Online 3-Taxi on General Metrics

Christian Coester*† Tze-Yang Poon*

Abstract. The online k-taxi problem, introduced in 1990 by Fiat, Rabani and Ravid, is a generalization of the k-server problem where k taxis must serve a sequence of requests in a metric space. Each request is a pair of two points, representing the pick-up and drop-off location of a passenger. In the interesting "hard" version of the problem, the cost is the total distance that the taxis travel without a passenger. The problem is known to be substantially harder than the k-server problem, and prior to this work even for $k = 3$ taxis it has been unknown whether a finite competitive ratio is achievable on general metric spaces. We present an $O(1)$-competitive algorithm for the 3-taxi problem.

1 Introduction. The k-taxi problem, originally proposed by Karloff and formalized by Fiat, Rabani and Ravid in 1990 [18], is a fundamental online problem that generalizes the k-server problem by associating each request with a source and destination. In this problem, k taxis move in a metric space and must serve a sequence of requests arriving online. Each request specifies a pick-up location and a drop-off location. To serve it, a taxi must move first to the pick-up point and then to the drop-off point. The taxi serving each request must be chosen by an online algorithm without knowledge of future requests.

There are two versions of the problem, known as the "easy" and "hard" k-taxi problem, which differ in how the cost is defined: In the easy k-taxi problem, the cost is the total distance traveled by all taxis. In the hard k-taxi problem, the cost is defined as only the total *overhead* distance of empty runs; that is, distances traveled to get to pick-up locations count towards the cost, but distances traveled from pick-up to drop-off (which are the same regardless of algorithm) are excluded from the cost. Although the optimal (offline) solutions are the same for both models, the smaller costs in the hard version mean that the competitive ratio between the cost of an online algorithm and an optimal (offline) solution is higher. Indeed, Coester and Koutsoupias [15] showed that the easy k-taxi problem is exactly equivalent to the k-server problem, with a deterministic competitive ratio between k and $2k - 1$, whereas the hard version is at least exponentially harder, with a lower bound of $\Omega(2^k)$ for deterministic algorithms. Therefore, research has focused on the hard version, and all mentions of the k-taxi problem hereafter refer to the hard version unless stated otherwise.

In terms of algorithms, prior to this work, a finite competitive ratio has been known to be achievable for general metric spaces only for the case of $k = 2$ taxis, where the deterministic competitive ratio is exactly 9 [15]. Additional results exist for special metric spaces: an $O(1)$-competitive algorithm for three taxis on a line metric [15], $O(2^k)$-competitive algorithms for ultrametrics [15, 11], and an $O(k^D)$-competitive algorithm for weighted trees of combinatorial depth D [11]. Using randomization, for n-point metric spaces with aspect ratio Δ there exist multiple different algorithms with the following competitive ratios: $O(2^k \log n)$ based on the aforementioned result for ultrametrics [15], $O((n \log k)^2 \log n)$ based on an algorithm for a more general problem of "metrical service systems with transformations" [7], $2^{O(\sqrt{\log k \log \Delta})} \log n$ based on a reverse-time primal-dual analysis of the Double Coverage algorithm on ultrametrics [11], and most recently $O\left(\log^3 \Delta \cdot \log^2(nk\Delta)\right)$ based on a new linear programming relaxation for the problem [20]. Note, however, that the latter collection of bounds is vacuous on general metric spaces where the number of points n and Δ could be infinite.

Despite the interest that the problem has generated, for general metric spaces (and even seemingly simple cases such as three taxis on the two-dimensional Euclidean plane) it has remained unknown since the problem's introduction 35 years ago whether any finite competitive ratio is achievable when $k > 2$. In this paper, we provide a positive answer for $k = 3$.

*Department of Computer Science, University of Oxford (christian.coester@cs.ox.ac.uk, poontzeyang@gmail.com)

†Funded by the European Union (ERC, CCOO, 101165139). Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Research Council. Neither the European Union nor the granting authority can be held responsible for them.

THEOREM 1.1. *There exists an $O(1)$-competitive deterministic online algorithm for the 3-taxi problem on general metric spaces.*

Additional Related Work. A competitive algorithm for the hard 2-taxi problem due to Karloff has been known since its introduction (see [18]), and [18] also gave a first algorithm for the *easy* k-taxi problem by adapting their algorithm for the k-server problem. A stochastic version of the (easy) k-taxi problem was studied in [16].

The k-server problem corresponds to the special case of the k-taxi problem where for each request, the pick-up point is equal to the drop-off point. Its competitive ratio is $\Theta(k)$ deterministically [22, 21]. For randomized algorithms, there are polylog(k, n)- and polylog(k, Δ)-competitive algorithms [2, 10] and a lower bound of $\Omega(\log^2 k)$ [9], which is also the best lower bound for randomized algorithms for the k-taxi problem.

Besides generalizing the k-server problem, [15] showed that the (deterministic) k-taxi problem is also a generalization of the width-k layered graph traversal problem, which is also equivalent to chasing sets of cardinality k in a metric space [17, 8]. The aforementioned $\Omega(2^k)$ lower bound on the k-taxi problem is inherited from the same lower bound on these problems.

2 Preliminaries. Let (S, d) be a metric space. To simplify notation, for two points $x, y \in S$, we will often write $xy := d(x, y)$ for their distance. A *configuration* is a multiset of k points in S, representing the locations of k taxis. For two configurations C and C', we denote by $d(C, C')$ the cost of a minimum weight perfect matching between them. This captures the total distance traveled to move taxis from C to C'.

An instance of the k-taxi problem on a metric space (S, d) consists of an initial configuration C_0 and a sequence $(r_1, s_1), (r_2, s_2), \ldots, (r_T, s_T)$ of requests, each of which is a pair of two points in S. An algorithm is said to serve the request sequence if it outputs a sequence of configurations $\hat{C}_1, \hat{C}_2, \ldots, \hat{C}_T$ such that for all $t \in \{1, \ldots, T\}$, $r_t \in \hat{C}_t$. After the algorithm reaches configuration $\hat{C}_t$, the taxi at r_t serves the request (r_t, s_t) by relocating to s_t. This changes the configuration to $C_t = \hat{C}_t - r_t + s_t$, where the $+$ and $-$ operators add/remove one copy of a point from a configuration. The cost of the algorithm is defined as $\sum_{t=1}^{T} d(C_{t-1}, \hat{C}_t)$.

An online algorithm must choose each configuration $\hat{C}_t$ after the request (r_t, s_t) is revealed and without knowledge of future requests. In contrast, an offline algorithm knows the request sequence in advance and can therefore serve it optimally. We denote by COST and OPT the costs of an online algorithm and the optimal offline algorithm, respectively. The online algorithm is ρ-competitive if $\text{COST} \leq \rho \cdot \text{OPT} + c$ for all request sequences, where c is a constant that may depend only on the metric space and the initial configuration, but not the request sequence.

Bridges and Tripods. We may assume without loss of generality that for any two points $x, y \in S$, the metric space contains a continuous *bridge* $B(x, y)$ between x and y. That is, $B(x, y) \subseteq S$ is isometric to a closed interval of length xy whose endpoints are x and y. This can be achieved by adding virtual points to the metric space. We describe our algorithm in a way that it may move taxis to these virtual points. To turn this into an algorithm on the original metric space (without virtual points), we can defer the movement of any taxi until it actually serves a request, which always happens at non-virtual points. By the triangle inequality, deferring movement cannot increase the cost of the algorithm.

