Algorithms for the On-line Travelling Salesman *

Giorgio Ausiello[†] Esteban Feuerstein[‡] Stefano Leonardi[†]
Leen Stougie[§] Maurizio Talamo[†]

July 13, 1999

Abstract

In this paper the problem of efficiently serving a sequence of requests presented in an on-line fashion located at points of a metric space is considered. We call this problem the On-Line Travelling Salesman Problem (OLTSP). It has a variety of relevant applications in logistics and robotics.

We consider two versions of the problem. In the first one the server is not required to return to the departure point after all presented requests have been served. For this problem we derive a lower bound on the competitive ratio of 2 on the real line. Besides, a 2.5-competitive algorithm for a wide class of metric spaces, and a 7/3-competitive algorithm for the real line are provided.

For the other version of the problem, in which returning to the departure point is required, we present an optimal 2-competitive algorithm for the above mentioned general class of metric spaces. If in this case the metric space is the real line we present a 1.75-competitive algorithm that compares with a ≈ 1.64 lower bound.

Keywords: on-line algorithms, competitive analysis, travelling salesman problem, vehicle routing.

---

*Part of the results of this paper have already appeared in the Proceedings of the 4th Scandinavian Workshop on Algorithm Theory, LNCS 824, Springer-Verlag, 1994, and in the Proceedings of the 4th International Workshop on Algorithms and Data Structures, LNCS 955, Springer-Verlag, 1995.

[†]Dipartimento di Informatica e Sistemistica, Università di Roma "La Sapienza", via Salaria 113, 00198-Roma, Italia. This work was partly supported by EU ESPRIT Long Term Research Project ALCOM-IT under contract No.20244, and by the Italian Ministry of Scientific Research, Project 40% "Efficienza di Algoritmi e Progetto di Strutture Informative". e-mail: {ausiello,leon,talamo}@dis.uniroma1.it

[‡]Departamento de Computación, Facultad de Ciencias Exactas y Naturales, Universidad de Buenos Aires and Instituto de Ciencias, Universidad de General Sarmiento, Argentina. Partly supported by the KIT program of the European Community under contract n. 131 (DYNDATA) and by UBA's "Programación para Investigadores Jóvenes" project "Algoritmos eficientes para problemas on-line con aplicaciones". e-mail: esteban@dc.uba.ar

[§]Department of Mathematics, Eindhoven University of Technology, PO Box 513, 5600MB Eindhoven, The Netherlands. Supported by the Human Capital Mobility Network DONET of the European Community. e-mail: leen@win.tue.nl

1 Introduction

The Travelling Salesman Problem (TSP) and in general vehicle routing and scheduling problems have been widely studied for more than three decades (see [14] for a survey on the subject). The input of an instance of the problem is generally a set of locations (points) in a metric space that are to be visited in such a way that the total distance travelled or the completion time is minimized. A common characteristic of almost all the approaches to the study of the problem is the off-line point of view. The input is known completely beforehand.

However, in many routing and scheduling applications the instance becomes only known in an on-line fashion. In other words, the input of the problem is communicated in successive steps. Often it is even not possible to determine which is the last request, i.e. when the instance is completely known. Anyhow, if the goal is to minimize the completion time, waiting till all the information is available could imply a costly loss of time.

In this paper we consider a class of on-line variations of TSP in a metric space: while the salesman is travelling, new sites to visit may be communicated to him. His goal is to visit all the sites, minimizing the completion time.

This setting models many natural applications. Think for example of a salesman or a repairman with a cellular phone, or of a robot that has to serve locations of its working space (for example in the Euclidean plane) and of many other routing and scheduling problems on a transportation network modeled with a graph. We will refer to this problem as the *On-line Travelling Salesman Problem* (OLTSP).

As the input to the salesman – from now on we will refer to him as the *server* – is communicated in an on-line way, the scheduled route will have to be updated also in an on-line way during the trip. The fact that the schedule must be constructed based on incomplete information means that in general no algorithm (polynomial or otherwise) can be guaranteed to construct an optimal schedule on-line.

The most widely accepted way of measuring the performance of on-line algorithms is *competitive analysis*. The quality of a certain on-line strategy is measured by the worst-case ratio between the time needed by the on-line algorithm for a sequence of requests and the optimal time needed by an algorithm that knows the sequence in advance. This ratio is called the *competitive ratio* of the on-line algorithm. Therefore, an algorithm is said to be ρ-competitive if for every input its completion time is at most ρ times the optimal completion time for the same input. The concept of competitive analysis has been

formalized in [18], although under the name of worst-case analysis of on-line algorithms it dates back at least until the work of Graham [10] and Johnson [11]. The performance of on-line strategies for a great variety of on-line problems has been analyzed according to this concept: performance analysis of computer systems, data structures, scheduling, motion planning, network management, financial decision making, etc. (for an overview of the subject we refer to [3]).

We present algorithms for the OLTSP and study their competitive ratio by comparing their performance to the optimal solution of the corresponding off-line problem, which is called the Vehicle Routing Problem with release times [16].

