We discuss the manner in which we have limited our scope after discussing the general definition of the car sharing problem.

Generally, consider a setting with m locations and k servers (denoted $s_1, s_2, ..., s_k$). These two values denote the most common way to describe the car-sharing problem: *the kSmL notation.* This notation is used to quickly provide information about the amount of servers and locations. For example, 2S2L refers to the car sharing problem, in which an algorithm has two servers to serve requests that release at two different locations. Let $\mathcal{I}$ be an instance of a series of requests released over time. Customers request servers over time and the decision whether or not to serve such a request must be made immediately and irrevocably, without any knowledge of future requests. Requests arriving can be handled in a multitude of ways, most commonly by arbitrarily deciding one arrives before the other. Denote request r as $r = (b_r, t_r, l_r, \bar{l}_r)$ in which b_r is the *booking time* of the request, t_r is the *pick-up time* or *starting time* of the request, l_r is the *pick-up location* of the request and $\bar{l}_r$ is the *drop-off location* of the request. Also, let $\tau(l, o)$ denote the *travel time* between location l and location o. Furthermore, let $\bar{t}_r = t_r + \tau(l_r, \bar{l}_r)$ denote the *drop-off time* of request r. For the purposes of this thesis, we assume that $\tau(l, o) = \tau(o, l)$.

The kSmL notation is often accompanied by a dash and then another letter denoting some other quality about the problem, usually concerning the booking time of the problem. The additive *-F* signifies a *fixed booking time.* This means that for every request r the time between the booking time b_r and the starting time t_r is some constant $\phi = t_r - b_r$. We call this $t_r - b_r$ the *notification interval.* More flexibility is provided in the *variable booking time* variant, designated by *-V*. This variant gives a time frame, denoted by a maximum notification interval υ and a minimum notification interval λ in which the booking for a certain start time should take place. More formally: $t - \upsilon \leq b \leq t - \lambda \; \forall r \in \mathcal{I}$. The variable booking time variant has one major difference from the fixed booking time variant. In the fixed booking time variant, requests are served in the order that they are booked. This need not be the case in the variable booking time variant. We call such requests served prior to a request that was booked earlier a *line-skipping request.* Formally, a request $r = (b_r, t_r, l_r)$ line-skips a request $j = (b_j, t_j, l_j)$ if $b_r > b_j$ and $t_r < t_j$.

We assume each server can serve at most one request at a time. If two requests are such that a server cannot serve both, we say these requests are *in conflict.* Formally, requests r and j with $t_r \leq t_j$ are in conflict if $t_j < \bar{t}_r + \tau(\bar{l}_r, l_j)$. Multiple different servers are needed to serve requests that are in conflict. A request r is called *acceptable* if there is a server that serves no request that is in conflict with r. Serving a request r grants a certain *reward* p_r. This reward usually is either a constant $p > 0$ or dependent on $\tau(l_r, \bar{l}_r)$. It is often allowed to move a server from one location to another whilst not serving a request, especially when doing so to serve a request immediately afterward. Such a move is called an *empty movement* or *empty move.* Performing an empty move for a request j accrues a cost $c_j \geq 0$, which might be a constant, or dependent on the distance travelled. However, *unprompted moves*, which are empty moves that are performed despite not having a released request mandating such a move, are often strictly prohibited. Empty moves made specifically to pick-up requests immediately after are also referred to as *distant pick-ups.*

The basic goal of the car-sharing problem is create an algorithm that finds a set of served requests R^A to maximize the *total profit* $P_{R^A} = \sum_{r \in R^A} p_r - \sum_{j \in R_e^A} c_j$, where R_e^A is the set of accepted requests that required an empty move to be served. The online variant focuses on trying to achieve a profit as close as possible to what an optimal offline algorithm could provide. To this purpose the *competitive analysis* technique is used [5]. The competitive ratio of algorithm A is defined as $\rho_A = \sup_R \frac{P_{R^{OPT}}}{P_{R^A}}$, in which R^{OPT} is the set of requests chosen by the optimal scheduler. Furthermore, $P_{R^{OPT}}$ stands for the profit earned by the optimal scheduler. We say A is *ρ-competitive* or has an *upper bound* of ρ if $P_{R^{OPT}} \leq \rho \cdot P_{R^A}$ for every possible request sequence R. On top of that, β is considered a *lower bound* on the competitive ratio if $\rho_A \geq \beta$ for all possible online algorithms A. With that basic outline, many different problems can be investigated by altering the parameters. For example, setting $c_j = 0$, the goal becomes equivalent to serving as many customers as possible. Fiddling around with these parameters leads to many angles that have not yet been investigated by prior work. In the same vein, some of the solutions that have been found thus far have very stringent parametric requirements, making them less impressive than they seem at first glance.

In this thesis, we limit our settings to the 2S2L-V car sharing problem. Because we limit ourselves to this setting, we can cull some notation. First, we simplify the notation $\tau(l, o)$ to simply τ. We can make this simplification because the travel time between the two locations is the same for all requests. Second, we can simplify the notation for request $r = (b_r, t_r, l_r, \bar{l}_r)$ to $r = (b_r, t_r, l_r)$. Because we only work with two locations, $\bar{l}_r$ follows directly from l_r. Third, we can simplify p_r and c_r to p and c, because we only need consider the same two locations, with the same profit and same cost of empty movement for each request. Furthermore, we limit ourselves to only those instances $\mathcal{I}$ in which, for every request $r = (b, t, l) \in \mathcal{I}$ there is another request $r' = (b, t, l) \in \mathcal{I}$. We call request r' the paired request to r. It is not necessary for a pair of request to be accepted in full. If r is accepted, r' is still allowed be rejected, and vice versa.

Finally, we make some small assumptions to simplify our analyses. The first small assumption is that requests booked at the same time are handled in deterministic order. Without loss of generality, we further narrow down this assumption by assuming that r is handled before its paired request r'. The second small assumption is that all servers are at location 0 at time 0. The third and final small assumption is that an empty movement is only performed at the last moment possible. If request $r = (b, t, l)$ requires an empty movement to be served, this empty movement is performed starting only at $t - \tau$.

We finish this chapter by showing a figure conveying a snapshot from a 1S2L timeline. We include this figure to familiarize the reader with this sort of imagery. The image is a time-location diagram, with time on the horizontal axis and location on the vertical axis. A colored line tracks the location of a server over time. This way, we can show the difference between the path taken by an online algorithm, and the path taken by an optimal schedule over that same instance, over some amount of time.

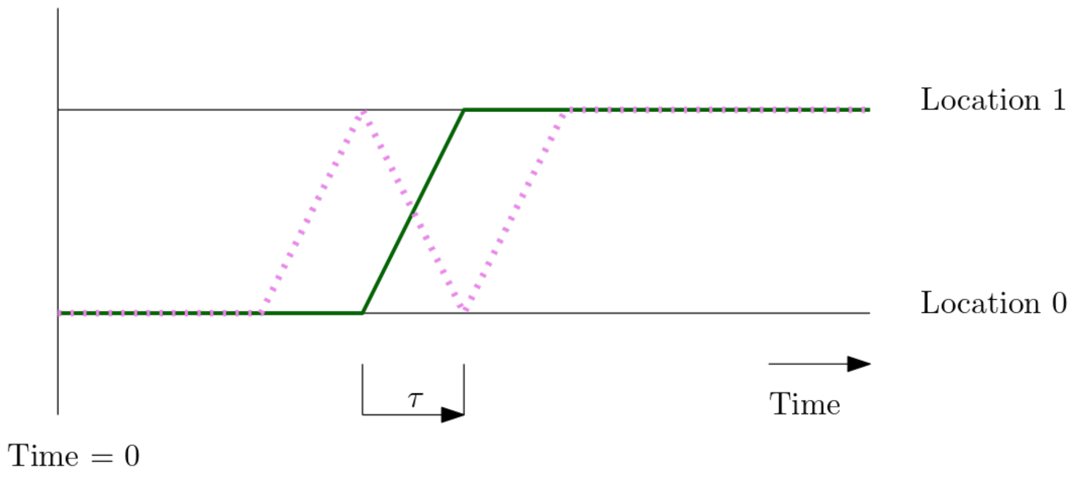


Figure 1: Example of a short 1S2L segment. The location of an online algorithm (in green) and the optimum algorithm (in pink, dotted) are shown over time, as is the length of a trip (τ).

3 Prior Work

This chapter takes a look at work that has been done in the field of not only car-sharing, but also several online problems that seem closely related to the main problem. Learning from the approaches of previous authors gives valuable insight into which strategies might work in advancing the field.

3.1 Car-Sharing

3.1.1 Two Location Networks

1S2L Work on the online car-sharing problem has only started rather recently. As far as we know, K. Luo et al. were the first ones to discuss any variation of the problem in detail in a 2018 paper. Their paper focuses on the 1S2L problem, with both the fixed and variable versions being investigated [12]. Many parameters were tweaked for a plethora of upper bounds and lower bounds. They considered various different lengths booking intervals, booking times and empty movement costs. They start by proving a series of lower bounds via extensive case distinction and induction. After that, they proceed to prove that an intuitively simple greedy algorithm achieves the proven lower bounds. This greedy algorithm accepts any request as long as two simple requirements hold. First, the request must not be in conflict with any already accepted request, and second, serving the request must yield a net profit. Regrettably, some upper bound proofs were omitted due to lack of space.

		$0 \leq c < p$	$c = p$
Problem	**Constraint**	LB (= UB)	LB (= UB)
1S2L-F	$0 \leq \phi < \tau$	1	1
1S2L-F	$\phi \geq \tau$	$\frac{2p}{p-c}$	1
1S2L-V	$0 < \upsilon < \tau$	3	3
1S2L-V	$\upsilon = \tau$	$\max\{\frac{2p}{p-c}, 3\}$	3
1S2L-V	$\upsilon > \tau$	$\frac{3p-c}{p-c}$	$1 + 2\lceil \frac{\upsilon-\lambda}{2\tau} \rceil$

Table 1: The results from [12]

2S2L-F Over the course of the next two years, members from this team would expand the scope of the problem in five further papers. The first of these delves into the 2S2L-F problem [13]. First they prove the lower bounds of various parameter settings of the problem. If the distant pick-up cost is lower than the reward of a request ($0 \leq c < p$), this lower bound is proven to be 2. If the cost of a distant pick-up is, instead, as expensive as the reward ($c = p$), they claim the proof is trivial, and results in a lower bound of 1. They subsequently provide a Smart Greedy algorithm, which achieves both of these lower bounds, which adheres to a few simple rules. If a request does not require a distant pick-up and is acceptable, it is accepted. If the request requires a distant pick-up, however, that request must start later than τ time units after the end of the last request that was served by either of the servers. If a request does not abide by these rules, it is rejected. Although more work has been done on 2-location models, this is the only paper focusing on specifically using two servers. That means the 2S2L-V problem has not yet been thoroughly analyzed.

kS2L Rather than continue their work on 2S2L, this group of authors decided to set their sights on the kS2L problem in their next paper [15]. They claim lower bounds and algorithms for both the variable and fixed booking time variants, but they add a condition simplifying the problem by a great amount: their algorithm and analyses rely on the starting time of requests being limited to multiples of the travel time ($\forall r \in \mathcal{I}\ t_r = \nu\tau\ \nu \in \mathcal{N}$). On top of that, this paper allows unprompted moves, and such movements are done with no cost. Furthermore, the authors assume that the difference between the booking time and starting time of every request i is no greater than the travel time ($b_i \leq t_i - \tau$). No reasoning is given as to why these specific restrictions were put in place, so the results this paper provides are not generally applicable for kS2L. As a further result of the restrictions, the algorithm to achieve the proposed lower bound is significantly different than those that came before it. Regardless of what requests arrive where and when, their Greedy Balanced Algorithm functions mostly the same throughout the release of requests. At the start of the algorithm servers are assigned one of three groups. Two of these groups are *specified* servers that start at one location and then periodically start movements (empty or not) to the location they are not at every τ time units. The final group contains *unspecified* servers that actually take into account where requests (that exceed the amount of specified servers) are posed, and tries to serve those. Though the theoretical competitive ratio of the algorithm is impressive, its narrow specifications make it hard to put it into practice.

Other kS2L findings S. Li et al. provide a less restricted look at the kS2L problem, placing only the restrictions in place in the first paper on the problem [10]. They claim lower bounds for variations in distant pick-up cost and the different booking variations, but leave out a lot of the proofs due to space constraints. For the lower bound proof they do provide, they attempt to distract the online server(s) and then release requests based on the reaction of the online servers. Furthermore, they provide two suboptimal online algorithms for this problem. The first one, Greedy Dispatching, is meant to address instances where distant pick-ups are impossible by simply greedily dispatching servers. A slightly altered version is used for the kS2L-F problem with distant pick-ups. The alteration ensures that at most a certain amount of requests are picked up at either of the two locations, within a time frame of 2τ. The second algorithm, Balanced Dispatching, instead used a smaller time frame of τ and tries to divide which servers are utilised between even and odd frames, as well as between the two locations, evenly.

Problem	**Booking Constraint**	Lower Bound	Upper Bound
kS2L-F, $(0 \leq c < p)$	$0 \leq \phi < \tau$	1	1
kS2L-F, $(c = p)$	$0 \leq \phi < \tau$	1	1
kS2L-F, $(0 \leq c < p)$	$\tau \leq \phi$	$\min\{\frac{k}{\lfloor \frac{2kp}{3p+c} \rfloor}, \frac{2kp}{2kp-(p+c)\lceil \frac{2kp}{3p+c} \rceil}\}$	$\max\{\frac{k}{\lfloor \frac{kp}{2p+c} \rfloor}, \frac{2kp}{kp-\lfloor \frac{kp}{2p+c} \rfloor c}\}$
kS2L-F, $(c = p)$	$\tau \leq \phi$	1	1
kS2L-V, $(0 \leq c < p)$	$0 < \upsilon < \tau$	3	3
kS2L-V, $(c = p)$	$0 < \upsilon < \tau$	3	3
kS2L-V, $(0 \leq c < p)$	$\tau \leq \upsilon$	$\min\{\frac{2k}{\lfloor \frac{2kp}{3p+c} \rfloor}, \frac{k(3p-c)}{kp-c\lceil \frac{2kp}{3p+c} \rceil}\}$	$\frac{k}{\lfloor \frac{k}{4} \rfloor}$ $(k \geq 4)$
kS2L-V, $(c = p)$	$\tau \leq \upsilon$	$\min\{\frac{2k}{\lfloor \frac{k}{2} \rfloor}, \frac{2k}{k-\lceil \frac{k}{2} \rceil}\}$	$\frac{k}{\lfloor \frac{k}{4} \rfloor}$ $(k \geq 4)$

Table 2: The results from [10]

Randomized results A different group further investigated the requirement that servers are only expected to start every τ time units. K. Lai et al. provide a randomized algorithm that assumes decisions on requests can be postponed until multiples of τ. They also look into booking intervals of longer than τ [9]. This way, they can provide a Greedy Balanced algorithm that relies on costless empty moves to facilitate a balanced split between the servers. However, this algorithm only makes moves when it is necessary, as opposed to the Balanced Greedy algorithm by K. Luo et al. Their Adaptive Greedy Balanced algorithm instead balances the amount of servers on each side relative to the proportional amount of requests on each side. The final server is then proportionally randomly decided (if necessary). This is the only result using any random techniques I have encountered thus far.

3.1.2 Star Network

Returning to the papers released by K. Luo et al., we find an investigation of the possibilities of online car-sharing with k servers on a star network [14]. Where this situation would naturally lend itself to empty moves back to the central nodes, the forbiddance of unprompted moves is once again put back into full effect. The restrictions on the starting times are lifted again, which is natural, because different edge lengths are now possible in a star network. The approach of the authors differs depending on the respective lengths of the edges. If the edges are of arbitrary length, they prove that a greedy algorithm will achieve a competitive ratio depending on the ratio between the longest and shortest edge. With edges of uniform length, the length of the notification interval dictates the lower bounds. To achieve these lower bounds as competitive ratio, the servers are split into groups and then run a greedy algorithm. If the notification interval is too short to serve any distant pick-up, the servers are split in as many groups as there are non-center vertices. If distant pick-ups are possible in the notification interval, the servers are instead split by whether they serve requests to or from the central node.

3.1.3 kSmL

K. Luo et al. also look into the most unrestricted basic scope one can imagine in their notation: kSmL-V in general metric space [11]. They work out a series of requests on the line-network to conclude a general lower bound dependent on the ratio between the longest and shortest edge. They do this by releasing requests in such a manner that serving the requests leads to being unable to serve all of the later requests. Not serving these later requests gives the optimal algorithm the chance to serve many requests that an online algorithm cannot. The greedy algorithm is able to reach this bound, making it a theoretically optimal algorithm.

3.1.4 Paired Requests on a Two Location Network

Finally, K. Luo et al. also looked into the consequences of having more than one trip per request in a kS2L model [16]. To gain a profit on a request, all trips requested must be served. Once again separating the problems by notification interval, the authors discover that a notification interval shorter than the travel time leads to a lower bound of infinity. For longer notification intervals, a lower bound of 4 is proven. To achieve that ratio of 4 as an upper bound, a greedy algorithm balancing the amount of servers on both sides is used. Note that this paper once again makes the assumption that the starting time of requests is a multiple of travel time τ.

Many of the lower bounds regarding this problem have been proven using an extensive case distinction. Meanwhile, most upper bounds were achieved using some variation on a greedy algorithm, often balancing the amount of servers at any location at any time. While the greedy approach is not an uncommon one to finding an online algorithm, some nearby problems use different techniques to achieve their competitive ratios. The field of car-sharing is young, and perhaps more complicated variations require inspiration from other problems. Problems similar to online car-sharing are the online travelling salesman problem, the k-server problem, or online dial-a-ride.

3.2 Related problems

3.2.1 Online Travelling Salesperson problem (OLTSP)

The travelling salesperson problem is a classic problem in computing science in which one server has to serve a series of requests in different locations as quickly as possible. In the online variant, however, these requests arrive over time, instead of all at once. Furthermore, the online variant has two subproblems. The first is the *homing* variant, where the server has to return to the origin at the end of the problem, after serving all requests. The second is the *nomadic* variant, in which the server does not need to return to the origin after serving all requests. The two different variations have different competitive ratios and use different algorithms to achieve them [2]. One such an algorithm for the nomadic variant recalculates a new Hamiltonian Path serving all unserved requests when a new request arrives. This theoretically is a 2.5-competitive algorithm, but calculating a Hamiltonian path is known to be NP-Complete, making such an algorithm infeasible in practice. An algorithm that returns to the origin before it starts planning a new Hamiltonian Path is introduced for the homing version. Even though this algorithm is theoretically 2-competitive, it also suffers from the same problem as the previous one. Using a polynomial-time 2-approximation of Hamiltonian path, both algorithms would become 3-competitive.

When compared to the online car-sharing problem, OLTSP is different in quite a multitude of ways. In OLTSP, requests have no start time, nor do they have a destination, or a travel time. Moving around without having any requests to serve is allowed in OLTSP, whereas it is often disallowed in online car-sharing. Nonetheless, the work done on different sorts of networks in OLTSP make it a valuable reference. Different networks require different approaches, with many bounds on specific networks not corresponding to the bound on a general network. An example of this is a cyclic network. In this network, serving requests far away from the origin first is very profitable, because all requests are placed between two extremes [7]. Another different network is a line network. A. Bjelde et al. propose an algorithm for the homing variant that only plans schedules at the origin, and waits with starting these schedules until serving them any later would violate the competitive ratio [4]. This strategy is reminiscent of an algorithm used in the Online Dial-a-Ride Problem.

3.2.2 Online Dial-a-Ride (OLDARP)

As a problem, OLDARP is somewhere in between the online travelling salesperson problem, and car-sharing. A series of requests for transporting objects from a source to a destination are released online. Similar to the online travelling salesman problem, the goal is still optimizing the amount of time it takes to serve all requests. Furthermore, the nomadic and homing variants are mirrored as open and closed variants. Similar to car-sharing, OLDARP can employ multiple servers, and requests consist of both a source and a destination. Well-known algorithms used for OLDARP are similar to the ones used on OLTSP. Three of these algorithms are the REPLAN, IGNORE and SMARTSTART algorithms [1]. The

REPLAN algorithm recalculates the optimum schedule to follow every time a new request is released. The IGNORE algorithm calculates an optimal route for serving all unserved requests while in the origin, then ignores any requests coming in while it is serving that route. After it returns to the origin, it calculates a new route with the requests that came in during its tour. The SMARTSTART algorithm is similar to IGNORE, only planning at the origin. However, SMARTSTART waits on starting a new route until postponing that route any longer would cause a loss in competitive ratio. As an example, imagine a request arriving at time unit 5. If SMARTSTART includes that request in a route of length 4, and wants to achieve a competitive ratio of 2, it waits until time unit 6 to start this route, instead of greedily starting it at time unit 5. This waiting technique of SMARTSTART could be of great interest if applied to variations on the car-sharing problem. In fact, most algorithms for car-sharing already implicitly use a similar strategy when deciding when to perform an empty move.

3.2.3 k-server

Perhaps the most well known online scheduling problem that is close to car-sharing is the k-server problem [8]. In that problem, multiple servers travel a metric space, serving requests at different locations as quickly as possible. The goal of k-server is to minimize the travel distance of the servers when serving all requests. The k-server problem has many open questions still, with the k-server conjecture unproven for over three decades now. One of the more well known algorithms for tackling this problem is the Work Function algorithm. The Work Function algorithm tries to assign servers to requests depending on both the optimum as calculated from the beginning and the cost of changing from the previous formation to the prospective one. Whether an algorithm inspired by the Work Function algorithm will work on the car-sharing problem is unclear.

4 Techniques

This chapter summarizes the strategies applied in the previous work. We look at what strategies were used, when they were used, and how they could be applied to different problems. This list is in no way exhaustive. Instead, it should serve as an indication of which techniques are common and likely to see an application in the thesis.

Most of the techniques described in this chapter are used to perform a *competitive analysis*. Competitive analysis is the standard for measuring the efficiency of any online algorithm. It compares the results an optimal offline algorithm achieves on an instance to the results of online algorithms. This instance is constructed by an all-knowing adversary, most commonly done in such a way that the results of the online algorithms are as poor as can be. The analysis consists of two parts: finding the lower bound and the upper bound. To find a lower bound, the adversary releases a sequence of requests that shows that any online algorithm cannot perform better than $\beta \cdot P_{R_{OPT}}$ (a multiplicative factor times the profit of an optimal algorithm), whatever decision it might make. This idea could also be applied to a single algorithm, focusing on their worst case performance instead of the worst case performance of any algorithm. Looking into an upper bound compares the performance of one specific online algorithm with the optimal performance. The adversary attempts to create a sequence that uses the properties of the algorithm against itself. Both bounds are of interest, but competitive analysis does not specify how to establish these bounds besides comparing the results of both algorithms. Different techniques are used for that.

4.1 Greedy Strategy

The crux of most algorithms devised thus far on online car-sharing is the greedy strategy. That is, accepting any request as long as it can be accepted without being in conflict with other already accepted requests. Slight variations often apply to avoid some particularly bad situations, but once some restrictions are put in place, the computations are often incredibly simplistic. This makes way to a conjecture that some variation of a greedy algorithm achieves the optimal competitive ratio on every standard instance of the online car-sharing problem. Whether this turns out to be true is to be seen, and probably outside the scope of this thesis. However, when designing a new algorithm for a new variation, a greedy algorithm might be an excellent starting point.

4.2 Wait to Start

Most algorithms on online car-sharing wait until the last moment to move a server to serve a distant request. They wait to see if there is a request to serve so that they can forego the empty move that would otherwise be induced. This is different from the ideas applied in algorithms like SMARTSTART, where more requests are gathered before any server leaves its starting location. If immediate decisions are part of the problem, waiting is usually a bad plan. But if we consider the proposed delayed booking instance (where requests can be postponed for a certain amount of time, perhaps with a penalty), looking into algorithms similar to SMARTSTART might yield improvements to the competitive ratio.

4.3 Balancing

The more locations and servers there are, the more servers can be tricked into a suboptimal decision. That could be serving a request with cost, or serving a request to an outpost of the network. Such a server might end up where it would be impossible to serve any other request without a preemptive move, which is often disallowed. One way to avoid mistakes such as these is to assign servers to groups that serve requests to or from certain locations. Balancing the server load evenly across the locations might be a way to reduce the competitive ratio of any algorithm with more servers than locations.

4.4 Case Distinction for Finding Lower Bounds

One such technique is case distinction. Most of the literature on car-sharing uses this to obtain the lower bounds, especially. The adversary places one or multiple requests, and every reaction possible by an algorithm is separated into different cases. Depending on what decision the algorithm made, the adversary might release new requests that this algorithm cannot serve, until the sequence that an optimum could serve is as good as it can be compared to any algorithm. After that the bounds found on the different cases are compared with each other, with the best case serving as a lower bound for the problem.

4.5 Potential Functions

Another technique which has not yet been truly utilized on online car-sharing is the use of a potential function. A potential function is used to balance out certain costs that are made throughout an algorithm with other decisions made later or earlier on. If the sum of the potential function and the cost function do not exceed the bound sought after, we can conclude that, in a general case, the bound holds. It can also be used to measure the difference between the optimal profit and the aggregate profit. Whether the analysis of a potential function will be necessary will remain to be seen.

5 2S2L-V, Lower Bounds

In this chapter, we discuss our findings on the lower bounds of the 2S2L-V problem. This problem remained untouched in the work by K. Luo et al. [12] on the two server two location model. This contrasts their work on 1S2L, which included findings on both the Fixed and Variable variants of that problem. Recall that we limit our instance to only releasing pairs of requests.

Throughout this chapter, we vary two parameters, the empty movement cost c and the range of notification intervals, denoted by $[\lambda, \upsilon]$. The setting of the first of these parameters, the empty movement cost c, is denoted relative to p, the reward for serving a single request. The settings we consider for this parameter are $c = p$ and $0 < c < p$. Note that the $0 < c < p$ setting of these is actually a range of settings. Every setting in that range follows the same analysis. Despite these $0 < c < p$ settings sharing the same analysis, specific results of this analysis vary depending on the specific setting c takes. This is because some approaches give a performance ratio that is a function on the value of c. Such a function might lead to the lowest performance ratio on some values of c, whereas a different approach might lead to a lower performance ratio on different values of c. In the scenario that the lowest performance ratio follows a function for some, but not all values of c, we will explicitly mention the point where the results cross over.

The other parameter, the range of notification intervals $[\lambda, \upsilon]$, also has multiple settings that need to be looked into. A λ setting of $\lambda < \tau$ gives an adversary a boost in strength. This boost in strength is caused by the possibility to release requests that are impossible to serve because of server positioning. To create the most potent adversaries, we assume $\lambda < \tau$ on the bounds we discuss in this chapter. On the other hand, an υ setting of $\upsilon < \tau$ weakens adversaries. This weakening is caused because it is harder to release serviceable distant requests. With $\upsilon < \tau$, it is impossible for an adversary to release requests that can be served by all servers, regardless of server positioning. Such distant requests often cause hard, costly decisions, that an adversary can take great advantage of, in order to achieve higher lower bounds. Finally, we consider the setting of the notification interval length $\upsilon - \lambda$. It takes τ time to serve a single request. The setting of $\upsilon - \lambda$ thus indicates how many requests can be planned ahead of time. In some settings, this limits the strength of the adversary. We consider the following settings for $[\upsilon, \lambda]$: $0 < \upsilon < \tau$, $\upsilon = \tau$, $\tau < \upsilon - \lambda \leq 2\tau$, $2\tau < \upsilon - \lambda \leq 3\tau$, $3\tau < \upsilon - \lambda \leq 5\tau$, $5\tau < \upsilon - \lambda \leq 7\tau$, $7\tau < \upsilon - \lambda \leq 9\tau$, $9\tau < \upsilon - \lambda$.

Many of these proofs in the following sections follow a similar structure, reacting to the response of the algorithms in the face of request arrivals. We use case distinction to differentiate between different reactions. This showcases the best reaction of OPT when provided a reaction by ALG. Note that the theorems that follow in this chapter might build upon theorems proven before them. A case that was discussed in one theorem is not reiterated in a following theorem. For each new theorem, we note which theorems handle cases that are relevant to the new theorem. Using this process, we start out simple, steadily increasing the complexity by steadily giving an adversary more freedom.

Finally, we introduce the following terminology for this chapter. We use ALG to refer to any online algorithm. Meanwhile, OPT is used to refer to the optimal scheduler. We refer to the servers as moved by ALG as s'_1 and s'_2, and to the servers as controlled by OPT as s^*_1 and s^*_2. We now show the lower bounds we discovered in this chapter:

	$0 \leq c < p$	Theorem
Constraint	LB	
$0 < \upsilon < \tau$	3	Theorem 5.4
$\upsilon = \tau$	$\max\{\frac{4p}{2p-c}, 3\}$	Theorem 5.5
$\tau < \upsilon - \lambda \leq 2\tau$	$\min\{\frac{3p-c}{p-c}, 4\}$	Theorem 5.6
$2\tau < \upsilon - \lambda \leq 3\tau$	$\min\{\frac{3p-c}{p-c}, 4\}$	Theorem 5.7
$3\tau < \upsilon - \lambda \leq 5\tau$	$\max\{\min\{\frac{3p-c}{p-c}, 4\}, \frac{10p}{4p-2c}\}$	Theorem 5.8
$5\tau < \upsilon - \lambda \leq 7\tau$	$\max\{\min\{\frac{3p-c}{p-c}, 4\}, \frac{12p-2c}{4p-2c}\}$	Theorem 5.9
$7\tau < \upsilon - \lambda \leq 9\tau$	$\max\{\min\{\frac{3p-c}{p-c}, 4\}, \min\{\frac{16p-2c}{6p-4c}, \frac{12p-2c}{3p-c}\}\}$	Theorem 5.10
$8\tau < \upsilon - \lambda$	$\max\{\min\{\frac{3p-c}{p-c}, 4\}, \min\{\frac{6p+4p(\lceil\frac{\upsilon-\lambda-4\tau}{4\tau}\rceil)+(2p-2c)(\lceil\frac{\upsilon-\lambda-6\tau}{4\tau}\rceil)}{2p+(2p-2c)(\lceil\frac{\upsilon-\lambda-4\tau}{4\tau}\rceil)}, \frac{12p-2c}{3p-c}\}\}$	Theorem 5.11

Table 3: The results of the lower bounds on the low-cost variant ($0 < c < p$) of 2S2L-V

	$c = p$	Theorem
Constraint	LB	
$0 < \upsilon - \lambda \leq 2\tau$	3	Theorem 5.4
$2\tau < \upsilon - \lambda \leq 3\tau$	4	Theorem 5.7
$3\tau < \upsilon - \lambda$	5	Theorem 5.8

Table 4: The results of the lower bounds on the high-cost variant ($c = p$) of 2S2L-V

5.1 Gadgets

Before we show the proofs themselves, we first introduce some *gadgets* that are commonplace in these proofs. Very often, an adversary can react to a request accepted by ALG in a set way. Instead of repeating ourselves using a similar strategy, we give these reactions names. We call these predetermined reactions *gadgets*, and we discuss them in this section, before we get into the proofs themselves.

First, we introduce the *Zig gadget.* A Zig gadget is a series of four requests, two overlapping pairs to be precise, starting at opposing sides. Specifically, *a Zig gadget starting at time a* is defined as four requests: $r_1 = r_2 = (..., a, l_{r_1})$ and $r_3 = r_4 = (..., a + \tau, \bar{l}_{r_1})$. Note that these requests can be served by the same pair of servers in quick succession. Also note that the second pair of requests does not require an empty movement to serve. Therefore, that second pair of requests will always grant a profit of $2p$. Furthermore, note that we do not specify the booking time of these requests.

Lemma 5.1 All requests in the Zig gadget, starting at time a, conflict with any request r_d for which $a - \tau < t_{r_d} < a + \tau$ and $l_{r_3} = l_{r_d}$.

Proof To prove this Lemma, we show that request r_1 and r_3 are in conflict with r_d. Since $r_1 = r_2$ and $r_3 = r_4$, if r_1 and r_3 are in conflict with r_d it follows that all four requests comprising the Zig gadget are in conflict with r_d. Therefore, if we prove that r_1 and r_3 are in conflict with r_d, we prove the entire lemma.

We first show that r_1 is in conflict with r_d. Because of the premise that $a - \tau < t_{r_d} < a + \tau$, we know that r_d is certainly in conflict with r_1 and thus r_2. Recall the definition of conflicting requests. Requests r and j are in conflict iff, for $t_r \leq t_j$, $t_j < \bar{t}_r + \tau(\bar{l}_r, l_j)$. Since r_1 ends where r_d starts ($l_{r_d} = l_{r_3} = \bar{l}_{r_1}$), we find that the travel time between the requests is 0 ($\tau(\bar{l}_{r_1}, l_{r_d}) = 0$). Thus, we only need to show that $t_{r_d} \leq t_{r_1} < \bar{t}_{r_d} = t_{r_d} + \tau$ or $t_{r_1} \leq t_{r_d} < t_{r_1} + \tau$ holds for any value of t_{r_d} allowed under the premise. Since $t_{r_1} = a$, we find $t_{r_d} \leq a < t_{r_d} + \tau$ and $a \leq t_{r_d} < a + \tau$. This is the case for $a - \tau < t_{r_d} \leq a$ and $a \leq t_{r_d} < a + \tau$, respectively. That covers all values t_{r_d} can take by the premise of the Lemma, proving that l_{r_1}, and thus l_{r_2} is in conflict with l_{r_d}.

Now, we show that r_3 is in conflict with r_d. This time, r_3 starts where r_d starts ($l_{r_3} = l_{r_d}$), we find that the travel time between these requests is τ ($\tau(\bar{l}_{r_3}, l_{r_d}) = \tau$). We also know, since $t_{r_3} = a + \tau > t_{r_d}$, that we need only check the conflict formula once. Following the formula to check whether requests are in conflict, we find that r_3 and r_d are in conflict iff $t_{r_3} < \bar{t}_{r_d} + \tau$. We note that $t_{r_3} = a + \tau$. Thus, we find $t_{r_d} \leq a + \tau < t_{r_d} + 2\tau$, which holds, because, by the premise of the Lemma, $a - \tau < t_{r_d} < a + \tau$. Thus, all requests in the Zig gadget are in conflict with r_d □

Next, we define the *Zig-Zag gadget.* The Zig-Zag gadget is a series of six requests. These six requests form a series of three overlapping pairs of requests, starting at alternating sides. Specifically, *a Zig-Zag gadget starting at time a* is defined as six requests: $r_1 = r_2 = (..., a, l_{r_1})$, $r_3 = r_4 = (..., a + \tau, \bar{l}_{r_1})$ and $r_5 = r_6 = (..., a+2\tau, l_{r_1})$. Once again, the booking times of these servers are not defined and the requests can be served by the same pair of servers in quick succession. The second and third of these pairs do not require an empty move to serve, so those four requests always grant $4p$ profit.