Similarly, we may assume without loss of generality that for any three points $x, y, z \in S$, the metric space contains a continuous *tripod* $B(x, y, z)$. That is, there is a point $e \in S$ (depending on x, y, z) such that $xy = xe + ye$, $xz = xe + ez$ and $yz = ye + ez$, and the tripod $B(x, y, z)$ is the union of the three bridges $B(x, e)$, $B(y, e)$ and $B(z, e)$. See Figure 2.1. The point e is called the *center* or *branching point* of the tripod $B(x, y, z)$ and can be added to the metric space by connecting it to x, y and z by edges of lengths

$$xe = \frac{xy + xz - yz}{2}, \quad ye = \frac{xy - xz + yz}{2}, \quad ze = \frac{-xy + xz + yz}{2}. \tag{2.1}$$

Note that the union of any two edges of $B(x, y, z)$ yields a bridge between two of its endpoints. In general, there may be multiple bridges/tripods for a given pair (x, y) or triple (x, y, z), and we use $B(x, y)$ and $B(x, y, z)$ to refer to any one of them chosen arbitrarily (unless further specified).

LEMMA 2.1. *Given points u, v, w_1, w_2, let h_1 and h_2 be the branching points of $B(u, v, w_1)$ and $B(u, v, w_2)$, respectively. Then any two corresponding edges (sharing the same endpoint u or v, and likewise the edges w_1h_1 and w_2h_2) differ in length by at most w_1w_2.*

Proof. This follows from equations (2.1) by the triangle inequality. ∎

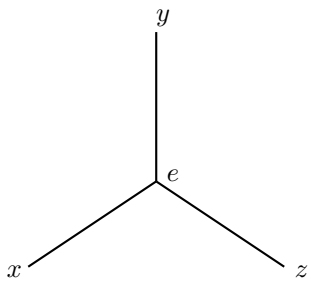

Figure 2.1: The tripod $B(x, y, z)$ with center e.

Active, passive and unobstructed taxis. At any time, we call the taxi that served the previous request the *active* taxi and the other taxis *passive*. If there was no previous request, an arbitrary taxi is considered active. In our 3-taxi algorithm, we denote the active online taxi by x_1 and the two passive taxis x_2 and x_3, reindexing the taxis between requests appropriately. We use x_i to refer to both the taxi as well as its location.

When the current request is (r, s), we call a passive taxi *unobstructed* if the shortest path between it and r on the tripod $B(x_2, x_3, r)$ does not contain the location of the other passive taxi. Equivalently, a passive taxi is unobstructed unless the other passive taxi is at the branching point of $B(x_2, x_3, r)$. In the degenerate case where both passive taxis are at the same location, we regard one of them as unobstructed, chosen arbitrarily.

3 An Algorithm for 3-Taxi on General Metrics. We will define a deterministic online algorithm TripodTracker for the 3-taxi problem on general metric spaces. Although it suffices to move a single taxi in response to a request, it is more convenient for the description and analysis of TripodTracker to simultaneously move several taxis continuously when a new request arrives, similarly to the DoubleCoverage algorithm for k-server [13, 14]. Our algorithm builds on ideas from the algorithm of [15] for three taxis on the line metric, but generalizing it to arbitrary metrics requires overcoming new structural challenges. We adapt both the algorithm and the potential function used for its analysis in ways that are crucial to handle general metrics. As a result, even when specialized to the line metric, our algorithm and potential function differ from [15]. Notably, our approach also streamlines certain aspects, avoiding the need for a separate treatment of seven different movement cases as in [15].

3.1 Overview. When a request (r, s) arrives, TripodTracker proceeds by moving the taxis simultaneously at different speeds towards r, until a taxi reaches r. The active taxi moves along the bridge $B(x_1, r)$, while the passive taxis move along the tripod $B(x_2, x_3, r)$. The active taxi plays a special role because at the start of the request we can always guarantee that there is an offline taxi at the same location. Thus, it is preferable to keep a taxi nearby, and accordingly TripodTracker moves the active taxi at a much slower speed than the passive taxis.

The movement speeds of the two passive taxis depend on the structure of $B(x_2, x_3, r)$. TripodTracker uses $B(x_2, x_3, r)$ to decide which of the two passive taxis is better suited to serve the request. Consider a scenario where x_2 is located at the center of the tripod $B(x_2, x_3, r)$ and thus on a shortest path from x_3 to r. Then there would be no reason to serve the request using x_3, since the online algorithm could instead serve the current request with x_2 and defer the movement of x_3 to the original location of x_2 to a later request. As such, if one of the passive taxis is at the branching point of $B(x_2, x_3, r)$, we move only that passive taxi and the active taxi towards the request point r. This is similar to the algorithm for 2-taxi on general metrics. Otherwise both passive taxis are unobstructed. In this case, it is less clear which passive taxi is better suited to serve the request, so we simultaneously move both passive taxis towards r and towards each other by moving them towards the branching point.

Inspired by the algorithm for the 3-taxi problem on the line [15], we keep track of two disjoint "intervals", each starting at a passive taxi and ending at some point along the bridge $B(x_2, x_3)$ that is part of the tripod $B(x_2, x_3, r)$. (Recall that there may be several bridges $B(x_2, x_3)$, so when a new request appears, we map these intervals to a possibly different bridge $B(x_2, x_3)$ that is formed by two edges of the tripod with the new request.) Intuitively, each interval mark a region where the passive taxi located at its endpoint holds "more responsibility" than the other passive taxi. Accordingly, the intervals are defined only along bridges between the passive taxis:

outside these bridges, the passive taxis never move simultaneously and therefore do not need to be distinguished by responsibility. The active taxi does not have an interval and is treated specially because, unlike in k-server, the active taxi can be relocated to a new location at no cost, making it volatile and "unfit for holding responsibility" beyond its current location.

These intervals – together with the role of the active taxi – encode the algorithm's entire memory about the past. This information is used by the algorithm to determine the passive taxi movement speeds, as well as by the potential functions to relate online movement costs to offline movement costs. How these intervals are updated, how they influence movement speeds, and how they affect the potential function all differ from the previous algorithm for the line metric [15].

3.2 Intuition. In order to make sense of the details of TripodTracker's behavior, it will be helpful to briefly describe part of the potential function we will use in the analysis. This potential will be the minimum weight of a certain *distorted* matching between the online and offline configurations. The active online taxi x_1 will always be matched to the active offline taxi, and this pair will contribute its undistorted distance to the potential. The pairs involving the passive online taxis x_2 and x_3 may contribute less than their actual distance and their matched partners will be selected to minimize the total weight of the matching.

In the scenario where the active offline taxi moves to serve the current request, we will charge our movement costs directly to the offline movement cost. This is possible because the total distance moved by the active online taxi is no greater than that of the offline active taxi, and all online movement is at most a constant multiple of the active online taxi's moved distance.

