Discrete Online TSP*

Mauro Aprea, Esteban Feuerstein, Gustavo Sadovoy, and
Alejandro Strejilevich de Loma

Departamento de Computación, Facultad de Ciencias Exactas y Naturales,
Universidad de Buenos Aires, Pabellón I, Ciudad Universitaria,
(1428) Capital Federal, Argentina

Abstract. In this paper we introduce a discrete version of the online traveling salesman problem (DOLTSP). We represent the metric space using a weighted graph, where the server is allowed to modify its route only at the vertices. This limitation directly affects the capacity of the server to react and increases the risk related to each decision. We prove lower bounds on the performance of deterministic online algorithms in different scenarios of DOLTSP, and we present distinct algorithms for the problem, some of them achieving the best possible performance. We measure the performance of the algorithms using competitive analysis, the most widely accepted method for evaluating online algorithms. Besides, we perform an empirical simulation on paths, generating a significant set of instances and measuring the quality of the solutions given by each algorithm. Our experiments show that algorithms with the best competitive ratio do not have the best performance in practice.

Key words: TSP, discrete metric spaces, online algorithms

1 Introduction

Numerous variations of the Vehicle Routing Problem (VRP) have been defined [1–4]. Many of these variations assume that the input is completely known when the solution is computed. However, there are many situations in which decisions must be made based on partial information, and the solution must be built or even executed before the input is completely known, what is usually known as *online optimization* [5, 6]. Think for example of a salesman with a cellular phone, or a fleet of vehicles equipped with radios that must collect and deliver packages at different locations, and many other transportation problems in which the itinerary can be modified during its execution. In the online versions of VRPs, a sequence of requests is posed to an algorithm that has to decide how to move the servers to satisfy the requests without knowledge of future requests.

Previous works about online VRPs [7–10, 1, 11] have considered that the servers move in a *continuous* metric space. In this scenario the servers can change direction at any time while they are moving from one point to another. However,

* Research supported in part by UBACyT projects X143 and X212, and by ANPCyT project PICT-2006-01600.

in some applications this may not be the case, and it is preferable to model the problem using a *discrete* metric space. For example, if a server is on a road network of freeways and a request arrives while the server is moving between two exits, the server has to proceed to the next exit before being able to change its plan. Online routing on discrete metric spaces appears a priori to be "harder" than the continuous counterpart, as in the former there are less opportunities to revise the plan, and thus the risk associated to each decision may be higher.

In this paper we consider the online version of the Traveling Salesman Problem (TSP) on discrete metric spaces. We call this variant *Discrete Online Traveling Salesman Problem* (DOLTSP). In an instance of DOLTSP, a server that travels at unit speed must visit the vertices of a graph in order to satisfy requests that are presented along time, finishing the service as early as possible. As in previous works about online routing, we consider two versions of the problem: one *Homing* (HDOLTSP), in which the journey of the server must finish at the same vertex where it starts, and one *Nomadic* (NDOLTSP), in which it can finish anywhere in the graph. Note that the associated offline problems are the same as in the continuous case, namely the *Vehicle Routing Problem with release times* [12].

We propose deterministic online algorithms for the two versions of DOLTSP, and we measure their performance using *competitive analysis* [13]. In this widely used framework the cost of an algorithm is compared to that of an *optimal offline adversary* that knows the whole input in advance. We consider two types of adversaries: a *standard* adversary with unrestricted power, and a *fair* adversary [10] that must keep the server inside the region where requests have already been presented. We analyze two classes of online algorithms: *zealous* algorithms [10] that keep working while there is something to do, and *cautious* algorithms that may halt the server even when there is pending work.

Most of our results hold on any (non trivial) graph. However, sometimes we focus our attention on *paths*, i.e. graphs in which the server can only move in two opposite directions which we refer as "left" (or negative) and "right" (or positive). Moreover, certain results are only valid for *halfpaths*, that are paths where the starting vertex is the leftmost one. Notice that paths and halfpaths are the discrete analogons of the real line and halfline, respectively. This family of graphs allows to capture many interesting applications, like the previously mentioned of a highway, an elevator, a stacker-crane moving on a track, or the radial movement of the read/write head in a hard disc drive.

A summary of our theoretical results is given in Table 1, where lower bounds that hold on any graph appear in boldface. The two lower bounds not in boldface are valid on halfpaths with a certain distribution of vertices (and then also on paths and trees). We found that, in general, DOLTSP is harder than its continuous counterpart. As expected, HDOLTSP is easier than NDOLTSP, the fair adversary is weaker than the standard one, and zealous algorithms are weaker than cautious ones.[1]

[1] In the Online Asymmetric Traveling Salesman Problem (OL-ATSP) a server moves in a not necessarily symmetric space [8]. This problem can be viewed as a generalization

Table 1. Summary of theoretical results

		zealous			cautious		
		lower	upper bound		lower	upper bound	
		bound	path	general	bound	halfpath	general
Homing	fair	**2**	2	3	$\approx$ **1.618**	$\approx$ 1.618	$\approx$ 2.618 [8]
	standard	**2**	2	3	$\approx$ 1.707	$\approx$ 1.707	$\approx$ 2.618 [8]
Nomadic	fair	**3**	3	3	$\approx$ **1.839**	2	(unknown)
	standard	**3**	3	3	2	2	(unknown)

Besides, we perform empirical simulations on paths. We find that in practice the zealous strategies we devised are better than the cautious ones. Nevertheless, the empirical studies also ratify the idea that waiting is profitable in a worst case sense.