Lemma 5.2 All requests in the Zig-Zag gadget, starting at time a, conflict with any request r_d for which $a < t_{r_d} < a + 2\tau$ and $l_{r_1} = l_{r_d}$.

Proof To prove this lemma, we need to show that requests r_1, r_3 and r_5 of the Zig-Zag gadget starting at time a are in conflict with r_d. Since r_2, r_4 and r_6 are identical to r_1, r_3 and r_5, respectively, their conflict with r_d follows.

First, we show that request r_1 is in conflict with r_d. Recall the definition of conflicting requests.

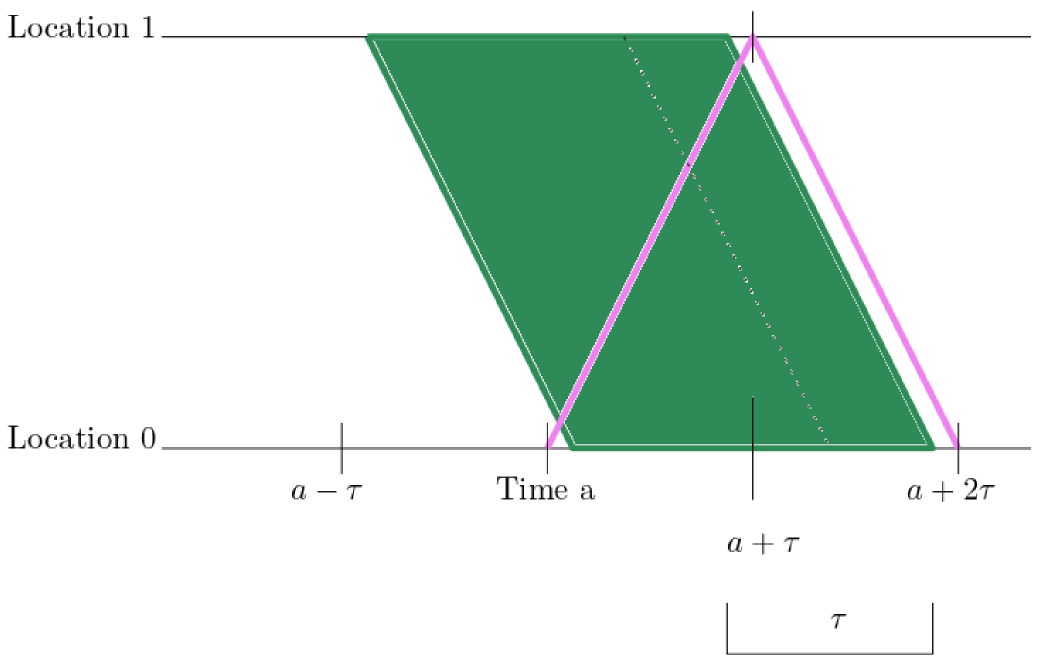


Figure 2: Lemma 5.1, colourized. Any request that fully takes place inside the green zone is in conflict with the pink Zig gadget. Note that all requests have the same gradient because of the consistent travel time τ

Requests r and j are in conflict iff, with $t_r \leq t_j$, $t_j < \bar{t}_r + \tau(\bar{l}_r, l_j)$. Per the premise of this Lemma, note that $t_{r_1} = a < t_{r_d}$. Also note that $l_{r_1} = l_{r_d}$. This means $\tau(\bar{l}_{r_1}, l_{r_d}) = \tau$. Therefore, we need to check that $t_{r_d} < a + 2\tau$. Per the premise of this Lemma, this holds. Therefore, $r_1 = r_2$ are in conflict with r_d.

Next, we show that request r_3 is in conflict with r_d. r_3 differs from r_1 in that $t_{r_3} = a+\tau$, and $l_{r_3} = \bar{l}_{r_1}$. This has an impact on two facts of the formula for conflicting requests. For one, neither $t_{r_3} \leq t_{r_d}$ nor $t_{r_d} \leq t_{r_3}$ is guaranteed. Therefore, we need to check both formulas. Furthermore, $\tau(\bar{l}_{r_3}, l_{r_d}) = 0$. Filling in the formulas, we find that either $t_{r_3} < t_{r_d} + \tau$, or $t_{r_d} < t_{r_3} + \tau$. Since $t_{r_3} = a + \tau$ we find that $a + \tau < t_{r_d} + \tau$ or $t_{r_d} < a + 2\tau$. Per the premise of the lemma, these two formulas hold. Therefore, $r_3 = r_4$ are in conflict with r_d.

Finally, we show that request r_5 is in conflict with r_d. r_5 effectively transposes r_1 by 2τ. This means $t_{r_5} = a + 2\tau$ Since $a < t_{r_d} < a + 2\tau$ we note that, $t_{r_d} < t_{r_5}$. This effectively inverses the roles of r_d and the request in the gadget. Note that $l_{r_5} = l_{r_d}$. This means $\tau(\bar{l}_{r_d}, l_{r_5}) = \tau$. Therefore, we need to check that $t_{r_5} < t_{r_d} + 2\tau$. Rephrasing t_{r_5} as $a + 2\tau$, we find $a + 2\tau < t_{r_d} + 2\tau$. Since $a < t_{r_d}$, per the premise of this Lemma, this holds. Therefore, $r_5 = r_6$ are in conflict with r_d. $\square$

Note that removing the final pair of requests from a Zig-Zag gadget produces a Zig gadget. This means, for every Zig-Zag gadget with all requests conflicting with one request, there must exist a Zig gadget with all requests conflicting with that same request. This produces the following corollary.

Corollary 5.3 All requests in the Zig gadget, starting at time a, conflict with any request r_d for which $a < t_{r_d} < a + 2\tau$ and $l_{r_1} = l_{r_d}$.

5.2 No distant Requests

The first sets of settings for c and $[\lambda, \upsilon]$ that we prove are those parameters that do not see any benefit from including distant requests in the cases. This is caused by the setting for υ, though the range for these settings is increased if $c = p$.

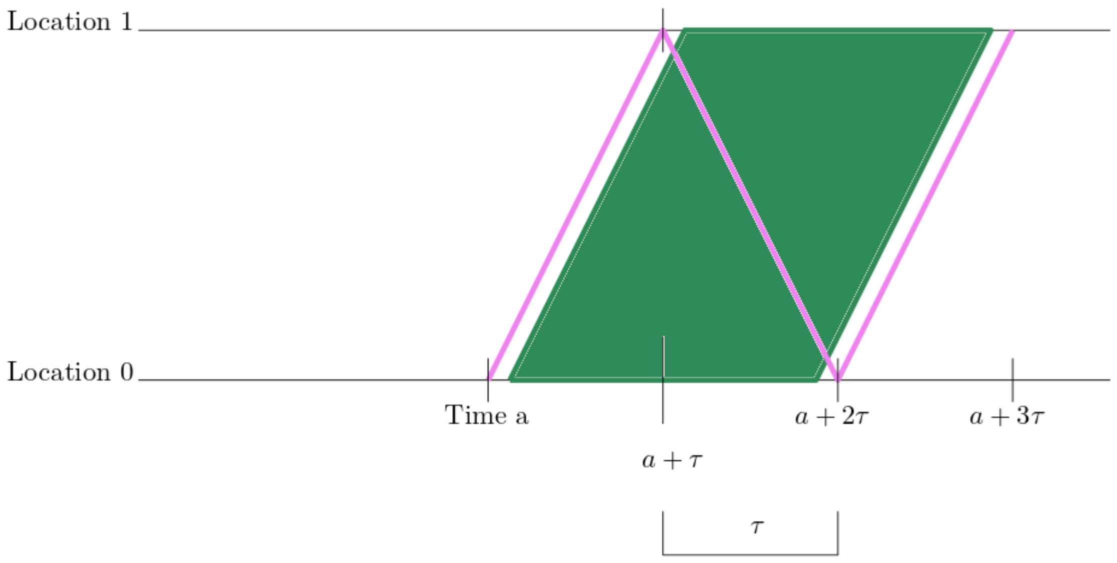


Figure 3: Lemma 5.2, colourized. Any request that fully takes place inside the green zone is in conflict with the pink Zig-Zag gadget. Note that all requests have the same gradient because of the consistent travel time τ

Theorem 5.4 For $0 < \upsilon < \tau$, or $\upsilon - \lambda \leq 2\tau$ and $c = p$, no deterministic online algorithm can achieve a competitive ratio of lower than 3.

These parameters are grouped together because they share the property that any distant request r_d is never worth considering. In the former situation, ($0 < \upsilon < \tau$) this is the case because it takes τ time to perform the movement to get to the starting situation of r_d, yet $b_{r_d} - t_{r_d} < \tau$. That means there is no time to perform the empty movement to pick up r_d. In the latter situation ($\upsilon - \lambda \leq 2\tau$ and $c = p$) there is time to perform the empty movement. However, the reward of serving r_d is completely mitigated by the cost of the empty movement. Therefore, serving r_d only distracts at least one of ALG's servers for no benefit. Because $\upsilon - \lambda \leq 2\tau$, the time in which OPT can make any profit over ALG is very limited. Therefore, using at least one server to serve a request that is not profitable is detrimental to the performance ratio of any ALG in this situation. Having discussed the caveats for serving distant requests in both situations laid out by the proof, we conclude that distant requests are not of interest for the settings of this particular proof.

Proof Initially, the adversary releases $r_1 = r_2 = (0, \upsilon, 0)$. We distinguish three cases.
Case 1: ALG rejects both r_1 and r_2. OPT subsequently accepts r_1 and r_2. We find $P_{R^A} = 0$ and $P_{R^*} = 2p$, hence $P_{R^*}/P_{R^A} = \infty$. (See also figure 4a).
Case 2: ALG accepts r_1 or r_2, but not both. The adversary subsequently releases $r_3 = r_4 = (\epsilon, \upsilon - \epsilon, 0)$. Note that these requests (r_3, r_4) are in conflict with the accepted request. Thus at most one of r_3 or r_4 can be accepted. ALG is faced with two options this time.
Case 2a: ALG rejects both r_3 and r_4. OPT then accepts both of these, and the adversary releases $r_5 = r_6 = (\tau + \epsilon, \upsilon + \tau - \epsilon, 1)$, which are not profitably acceptable by ALG, but are acceptable to OPT. Regardless of whether ALG accepts any of r_5 or r_6, we find $P_{R^A} = p$ and $P_{R^*} = 4p$, hence $P_{R^*}/P_{R^A} = 4$. (See also figure 4b).
Case 2b: ALG accepts r_3 or r_4. The adversary then releases a Zig-Zag gadget starting at location 0 at time $\upsilon - 2\epsilon$. Per Lemma 5.2, all of these requests are in conflicted with the requests ALG accepted, yet OPT is free to accept them all. We find $P_{R^A} = 2p$ and $P_{R^*} = 6p$, hence $P_{R^*}/P_{R^A} = 3$. (See also figure 4c).
Case 3: ALG accepts both r_1 and r_2. The adversary then releases a Zig-Zag gadget starting at location 0 at time $\upsilon - \epsilon$. We know ALG's servers cannot serve this Zig-Zag gadget (Lemma 5.2), whereas OPT can. OPT subsequently accepts all requests in the gadget. We find $P_{R^A} = 2p$ and $P_{R^*} = 6p$, hence

$P_{R^*}/P_{R^A} = 3$. (See also figure 4d). □

5.3 Short Notification Intervals

In this section, we discuss the settings with a notification interval shorter than 3τ. These settings are substantially different from the settings discussed in the previous sections, because ALG and OPT can benefit from accepting distant requests with these settings. This access to distant requests strengthens the adversary. We still separate the cases with these notification intervals from the cases with notification intervals longer than 3τ, because the adversary gains an important extra tool with notification intervals longer than 3τ. This tool will be explained in the section pertaining Long Notification Intervals.

Theorem 5.5 For $\upsilon = \tau$ and $0 < c \leq p$, no deterministic online algorithm can achieve a competitive ratio of lower than $\max\{\frac{4p}{2p-c}, 3\}$.

We group these parameter settings together mainly because of the $\upsilon = \tau$. This setting ensures that distant requests are possible, whilst making it impossible for a distant request to line-skip an accepted distant request. Because distant requests are possible, and profitable, they need to be considered in the following proof.

Next, please note that the crossover point between these two bounds can be calculated beforehand. These lower bounds are equal when the cost for an empty movement $c = \frac{2p}{3}$. Following this, we conclude that the bound of $\frac{4p}{2p-c}$ is higher when the cost is higher than $\frac{2p}{3}$, whereas the bound of 3 is higher when the cost is smaller than $\frac{2p}{3}$. Since an adversary decides the instance showing the lower bound, it can release the instance resulting in the higher lower bound for the chosen setting of c. This explains the max function present when describing the lower bound of this setting.

The bound of 3 can be proven by using the same steps taken in **Theorem 5.4**. We do not repeat the analysis of the cases used in the proof of that theorem. Instead, we need to investigate the impact of distant requests, since the current settings make distant requests possible. To investigate what happens when the first request is a distant request, we will distinguish three main cases.

Proof Initially, the adversary releases $r_1 = r_2 = (0, \tau, 1)$. We distinguish three cases.
Case 1: ALG accepts neither r_1 nor r_2. OPT then accepts both of the requests. We find $P_{R^A} = 0$ and $P_{R^*} = 2p$, hence $P_{R^*}/P_{R^A} = \infty$. (See also figure 5a).
Case 2: ALG accepts either r_1 or r_2. Following this, the adversary releases two more requests: $r_3 = r_4 = (\epsilon, \upsilon + \epsilon, 0)$. ALG is faced with two options this time.
Case 2a: ALG accepts neither r_3 nor r_4. OPT subsequently accepts both of them. We find $P_{R^A} = p - c$ and $P_{R^*} = 2p$, hence $P_{R^*}/P_{R^A} = \frac{2p}{p-c}$. (See also figure 5b). For completeness' sake, note that it is possible for an adversary to release two more requests from 1 to 0, such that OPT can serve them, but ALG cannot. This would make the performance ratio of any algorithm entering this case even higher than the one shown. Since we explore this setting for the lower bound, we end this case here.
Case 2b: ALG accepts r_3 or r_4. The adversary then releases a Zig gadget starting at location 0 at time $\lambda + 2\epsilon$. By Lemma 5.1 and 5.2, these requests are in conflict with ALG's accepted requests. OPT accepts all four. We find $P_{R^A} = 2p - c$ and $P_{R^*} = 4p$, hence $P_{R^*}/P_{R^A} = \frac{4p}{2p-c}$. (See also figure 5c).
Case 3: ALG accepts both r_1 and r_2. The adversary then releases a Zig gadget starting at location 0 and time $\lambda + \epsilon$. By Lemma 5.2, we know all of these requests fully conflict with the requests ALG accepted, but are fully acceptable to OPT. Indeed, OPT accepts all of the requests in the Zig-gadget. We find $P_{R^A} = 2p - 2c$ and $P_{R^*} = 4p$, hence $P_{R^*}/P_{R^A} = \frac{2p}{p-c}$. (See also figure 5d). □

Theorem 5.6 For $\tau < \upsilon - \lambda \leq 2\tau$ and $0 < c < p$, no deterministic online algorithm can achieve a competitive ratio of lower than $\min\{\frac{3p-c}{p-c}, 4\}$

We note that the bound we prove is higher than the one found in Theorem 5.4 and Theorem 5.5. While the cases covered in those theorems are still possible, they are no longer the cases leading to the highest lower bound. This change comes about because line-skipping distant requests are possible when $\tau < \upsilon - \lambda$. (See, for example, Figure 6a). This extra freedom gives the adversary opportunities to create a higher lower bound. The crossover point between $\frac{3p-c}{p-c}$ and 4 is $c = \frac{p}{3}$. For $c < \frac{p}{3}$, the lower bound is $\frac{3p-c}{p-c}$, whereas for $c > \frac{p}{3}$ the lower bound is 4.

A big difference between the lower bound found in Theorem 5.5, and the lower bound discussed in this theorem, is that the resulting lower bound from Theorem 5.5 includes a max function, whereas the

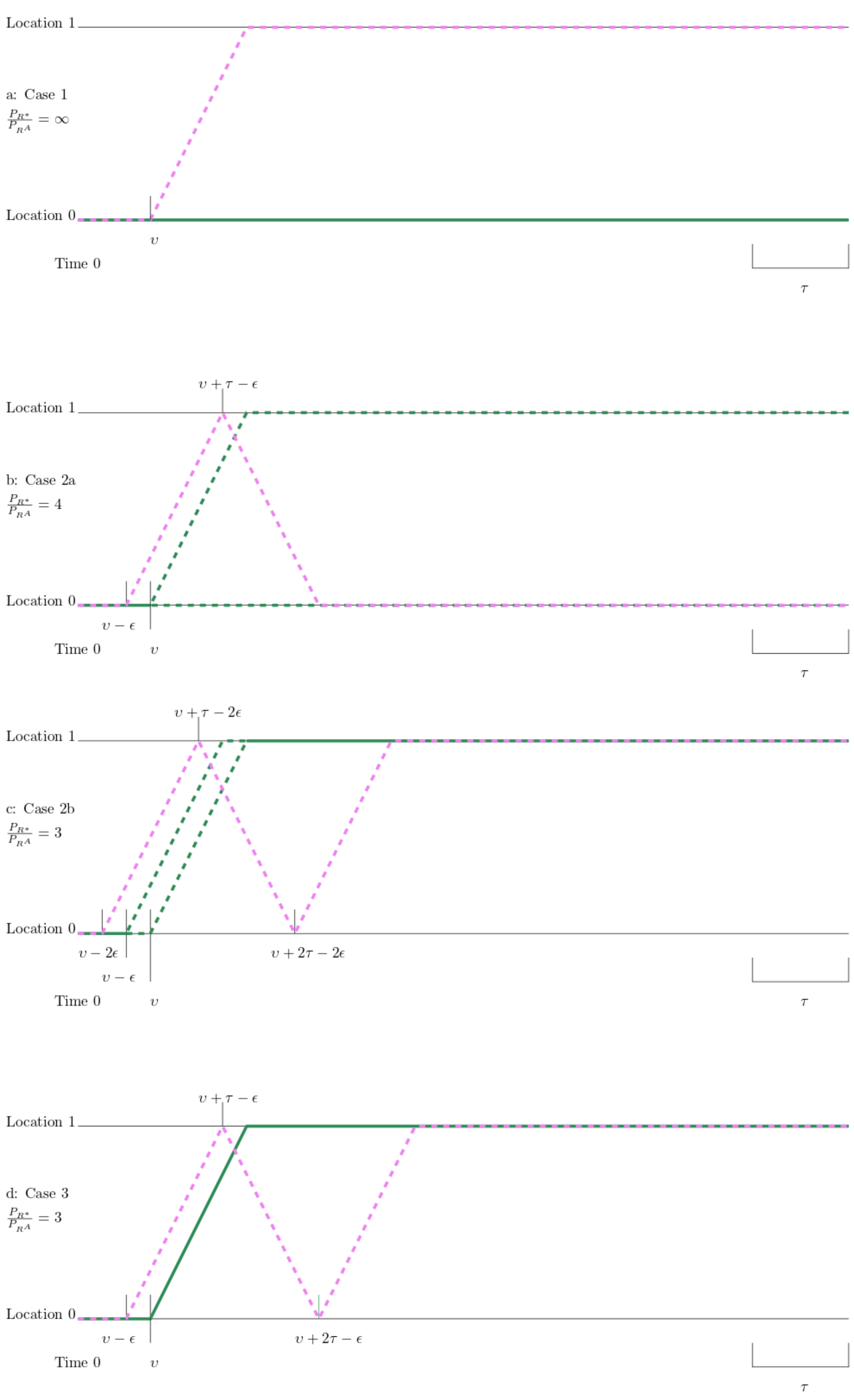


Figure 4: All cases of Theorem 5.4, illustrated with appropriate competitive ratio. The location of the online algorithm (in green) and the optimum algorithm (in pink, dashed) are shown over time, as is the length of a trip (τ). If the green line is dashed, that means the two servers are not at the same location at that time.

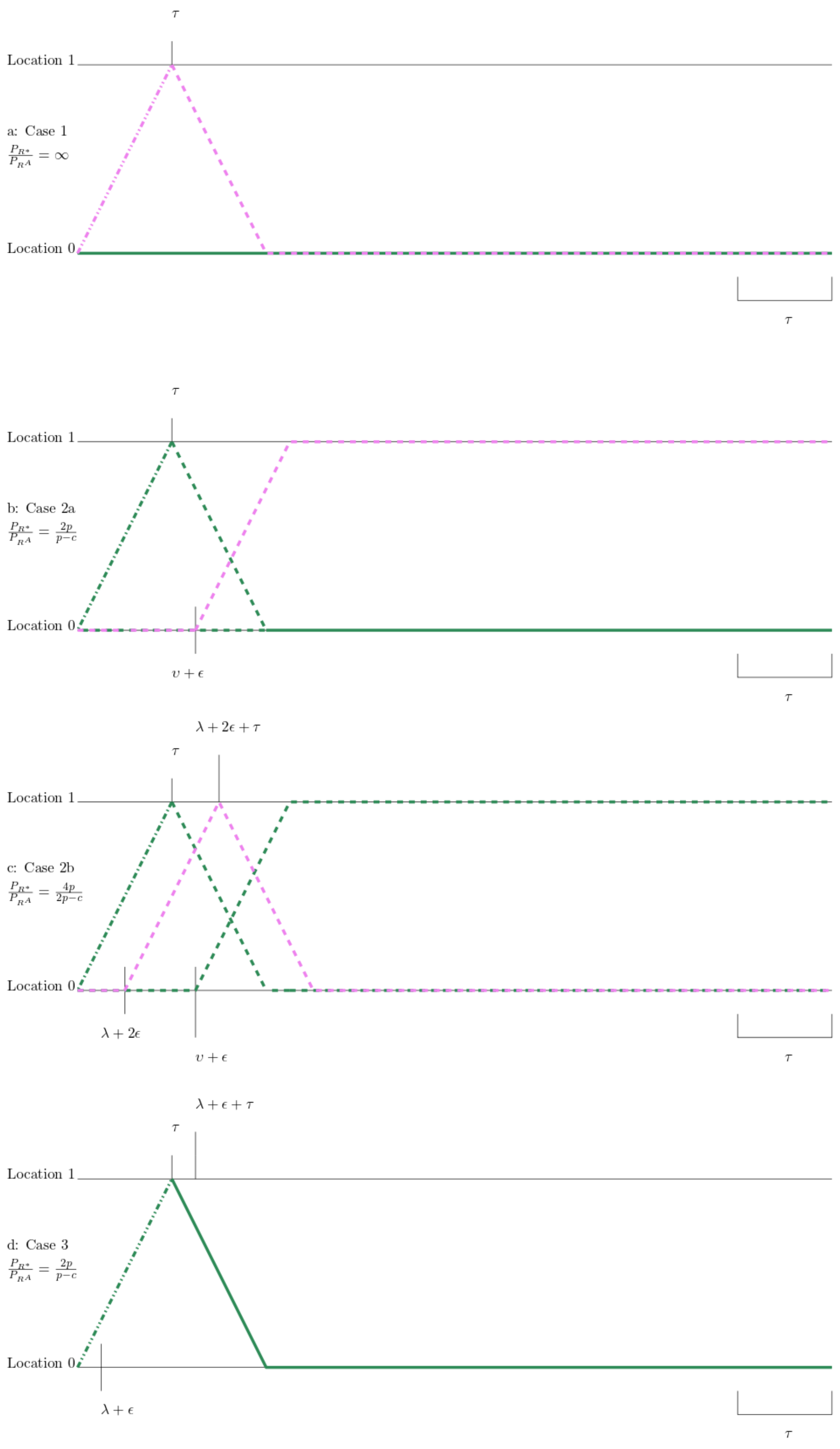


Figure 5: All cases of Theorem 5.5, illustrated with appropriate competitive ratio. The location of the online algorithm (in green) and the optimum algorithm (in pink, dotted) are shown over time, as is the length of a trip (τ). If the green line is dashed, that means the two servers are not at the same location at that time. If a line is both dotted and dashed, it indicates an empty movement performed by a number of servers equal to the amount of dots.

lower bound in this theorem includes a min function. This change comes about because the reason why these functions are included are very different. In Theorem 5.5, the adversary had to fully change its released instance to show the lower bound. In this theorem, however, it is the behaviour of ALG that should change when $c = \frac{p}{3}$ is reached. Since the instance is the same for both parts of the function, the decision which path to take is solely on ALG, explaining the min function in this lower bound.

Note that releasing a non-distant request as first request will not lead to the highest possible lower bound. There is no way to release a request turning the non-distant request into a distant request with the settings for λ and υ used in this theorem. We can use the adversary laid out in Theorem 5.4 to show that starting the releases with a non-distant request cannot achieve a lower bound higher than 3. That is lower than the bounds we prove in this theorem.

Proof Initially, the adversary releases $r_1 = r_2 = (0, \upsilon, 1)$. We distinguish three cases.
Case 1: ALG accepts neither r_1 nor r_2. OPT then accepts both r_1 and r_2. We find $P_{R^A} = 0$ and $P_{R^*} = 2p - 2c$, hence $P_{R^*}/P_{R^A} = \infty$. (See also Figure 6a).
Case 2: ALG accepts either r_1 or r_2. The adversary then releases two more requests $r_{s_1} = r_{s_2} = (\epsilon, \upsilon - 2\epsilon, 1)$. ALG is faced with two options after that.
Case 2a: ALG rejects both r_{s_1} and r_{s_2}. OPT accepts both of these requests, and the adversary releases two more requests $r_{s_3} = r_{s_4} = (2\tau, \upsilon + 2\tau - 2\epsilon, 1)$. These requests are still in conflict with r_1 and r_2. Hence at most one of them can be accepted by ALG, which is faced with two choices.
Case 2a-a: ALG accepts neither r_{s_3} nor r_{s_4}. OPT accepts both of those requests. Here, we can end the release of requests. We find $P_{R^A} = p$ and $P_{R^*} = 4p$, hence $P_{R^*}/P_{R^A} = 4$. (See also Figure 6b).
Case 2a-b: ALG accepts r_{s_3} or r_{s_4}. The adversary then releases a Zig gadget starting at location 0 and time $\upsilon + \tau - \epsilon$. Both of the requests accepted by ALG are in conflict with this gadget (Lemma 5.1). Therefore, these requests can only be accepted by OPT. We find $P_{R^A} = 2p - 2c$ and $P_{R^*} = 6p - 2c$, hence $P_{R^*}/P_{R^A} = \frac{3p-c}{p-c}$. (See also Figure 6c).
Case 2b: ALG accepts r_{s_1} or r_{s_2}. The adversary then releases the same Zig-Zag gadget as described in Case 1: starting at location 1 at time $\upsilon - 3\epsilon$. Per Lemma 5.2, These requests are in conflict with the requests ALG accepted, but OPT accepts all of them. We find $P_{R^A} = 2p - 2c$ and $P_{R^*} = 6p - 2c$, hence $P_{R^*}/P_{R^A} = \frac{3p-c}{p-c}$. (See also Figure 6d).
Case 3: ALG accepts both r_1 and r_2. The adversary then releases a Zig-Zag gadget starting at location 1 and time $\upsilon - 3\epsilon$. These requests are in conflict with the requests that have been accepted by ALG (Lemma 5.2), and OPT accepts all of them. Note that the first two requests of this Zig-Zag gadget are distant requests, as are both requests ALG accepted. We find $P_{R^A} = 2p - 2c$ and $P_{R^*} = 6p - 2c$, hence $P_{R^*}/P_{R^A} = \frac{3p-c}{p-c}$. (See also Figure 6e). □

Theorem 5.7 For $2\tau < \upsilon - \lambda \leq 3\tau$ and $0 < c \leq p$, no deterministic online algorithm can achieve a competitive ratio of lower than $\min\{\frac{3p-c}{p-c}, 4\}$

We note that the bound is the exact same as in Theorem 5.6. This is because releasing a distant request first still leads to this lower bound. To show this, we expand our case distinction with a new beginning, where the first request released is a non-distant request with $b_{r_1} - t_{r_1}$ between 2τ and 3τ. We show that the adversary has no reaction improving upon the $\min\{\frac{3p-c}{p-c}, 4\}$ bound. We do not reiterate the case distinction of releasing a distant request first in this theorem. For that, we refer to **Theorem 5.6**.

In this preamble, we give special attention to the setting that $c = p$. This setting differs from $0 < c < p$, in that starting the timeline with a distant request does not lead to the lower bound of the problem. Instead, the lower bound for that setting (4) is supported by the findings in this proof. Filling in $c = p$ in all of the ratios we find, we find that case $2a$ still leads to a lower bound of 4 in this setting, even if distant requests do not give any profit.

Proof Initially, the adversary releases $r_1 = r_2 = (0, \upsilon, 0)$. We distinguish three cases.
Case 1: ALG accepts neither r_1 nor r_2. OPT then accepts both r_1 and r_2. We find $P_{R^A} = 0$ and $P_{R^*} = 2p$, hence $P_{R^*}/P_{R^A} = \infty$. (See also Figure 7a).
Case 2: ALG accepts either r_1 or r_2. OPT rejects both r_1 and r_2. The adversary then releases $r_3 = r_4 = (\epsilon, \upsilon - \epsilon, 0)$. We distinguish two cases.
Case 2a: ALG accepts neither r_3 nor r_4. OPT then accepts both r_3 and r_4. The adversary then waits until it releases $r_5 = r_6 = (\upsilon - \lambda, \upsilon + \tau - \epsilon, 1)$. These requests conflict with r_1, so s'_1 cannot serve it. Furthermore, these requests are distant to s'_2. Assuming $\lambda < \tau$, they are in conflict with s'_2 as well. OPT

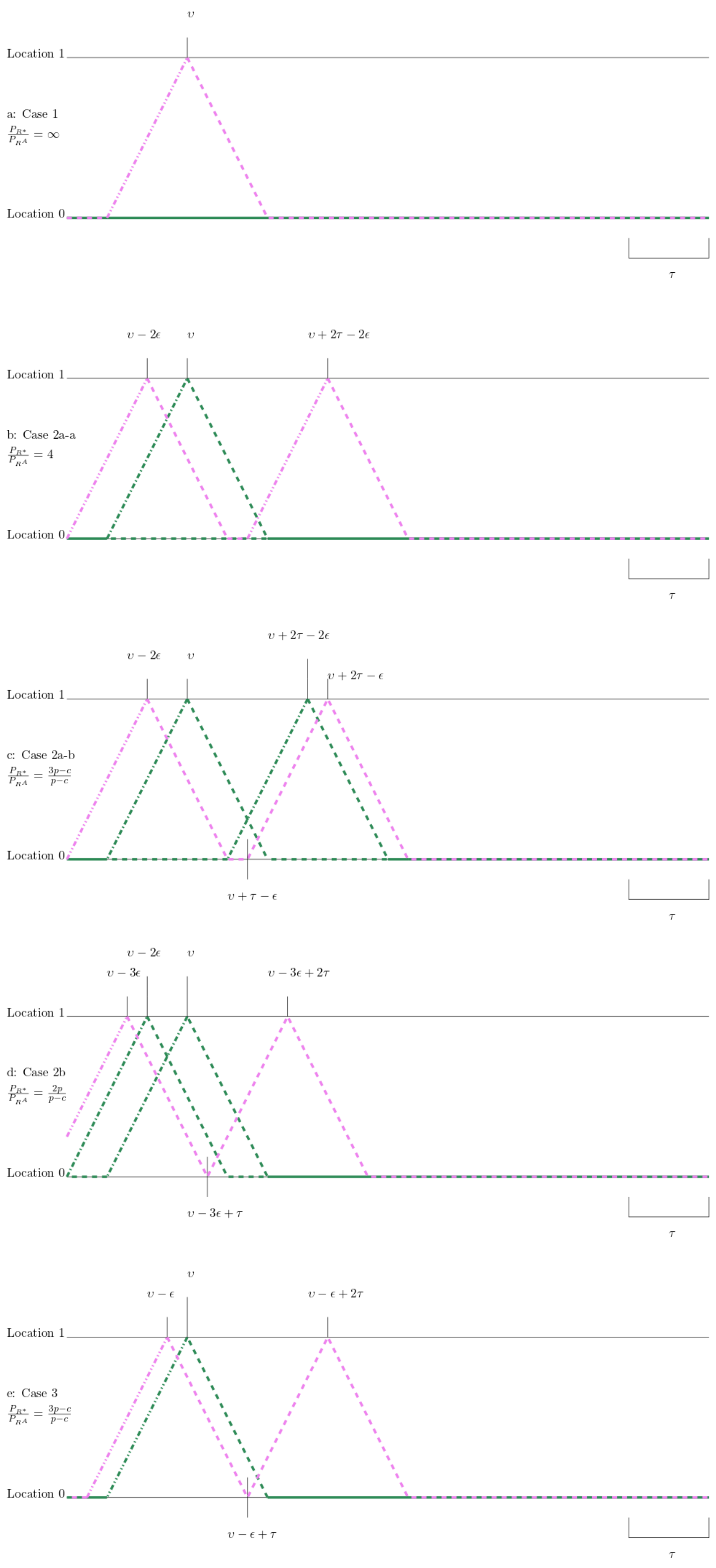

υ
Location 1
a: Case 1
$\frac{P_{R^*}}{P_{R^A}} = \infty$
Location 0
τ
$\upsilon - 2\epsilon$
υ
$\upsilon + 2\tau - 2\epsilon$
Location 1
b: Case 2a-a
$\frac{P_{R^*}}{P_{R^A}} = 4$
Location 0
τ
$\upsilon + 2\tau - 2\epsilon$
$\upsilon - 2\epsilon$
υ
$\upsilon + 2\tau - \epsilon$
Location 1
c: Case 2a-b
$\frac{P_{R^*}}{P_{R^A}} = \frac{3p-c}{p-c}$
Location 0
$\upsilon + \tau - \epsilon$
τ
$\upsilon - 2\epsilon$
υ
$\upsilon - 3\epsilon$
$\upsilon - 3\epsilon + 2\tau$
Location 1
d: Case 2b
$\frac{P_{R^*}}{P_{R^A}} = \frac{2p}{p-c}$
Location 0
$\upsilon - 3\epsilon + \tau$
τ
υ
$\upsilon - \epsilon$
$\upsilon - \epsilon + 2\tau$
Location 1
e: Case 3
$\frac{P_{R^*}}{P_{R^A}} = \frac{3p-c}{p-c}$
Location 0
$\upsilon - \epsilon + \tau$
τ