Otherwise, we gain the additional assumption that the active offline taxi did not move. In this case, we instead charge our movement costs to a net decrease in the aforementioned potential. To achieve this, we will first assume that the offline algorithm moves a taxi to r before the online algorithm moves, such that during TripodTracker's movement at least one of the passive online taxis is getting closer to its matched partner at r and thus decreasing its matching potential contribution. By moving the active taxi at a sufficiently slow speed, we ensure that its increase in matching contribution is smaller than the above decrease. However, in the movement cases where there are two passive taxis moving, we will need to ensure that the passive taxi moving towards its matched partner at r can decrease the potential more than the other passive taxi increases it by possibly moving away from its matched partner. We achieve this by designing the potential to assign a discount factor to specific portions of the path from each passive taxi to its matched partner, and accordingly these portions will contribute less to the potential. The assignments of this discount is implicitly tracked by the aforementioned intervals and the structure of several tripods, and is described in Section 3.4. See Figure 3.1 for an example, where $1 - \psi < 1$ is the discount factor.

TripodTracker makes use of the differing discounts in two possible ways. If it is able to decrease the amount of undiscounted distance between a passive taxi and r, this means the distorted distance of that pair decreases relatively quickly, which we leverage in the analysis. Alternatively, if the entire distance between a passive taxi and r is already discounted, then moving this taxi towards r decreases the distorted distance more slowly. In this case, TripodTracker moves that passive taxi towards r at a slightly faster speed (i.e., $1 + b$ instead of 1), so that the distorted distance to r decreases at a comparable rate.

A major challenge is that TripodTracker does not actually know which of the three online taxis is matched to the offline taxi at r, as this depends on the unknown locations of the other offline taxis. This is the main reason for the significant care required in the choice of movement speeds and discount factors. As we will see later, there are also cases where the movement cost cannot be charged to the matching potential, and we will also employ an additional potential function for this purpose.

The first online taxi which reaches the request location r will be used to serve the request. If one of the passive taxis is the first to reach r, it will additionally become the new active taxi. However, this results in a reassignment of matching partners, which could cause an increase in the overall matching potential due to an increase in the amount of undiscounted distance included in the matching. TripodTracker handles this by reorganizing the intervals, having the newly passive taxi *inherit* some of the interval that previously belonged to the newly active taxi that just served the current request and giving the passive taxi that remains passive an additonal amount of interval to reflect its "additional responsibility" relative to the newly passive taxi. Through these reorganizations, TripodTracker ensures that any overall change in potential can be charged to the offline movement in the current request.

3.3 Algorithm Description. A pseudocode of our algorithm TripodTracker is provided in Algorithm 3.1, and an example of it serving a request is depicted in Appendix A. We keep track of an interval for each passive taxi x_i, whose first endpoint is the passive taxi location x_i and the other endpoint is denoted by q_i. We use $\ell_i := x_i q_i$ for the length of this interval.[1] Before the first request arrives, we initialize both interval lengths ℓ_i to 0 in line 1 of the algorithm.

When a request (r, s) arrives, the main task is to move a taxi to the pick-up location r. The interval endpoint q_i is chosen as the point at distance ℓ_i from x_i on the path between the two passive taxis that is part of the tripod $B(x_2, x_3, r)$. We will maintain throughout the run of the algorithm that the two intervals of the passive taxis are interior-disjoint, i.e., $\ell_2 + \ell_3 \leq x_2 x_3$.

Next (lines 6-14), we continuously and simultaneously move the active taxi and each unobstructed passive taxi towards r, until one of them reaches r. The active taxi moves at some small speed $a \in (0, 1)$, to be determined later, along a bridge from its old location to r. Each unobstructed passive taxi x_i moves towards r along the tripod $B(x_2, x_3, r)$. We denote by e the branching point of this tripod. Note that e remains unchanged while both passive taxis are unobstructed, and if there is a single unobstructed passive taxi, then e is at the same location as this taxi and they move together towards r. The movement speed of each unobstructed passive taxi x_i depends on whether the branching point e belongs to this taxi's interval or not: if it is inside the interval, but not at x_i, then the movement is at a fast speed of $1 + b$, for some constant $b > 0$ to be determined later. Otherwise the movement is at speed 1. Note that the case where $x_i e = 0$ is precisely the case where x_i is the only unobstructed taxi. At the same time, we also move the associated interval endpoint q_i towards the branching point e at speed 1 if it is not at e already, and update ℓ_i to maintain that it is the length of the interval between x_i and q_i. In the case where e itself is moving (at speed 1 towards r, because it is at the location of the single unobstructed passive taxi), this means that the distance between q_i and e remains unchanged, since q_i and e then both move towards r on the path from q_i through $x_i = e$ to r. If a passive taxi reaches point e during the movement, the other taxi is no longer unobstructed and stops moving.

Algorithm 3.1 TripodTracker

Require: Initial taxi location x_1, x_2, x_3

1: $(\ell_2, \ell_3) \leftarrow (0, 0)$
2: **for each** request (r, s) **do**
3: **for** $i \in \{2, 3\}$ **do**
4: $q_i \leftarrow$ point on $B(x_2, x_3) \subseteq B(x_2, x_3, r)$ with $\ell_i = x_i q_i$.
5: **end for**
6: **while** $r \notin \{x_1, x_2, x_3\}$ **do**
7: move x_1 along $B(x_1, r)$ towards r at speed a
8: $e \leftarrow$ branching point of $B(x_2, x_3, r)$
9: **for each** unobstructed passive taxi x_i **do**
10: move x_i along $B(x_2, x_3, r)$ towards r at speed $\begin{cases} 1+b & \text{if } \ell_i \geq x_i e > 0, \\ 1 & \text{otherwise} \end{cases}$
11: move q_i towards e at speed 1 (if it is not already at e)
12: $\ell_i \leftarrow x_i q_i$
13: **end for**
14: **end while**
15: **if** passive taxi at r **then** ▷ WLOG $x_3 = r$
16: $f \leftarrow$ the branching point of $B(x_1, x_2, x_3)$
17: $\ell_2 \leftarrow \min(\ell_2 + x_1 f, x_1 x_2)$
18: $\ell_3 \leftarrow \max(0, \ell_3 - x_1 x_3)$
19: reindex $(x_1, x_3) \leftarrow (x_3, x_1)$
20: **end if**
21: serve the request with x_1
22: **end for**

[1] In our analysis, each passive taxi will enjoy a discount in any portion of a path that belongs to its own interval.

Once a taxi reaches the pick-up point r, all taxis stop moving. If the taxi reaching r is passive, some reorganization is necessary as the roles of active and passive taxis change, and we need to update the interval lengths to ensure that the intervals around the new passive taxis can again be placed in an interior-disjoint way on a bridge between the new passive taxis, and that the potential used in the analysis does not increase. The details of this reorganization are described in lines 15-20 for the case that x_3 is the passive taxi reaching the request. The other case is symmetric. Intuitively, the interval length ℓ_2 of the passive taxi that remains passive increases, as it is in some sense "more passive" than the newly passive taxi, and is then truncated to at most the new distance x_1x_2 between the two passive taxis. The active taxi that becomes passive inherits the interval length ℓ_3 of the passive taxi that becomes active, but reduced by the distance between these two taxis, and truncated at 0.

Finally, the new active taxi serves the request by moving from r to s.

The following claim establishes the aforementioned invariant that the two intervals around the passive taxis remain interior-disjoint.