2 Basic Definitions and Notation

2.1 DOLTSP

The input of DOLTSP consists of a graph $G = (V, E)$ with a positive length associated to each edge $e \in E$, a distinguished vertex $o \in V$ (the *origin*), and a sequence σ of requests $r_i = (t_i, v_i)$, where $v_i \in V$, and $t_i \in \mathbb{R}_{\geq 0}$ is a *release time* representing the moment at which r_i is presented. These moments form an ordered sequence in the sense that $t_i \leq t_j$ if $i < j$. At time 0 a server is located at the origin o, and must serve all requests. With this purpose it has to move through the edges of E at unit speed and visit each vertex v_i at some moment not earlier than t_i. The server cannot change direction while traversing and edge. We consider two variants of DOLTSP: in *Homing* DOLTSP (HDOLTSP) the server must return to the origin after serving all requests, while in *Nomadic* DOLTSP (NDOLTSP) the journey can finish anywhere.

For every pair of vertices $v, w \in V$ we denote with $d(v, w) = d(w, v)$ the distance between them, that is, the length of a shortest path joining them. We denote with $\bar{v}$ the distance of v to the origin o. We assume that the graph G is not trivial (it has at least one vertex apart from o), and that all lengths associated to edges satisfy the triangle inequality.

An algorithm for DOLTSP must decide the movements of the server with the goal of ending its work as soon as possible. An *online* algorithm has to execute

of DOLTSP, by replacing each edge of the graph by two directed arcs of the same length, and considering a particular notion of distance (the cost of changing direction while traversing an arc from x to y, is the cost of reaching y plus the cost of going back to x). Thus, the algorithms presented in [8] mantain their performance when they are used for DOLTSP. However, we obtain strictly better performances on paths and for NDOLTSP.

each movement without knowledge of unreleased requests. On the contrary, an *offline* algorithm can decide based on the whole sequence of requests.

2.2 Competitive Analysis and Adversaries

Competitive analysis [13, 14] is a type of worst case analysis where the performance of an algorithm for a problem is compared to that of an optimal offline algorithm. The measure of performance used in competitive analysis is the *competitive ratio*. We say that an algorithm `ALG` for DOLTSP is ρ-competitive if and only if for every sequence of request σ we have $C_{\mathrm{ALG}}(\sigma) \leq \rho \cdot C_{\mathrm{OPT}}(\sigma)$, where $C_{\mathrm{ALG}}(\sigma)$ is the cost of `ALG` for σ, and $C_{\mathrm{OPT}}(\sigma)$ is the cost of an optimal offline algorithm that knows the whole input sequence in advance.

It is usually useful to see competitive analysis as a game between an *online player* and an *offline adversary*. The former tries to find a good solution for a sequence of requests generated by the latter, who knows the online strategy and tries to maximize the ratio between both costs. Thus, we use the terms *optimal offline algorithm* and *adversary* interchangeably.

Competitive analysis is sometimes criticized for its excessive pessimism [15]. With the aim of attenuating this situation, different alternative measures have been proposed. One of them is known as *comparative analysis*, in which the adversary is restricted in some sense. For online routing problems, a form of comparative analysis consists of using, instead of a *standard*, unrestricted adversary, a *fair* adversary that is required to move the server inside the region where requests have already been presented. Fair adversaries have been originally proposed in [10] for (continuous) OLTSP. For DOLTSP we define this class of adversaries as follows. At any given moment t, the *fair region* is the closure under shortest paths of the set of vertices formed by the origin and the vertices where requests have been presented. An adversary is *fair* if at every moment its server is in the subgraph induced by the fair region at that moment.

3 Zealous Algorithms

In this section we propose and analyze simple and intuitive online algorithms for DOLTSP. As we will see, these algorithms are members of a natural class of algorithms that keep working as long as there is something to do.

The first online algorithm is known as Replan (`REP`). This algorithm was well studied in the context of distinct online optimization problems, and it consists of adjusting the solution each time a new request is presented. In the case of DOLTSP, this means computing a new itinerary that allows the server to satisfy all pending requests in the least possible time. If new requests are presented when the server is not at a vertex of the graph, the new route is computed as soon as the server reaches a vertex.

We will analyze the performance of `REP` for the particular case of paths. For doing that, we need to introduce the concept of *extreme* vertices.