Figure 6: All cases of Theorem 5.6, illustrated with appropriate competitive ratio. The location of the online algorithm (in green) and the optimum algorithm (in pink, dotted) are shown over time, as is the length of a trip (τ). If the green line is dashed, that means the two servers are not at the same location at that time. If a line is both dotted and dashed, it indicates an empty movement performed by a number of servers equal to the amount of dots.

accepts both r_5 and r_6. We find $P_{R^A} = p$ and $P_{R^*} = 4p$, hence $P_{R^*}/P_{R^A} = 4$. (See also Figure 7b)
Case 2b: ALG accepts either r_3 or r_4. The reaction to this case mirrors the reaction used in Case 3. The adversary's reaction in Case 3 can be used in this case, changing only a few ϵ counts. The results from Case 3 reflect the results achieved in this case.
Case 3: ALG accepts both r_1 and r_2. Over time, the adversary then releases a Zig-Zag gadget starting at location 0 and time $\upsilon - \epsilon$. These requests are in conflict with the requests that have been accepted by ALG (Lemma 5.2), and OPT accepts all of them. Whilst releasing this gadget, the adversary also releases $r_3 = r_4 = (\epsilon, \lambda + 5\epsilon, 0)$. Note that there is at least τ time between $\bar{t}_{r_3}$ and the Zig-Zag gadget. Therefore, there are no conflicts regarding r_3 and r_4. We distinguish three cases.
Case 3a: ALG rejects both r_3 and r_4. OPT accepts both r_3 and r_4. The adversary then releases $r_5 = r_6 = (\tau + \epsilon, \lambda + 5\epsilon + \tau, 1)$. OPT accepts both r_5 and r_6. These requests are not acceptable to ALG. We find $P_{R^A} = 2p$ and $P_{R^*} = 10p$, hence $P_{R^*}/P_{R^A} = 5$. (See also Figure 7c)
Case 3b: ALG accepts either r_3 or r_4. OPT rejects both r_3 and r_4. The adversary then releases $r_5 = r_6 = (2\epsilon, \lambda + 4\epsilon, 0)$. We distinguish two cases.
Case 3b-a: ALG rejects both r_5 and r_6. OPT accepts both r_5 and r_6. The adversary then releases $r_7 = r_8 = (\tau + 2\epsilon, \lambda + 4\epsilon + \tau, 1)$. OPT accepts both r_7 and r_8. These requests are not acceptable to ALG. We find $P_{R^A} = 3p - c$ and $P_{R^*} = 10p$, hence $P_{R^*}/P_{R^A} = \frac{10p}{3p-c}$. (See also Figure 7d)
Case 3b-b: ALG accepts either r_5 and r_6. OPT rejects both r_5 and r_6. The adversary then releases a Zig gadget at location 0 starting at time $\lambda + 3\epsilon$. OPT fully accepts this Zig gadget. Since this Zig gadget fully conflicts with r_3 and r_5, by Corollary 5.3, these requests are not acceptable to ALG. We find $P_{R^A} = 4p - 2c$ and $P_{R^*} = 10p$, hence $P_{R^*}/P_{R^A} = \frac{10p}{4p-2c}$. (See also Figure 7e)
Case 3c: ALG accepts both r_3 and r_4. OPT rejects both r_3 and r_4. The adversary then releases a Zig gadget at location 0 starting at time $\lambda + 4\epsilon$. OPT fully accepts this Zig gadget. Since this Zig gadget fully conflicts with r_3 and r_4, by Corollary 5.3, these requests are not acceptable to ALG. We find $P_{R^A} = 4p - 2c$ and $P_{R^*} = 10p$, hence $P_{R^*}/P_{R^A} = \frac{10p}{4p-2c}$. (See also Figure 7f) □

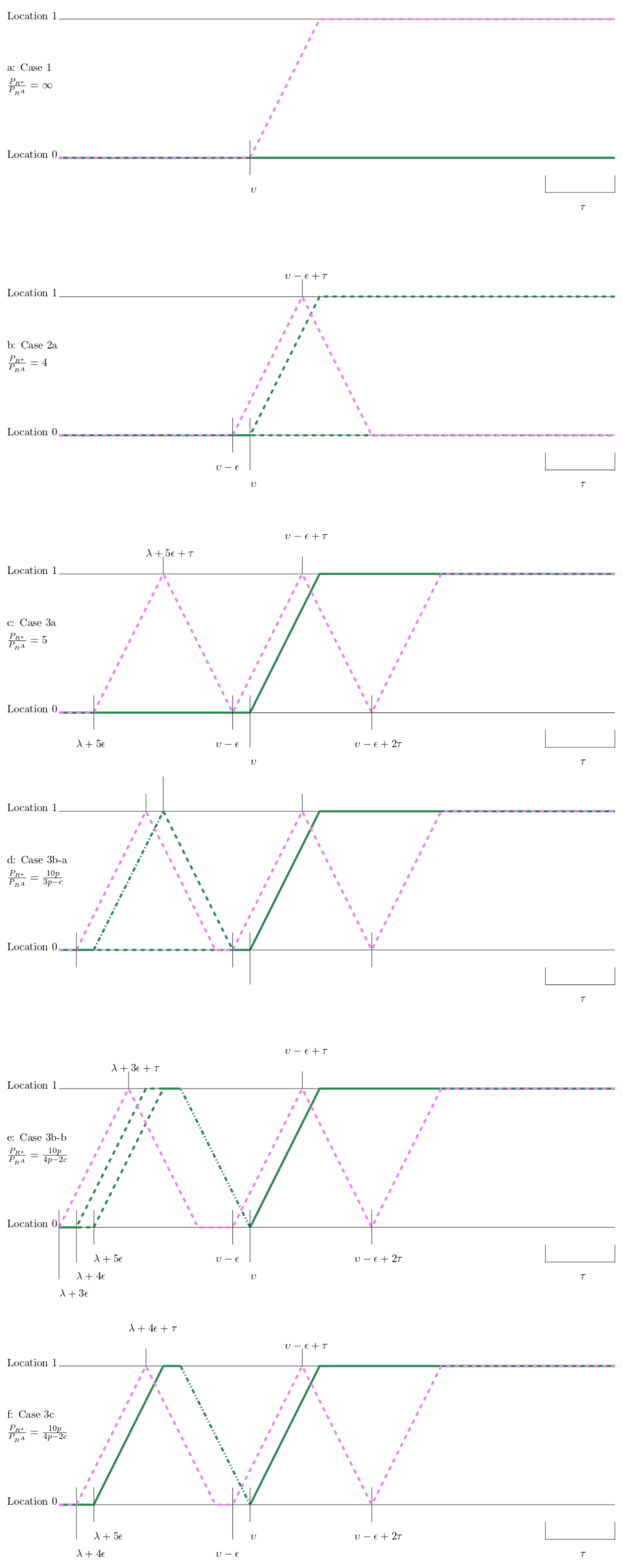
5.4 Long Notification Intervals

In this section we discuss the impacts of notification intervals of longer than 3τ on the lower bounds 2S2L-V problem. In section 5.2, we found the lower bound for the settings disallowing distant requests by first releasing a pair of non-distant requests. In section 5.3, we found the lower bound for the settings with notification interval shorter than 3τ by first releasing a pair of distant requests. In this section, we find that the way to find the lower bounds for settings with $\upsilon - \lambda > 3\tau$ may differ based on the setting for c. This is the case because requests released as non-distant requests may become distant by the time the requests need to be served. The longer the notification interval, the more an adversary can take advantage of non-distant requests turning into distant requests.

Importantly, in this section, we do not analyze the case starting with a pair of distant requests again. This is unnecessary, because even distant requests planned with a large notification interval do not get any less profitable as time goes on. Therefore, the reaction and bounds resulting from releasing a distant request first do not change with more time. For analysis of the lower bounds caused by a pair of distant requests released first, we refer to **Theorem 5.6**.

Before we analyze the lower bounds for these long booking intervals, we first discuss a new gadget. We call this gadget the *commitment gadget*. The commitment gadget is released in the far future to influence the behaviour of ALG. ALG has to accept part of the commitment gadget to remain competitive. Because of that ALG has to move its servers to serve the requests accepted in the commitment gadget. This is how OPT manages to make more profit over ALG than possible in the short booking interval settings, despite starting with a non-distant request.

The Commitment gadget is constructed using the following steps. First, the adversary releases requests $r_1 = r_2 = (0, \upsilon - \tau, 0)$. Note that this request is booked nearly as far in advance as possible. If an algorithm rejects all of these requests, that immediately classifies that algorithms as uncompetitive, because OPT can accept r_1 to make uncontested profit over the algorithm. If an algorithm accepts both of these requests, an adversary can release a Zig-Zag gadget starting at location 0 at time $\upsilon - \tau - \epsilon$. Per Lemma 5.2, all of this Zig-Zag gadget is in conflict with r_1 and r_2. Therefore, ALG cannot accept this Zig-Zag gadget. OPT then accepts this Zig-Zag gadget. We call this variation of the commitment



Location 1
a: Case 1
$\frac{P_{R^*}}{P_{R^A}} = \infty$
Location 0
υ
τ
$\upsilon - \epsilon + \tau$
Location 1
b: Case 2a
$\frac{P_{R^*}}{P_{R^A}} = 4$
Location 0
$\upsilon - \epsilon$
υ
τ
$\lambda + 5\epsilon + \tau$
$\upsilon - \epsilon + \tau$
Location 1
c: Case 3a
$\frac{P_{R^*}}{P_{R^A}} = 5$
Location 0
$\lambda + 5\epsilon$
$\upsilon - \epsilon$
υ
$\upsilon - \epsilon + 2\tau$
τ
Location 1
d: Case 3b-a
$\frac{P_{R^*}}{P_{R^A}} = \frac{10p}{3p-c}$
Location 0
τ
$\lambda + 3\epsilon + \tau$
$\upsilon - \epsilon + \tau$
Location 1
e: Case 3b-b
$\frac{P_{R^*}}{P_{R^A}} = \frac{10p}{4p-2c}$
Location 0
$\lambda + 5\epsilon$
$\lambda + 4\epsilon$
$\lambda + 3\epsilon$
$\upsilon - \epsilon$
υ
$\upsilon - \epsilon + 2\tau$
τ
$\lambda + 4\epsilon + \tau$
$\upsilon - \epsilon + \tau$
Location 1
f: Case 3c
$\frac{P_{R^*}}{P_{R^A}} = \frac{10p}{4p-2c}$
Location 0
$\lambda + 5\epsilon$
$\lambda + 4\epsilon$
$\upsilon - \epsilon$
υ
$\upsilon - \epsilon + 2\tau$
τ

Figure 7: All cases of Theorem 5.7, illustrated with appropriate competitive ratio. The location of the online algorithm (in green) and the optimum algorithm (in pink, dotted) are shown over time, as is the length of a trip (τ). If the green line is dashed, that means the two servers are not at the same location at that time. If a line is both dotted and dashed, it indicates an empty movement performed by a number of servers equal to the amount of dots.

gadget, a *Class A* commitment gadget (See also Figure 8 Class A).

Having handled the cases where ALG accepts either both or neither of r_1 and r_2, we now consider the case where ALG accepts just one of these requests. The first step the adversary takes after ALG accepts either r_1 or r_2 is releasing $r_3 = r_4 = (\epsilon, \upsilon - \tau - 2\epsilon, 0)$. If ALG accepts either r_3 or r_4 (ALG cannot accept both, because r_3 and r_4 conflict with r_1 or r_2), the adversary simplifies the situation into a Class A commitment gadget. The adversary does this by releasing a Zig-Zag gadget at location 0 at time $\upsilon - \tau - 3\epsilon$.

We now handle the final cases in the commitment gadget. We know one of ALG's servers is busy serving r_1 or r_2 at $\upsilon - \epsilon$, while the other is available. At the same time, both of OPT's servers have accepted a request line-skipping r_1 or r_2, making them available at $\upsilon - \epsilon$. The adversary now releases $r_5 = r_6 = (2\epsilon, \upsilon - \epsilon, 1)$. This gives ALG two choices: either accepting or rejecting r_5 (or r_6). If ALG accepts either of these requests, the adversary releases a Zig-gadget starting at location 1 at time $\upsilon - 2\epsilon$. By Corollary 5.3 and Lemma 5.1, all requests in this Zig-gadget conflict with r_1 and r_5. Since r_1 and r_5 also conflict with one another, it is impossible for ALG to accept this Zig-gadget. OPT accepts the Zig-gadget. We call this variation of the commitment gadget a *Class B* commitment gadget (See also Figure 8 Class B). Meanwhile, if ALG rejects both r_5 and r_6, OPT accepts both of these requests and the releases for the commitment gadget end. We call this final variation a *Class C* commitment gadget (See also Figure 8 Class C).

The commitment gadget is a strong tool, that forces ALG into committing to a certain course of action near the end of the notification interval. Because ALG needs to stick to its commitment, it becomes vulnerable to shorter term releases. Indeed, because of the request(s) OPT accepted in the commitment gadget, non-distant requests line-skipping the commitment gadget come with an empty movement cost later down the line. This has a profound impact on the lower bounds of this problem.

The lower bound proofs on the 2S2L-V problem from here on all have very similar structures. The adversary begins by either releasing a distant request, or a commitment gadget. The cases starting with a distant request have been thoroughly discussed in Theorem 5.6. Following Theorem 5.6, we find that an adversary can force a ratio of $\{\min\{\frac{3p-c}{p-c}, 4\}$. Alternatively, the adversary can try to find a higher lower bound by releasing a non-distant request first. To this end, the adversary first releases a commitment gadget. Once again note that this commitment gadget is initiated such that it is near the end of the notification interval, despite being released first. The cases starting with a commitment gadget continue with the release of non-distant requests line-skipping the gadget. Depending on the reaction of ALG, and the amount of time between the commitment gadget and the start of the timeline, the adversary releases Zig gadgets or Zig-Zag gadgets in conflict with requests ALG accepts. This approach constructs a formula for the lower bound that trends towards $\frac{3p-c}{p-c}$ as $\upsilon - \lambda$ trends to infinity. However, for $\upsilon - \lambda > 9\tau$, this formula becomes less competitive than accepting a Class B commitment gadget. Therefore, the lower bound no longer changes when $\upsilon - \lambda > 9\tau$.

Before we show the proofs, we once more reiterate what the min functions and max functions used in the lower bounds stand for. min functions are caused by a change in ALG behaviour. The instance released by the adversary remains the same for both of the elements in the min function. Meanwhile, max functions are caused by a change in instance released by the adversary. Even if ALG's behaviour remains consistent, the adversary forces a worse lower bound purely by releasing an instance that has no response causing a lower bound that is lower than the other element in the max function.

Theorem 5.8 For $3\tau < \upsilon - \lambda \leq 5\tau$ and $0 < c \leq p$, no deterministic online algorithm can achieve a competitive ratio of lower than $\max\{\min\{\frac{3p-c}{p-c}, 4\}, \frac{10p}{4p-2c}\}$

Proof We prove this theorem by use of case distinction. First the adversary releases either a distant request or a non-distant request. We note that releasing a distant request first leads to the analysis seen in **Theorem 5.6**. Therefore, the lower bound of this setting cannot be lower than $\{\min\{\frac{3p-c}{p-c}, 4\}$. We note that this value is higher than $\frac{10p}{4p-2c}$ for $c \leq \frac{3p}{4}$. Thus, the lower bound is $\frac{3p-c}{p-c}$ for $c \leq \frac{p}{3}$, 4 for $\frac{p}{3} \leq c \leq \frac{3p}{4}$ and $\frac{10p}{4p-2c}$ for $c \geq \frac{3p}{4}$. To prove the lower bound, we only have to use case distinction on new cases. Those are found by having the adversary release non-distant requests first.

The adversary first releases the commitment gadget. If ALG's reaction causes a Class B commitment gadget, the release of requests ends. We find $P_{R^A} = 2p - c$ and $P_{R^*} = 6p$, hence $P_{R^*}/P_{R^A} = \frac{6p}{2p-c}$. Alternatively, either a Class A or Class C commitment gadget is caused. This does not matter for the response of the adversary, who releases $r_1 = r_2 = (3\epsilon, \lambda + 7\epsilon, 0)$. We distinguish three cases.

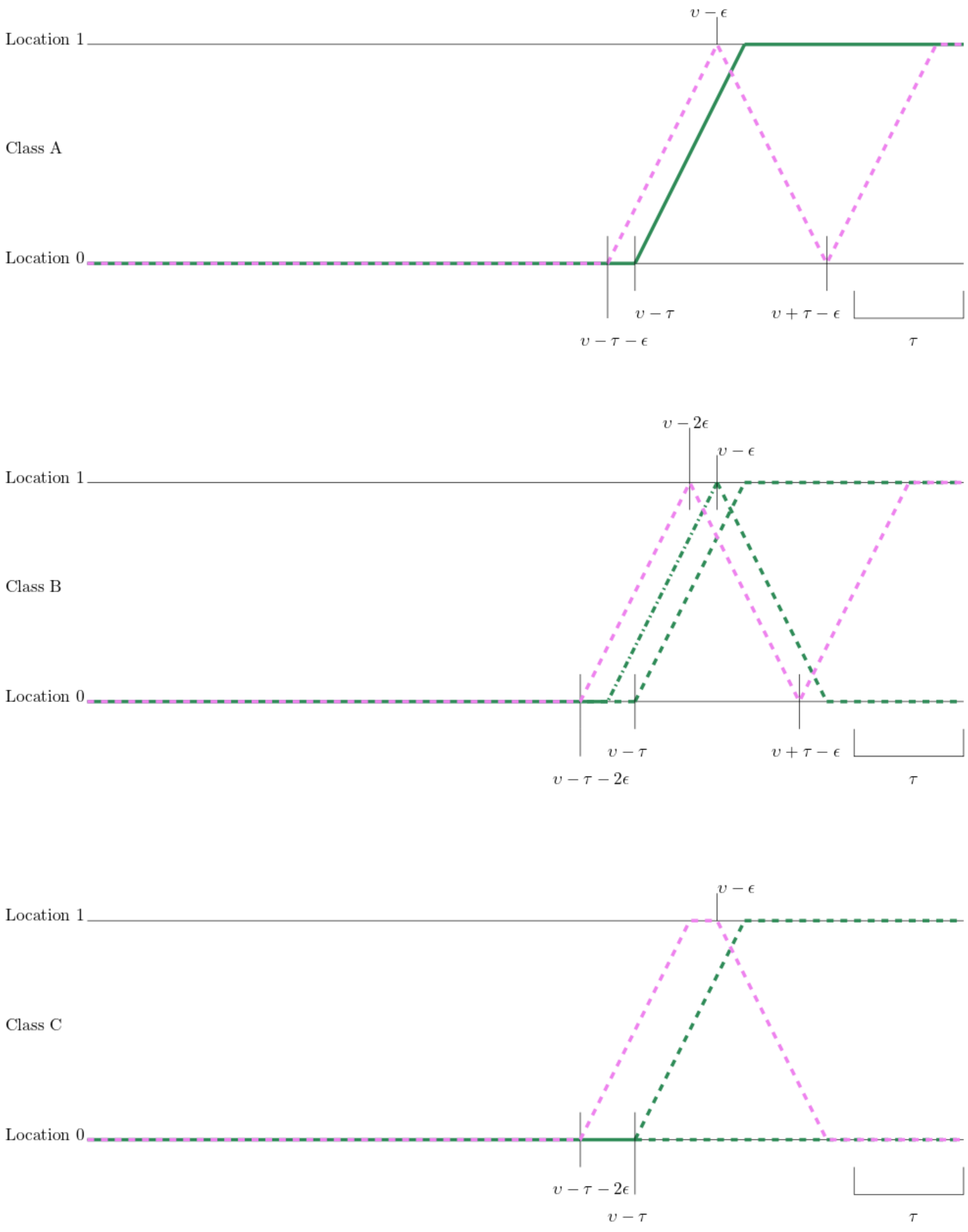


Figure 8: All classes of the commitment gadget. The location of the online algorithm (in green) and the optimum algorithm (in pink, dotted) are shown over time, as is the length of a trip (τ). If the green line is dashed, that means the two servers are not at the same location at that time. If a line is both dotted and dashed, it indicates an empty movement performed by a number of servers equal to the amount of dots.

Case 1: ALG accepts neither r_1 nor r_2. OPT accepts both of these requests. The adversary then waits to release $r_3 = r_4 = (3\epsilon + \tau, \tau + \lambda + 7\epsilon, 1)$. Since neither of ALG's servers can be at location 1 in time to serve r_3 and r_4 if $\lambda < \tau$, these requests cannot be accepted by ALG. OPT accepts both r_3 and r_4. If ALG's reaction to the commitment gadget created a Class A commitment gadget, we find $P_{R^A} = 2p$ and $P_{R^*} = 10p$, hence $P_{R^*}/P_{R^A} = 5$. Alternatively, if ALG's reaction created a Class C commitment gadget, we find $P_{R^A} = p$ and $P_{R^*} = 8p$, hence $P_{R^*}/P_{R^A} = 8$.
Case 2: ALG accepts either r_1 or r_2. OPT rejects both requests. The adversary releases $r_3 = r_4 = (4\epsilon, \lambda + 6\epsilon, 0)$. We distinguish two cases.
Case 2a: ALG rejects both r_3 and r_4. OPT then accepts both r_3 and r_4. The adversary releases $r_5 = r_6 = (6\epsilon + \tau, \lambda + 6\epsilon + \tau, 1)$. Since neither of ALG's servers can be at location 1 in time to serve r_5 and r_6 if $\lambda < \tau$, these requests cannot be accepted by ALG. If ALG's reaction to the commitment gadget created a Class A commitment gadget, we find $P_{R^A} = 3p - c$ and $P_{R^*} = 10p$, hence $P_{R^*}/P_{R^A} = \frac{10p}{3p-c}$. Alternatively, if ALG's reaction created a Class C commitment gadget, the adversary releases two more requests $r_7 = r_8 = (\upsilon + \tau - \epsilon - \lambda, \upsilon + \tau - \epsilon, 0)$. Since neither of ALG's servers can be at location 0 in time to serve r_7 and r_8 if $\lambda < \tau$, these requests cannot be accepted by ALG. OPT accepts both of these requests. We find $P_{R^A} = 2p$ and $P_{R^*} = 10p$, hence $P_{R^*}/P_{R^A} = 5$.
Case 2b: ALG accepts either r_3 or r_4. OPT rejects both requests. The adversary releases a Zig-gadget starting from location 0 at time $5\epsilon + \lambda$. By Corollary 5.3, ALG cannot accept any requests in this Zig-gadget. OPT fully accepts the Zig-gadget. If ALG's reaction to the commitment gadget created a Class A commitment gadget, we find $P_{R^A} = 4p - 2c$ and $P_{R^*} = 10p$, hence $P_{R^*}/P_{R^A} = \frac{10p}{4p-2c}$. Alternatively, if ALG's reaction created a Class C commitment gadget, the adversary releases two more requests $r_9 = r_{10} = (\upsilon + \tau - \epsilon - \lambda, \upsilon + \tau - \epsilon, 0)$. Since neither of ALG's servers can be at location 0 in time to serve r_9 and r_{10} if $\lambda < \tau$, these requests cannot be accepted by ALG. OPT accepts both of these requests. We find $P_{R^A} = 3p - c$ and $P_{R^*} = 10p$, hence $P_{R^*}/P_{R^A} = \frac{10p}{3p-c}$.
Case 3: ALG accepts both r_1 and r_2. The reaction to this case mirrors the reaction used in Case 2b. The adversary's reaction in Case 2b can be used in this case, changing only a few ϵ counts. The results from Case 2b reflect the results achieved in this case.
We find that the lower bound of first releasing a non-distant request is achieved by first releasing a Class A commitment gadget, followed by either Case 2b or Case 3. The lower bound of first releasing a distant request remains unchanged from the one discovered in Theorem 5.6. Therefore, we find a lower bound of $\max\{\min\{\frac{3p-c}{p-c}, 4\}, \frac{10p}{4p-2c}\}$ □

Theorem 5.9 For $5\tau < \upsilon - \lambda \leq 7\tau$ and $0 < c \leq p$, no deterministic online algorithm can achieve a competitive ratio of lower than $\max\{\min\{\frac{3p-c}{p-c}, 4\}, \frac{12p-2c}{4p-2c}\}$

Proof We prove this theorem by use of case distinction. First the adversary releases either a distant request or a non-distant request. We note that releasing a distant request first leads to the analysis seen in **Theorem 5.6**. Therefore, the lower bound of this setting cannot be lower than $\{\min\{\frac{3p-c}{p-c}, 4\}$. We note that this value is higher than $\frac{12p-2c}{4p-2c}$ for $c \leq \frac{2p}{3}$. Thus, the lower bound is $\frac{3p-c}{p-c}$ for $c \leq \frac{p}{3}$, 4 for $\frac{p}{3} \leq c \leq \frac{2p}{3}$ and $\frac{12p-2c}{4p-2c}$ for $c \geq \frac{2p}{3}$. To prove the lower bound, we only have to use case distinction on new cases. Those are created by releasing non-distant requests first.

The adversary first releases the commitment gadget. If ALG's reaction causes a Class B commitment gadget, the release of requests ends. We find $P_{R^A} = 2p - c$ and $P_{R^*} = 6p$, hence $P_{R^*}/P_{R^A} = \frac{6p}{2p-c}$. Alternatively, either a Class A or Class C commitment gadget is caused. This does not matter for the response of the adversary, who releases $r_1 = r_2 = (3\epsilon, \lambda + 7\epsilon, 0)$. We distinguish three cases.

Case 1: ALG accepts neither r_1 nor r_2. OPT accepts both of these requests. The adversary then waits to release $r_3 = r_4 = (3\epsilon + \tau, \tau + \lambda + 7\epsilon, 1)$. Since neither of ALG's servers can be at location 1 in time to serve r_3 and r_4 if $\lambda < \tau$, these requests cannot be accepted by ALG. OPT accepts both r_3 and r_4. If ALG's reaction to the commitment gadget created a Class A commitment gadget, we find $P_{R^A} = 2p$ and $P_{R^*} = 10p$, hence $P_{R^*}/P_{R^A} = 5$. Alternatively, if ALG's reaction created a Class C commitment gadget, we find $P_{R^A} = p$ and $P_{R^*} = 8p$, hence $P_{R^*}/P_{R^A} = 8$.
Case 2: ALG accepts either r_1 or r_2. OPT rejects both requests. The adversary releases $r_3 = r_4 = (4\epsilon, \lambda + 6\epsilon, 0)$. We distinguish two cases.
Case 2a: ALG rejects both r_3 and r_4. OPT then accepts both r_3 and r_4. The adversary releases

$r_5 = r_6 = (6\epsilon + \tau, \lambda + 6\epsilon + \tau, 1)$. Since neither of ALG's servers can be at location 1 in time to serve r_5 and r_6 if $\lambda < \tau$, these requests cannot be accepted by ALG. If ALG's reaction to the commitment gadget created a Class A commitment gadget, we find $P_{R^A} = 3p - c$ and $P_{R^*} = 10p$, hence $P_{R^*}/P_{R^A} = \frac{10p}{3p-c}$. Alternatively, if ALG's reaction created a Class C commitment gadget, the adversary releases two more requests $r_7 = r_8 = (\upsilon + \tau - \epsilon - \lambda, \upsilon + \tau - \epsilon, 0)$. Since neither of ALG's servers can be at location 0 in time to serve r_7 and r_8 if $\lambda < \tau$, these requests cannot be accepted by ALG. OPT accepts both of these requests. We find $P_{R^A} = 2p$ and $P_{R^*} = 10p$, hence $P_{R^*}/P_{R^A} = 5$.
Case 2b: ALG accepts either r_3 or r_4. OPT rejects both requests. The adversary releases a Zig-Zag gadget starting from location 0 at time $5\epsilon + \lambda$. By Lemma 5.2, ALG cannot accept any requests in this Zig-Zag gadget. OPT fully accepts the Zig-Zag gadget. If ALG's reaction to the commitment gadget created a Class A commitment gadget, we find $P_{R^A} = 4p - 2c$ and $P_{R^*} = 12p - 2c$, hence $P_{R^*}/P_{R^A} = \frac{12p-2c}{4p-2c}$. Alternatively, if ALG's reaction created a Class C commitment gadget, the adversary releases two more requests $r_9 = r_{10} = (\upsilon + \tau - \epsilon - \lambda, \upsilon + \tau - \epsilon, 0)$. Since neither of ALG's servers can be at location 0 in time to serve r_9 and r_{10} if $\lambda < \tau$, these requests cannot be accepted by ALG. OPT accepts both of these requests. We find $P_{R^A} = 3p - c$ and $P_{R^*} = 12p - 2c$, hence $P_{R^*}/P_{R^A} = \frac{12p-2c}{3p-c}$.
Case 3: ALG accepts both r_1 and r_2. The reaction to this case mirrors the reaction used in Case 2b. The adversary's reaction in Case 2b can be used in this case, changing only a few ϵ counts. The results from Case 2b reflect the results achieved in this case.
We find that the lower bound of first releasing a non-distant request is achieved by first releasing a Class A commitment gadget, followed by either Case 2b or Case 3. The lower bound of first releasing a distant request remains unchanged from the one discovered in Theorem 5.6. Therefore, we find a lower bound of $\max\{\min\{\frac{3p-c}{p-c}, 4\}, \frac{12p-2c}{4p-2c}\}$ □

Theorem 5.10 For $7\tau < \upsilon - \lambda \leq 9\tau$ and $0 < c \leq p$, no deterministic online algorithm can achieve a competitive ratio of lower than $\max\{\min\{\frac{3p-c}{p-c}, 4\}, \min\{\frac{16p-2c}{6p-4c}, \frac{12p-2c}{3p-c}\}\}$

Proof We prove this theorem by use of case distinction. First the adversary releases either a distant request or a non-distant request. We note that releasing a distant request first leads to the analysis seen in **Theorem 5.6**. Therefore, the lower bound of this setting cannot be lower than $\{\min\{\frac{3p-c}{p-c}, 4\}$. Next, we consider the two possible lower bounds claimed by releasing a non-distant request first. $\frac{16p-2c}{6p-4c} \leq \frac{12p-2c}{3p-c}$ for $c \leq \frac{19-\sqrt{217}}{6}p$. Furthermore $\frac{16p-2c}{6p-4c} \leq 4$ for $c \leq \frac{4p}{7}$ Thus, we find that the complete lower bound is $\frac{3p-c}{p-c}$ for $0 < c \leq \frac{p}{3}$, 4 for $\frac{p}{3} \leq c \leq \frac{4p}{7}$, $\frac{16p-2c}{6p-4c}$ for $\frac{4p}{7} \leq c \leq \frac{19-\sqrt{217}}{6}p$, and finally $\frac{12p-c}{3p-c}$ for $\frac{19-\sqrt{217}}{6}p \leq c \leq 1$.

The adversary first releases the commitment gadget. If ALG's reaction causes a Class B commitment gadget, the release of requests ends. We find $P_{R^A} = 2p - c$ and $P_{R^*} = 6p$, hence $P_{R^*}/P_{R^A} = \frac{6p}{2p-c}$.

Alternatively, either a Class A or Class C commitment gadget is caused. This does not matter for the response of the adversary, who releases $r_1 = r_2 = (3\epsilon, \upsilon - 5\tau - 10\epsilon, 0)$, We distinguish three cases.