Claim 3.1. *At all times during the continuous movement, the interval endpoints appear on the path from x_2 to x_3 in the order $x_2 \to q_2 \to q_3 \to x_3$, possibly with equality between consecutive points in the sequence.*

Proof. By definition, we maintain $\ell_i = x_iq_i$ at all times, so the invariant is true following the initialization in line 1. During the continuous movement, if a single passive taxi x_i is unobstructed, then $x_ie = 0$ and x_i and q_i travel away from the endpoints of the other interval at the same speed, maintaining the invariant. If both passive taxis are unobstructed, then all four interval endpoints move towards e (except some q_i which may be at e already). If all movements are at speed 1, then this clearly maintains the invariant. The case where a passive taxi moves at speed $1 + b$ also doesn't violate the invariant, as this only occurs if there is a positive gap $\ell_i > 0$ between x_i and q_i.

It remains to show that after line 18 we have $\ell_2^{\text{new}} + \ell_3^{\text{new}} \le x_1x_2$, where ℓ_i^{new} denotes the value after the update. Since x_1x_2 will be the new distance between the passive taxis, this will ensure the invariant holds for the next request. Clearly the statement is true if $\ell_3^{\text{new}} = 0$. Otherwise, we get

$$\ell_2^{\text{new}} + \ell_3^{\text{new}} \le \ell_2 + x_1f + \ell_3 - x_1x_3 = \ell_2 + \ell_3 - x_3f \le x_2x_3 - x_3f = x_2f \le x_1x_2,$$

where we used that $\ell_2 + \ell_3 \le x_2x_3$ was true since the invariant held previously. ∎

3.4 Analysis. We refer to the three offline taxis as y_1, y_2, y_3. Let COST and OPT be the costs incurred by TripodTracker and the optimal offline algorithm, respectively, in serving a given request sequence. Let COST_t and OPT_t be the costs incurred in serving the t^{th} request by the online and offline algorithms, respectively. Let Φ_t be the value of a non-negative potential function after both algorithms have served the t^{th} request and let $\Delta\Phi_t = \Phi_t - \Phi_{t-1}$. To show that TripodTracker is κ-competitive for some constant κ, it suffices to show that for all requests

$$\text{COST}_t + \Delta\Phi_t \le \kappa\text{OPT}_t, \tag{3.1}$$

if we additionally require that $\Phi_0 = 0$.

For each request, we assume that the offline algorithm moves a taxi to r first, followed by the online algorithm. This allows us to guarantee that during the online algorithm's movement, there is an offline taxi located at r. We split each request into four phases: offline taxi movement, online taxi movement (lines 6-14), reorganization (lines 15-20) and relocation (line 21). Let the change in potential during these phases for the t^{th} request be $\Delta\Phi_t^{\text{off}}, \Delta\Phi_t^{\text{on}}$, $\Delta\Phi_t^{\text{org}}$ and $\Delta\Phi_t^{\text{rel}}$, respectively. Then online cost COST_t is only incurred during the online movement phase and offline cost OPT_t is only incurred during the offline movement phase. We consider the former two phases in Section 3.4.1 and the latter two in Section 3.4.2.

We use a potential of the form $\Phi = \alpha M + \beta\Sigma$, where $\alpha > \beta > 1$ are constants and M and Σ are two components of the potential.

To define these components, we first generalize the definition of intervals around the passive taxis: Recall that in the algorithm, we used q_i as a point on the tripod $B(x_2, x_3, r)$, and by Claim 3.1 they always reside on the path between x_2 and x_3. Since this path embeds isometrically into any other tripod $B(x_2, x_3, z)$, we can define points q_i on the (x_2, x_3)-path in any such tripod as well, at the same relative distance from x_2 and x_3 (i.e., such that $\ell_i = x_iq_i$). For convenience, we will denote these points by q_i again.

The component M is the minimum weight of the aforementioned distorted matching between the online and offline taxi locations. We restrict the active online and offline taxis (denoted x_1 and y_1) to always be matched to

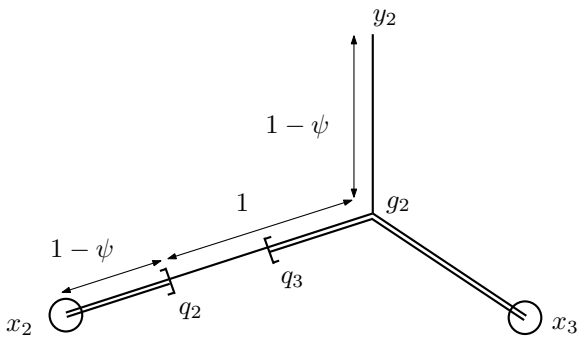

Figure 3.1: The matching contribution of the (x_2, y_2)-pair is determined by the undiscounted factor of 1 and the discounted factor of $1-\psi$ over the different portions of the path P_2 from x_2 to y_2.

each other, and require that this pair contributes its actual (undistorted) distance to M. The pairs involving the passive online taxis x_2 and x_3 may contribute less than their actual distance. The actual contribution of a pair (x_i, y_i), where x_i is a passive online taxi and y_i the offline taxi it is matched to, is determined by the structure of the tripod $B(x_2, x_3, y_i)$, including the location of q_2, q_3 on this tripod. Denote by g_i the branching point of $B(x_2, x_3, y_i)$. Let the unique (x_i, y_i)-path on $B(x_2, x_3, y_i)$ be P_i. The two passive offline taxis y_2, y_3 are indexed to minimize the expression

$$M = x_1 y_1 + \int_{P_2} f_2(z)\, dz + \int_{P_3} f_3(z)\, dz,$$

where

$$f_i(z) = \begin{cases} 1-\psi & \text{if } x_i z \le \ell_i \text{ or } x_i z \ge x_i g_i, \\ 1 & \text{otherwise,} \end{cases}$$

for some constant $0 < \psi < 1$. In other words, portions of P_i in x_i's own interval and on the y_i edge of $B(x_2, x_3, y_i)$ are discounted to the smaller factor of $1-\psi$ and the remaining portion is not discounted and contributes its entire distance to the matching. See Figure 3.1.

The component Σ of the potential is defined as

$$\Sigma = \max(0, \min(\ell_2, \ell_3) - q_2 q_3).$$

The Σ component plays a less prominent role than M in the proof, and we need it to pay for the algorithm's movement only in the case where both interval endpoints q_2 and q_3 are located at the branching point e of $B(x_2, x_3, r)$, as the M potential might not decrease in this case. The coefficient β of Σ in the overall potential will be much smaller than the coefficient α of M, so that any adverse change to Σ in other cases will be negligible compared to the change in M. The Σ potential plays a similar role to the sum of pairwise server distances potential used in the analysis of the DoubleCoverage algorithm for k servers. However, it has been altered so that it does not increase under relocation requests that can bring taxis arbitrarily far apart.

The constants a and b determining the movement speeds of the algorithm and the constants $0 < \psi < 1 < \beta < \alpha$ used in this analysis are chosen[2] to obey the following hierarchy:

$$a \ll b \ll \psi \ll 1 \ll \alpha a \ll \beta \ll \alpha b. \tag{3.2}$$

Terms higher in the hierarchy are assumed to be much larger than those lower in the hierarchy. Hence, to demonstrate that $A \le 0$ for some expression A, as long as the coefficients of all terms are bounded, it suffices to show that some term in A is < 0 and all higher order terms in A are ≤ 0.

[2]We can express all constants in terms of a sufficiently small positive constant $0 < \epsilon \ll 1$. In ascending order, the constants are $a = \epsilon^4, b = \epsilon^2, \psi = \epsilon, \beta = \epsilon^{-2}, \alpha = \epsilon^{-5}$.