Definition 1. *Given an instance of DOLTSP on a path, at any given moment consider the set S of vertices that* `REP` *has yet to visit. This set contains the vertices of unserved requests, and in HDOLTSP includes also the origin. We call* left (resp. right) extreme *the leftmost (resp. rightmost) vertex of S.*

Note that for HDOLTSP on paths, the left (resp. right) extreme is located in the left (resp. right) halfpath. However, one or both extremes could be the origin. This implies that at least one of the extremes is in the same halfpath where the server is. Among the extremes that are in the same halfpath that the server, let F be the extreme that is farthest from the origin. It is easy to see that for HDOLTSP on paths the route computed by `REP` is as follows: move to extreme F, then to the other extreme, and finally to the origin.

In NDOLTSP we cannot assume any order between the extremes and the origin. However, the route computed by `REP` is simpler: move to the extreme that is nearest to the server, and then to the other extreme.

Knowing the routes computed by `REP` for DOLTSP on paths, we are ready to analyze its competitiveness. The following results show that `REP` is 2-competitive for HDOLTSP and 3-competitive for NDOLTSP. This is valid on any path (even a halfpath) against both fair and standard adversaries.

Theorem 2. *Algorithm* `REP` *is* 2*-competitive for HDOLTSP on any path against both fair and standard adversaries.*

Proof. Let σ be any sequence of requests. Let L and R be respectively the leftmost and rightmost vertices that must be visited to serve all the requests of σ, including the origin. Clearly, at any moment, the left and right extremes are located between L and R. More precisely, the left extreme is between L and the origin, and the right extreme is between the origin and R. Besides, the server of `REP` is always between L and R, because it only moves to serve a request or to return to the origin. Let T be the moment in which the last request of σ is presented. Two situations can occur.

1. At time T, the online server is at a vertex or traversing an edge that moves it away from the origin. To complete its work, the server will first move to an extreme, then to the other extreme, and finally to the origin, ending at most at time $T + 2\bar{L} + 2\bar{R}$. Since T and $2\bar{L} + 2\bar{R}$ are lower bounds to the optimal offline cost, we have
$$\frac{C_{\mathrm{REP}}(\sigma)}{C_{\mathrm{OPT}}(\sigma)} \leq \frac{T}{C_{\mathrm{OPT}}(\sigma)} + \frac{2\bar{L} + 2\bar{R}}{C_{\mathrm{OPT}}(\sigma)} \leq 2 \ .$$
2. At time T, the online server is traversing an edge that brings it nearer to the origin. Let $v \neq o$ be the vertex where that move starts. We know that v is between L and R. There are two possibilities.
   (a) After the server leaves v, no new request is presented at v or at any other vertex in the same halfpath farther away from the origin than v. As in the previous situation, the server will first move to an extreme, then to the other extreme, and finally to the origin, with a total cost of at most $T + 2\bar{L} + 2\bar{R}$. And again, this is at most twice the optimal offline cost.

(b) The last situation is when at least one of those requests is presented. Let r be one of them, presented at time t at vertex x. Clearly, the length of the edge that the online server is traversing is at most $\bar{x}$ and, as we said, the movement started before time t. Then, before time $t + \bar{x}$ the server arrives to a vertex and can replan its tour, ending its job at most at time $t + \bar{x} + 2\bar{L} + 2\bar{R}$. Since the optimal offline cost is lower bounded by $t + \bar{x}$ and by $2\bar{L} + 2\bar{R}$, we obtain

$$\frac{C_{\mathrm{REP}}(\sigma)}{C_{\mathrm{OPT}}(\sigma)} \leq \frac{t + \bar{x}}{C_{\mathrm{OPT}}(\sigma)} + \frac{2\bar{L} + 2\bar{R}}{C_{\mathrm{OPT}}(\sigma)} \leq 2 \ .$$

□

Theorem 3. *Algorithm* `REP` *is* 3*-competitive for NDOLTSP on any path against both fair and standard adversaries.*

Proof. Let σ be any sequence of requests. Let L, R and T be as in the previous proof, and note that also in NDOLTSP the server of `REP` is always between L and R. Until time T the cost of the algorithm is obviously T. Consider the route of the server from this moment on. If the server is traversing and edge, it completes the movement, and then the server moves to its nearest extreme, with a total cost of at most $d(L, R)$, that is, the distance between L and R. To complete its job, the server moves to the other extreme, again with a cost of at most $d(L, R)$. Since T and $d(L, R)$ are lower bounds for the optimal offline cost, we have

$$\frac{C_{\mathrm{REP}}(\sigma)}{C_{\mathrm{OPT}}(\sigma)} \leq \frac{T}{C_{\mathrm{OPT}}(\sigma)} + \frac{2d(L, R)}{C_{\mathrm{OPT}}(\sigma)} \leq 3 \ .$$

□

Our second online algorithm for DOLTSP is called Zig-Zag (`ZZG`), and it is defined only on paths. The algorithm repeatedly moves the server to the left and to the right while there are pending requests in each direction. In HDOLTSP, while there are no pending requests, `ZZG` moves the server towards the origin.