Case 1: ALG accepts neither r_1 nor r_2. OPT accepts both of these requests. The adversary then waits to release $r_3 = r_4 = (3\epsilon + \upsilon - \lambda - 4\tau - 10\epsilon, \upsilon - 4\tau - 10\epsilon, 1)$. Since neither of ALG's servers can be at location 1 in time to serve r_3 and r_4 if $\lambda < \tau$, these requests cannot be accepted by ALG. OPT accepts both r_3 and r_4. If ALG's reaction to the commitment gadget created a Class A commitment gadget, we find $P_{R^A} = 2p$ and $P_{R^*} = 10p$, hence $P_{R^*}/P_{R^A} = 5$. Alternatively, if ALG's reaction created a Class C commitment gadget, we find $P_{R^A} = p$ and $P_{R^*} = 8p$, hence $P_{R^*}/P_{R^A} = 8$.
Case 2: ALG accepts either r_1 or r_2. OPT rejects both requests. The adversary releases $r_3 = r_4 = (4\epsilon, \upsilon - 5\tau - 12\epsilon, 0)$. We distinguish two cases.
Case 2a: ALG rejects both r_3 and r_4. OPT accepts both requests. The adversary then releases $r_5 = r_6 = (5\epsilon, \upsilon - 3\tau - 12\epsilon, 0)$. We distinguish two more cases.
Case 2a-a: ALG rejects both r_5 and r_6. OPT accepts both requests. The adversary then waits to release $r_7 = r_8 = (\upsilon - 4\tau - \lambda - 12\epsilon, \upsilon - 4\tau - 12\epsilon, 1)$. Since neither of ALG's servers can be at location 1 in time to serve r_7 and r_8 if $\lambda < \tau$, these requests cannot be accepted by ALG. OPT accepts both r_7 and r_8. If ALG's reaction to the commitment gadget created a Class A commitment gadget, we find $P_{R^A} = 3p - c$ and $P_{R^*} = 12p - 2c$, hence $P_{R^*}/P_{R^A} = \frac{12p-2c}{3p-c}$. Alternatively, if ALG's reaction created a Class C commitment gadget, the adversary releases two more requests $r_7 = r_8 = (\upsilon + \tau - \epsilon - \lambda, \upsilon + \tau - \epsilon, 0)$. Since neither of ALG's servers can be at location 0 in time to serve r_7 or r_8 if $\lambda < \tau$, these requests cannot be accepted by ALG. OPT accepts both of these requests. We find $P_{R^A} = 2p$ and $P_{R^*} = 12p - 2c$, hence

$P_{R^*}/P_{R^A} = \frac{12p-2c}{2p}$.
Case 2a-b: ALG accepts either r_5 or r_6. OPT rejects both requests. The adversary releases a Zig-gadget starting at location 1 at time $\upsilon - 4\tau - 11\epsilon$. By Corollary 5.3, this Zig-gadget is in conflict with requests r_1 and r_5. Therefore, no part of this Zig gadget can be accepted by ALG. OPT fully accepts the Zig-gadget. The adversary then releases $r_7 = r_8 = (5\epsilon, \lambda + 9\epsilon, 0)$. We distinguish three cases.
Case 2a-b-1: ALG accepts neither r_7 nor r_8. OPT accepts both of these requests. The adversary then waits to release $r_9 = r_{10} = (6\epsilon + \tau, \lambda + \tau + 9\epsilon, 1)$. Since neither of ALG's servers can be at location 1 in time to serve r_9 and r_{10} if $\lambda < \tau$, these requests cannot be accepted by ALG. OPT accepts both r_9 and r_{10}. If ALG's reaction to the commitment gadget created a Class A commitment gadget, we find $P_{R^A} = 4p - 2c$ and $P_{R^*} = 16p - 2c$, hence $P_{R^*}/P_{R^A} = \frac{16p-2c}{4p-2c}$. Alternatively, if ALG's reaction created a Class C commitment gadget, the adversary releases two more requests $r_{11} = r_{12} = (\upsilon + \tau - \epsilon - \lambda, \upsilon + \tau - \epsilon, 0)$. Since neither of ALG's servers can be at location 0 in time to serve r_{11} or r_{12} if $\lambda < \tau$, these requests cannot be accepted by ALG. OPT accepts both of these requests. We find $P_{R^A} = 3p - c$ and $P_{R^*} = 16p - 2c$, hence $P_{R^*}/P_{R^A} = \frac{16p-2c}{3p-c}$.
Case 2a-b-2: ALG accepts either r_5 or r_6. OPT rejects both of these requests. The adversary releases $r_7 = r_8 = (6\epsilon, \lambda + 8\epsilon, 0)$. We distinguish two cases.
Case 2a-b-2a: ALG accepts neither r_7 nor r_8. OPT accepts both of these requests. The adversary releases $r_9 = r_{10} = (8\epsilon + \tau, \lambda + 8\epsilon + \tau, 1)$. Since neither of ALG's servers can be at location 1 in time to serve r_9 and r_{10} if $\lambda < \tau$, these requests cannot be accepted by ALG. OPT accepts both of them. If ALG's reaction to the commitment gadget created a Class A commitment gadget, we find $P_{R^A} = 5p - 3c$ and $P_{R^*} = 16p - 2c$, hence $P_{R^*}/P_{R^A} = \frac{16p-2c}{5p-3c}$.. Alternatively, if ALG's reaction created a Class C commitment gadget, the adversary releases two more requests $r_{11} = r_{12} = (\upsilon + \tau - \epsilon - \lambda, \upsilon + \tau - \epsilon, 0)$. Since neither of ALG's servers can be at location 0 in time to serve r_{11} or r_{12} if $\lambda < \tau$, these requests cannot be accepted by ALG. OPT accepts both of these requests. We find $P_{R^A} = 4p - 2c$ and $P_{R^*} = 16p - 2c$, hence $P_{R^*}/P_{R^A} = \frac{16p-2c}{4p-2c}$.
Case 2a-b-2b: ALG accepts either r_7 or r_8. OPT rejects both requests. The adversary releases a Zig gadget starting at location 0 at time $\lambda + 7\epsilon$. By Corollary 5.3, ALG cannot accept any requests in this Zig gadget. OPT fully accepts the Zig gadget. If ALG's reaction to the commitment gadget created a Class A commitment gadget, we find $P_{R^A} = 6p - 4c$ and $P_{R^*} = 16p - 4c$, hence $P_{R^*}/P_{R^A} = \frac{16p-4c}{6p-4c}$.. Alternatively, if ALG's reaction created a Class C commitment gadget, the adversary releases two more requests $r_{11} = r_{12} = (\upsilon + \tau - \epsilon - \lambda, \upsilon + \tau - \epsilon, 0)$. Since neither of ALG's servers can be at location 0 in time to serve r_{11} or r_{12} if $\lambda < \tau$, these requests cannot be accepted by ALG. OPT accepts both of these requests. We find $P_{R^A} = 5p - 3c$ and $P_{R^*} = 16p - 4c$, hence $P_{R^*}/P_{R^A} = \frac{16p-4c}{5p-3c}$.
Case 2a-b-3: ALG accepts both r_7 and r_8. This reaction mirrors the reaction used in Case 2a-2b, changing only a few ϵ counts. The results from Case 2a-2b reflect the results achieved in this case.
Case 2b: ALG accepts either r_3 or r_4. OPT rejects both requests. The adversary releases a Zig-Zag gadget starting at location 0 at time $\upsilon - 5\tau - 13\epsilon$. By Lemma 5.2, ALG cannot accept any requests in this Zig-Zag gadget. OPT fully accepts the Zig-Zag gadget. This reaction mirrors the reaction used in Case 2a-b, changing a few ϵ counts, and slightly delaying the final four requests in the Zig-Zag gadget. The results from Case 2a-b reflect the results achieved in this case.
Case 3: ALG accepts both r_1 and r_2. This reaction mirrors the reaction used in Case 2a-b, changing only a few ϵ counts. The results from Case 2a-b reflect the results achieved in this case.

Theorem 5.11 For $8\tau < \upsilon - \lambda$ and $0 < c \leq p$, no deterministic online algorithm can achieve a competitive ratio of lower than $\max\{\min\{\frac{3p-c}{p-c}, 4\}, \min\{\frac{6p+4p(\lceil\frac{\upsilon-\lambda-4\tau}{4\tau}\rceil)+(2p-2c)(\lceil\frac{\upsilon-\lambda-6\tau}{4\tau}\rceil)}{2p+(2p-2c)(\lceil\frac{\upsilon-\lambda-4\tau}{4\tau}\rceil)}, \frac{12p-2c}{3p-c}\}\}$

Proof We prove this theorem by use of case distinction. First the adversary releases either a distant request or a non-distant request. We note that releasing a distant request first leads to the analysis seen in Theorem 5.6. Therefore, the lower bound of this setting cannot be lower than $\{\min\{\frac{3p-c}{p-c}, 4\}$. Next, we consider the two possible lower bounds claimed by releasing a non-distant request first. $\frac{6p+4p(\lceil\frac{\upsilon-\lambda-4\tau}{4\tau}\rceil)+(2p-2c)(\lceil\frac{\upsilon-\lambda-6\tau}{4\tau}\rceil)}{2p+(2p-2c)(\lceil\frac{\upsilon-\lambda-4\tau}{4\tau}\rceil)}$ and $\frac{12p-2c}{3p-c}$. For what values of c each of these bounds is the lower of the bounds depends on the maximum difference between notification intervals. We first explain that releasing a traditional commitment gadget for these lengths of notification interval does not always result in the highest lower bound for the adversary. This is because for values of $\upsilon - \lambda > 8\tau$, an algorithm could react to the commitment gadget with a class B commitment. For a class B reaction, we find $P_{R^A} = 2p - c$ and $P_{R^*} = 6p$, hence $P_{R^*}/P_{R^A} = \frac{6p}{2p-c}$.. For these different sizes of notification intervals, we find that $\frac{6p+4p(\lceil\frac{\upsilon-\lambda-4\tau}{4\tau}\rceil)+(2p-2c)(\lceil\frac{\upsilon-\lambda-6\tau}{4\tau}\rceil)}{2p+(2p-2c)(\lceil\frac{\upsilon-\lambda-4\tau}{4\tau}\rceil)} > \frac{6p}{2p-c}$. Therefore, reaching a class B commitment gadget would not be good for the adversary. However, we still need to show that a class B commitment gadget might lead to a lower ratio, if the adversary releases requests prior to or after the commitment gadget. To show this, we look at potential requests released prior to or after the commitment gadget.

For requests starting prior to the commitment gadget, we note that we know that both of ALG's servers need to be at opposite locations before the start of the commitment gadget. Therefore, a request released from location 0 would first remove the empty movement needed to serve the distant request served during the commitment gadget. Thus, releasing such a request would increase the profit of ALG by $p + c$. This bypasses the strength of the commitment gadget, which forces ALG to accept requests that cause empty movements after being line-skipped. Thus, we know that the adversary cannot release requests starting from 0 without ALG gaining a lot of profit proportional to OPT.

Being able to release only requests starting from 1 prior to the commitment is also not very proportionally profitable. These requests are guaranteed to be distant requests for both OPT and ALG until a request is released from 0. Releasing only requests starting from 1 prior to the commitment asymptotically trends towards the $\min\{\frac{3p-c}{p-c}, 4\}$ bound we discussed in **Theorem 5.6**. That asymptote is not an improvement on $\frac{6p}{2p-c}$ for $0.5 < c$. Therefore, releasing requests prior to the commitment does not create a higher lower bound than the Class B commitment gadget.

For requests starting after the commitment gadget, we note that we know both of ALG's servers are at opposite locations immediately after the commitment gadget. Therefore, the first request released on either side is fully profitable to ALG. Furthermore, since both servers of ALG are at different locations, it is not possible to release a Zig or Zig-Zag gadget without giving ALG the option to accept two profitable requests. Furthermore, after those requests, ALG's servers are still at different locations. So, releasing a Zig or Zig-Zag gadget still would not work. Tirelessly releasing Zig-Zag gadgets in this situation has an asymptotic Lower bound of 3, lower than $\frac{6p}{2p-c}$. To reunite ALG's servers at the same location, an adversary has to release just one pair of requests. This action increases ALG's profit by p, whereas OPT's profit only increases by $2p$. Releasing just such a pair leads to a situation as if starting from time 0, but with $P_{R^A} = 3p - c$ and $P_{R^*} = 8p$, which is lower than the bounds we have seen starting from 0 profit. This shows us that the ratio provided by a Class B commitment gadget cannot be improved upon once the gadget has been accepted.

From there, we need to show that releasing a different stream of requests causes a slightly higher lower bound. For this, we release the following requests. First, the adversary releases $r_1 = r_2 = (0, \upsilon, 0)$. As always, ALG has three options. Either rejecting both of these requests, accepting one of them and rejecting the other, or accepting both requests. If both requests are rejected, OPT can accept both requests, and we find $P_{R^A} = 0$ and $P_{R^*} = 2p$, hence $P_{R^*}/P_{R^A} = \infty$. If both requests are accepted, the adversary is free to release a zig-zag gadget starting at $\upsilon - \epsilon$. This essentially causes a Class A commitment gadget. Meanwhile, if only one request is accepted by ALG, we release the following pair.

The adversary releases $r_3 = r_4 = (\epsilon, \upsilon - 2\epsilon, 0)$. Since one of ALG's servers is busy serving r_1 or r_2, ALG can either accept or reject one of these requests. If ALG accepts one of these requests, we can simplify to a Class A commitment gadget. If ALG rejects both of these requests, OPT accepts r_3 and r_4. We then wait for 2τ time to release the next requests.

The adversary releases $r_5 = r_6 = (2\tau, \upsilon + 2\tau - 2\epsilon, 0)$. Note that these requests are in conflict with r_1 and r_2. Since one of ALG's servers has served r_1 or r_2, ALG can accept at most one request from r_5 and r_6. If ALG accepts one of these requests, We can simplify to a Class A commitment gadget by releasing a Zig gadget starting at time $\upsilon + \tau - \epsilon$ at location 1. Meanwhile, if ALG rejects r_5 and r_6, OPT can accept these requests. Per our assumption that $\lambda < \tau$, we wait with releasing the next request until υ. Then, the adversary releases $r_7 = r_8 = (\upsilon, \upsilon + \tau - 2\epsilon, 1)$. These requests are not acceptable to ALG, but can be accepted by OPT. After this, the adversary can stop releasing requests. We find $P_{R^A} = p$ and $P_{R^*} = 6p$, hence $P_{R^*}/P_{R^A} = 6$. Since $6 > \frac{12p-2c}{3p-c}$ for all $0 \leq c \leq p$, and all other reactions are either non-competitive, or cause a Class A commitment gadget, we can limit our further reasoning to only Class A commitment gadgets. Note that this release schedule takes 2τ extra time out of $\upsilon - \lambda$ compared to the standard Class A commitment gadget.

The position of the servers after this alternative gadget is symmetrical to the position of the servers at time 0. Therefore, any release performed after the alternative gadget is discussed by showing possible releases from time 0. Therefore, we only need to find a strategy to fill the time between $2\tau + \lambda$ and υ as conveniently as possible. To this purpose, we formalize the line-skip gadget. We have used this strategy repeatedly in Theorem 5.7-5.10.

The line-skip gadget is constructed in the following way: the adversary first releases the alternative gadget, then releases a pair of requests $r_1 = r_2$ at $(x, y, 0)$, with $x > 2\tau$ and $2\tau + \lambda \leq y \leq \upsilon - 4\tau$. We distinguish three cases based on the reaction of the Algorithm.

Case 1: ALG accepts neither r_1 nor r_2. OPT accepts both of these requests. The adversary then waits to release $r_3 = r_4 = (y - \lambda + \tau, y + \tau, 1)$. Since neither of ALG's servers can be at location 1 in time to serve r_3 and r_4 if $\lambda < \tau$, these requests cannot be accepted by ALG. OPT accepts both r_3 and r_4. The adversary ends their releases here.
Case 2: ALG accepts either r_1 or r_2. OPT rejects both requests. The adversary releases $r_3 = r_4 = (x + \epsilon, y - 2\epsilon, 0)$. We distinguish two cases.
Case 2a: ALG rejects both r_3 and r_4. OPT accepts both requests. The adversary then releases $r_5 = r_6 = (x + 2\epsilon, y + 2\tau - 2\epsilon, 0)$. We distinguish two more cases.
Case 2a-a: ALG rejects both r_5 and r_6. OPT accepts both requests. The adversary then waits to release $r_7 = r_8 = (y - \lambda - 2\epsilon + \tau, y + \tau - 2\epsilon, 1)$. Since neither of ALG's servers can be at location 1 in time to serve r_7 and r_8 if $\lambda < \tau$, these requests cannot be accepted by ALG. OPT accepts both r_7 and r_8. The adversary ends their releases here.
Case 2a-b: ALG accepts either r_5 or r_6. OPT rejects both requests. The adversary releases a Zig-gadget starting at location 1 at time $y - \epsilon + \tau$. By Corollary 5.3, this Zig-gadget is in conflict with requests r_1 and r_5. Therefore, no part of this Zig gadget can be accepted by ALG. OPT fully accepts the Zig-gadget. The adversary ends their releases here.
Case 2b: ALG accepts either r_3 or r_4. OPT rejects both requests. The adversary releases a Zig-Zag gadget starting at location 0 at time $y - 3\epsilon$. By Lemma 5.2, ALG cannot accept any requests in this Zig-Zag gadget. OPT fully accepts the Zig-Zag gadget. This reaction mirrors the reaction used in Case 2a-b, changing a few ϵ counts. The adversary ends their releases here
Case 3: ALG accepts both r_1 and r_2. This reaction mirrors the reaction used in Case 2a-b, changing only a few ϵ counts. The results from Case 2a-b reflect the results achieved in this case. The adversary ends their releases here.

We notice three different general responses to the line-skip gadget. ALG can accept nothing, ALG can accept one request, or ALG can accept two requests. If the Algorithm accepts nothing, the adversary can react with a zig-gadget that ALG can no longer accept. If ALG accepts one request, the adversary reacts with a zig-zag gadget. After using that zig-zag gadget, both of OPT's servers are at location 1, whereas ALG has one server at location 0 and one server at location 1. However, to have OPT serve this particular zig-zag gadget, no further releases by the adversary can be made. Accepting an alternative gadget, then accepting a Case 2a-a line-skip gadget achieves a lower bound of $\frac{12p-2c}{3p-c}$.

If ALG accepts two requests, however, the adversary reacts with a zig-zag gadget. The adversary does not need to wait to release any requests for this zig-zag gadget. After using this zig-zag gadget, all servers of both OPT and ALG are at location 1. However, to serve another line-skip gadget, or to serve the alternative gadget, all servers need to be moved to location 0. This costs $2c$ for both OPT and ALG. That means that each line-skip gadget adds $6p - 2c$ to OPT's profits and $2p - 2c$ to ALG's profits. It takes 4τ time to serve a line-skip gadget. Therefore an adversary can force an algorithm using this reaction to line-skip gadgets to serve $\lfloor \frac{\upsilon-\lambda}{4\tau} \rfloor$ line-skip gadgets. Furthermore, if there is more than 2τ time between the end of the final line-skip gadget and the start of the alternative gadget, the adversary can release one more line-skip gadget, but only release a zig gadget for OPT to serve instead of a zig-zag gadget. This would add an additional $4p$ to OPT's profits at a cost of adding at most $2p - 2c$ to ALG's profits. Combining this with the 2τ time it takes to set up the alternative gadget explains the $\frac{6p+4p(\lceil \frac{\upsilon-\lambda-4\tau}{4\tau} \rceil)+(2p-2c)(\lceil \frac{\upsilon-\lambda-6\tau}{4\tau} \rceil)}{2p+(2p-2c)(\lceil \frac{\upsilon-\lambda-4\tau}{4\tau} \rceil)}$ part of the claimed lower bound.

This all shows that by releasing a distant request pair first, an adversary can achieve a lower bound of $\min\{\frac{3p-c}{p-c}, 4\}$, and that by releasing a non-distant request pair first, an adversary can achieve a lower bound of $\min\{\frac{6p+4p(\lceil \frac{\upsilon-\lambda-2\tau}{4\tau} \rceil)+(2p-2c)(\lceil \frac{\upsilon-\lambda-4\tau}{4\tau} \rceil)}{2p+(2p-2c)(\lceil \frac{\upsilon-\lambda-2\tau}{4\tau} \rceil)}, \frac{12p-2c}{3p-c}\}$. Therefore, the lower bound of 2S2L-V with $0 < c \leq p$ and $8\tau < \upsilon - \lambda$ is shown to be $\max\{\min\{\frac{3p-c}{p-c}, 4\}, \min\{\frac{6p+4p(\lceil \frac{\upsilon-\lambda-4\tau}{4\tau} \rceil)+(2p-2c)(\lceil \frac{\upsilon-\lambda-6\tau}{4\tau} \rceil)}{2p+(2p-2c)(\lceil \frac{\upsilon-\lambda-4\tau}{4\tau} \rceil)}, \frac{12p-2c}{3p-c}\}\}$
□

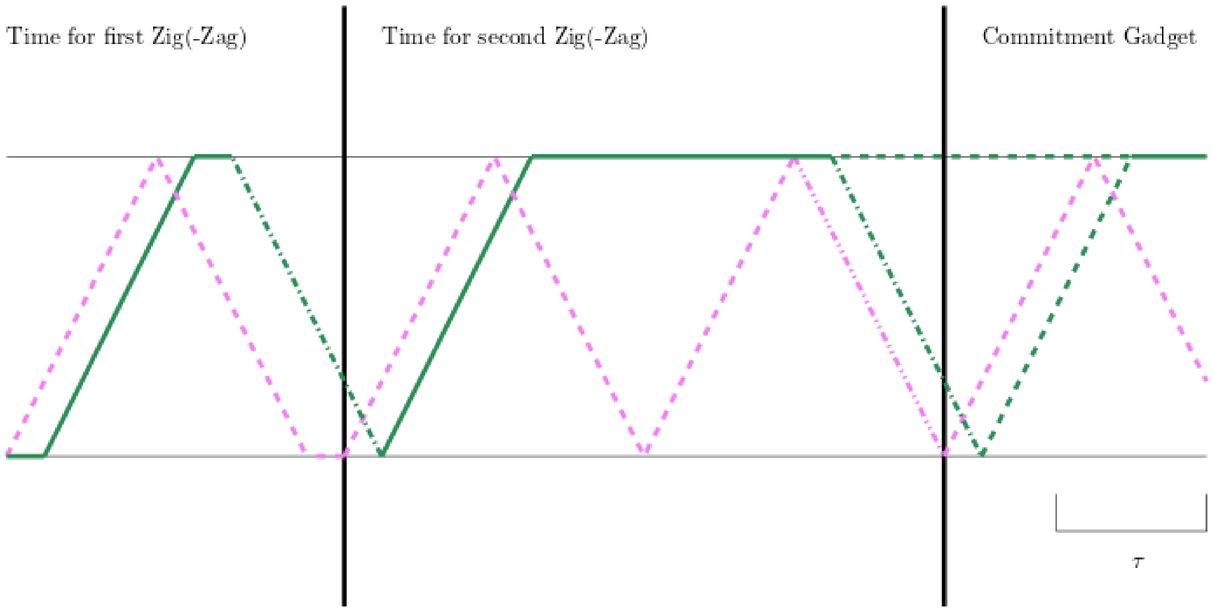


Figure 9: An example of the flow of proofs 5.8-5.11. The location of the online algorithm (in green) and the optimum algorithm (in pink, dotted) are shown over time, as is the length of a trip (τ). If the green line is dashed, that means the two servers are not at the same location at that time. If a line is both dotted and dashed, it indicates an empty movement performed by a number of servers equal to the amount of dots. We subdivided the total time-location line into possible gadgets, to help ease the understanding.

6 Previous Algorithms

In this chapter, we analyze the performance lower bounds of two algorithms created for car sharing on 2S2L. These algorithms are a simple Greedy Algorithm that can be run on (nearly) all settings for car sharing, and Smart Greedy by K. Luo et al., designed for 2S2L-F [13]. Smart Greedy was optimal for 2S2L-F. We find that Smart Greedy needs to be redesigned for 2S2L-V. A simple redesign is not enough to make Smart Greedy very competitive on 2S2L-V, however. It performs worse than Greedy on some settings for c and $[\upsilon, \lambda]$. Even when limited to our special instance that releases a paired request for every request in the instance, Greedy and Smart Greedy do not perform well. The sub-optimal performance of these previous algorithms on this problem informs us that a new algorithm is needed to achieve a respectable competitive ratio on 2S2L-V.

6.1 The Greedy Algorithm

In this first section, we discuss the strengths and weaknesses of *The Greedy Algorithm* (GA). This algorithm is conceptually simple, and can be described as follows. If a request seems acceptable at the booking time, it is accepted. Otherwise, the Greedy Algorithm rejects the request. Keep in mind that this means that profit is not a consideration for GA when deciding whether or not to accept a request. We formally define Greedy Algorithm as follows:

Algorithm 1 Greedy Algorithm ($GA(\mathcal{I})$)

```
Input: Instance $\mathcal{I}$ of requests $r_i$ arriving over time with $\upsilon \geq t_{r_i} - b_{r_i} \geq \lambda$.
Output: Series of accepted requests $R^A$
begin
    $R^A = \emptyset$
    foreach $r \in \mathcal{I}$ do
        if $r$ is acceptable then
            $R^A = R^A \cup r$
        else
            Reject $r$
        end
    end
end
```

This greedy approach ensures that the servers of the algorithm are as busy as they can be, without violating acceptability. This means that the set of requests accepted by GA can always be transformed into a feasible schedule. Note that no acceptable request is ever declined, even if there is no profit to be gained by serving them. That is both a benefit and a detriment to the lower bound achieved by this approach.

We need only differentiate between three settings to discuss all lower bounds of this algorithm. One of the reasons why this is the case, is because GA makes no exceptions for requests with 0 profit. All requests are only checked based on acceptability. That ensures that the setting of c is not relevant to GA. Therefore, we need only consider various settings of $[\upsilon, \lambda]$. For these settings, it is important to differentiate the settings where distant requests are possible from those where they are not. Furthermore, we need to differentiate between the settings where distant requests can be line-skipped and where they cannot. This explains the three settings for which we prove the lower bounds.

	$0 \leq c \leq p$	Theorem
Constraint	LB	
$0 < \upsilon < \tau$	3	Theorem 6.1
$\upsilon = \tau$	$\max\{\frac{2p}{p-c}, 3\}$	Theorem 6.2
$\tau < \upsilon - \lambda$	$\frac{3p-c}{p-c}$	Theorem 6.3

Table 5: The results of the lower bounds of Algorithm GA on 2S2L-V

We prove the results shown in this table by providing an adversary that releases an instance on which GA does not perform better than the claimed lower bound. Because GA's reactions are deterministic, we need only show one reaction of an adversary to show the claimed lower bound.

Theorem 6.1 For $0 < \upsilon < \tau$, Algorithm GA cannot achieve a competitive ratio of lower than 3.

Proof We prove this theorem by providing an adversary. This adversary first releases $r_1 = r_2 = (0, \upsilon, 0)$. By the workings of GA, GA accepts these requests. The adversary then releases a zig-zag gadget at location 0 starting at time $\upsilon - \epsilon$. By Lemma 5.2, all requests in this gadget are in conflict with r_1 and r_2. Therefore, the gadget is not acceptable to GA, and GA thus rejects it. OPT accepts all requests in the gadget. We find $P_{R^A} = 2p$ and $P_{R^*} = 6p$, hence $P_{R^*}/P_{R^A} = 3$. $\square$

If this lower bound is tight, we find that Algorithm GA is optimal for the settings $0 < c \leq p$ and $0 < \upsilon \leq \tau$. We now investigate the lower bound achieved by GA on the settings of $\upsilon = \tau$

Theorem 6.2 For $\upsilon = \tau$, Algorithm GA cannot achieve a competitive ratio of lower than $\max\{\frac{2p}{p-c}, 3\}$.

Proof We prove this theorem by providing two adversaries. The first of these adversaries is the one discussed in **Theorem 6.1**. This adversary is used when $c < \frac{p}{3}$, because $3 > \frac{2p}{p-c}$ for those settings of c. When $c \geq \frac{p}{3}$, we use a different adversary. This adversary first releases $r_1 = r_2 = (0, \upsilon, 1)$. By the workings of GA, GA accepts these requests. The adversary then releases a zig gadget at location 0 starting at time $\upsilon - \epsilon$. By Lemma 5.1, all requests in this gadget are in conflict with r_1 and r_2. Therefore, all requests in the gadget are not acceptable to GA, and GA thus rejects them. OPT accepts all requests in the gadget. We find $P_{R^A} = 2p - 2c$ and $P_{R^*} = 4p$, hence $P_{R^*}/P_{R^A} = \frac{2p}{p-c}$. $\square$

If this lower bound is tight, we find that Algorithm GA is optimal for the settings $0 < c \leq \frac{p}{3}$ and $\upsilon = \tau$. That means its utility on this setting is limited. This is the case because its reaction to the pair of distant requests is not optimal.

Finally, we investigate the performance lower bound of GA on all other settings of 2S2L-V.

Theorem 6.3 For $\tau < \upsilon - \lambda$, Algorithm GA cannot achieve a competitive ratio of lower than $\frac{3p-c}{p-c}$.

Proof We prove this theorem by providing an adversary. This adversary first releases $r_1 = r_2 = (0, \upsilon, 1)$. By the workings of GA, GA accepts these requests. The adversary then releases a zig-zag gadget at location 1 starting at time $\upsilon - \epsilon$. By Lemma 5.2, all requests in this gadget are in conflict with r_1 and r_2. Therefore, the gadget is not acceptable to GA, and GA thus rejects it. OPT accepts all requests in the gadget. We find $P_{R^A} = 2p - 2c$ and $P_{R^*} = 6p$, hence $P_{R^*}/P_{R^A} = \frac{3p-c}{p-c}$. $\square$

If this lower bound is tight, we find that Algorithm GA is optimal for the settings $0 < c \leq \frac{p}{3}$ and $\upsilon - \lambda > \tau$.

Despite GA seeming without merit on the setting with $\upsilon = \tau$, it seems that this algorithm performs quite well, taking into account its relative simplicity. Even when booking times get very long, if the cost of an empty movement is relatively low, we find that GA performs well. If these bounds prove tight, we can consider using GA as a basis for an algorithm dealing with low cost settings for 2S2L-V. The downside of GA, however, is that it is extremely poor on settings with high empty movement cost. In those settings, GA accepts distant requests that are barely profitable, making way for OPT to accept requests that are much more profitable. For c close to p, the lower bound of this algorithm trends toward infinity. That means we definitely need different algorithms for settings with high empty movement costs.

6.2 Smart Greedy

The results we found on GA imply that we need to take more care to not accept requests with expensive empty movement costs too often. To this end, we look at an improvement upon Greedy for 2S2L-F created by K. Luo et al [13]. Smart Greedy might perform well on 2S2L-V, or at least provide inspiration for new algorithms. The reasoning leading to Smart Greedy is admirable. Serving a distant request is often a risky endeavor. Accepting two distant requests that start close to one another, an adversary could make use of the time that an algorithm has to take to serve these distant requests. In this time, the adversary could release two pairs of non-distant requests, in conflict with the requests accepted by the algorithm. In order to avoid this, Smart greedy was created with the following three main rules. Firstly, if a request does not require an empty move to be served, it is accepted. Secondly, if a request requires an empty move, and neither server is busy at or after the time of the empty movement, accept this distant request.

Finally, if neither of these two conditions are met, reject the incoming request.

The implementation that K. Luo et al provided for Smart Greedy was geared towards 2S2L-F. It checked whether adding a request to the set of accepted requests would increase profit by p to check whether a request was distant or not. This might not be accurate for 2S2L-V. Therefore, Smart Greedy needs to be slightly refactored for 2S2L-V. Note that this is an algorithm that checks profitability of requests alongside acceptability. This should ensure a better lower bound than Greedy provided. After a simple refactor, we find our new definition of Smart Greedy to be as follows.

Algorithm 2 Smart Greedy ($SG(\mathcal{I})$)

```
Input: Instance $\mathcal{I}$ of requests $r_i$ arriving over time with $\upsilon \geq t_{r_i} - b_{r_i} \geq \lambda$.
Output: Series of accepted requests $R^A$
begin
    $R^A = \emptyset$
    foreach $r \in \mathcal{I}$ do
        if $r$ is acceptable and $P_{R^A \cup r} - P_{R^A} \geq p$; then
            $R^A = R^A \cup r$
        else if $r$ is acceptable and $P_{R^A \cup r} - P_{R^A} > 0$ and $\nexists r_j \in R^A$ such that $t_r - \bar{t}_{r_j} < \tau$ then
            $R^A = R^A \cup r$
        else
            Reject $r$
        end
    end
end
```

Note that Smart Greedy creates a set of requests that can be converted into a viable schedule. This is the case because it explicitly mandates all accepted requests to be acceptable. Since requests are only acceptable if they fit into a viable schedule, we conclude that the set of requests accepted by Smart Greedy only includes requests that can be served in a schedule.

Next, we show the lower bounds of Smart Greedy. Note that this algorithm takes profit into account when deciding whether to accept a request or not. Therefore, the amount of lower bounds that can be proven with one proof is lower. Hence, we have more lower bounds we need to prove. The lower bounds we prove are as follows:

	$0 < c < p$	Theorem
Constraint	LB	
$0 < \upsilon < \tau$	3	Theorem 6.4
$\upsilon = \tau$	$\max\{\frac{4p}{2p-c}, 3\}$	Theorem 6.5
$\tau < \upsilon - \lambda \leq 2\tau$	4	Theorem 6.6
$2\tau < \upsilon - \lambda$	$2\lfloor\frac{\upsilon-\lambda}{2\tau}\rfloor + 3$	Theorem 6.4

Table 6: The results of the lower bounds of Algorithm GA on 2S2L-V with $0 < c < p$

$c = p$	Theorem
LB	
$2\lfloor\frac{\upsilon-\lambda}{2\tau}\rfloor + 3$	Theorem 6.4

Table 7: The results of the lower bounds of Algorithm GA on 2S2L-V with $c = p$

We prove the results shown in these tables by providing an adversary that releases an instance on which SG does not perform better than the claimed lower bound. Because SG's reactions are deterministic, we need only show one reaction of an adversary to prove the claimed lower bound.

Theorem 6.4 On the 2S2L-V problem, Algorithm GA cannot achieve a competitive ratio of lower than $2\lfloor\frac{\upsilon-\lambda}{2\tau}\rfloor + 3$.

Proof We prove this theorem by providing an adversary. This adversary first releases $r_1 = r_2 = (0, \upsilon, 0)$. By the workings of SG, SG accepts these requests. With r_1 and r_2 accepted, we show that SG accepts no requests r with $t_r < \upsilon$, regardless of starting location.

Assume the release of non-distant request $r_n = (\epsilon, a, 0)$, with $a < v - 2\tau$ such that r_n is acceptable to SG. If r_n were not acceptable, the request would be rejected, so we need not continue that line of reasoning. Observe that accepting r_n would require an empty movement to serve r_1. Thus, $P_{R^A \cup r_n} - P_{R^A} = p - c < p$. For non-distant request r_n to be accepted by SG, SG requires that, for all requests r_j in R^A, $t_{r_n} - \bar{t}_{r_j} \geq \tau$. However, because $t_{r_1} > t_{r_n}$, we already find that $t_{r_n} - \bar{t}_{r_1} < \tau$. Therefore, any non-distant request with starting time lower than t_{r_1} would be rejected.

Next, assume the release of distant request $r_d = (\epsilon, a, 1)$, with $\tau < a < v - \tau$ such that r_d is acceptable to SG. If r_d were not acceptable, the request would be rejected, so we need not continue that line of reasoning. Observe that accepting r_d would require an empty movement. Thus, $P_{R^A \cup r_d} - P_{R^A} = p - c < p$. For distant request r_d to be accepted by SG, SG requires that, for all requests r_j in r^A, $t_{r_d} - \bar{t}_{r_j} \geq \tau$. However, because $t_{r_1} > t_{r_d}$, we already find that $t_{r_d} - \bar{t}_{r_1} < \tau$. Therefore, any non-distant request with starting time lower than r_1 would be rejected.

Having proven that any request with $t_r < t_{r_1}$ is rejected by SG, we observe that an adversary is free to use the time between λ and $t_{r_1} = v$. The limit on the profit an optimal algorithm can make over SG during this time is found by the size of the booking interval $v - \lambda$, and the time it takes to serve requests τ. The adversary can release Zig-gadgets starting at location 0 on an interval of 2τ, starting at λ. These gadgets are rejected by SG, whereas OPT accepts all of them without conflict. This grants OPT a profit of $4p$ for every 2τ in $v - \lambda$. Following just the release of the zig-gadgets in the pattern described, OPT would make $4\lfloor \frac{v-\lambda}{2\tau} \rfloor$ profit.

An adversary can still implement an improvement over the $4\lfloor \frac{v-\lambda}{2\tau} \rfloor$ profit. To this purpose, we focus on the timeframe of $[v - 2\tau, v]$. Because Zig-gadgets are started every 2τ during the $[\lambda, v]$, we know that one such Zig-gadget must end during this timeframe. We now discuss what should be done with the remainder of this timeframe. We propose the release of a Zig-Zag gadget directly after the end of the Zig-gadget ending in the $[v - 2\tau, v]$ timeframe. By Lemma 5.2, all requests in this gadget conflict with r_1 and r_2. Therefore, these requests do not abide by the acceptability constraint set by SG, and are thus rejected. This Zig-Zag gadget is subsequently accepted by OPT. After the Zig-Zag gadget, however, both OPT and SG's servers are at location 1, and SG will start accepting requests again. After all the profit OPT has made over SG, we find $P_{R^A} = 2p$ and $P_{R^*} = 4p\lfloor \frac{v-\lambda}{2\tau} \rfloor + 6p$, hence $P_{R^*}/P_{R^A} = 2\lfloor \frac{v-\lambda}{2\tau} \rfloor + 3$. □

Theorem 6.4 has proven that there is a gap between the proven problem lower bound for 2S2L-V, and the lower bound achieved by Smart Greedy. For any setting with $v - \lambda > 2\tau$, we find that Smart Greedy's performance is subpar compared to the problem lower bounds. For $v < \tau$, we note that there is no request that is acceptable to SG and has a profit lower than p. Therefore, for those settings, SG's performance is the same as GA's. This leaves the settings $2\tau \geq v - \lambda \geq \tau$ open for further exploration.