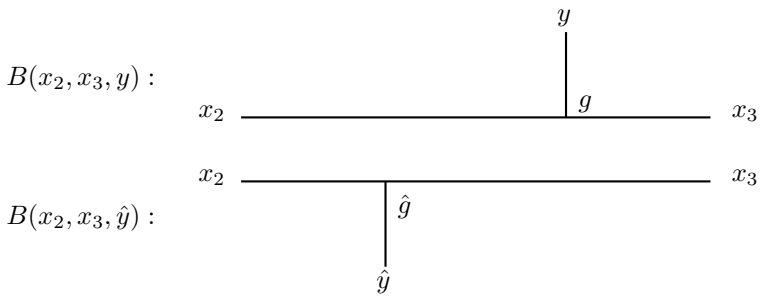

Figure 3.2: When the two passive online taxis x_2 and x_3 are matched to two offline taxis y and $\hat{y}$, the relative distance of the branching points g and $\hat{g}$ to each passive taxi determines the minimum matching. In the minimum matching of the above scenario, x_2 is matched to $\hat{y}$ and x_3 is matched to y.

Our analysis will repeatedly make use of the following lemma to restrict how the passive taxis can be matched to each other in the minimum matching. Concretely, if we consider the two tripods each with endpoints comprising the two passive online taxis and one of the offline taxis, Lemma 3.2 states that in the minimum matching each offline taxi is matched to the online taxi which is relatively closer to the branching point of the corresponding tripod. See Figure 3.2. This generalizes the intuitive property from on a line metric that a minimum matching would match online to offline taxis in left-to-right order (e.g., the leftmost online taxi to the leftmost offline taxi etc.), and it holds even in the presence of discount intervals.

LEMMA 3.2. *Let $y, \hat{y}$ be the two passive offline taxis. Let g and $\hat{g}$ be the branching points of $B(x_2, x_3, y)$ and $B(x_2, x_3, \hat{y})$, respectively. Then x_2 is matched to $\hat{y}$ and x_3 is matched to y in a minimum matching iff $x_2\hat{g} \leq x_2 g$.*

Proof. The only other matching to consider is when x_3 is matched to $\hat{y}$ and x_2 to y. The discount factors over the distance from each online/offline taxi to the nearer branching point (i.e. $\min(x_3 g, x_3\hat{g})$, $\min(x_2 g, x_2\hat{g})$, yg and $\hat{y}\hat{g}$) are unchanged, so these distances have the same contribution to the cost of either matching. Hence, the only change in matching contribution from swapping from the matching in the lemma statement to this other matching is due to the change in the distance from each taxi to its matched partner by $x_2 g - x_2\hat{g}$. Therefore, this distance is non-negative iff the matching in the lemma statement is a minimum matching. ∎

3.4.1 Online and Offline Movement. In this section we will show that

$$\text{COST}_t + \Delta\Phi_t^{\text{off}} + \Delta\Phi_t^{\text{on}} \leq \kappa \text{OPT}_t. \tag{3.3}$$

Without loss of generality, we can assume that the offline algorithm only moves one taxi when serving each request, since delaying any other movements to a later request does not increase its overall cost.

LEMMA 3.3. *During the offline movement phase it holds that $\Delta\Phi_t^{\text{off}} \leq \alpha \text{OPT}_t$.*

Proof. Since there are no changes to the online taxi locations or q_2, q_3, we have that $\Delta\Sigma_t^{\text{off}} = 0$. Considering the offline taxi moved by the offline algorithm, the change in the distance to its matched online taxi is no greater than its moved distance, due to the triangle inequality. Furthermore, since M is a weighted sum of distances of the minimal matching where the weights are all no greater than 1, we have that $\Delta M_t^{\text{off}} \leq \text{OPT}_t$, which suffices to give the claim. ∎

The case where the offline algorithm serves the current request with the active offline taxi y_1 is simple. Then $\text{OPT}_t = y_1 r = x_1 r$ is an upper bound on the distance moved by the active online taxi. The distances moved by the passive online taxis are at most a constant factor larger. Furthermore, $\Delta\Phi^{\text{on}}$ is no more than $O(\alpha + \beta)$ times the total distance moved by all online taxis. Hence, using Lemma 3.3 we get that $\text{COST}_t + \Delta\Phi_t^{\text{off}} + \Delta\Phi_t^{\text{on}}$ is at most a constant factor larger than OPT_t, so for κ sufficiently large this suffices to show (3.3).

This leaves us in the case where the offline algorithm moves a passive taxi. Therefore, y_1 does not move and there is a passive offline taxi at r. Our goal is to show the following claim.

CLAIM 3.4. *If the offline algorithm moves a passive taxi, then during the online movement phase,*

$$\text{COST}_t + \Delta\Phi_t^{\text{on}} \leq 0. \tag{3.4}$$

Together, Lemma 3.3 and Claim 3.4 suffice to prove (3.3) in the case where the offline algorithm moves a passive taxi. Using the following corollary on the matched partners of the passive taxis, we will show that the passive online taxi that is not matched to r (and therefore could be moving away from its matched partner) only increases its matching contribution at a rate of $(1-\psi)$.

LEMMA 3.5. *Let y be any point and let the rates of change of x_2y, x_3y, x_2x_3 be x_2y', x_3y', x_2x_3' respectively. Let g be the branching point of $B(x_2, x_3, y)$. Then the rates of change of x_2g, x_3g, yg are respectively*

$$x_2g' = \frac{x_2y' - x_3y' + x_2x_3'}{2}, \quad x_3g' = \frac{-x_2y' + x_3y' + x_2x_3'}{2}, \quad yg' = \frac{x_2y' + x_3y' - x_2x_3'}{2}.$$

Proof. This follows from (2.1). ∎

COROLLARY 3.6. *In the movement cases with two unobstructed passive taxis, if one of the passive taxis x_i is matched to r and the other passive taxi x_j moves at speed c_j, then x_j's matching contribution increases at a rate of at most $c_j(1-\psi)$.*

Proof. Let c_i be the movement speed of x_i, let x_j be matched to y and let e, g be the branching points of $B(x_i, x_j, r), B(x_i, x_j, y)$ respectively.

Due to Lemma 3.5, since x_i and x_j move towards each other, the total length of the segment x_jg in $B(x_i, x_j, y)$ does not increase. Then it suffices to show that the length of the undiscounted region for x_j in the segment x_jg does not increase. This length is only positive when $x_jq_j < x_jg$. Then by Lemma 3.2, we also have that $x_jq_j < x_je$. In this case, both x_j and q_j move towards e at the same speed, so the distance x_jq_j does not change. Thus, the length of the undiscounted region on x_jg cannot increase. ∎

We can now prove Claim 3.4 by showing that it holds over all movement cases.

Proof of Claim 3.4. From ordering (3.2), we have the following hierarchy on terms which we will use in this proof:

$$\alpha a \ll \beta \ll \alpha b \ll \alpha\psi \ll \alpha. \tag{3.5}$$

Let COST' and Φ' be the sum of the movement speeds of all online taxis and the rate of change of Φ, respectively. We show that in all cases, $\text{COST}' + \Phi' \leq 0$. Technically COST' and Φ' are not defined when TRIPODTRACKER switches from one movement case to another, but these changes happen finitely often and do not affect the proof. In all cases, the movement of x_1 away from its matched partner contributes a to M', so it contributes αa to Φ'. In all movement cases, we will show that some term strictly higher than αa in the ordering (3.5) is < 0 and all higher terms are ≤ 0. Hence, we can ignore COST' and x_1's contribution to the matching for the rest of the analysis.