The following results show that `ZZG` can achieve the same competitive ratios we obtained for `REP`. The proofs are very similar to those of Theorem 2 and Theorem 3.

Theorem 4. *Algorithm* `ZZG` *is* 2*-competitive for HDOLTSP on any path against both fair and standard adversaries.*

Theorem 5. *Algorithm* `ZZG` *is* 3*-competitive for NDOLTSP on any path against both fair and standard adversaries.*

Our last online algorithm for DOLTSP is called Delayed Replan (`DREP`), and it is very similar to `REP`. The only difference is that when a new request is presented, `DREP` delays the computation of a new optimal tour until the server is at the origin or it has just served a request. This implies that at any given moment the server of `DREP` is on a shortest path from x to y, with x and y being the origin or vertices where requests have been presented. According to the following results, `DREP` is 3-competitive on any graph, in all versions of DOLTSP.

Theorem 6. *Algorithm* DREP *is* 3*-competitive for HDOLTSP on any graph against both fair and standard adversaries.*

Proof. Let σ be any sequence of requests, and let T be the moment in which the last request is presented. WLOG, assume that at time T the server of DREP is on a shortest path from x to y, with x and y being the origin or vertices where requests have been presented. Once the server reaches y, it will follow an optimal tour that serves all the pending requests ending at the origin. Since the server can always go from y to the origin and then follow an optimal offline tour for σ, the cost of the last tour of DREP is at most $\bar{y} + C_{\mathrm{OPT}}(\sigma)$. Besides, the server of any algorithm must visit x and y, and return to the origin, which implies $\bar{y} + d(x, y) \leq C_{\mathrm{OPT}}(\sigma)$. Putting all the above things together we obtain

$$\frac{C_{\mathrm{DREP}}(\sigma)}{C_{\mathrm{OPT}}(\sigma)} \leq \frac{T + d(x, y) + \bar{y} + C_{\mathrm{OPT}}(\sigma)}{C_{\mathrm{OPT}}(\sigma)} \leq 3 \ .$$

□

Theorem 7. *Algorithm* DREP *is* 3*-competitive for NDOLTSP on any graph against both fair and standard adversaries.*

Proof. Let σ, T, x and y be as in the previous proof. Let a be the ending vertex of an optimal offline tour for σ. Once the server reaches y, it will follow an optimal tour that serves all the pending requests. Since the server can always go from y to either the origin or a, and then follow an optimal offline tour for σ, the cost of the last tour of DREP is at most $\min(\bar{y}, d(y, a)) + C_{\mathrm{OPT}}(\sigma)$. Finally, the server of any algorithm must visit x and y, being $d(x, y) + d(y, a) \leq C_{\mathrm{OPT}}(\sigma)$ if OPT goes first to x, and $\bar{y} + d(y, x) \leq C_{\mathrm{OPT}}(\sigma)$ otherwise. Therefore we have

$$\frac{C_{\mathrm{DREP}}(\sigma)}{C_{\mathrm{OPT}}(\sigma)} \leq \frac{T + d(x, y) + [\min(\bar{y}, d(y, a)) + C_{\mathrm{OPT}}(\sigma)]}{C_{\mathrm{OPT}}(\sigma)} \leq 3 \ .$$

□

Algorithms REP, ZZG and DREP are members of a very natural class of algorithms, namely *zealous* algorithms. The idea behind them is simple: whenever there is pending work, do it without wasting time. Zealous algorithms were introduced in [10] for (continuous) OLTSP. As mentioned in that work, a formal definition of zealous algorithms requires some care. This is particularly true for DOLTSP because the server can change direction only at the vertices.

Definition 8. *An online algorithm for DOLTSP is called* zealous *if and only if each time the server is at any vertex, the following conditions are met.*

1. *If there are pending requests, the server moves to serve one of them or to the origin using a shortest path.*
2. *In HDOLTSP, if there are no pending requests, the server moves to the origin using a shortest path.*

3. *If the server arrived to the vertex while traveling to another one, it can change the planned tour only if a new request has been presented after leaving the previous vertex in the route.*

Now we have a precise description of zealous online algorithms for DOLTSP, we are able to prove lower bounds on their competitiveness. Our lower bounds are valid on any graph against both fair and standard adversaries.

Theorem 9. *No zealous online algorithm for HDOLTSP on any graph is better than 2-competitive against neither fair nor standard adversaries.*

Proof. Let x be the closest vertex to the origin. At time 0 request r_1 is presented at vertex x. Since the online algorithm is zealous, it starts moving the server to x immediately. Once r_1 is served at time $\bar{x}$, there are no pending requests, so the server starts moving to the origin. At this moment request r_2 is presented again at vertex x, but the server must arrive to the origin before it can return to x. Once r_2 is satisfied, the server goes again to the origin, with a total cost of at least $4\bar{x}$. The adversary can serve both requests at time $\bar{x}$ moving its server only once to x, and then returning it to the origin, with a total cost of at most $2\bar{x}$, which proves the claim. □

Theorem 10. *No zealous online algorithm for NDOLTSP on any graph is better than 3-competitive against neither fair nor standard adversaries.*