Theorem 6.5 For $v - \lambda = \tau$ and $0 < c < p$ Algorithm GA cannot achieve a competitive ratio of lower than $\max\{\frac{4p}{2p-c}, 3\}$.

Proof First note that the Lower Bound of this problem is decided by the value of c. The adversary can decide whether it pursues the lower bound of $\frac{4p}{2p-c}$ or the lower bound of 3, depending on the value of c. For $c \leq \frac{2p}{3}$, the adversary rather lures SG to the bound of 3. It can achieve this by following the release schedule laid out in **Theorem 6.4** or, equivalently, **Theorem 6.1**. For the lower bound of $\frac{4p}{2p-c}$ we need to provide a different adversary.

We prove this theorem by providing an adversary. This adversary first releases $r_1 = r_2 = (0, \tau, 1)$. By the workings of SG, SG accepts one of these requests and rejects the other because these requests provide less than p profit. The adversary then releases $r_3 = r_4 = (\epsilon, \lambda + 3\epsilon, 0)$. By the workings of SG, SG accepts one of these requests (without loss of generality, we assume this to be r_3), needing to reject the other because it conflicts with r_1 and r_3. Thus, r_4 is not acceptable to SG. Finally, the adversary releases a Zig-gadget starting at time $2\epsilon + \lambda$ at location 0. By Lemma 5.1 and Corollary 5.3, these requests are in conflict with r_1 and r_3. Therefore, the gadget is not acceptable to SG, and SG thus rejects it. OPT accepts all of the requests in this gadget. We find $P_{R^A} = 2p - c$ and $P_{R^*} = 4p$, hence $P_{R^*}/P_{R^A} = \frac{4p}{2p-c}$. □

If this Lower Bound proves tight, Smart Greedy would prove optimal for the very specific setting of $0 < c < p$ and $v = \tau$ for 2S2L-V.

Theorem 6.6 For $\tau < v - \lambda \leq 2\tau$ and $0 < c < p$, Algorithm SG cannot achieve a competitive ratio of lower than 4.

Proof We prove this theorem by providing an adversary. This adversary first releases $r_1 = r_2 = (0, \upsilon, 1)$. By the workings of GA, GA accepts these requests. The adversary then releases $r_3 = r_4 = (\epsilon, \tau + \epsilon, 1)$ and $r_5 = r_6 = (2\tau, 3\tau + \epsilon, 1)$. We observe that both $t_{r_3} - \bar{t}_{r_1} < \tau$ and $t_{r_5} - \bar{t}_{r_1} < \tau$. We also note that r_3, r_4, r_5 and r_6 require an empty movement to be served by SG. Therefore, their profit is $p - c$. This means, by the workings of SG, that r_3 through r_6 are rejected by SG. OPT accepts all four of these requests. We find $P_{R^A} = p - c$ and $P_{R^*} = 4p - 4c$, hence $P_{R^*}/P_{R^A} = 4$. $\square$

We observe that Smart Greedy's decisions regarding requests to be served with empty movement do not work on 2S2L-V. This is because Smart Greedy was not designed with line-skipping in mind. Requests that line-skip a pair of non-distant requests always require an empty movement to be served. Since SG rejects any requests with profit lower than p, unless it is the final known request, it rejects nearly all line-skipping requests. This often means that SG performs worse than even GA.

We have seen in this chapter that thoughtlessly accepting requests without scrutinizing costs does not create an algorithm that can achieve the problem lower bounds. Similarly, rejecting requests purely based on their profit does not achieve the 2S2L-V problem Lower Bounds, either. We need a different algorithm that does not falter when empty movements become prominently involved.

7 The Exclusionary family

In this chapter, we discuss the competitive ratio upper bounds of *the Exclusionary Family* of algorithms (to be defined). We first observe properties of the family, and then investigate the performance of various algorithms inside this family on the 2S2L-V problem. We vary the following settings in the 2S2L-V problem: maximum notification interval υ, minimum notification interval λ and cost of empty movement c. As we have done throughout, we focus on a special instance that contains only of paired requests.

7.1 The Exclusionary Family

In this section, we define a family of Algorithms called the *Exclusionary Family* F_E. All of the algorithms we discuss in this chapter are a part of this family. We prove that these algorithms perform well on all possible paired 2S2L-V instances. We first prove a set of properties over the entire family that will be reused throughout this chapter.

The idea behind these algorithms is as follows: the number of servers available to any algorithm for 2S2L-V is fixed. In order to serve any request, one server has to spend time performing the movements required to get to the starting location, then serve the request. This means that having a server serve a request blocks that server from serving different request with a similar starting time. If both servers are busy serving requests at around the same time, it is be impossible to serve any requests starting at close to that time.

To formalize this idea, we introduce the notion of *location-time pairs*. This notation is meant to succinctly describe a location at a certain time. We denote a location-time pair as follows: (l, t). In this notation, l is the location we indicate, and t is the time. To describe a location over a longer stretch of time, we introduce *location-time intervals*: $(l, (x, y))$. In essence, this is the contiguous set of location-time pairs with the same location, starting from the pair at (l, x), ending at the pair at (l, y), including all pairs in between. We measure the size of a location-time interval $|(l, (x, y))|$ by the size of the time-interval. In other words, $|(l, (x, y))| = y - x$. All this notation will simplify other constructs that we need to define the Exclusionary Family.

Using these location-time intervals, we define *Exclusion Zones*, and upon those Exclusion Zones, we define the *Exclusionary Family*. Exclusion Zones are a continually updated set of location-time intervals, constructed in accordance with some deterministic *Method* M. These location-time intervals that comprise the Exclusion Zones are used to specify when an algorithm in the Exclusionary Family accepts or rejects a request. If a request $r = (b_r, t_r, l_r)$ is released in such a way that (l_r, t_r) is in the set of Exclusion Zones as constructed by Method M at time b_r, this request is rejected by the algorithm that is part of the Exclusionary Family and follows this set of Exclusion Zones. If request r is not in the Exclusion Zones at b_r, the request is accepted. The Method M changes per algorithm in the Exclusionary Family. Two different algorithms in the Exclusionary Family may very well have two very different sets of Exclusion Zones when ran on the same instance. The set of Exclusion Zones changes over time.

We differentiate three different notations for Exclusion Zones. The first notation $EZ_M(b)$, denotes the set of Exclusion Zones as constructed by Method M at time b on the timeline, with time b not included. This means any request with release time b is not considered when constructing the set of Exclusion Zones at time b. Note that time b is compared to booking times (which is a continuously increasing value) rather than starting times (which vary from request to request). Meanwhile, $EZ_M(r-)$ denotes the set of Exclusion Zones constructed by Method M immediately before processing request r. Very specifically, $EZ_M(r-)$ is constructed at b_r right before deciding whether request r is accepted or rejected. Finally, $EZ_M(r+)$ denotes the set of Exclusion Zones constructed by Method M immediately after processing request r.

Definition 7.1 Method $(M(R, b))$
A deterministic algorithm M that takes a set of requests R with $b_r \leq b \ \forall r = (b_r, t_r, l_r) \in R$ as input, and processes them to create a set of Exclusion Zones.

Definition 7.2.1 Exclusion Zones $(EZ_M(b))$
A set of location-time intervals $EZ_M(b) = \{(l_1, (x_1, y_1)), (l_2, (x_2, y_2)), ..., (l_n, (x_n, y_n))\}$ constructed by a Method M at time b on the timeline.

Definition 7.2.2 Exclusion Zones ($EZ_M(r-)$)
The set of location-time intervals $EZ_M(r-) = \{(l_1, (x_1, y_1)), (l_2, (x_2, y_2)), ..., (l_n, (x_n, y_n))\}$ constructed by a method M immediately before request r has been processed by an algorithm in the Exclusionary Family.

Definition 7.2.3 Exclusion Zones ($EZ_M(r+)$)
The set of location-time intervals $EZ_M(r+) = \{(l_1, (x_1, y_1)), (l_2, (x_2, y_2)), ..., (l_n, (x_n, y_n))\}$ constructed by a method M immediately after request r has been processed by an algorithm in the Exclusionary Family.

Having defined Methods and Exclusion Zones, we now define the Exclusionary Framework. Adherence to this framework is the sole requirement to test whether an algorithm A is part of the Exclusionary Family or not.

Definition 7.3 Exclusionary Framework ($EF(\mathcal{I}, M(R^A, b))$)
Given an instance $\mathcal{I}$ of requests r arriving over time by increasing b_r, with $\upsilon \geq t_r - b_r \geq \lambda \; \forall r \in \mathcal{I}$ and a method $M(R^A, b)$ to construct $EZ(b)$ for any time b, over an accepted set of requests R^A, the Exclusionary Framework mandates that that the following all hold:

1. At $t = 0$, $R^A = \emptyset$
2. For each $r = (b_r, t_r, l_r) \in \mathcal{I}$:
    3. $EZ_M(r-) = M(R^A, b_r)$
    4. If $(l_r, t_r) \notin EZ_M(r-)$: $R^A = R^A \cup r$
    5. Else: reject r

We now define our Exclusionary Family based on this Exclusionary Framework.

Definition 7.4 Exclusionary Family (F_E)
Let $\mathcal{I}$ be an instance of requests $r = (b_r, t_r, l_r)$ arriving over time.
Let $M(R, b)$ be a method to construct $EZ_M(b)$.
Let $A(\mathcal{I})$ be an algorithm producing a set of requests $R^A \in \mathcal{I}$ that can be feasibly scheduled.
$A \in F_E$ if and only if there exists a Method $M(R, b)$ such that $EF(\mathcal{I}, M(R, b)) = A(\mathcal{I}) \; \forall \mathcal{I}$.

Note that there is a one-to-one relation between an algorithm in the Exclusionary Family, and a Method. Each algorithm in the Exclusionary Family has a Method that describes the Exclusion Zones the algorithm uses. Similarly, every Method has a corresponding algorithm that is a part of the Exclusionary Family.

Now that the Exclusionary Family has been defined, we expand on some of the properties of algorithms in this family. These properties are helpful in proving the competitive ratio of the algorithms in the Exclusionary Family. We derive these properties using details from Definitions 7.1-7.4.

We observe the following properties based on the definition of F_E.

Observation 7.5 Any Algorithm A in F_E accepts requests greedily with regards to its Exclusion Zones.

Proof By definition 7.4, any $A \in F_E$ has a formulation that adheres to the Exclusionary Framework laid out in Definition 7.3. Definition 7.3 defines only one check that decides whether a request is accepted or rejected. This check regards only whether a request is inside the Exclusion Zones or not. Therefore, we can safely say that any Algorithm A in F_E accepts requests greedily with regards to its Exclusion Zones. □

Observation 7.6 For any two conflicting requests r, j accepted by any algorithm $A \in E_F$ that accepts requests that can feasibly be scheduled, any set of exclusion zones $EZ_M(b)$ with $b > \max\{b_r, b_j\}$ includes the following location-time intervals:
If $l_r = l_j$, $[\max\{t_r, t_j\} - 2\tau, \min\{t_r, t_j\} + 2\tau]$ at l_r and $[\max\{t_r, t_j\} - \tau, \min\{t_r, t_j\} + \tau]$ at $\bar{l}_r$, **or**
If $l_r \neq l_j$, $[t_j - \tau, t_j + \tau]$ at l_r and $[t_r - \tau, t_r + \tau]$ at l_j.

Proof By Observation 7.5, we know any algorithm A accepts requests greedily with regards to its Exclusion Zones. Thus, any requests outside the Exclusion Zones will be accepted by A. If requests r

and j are in conflict, that means both servers must serve one of these requests. Therefore, this means both servers are unavailable for some time. During that time, neither server can serve a different request. Requests released during that time should, thus, be rejected. Requests are only rejected if they are inside the set of Exclusion Zones present during their booking time. Therefore, any request released at a time in conflict with r and j should be inside an Exclusion Zone.

To support our further specification, we investigate for which location-time intervals $(l,(x,y))$ requests $k=(b_k,t_k,l_k)$ with $l_k=l$ and $x<t_k<y$ are in conflict with both r and j. For that, we need the definition of conflicting requests. Formally, requests i and j with $t \le t_j$ are in conflict if $t_j < \bar{t}+\tau(\bar{l},l_j)$. Recall that $\bar{t}=t+\tau$. This means, request k is in conflict with r if $t_r \le t_k < t_r+\tau+\tau(\bar{l}_r,l_k)$ or $t_k \le t_r < t_k+\tau+\tau(\bar{l}_k,l_r)$. In other words: if $\max\{t_r,t_k\}-\min\{t_r,t_k\} < \tau+\tau(\bar{l}_k,l_r)$, r and k are in conflict. Similarly, j conflicts with k if $\max\{t_j,t_k\}-\min\{t_j,t_k\}<\tau+\tau(\bar{l}_k,l_j)$, j and k are in conflict. And, since r and j are in conflict, we know $\max\{t_r,t_j\}-\min\{t_r,t_j\}<\tau+\tau(\bar{l}_r,l_j)$.

Case distinction

If $l_r=l_j=l_k$, we find $\max\{t_r,t_j\}-\min\{t_r,t_j\}<2\tau$, $\max\{t_r,t_k\}-\min\{t_r,t_k\}<2\tau$ and $\max\{t_j,t_k\}-\min\{t_j,t_k\}<2\tau$. In other words, for $l_r=l_j=l_k$ we find a conflict for any t_k such that $|t_r-t_k|<2\tau$ and $|t_j-t_k|<2\tau$. This holds for $t_k>\max\{t_r,t_j\}-2\tau$ if $t_k<\min\{t_r,t_j\}$, for $t_k<\min\{t_r,t_j\}+2\tau$ if $t_k>\max\{t_r,t_j\}$ and both, for any $\min\{t_r,t_j\}\le t_k\le\max\{t_r,t_j\}$. Concluding, any $k=(b_k,t_k,l_k)$ conflicts with any r,j in conflict for $l_r=l_j=l_k$ if $t_k\in[\max\{t_r,t_j\}-2\tau,\min\{t_r,t_j\}+2\tau]$

If $l_r=l_j=\bar{l}_k$, we find $\max\{t_r,t_j\}-\min\{t_r,t_j\}<2\tau$, $\max\{t_r,t_k\}-\min\{t_r,t_k\}<\tau$ and $\max\{t_j,t_k\}-\min\{t_j,t_k\}<\tau$. In other words, for $l_r=l_j=\bar{l}_k$ we find a conflict for any t_k such that $|t_r-t_k|<\tau$ and $|t_j-t_k|<\tau$. This holds for $t_k>\max\{t_r,t_j\}-\tau$ if $t_k<\min\{t_r,t_j\}$, for $t_k<\min\{t_r,t_j\}+\tau$ if $t_k>\max\{t_r,t_j\}$ and both, for any $\min\{t_r,t_j\}\le t_k\le\max\{t_r,t_j\}$. Concluding, any $k=(b_k,t_k,l_k)$ conflicts with any r,j in conflict for $l_r=l_j=\bar{l}_k$ if $t_k\in[\max\{t_r,t_j\}-\tau,\min\{t_r,t_j\}+\tau]$

If $l_r=\bar{l}_j=l_k$, we find $\max\{t_r,t_j\}-\min\{t_r,t_j\}<\tau$, $\max\{t_r,t_k\}-\min\{t_r,t_k\}<2\tau$ and $\max\{t_j,t_k\}-\min\{t_j,t_k\}<\tau$. In other words, for $l_r=\bar{l}_j=l_k$ we find a conflict for any t_k such that $|t_r-t_k|<2\tau$ and $|t_j-t_k|<\tau$. Since $|t_r-t_j|<\tau$, we note that any k in conflict with j also conflicts with r. Therefore, any $k=(b_k,t_k,l_k)$ conflicts with any r,j in conflict for $l_r=\bar{l}_j=l_k$ if $t_k\in[t_j-\tau,t_j+\tau]$

If $l_r=\bar{l}_j=\bar{l}_k$, we find $\max\{t_r,t_j\}-\min\{t_r,t_j\}<\tau$, $\max\{t_r,t_k\}-\min\{t_r,t_k\}<\tau$ and $\max\{t_j,t_k\}-\min\{t_j,t_k\}<2\tau$. In other words, for $l_r=\bar{l}_j=\bar{l}_k$ we find a conflict for any t_k such that $|t_r-t_k|<\tau$ and $|t_j-t_k|<2\tau$. Since $|t_r-t_j|<\tau$, we note that any k in conflict with r also conflicts with j. Therefore, any $k=(b_k,t_k,l_k)$ conflicts with any r,j in conflict for $l_r=\bar{l}_j=\bar{l}_k$ if $t_k\in[t_r-\tau,t_r+\tau]$
□

Observation 7.7 Any request r in instance $\mathcal{I}$, not accepted by Algorithm $A\in F_E$ does not change $EZ_M(b)$ for any time b.

Proof We have seen in Definition 7.3 that a Method $M(R^A,b)$ elaborating Exclusion Zones receives only the requests accepted by A and time b when setting up the Exclusion Zones. Therefore, any request not in R^A does not have any influence on the creation of Exclusion Zones by the Method. □

Corollary 7.7.1 For any request r in instance $\mathcal{I}$, not accepted by Algorithm $A\in F_E$ we find $EZ_M(r-)=EZ_M(r+)$.

Using these properties, we investigate a Lemma that shows the power of the Exclusion Zone approach. Specifically, this lemma limits the amount of time an adversary has that it can use to make profit over $A\in F_E$.

Lemma 7.8 For any instance over paired requests $\mathcal{I}$, for any Method $M(R,b)$ dictating the Exclusion Zones for Algorithm $A\in E_F$, any request $r=(b_r,t_r,l_r)\in\mathcal{I}$, has $(l_r,t_r)\in EZ_M(r'+)$, in which r' is the paired request of r.

Proof We prove this Lemma by contradiction. Assume, for that contradiction, that there exists a request $r=(b_r,t_r,l_r)$ with $(l_r,t_r)\notin EZ_M(r'+)$. Recall the assumption that request r is handled prior to request r'. By Definition 7.2.3, request r and r' have been processed by A. Here, we distinguish two cases: either A accepts both r and r', or A rejects r'.

For the first case, where both r and r' are accepted by A, we note that r and r' are in conflict with each other. By Observation 7.6, this implies an Exclusion Zone at $(l_r,[t_r-2\tau,t_r+2\tau])$. This interval clearly includes (l_r, t_r). This contradicts our assumption that $(t_r, l_r) \notin EZ_M(r'+)$.

In the second case, A does not accept r'. Recall Observation 7.5 stating that any algorithm in F_E accepts requests greedily with regards to its Exclusion Zones. If r' is rejected, we find $(l_{r'}, t_{r'}) \in EZ_M(r'-)$. By Corollary 7.7.1, we then find $EZ_M(r'+) = EZ_M(r'-)$. Since $r' = r$, we find $(l_r, t_r) \in EZ_M(r'+)$, once again contradicting our assumption that $(t_r, l_r) \notin EZ_M(r'+)$. This proves a contradiction in our assumption, showing that any request $r = (b_r, t_r, l_r) \in \mathcal{I}$, has $(l_r, t_r) \in EZ_M(r'+)$, in which r' is the paired request of r. $\square$

Corollary 7.8.1 OPT only accepts requests with $(l_r, t_r) \in EZ_M(r'+)$.

Next, we further delve into the exact reasons that could cause location-time intervals $(l, (x, y))$ to be part of Exclusion Zones. We subdivide these reasons into three categories by these three cases: Either a Method excludes a certain location-time interval from the outset, or a Method excludes certain location-time intervals because it is impossible to serve such a request given the current or predicted locations of the servers, or a Method excludes location-time intervals because they would cause conflicts with already accepted requests.

The first category of Exclusion Zone are the location-time intervals rejected from the outset. The Exclusionary Family includes Algorithms that reject certain location-time pairs per their definition. For instance, consider an algorithm rejecting any request r with $l_r = 0$ and $50 < t_r < 100$. This condition can easily be included in a Method. Despite this being unwanted behaviour, we quickly discuss this case for completeness. Therefore, we define a subfamily of the Exclusionary Family. We refer to this subfamily as the *lunchbreak-algorithm* subfamily. Exclusion Zones caused by lunchbreak algorithms are called **lunchbreak Exclusion Zones**. We define this subfamily, then show that any algorithm in the lunchbreak-algorithm subfamily is not competitive.

Definition 7.9 Any Algorithm A $\in F_E$ is part of the *lunchbreak-algorithm* F_{LA} subfamily if there exists a location-time pair (l, t) with $t > \tau$ or $l = 0$ such that $(l, t) \in EZ(b) \forall b$.

Lemma 7.10 Any Algorithm A$\in F_{LA}$ is not competitive.

Proof To prove this Lemma, we construct an adversarial instance $\mathcal{I}$ to show non-competitiveness of any Algorithm A $\in F_{LA}$. By Definition 7.9, there exists a location-time pair (l, t) such that $(l, t) \in EZ(b) \forall b$. Consider instance $\mathcal{I}$ containing two requests $r = r' = (b_r, t, l)$. Note that these requests are rejected by Algorithm A, since $(l, t) \in EZ(r-)$. OPT can accept these requests to make x profit over A. Thus $\frac{P(OPT(\mathcal{I}))}{ALG(\mathcal{I})} = \frac{x}{0}$, which shows the non-competitive nature of F_{LA}. $\square$

Since lunchbreak Exclusion Zones only occur in the non-competitive lunchbreak-algorithm subfamily, we need not consider them for the rest of this analysis.

The second category of Exclusion Zone is the Exclusion Zones covering location-time intervals in which serving requests is impossible by the current or predicted location of the algorithm's servers. Specifically, if both of an Algorithm's servers are at the same location l at booking time b, it is impossible for these servers to reach $1 - l$ within τ time. Hence it is impossible to serve any requests $r = (b, t < b + \tau, 1 - l)$ in this case and request r must be in an Exclusion Zone. If r were not in an Exclusion Zone, by Definition 7.3 it would be accepted by an algorithm in the Exclusionary Family. This would lead to accepting a request that cannot be served, which would lead to an infeasible schedule. Alternatively, some Algorithms A $\in F_E$ reject requests that provide 0 profit. In such a case, if all of A's servers are at l at booking time b, A would not accept any request $r = (b, t, 1 - l)$ if $c = p$. Different yet, some Algorithms A $\in F_E$ reject requests that are booked very far in advance, if another request has already been booked as far in advance. We call these Exclusion Zones **location-based Exclusion Zones**. In short, location-based Exclusion Zones are caused because, by the location of A's servers at time b, A's servers would need to perform an inconvenient or even impossible amount of movements to serve a request inside those Exclusion Zones.

The third and final category of Exclusion Zone are the location-time intervals during which requests are rejected because these requests start too close in location-time relative to other requests served by the algorithm. We call these Exclusion Zones **request-based Exclusion Zones**.

We now more succinctly define request-based Exclusion Zones.

Definition 7.11 We define $ez(r)$, the Exclusion Zone caused by a request r, as $EZ(r+) \setminus EZ(r-)$.

Using the definition of the location- and request-based Exclusion Zones, we now create a framework for the upper bound analyses we run over the Exclusionary family. Recall Corollary 7.8.1, that states that all requests that OPT accepts must be inside an Exclusion Zone. This means the amount of requests OPT can serve in an Exclusion Zone is limited by its size and location. Indeed, it takes τ time to serve one non-distant request, and 2τ for one distant request. If we prove a cost for creating location- or request-based Exclusion Zones of predetermined size, we can show the performance of an algorithm on the worst possible Exclusion Zone.

We note that location-based Exclusion Zones only exist when constructed at specific times $EZ_M(b)$. In other words, these exclusion zones require a costly set-up to create an opportunity so that OPT can serve these requests. For upper bound proofs on A, we note the presence of location-based Exclusion Zones whenever necessary, and then reason on their impact.

Next, we reason on request-based Exclusion Zones. We can calculate the competitive ratio of algorithms in F_E by calculating the profit of R^{OPT} over $ez(r)$, and dividing that by the profit of r. Since the amount of profit OPT can make over ALG depends on the size of $ez(r)$, there is little an adversary can do to increase the amount of requests OPT can serve inside $ez(r)$. As such, the best way to increase the ratio of OPT over ALG in $\mathcal{I}$ is to try and maximize the average size $|ez(r)|$ of each $ez(r) \forall r \in R^A$. Formalizing this, we find

Observation 7.12 The ratio of $\frac{P(R^{OPT}(ez(r)))}{P(r)}$ is maximized in $ez(r)$ when $|ez(r)|$ is maximized. Furthermore, the ratio of $\frac{P(R^{OPT})}{P(R^A)}$ is maximized when $\frac{|\cup_{r \in R^A} ez(r)|}{|R^A|}$ is maximized.

Finally we prove a property that, to maximize the average of the sizes of the request-based Exclusion Zones, A either needs to accept pairs of requests, or requests for which the exclusion zones do not overlap. This Lemma limits the amount of instances that could show the competitive ratio for each of the algorithms in F_E.

Lemma 7.13 To maximize $\frac{|\cup_{r \in R^A} ez(r)|}{|R^A|}$ for every two requests $r, j \in R^A$, either $r = j$, or $ez(r) \cap ez(j) = \emptyset$.

Proof We prove this Lemma by contradiction. Assume that there exists some pair of requests r and j for which $|ez(r) \cup ez(j)|$ is maximized, whilst $r \neq j$ and $ez(r) \cap ez(j) \neq \emptyset$. We distinguish two cases. Either $ez(r) = ez(j)$ or $ez(r) \neq ez(j)$.

If $ez(r) \neq ez(j)$, then $|ez(r) \cup ez(j)| = |ez(r)| + |ez(j)| - |ez(r) \cap ez(j)| < |ez(r)| + |ez(j)|$. That contradicts the size of location-time interval $|ez(r) \cup ez(j)|$ being maximized with $ez(r) \cap ez(j) \neq \emptyset$.

If $ez(r) = ez(j)$, then $|ez(r) \cup ez(j)| = |ez(r) \setminus ez(j)| + |ez(j) \setminus ez(r)| + |ez(r) \cap ez(j)| = 0 + 0 + |ez(r) \cap ez(j)|$. Therefore, to maximize $|ez(r) \cup ez(j)|$ when $ez(r) = ez(j)$, we maximize $|ez(r) \cap ez(j)|$. That happens when $r = j$, contradicting our premise. $\square$

All this preparation work has severely limited the amount of instances we need to construct to attempt to prove the competitiveness of the Algorithms in F_E.

7.2 Restriction Zone Algorithm (RZA)

In this section, we introduce the *Restriction Zone Algorithm (RZA)*. This Algorithm is very rigid in its adherence to the Exclusionary Framework, but that rigidity delivers a steady competitive ratio that is not influenced by the size of the booking interval. This makes Restriction Zones an excellent pick for the setting with $\upsilon - \lambda > 9\tau$ and $\frac{5-\sqrt{13}}{2}p > c > \frac{p}{3}$. These settings give ample time to benefit from the steadfast nature of RZA.

RZA is created as a cautionary response to the eagerness of many algorithms to accept a pair of similar requests. We do this because accepting a pair of requests with similar starting times, booking times and location comes with a drawback. This drawback is that accepting such similar requests often gives the adversary a chance to release a series of requests that require empty movements. If too many requests are released in such a series, the competitive ratio of an algorithm accepting that series will become very dependant on the cost of empty movement c. Meanwhile, rejecting such a series will cause an algorithm to miss out on profit. These are the pitfalls GA and Smart Greedy fall into, respectively. Smart Greedy, particularly, performs poorly, because it refuses to accept requests requiring an empty movement.

RZA takes a very different approach. Instead of focusing on profit like Smart Greedy, or on acceptability like GA, RZA accepts no request that conflicts with another request, unless these requests start at different locations. It does this by adhering to a strict method, based on Observation 7.6. This method creates an Exclusion Zone spanning 2τ around the start time of any request r that is accepted at location l_r. If a request j starting from $\bar{l}_r$ conflicts with r, this request is still acceptable, and the method shortens the zone around r to the 2τ before the end time of j. It also creates a new Exclusion Zone at l_j, spanning the 2τ before the end time of r. Formally, the method is defined as follows:

Algorithm 3 Method $RZ(R^A, b)$

```
Input: Set of requests $R^A$ with $b > b_r \forall r \in R^A$, sorted by increasing $b$.
Output: Set of location-time intervals $EZ(b)$
begin
    $EZ(b) = \emptyset$
    request-based Exclusion Zones
    foreach $r \in R^A$ do
        if there is a request $j \in R^A$ with $b_j < b_r$ such that $\bar{l}_j = l_r$ and $|t_r - t_{r_j}| < \tau$: then
            $EZ(b) = EZ(b) \setminus ((l_j, (t_j - 2\tau, t_r - \tau)) \cup (l_j, (t_r + \tau, t_j + 2\tau))) \cup (\bar{l}_r, (t_j - \tau, t_j + \tau))$
        else
            $EZ(b) = EZ(b) \cup (l_r, (t_r - 2\tau, t_r + 2\tau))$
        end
    end
    location-based Exclusion Zones
    if It is not feasible, given $R^A$, that any server is at location $l$ at time $b$ then
        if It is feasible, given $R^A$, that any server is at location $l$ at time $c$ with $b < c < b + \tau$ then
            $EZ(b) = EZ(b) \cup (l, (b, c))$
        else
            $EZ(b) = EZ(b) \cup (l, (b, b + \tau))$
        end
end
```

This method differentiates between Exclusion Zones caused by a single request, and Exclusion Zones caused by two requests. Note that Exclusion Zones caused by two requests cause two noticeably smaller zones than a Exclusion Zones caused by a single request. We call two requests causing such smaller Exclusion Zones *crossing requests*. Note that a clever adversary can take advantage of the enlarged Exclusion Zone around a single request before introducing its crossing request. We now formally define the Restriction Zones Algorithm:

Algorithm 4 Algorithm $RZA(\mathcal{I})$

```
Input: Instance $\mathcal{I}$ of requests $r$ arriving by increasing $b_r$.
Output: Set of requests accepted by RZA $R^A$
begin
    $R^A = \emptyset$
    foreach $r = (b_r, t_r, l_r) \in \mathcal{I}$ do
        $EZ(b_r) = RZ(R^A, b_r)$ if $(l_r, t_r) \notin EZ(b_r)$ then
            $R^A = R^A \cup r$
        else
            reject $r$
        end
    end
end
```

Observe that Algorithm RZA perfectly follows $EF(\mathcal{I}, RZ(R, b)$. Therefore, we conclude that Algorithm RZA is a part of the Exclusionary Family. That means Observations 7.5 – 7.7 and Lemmas 7.8 – 7.10 hold. We now analyze the upper bound of this Algorithm on the instance with $\lambda > \tau$.

We prove this upper bound by systematically enumerating all possible Exclusion Zone sets. For every Exclusion Zone set, we observe the profit RZA gains when creating that Exclusion Zone set, the size of the created Exclusion Zone set, and the repeatability of that Exclusion Zone set. We use this data to find the Exclusion Zone set that causes the greatest ratio between size and profit for RZA, which, according to Observation 7.12, leads us to the competitive ratio. In order to find this Exclusion Zone set with greatest ratio between size and profit, we enumerate all cases that warrant a distinct reaction from RZA. We perform this enumeration in three phases. In the first phase, we consider different series of requests from the starting position in which all of RZA's servers and OPT's servers are at location 0. In the second phase, we consider possible follow-ups to the series released in the starting position, and how well RZA performs compared to OPT on the zone sets created by these follow-up requests. In the third and final phase, we instead consider the impact of requests line skipping the starting situation. We enumerate in phases, because that allows us to eliminate some cases with a low performance ratio early on. This cuts down on the amount of cases we need to check.

Observation 7.14.1.1 No request released in the location-based Exclusion Zone is acceptable from the starting position, if no other request is released first.

Proof By Algorithm 3, there is a location-based Exclusion Zones at the starting position. This location-based Exclusion Zone is one of size τ at location 1. Because a movement from location 0 to location 1 takes τ time, and because all servers start at location 0, it is impossible for OPT to accept requests to be served in that Exclusion Zone, without other requests being released first. Hence, the location-based Exclusion Zone is inaccessible from the starting position without a different request being released first. $\square$

Lemma 7.14.1 Case distinction starting from starting position: all servers at one location.

Proof We enumerate all possible reactions to requests released by an adversary in the starting scenario. This enumerates four cases: RZA accepting a single non-distant requests starting at 0, RZA accepting a single distant request at location 1, RZA accepting a pair of crossing requests with the first request arriving at location 0, and RZA accepting a pair of crossing requests, with the first request arriving at location 1. Other cases that would cause an interaction between Exclusion Zones caused by all released and accepted requests after the starting scenario would violate Lemma 7.13. Violating Lemma 7.13 shows that such cases do not maximize the performance ratio. Therefore, we can discard such cases.