In all movement cases with two unobstructed passive taxis, we have that $\Sigma' \leq 2$. We now analyze Φ', using Corollary 3.6 to identify the matching contribution of the passive taxi not matched to r.

- If the passive taxi x_i matched to r satisfies $x_iq_i < x_ie$, it contributes ≤ -1 to M'. Then the other passive taxi contributes $\leq (1+b)(1-\psi)$ to M'. Hence, the $\alpha\psi$ term in Φ' is < 0 and all higher terms in ordering (3.5) are ≤ 0.

- Suppose the passive taxi x_i matched to r satisfies $x_iq_i \geq x_ie$ and the other passive taxi x_j satisfies $x_jq_j < x_je$. Then x_i contributes $-(1+b)(1-\psi)$ to M' and x_j contributes $\leq (1-\psi)$. Hence, the αb term in Φ' is < 0 and all higher terms in ordering (3.5) are ≤ 0.

- Otherwise, the passive taxi x_i matched to r satisfies $x_iq_i = x_ie$ and the other passive taxi x_j satisfies $x_jq_j = x_je$. Then x_i contributes $-(1+b)(1-\psi)$ to M' and x_j contributes $\leq (1+b)(1-\psi)$. Furthermore, $\Sigma' = -(1+b) < 0$ and all terms higher than β in ordering (3.5) are ≤ 0.

When there is only one unobstructed passive taxi, by Lemma 3.2, the unobstructed taxi is moving towards its matched partner. Hence, the α term of Φ' is < 0, so by ordering (3.5), the claim $\text{COST}' + \Phi' \leq 0$ holds. ∎

3.4.2 Reorganization and Relocation. For the remaining phases, the online and offline taxis no longer move. Hence, we will show that $\Delta\Phi_t^{\text{org}} \le \alpha \cdot \psi \cdot \text{OPT}_t$ and $\Delta\Phi_t^{\text{rel}} \le 0$, which, together with the previous section, suffices to show (3.1) for sufficiently large κ. To avoid ambiguity, we consider the reindexing of the offline active taxi to happen in the reorganization phase.

We begin with the reorganization phase.

LEMMA 3.7. *During reorganization it holds that* $\Delta\Sigma_t^{\text{org}} \le 0$.

Proof. If $x_1 = r$, no changes take place during reorganization, so $\Delta\Sigma_t^{\text{org}} = 0$. Otherwise, without loss of generality $x_3 = r$. If $\ell_3 \le x_1x_3$, then $\ell_3^{\text{new}} = 0$ after the reorganization and Σ goes to 0. Otherwise, both q_2 and q_3 are on the x_2 edge of $B(x_1, x_2, x_3)$ and $q_2f \ge q_3f > x_1f$ (as in Figure A.1). Prior to reorganization, $\Sigma = \max(0, \min(\ell_2, \ell_3) - q_2q_3)$. After the reorganization, the distance q_2q_3 increases by x_1f. If Σ was determined by ℓ_3 prior to the reorganization, $\ell_3^{\text{new}} \le \ell_3$ so $\Delta\Sigma_t^{\text{org}} \le 0$. Otherwise Σ was determined by ℓ_2 prior to the reorganization and $\ell_2^{\text{new}} \le \ell_2 + x_1f$, so $\Delta\Sigma_t^{\text{org}} = 0$. ∎

LEMMA 3.8. *During reorganization, it holds that* $\Delta M_t^{\text{org}} \le \psi \cdot \text{OPT}_t$.

Proof. We compare the matching potentials before and after reorganization. We consider four cases, defined by whether the online/offline algorithm serves the request with an active/passive taxi. In all cases except the one where an online passive taxi and the offline active taxi serve the request, we will prove the stronger claim that $\Delta M_t^{\text{org}} \le 0$. Note that reorganization involves reindexing the active online/offline taxis to the taxis at r, which may change the set of matchings we consider.

If both active taxis serve the request, no changes in the matching happen during reorganization so $\Delta M_t^{\text{org}} = 0$.

Suppose that the active online taxi x_1 and a passive offline taxi y_3 serve the request. It suffices to show that reorganizing y_3 to be the new active offline taxi by matching x_1 to y_3 and x_3 to y_1 does not increase the matching potential. Then the matching contribution of the (x_2, y_2)-pair does not change. The matching contribution of the (x_1, y_1)-pair was x_1y_1 and this decreases to 0 in the (x_1, y_3)-pair. When changing x_3's partner from y_3 to y_1, its matching contribution will be defined on the tripod $B(x_2, x_3, y_1)$ instead of $B(x_2, x_3, y_3)$. By the triangle inequality, the distance from x_3 to its matched partner increases by at most $y_3y_1 = x_1y_1$. Furthermore, by Lemma 2.1, the corresponding x_3-edges of these tripods differ in length by at most x_1y_1. Hence, x_3's matching contribution increases by at most x_1y_1, since in the worst case the entire increase in x_3's distance to its matched partner is incurred on the x_3 edge of the corresponding tripods and is discounted at rate 1 because the amount of x_3's interval used in the matching does not increase. Therefore, the decrease in x_1's matching contribution is greater than the increase in x_3's, so $\Delta M_t^{\text{org}} \le 0$.

In the remaining cases, the passive online taxi x_3 serves the request. We begin by showing that x_2's matched partner is never the offline taxi that will become active.

CLAIM 3.9. *Before reorganization, there exists a minimum matching where either* x_1 *or* x_3 *is matched to* r.

Proof. If the offline algorithm served the current request with the active taxi y_1, then x_1 is matched to r. Otherwise, one of the two passive taxis is matched to r. We can apply Lemma 3.2 with $y = r$. Since $x_3 = r = e$, the lemma shows that there is a minimum matching where x_3 is matched to r. ∎

Therefore, despite reindexing the active taxi, it is always valid for x_2's partner y_2 to be unchanged after reorganization and for y_2 to remain passive.

CLAIM 3.10. *If* x_3 *serves the request and* x_2*'s matched partner does not change, then* x_2*'s matching contribution does not increase after reorganization.*

Proof. Recall that x_2 is matched to y_2 before reorganization. We show that the cost of the (x_2, y_2)-pair in $B(x_2, x_1, y_2)$ after reorganization is no larger than the cost of the pair in $B(x_2, x_3, y_2)$ before reorganization. Let f, g, h be the branching points of $B(x_1, x_2, x_3)$, $B(x_2, x_3, y_2)$ and $B(x_2, x_1, y_2)$, respectively. It suffices to show that the amount of $(1-\psi)$-discounted region in the (x_2, y_2) matching contribution does not decrease. This is true because either the entire path from x_2 to y_2 is discounted after reorganization or the size of the of the discounted region increases by at least x_1f on the x_2-edge and changes by $y_2h - y_2g$ on the y_2-edge. The total change in the amount of discounted region is thus at least $x_1f + y_2h - y_2g = x_1f + x_2g - x_2h$. From (2.1), we have

$$x_1f = \frac{1}{2}(x_1x_2 + x_1x_3 - x_2x_3), \quad x_2g = \frac{1}{2}(x_2x_3 + x_2y_2 - x_3y_2), \quad x_2h = \frac{1}{2}(x_1x_2 + x_2y_2 - x_1y_2).$$

Hence,

$$x_1 f + x_2 g - x_2 h = \frac{1}{2}(x_1 x_3 - x_3 y_2 + x_1 y_2) \geq 0,$$

which suffices to show that x_2's matching contribution does not increase during reorganization. ∎

It remains to show that x_1 and x_3's matching contributions do not increase too much. Recall that the active and the remaining passive offline taxis are y_1 and y_3, respectively.