Proof. Let x be the closest vertex to the origin. At time 0 request r_1 is presented at vertex x, and the zealous algorithm starts moving its server to x immediately. At this moment request r_2 is presented at the origin, but the server must complete its movement to x (at time $\bar{x}$), and then it starts returning to the origin. At this moment request r_3 is presented at vertex x, but the server must reach the origin (at time $2\bar{x}$), and then it goes again to x, with a total cost of $3\bar{x}$. The adversary can end its job at time $\bar{x}$, serving r_2 at time 0 at the origin, and the other requests at time $\bar{x}$ at vertex x, which completes the proof. □

Note that in most of the cases our zealous algorithms `REP`, `ZZG` and `DREP` achieve competitive ratios coincident with the lower bounds we have just presented. More precisely, `REP` and `ZZG` are optimal zealous algorithms for DOLTSP on paths, while `DREP` is optimal for the Nomadic problem on any graph. This implies that in general the competitiveness achievable by zealous algorithms for DOLTSP does not depend on the type of adversary: the lower bounds of Theorem 9 and Theorem 10 are the same against both fair and standard adversaries, and those lower bounds are achieved in most of the cases.

Another interesting observation is that HDOLTSP is easier than NDOLTSP, at least on paths, since optimal zealous algorithms for HDOLTSP on paths are 2-competitive, while for NDOLTSP we have 3-competitive optimal algorithms. This is not surprising, if we consider that online algorithms for HDOLTSP have an extra bit of information: the server must always end at the origin.

In [8] it was proved that a zealous algorithm called Plan At Home is 3-competitive for (continuous) Homing OL-ATSP. Since OL-ATSP can be viewed

as a generalization of DOLTSP, that result can be applied to our problem, achieving the same upper bound as Theorem 6.

4 General Lower Bounds

Lower bounds shown in Sect. 3 use the fact that the online algorithms are zealous. In this section we remove this restriction and present lower bounds valid for any online algorithm for DOLTSP. Some of the lower bounds hold on any graph, while others need a special distribution of the vertices.

We will start with HDOLTSP (Theorem 11 and Theorem 12), and then we will consider NDOLTSP (Theorem 13 and Theorem 14).

Theorem 11. *No online algorithm for HDOLTSP on any graph is better than ρ-competitive against a fair adversary, with $\rho = \frac{1+\sqrt{5}}{2} \approx 1.618$.*

Proof. Let x be the closest vertex to the origin. At time 0 request r_1 is presented at vertex x. The online server must visit x and return to the origin at a certain moment, because we are in the Homing problem. Let $\tau \geq \bar{x}$ be the moment in which the online server starts moving to the origin after serving r_1. If $\tau + \bar{x} \geq 2\bar{x}\rho$, no more requests are presented. In this case the online server arrives to the origin not before time $\tau + \bar{x}$, while the adversary can move its server to x and return it to the origin at time $2\bar{x}$, so we have

$$\frac{C_{\mathrm{ALG}}(\sigma)}{C_{\mathrm{OPT}}(\sigma)} \geq \frac{\tau + \bar{x}}{2\bar{x}} \geq \frac{2\bar{x}\rho}{2\bar{x}} = \rho \ .$$

On the other hand, if $\tau + \bar{x} < 2\bar{x}\rho$, a new request r_2 is presented at time τ at vertex x. In this situation the adversary can serve both requests at time τ, with a total cost of at most $\tau + \bar{x}$. The online cost is at least $\tau + 3\bar{x}$, because when the online server reaches the origin after serving r_1, it must visit again x and return again to the origin. Therefore we have

$$\frac{C_{\mathrm{ALG}}(\sigma)}{C_{\mathrm{OPT}}(\sigma)} \geq \frac{\tau + 3\bar{x}}{\tau + \bar{x}} = 1 + \frac{2\bar{x}}{\tau + \bar{x}} > 1 + \frac{2\bar{x}}{2\bar{x}\rho} = 1 + \frac{1}{\rho} = \rho \ .$$

□

Theorem 12. *There exists a family of halfpaths where no online algorithm for HDOLTSP is better than ρ-competitive against a standard adversary, with $\rho = \frac{2+\sqrt{2}}{2} \approx 1.707$.*

Theorem 13. *No online algorithm for NDOLTSP on any graph is better than ρ-competitive against a fair adversary, with $\rho = \frac{1+\sqrt[3]{19-3\sqrt{33}}+\sqrt[3]{19+3\sqrt{33}}}{3} \approx 1.839$.*

Proof. Let x be the closest vertex to the origin. At time 0 request r_1 is presented at vertex x. Let $\tau_1 \geq 0$ be the moment in which the online server leaves the origin. If $\tau_1 + \bar{x} \geq \rho\bar{x}$, the sequence of requests ends. The online server arrives to x not

before time $\tau_1 + \bar{x}$, while the adversary can reach the vertex at time $\bar{x}$, and then we have

$$\frac{C_{\mathrm{ALG}}(\sigma)}{C_{\mathrm{OPT}}(\sigma)} \geq \frac{\tau_1 + \bar{x}}{\bar{x}} \geq \frac{\rho\bar{x}}{\bar{x}} = \rho \ .$$