Case 1 For the first possible Exclusion Zone set from the starting scenario, we consider a pair of requests $r_1 = r_2 = (b, t, 0)$. Note that we do not require a specific b or t for this case, representing a general pair of non-distant requests. Introducing this pair of requests, we find that RZA accepts one of them, gaining a profit of p. This adds location-time interval $(0, (t - 2\tau, t + 2\tau))$ to $EZ(r_2+)$. The size of that combined location-time interval is 4τ. By Corollary 7.8.1, this is all the time OPT has to accept requests. Since requests need to alternate to not require empty movements to be served, OPT can serve at most four requests in this Exclusion Zone, two of which requiring an empty movement. Therefore, serving these four requests, with two empty movements, OPT gains a profit of $4p - 2c$. The requests causing this Exclusion Zone set moves all of OPT's servers, and one of RZA's servers from location 0 to location 1, with the other RZA server staying at location 0 Therefore, this Exclusion Zone set is not infinitely repeatable. We note that we need to analyze this new formation in the second phase. For this Exclusion Zone set, we find the performance of OPT compared to RZA to be $\frac{4p-2c}{p}$.
Case 2 For the second possible Exclusion Zone set from the starting scenario, we consider a pair of requests $r_1 = r_2 = (b, t, 1)$. Note that we do not require a specific b or t for this case, representing a general pair of distant requests. Introducing this pair of requests, we find that RZA accepts one of them, gaining a profit of $p - c$. This adds location-time interval $(1, (t - 2\tau, t + 2\tau))$ to $EZ(r_2+)$. The size of that location-time interval is 4τ. Since there is no Exclusion Zone at 0, by Corollary 7.8.1, OPT can only accept requests starting between $t - 2\tau$ and $t + 2\tau$ at location 1. Since requests need to alternate to not require empty movements to be served, OPT can serve at most four requests in this Exclusion Zone. Since the first requests already require an empty movement, all four accepted requests require empty movements, noting a profit of $4p - 4c$ for OPT. Since all servers end where they started this Exclusion Zone set, this Exclusion Zone set is infinitely repeatable. For this Exclusion Zone set, we find the performance of OPT compared to RZA to be $\frac{4p-4c}{p-c} = 4$.
Case 3 For the third possible Exclusion Zone set from the starting scenario, we consider a pair of re-

quests $r_1 = r_2 = (b, t, 0)$, followed by the release of another pair of crossing requests $r_3 = r_4 = (d, u, 1)$, with $d \geq b$ and $t - \tau < u < t + \tau$. Note that we do not require a specific b or t for this case, and that d and u are only relative to b and t, representing a general pair of non-distant requests followed by a general pair of crossing distant requests. We start this analysis by looking at the impact of the adversary's first release: r_1 and r_2. Introducing this first pair of requests, we find that RZA accepts one of them, gaining a profit of p. The first accepted request adds location-time interval $(0, (t - 2\tau, t + 2\tau))$ to $EZ(r_2+)$. The size of that location-time interval is 4τ. Since there is no Exclusion Zone at 1, by Corollary 7.8.1, OPT only accepts requests starting between $t - 2\tau$ and $t + 2\tau$ at location 0. Since requests need to alternate to not require empty movements to be served, OPT can serve at most 4 requests in this Exclusion Zone. Since the first two requests are at location 0, and OPT's servers are at location 0, only the latter two requests require an empty movement. After this, we analyze the impact the adversary's subsequent release of r_3, r_4 has. Introducing this pair of requests, we find that RZA accepts one of them, gaining a profit of $p - c$. This shortens the Exclusion Zone at $(0, (t - 2\tau, t + 2\tau))$, and adds an extra zone to the set at $(1, (t - \tau, t + \tau))$. This zone is of size 2τ. Thus, since OPT only serves requests starting in Exclusion Zones by Corollary 7.8.1, OPT can only serve two requests from location 0. These requests do not require an empty movement, and, in fact, serve as two of the required empty movements. This means OPT serves six requests for a profit of $6p$. The requests causing this Exclusion Zone set causes RZA's servers to be spread between location 0 and location 1. Therefore, this Exclusion Zone set is not infinitely repeatable. For this Exclusion Zone set, we find the performance of OPT compared to RZA to be $\frac{6p}{2p-c}$.
Case 4 For the fourth possible Exclusion Zone set from the starting scenario, we consider a pair of requests $r_1 = r_2 = (b, t, 1)$, followed by the release of another pair of crossing requests $r_3 = r_4 = (d, u, 0)$, with $d \geq b$ and $t - \tau < u < t + \tau$. Note that we do not require a specific b or t for this case, and that d and u are only relative to b and t, representing a general pair of distant requests followed by a general pair of crossing non-distant requests. We start this analysis by looking at the impact of the adversary's first release: r_1 and r_2. Introducing this first pair of requests, we find that RZA accepts one of them, gaining a profit of $p - c$. The first accepted request adds location-time interval $(1, (t - 2\tau, t + 2\tau))$ to $EZ(r_2+)$. The size of that location-time interval is 4τ. Since there is no Exclusion Zone at 0, by Corollary 7.8.1, OPT only accepts requests starting between $t - 2\tau$ and $t + 2\tau$ at location 1. Since requests need to alternate to not require empty movements to be served, OPT can serve at most 4 requests in this Exclusion Zone. Since the first two requests are at location 1, while OPT's servers are at location 0, all four of those require an empty movement. After this, we analyze the impact the adversary's subsequent release of r_3, r_4 has. Introducing this pair of requests, we find that RZA accepts one of them, gaining a profit of p. This shortens the Exclusion Zone at $(1, (t - 2\tau, t + 2\tau))$, and adds an extra zone to the set at $(0, (t - \tau, t + \tau))$. This zone is of size 2τ. Thus, since OPT only serves requests starting in Exclusion Zones by Corollary 7.8.1, OPT can only serve two requests from location 0. These requests do not require an empty movement, and, in fact, serve as two of the required empty movements. This means OPT serves six requests for a profit of $6p - 2c$. The requests causing this Exclusion Zone set causes RZA's servers to be spread between location 0 and location 1. Therefore, this Exclusion Zone set is not infinitely repeatable. For this Exclusion Zone set, we find the performance of OPT compared to RZA to be $\frac{6p-2c}{2p-c}$. $\square$

This was a distinction of all possible releases by an adversary from the starting scenario. Note that, in order to continue releasing different requests, any adversary first has to release one of these four cases. With that, we end the case distinction on this phase. From the case distinction on phase one, we make the following observations.

Corollary 7.14.1.2 The only formation of servers that the starting scenarios can result in, that is not symmetric to the starting scenario, is a scenario with one of RZA's servers at location 0 and location 1 each, and OPT's servers together at one location.

Having explored all four possible Exclusion Zone sets from the starting positions of the servers, we move on to the second phase. In this second phase, we consider starting from the different configurations of servers we encountered. By Corollary 7.14.1.2, we have only seen one different configuration of servers after the starting configuration: a configuration where one of RZA's servers is at location 0, with the other server at location 1, and OPT's servers together at one location. Without loss of generality, we assume location 1 to be the location of OPT's servers. We refer to this new scenario as *the follow-up scenario.*

Observation 7.14.2.1 There is no location-based Exclusion Zone in the follow-up scenario.

Proof Since RZA has a server at both location 0 and 1, any request released at 0 or 1 can be acceptable. Therefore, there is no need for a location-based Exclusion Zone. □

Lemma 7.14.2 Case distinction starting from follow-up scenario: one RZA server at 0, one RZA server at 1, and OPT servers united at one location.

Proof We enumerate all possible reactions to requests released by an adversary in the follow-up scenario. This requires four cases: either RZA accepts one request from location 0, or RZA accepts one request from location 1, or RZA accepts one request from location 0, then one from location 1, or RZA accepts one request from location 1, then one from location 0. Other cases besides the ones enumerated require a substantial gap between accepted requests, from where we would be able to analyze before and after the gap separately. If this gap were smaller, the Exclusion Zones on both sides of the gap would overlap, violating Lemma 7.13. Thus, such cases need not be handled in this Lemma.

Case 1 For the first possible follow-up Exclusion Zone set, we consider a pair of requests $r_5 = r_6 = (e, w, 0)$, with $e > b$, and $w > t$. Note that e and w are only specified with a relation to b and t. This makes r_5, r_6 a pair of general requests at location 0 released and served after r_1, r_2. Introducing this pair of requests, we find that RZA accepts one of them, gaining a profit of p. Accepting this request adds location-time interval $(0, (w - 2\tau, w + 2\tau))$ to $EZ(r_6+)$. The size of that location-time interval is 4τ. Since there is no Exclusion Zone at 1, by Corollary 7.8.1, OPT can only accept requests starting between $t - 2\tau$ and $t + 2\tau$ at location 0. Since the first two requests are at location 0, while OPT's servers are at location 1, all four of those require an empty movement. This means OPT serves four requests for a profit of $4p - 4c$. The requests causing this Exclusion Zone set unites RZA's servers at location 1, with OPT's servers also at location 1. Therefore, this Exclusion Zone set is infinitely repeatable in tandem with Lemma 7.14.1's Cases 1, 3 and 4. However, even when combining the ratio of this case with the ratio of Lemma 7.14.1's Case 3, we find the performance of OPT compared to RZA to be lower than $\frac{6p}{2p-c}$.

Case 2 For the second possible follow-up Exclusion Zone set, we consider a pair of requests $r_5 = r_6 = (e, w, 1)$, with $e > b$, and $w > t$. Note that e and w are only specified with a relation to b and t. This makes r_5, r_6 a pair of general requests at location 1 released and served after r_1, r_2. Introducing this pair of requests, we find that RZA accepts one of them, gaining a profit of p. Accepting this request adds location-time interval $(1, (w - 2\tau, w + 2\tau))$ to $EZ(r_6+)$. The size of that location-time interval is 4τ. Since there is no Exclusion Zone at 0, by Corollary 7.8.1, OPT can only accept requests starting between $t - 2\tau$ and $t + 2\tau$ at location 1. Since the first two requests are at location 1, and OPT's servers are at location 1, only two of those require an empty movement. This means OPT serves four requests for a profit of $4p - 2c$. The requests causing this Exclusion Zone set unites RZA's servers at location 0, with OPT's servers also at location 0. Therefore, this Exclusion Zone set is infinitely repeatable in tandem with Lemma 7.14.1's Cases 1, 3 and 4. However, for this combined Exclusion Zone set, we still find the performance of OPT compared to RZA to be lower than $\frac{6p}{2p-c}$.

Case 3 For the third possible follow-up Exclusion Zone set is created by releasing a pair of requests $r_5 = r_6 = (e, w, 0)$, with $e > b$, and $w > t$, then following up with a pair of requests $r_7 = r_8 = (f, x, 1)$ with $b < e \leq f$ and $t < x - \tau < w < x + \tau$. Note that e and w are only specified with a relation to b and t. This makes r_5, r_6 a pair of general requests at location 0 released and served after r_1, r_2. Also, note that f and x are only specified with a relation to e, w, b and t. This makes r_7, r_8 a pair of general requests crossing r_5, r_6. We first discuss the impact of the first pair of requests. Introducing this pair of requests, we find that RZA accepts one of them, gaining a profit of p. Accepting this request adds location-time interval $(0, (w - 2\tau, w + 2\tau))$ to $EZ(r_6+)$. The size of that location-time interval is 4τ. Since there is no Exclusion Zone at 1, by Corollary 7.8.1, OPT can only accept requests starting between $t - 2\tau$ and $t + 2\tau$ at location 0. Since the first two requests are at location 0, while OPT's servers are at location 1, all four of those require an empty movement.

After this, we consider the pair of requests $r_7 = r_8 = (f, x, 1)$. Introducing this pair of requests, we find that RZA accepts one of them, gaining a profit of p. This shortens the Exclusion Zone at $(0, (w - 2\tau, w + 2\tau))$, and adds an extra zone to the set at $(1, (w - \tau, w + \tau))$. This zone is of size 2τ. Thus, OPT can only serve two requests from location 1. These requests do not require an empty movement, and, in fact, serve as two of the required empty movements. This means OPT serves six requests for a profit of $6p - 2c$. After the requests causing this Exclusion Zone set are served, RZA's servers are still at the two separate locations: one at location 0 and the other at location 1. OPT's servers stay at the same location. Therefore, this Exclusion Zone set is infinitely repeatable. Repeating this follow-up Exclusion Zone set infinitely many times we find the performance of OPT compared to RZA to be $\frac{6p-2c}{2p}$. This is markedly lower than $\frac{6p}{2p-c}$. Furthermore, even when combining this Exclusion Zone set with Lemma 7.14.1's Case 3 from the starting scenario, we find the performance of OPT compared to RZA to be lower than $\frac{6p}{2p-c}$.

Case 4 For the fourth possible follow-up Exclusion Zone set we consider a pair of requests $r_5 = r_6 = (e, w, 1)$, with $e > b$, and $w > t$, then following up with a pair of requests $r_7 = r_8 = (f, x, 0)$ with $b < e \leq f$ and $t < x - \tau < w < x + \tau$. Note that e and w are only specified with a relation to b and t. This makes r_5, r_6 a pair of general requests at location 0 released and served after r_1, r_2. Also, note that f and x are only specified with a relation to e, w, b and t. This makes r_7, r_8 a pair of general requests crossing r_5, r_6. We first discuss the impact of the first pair of requests. Introducing the first pair of requests, we find that RZA accepts one of them, gaining a profit of p. Accepting this request adds location-time interval $(1, (w - 2\tau, w + 2\tau))$ to $EZ(r_6+)$. The size of that location-time interval is 4τ. Since there is no Exclusion Zone at 0, by Corollary 7.8.1, OPT can only accept requests starting between $t - 2\tau$ and $t + 2\tau$ at location 1. Since the first two requests are at location 1, and OPT's servers are at location 1, only two of these require an empty movement.

After this, we consider the pair of requests $r_7 = r_8 = (f, x, 0)$. Introducing this pair of requests, we find that RZA accepts one of them, gaining a profit of p. This shortens the Exclusion Zone at $(1, (w - 2\tau, w + 2\tau))$, and adds an extra zone to the set at $(0, (w - \tau, w + \tau))$. This zone is of size 2τ. Thus, OPT can only serve two requests from location 0. These requests do not require an empty movement, and, in fact, serve as two of the required empty movements. This means OPT serves six requests for a profit of $6p$. After the requests causing this Exclusion Zone set are served, RZA's servers are still at the two separate locations: one at location 0 and the other at location 1. OPT's servers stay at the same location. Therefore, this Exclusion Zone set is infinitely repeatable. Repeating this follow-up Exclusion Zone set infinitely many times we find the performance of OPT compared to RZA to be $\frac{6p}{2p} = 3$. This is markedly lower than 4. Furthermore, even when combining this Exclusion Zone set with Lemma 7.14.1's Case 3, we find the performance of OPT compared to RZA to be lower than $\frac{6p}{2p-c}$. □

This was a distinction of all possible releases by an adversary from the follow-up scenario reached after Lemma 7.14.1's Cases 1, 3 and 4. With that, we end the case distinction on this phase. From this phase, we make the following observations.

Corollary 7.14.2.2 An adversary cannot release a series of requests starting from the follow-up scenario such that the performance ratio of RZA and OPT rises above 4.

Corollary 7.14.2.3 An adversary cannot release a series of requests starting from follow-up scenario such that the formation the servers at the end of the follow-up scenario are left in a scenario other than the starting scenario, or the follow up scenario.

Having explored all four follow-up Exclusion Zone sets, we move on to the third and final phase. In this third phase, we focus on the line-skipping case. The line-skipping case can be very beneficial to the performance. Non-distant requests that were booked in Lemma 7.14.1 could require an empty movement to serve if they are line-skipped. However, RZA is very particular in dealing with this problem.

Lemma 7.14.3 Line-skipping any of RZA's reactions to the starting scenario does not increase the performance ratio achieved upon those reactions.

Proof Recall that any non-crossing request r accepted by RZA causes an Exclusion Zone of size 4τ centered on the accepted request. This guarantees that any non-crossing request can be served by either server. Therefore, Observation 7.6 never mandates that any Exclusion Zone exists at $\bar{l}_r$. To create an Exclusion Zone at $\bar{l}_r$, a request needs to be released at $\bar{l}_r$. If such a request is not released - aside from the first Exclusion Zone created, which we discussed in Lemma 7.14.1 - OPT can only accept requests requiring empty movements in 4τ long Exclusion Zones. In such a zone, OPT can accept at most 4 requests, all of which require empty movements, for a profit of $4p - 4c$. Since RZA also accepts one request to create these Exclusion Zones, the performance ratio on such Exclusion Zones does not exceed 4.

In the case a crossing request is released, note that RZA's servers are spread out, with one server at location 0 and one server at location 1 afterwards. Serving any formation of requests described in Lemma 7.14.1 does not mandate an extra distant request from that formation. Therefore, we can still treat the line-skipping requests and the starting position requests separately. This means the performance ratio is not improved upon introducing crossing line-skipping requests to the consideration. $\square$

We conclude that there is no need to further distinct the line-skipping scenario to the reactions provided in Lemma 7.14.1. We have found closing arguments for all possible releases by an adversary, starting from the starting scenario, following-up on that scenario, or line-skipping that scenario. Having enumerated everything, we provide the Competitive Ratio of RZA.

Theorem 7.14 Algorithm RZA is $\max\{4, \frac{6p}{2p-c}\}$ competitive for the 2S2L-V problem with arbitrary c, and $\lambda > \tau$.

Proof We proof this Theorem by enumerating all possible releases of an adversary that do not violate Lemma 7.13. We have done this in Lemmas 7.14.1 and 7.14.2. We know that these Lemma's enumerate all cases relevant to the performance ratio by Corollaries 7.14.1.2, 7.14.2.3, and Lemma 7.14.3. In those numerous cases, we found that the releases leading to Lemma 7.14.1's case 3 provided the highest performance ratio for $c > \frac{p}{2}$. For $c < \frac{p}{2}$, we found various cases achieving a performance ratio of 4. Therefore, the competitive ratio of RZA is $\max\{4, \frac{6p}{2p-c}\}$. $\square$

7.3 Arrogant Greedy (AG)

In this section, we introduce the *Arrogant Greedy Algorithm (AG)*. As with all algorithms in this chapter, AG is part of the Exclusionary Family. This particular algorithm takes inspiration from Smart Greedy, whilst becoming more flexible to avoid the problems SG runs into. Whereas Smart Greedy simply investigates whether a request gains maximum profit, Arrogant Greedy investigates whether a request requires a distant movement to be served prior to the request. That way, AG accepts more requests than Smart Greedy would. Furthermore, AG is less strict about accepting less profitable requests, accepting them when the servers are not predicted to be busy. Settings with long booking time intervals still cause issues for AG. In such settings, Arrogant Greedy will keep accepting requests to the point of being exploitable by a clever adversary.

After properly introducing the algorithm, we show its competitive ratio. AG achieves a tight upper bound for various settings of c with $2\tau < \upsilon - \lambda \leq 9\tau$. For $2\tau < \upsilon - \lambda \leq 7\tau$, AG achieves a tight upper bound for all settings $\frac{p}{3} < c \leq p$. Additionally, for $7\tau < \upsilon - \lambda \leq 9\tau$, AG achieves a tight upper bound for $\frac{p}{3} < c < \frac{19-\sqrt{217}}{6}p$.

In this section, we first discuss the definition of AG. Next, we show the Method to construct its Exclusion Zones. This shows that AG is part of the Exclusionary Family. Finally, we prove its performance on 2S2L-V.

For the definition of AG, we harken back to Smart Greedy. The great difficulty Smart Greedy has on 2S2L-V is line-skipping. SG does not perform well, because it does not anticipate that a request booked later can be performed earlier. Furthermore, Smart Greedy is very strict with its profit margins. We loosen up the prior problem by allowing Arrogant Greedy to accept requests requiring an empty movement if neither server has to be used to serve a request in conflict with the new request. Furthermore, we loosen up the empty movement requirements. Instead of being strict for all empty movements, Arrogant Greedy only focuses on whether a request seems to require an empty movement prior to serving it. This gives it more flexibility that Smart Greedy did not have.

Having discussed the main ideas behind AG, we introduce one more optimization before we introduce the algorithm itself. This optimization concerns requests to be served $\upsilon - \tau$ or more time after they were booked. Algorithms greedily accepting non-distant requests, such as AG, can be taken advantage of by releasing a pair of non-distant requests that are served close to υ after being booked. Normally, algorithms such as AG would accept these requests. This causes a request-based Exclusion Zone, that stretches at most 2τ beyond υ. such a request based Exclusion Zone can be exploited with a Zig-Zag gadget. We are able to shorten that zone stretching beyond υ by withholding accepting requests near υ away from the booking time. This forces the adversary to waste a bit of time to set up this faraway Exclusion Zone. (See also section 5.4 concerning the commitment gadget, where waiting to accept such requests caused the second request pair accepted by OPT to also need to be released prior to releasing line-skipping zig-gadgets). Hence, we introduce a clause concerning requests released in the final τ time in the booking interval.

The eventual Algorithm is as follows:

Algorithm 5 Algorithm $AG(\mathcal{I})$

```
Input: Instance $\mathcal{I}$ of requests $r$ arriving by increasing $b_r$.
Output: Set of requests accepted by AG $R^A$
begin
    R$^A$ =$\emptyset$
    foreach $r = (b_r, t_r, l_r) \in \mathcal{I}$ do
        if $r$ is acceptable then
            if There is a request $j \in R^A$ such that $|t_j - t_r| < \tau + \tau(\bar{l}_r, l_j)$ and $t_r - b_r > \upsilon - \tau$ then
                reject $r$
            else if $r$ does not require an empty movement to be performed prior to serving it. then
                $R^A = R^A \cup r$
            else if There is no request $j \in R^A$ such that $|t_j - t_r| < \tau + \tau(\bar{l}_r, l_j)$ then
                $R^A = R^A \cup r$
        else
            reject $r$
        end
    end
end
```

Having shown the contents of Arrogant Greedy, we now need to show that it is part of the Exclusionary Family. To that purpose, we need a method encapsulating the if-clauses of AG into a set of Exclusion Zones. To that purpose, we analyze method AZ.

Algorithm 6 Method $AZ(R^A, b)$

```
Input: Set of requests $R^A$ of requests with $b > b_r \forall r \in R^A$, sorted by increasing $b$.
Output: Set of location-time intervals $EZ(b)$
begin
    $EZ(b) = \emptyset$
    request-based Exclusion Zones
    foreach $r \in R^A$ do
        if request $r$ or paired request $r'$ requires an empty movement to be served starting at $t_r - \tau$, in the most profitable schedules then
            $EZ(b) = EZ(b) \cup (l_r, (t_r - 2\tau, t_r + 2\tau))$
        if request $r$ requires no empty movement to be served, and does not change the amount of empty movements required to serve all requests in $R^A$ then
            $EZ(b) = EZ(b) \setminus (\bar{l}_r, (\bar{t}_r, \infty)) \cup (\bar{l}_r, (\bar{t}_r - 2\tau, \bar{t}_r))$
        if there is a request $j \in R^A$ with $b_j < b_r$ such that $\bar{l}_j = l_r$ and $|t_r - t_{r_i}| < \tau$: then
            $EZ(b) = EZ(b) \cup (l_r, (t_j - \tau, t_j + \tau)) \cup (l_j, (t_r - \tau, t_r + \tau))$
        else if there is a request $j \in R^A$ with $b_j < b_r$ such that $l_j = l_r$ and $|t_r - t_{r_i}| < 2\tau$: then
            $EZ(b) = EZ(b) \cup (l_r, (\max\{t_j, t_r\} - 2\tau, \max\{t_j, t_r\} + 2\tau)) \cup (\bar{l}_j, (\max\{t_j, t_r\} - \tau, \min\{t_j, t_r\} + \tau))$
    end
    location-based Exclusion Zones if It is not feasible, given $R^A$, that any server is at location $l$ at time $b$ then
        if It is feasible, given $R^A$, that any server is at location $l$ at time $c$ with $b < c < b + \tau$ then
            $EZ(b) = EZ(b) \cup (l, (b, c))$
        else
            $EZ(b) = EZ(b) \cup (l, (b, b + \tau))$
        end
end
End of line Exclusion Zone if There exists a request $j \in R^A$ such that $t_j > b + \upsilon - \tau$ and there exists a schedule over $R$ of maximum profit such that both servers are at $l_j$ at time $b$ then
    $EZ(b) = EZ(b) \cup (l_j, (b + \upsilon - \tau, b + \upsilon))$
```

We observe seven different if-clauses in method AZ. We discuss them one by one, to see how they compare to the if-clauses of algorithm AG. If we can show a one to one correspondence, we have constructed a set of Exclusion Zones that - when followed in the Exclusionary Framework- emulates the behaviour of AG. That proves that AG is part of the Exclusionary Family.

The first if-clause of AZ corresponds to the else if-clause of AG. We observe that this first if-clause creates an Exclusion Zone of 2τ around the starting time t_r of any request r served without its paired request. This could be the case because the two servers are at different locations, or because r requires an empty movement. This bars an Algorithm following the AZ zones from accepting a distant request close to an already accepted distant request. AG also does not accept such requests.

The second if-clause of AZ is necessary to ensure that the Exclusionary Algorithm following AZ handles an edge case correctly. This edge case happens when a distant request is accepted, followed by two line-skipping non-distant requests. Because the Exclusionary Algorithm would follow the Exclusion Zones caused by the first if-clause, the algorithm would differ from AG's decision when it comes to requests around the accepted distant request (that request would now no longer be distant). Removing the zone around this request in this specific case rectifies this behaviour. This way, every acceptable non-distant request is accepted by the Exclusionary Algorithm implementing AZ.

The third if-clause of AZ corresponds to a reaction to crossing requests. If a newly introduced request crosses a previously accepted one, Exclusion Zones are appended, such that Observation 7.6 is followed. Similarly, the fourth if-clause of AZ corresponds to the standard reaction to two nearby requests being accepted. Once again, this appends Exclusion Zones, such that Observation 7.6 is followed. The fifth and sixth if-clauses of AZ correspond to setting up location-based Exclusion Zones in positions that the servers just cannot reach. These four if-clauses combined make sure that all unacceptable requests start in Exclusion Zones. That takes care of the acceptability constraint on AG.

The seventh and final if-clause of AZ corresponds to the check whether a request which is booked long in advance has been accepted. If such a request exists, no other request can be accepted with a booking time near υ. This corresponds to the first if-statement after acceptability in AG.

Having explored all if-clauses, and comparing them to AG, we find that $EF(\mathcal{I}, AZ(R^A, b))$ abides by the exact same constraints as AG. Therefore, AG is in the Exclusionary Family. That means Observations, Corollaries and Lemmas $7.5 - 7.13$ hold on AG. Having shown this, we now set out to prove the competitive ratio of AG on the unbounded instance.

We prove this upper bound by systematically enumerating all possible Exclusion Zone sets. For every Exclusion Zone set, we observe the profit AG gains when creating that Exclusion Zone set, the size of the created Exclusion Zone set, and the repeatability of that Exclusion Zone set. We use this data to find the Exclusion Zone set that causes the greatest ratio between size and profit for AG, which, according to Observation 7.12, leads us to the competitive ratio. In order to find this Exclusion Zone set with greatest ratio between size and profit, we enumerate all cases that warrant a distinct reaction from AG. We perform this enumeration in three phases. In the first phase, we consider different series of requests from the starting position in which all of AG's servers and OPT's servers are at location 0. In the second phase, we consider possible follow-ups to the series released in the starting position, and how well AG performs compared to OPT on the zone sets created by these follow-up requests. In the third and final phase, we instead consider the impact of requests line skipping the starting situation. We enumerate in phases, because that allows us to eliminate some cases with a low performance ratio early on. This cuts down on the amount of cases we need to check.

Observation 7.15.1.1 No request released in the location-based Exclusion Zone is acceptable from the starting position, if no other request is released first.

Proof By Algorithm 6, there is a location-based Exclusion Zones at the starting position. This location-based Exclusion Zone is one of size τ at location 1. Because a movement from location 0 to location 1 takes τ time, and because all servers start at location 0, it is impossible for OPT to accept requests to be served in that Exclusion Zone, without other requests being released first. Hence, the location-based Exclusion Zone is inaccessible from the starting position without a different request being released first. □

Lemma 7.15.1 Case distinction starting from starting position: all servers at one location.

Proof We enumerate all possible reactions to requests released by an adversary in the starting scenario. This enumerates four cases: AG accepting a pair of non-distant requests starting at 0 with notification interval no larger than $\upsilon - \tau$, AG accepting a single distant request at location 1, AG accepting a pair of crossing requests, with the first request arriving at location 1, and AG accepting a single non-distant request starting at 0 with notification interval larger than $\upsilon - \tau$. Other cases that would cause an interaction between Exclusion Zones caused by all released and accepted requests after the starting scenario, would violate Lemma 7.13. Violating Lemma 7.13 shows that such cases do not maximize the performance ratio. Therefore, we can discard such cases. We now bring special attention to the fact that there is no case with a pair of crossing requests, with the first request accepted at 0. The reason for this is twofold. Either requests arriving at location 0 are released at a short enough notification interval that the paired request released alongside would also be accepted, or the notification interval is so long that any other request would be rejected. This would put us in Case 1 or 4, respectively, instead.

Case 1 For the first possible Exclusion Zone set from the starting scenario, we consider a pair of requests $r_1 = r_2 = (b, t, 0)$, with $t - b < \upsilon - \tau$. Note that we do not require a specific b or t for this case, representing a general pair of non-distant requests with notification interval smaller than $\upsilon - \tau$. Introducing this pair of requests, we find that AG accepts both of them, gaining a profit of $2p$. This adds location-time interval $(0, (t - 2\tau, t + 2\tau))$ and $(1, (t - \tau, t + \tau))$ to $EZ(r_2+)$. The size of those combined location-time intervals is 6τ. By Corollary 7.8.1, this is all the time OPT has to accept requests. Since there is an Exclusion Zone at both locations, OPT can accept requests starting between $t-2\tau$ and $t+2\tau$ at location 0, and between $t - \tau$ and $t + \tau$ at location 1. In this time, OPT can serve six requests: four starting from 0, and two more starting from 1. None of these requests require an empty movement to be served. Therefore, serving these six requests, OPT gains a profit of $6p$. The requests causing this Exclusion Zone set moves all servers from location 0 to location 1. Therefore, this Exclusion Zone set is infinitely repeatable. For this Exclusion Zone set, we find the performance of OPT compared to AG to be $\frac{6p}{2p}$.

Case 2 For the second possible Exclusion Zone set from the starting scenario, we consider a pair of requests $r_1 = r_2 = (b, t, 1)$. Note that we do not require a specific b or t for this case, representing a general pair of distant requests. Introducing this pair of requests, we find that AG accepts one of them, gaining a profit of $p - c$. This adds location-time interval $(1, (t - 2\tau, t + 2\tau))$ to $EZ(r_2+)$. The size of that location-time interval is 4τ. Since there is no Exclusion Zone at 0, by Corollary 7.8.1, OPT can only accept requests starting between $t - 2\tau$ and $t + 2\tau$ at location 1. Since requests need to alternate to not require empty movements to be served, OPT can serve at most four requests in this Exclusion Zone. Since the first requests already require an empty movement, all four accepted requests require empty movements, noting a profit of $4p - 4c$ for OPT. Since all servers end where they started this Exclusion Zone set, this Exclusion Zone set is infinitely repeatable. For this Exclusion Zone set, we find the performance of OPT compared to GRZA to be $\frac{4p-4c}{p-c} = 4$.

Case 3 For the third possible Exclusion Zone set from the starting scenario, we consider a pair of requests $r_1 = r_2 = (b, t, 1)$, followed by the release of another pair of crossing requests $r_3 = r_4 = (d, u, 0)$, with $d \geq b$ and $t-\tau < u < t+\tau$. Note that we do not require a specific b or t for this case, and that d and u are only relative to b and t, representing a general pair of distant requests followed by a general pair of crossing non-distant requests. We start this analysis by looking at the impact of the adversary's first release: r_1 and r_2. Introducing this first pair of requests, we find that AG accepts one of them, gaining a profit of $p-c$. The first accepted request adds location-time interval $(1, (t-2\tau, t+2\tau))$ to $EZ(r_2+)$. The size of that location-time interval is 4τ. Since there is no Exclusion Zone at 0, by Corollary 7.8.1, OPT only accepts requests starting between $t - 2\tau$ and $t + 2\tau$ at location 1. Since requests need to alternate to not require empty movements to be served, OPT can serve at most 4 requests in this Exclusion Zone. Since the first two requests are at location 1, while OPT's servers are at location 0, all four of those require an empty movement. After this, we analyze the impact the adversary's subsequent release of r_3, r_4 has. Introducing this pair of requests, we find that AG accepts one of them, gaining a profit of p. This shortens the Exclusion Zone at $(1, (t-2\tau, t+2\tau))$, and adds an extra zone to the set at $(0, (t-\tau, t+\tau))$. This zone is of size 2τ. Thus, since OPT only serves requests starting in Exclusion Zones by Corollary 7.8.1, OPT can only serve two requests from location 0. These requests do not require an empty movement, and, in fact, serve as two of the required empty movements. This means OPT serves six requests for a profit of $6p - 2c$. The requests causing this Exclusion Zone set causes AG's servers to be spread between location 0 and location 1. Therefore, this Exclusion Zone set is not infinitely repeatable. For this Exclusion Zone set, we find the performance of OPT compared to AG to be $\frac{6p-2c}{2p-c}$.

Case 4 For the fourth possible Exclusion Zone set from the starting scenario, we consider a pair of requests $r_1 = r_2 = (b, t, 0)$ with $t - b > \upsilon - t$. Note that we do not require a specific b or t for this case, representing a general pair of non-distant requests with notification interval greater than $v - \tau$. Introducing this pair of requests, we find that AG accepts one of them, gaining a profit of p. This creates two different kinds of Exclusion Zones. A request-based Exclusion Zone at $(1, (t - \tau, t + \tau))$ is added to $EZ(r_2+)$, and a location-based Exclusion Zone based at $(0, (b + v - \tau, b + v))$ is added to $EZ(a)$ for any time a with $b < a \leq t - v - \tau$. The size of these location-time intervals are 2τ and τ, respectively. In these intervals, by Corollary 7.8.1, OPT can accept at most four requests, none of which require an empty movement, for a profit of $4p$. The request causing this Exclusion Zone causes AG's servers to spread between location 0 and location 1, whilst OPT's servers move together, ending together at location 1. Therefore, this Exclusion Zone set is not infinitely repeatable. For this Exclusion Zone set, we find a performance ratio of OPT compared to AG to be $\frac{4p}{p}$. □

This was a distinction of all possible releases by an adversary from the starting scenario. Note that, in order to continue releasing different requests, any adversary first has to release one of these four cases. With that, we end the case distinction on this phase. From the case distinction on phase one, we make the following observations.

Corollary 7.15.1.2 The only formation of servers that the starting scenarios can result in, that is not symmetric to the starting scenario, is a scenario with one of AG's servers at location 0 and location 1 each, and OPT's servers together at one location.

Corollary 7.15.1.3 Only one case of the starting scenario requires both of AG's servers to be at location 0 at time t, the starting time of the first request.

Corollary 7.15.1.4 Two cases of the starting scenario require one of AG's servers to be at location 1 at time t, the starting time of the first request.

Corollary 7.15.1.5 One case of the starting scenario does not require the use of both of AG's servers.

Having explored all four possible Exclusion Zone sets from the starting positions of the servers, we move on to the second phase. In this second phase, we consider starting from the different configurations of servers we encountered. By Corollary 7.15.1.2, we have only seen one different configuration of servers after the starting configuration: Lemma 7.15.1's Cases 3 and 4 cause a configuration where one of AG's servers is at location 0, with the other server at location 1, and OPT's servers together at one location. Without loss of generality, we assume location 1 to be the location of OPT's servers. We refer to this new scenario as *the follow-up scenario.*

Observation 7.15.2.1 There is no location-based Exclusion Zone in the follow-up scenario.

Proof Since AG has a server at both location 0 and 1, any request released at 0 or 1 can be acceptable. Therefore, there is no need for a location-based Exclusion Zone. □

Lemma 7.15.2 Case distinction starting from follow-up scenario: one AG server at 0, one AG server at 1, and OPT servers united at one location.

Proof We enumerate all possible reactions to requests released by an adversary in the follow-up scenario. This requires four cases: either AG accepts one request from location 0, or AG accepts one request from location 1, or AG accepts one request from location 0, then one from location 1, or AG accepts one request from location 1, then one from location 0. Other cases besides the ones enumerated require a substantial gap between accepted requests, from where we would be able to analyze before and after the gap separately. If this gap were smaller, the Exclusion Zones on both sides of the gap would overlap, violating Lemma 7.13. Thus, such cases need not be handled in this Lemma.