If passive taxis x_3 and y_3 serve the request, then x_3's matching contribution is 0 before and after reorganization. Furthermore, x_1's discount factor over its entire matching only changes from 1 to at most 1, which does not increase its matching contribution.

Otherwise, x_3 and y_1 serve the request. We will show that matching the online taxi at r ($x_3 = x_1^{\text{new}}$) to r and the other online taxi ($x_1 = x_3^{\text{new}}$) to y_3 after reorganization increases the matching contribution by no more than $\psi \cdot x_1 x_3$. This suffices to prove the lemma since $x_1 x_3 = x_1 r \leq \text{OPT}_t$, as the offline taxi y_1 moved from the *old* location of x_1 to r, and the new location of x_1 can only be closer to r. The matching contribution of the (x_1, r)-pair was $x_1 x_3$ and this decreases to 0 in the (x_1^{new}, r)-pair. Therefore, it suffices to show that the matching contribution of the new (x_3^{new}, y_3)-pair exceeds that of the old (x_3, y_3)-pair by at most $(1 + \psi) \cdot x_1 x_3$. By the triangle inequality the total increase in the distance of this pair is at most $x_1 x_3$ and by Lemma 2.1 the length of the x_3-edge of $B(x_2, x_3, y_3)$ increases by at most $x_1 x_3$ to give the x_1-edge in $B(x_2, x_1, y_3)$. This contributes an increase of at most $x_1 x_3$ to the matching potential, since in the worst case the entire increase in distance is incurred on what was originally the x_3-edge and is undiscounted. Due to line 18, ℓ_3 also decreases by at most $x_1 x_3$, contributing an additional $\psi \cdot x_1 x_3$ to the matching potential. Hence, the total increase in matching potential is at most $(1 + \psi) \cdot x_1 x_3$, giving the lemma. ∎

Finally, we consider the relocation phase.

LEMMA 3.11. *During line 21 it holds that* $\Delta\Phi_t^{\text{rel}} \leq 0$.

Proof. There are no changes to the passive taxi locations or q_2, q_3, so Σ is unchanged. The active taxis continue to share the same location, so M is also unchanged. ∎

This completes the proof of our main theorem.

THEOREM 3.12. *TripodTracker is a κ-competitive algorithm for the hard 3-taxi problem.*

4 Conclusion and $k > 3$. Our result shows that competitive algorithms for the k-taxi problem on general metrics exist beyond the previous barrier of $k = 2$. The obvious open question is whether our result can be further extended to general k, with a competitive ratio depending only on k. We make the following two observations.

First, our proof continues to make use of the idea of distinguishing the active taxi from the passive taxi(s) as per the previous proof for $k = 2$, but extends on this by further distinguishing the two passive taxis with intervals. Understanding these intervals more deeply (and beyond our current level of understanding) seems crucial for extending the result to general k. The rest of this paragraph recapitulates the authors' current understanding. As alluded to in our intuition section, we interpret them as marking regions where one passive taxi holds "more responsibility" than the other passive taxi. Note that the intervals are fully specified by their lengths ℓ_i and the locations of the passive taxis, so we can view ℓ_i as the "responsibility score" of taxi x_i. Recall that we interpret the active taxi as special and "unfit for holding responsibility". Accordingly, the only time when some ℓ_i can grow is in line 17 of the algorithm, when the previously active taxi x_1 becomes passive: Taxi x_2 (which was passive before and remains passive) used to be more responsible than x_1 (which was active and now becomes passive). To record this, we increase the responsibility score ℓ_2 by the part of the distance from x_2 to x_1 that was outside any bridge $B(x_2, x_3)$ (i.e., $x_1 f$ in line 17). Similarly, when a passive taxi becomes active, it loses its responsibility, and the responsibility is inherited by the newly passive taxi (except it is reduced by the distance between these two taxis, corresponding to the fact that the interval start point moves by this distance; line 18). For $k = 3$, we need intervals only to distinguish the relative responsibility between the two passive taxis. For $k > 3$, a generalization of our approach might involve distinguishing responsibility levels between any pair of passive taxis. In fact, it might be more natural to encode the special role of the active taxi (and its lack of responsibility) also through such intervals.

Second, tripods currently exactly capture the distances between their three endpoints, and are used in this proof to dynamically embed the two passive taxis and any third point into a tree structure. Hierarchical Separated Trees (HSTs) are a natural candidate to replace tripods when $k > 3$. We suspect that our algorithm may have an alternative interpretation in terms of dynamic embeddings into HSTs. Specifically, intervals with small discount factors, which stimulate faster movement, suggest that their endpoints are embedded to nearby points in the HST. Our distorted matching potential then corresponds to a minimum matching with respect to HST distances. Understanding our algorithm through such a lens may give insights into how such an HST embedding should evolve dynamically. We note that dynamic HST embeddings have been used successfully for the k-server problem [10].

A useful intermediate step would be to consider k taxis on the line metric. On the line, all tripods have at least one edge of length 0, and any generalizations of tripods to higher k's are similarly more restricted.

Finally, we hope that our techniques can inspire progress on other variants of the k-server problem where existing results are limited to $k = 2$ or restricted metrics, such as the weighted k-server problem and the generalized k-server problem [19, 24, 23, 3, 5, 12, 1, 4, 6].

A Example of TripodTracker Serving a Request.