On the contrary, if $\tau_1 + \bar{x} < \rho\bar{x}$, request r_2 is presented at time τ_1 at the origin. Let $\tau_2 \geq \tau_1 + \bar{x}$ be the moment in which the online server starts moving to the origin for serving r_2. If $\tau_2 + \bar{x} \geq \rho(\tau_1 + \bar{x})$, no more requests are presented. In this case the cost of the online algorithm is at least $\tau_2 + \bar{x}$, because its server must arrive to the origin. Since the adversary can wait at the origin until time τ_1 for serving r_2, and then move to x for serving r_1, we obtain

$$\frac{C_{\mathrm{ALG}}(\sigma)}{C_{\mathrm{OPT}}(\sigma)} \geq \frac{\tau_2 + \bar{x}}{\tau_1 + \bar{x}} \geq \frac{\rho(\tau_1 + \bar{x})}{\tau_1 + \bar{x}} = \rho \ .$$

Finally, if $\tau_1 + \bar{x} < \rho\bar{x}$ and $\tau_2 + \bar{x} < \rho(\tau_1 + \bar{x})$, request r_3 is presented at time τ_2 at vertex x. In this situation the online cost is at least $\tau_2 + 2\bar{x}$, because the server must visit x after it arrives to the origin. Once again the adversary can wait at the origin until time τ_1 for serving r_2, and then move to x for serving r_1, ending its job at time τ_2 when r_3 is presented. Therefore we have

$$\frac{C_{\mathrm{ALG}}(\sigma)}{C_{\mathrm{OPT}}(\sigma)} \geq \frac{\tau_2 + 2\bar{x}}{\tau_2} > 1 + \frac{2\bar{x}}{\rho(\tau_1 + \bar{x}) - \bar{x}} > 1 + \frac{2\bar{x}}{\rho^2\bar{x} - \bar{x}} = 1 + \frac{2}{\rho^2 - 1} = \rho \ .$$

□

Theorem 14. *There exists a family of paths where no online algorithm for NDOLTSP is better than 2-competitive against a standard adversary.*

Proof. Consider a path with at least two vertices $x > 0$ and $-x$. WLOG, assume that at time $\bar{x}$ the online server is in the left halfpath. At that moment, a single request at vertex x is presented. The online cost is at least $2\bar{x}$, while the adversary can serve the request at time $\bar{x}$. □

Notice that the last lower bound is valid on a certain group of paths that are not halfpaths. It is essentially the same result presented in [9] for (continuous) OLTSP. In the full version of this paper we prove that the same lower bound holds on a particular group of halfpaths.

Online Dial-a-Ride Problem (OLDARP) generalizes (continuous) OLTSP to the case in which requests are pairs of points and a server must take an object from the first point to the second point. It is interesting to note that the lower bounds of this section are very similar to the corresponding lower bounds known for OLDARP on the real line. For instance, the lower bounds of Theorem 11 and Theorem 12 are coincident with the lower bounds given in [11] and [7] for Homing OLDARP on the real halfline against fair and standard adversaries, respectively. While Theorem 11 uses the same idea presented in [11], we derived Theorem 12 in a completely different way. The relation between our lower bounds and those for OLDARP on the real line must be studied further. However, a possible explanation for this phenomenon could be that, even though OLDARP

is defined on a continuous metric space, once the server picks up an object it cannot satisfy other requests until the object is delivered. A similar situation occurs in DOLTSP, where the server cannot change direction until it arrives to the next vertex. A summary of results for OLDARP can be found in [11].

5 Cautious Algorithms

In Sect. 3 we saw that `REP` and `ZZG` achieve the best competitive ratio for zealous online algorithms. However, those ratios are notably higher than the general lower bounds shown in Sect. 4. It would be nice to have online algorithms with competitiveness closer to these general lower bounds. In order to succeed, we must consider a distinct class of algorithms.

An online *cautious* algorithm may wait without moving its server even when there is pending work. New requests presented while the algorithm is waiting (and even the absence of them), give additional information that the algorithm can use to improve its performance. On the contrary, a zealous algorithm faced with the same sequence of requests would take an early decision that could be inappropriate a posteriori.

A key point in the design of a cautious online algorithm is to decide how much time the server should wait when there is pending work. A longer waiting increases the possibilities to obtain additional information. Nonetheless, the caution must not be against the main goal of the algorithm, which is to minimize the total time to complete its job.

Our cautious online algorithms aim at obtaining competitive ratios coincident with the general lower bounds of Sect. 4. Each time a new request arrives the algorithms compute how long the adversary needs to serve all the known requests. Then, cautious algorithms wait just till the moment in which extending the waiting time would prevent obtaining the desired competitiveness. A number of online algorithms that wait taking into account the cost of the adversary were considered for continuous problems related to DOLTSP, such as OLTSP [10, 11], OLDARP [7] and OL-ATSP [8].