Case 1 For the first possible follow-up Exclusion Zone set, we consider a pair of requests $r_5 = r_6 = (e, w, 0)$, with $e > b$, and $w > t$. Note that e and w are only specified with a relation to b and t. This makes r_5, r_6 a pair of general requests at location 0 released and served after r_1, r_2. Introducing this pair of requests, we find that AG accepts one of them, gaining a profit of p. Accepting this request adds location-time interval $(0, (w - 2\tau, w + 2\tau))$ to $EZ(r_6+)$. The size of that location-time interval is 4τ. Since there is no Exclusion Zone at 1, by Corollary 7.8.1, OPT can only accept requests starting between

$t-2\tau$ and $t+2\tau$ at location 0. Since the first two requests are at location 0, while OPT's servers are at location 1, all four of those require an empty movement. This means OPT serves four requests for a profit of $4p-4c$. The requests causing this Exclusion Zone set unites AG's servers at location 1, with OPT's servers also at location 1. Therefore, this Exclusion Zone set is infinitely repeatable in tandem with Lemma 7.15.1's Cases 3 and 4. However, even when combining the ratio of this case with the ratio of Lemma 7.15.1's Cases 3 and 4, we find the performance of OPT compared to AG to be lower than 4.
Case 2 For the second possible follow-up Exclusion Zone set, we consider a pair of requests $r_5 = r_6 = (e, w, 1)$, with $e > b$, and $w > t$. Note that e and w are only specified with a relation to b and t. This makes r_5, r_6 a pair of general requests at location 1 released and served after r_1, r_2. Introducing this pair of requests, we find that AG accepts one of them, gaining a profit of p. Accepting this request adds location-time interval $(1, (w-2\tau, w+2\tau))$ to $EZ(r_6+)$. The size of that location-time interval is 4τ. Since there is no Exclusion Zone at 0, by Corollary 7.8.1, OPT can only accept requests starting between $t-2\tau$ and $t+2\tau$ at location 1. Since the first two requests are at location 1, and OPT's servers are at location 1, only two of those require an empty movement. This means OPT serves four requests for a profit of $4p-2c$. The requests causing this Exclusion Zone set unites AG's servers at location 0, with OPT's servers also at location 0. Therefore, this Exclusion Zone set is infinitely repeatable in tandem with Lemma 7.15.1's Cases 3 and 4. However, for this combined Exclusion Zone set, we still find the performance of OPT compared to AG to be lower than 4.
Case 3 For the third possible follow-up Exclusion Zone set is created by releasing a pair of requests $r_5 = r_6 = (e, w, 0)$, with $e > b$, and $w > t$, then following up with a pair of requests $r_7 = r_8 = (f, x, 1)$ with $b < e \leq f$ and $t < x-\tau < w < x+\tau$. Note that e and w are only specified with a relation to b and t. This makes r_5, r_6 a pair of general requests at location 0 released and served after r_1, r_2. Also, note that f and x are only specified with a relation to e, w, b and t. This makes r_7, r_8 a pair of general requests crossing r_5, r_6. We first discuss the impact of the first pair of requests. Introducing this pair of requests, we find that AG accepts one of them, gaining a profit of p. Accepting this request adds location-time interval $(0, (w-2\tau, w+2\tau))$ to $EZ(r_6+)$. The size of that location-time interval is 4τ. Since there is no Exclusion Zone at 1, by Corollary 7.8.1, OPT can only accept requests starting between $t-2\tau$ and $t+2\tau$ at location 0. Since the first two requests are at location 0, while OPT's servers are at location 1, all four of those require an empty movement.

After this, we consider the pair of requests $r_7 = r_8 = (f, x, 1)$. Introducing this pair of requests, we find that AG accepts one of them, gaining a profit of p. This shortens the Exclusion Zone at $(0, (w-2\tau, w+2\tau))$, and adds an extra zone to the set at $(1, (w-\tau, w+\tau))$. This zone is of size 2τ. Thus, OPT can only serve two requests from location 1. These requests do not require an empty movement, and, in fact, serve as two of the required empty movements. This means OPT serves six requests for a profit of $6p-2c$. After the requests causing this Exclusion Zone set are served, AG's servers are still at the two separate locations: one at location 0 and the other at location 1. OPT's servers stay at the same location. Therefore, this Exclusion Zone set is infinitely repeatable. Repeating this follow-up Exclusion Zone set infinitely many times we find the performance of OPT compared to AG to be $\frac{6p-2c}{2p}$. This is markedly lower than 4. Furthermore, even when combining this Exclusion Zone set with Lemma 7.15.1's Cases 3 and 4 from the starting scenario, we find the performance of OPT compared to AG to be lower than 4.
Case 4 For the fourth possible follow-up Exclusion Zone set we consider a pair of requests $r_5 = r_6 = (e, w, 1)$, with $e > b$, and $w > t$, then following up with a pair of requests $r_7 = r_8 = (f, x, 0)$ with $b < e \leq f$ and $t < x-\tau < w < x+\tau$. Note that e and w are only specified with a relation to b and t. This makes r_5, r_6 a pair of general requests at location 0 released and served after r_1, r_2. Also, note that f and x are only specified with a relation to e, w, b and t. This makes r_7, r_8 a pair of general requests crossing r_5, r_6. We first discuss the impact of the first pair of requests. Introducing the first pair of requests, we find that AG accepts one of them, gaining a profit of p. Accepting this request adds location-time interval $(1, (w-2\tau, w+2\tau))$ to $EZ(r_6+)$. The size of that location-time interval is 4τ. Since there is no Exclusion Zone at 0, by Corollary 7.8.1, OPT can only accept requests starting between $t-2\tau$ and $t+2\tau$ at location 1. Since the first two requests are at location 1, and OPT's servers are at location 1, only two of these require an empty movement.

After this, we consider the pair of requests $r_7 = r_8 = (f, x, 0)$. Introducing this pair of requests, we find that AG accepts one of them, gaining a profit of p. This shortens the Exclusion Zone at $(1, (w-2\tau, w+2\tau))$, and adds an extra zone to the set at $(0, (w-\tau, w+\tau))$. This zone is of size 2τ. Thus, OPT can only serve two requests from location 0. These requests do not require an empty movement, and, in fact, serve as two of the required empty movements. This means OPT serves six requests for a profit of $6p$. After the requests causing this Exclusion Zone set are served, AG's servers are still at the two separate locations: one at location 0 and the other at location 1. OPT's servers stay at the same location. Therefore, this Exclusion Zone set is infinitely repeatable. Repeating this follow-up Exclusion Zone set infinitely many times we find the performance of OPT compared to AG to be $\frac{6p}{2p} = 3$. This is markedly lower than 4. Furthermore, even when combining this Exclusion Zone set with Lemma 7.15.1's Cases 3 and 4, we find the performance of OPT compared to AG to be lower than 4. □

This was a distinction of all possible releases by an adversary from the follow-up scenario reached after Lemma 7.15.1's Cases 3 and 4. With that, we end the case distinction on this phase. From this phase, we make the following observations.

Corollary 7.15.2.2 An adversary cannot release a series of requests starting from the follow-up scenario such that the performance ratio of AG and OPT rises above 4.

Corollary 7.15.2.3 An adversary cannot release a series of requests starting from follow-up scenario such that the formation the servers at the end of the follow-up scenario are left in a scenario other than the starting scenario, or the follow up scenario.

Having explored all four follow-up Exclusion Zone sets, we move on to the third and final phase. In this third phase, we focus on the line-skipping case. Specifically, we focus on line-skipping Lemma 7.15.1's Case 1, in which two non-distant requests were accepted by AG. This ensures that both of AG's servers need to be at location 0 come time t. This causes line-skipping requests with starting location 0 to require an empty movement to be served, unless there is another request accepted starting at location 1 in between. We will refer to this new scenario as the *line-skipping scenario.*

To explain why we do not consider Lemma 7.15.1's Cases 2, 3 and 4, we prove that line-skipping a scenario requiring a distant request - such as these cases - does not create a performance ratio exceeding 4.

Observation 7.15.3.1 A request line-skipping a scenario in which an empty movement is required causes an Exclusion Zone in which the performance ratio of AG compared to OPT does not exceed 4.

Proof By the premise of the Lemma, one server serves a request starting at a distant location. If a request is released prior to that distant request, to be served from a non-distant location, that would replace the empty movement. This replaced empty movement would cause AG's profit to rise by $p + c$, whilst only creating an Exclusion Zone of size 4τ. It is not possible to gain more than $4p + 4c$ profit in such an Exclusion Zone. Therefore, the performance ratio of AG compared to OPT on an Exclusion Zone created by a non-distant request line-skipping a scenario in which an empty movement is required would not rise above 4. If the line-skipping request is distant instead, that creates an Exclusion Zone of size 4τ at a distant location. We note that all servers are at the same location when this request needs to be served. If AG's servers were not at the same location, the request would not require an empty movement. If OPT's severs were not with AG's, an adversary could release a pair of requests to be served in the location-based Exclusion Zone to unite all servers at the same location. This implies that all requests to be served at the distant location are ultimately distant, providing no more than $p - c$ profit per request. Therefore, the performance ratio of AG compared to OPT on an Exclusion Zone created by a distant request line-skipping a scenario in which an empty movement is required would not rise above 4 □.

Observation 7.15.3.2 If the two requests with the lowest starting time accepted by AG start at location 0, they do not require an empty movement, and can thus be served as a pair.

By Observation 7.15.3.1, line-skipping Lemma 7.15.1's Cases 2, 3 and 4 does not create a performance ratio between AG and OPT exceeding 4. Since we intend to prove an upper bound exceeding 4, we need only focus on Lemma 7.15.1's Case 1. Next, we note the existence of a location-based Exclusion Zone of size τ at location 1 from the time at which the Exclusion Zones are checked, until τ time after that. To explain this, we refer to Observation 7.15.1.1. We now continue the enumeration of all possible releases an adversary might do.

Lemma 7.15.3 Case distinction starting from line-skipping scenario: all servers at one location, Lemma 7.15.1's Case 1 planned.

Proof We enumerate three options in the line-skipping scenario: either AG accepts a pair of requests from location 0, or AG accepts one request from location 1, or AG accepts one request from location 1, then one crossing request from location 0. Exclusion Zones are created for each request, unless both servers could serve that request without performing an empty movement first. By Observation 7.15.3.2 we find that, since we line-skip all other requests, a pair of non-distant requests can be accepted without first having to perform an empty movement. This explains why we have no case that accepts just one non-distant line-skipping request. Note that Lemma 7.15.1's case 1 demands that both of AG's servers are at location 0 at time t, and that those servers are at location 0 currently. Releasing any sequence of requests other than the ones we enumerated would either release requests in an Exclusion Zone (causing them to be rejected by definition 7.3), or require such a large gap between different pairs of requests, that their Exclusion Zones do not overlap, and therefore do not relate to each other. If the Exclusion Zones did overlap, that would violate Lemma 7.13. Such instances violating Lemma 7.13 do not show the worst performance ratio on any settings, and are, therefore, irrelevant to this proof.

Case 1 For the first possible Exclusion Zone set line-skipping Lemma 7.15.1's Case 1, we consider a pair of requests $r_5 = r_6 = (e, w, 0)$, with $e > b$, but $w < t$. Note that e and w are only specified with a relation to b and t. This makes r_5, r_6 a pair of general requests at location 1 released and served after r_1, r_2. Introducing this pair of requests, we find that AG accepts both of them, gaining a profit of $2p - 2c$, because the servers serving these request need to perform an empty movement in order to serve the requests accepted in Lemma 7.15.1's Case 1. Accepting these requests add location-time interval $(0, (w - 2\tau, w + 2\tau))$ and $(1, (w - \tau, w + \tau))$ to $EZ(r_6+)$. The size of those location-time interval is 6τ. Since there are Exclusion Zones at both 0 and 1, by Corollary 7.8.1, this allows OPT to serve six alternating requests. This means OPT serves at most six requests for a profit of $6p$, ending its servers at location 1. However, since OPT has to serve its accepted requests from Lemma 7.15.1's Case 1 after this case, and the requests in Lemma 7.15.1's Case 1 start at location 0, OPT has to perform two empty movements to serve all accepted requests.
Case 2 For the second possible Exclusion Zone set line-skipping Lemma 7.15.1's Case 1, we consider a pair of requests $r_5 = r_6 = (e, w, 1)$, with $e > b$, but $w < t$. Note that e and w are only specified with a relation to b and t. This makes r_5, r_6 a pair of general requests at location 1 released and served after r_1, r_2. Introducing this pair of requests, we find that AG accepts one of them, gaining a profit of $p - c$. Accepting this request adds location-time interval $(1, (w - 2\tau, w + 2\tau))$ to $EZ(r_6+)$. The size of that location-time interval is 4τ. Since there is no Exclusion Zone at 0, by Corollary 7.8.1, OPT can only accept requests starting between $w - 2\tau$ and $w + 2\tau$ at location 1. Since the first two requests are at location 1, while OPT's servers are at location 0, all four of those require an empty movement. This means OPT serves four requests for a profit of $4p - 4c$. This case ends with all of OPT's servers and all of ALG's server's at location 0. Therefore, this Exclusion Zone set is infinitely repeatable. However, neither repeating this case, nor combining this set of Exclusion Zones with Lemma 7.15.1's Case 1 causes the performance of OPT compared to AG to exceed 4.
Case 3 For the third and final possible Exclusion Zone set line-skipping Lemma 7.15.1's Case 1, we consider a pair of requests $r_5 = r_6 = (e, w, 1)$, with $e > b$, but $w < t$, then releasing another pair of requests $r_7 = r_8 = (f, x, 0)$, with $f \geq e$, and $w - \tau < x < w + \tau < t$. Note that e and w are only specified with a relation to b and t. This makes r_5, r_6 a pair of general requests at location 0 released and served after r_1, r_2. Also, note that f and x are only specified with a relation to e, w, b and t. This makes r_7, r_8 a pair of general requests crossing r_5, r_6. We first discuss the impact of the first pair of requests. Introducing r_5, r_6 first, we find that AG accepts one of them, gaining a profit of $p - c$. Accepting this request adds location-time interval $(1, (w - 2\tau, w + 2\tau))$ to $EZ(r_6+)$. The size of that location-time interval is 4τ. Since there is no Exclusion Zone at 0, by Corollary 7.8.1, OPT can only accept requests starting between $w - 2\tau$ and $w + 2\tau$ at location 1. Since the first two requests are at location 1, while OPT's servers are at location 0, all four of those require an empty movement.

After this, we consider the pair r_7, r_8. Introducing this pair of requests, we find that AG accepts one of them, gaining a profit of $p - c$, because serving this request requires an empty movement to serve the requests accepted in Lemma 7.15.1's Case 1. This shortens the Exclusion Zone at $(1, (w - 2\tau, w + 2\tau))$, and adds an extra zone to the set at $(0, (w - \tau, w + \tau))$. This zone is of size 2τ. Thus, by Corollary 7.8.1, OPT can only serve two requests from location 0. These requests do not require an empty movement, and, in fact, serve as two of the required empty movements to serve the requests accepted at location 1. This means OPT serves six requests for a profit of $6p - 2c$. Since OPT's servers end at location 0, no additional empty movements are required to serve the requests accepted in starting scenario case 1. We note that the requests causing this Exclusion Zone set moves AG's servers, with one at location 0, and another at location 1, while OPT's servers are at location 0. We discuss the implications of this new server formation in a separate Lemma. □

This was a distinction of all possible releases by an adversary line-skipping the scenario described in Lemma 7.15.1's Case 1. Throughout the Lemma, we have encountered two new scenarios. One is Lemma 7.15.3's Case 1 followed up by Lemma 7.15.1's Case 1. The other is Lemma 7.15.3's Case 3, followed up by Lemma 7.16.1's Case 1. In these cases, we need to explore what happens when a set of requests is released either between or before the two cases.

Observation 7.15.3.3 Further analysis of Lemma 7.15.3's Case 1.

Proof By Observation 7.15.3.2, we now note that AG reacts to non-distant requests line-skipping to the front of the line, line-skipping all other accepted requests, in the same way. Therefore, an adversary can use this trick as many times as is possible between υ and λ. This trick creates an Exclusion Zone of size 4τ at location 0. Therefore, the adversary only has to release a pair of requests every 4τ to effectively lock AG out of using location 0 until the requests accepted in the starting scenario are served. Those starting scenario requests can be accepted as long as υ before they need to be served. This means that this trick is repeatable $\lfloor \frac{\upsilon-\lambda}{4\tau} \rfloor$ times. We make one more observation here. Currently the adversary releases a Zig-Zag gadget to have OPT gain maximum profit over AG. However, if there is less than 4τ time between the end of the penultimate Exclusion Zone, and the Exclusion Zones caused by the starting scenario, performing a full Zig-Zag gadget is impossible. Therefore, if $(\upsilon - \lambda)\%4\tau > 2\tau$, instead of not using 2τ time, the adversary could introduce one more pair for AG, accompanied by a Zig-gadget for OPT. That way, OPT could gain a little more profit over AG. Furthermore, we take note that Lemma 7.15.1's Case 1 can only be implemented if $t - b < \upsilon - \tau$. This takes a further τ time out of $\upsilon - \lambda$. To summarize, after releasing case one concerning the starting scenario at $\upsilon - \tau$, the adversary can repeatedly release paired requests, 4τ apart, (which AG accepts) to then give OPT access to Zig-Zag gadgets, 4τ apart, and possibly one Zig gadget, at the final 2τ. Note that both the pairs accepted by AG and the Zig-Zag gadgets accepted by OPT require a pair of empty movements to be served together. □

Observation 7.15.3.4 Further analysis of Lemma 7.15.3's Case 3.

Proof By Observation 7.15.3.1, we note that line-skipping Lemma 7.15.3's Case 3 does not result in an improvement to the performance ratio between AG and OPT. Hence, we need to focus on filling the time between Lemma 7.15.3's Case 3 and Lemma 7.15.1's Case 1. We note that Lemma 7.15.3's Case 3 is not repeatable, since AG's servers are split between location 0 and location 1 after serving Lemma 7.15.3's Case 3. From there, we note that releasing any other request between Lemma 7.15.3's Case 3 and Lemma 7.15.1's Case 1 creates a one-sided Exclusion Zone of size 4τ. In that Exclusion Zone, OPT cannot make a profit greater than $4p - 4c$, where AG makes $p - c$ profit or better. This would pull the overall ratio closer to 4, which is lower than the performance ratio just employing Lemma 7.15.3's Case 3 and Lemma 7.15.1's Case 1. □

We have now finished exploring all possible releases by an adversary from the line-skipping scenario, reached after Lemma 7.15.1's Case 1 was released (but served before that case starts). We have found closing arguments for all possible releases by an adversary, starting from the starting scenario, following-up on that scenario, or line-skipping that scenario. Having enumerated everything, we provide the Competitive Ratio of AG.

Theorem 7.15 Algorithm AG is $\max\{4, \frac{6p+4p\lceil\frac{\upsilon-\lambda-3\tau}{4\tau}\rceil+(2p-2c)\lceil\frac{\upsilon-\lambda-5\tau}{4\tau}\rceil}{2p+(2p-2c)\lceil\frac{\upsilon-\lambda-3\tau}{4\tau}\rceil}\}$ competitive for the 2S2L-V problem with arbitrary c, and $\upsilon - \lambda > \tau$.

Proof We proof this Theorem by enumerating all possible releases of an adversary that do not violate Lemma 7.13. We have done this in Lemmas 7.15.1, 7.15.2, 7.15.3 and Observations 7.15.3.3 and 7.15.3.4.

Combining our findings on Lemma 7.15.1's Case 1 with our findings on Lemma 7.15.3's Case 1 and Observation 7.15.3.3, we conclude the following. AG gains $2p$ profit from the the requests released by the adversary in Lemma 7.15.1's Case 1, plus an additional $2p-2c$ per pair of requests released the adversary line-skipping Lemma 7.15.1's Case 1. These pairs are released $\lceil\frac{\upsilon-\lambda-3\tau}{4\tau}\rceil$ times. In the Exclusion Zones created by these request pairs, OPT can accept the following. For the first case of the starting scenario, OPT gains $6p$ profit. Furthermore, for the $\lceil\frac{\upsilon-\lambda-3\tau}{4\tau}\rceil$ pairs accepted by AG, OPT can gain an additional

$4p$ profit. Finally, for $\lceil \frac{\upsilon-\lambda-5\tau}{4\tau} \rceil$ of these request pairs accepted by AG, OPT gains an additional $2p-2c$ profit. Putting all that together, we find a competitive ratio of $\frac{6p+4p\lceil \frac{\upsilon-\lambda-3\tau}{4\tau} \rceil+(2p-2c)\lceil \frac{\upsilon-\lambda-5\tau}{4\tau} \rceil}{2p+(2p-2c)\lceil \frac{\upsilon-\lambda-3\tau}{4\tau} \rceil}$. We have also found multiple cases that were 4-competitive. Since the adversary is in charge of deciding what case is released, it should release the case with the highest performance ratio. We conclude that AG has a competitive ratio of $\max\{4, \frac{6p+4p\lceil \frac{\upsilon-\lambda-3\tau}{4\tau} \rceil+(2p-2c)\lceil \frac{\upsilon-\lambda-5\tau}{4\tau} \rceil}{2p+(2p-2c)\lceil \frac{\upsilon-\lambda-3\tau}{4\tau} \rceil}\}$. □

Having proven this upper bound, we find the settings where the competitive upper bound AG achieves is tight. The range in which AG's performance is tight might not be large, but it is very important. Combining the power of AG, RZA and GA, we have algorithms achieving tight upper bounds for nearly all settings. The only setting that we have yet to find a tight upper bound for, is $\upsilon - \lambda > 7\tau$ and high c. We introduce an algorithm fill this final gap in the next section.

7.4 Greedy Restriction Zones (GRZA)

In this section, we introduce the *Greedy Restriction Zones Algorithm (GRZA)*. As with all algorithms in this chapter, GRZA is a part of the Exclusionary Family. This algorithm implements Exclusion Zones very similar to RZA, with one major difference. GRZA allows the first non-distant request pair to be accepted as a pair, whereas RZA does not. This gives GRZA a slightly better performance ratio than RZA from the starting scenario (where both of the algorithms' servers are at location 0), and a very consistent performance ratio at that. However, that consistency make GRZA a worse choice than other algorithms we discussed thus far when it comes to settings such as short booking interval size, or low empty movement cost.

In this section, we first introduce GRZ. GRZ is the method for composing the Exclusion Zones used to define GRZA. By first providing the Method for creating the appropriate Exclusion Zones, we show that GRZA is part of the Exclusionary Family. This gives us access to the Lemmas and Observations we used to find the competitive ratios of RZA and AG before this. Using these Lemmas and Observations, we prove the competitive ratio of GRZA, showing that it is $\frac{12p-2c}{3p-c}$, for $\upsilon - \lambda > 4\tau$. This shows that GRZA performs as well as the problem lower bound for $\upsilon - \lambda > 9\tau$ and $\frac{5-\sqrt{13}}{2}p \leq c \leq p$.

Just as with RZA, we show the Method of deciding Exclusion Zones for GRZA, and define GRZA upon those. That way, we automatically prove that GRZA is part of the Exclusionary Family. Before we show this Method, however, we first clarify what these Exclusion Zones stand for. The Method GRZ is comprised of three main statements. The first if-statement checks whether the accepted request is such that both servers can serve it without accruing an empty movement cost. If this is the case, this accepted request does not cause a new request-based Exclusion Zone. This is the major difference from RZA. The second and third if-statement for Exclusion Zones are there to ensure that Observation 7.6 is not violated. Note that the third if-statement and the accompanying else-statement are the exact same as the first if-statement in Method RZ. The behaviour caused by this if-statement is equivalent to that of RZA. Thus, these request-based exclusion zones function similarly to those in RZA. Finally, the checks for location-based exclusion zones are put in place using the fourth and fifth if statements in GRZ. The result is the following Method:

Algorithm 7 Method $GRZ(R^A, b)$

```
Input: Set of requests $R^A$ accepted by algorithm A, with $b > b_r \forall r \in R^A$, sorted by increasing $b$.
Output: Set of location-time intervals $EZ(b)$
begin
    $EZ(b) = \emptyset$
    Request-based Exclusion Zones:
    foreach $r \in R^A$ do
        if it is feasible for both servers to be at location $l_r$ at time $t_r$, without needing to perform an extra empty movement beforehand, and there is no request $j \in R^A$ with $b_j < b_r$ such that $l_j = l_r$ and $t_r > t_j$ then
            $EZ(b) = EZ(b)$
        else if there is a request $j \in R^A$ with $b_j < b_r$ such that $l_j = l_r$ and $|t_r - t_{r_j}| < 2\tau$: then
            $EZ(b) = EZ(b) \cup (l_r, (\max\{t_j, t_r\} - 2\tau, \min\{t_j, t_r\} + 2\tau)) \cup (\bar{l}_j, (\max\{t_j, t_r\} - \tau, \min\{t_j, t_r\} + \tau))$
        else if there is a request $j \in R^A$ with $b_j < b_r$ such that $\bar{l}_j = l_r$ and $|t_r - t_{r_i}| < \tau$: then
            $EZ(b) = EZ(b) \setminus ((l_j, (t_j - 2\tau, t_r - \tau)) \cup (l_j, (t_r + \tau, t_j + 2\tau))) \cup (l_r, (t_j - \tau, t_j + \tau))$
        else
            $EZ(b) = EZ(b) \cup (l_r, (t_r - 2\tau, t_r + 2\tau))$
        end
    end
    Location-based Exclusion Zones:
    if There exists no schedule such that any server is at location $l$ at time $b$ then
        if There exists a schedule such that any server is at location $l$ at time $c$ with $b < c < b + \tau$ then
            $EZ(b) = EZ(b) \cup (l, (b, c))$
        else
            $EZ(b) = EZ(b) \cup (l, (b, b + \tau))$
        end
end
```

Using this method, we construct the Greedy Restriction Zones Algorithm by following the Exclusionary Framework. We define GRZA as follows:

Algorithm 8 Algorithm $GRZA(\mathcal{I})$

```
Input: Instance $\mathcal{I}$ of requests $r$ arriving by increasing $b_r$.
Output: Set of requests accepted by GRZ $R^A$
begin
    $R^A = \emptyset$
    foreach $r = (b_r, t_r, l_r) \in \mathcal{I}$ do
        $EZ(r-) = GRZ(R^A, b_r)$
        if $(l_r, t_r) \notin EZ(r-)$ then
            $R^A = R^A \cup r$
        else
            reject $r$
        end
    end
end
```

Observe that Algorithm GRZA perfectly follows $EF(\mathcal{I}, RZ(R, b)$. Therefore, we conclude that Algorithm GRZA is a part of the Exclusionary Family. That means Observations, Corollaries and Lemmas 7.5 − 7.13 hold on GRZA. We now analyze the upper bound of this Algorithm on the unbounded instance.

We prove this upper bound by systematically enumerating all possible Exclusion Zone sets. For every Exclusion Zone set, we observe the profit GRZA gains when creating that Exclusion Zone set, the size of the created Exclusion Zone set, and the repeatability of that Exclusion Zone set. We use this data to find the Exclusion Zone set that causes the greatest ratio between size and profit for GRZA, which, according to Observation 7.12, leads us to the competitive ratio. In order to find this Exclusion Zone set with greatest ratio between size and profit, we enumerate all cases that warrant a distinct reaction from

GRZA. We perform this enumeration in three phases. In the first phase, we consider different series of requests from the starting position in which all of GRZA's servers and OPT's servers are at location 0. In the second phase, we consider possible follow-ups to the series released in the starting position, and how well GRZA performs compared to OPT on the zone sets created by these follow-up requests. In the third and final phase, we instead consider the impact of requests line skipping the starting situation. We enumerate in phases, because that allows us to eliminate some cases with a low performance ratio early on. This cuts down on the amount of cases we need to check.

Observation 7.16.1.1 No request released in the location-based Exclusion Zone is acceptable from the starting position, if no other request is released first.

Proof By Algorithm 7, there is a location-based Exclusion Zones at the starting position. This location-based Exclusion Zone is one of size τ at location 1. Because a movement from location 0 to location 1 takes τ time, and because all servers start at location 0, it is impossible for OPT to accept requests to be served in that Exclusion Zone, without other requests being released first. Hence, the location-based Exclusion Zone is inaccessible from the starting position without a different request being released first. □

Lemma 7.16.1 Case distinction starting from starting position: all servers at one location.

Proof We enumerate all possible reactions to requests released by an adversary in the starting scenario. This enumerates three cases: GRZA accepting a pair of non-distant requests starting at 0, GRZA accepting a single distant request at location 1, and GRZA accepting a pair of crossing requests, with the first request arriving at location 1. Other cases, that would cause an interaction between Exclusion Zones caused by all released and accepted requests after the starting scenario, would violate Lemma 7.13. Violating Lemma 7.13 shows that such cases do not maximize the performance ratio. Therefore, we can discard such cases. We now bring special attention to the fact that there is no case with a pair of crossing requests, with the first request accepted at 0. This is because we only consider the instance in which every request is released alongside an identical paired request. If we release a pair of requests at location 0 first, GRZA will, by Observation 7.5, accept both of these requests, putting us firmly in the first case instead.

Case 1 For the first possible Exclusion Zone set from the starting scenario, we consider a pair of requests $r_1 = r_2 = (b, t, 0)$. Note that we do not require a specific b or t for this case, representing a general pair of non-distant requests. Introducing this pair of requests, we find that GRZA accepts both of them, gaining a profit of $2p$. This adds location-time interval $(0, (t-2\tau, t+2\tau))$ and $(1, (t-\tau, t+\tau))$ to $EZ(r_2+)$. The size of those combined location-time intervals is 6τ. By Corollary 7.8.1, this is all the time OPT has to accept requests. Since there is an Exclusion Zone at both locations, OPT can accept requests starting between $t-2\tau$ and $t+2\tau$ at location 0, and between $t-\tau$ and $t+\tau$ at location 1. In this time, OPT can serve six requests: four starting from 0, and two more starting from 1. None of these requests require an empty movement to be served. Therefore, serving these six requests, OPT gains a profit of $6p$. The requests causing this Exclusion Zone set moves all servers from location 0 to location 1. Therefore, this Exclusion Zone set is infinitely repeatable. For this Exclusion Zone set, we find the performance of OPT compared to GRZA to be $\frac{6p}{2p}$.
Case 2 For the second possible Exclusion Zone set from the starting scenario, we consider a pair of requests $r_1 = r_2 = (b, t, 1)$. Note that we do not require a specific b or t for this case, representing a general pair of distant requests. Introducing this pair of requests, we find that GRZA accepts one of them, gaining a profit of $p-c$. This adds location-time interval $(1, (t-2\tau, t+2\tau))$ to $EZ(r_2+)$. The size of that location-time interval is 4τ. Since there is no Exclusion Zone at 0, by Corollary 7.8.1, OPT can only accept requests starting between $t-2\tau$ and $t+2\tau$ at location 1. Since requests need to alternate to not require empty movements to be served, OPT can serve at most four requests in this Exclusion Zone. Since the first requests already require an empty movement, all four accepted requests require empty movements, noting a profit of $4p-4c$ for OPT. Since all servers end where they started this Exclusion Zone set, this Exclusion Zone set is infinitely repeatable. For this Exclusion Zone set, we find the performance of OPT compared to GRZA to be $\frac{4p-4c}{p-c} = 4$.
Case 3 For the third possible Exclusion Zone set from the starting scenario, we consider a pair of requests $r_1 = r_2 = (b, t, 1)$, followed by the release of another pair of crossing requests $r_3 = r_4 = (d, u, 0)$, with $d \geq b$ and $t-\tau < u < t+\tau$. Note that we do not require a specific b or t for this case, and that d and u are only relative to b and t, representing a general pair of distant requests followed by a general

pair of crossing non-distant requests. We start this analysis by looking at the impact of the adversary's first release: r_1 and r_2. Introducing this first pair of requests, we find that GRZA accepts one of them, gaining a profit of $p - c$. The first accepted request adds location-time interval $(1, (t - 2\tau, t + 2\tau))$ to $EZ(r_2+)$. The size of that location-time interval is 4τ. Since there is no Exclusion Zone at 0, by Corollary 7.8.1, OPT only accepts requests starting between $t - 2\tau$ and $t + 2\tau$ at location 1. Since requests need to alternate to not require empty movements to be served, OPT can serve at most 4 requests in this Exclusion Zone. Since the first two requests are at location 1, while OPT's servers are at location 0, all four of those require an empty movement. After this, we analyze the impact the adversary's subsequent release of r_3, r_4 has. Introducing this pair of requests, we find that GRZA accepts one of them, gaining a profit of p. This shortens the Exclusion Zone at $(1, (t - 2\tau, t + 2\tau))$, and adds an extra zone to the set at $(0, (t - \tau, t + \tau))$. This zone is of size 2τ. Thus, since OPT only serves requests starting in Exclusion Zones by Corollary 7.8.1, OPT can only serve 2 requests from location 0. These requests do not require an empty movement, and, in fact, serve as two of the required empty movements. This means OPT serves six requests for a profit of $6p - 2c$. The requests causing this Exclusion Zone set causes GRZA's servers to be spread between location 0 and location 1. Therefore, this Exclusion Zone set is not infinitely repeatable. For this Exclusion Zone set, we find the performance of OPT compared to GRZA to be $\frac{6p-2c}{2p-c}$. □

This was a distinction of all possible releases by an adversary from the starting scenario. Note that, in order to continue releasing different requests, any adversary first has to release one of these three cases. With that, we end the case distinction on this phase. From the case distinction on phase one, we make the following observations.

Corollary 7.16.1.2 The only formation of servers that the starting scenarios can result in, that is not symmetric to the starting scenario, is a scenario with one of GRZA's servers at location 0 and location 1 each, and OPT's servers together at one location.

Corollary 7.16.1.3 Only one case of the starting scenario requires both of GRZA's servers to be at location 0 at time t, the starting time of the first request.

Corollary 7.16.1.4 Two cases of the starting scenario require one of GRZA's servers to be at location 1 at time t, the starting time of the first request.

Having explored all three possible Exclusion Zone sets from the starting positions of the servers, we move on to the second phase. In this second phase, we consider starting from the different configurations of servers we encountered. By Corollary 7.16.1.2, we have only seen one different configuration of servers after the starting configuration: starting scenario Case 3 causes a configuration where one of GRZA's servers is at location 0, with the other server at location 1, and OPT's servers together at one location. Without loss of generality, we assume location 1 to be the location of OPT's servers. We refer to this new scenario as *the follow-up scenario.*

Observation 7.16.2.1 There is no location-based Exclusion Zone in the follow-up scenario.