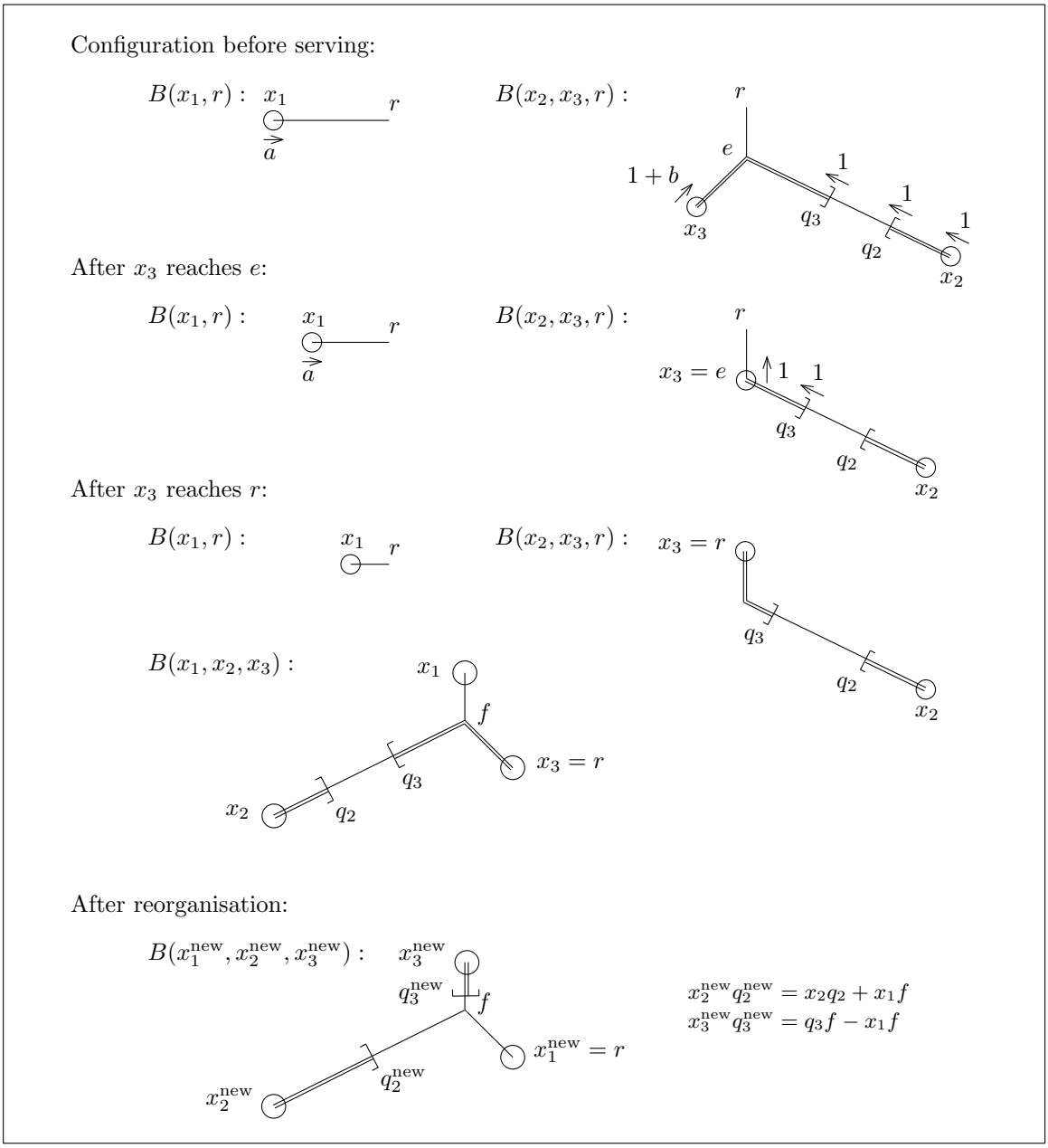

Figure A.1: Example of TripodTracker serving a request. Since the bridges and tripods change over the course of serving a request, each diagram displays the current state of each bridge or tripod at that point in the algorithm.

References

[1] N. Ayyadevara and A. Chiplunkar, *The randomized competitive ratio of weighted k-server is at least exponential*, in 29th Annual European Symposium on Algorithms, ESA, P. Mutzel, R. Pagh, and G. Herman, eds., 2021.

[2] N. Bansal, N. Buchbinder, A. Madry, and J. Naor, *A polylogarithmic-competitive algorithm for the* k*-server problem*, J. ACM, 62 (2015), pp. 40:1–40:49.

[3] N. Bansal, M. Eliás, and G. Koumoutsos, *Weighted k-server bounds via combinatorial dichotomies*, in 58th IEEE Annual Symposium on Foundations of Computer Science, FOCS, 2017.

[4] N. Bansal, M. Eliás, G. Koumoutsos, and J. Nederlof, *Competitive algorithms for generalized* k*-server in uniform metrics*, ACM Trans. Algorithms, 19 (2023), pp. 8:1–8:15.

[5] M. Bienkowski, L. Jez, and P. Schmidt, *Slaying hydrae: Improved bounds for generalized k-server in uniform metrics*, in 30th International Symposium on Algorithms and Computation, ISAAC, 2019.

[6] A. Bijoy, A. Mondal, and A. Chiplunkar, *Weighted* k*-server admits an exponentially competitive algorithm*, in Proceedings of the 2026 ACM-SIAM Symposium on Discrete Algorithms, SODA, 2026.

[7] S. Bubeck, N. Buchbinder, C. Coester, and M. Sellke, *Metrical service systems with transformations*, in 12th Innovations in Theoretical Computer Science Conference, ITCS, 2021.

[8] S. Bubeck, C. Coester, and Y. Rabani, *Shortest paths without a map, but with an entropic regularizer*, in 63rd IEEE Annual Symposium on Foundations of Computer Science, FOCS, 2022.

[9] S. Bubeck, C. Coester, and Y. Rabani, *The randomized k-server conjecture is false!*, in Proceedings of the 55th Annual ACM Symposium on Theory of Computing, STOC, 2023.

[10] S. Bubeck, M. B. Cohen, Y. T. Lee, J. R. Lee, and A. Madry, *k-server via multiscale entropic regularization*, in Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, STOC, 2018.

[11] N. Buchbinder, C. Coester, and J. Naor, *Online k-taxi via double coverage and time-reverse primal-dual*, Math. Program., 197 (2023), pp. 499–527.

[12] A. Chiplunkar and S. Vishwanathan, *Randomized memoryless algorithms for the weighted and the generalized* k*-server problems*, ACM Trans. Algorithms, 16 (2020), pp. 14:1–14:28.

[13] M. Chrobak, H. J. Karloff, T. H. Payne, and S. Vishwanathan, *New results on server problems*, SIAM J. Discret. Math., 4 (1991), pp. 172–181.

[14] M. Chrobak and L. L. Larmore, *An optimal on-line algorithm for k-servers on trees*, SIAM J. Comput., 20 (1991), pp. 144–148.

[15] C. Coester and E. Koutsoupias, *The online k-taxi problem.*, in Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing, STOC, 2019.

[16] S. Dehghani, S. Ehsani, M. Hajiaghayi, V. Liaghat, and S. Seddighin, *Stochastic k-server: How should uber work?*, in 44th International Colloquium on Automata, Languages, and Programming, ICALP, 2017.

[17] A. Fiat, D. P. Foster, H. J. Karloff, Y. Rabani, Y. Ravid, and S. Vishwanathan, *Competitive algorithms for layered graph traversal*, SIAM J. Comput., 28 (1998), pp. 447–462.

[18] A. Fiat, Y. Rabani, and Y. Ravid, *Competitive k-server algorithms (extended abstract)*, in 31st Annual Symposium on Foundations of Computer Science FOCS, 1990.

[19] A. Fiat and M. Ricklin, *Competitive algorithms for the weighted server problem*, Theor. Comput. Sci., 130 (1994), pp. 85–99.

[20] A. Gupta, A. Kumar, and D. Panigrahi, *Poly-logarithmic competitiveness for the* k*-taxi problem*, in Proceedings of the 2024 ACM-SIAM Symposium on Discrete Algorithms, SODA, 2024.

[21] E. Koutsoupias and C. H. Papadimitriou, *On the k-server conjecture*, J. ACM, 42 (1995), pp. 971–983.

[22] M. S. Manasse, L. A. McGeoch, and D. D. Sleator, *Competitive algorithms for on-line problems*, in Proceedings of the 20th Annual ACM Symposium on Theory of Computing, STOC, 1988.

[23] R. Sitters, *The generalized work function algorithm is competitive for the generalized 2-server problem*, SIAM J. Comput., 43 (2014), pp. 96–125.

[24] R. A. Sitters and L. Stougie, *The generalized two-server problem*, J. ACM, 53 (2006), pp. 437–458.