We devised two cautious online algorithms using the general scheme described above. The main difference between them is when they decide to wait. Our first cautious algorithm is called Wait-Before-Return (`WBR`). The algorithm is only defined for HDOLTSP on halfpaths. It serves as soon as possible pending requests away from the origin. The remaining requests are satisfied when the server returns to the origin. Before doing so, `WBR` waits as explained above.

The other cautious algorithm is Wait-Before-Begin (`WBB`). It is possible to use this algorithm for both HDOLTSP and NDOLTSP, on any path (though we only prove results for NDOLTSP on halfpaths). Each time the server is halted and a new request is presented, `WBB` computes an optimal route that serves all the pending requests. Before starting its tour, the algorithm waits according to the general scheme explained above.

A more detailed description of WBR and WBB can be found in the full version of this paper. The following results establish the competitiveness of the algorithms in different variants of DOLTSP.

Theorem 15. *Algorithm WBR is ρ-competitive for HDOLTSP on any halfpath, with $\rho = \frac{1+\sqrt{5}}{2} \approx 1.618$ against a fair adversary, and $\rho = \frac{2+\sqrt{2}}{2} \approx 1.707$ against a standard adversary.*

Theorem 16. *Algorithm WBB is 2-competitive for NDOLTSP on any halfpath against both fair and standard adversaries.*

Notice that the above upper bounds match the general lower bounds of Sect. 4, with the exception of WBB for NDOLTSP against a fair adversary. If we compare these lower and upper bounds, against those of Sect. 3 for zealous algorithms, we observe that zealous algorithms for DOLTSP are weaker than cautious algorithms.

Recall that in Sect. 3 we showed that for zealous online algorithms, HDOLTSP is easier than NDOLTSP, and there is no difference between the two types of adversaries. If we review the results of this section and those of Sect. 4, we can observe that for cautious online algorithms once again HDOLTSP is easier than NDOLTSP. Besides, for cautious algorithms we have better performances against a fair adversary than against a standard adversary. That is, for cautious algorithms the fair adversary is weaker than the standard one. This is not surprising taking into account that the fair adversary has a restricted power.

As for general graphs, Theorem 3.2 in [8] states that a cautious algorithm called SmartStart is $\frac{3+\sqrt{5}}{2}$-competitive for (continuous) Homing OL-ATSP. The algorithm is a variation of an algorithm presented in [7]. In the same spirit as our algorithm WBB, SmartStart waits at the origin until the moment in which waiting more would prevent him from being competitive. At that moment it starts an optimal tour that serves all the pending requests and returns to the origin. As we said before, OL-ATSP can be viewed as a generalization of DOLTSP, so we can derive the following result.

Theorem 17. *Algorithm SmartStart for Homing OL-ATSP [8] determines a ρ-competitive algorithm for HDOLTSP on any graph against both fair and standard adversaries, with $\rho = \frac{3+\sqrt{5}}{2} \approx 2.618$.*

The above upper bound is far from the lower bounds of Theorem 11 and Theorem 12 (Section 4). It is not clear for now whether the lower bounds can be improved, or a better cautious algorithm can be found. In trying to improve the lower bounds, requests outside a halfpath must be considered, since Theorem 15 states that WBR achieves a performance coincident with those lower bounds for HDOLTSP on halfpaths.

6 Empirical Analysis of Online Algorithms

We designed a set of tests in order to get empirical evidence about how online algorithms work in practice on paths. In our tests, we included all our competitive

algorithms, as well as other algorithms with some intuitive improvements. One of these new algorithms is Statistic–Replan (`STR`), a subtle enhancement to regular `REP`. The only difference occurs in NDOLTSP when the server is idle at some vertex. Instead of waiting passively for a new request, `STR` moves the server to the vertex that is the most likely to receive the next request, assuming the existence of a pattern. Another algorithm we considered is Statistic–Wait-Before-Begin (`STW`), which ends the waiting time when it is likely enough that no more requests will be presented.

All the algorithms were tested using a large collection of instances designed to cover a representative set of cases. This collection takes into account different aspects of an instance: quantity and distribution of vertices, quantity and distribution (over time and over space) of requests, etc. A set of 7506 distinct scenarios was considered by combining different values for each aspect. A detailed description of the test set is given in the full version of this paper.

We executed all strategies over all instances on our test set. In each case we computed the *approximate ratio*, i.e. the ratio between the cost of the online solution and the optimal offline cost. Note that the approximate ratio of any strategy is upper bounded by its competitive ratio.