Proof Since GRZA has a server at both location 0 and 1, any request released at 0 or 1 can be acceptable. Therefore, there is no need for a location-based Exclusion Zone. □

Lemma 7.16.2 Case distinction starting from follow-up scenario: one GRZA server at 0, one GRZA server at 1, and OPT servers united at one location.

Proof We enumerate all possible reactions to requests released by an adversary in the follow-up scenario. This requires four cases: either GRZA accepts one request from location 0, or GRZA accepts one request from location 1, or GRZA accepts one request from location 0, then one from location 1, or GRZA accepts one request from location 1, then one from location 0. Since both of GRZA's servers are at different locations at the start of this scenario, we can conclude that no pair of requests can be accepted without an empty movement. That means that any request accepted by GRZA in the follow-up scenario creates a new Exclusion Zone. Therefore, other cases besides the ones enumerated require a substantial gap between accepted requests, from where we would be able to analyze before and after the gap separately. If this gap were smaller, the Exclusion Zones on both sides of the gap would overlap, violating Lemma 7.13. Thus, such cases need not be handled in this Lemma.

Case 1 For the first possible follow-up Exclusion Zone set, we consider a pair of requests $r_5 = r_6 = (e, w, 0)$, with $e > b$, and $w > t$. Note that e and w are only specified with a relation to b and t. This makes r_5, r_6 a pair of general requests at location 0 released and served after r_1, r_2. Introducing this pair of requests, we find that GRZA accepts one of them, gaining a profit of p. Accepting this request adds location-time interval $(0, (w - 2\tau, w + 2\tau))$ to $EZ(r_6+)$. The size of that location-time interval is 4τ. Since there is no Exclusion Zone at 1, by Corollary 7.8.1, OPT can only accept requests starting between $t-2\tau$ and $t+2\tau$ at location 0. Since the first two requests are at location 0, while OPT's servers are at location 1, all four of those require an empty movement. This means OPT serves four requests for a profit of $4p - 4c$. The requests causing this Exclusion Zone set unites GRZA's servers at location 1, with OPT's servers also at location 1. Therefore, this Exclusion Zone set is infinitely repeatable in tandem with Lemma 7.16.1's Case 3. However, even when combining the ratio of this case with the ratio of Lemma 7.16.1's case 3, we find the performance of OPT compared to GRZA to be lower than 4.
Case 2 For the second possible follow-up Exclusion Zone set, we consider a pair of requests $r_5 = r_6 = (e, w, 1)$, with $e > b$, and $w > t$. Note that e and w are only specified with a relation to b and t. This makes r_5, r_6 a pair of general requests at location 1 released and served after r_1, r_2. Introducing this pair of requests, we find that GRZA accepts one of them, gaining a profit of p. Accepting this request adds location-time interval $(1, (w - 2\tau, w + 2\tau))$ to $EZ(r_6+)$. The size of that location-time interval is 4τ. Since there is no Exclusion Zone at 0, by Corollary 7.8.1, OPT can only accept requests starting between $t-2\tau$ and $t+2\tau$ at location 1. Since the first two requests are at location 1, and OPT's servers are at location 1, only two of those require an empty movement. This means OPT serves four requests for a profit of $4p - 2c$. The requests causing this Exclusion Zone set unites GRZA's servers at location 0, with OPT's servers also at location 0. Therefore, this Exclusion Zone set is infinitely repeatable in tandem with Lemma 7.16.1's Case 3. However, for this combined Exclusion Zone set, we still find the performance of OPT compared to GRZA to be lower than 4.
Case 3 For the third possible follow-up Exclusion Zone set is created by releasing a pair of requests $r_5 = r_6 = (e, w, 0)$, with $e > b$, and $w > t$, then following up with a pair of requests $r_7 = r_8 = (f, x, 1)$ with $b < e \leq f$ and $t < x - \tau < w < x + \tau$. Note that e and w are only specified with a relation to b and t. This makes r_5, r_6 a pair of general requests at location 0 released and served after r_1, r_2. Also, note that f and x are only specified with a relation to e, w, b and t. This makes r_7, r_8 a pair of general requests crossing r_5, r_6. We first discuss the impact of the first pair of requests. Introducing this pair of requests, we find that GRZA accepts one of them, gaining a profit of p. Accepting this request adds location-time interval $(0, (w - 2\tau, w + 2\tau))$ to $EZ(r_6+)$. The size of that location-time interval is 4τ. Since there is no Exclusion Zone at 1, by Corollary 7.8.1, OPT can only accept requests starting between $t - 2\tau$ and $t + 2\tau$ at location 0. Since the first two requests are at location 0, while OPT's servers are at location 1, all four of those require an empty movement.

After this, we consider the pair of requests $r_7 = r_8 = (f, x, 1)$. Introducing this pair of requests, we find that GRZA accepts one of them, gaining a profit of p. This shortens the Exclusion Zone at $(0, (w - 2\tau, w + 2\tau))$, and adds an extra zone to the set at $(1, (w - \tau, w + \tau))$. This zone is of size 2τ. Thus, OPT can only serve two requests from location 1. These requests do not require an empty movement, and, in fact, serve as two of the required empty movements. This means OPT serves six requests for a profit of $6p - 2c$. After the requests causing this Exclusion Zone set are served, GRZA's servers are still at the two separate locations: one at location 0 and the other at location 1. OPT's servers stay at the same location. Therefore, this Exclusion Zone set is infinitely repeatable. Repeating this follow-up Exclusion Zone set infinitely many times we find the performance of OPT compared to GRZA to be $\frac{6p-2c}{2p}$. This is markedly lower than 4. Furthermore, even when combining this Exclusion Zone set with Lemma 7.16.1's Case 3 from the starting scenario, we find the performance of OPT compared to GRZA to be lower than 4.
Case 4 For the fourth possible follow-up Exclusion Zone set we consider a pair of requests $r_5 = r_6 = (e, w, 1)$, with $e > b$, and $w > t$, then following up with a pair of requests $r_7 = r_8 = (f, x, 0)$ with $b < e \leq f$ and $t < x - \tau < w < x + \tau$. Note that e and w are only specified with a relation to b and t. This makes r_5, r_6 a pair of general requests at location 0 released and served after r_1, r_2. Also, note that f and x are only specified with a relation to e, w, b and t. This makes r_7, r_8 a pair of general requests crossing r_5, r_6. We first discuss the impact of the first pair of requests. Introducing the first pair of requests, we find that GRZA accepts one of them, gaining a profit of p. Accepting this request adds location-time interval $(1, (w - 2\tau, w + 2\tau))$ to $EZ(r_6+)$. The size of that location-time interval is 4τ. Since there is no Exclusion Zone at 0, by Corollary 7.8.1, OPT can only accept requests starting

between $t - 2\tau$ and $t + 2\tau$ at location 1. Since the first two requests are at location 1, and OPT's servers are at location 1, only two of these require an empty movement.

After this, we consider the pair of requests $r_7 = r_8 = (f, x, 0)$. Introducing this pair of requests, we find that GRZA accepts one of them, gaining a profit of p. This shortens the Exclusion Zone at $(1, (w - 2\tau, w + 2\tau))$, and adds an extra zone to the set at $(0, (w - \tau, w + \tau))$. This zone is of size 2τ. Thus, OPT can only serve two requests from location 0. These requests do not require an empty movement, and, in fact, serve as two of the required empty movements. This means OPT serves six requests for a profit of $6p$. After the requests causing this Exclusion Zone set are served, GRZA's servers are still at the two separate locations: one at location 0 and the other at location 1. OPT's servers stay at the same location. Therefore, this Exclusion Zone set is infinitely repeatable. Repeating this follow-up Exclusion Zone set infinitely many times we find the performance of OPT compared to GRZA to be $\frac{6p}{2p} = 3$. This is markedly lower than 4. Furthermore, even when combining this Exclusion Zone set with Lemma 7.16.1's Case 3, we find the performance of OPT compared to GRZA to be lower than 4. $\square$

This was a distinction of all possible releases by an adversary from the follow-up scenario reached after Lemma 7.16.1's Case 3. With that, we end the case distinction on this phase. From this phase, we make the following observations.

Corollary 7.16.2.2 An adversary cannot release a series of requests starting from the follow-up scenario such that the performance ratio of GRZA and OPT rises above 4.

Corollary 7.16.2.3 An adversary cannot release a series of requests starting from follow-up scenario such that the formation the servers at the end of the follow-up scenario are left in a scenario other than the starting scenario, or the follow up scenario.

Having explored all four follow-up Exclusion Zone sets, we move on to the third and final phase. In this third phase, we focus on the line-skipping case. Specifically, we focus on line-skipping Lemma 7.16.1's Case 1, in which two non-distant requests were accepted by GRZA. This ensures that both of GRZA's servers need to be at location 0 come time t. This causes line-skipping requests with starting location 0 to require an empty movement to be served, unless there is another request accepted starting at location 1 in between. We will refer to this new scenario as the *line-skipping scenario*.

To explain why we do not consider Lemma 7.16.1's Cases 2 and 3, we prove that line-skipping a scenario requiring a distant request - such as these cases - does not create a performance ratio exceeding 4.

Observation 7.16.3.1 A request line-skipping a scenario in which an empty movement is required causes an Exclusion Zone in which the performance ratio of GRZA compared to OPT does not exceed 4.

Proof By the premise of the Lemma, one server serves a request starting at a distant location. If a request is released prior to that distant request, to be served from a non-distant location, that would replace the empty movement. This replaced empty movement would cause GRZA's profit to rise by $p+c$, whilst only creating an Exclusion Zone of size 4τ. It is not possible to gain more than $4p + 4c$ profit in such an Exclusion Zone. Therefore, the performance ratio of GRZA compared to OPT on an Exclusion Zone created by a non-distant request line-skipping a scenario in which an empty movement is required would not rise above 4. If the line-skipping request is distant instead, that creates an Exclusion Zone of size 4τ at a distant location. We note that all servers are at the same location when this request needs to be served. If GRZA's servers were not at the same location, the request would not require an empty movement. If OPT's severs were not with GRZA's, an adversary could release a pair of requests to be served in the location-based Exclusion Zone to unite all servers at the same location. This implies that all requests to be served at the distant location are ultimately distant, providing no more than $p-c$ profit per request. Therefore, the performance ratio of GRZA compared to OPT on an Exclusion Zone created by a distant request line-skipping a scenario in which an empty movement is required would not rise above 4 $\square$.

By Observation 7.16.3.1, line-skipping Lemma 7.16.1's Cases 2 and 3 does not create a performance ratio between GRZA and OPT exceeding 4. Since we intend to prove an upper bound exceeding 4, we need only focus on Lemma 7.16.1's Case 1. Next, we note the existence of a location-based Exclusion Zone of size τ at location 1 from the time at which the Exclusion Zones are checked, until τ time after that. To explain this, we refer to Observation 7.16.1.1. We now continue the enumeration of all possible releases an adversary might do.

Lemma 7.16.3 Case distinction starting from line-skipping scenario: all servers at one location, Lemma 7.16.1's Case 1 planned.

Proof We enumerate four options in the line-skipping scenario: either GRZA accepts one request from location 0, or GRZA accepts one request from location 1, or GRZA accepts one request from location 0, then one crossing request from location 1, or GRZA accepts one request from location 1, then one crossing request from location 0. Exclusion Zones are created for each request, unless both servers could serve that request without accruing an empty movement. Note that Lemma 7.16.1's case 1 demands that both of GRZA's servers are at location 0 at time t, and that those servers are at location 0 currently. Therefore, every request that is accepted requires an extra empty movement to be served, meaning that every request in this scenario causes new Exclusion Zones. Releasing any sequence of requests other than the ones we enumerated would either release requests in an Exclusion Zone (causing them to be rejected by definition 7.3), or require such a large gap between different pairs of requests, that their Exclusion Zones do not overlap, and therefore do not relate to each other. If the Exclusion Zones did overlap, that would violate Lemma 7.13. Such instances violating Lemma 7.13 do not show the worst performance ratio on any settings, and are, therefore, irrelevant to this proof.

Case 1 For the first possible Exclusion Zone set line-skipping Lemma 7.16.1's Case 1, we consider a pair of requests $r_5 = r_6 = (e, w, 0)$, with $e > b$, but $w < t$. Note that e and w are only specified with a relation to b and t. This makes r_5, r_6 a pair of general requests at location 1 released and served after r_1, r_2. Introducing this pair of requests, we find that GRZA accepts one of them, gaining a profit of $p - c$, because the server serving this request needs to perform an empty movement in order to serve the requests accepted in Lemma 7.16.1's Case 1. Accepting this request adds location-time interval $(0, (w - 2\tau, w + 2\tau))$ to $EZ(r_6+)$. The size of that location-time interval is 4τ. While there is no request-based Exclusion zone at 1, there is a location-based Exclusion Zone. By releasing the requests at location 1 such that they are booked after OPT has started serving its requests, an adversary can give OPT access to the location-based Exclusion Zone, without releasing any additional requests. By Corollary 7.8.1, this allows OPT to serve two more requests, on top of the four it can serve in the request-based Exclusion Zone at location 0. This means OPT serves six requests for a profit of $6p$, ending its servers at location 1. However, since OPT has to serve its accepted requests from Lemma 7.16.1's Case 1 after this case, and the requests in Lemma 7.16.1's Case 1 start at location 0, OPT has to perform two empty movements to serve all accepted requests. Therefore, OPT makes a total profit of $12p - 2c$. Meanwhile, GRZA's accepted requests requires one empty movement, making a profit of $3p - c$. The requests causing this Exclusion Zone set spreads out GRZA's servers, with one at location 0, and another at location 1, with OPT's servers also at location 1. Therefore, this Exclusion Zone set is not infinitely repeatable. We discuss the implications of this new scenario in a separate Lemma. For now, we find the performance of OPT compared to GRZA to be $\frac{12p-2c}{3p-c}$.
Case 2 For the second possible Exclusion Zone set line-skipping Lemma 7.16.1's Case 1, we consider a pair of requests $r_5 = r_6 = (e, w, 1)$, with $e > b$, but $w < t$. Note that e and w are only specified with a relation to b and t. This makes r_5, r_6 a pair of general requests at location 1 released and served after r_1, r_2. Introducing this pair of requests, we find that GRZA accepts one of them, gaining a profit of $p - c$. Accepting this request adds location-time interval $(1, (w - 2\tau, w + 2\tau))$ to $EZ(r_6+)$. The size of that location-time interval is 4τ. Since there is no Exclusion Zone at 0, by Corollary 7.8.1, OPT can only accept requests starting between $w - 2\tau$ and $w + 2\tau$ at location 1. Since the first two requests are at location 1, while OPT's servers are at location 0, all four of those require an empty movement. This means OPT serves four requests for a profit of $4p - 4c$. This case ends with all of OPT's servers and all of ALG's server's at location 0. Therefore, this Exclusion Zone set is infinitely repeatable. However, neither repeating this case, nor combining this set of Exclusion Zones with Lemma 7.16.1's Case 1 causes the performance of OPT compared to GRZA to exceed 4, which does not exceed $\frac{12p-2c}{3p-c}$, either.
Case 3 For the third possible Exclusion Zone set line-skipping Lemma 7.16.1's Case 1, we consider a pair of requests $r_5 = r_6 = (e, w, 0)$, with $e > b$, but $w < t$, then releasing another pair of requests $r_7 = r_8 = (f, x, 1)$, with $f \geq e$, and $w - \tau < x < w + \tau < t$. Note that e and w are only specified with a relation to b and t. This makes r_5, r_6 a pair of general requests at location 0 released and served after r_1, r_2. Also, note that f and x are only specified with a relation to e, w, b and t. This makes r_7, r_8 a pair of general requests crossing r_5, r_6. This case is very comparable to Case 1 of this line-skipping discussion. GRZA accepts one request from r_5 or r_6, creating an Exclusion Zone at location 0, spanning $(w - 2\tau, w + 2\tau)$. That Exclusion Zone is of size 4τ, allowing OPT to serve four requests by Corollary

7.8.1. However, instead of using the location-based Exclusion Zone at location 1 to serve two more requests, another pair of requests is released at location 1. GRZA accepts one of these requests, causing an Exclusion Zone between $(w - \tau, w + \tau)$ at location 1. This Exclusion Zone is of size 2τ, allowing OPT to serve two more requests, whilst still in compliance with Corollary 7.8.1. As in Case 1 of this line-skipping discussion, OPT serves a total of 6 requests, two of which require an empty movement. Meanwhile, GRZA serves two requests in this case, compared to the one request served in Case 1. Therefore, we find that the profit of OPT on this third case of line-skipping, combined with Lemma 7.16.1's Case 1 is $12p-2c$, whereas GRZA gains $4p-2c$. This makes the performance of OPT compared to GRZA $\frac{12p-2c}{4p-2c}$, which is lower than the $\frac{12p-2c}{3p-c}$ shown for Case 1 of line-skipping, combined with Lemma 7.16.1's Case 1. Finally, we note that the requests causing this Exclusion Zone set moves GRZA's servers, with one at location 0, and another at location 1, while OPT's servers are at location 1. This server formation is the same as the formation after Lemma 7.16.3's Case 1, making this case an overall inferior approach to Case 1.

Case 4 For the fourth and final possible Exclusion Zone set line-skipping Lemma 7.16.1's Case 1, we consider a pair of requests $r_5 = r_6 = (e, w, 1)$, with $e > b$, but $w < t$, then releasing another pair of requests $r_7 = r_8 = (f, x, 0)$, with $f \geq e$, and $w - \tau < x < w + \tau < t$. Note that e and w are only specified with a relation to b and t. This makes r_5, r_6 a pair of general requests at location 0 released and served after r_1, r_2. Also, note that f and x are only specified with a relation to e, w, b and t. This makes r_7, r_8 a pair of general requests crossing r_5, r_6. We first discuss the impact of the first pair of requests. Introducing r_5, r_6 first, we find that GRZA accepts one of them, gaining a profit of $p - c$. Accepting this request adds location-time interval $(1, (w - 2\tau, w + 2\tau))$ to $EZ(r_6+)$. The size of that location-time interval is 4τ. Since there is no Exclusion Zone at 0, by Corollary 7.8.1, OPT can only accept requests starting between $w - 2\tau$ and $w + 2\tau$ at location 1. Since the first two requests are at location 1, while OPT's servers are at location 0, all four of those require an empty movement.

After this, we consider the pair r_7, r_8. Introducing this pair of requests, we find that GRZA accepts one of them, gaining a profit of $p - c$, because serving this request requires an empty movement to serve the requests accepted in Lemma 7.16.1's Case 1. This shortens the Exclusion Zone at $(1, (w-2\tau, w+2\tau))$, and adds an extra zone to the set at $(0, (w - \tau, w + \tau))$. This zone is of size 2τ. Thus, by Corollary 7.8.1, OPT can only serve two requests from location 0. These requests do not require an empty movement, and, in fact, serve as two of the required empty movements to serve the requests accepted at location 1. This means OPT serves six requests for a profit of $6p - 2c$. Since OPT's servers end at location 0, no additional empty movements are required to serve the requests accepted in starting scenario case 1. This makes OPT's total profit $12p - 2c$. Meanwhile, GRZA's combined profit adds up to $4p - 2c$. This that OPT's performance compared to GRZA over this instance is $\frac{12p-2c}{4p-2c}$. That performance Ratio is lower than the $\frac{12p-2c}{3p-c}$ shown for case 1 of line-skipping, combined with Lemma 7.16.1's Case 1. Finally, we note that the requests causing this Exclusion Zone set moves GRZA's servers, with one at location 0, and another at location 1, while OPT's servers are at location 0. We discuss the implications of this new server formation in a separate Lemma. □

This was a distinction of all possible releases by an adversary line-skipping the scenario described in Lemma 7.16.1's Case 1. Throughout the Lemma, we have encountered two new scenarios. One is Lemma 7.16.3's Case 1 followed up by Lemma 7.16.1's Case 1. The other is Lemma 7.16.3's Case 4, followed up by Lemma 7.16.1's Case 1. In these cases, we need to explore what happens when a set of requests is released either between or before the two cases.

Observation 7.16.3.2 No set of requests released by an adversary line-skipping Lemma 7.16.3's Case 1 or 3 improves the performance ratio of OPT compared to GRZA.

Proof Recall that Lemma 7.16.3's Case 1 utilizes the location-based Exclusion Zone. If an adversary were to release any non distant request prior to Lemma 7.16.3's Case 1, that would remove the location-based Exclusion Zone, because it causes one of GRZA's servers to arrive at location 1. If an adversary were to release any distant request prior to Lemma 7.16.3's Case 1, that would follow the structure found in Lemma 7.16.1's Case 2, which had a performance ratio of 4, which is lower than Lemma 7.16.3's Case 1. This proves this Observation. □

Observation 7.16.3.3 No set of requests released by an adversary line-skipping Lemma 7.16.3's Case 4 improves the performance ratio of OPT compared to GRZA.

Proof Recall that Lemma 7.16.3's Case 4 requires an empty movement to be served. By observation 7.16.3.1, A request line-skipping a scenario in which an empty movement is required causes an Exclusion Zone in which the performance ratio of GRZA compared to OPT does not exceed 4. Lemma 7.16.3's Case 1 shows a performance ratio exceeding 4, meaning this performance ratio is worse than $\frac{12p-2c}{3p-c}$. □

Observation 7.16.3.4 No set of requests released by an adversary between Lemma 7.16.3's cases 1, 3 or 4 and 7.16.1's case 1 improves the performance ratio of OPT compared to GRZA.

Proof In the calculations used for finding Lemma 7.16.3's performance ratios, we already accounted for the empty movement(s) required in between the line-skipping and the starting scenario. This leaves us with servers in follow-up scenario locations, without a requirement where the servers end. Lemma 7.16.2 showed that the performance ratio of OPT compared to GRZA starting from the follow-up scenario does not exceed 4, which is worse than $\frac{12p-2c}{3p-c}$. □

We have now finished exploring all possible releases by an adversary from the line-skipping scenario, reached after Lemma 7.16.1's Case 1 was released (but served before that case starts). We have found closing arguments for all possible releases by an adversary, starting from the starting scenario, following-up on that scenario, or line-skipping that scenario. Having enumerated everything, we provide the Competitive Ratio of GRZA.

Theorem 7.16 Algorithm GRZA is $\frac{12p-2c}{3p-c}$ competitive for the 2S2L-V problem with arbitrary c, and $\upsilon - \lambda > 4\tau$.

Proof We proof this Theorem by enumerating all possible releases of an adversary that do not violate Lemma 7.13. We have done this in Lemmas 7.16.1, 7.16.2 and 7.16.3. We know that these Lemma's enumerate all cases by Corollaries 7.16.1.2, 7.16.1.4, 7.16.2.3, 7.16.3.2, 7.16.3.3 and 7.16.3.4. In those numerous cases, we found that the releases leading to Lemma 7.16.3's case 1 provided the highest performance ratio. Therefore, the competitive ratio of GRZA is $\frac{12p-2c}{3p-c}$. □

Having proven this upper bound of GRZA, we have found algorithm that come close to the lower bounds found in chapter 5 for each possible setting. This is a great achievement, especially since none of the Lower Bounds ever get higher than 5, making the correct combination of these algorithms very competitive on paired instances over 2S2L-V.

8 Conclusion and Future Work

In this thesis, we have expanded the results on the Online Car Sharing Problem with two servers and two locations, variable booking times, and with a focus on the special instance that all requests are released alongside an identical paired request. We have proven the Lower Bounds on this instance, and shown that no existing algorithm achieves an optimal competitive ratio for all settings of this problem. We have also shown that simply translating existing online car sharing algorithms to this setting would result in relatively poor results. Finally, we have provided three new algorithms that - when combined - are able to achieve good results on any paired 2S2L-V instance. We proved the bounds of these algorithms with a novel approach, limiting where an optimal scheduler could accept requests by the requests accepted by our algorithms.

There are various avenues for future research on the online car sharing problem. Because we focused on the 2S2L-V case with paired instances, there is much yet to be discovered about 2S2L-V in the general case. We conjecture that the Lower Bound proofs provided here work on the general case, and that the algorithms we provided achieve a tight performance even on the general case. To prove this, we see two possibilities. One could prove that paired instances provide the lowest possible Lower Bounds on the 2S2L-V Problem. This is far from trivial, but proving it would confirm our conjecture. Alternatively, one could prove Lemma 8.8 on the general case. Limiting the requests OPT can accept - without an algorithm making further profit - to specific location-times can be of great benefit to car sharing. Furthermore, one could prove the Upper Bound of the Greedy Algorithm or investigate Arrogant Greedy with a 2τ end of the line Exclusion Zone. We have a hypothesis that these algorithms are the final pieces to achieve truly tight upper bounds.

2S2L-V is not the only online car sharing variant that warrants further investigation. Increasing the scope, we see many different avenues of research for this young field. In our proposal for this thesis, we provided many examples, settings and variations of online car sharing problems that could still be researched. We provide some of these ideas here, to serve as inspiration for continued work on the subject.

Server Cleaning

The server cleaning variation is an extension of the car-sharing problem, in which a server goes into a certain cooldown after serving one or multiple requests. There are many practical reasons for not being able to serve requests back to back. Such reasons include cleaning the server or refueling the server. The amount of cooldown can either be a constant, or solely reliant on the length of the trip preceding it.

Late Rejection

In some situations, a company could stand to benefit from cancelling a request it already accepted. In this variation, one would allow already accepted requests to be rejected before their starting time, still, for a certain cost. Realistically, a company cancelling on a request it already accepted is extremely poor form, and would lead to major losses due to bad reviews and lower overall usage. Modelling such long term effects could also be very interesting.

Delayed Booking

Another way to increase the complexity of this problem is if we allow *Delayed Booking.* With delayed booking, requests do not need to be handled as soon as they arrive. Instead, the decision to serve a request or not should be made before a certain deadline. Note that the general case is a subcase of the Delayed Booking case, with a deadline of $b + 0$ per request.

Different Networks

Without requiring additional problems, there are still many special networks for which the bounds found by K. Luo et al in [11] might be improved upon. There are multiple networks that are of particular interest. One such a network would be a two location network with a storage in the middle, that would not receive requests, but would allow unprompted moves to and fro. Another interesting network would be a tree. Other interesting networks could be a double star network - that is, two stars connected by a single edge - or a line network.

Multi-Capacitated Car Pooling
One of the assumptions that has implicitly been made in the car-sharing literature is that every request is served for one client. Most cars - or servers - have more than one seat, however. That means the full capacity of servers is seldomly used. To alleviate this shortcoming, some clients may be willing to share their ride with a different client if the trip is the same for both of them. This would probably cut in on the price of a trip, and requires some sort of delayed booking.

Prize Collection
The concept of prize collection was used by E. Balas in an analysis of the travelling salesperson problem, allowing the server to skip serving requests that are too far off course for a certain cost [3]. Applying this theory to car-sharing could be interesting, for it would penalize rejecting requests. This could change the competitive ratios of some problems or algorithms such as those developed for 2S2L-F. One of the bigger problems with a direct application of this line of thinking would be that rejecting requests is an integral part of the car-sharing problem and removing them could lead to very harsh competitive ratios. A way to reduce the harshness might be to only apply the rejection penalty if the rejected request was acceptable to any server.

Car-Sharing with Advice
For some online problems, authors decide to give online algorithms a helping hand in trying to find a good competitive ratio. They do this by giving the online algorithm access to an *oracle*: a separate entity that is familiar with the offline variant of the problem instance given to the online algorithm [6]. The algorithm can consult the oracle on different matters, such as when the next request is going to arrive, whether the optimum algorithm served this current request or the amount of requests. To our knowledge, no literature for car-sharing with advice has been released.

Probabilistic Car-Sharing
Using real life traffic information, one could model a less adversarial input. Providing an online algorithm that is likely to perform well on real life data could be a fascinating line of research all on its own.

Uncertainty Caused by Randomized Travel Times One of the assumptions made in the car-sharing problem is that the same distance takes the same amount of time to travel each and every time. This is an oversimplification of reality, because traffic lights, weather conditions or accidents could all drastically impact the travel time of each and every trip. Replacing the static travel time of requests with a randomized one might lend itself to a realistic simulation. This would impact the analysis of algorithms in an unexpected way and would probably lead to questions of robustness and dealing with the uncertainty, rather than competitiveness.

References

[1] Norbert Ascheuer, Sven Oliver Krumke, and Jörg Rambau. “Online Dial-a-Ride Problems: Minimizing the Completion Time”. In: *STACS 2000, 17th Annual Symposium on Theoretical Aspects of Computer Science, Lille, France, February 2000, Proceedings*. Ed. by Horst Reichel and Sophie Tison. Vol. 1770. Lecture Notes in Computer Science. Springer, 2000, pp. 639–650. DOI: 10.1007/3-540-46541-3_53. URL: https://doi.org/10.1007/3-540-46541-3%5C_53.

[2] Giorgio Ausiello et al. “Algorithms for the On-Line Travelling Salesman”. In: *Algorithmica* 29.4 (2001), pp. 560–581. DOI: 10.1007/s004530010071. URL: https://doi.org/10.1007/s004530010071.

[3] Egon Balas. “The prize collecting traveling salesman problem”. In: *Networks* 19.6 (1989), pp. 621–636. DOI: 10.1002/net.3230190602. URL: https://doi.org/10.1002/net.3230190602.

[4] Antje Bjelde et al. “Tight Bounds for Online TSP on the Line”. In: *ACM Trans. Algorithms* 17.1 (2021), 3:1–3:58. DOI: 10.1145/3422362. URL: https://doi.org/10.1145/3422362.

[5] Allan Borodin and Ran El-Yaniv. *Online computation and competitive analysis*. Cambridge University Press, 1998. ISBN: 978-0-521-56392-5.

[6] Joan Boyar et al. “Online Algorithms with Advice: A Survey”. In: *SIGACT News* 47.3 (2016), pp. 93–129. DOI: 10.1145/2993749.2993766. URL: https://doi.org/10.1145/2993749.2993766.

[7] Vinay A. Jawgal, V. N. Muralidhara, and P. S. Srinivasan. “Online Travelling Salesman Problem on a Circle”. In: *Theory and Applications of Models of Computation - 15th Annual Conference, TAMC 2019, Kitakyushu, Japan, April 13-16, 2019, Proceedings*. Ed. by T. V. Gopal and Junzo Watada. Vol. 11436. Lecture Notes in Computer Science. Springer, 2019, pp. 325–336. DOI: 10.1007/978-3-030-14812-6_20. URL: https://doi.org/10.1007/978-3-030-14812-6%5C_20.

[8] Elias Koutsoupias. “The k-server problem”. In: *Comput. Sci. Rev.* 3.2 (2009), pp. 105–118. DOI: 10.1016/j.cosrev.2009.04.002. URL: https://doi.org/10.1016/j.cosrev.2009.04.002.

[9] Kuan-Yun Lai et al. “Randomized Scheduling for the Online Car-sharing Problem”. In: *CoRR* abs/2103.07367 (2021). arXiv: 2103.07367. URL: https://arxiv.org/abs/2103.07367.

[10] Songhua Li, Leqian Zheng, and Victor C. S. Lee. “Car-Sharing: Online Scheduling k Cars Between Two Locations”. In: *Frontiers in Algorithmics - 14th International Workshop, FAW 2020, Haikou, China, October 19-21, 2020, Proceedings*. Ed. by Minming Li. Vol. 12340. Lecture Notes in Computer Science. Springer, 2020, pp. 49–61. DOI: 10.1007/978-3-030-59901-0_5. URL: https://doi.org/10.1007/978-3-030-59901-0%5C_5.

[11] Haodong Liu et al. “Car-Sharing Problem: Online Scheduling with Flexible Advance Bookings”. In: *Combinatorial Optimization and Applications - 13th International Conference, COCOA 2019, Xiamen, China, December 13-15, 2019, Proceedings*. Ed. by Yingshu Li, Mihaela Cardei, and Yan Huang. Vol. 11949. Lecture Notes in Computer Science. Springer, 2019, pp. 340–351. DOI: 10.1007/978-3-030-36412-0_27. URL: https://doi.org/10.1007/978-3-030-36412-0%5C_27.

[12] Kelin Luo, Thomas Erlebach, and Yinfeng Xu. “Car-Sharing Between Two Locations: Online Scheduling with Flexible Advance Bookings”. In: *Computing and Combinatorics - 24th International Conference, COCOON 2018, Qing Dao, China, July 2-4, 2018, Proceedings*. Ed. by Lusheng Wang and Daming Zhu. Vol. 10976. Lecture Notes in Computer Science. Springer, 2018, pp. 242–254. DOI: 10.1007/978-3-319-94776-1_21. URL: https://doi.org/10.1007/978-3-319-94776-1%5C_21.

[13] Kelin Luo, Thomas Erlebach, and Yinfeng Xu. “Car-Sharing between Two Locations: Online Scheduling with Two Servers”. In: *43rd International Symposium on Mathematical Foundations of Computer Science, MFCS 2018, August 27-31, 2018, Liverpool, UK*. Ed. by Igor Potapov, Paul G. Spirakis, and James Worrell. Vol. 117. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018, 50:1–50:14. DOI: 10.4230/LIPIcs.MFCS.2018.50. URL: https://doi.org/10.4230/LIPIcs.MFCS.2018.50.

[14] Kelin Luo, Thomas Erlebach, and Yinfeng Xu. “Car-Sharing on a Star Network: On-Line Scheduling with k Servers”. In: *36th International Symposium on Theoretical Aspects of Computer Science, STACS 2019, March 13-16, 2019, Berlin, Germany*. Ed. by Rolf Niedermeier and Christophe Paul. Vol. 126. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019, 51:1–51:14. DOI: 10.4230/LIPIcs.STACS.2019.51. URL: https://doi.org/10.4230/LIPIcs.STACS.2019.51.

[15] Kelin Luo, Thomas Erlebach, and Yinfeng Xu. "Online Scheduling of Car-Sharing Requests Between Two Locations with Many Cars and Flexible Advance Bookings". In: *29th International Symposium on Algorithms and Computation, ISAAC 2018, December 16-19, 2018, Jiaoxi, Yilan, Taiwan.* Ed. by Wen-Lian Hsu, Der-Tsai Lee, and Chung-Shou Liao. Vol. 123. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018, 64:1–64:13. DOI: `10.4230/LIPIcs.ISAAC.2018.64`. URL: `https://doi.org/10.4230/LIPIcs.ISAAC.2018.64`.

[16] Kelin Luo, Yinfeng Xu, and Haodong Liu. "Online scheduling of car-sharing request pairs between two locations". In: *Journal of Combinatorial Optimization* (2020), pp. 1–24. DOI: `10.1007/s10878-020-00635-8`. URL: `https://doi.org/10.1007/s10878-020-00635-8`.