Figure 1 shows the average approximate ratio of all algorithms (as a bar), and the distribution of values among the different trials (as a telescopic-plot). One thing we can conclude is that, as suggested by the theoretical part of our study, HDOLTSP is easier than NDOLTSP. Besides, more than 50% of the times, the simplest algorithms generated quasi optimal solutions. Opposed to this, most approximate ratios of cautious strategies (`WBR` and `WBB`) exceed the threshold of 1.5. However, as we could expect, the worst case ratios of the simplest algorithms are much higher than those of the more sophisticated (competitive) ones. The cautious algorithm with best results is `STW`, although it was better than all zealous strategies only in 5% of the cases. The other cautious strategies (`WBR` and `WBB`) were surpassed by a zealous algorithm in 90% of the cases. With this evidence, we can conclude that waiting was not so profitable in practice. Finally, our empirical study was useful to analyze how the different aspects of the instances affect their complexity. We found that the distribution of requests over time is the most influential aspect: it does not matter where vertices or requests are located; the later the requests become visible, the worse are the results that the online algorithm gets.

References

1. de Paepe, W.E.: Complexity Results and Competitive Analysis for Vehicle Routing Problems. PhD thesis, Technische Universiteit Eindhoven (2002)
2. Deif, I., Bodin, L.: Extension of the Clarke and Wright algorithm for solving the vehicle routing problem with backhauling. In: Babson Conference on Software Uses in Transportation and Logistics Management, Babson Park, MA (1984)
3. Savelsbergh, M.W.P.: Local search in routing problems with time windows. Annals of Operations Research **4**(1-4) (1985) 285–305

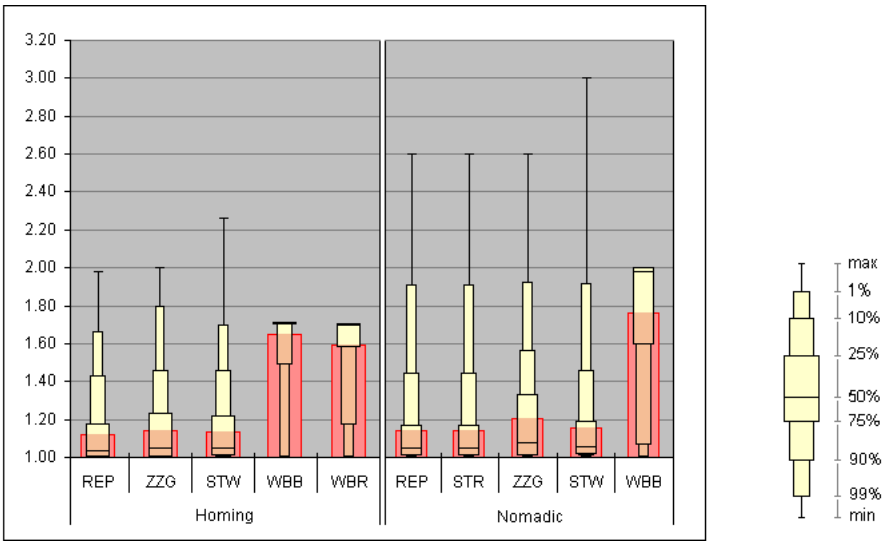

Fig. 1. Mean (*bar*) and distribution (*telescopic-plot*) of approximate ratios

4. Stein, D.: An asymptotic, probabilistic analysis of a routing problem. Mathematics of Operations Research **3**(2) (1978) 89–101
5. Borodin, A., El-Yaniv, R.: Online Computation and Competitive Analysis. Cambridge University Press (1998)
6. Fiat, A., Woeginger, G.J., eds.: Online Algorithms: The State of the Art. Volume 1442 of LNCS. Springer-Verlag (1998)
7. Ascheuer, N., Krumke, S.O., Rambau, J.: Online dial-a-ride problems: Minimizing the completion time. In: Proc. 17th International Symposium on Theoretical Aspects of Computer Science (STACS 2000). Volume 1770 of LNCS., Springer (2000) 639–650
8. Ausiello, G., Bonifaci, V., Laura, L.: The on-line asymmetric traveling salesman problem. Journal of Discrete Algorithms **6**(2) (2008) 290–298
9. Ausiello, G., Feuerstein, E., Leonardi, S., Stougie, L., Talamo, M.: Algorithms for the on-line travelling salesman. Algorithmica **29**(4) (2001) 560–581
10. Blom, M., Krumke, S.O., de Paepe, W., Stougie, L.: The online-TSP against fair adversaries. INFORMS Journal on Computing **13**(2) (2001) 138–148
11. Lipmann, M.: On-line Routing. PhD thesis, Technische Universiteit Eindhoven (2003)
12. Psaraftis, H.N., Solomon, M.M., Magnanti, T.L., Kim, T.U.: Routing and scheduling on a shoreline with release times. Management Science **36**(2) (1990) 212–223
13. Sleator, D.D., Tarjan, R.E.: Amortized efficiency of list update and paging rules. Communications of ACM **28** (1985) 202–208
14. Karlin, A.R., Manasse, M.S., Rudolph, L., Sleator, D.D.: Competitive snoopy caching. Algorithmica **3** (1988) 79–119
15. Koutsoupias, E., Papadimitriou, C.H.: Beyond competitive analysis. SIAM Journal on Computing **30**(1) (February 2000) 300–317