In that problem, each site must be visited at or after a given release time. The release time of a request corresponds to the time in which the request is communicated to the on-line server. The problem is NP-hard since it contains the Hamiltonian Path problem as a particular case.

Several off-line variations of the problem have been studied, in which additional constraints are imposed and particular metric spaces are considered. In [16] it has been shown that if the metric space is a line the optimal solution may be found in quadratic time. In [12] the metric space is restricted to be a tree and each request has, besides a release time, an associated *handling time* that is the time needed to serve it. The problem is shown to be NP-hard in that context, and a 2-approximate solution is given. In general these problems are called *routing and scheduling with time window constraints*. Sometimes more than one server is considered, and other restrictions are given by requiring that requests must be served before a specified *deadline* (see for example [20, 21]). Other related works are [2, 6, 7, 8].

A related on-line work [13] considers the problem of visiting the whole set of vertices of an unknown graph, when the set of edges leaving a node is revealed only once the node is visited. In our case, the metric space is completely known from the beginning, but what is revealed in an on-line way is the set of locations that must be visited.

It is important to note that OLTSP is different from the famous *k-server problem* [15]. In that problem the requests have to be served in the order in which they are presented, with the goal of minimizing the total distance travelled by the k servers. On the contrary, in OLTSP the task is precisely to decide the order in which the requests will be served.

A recent paper [1] considers a *fixed* number of clients presenting sequences of requests in a metric space, that must be served by a single server. At any time, each client has at most one request to be served, after which a new one may be presented. The main

difference with our approach is that, in this case the request sequence is dependent on the behaviour of the algorithm.

We consider two versions of the basic problem that requires that all points presented are visited. In the first version, that we call *Nomadic*-OLTSP (or simply N-OLTSP) this is the only requirement. Adding the constraint that the trip must end at its departure point defines the problem that we call *Homing*-OLTSP (or simply H-OLTSP).

Addition of the constraint of ending the trip at the departure point changes the nature of the problem (and hence the kind of applications). Lower bounds and algorithms for the two versions are quantitatively and qualitatively different. In fact, knowing that the server has to return to the departure point provides additional information to the on-line algorithm, that allows it to achieve a better competitive ratio.

In this work we propose on-line deterministic algorithms for both N-OLTSP and H-OLTSP, and show that they are ρ-competitive for suitable constants ρ. We establish such results for the problems defined on a wide class of metric spaces that we call the class $\mathcal{M}$, whose precise definition can be found at the beginning of the following section. In the particular case in which the metric space is the *real line* different algorithms are devised and stronger ratios of competitiveness are derived.

For N-OLTSP, no on-line algorithm can be better than 2-competitive, even for the line. For metric spaces belonging to the class $\mathcal{M}$ we propose a 2.5-competitive algorithm; for the line the best proposed algorithm has competitive ratio 7/3.

For H-OLTSP we propose a *best possible* 2-competitive algorithm for metric spaces belonging to the class $\mathcal{M}$, while for the line we devise a 1.75-competitive algorithm that compares with a ≈ 1.64 lower bound.

Our best algorithms for metric spaces belonging to $\mathcal{M}$ do not run in polynomial time unless P=NP, since they use subroutines for optimally solving the TSP. However, one can obtain almost as good performance from polynomial-time algorithms: we show how to obtain 3-competitive polynomial-time algorithms for both N-OLTSP and H-OLTSP. As we mentioned before, the on-line nature of the problem is a source of difficulty independent of its computational complexity, and therefore on-line algorithms achieving good competitive ratios are of interest also if their time requirements are not polynomially bounded.

The paper is organized as follows. In Section 2 we formally define the model. In Section 3 we present our lower bounds for the different versions of the problem. Section 4 contains our best algorithms for metric spaces belonging to $\mathcal{M}$, while Section 5 deals with polynomial time algorithms. Section 6 proposes algorithms for the real line. To

facilitate the exposition, every section first describes results on N-OLTSP and afterwards for H-OLTSP. Finally Section 7 contains open problems and interesting related problems for future research.

2 The model

The input of OLTSP consists of a metric space M, from the class $\mathcal{M}$ defined below, a distinguished point o (the origin) of M, and a sequence of pairs $< t_i, p_i >$ where p_i is a point of M and t_i is a number representing the moment in which the request is presented. The t_i's form an ordered sequence in the sense that $0 \leq t_i \leq t_j$ if $i < j$.

A server is located at the origin o of the metric space at time 0, and moves not faster than unit speed.

We use the definition of metric space as a space M with the following properties: (1) It is symmetric, i.e., for every pair of points x, y in M, $d(x, y) = d(y, x)$, where $d(x, y)$ denotes the distance from x to y; (2) $d(x, x) = 0$ for any point x in M; (3) It satisfies the triangle inequality, i.e., for any triple of points x, y, and z in M, $d(x, y) \leq d(x, z) + d(z, y)$.

Our class of metric spaces $\mathcal{M}$ contains all continuous metric spaces, i.e., every metric space M having the property that the shortest path from $x \in M$ to $y \in M$ is continuous, formed by points in M, and has length $d(x, y)$. For continuous metric spaces the times at which a request can be made can be any non-negative real number.

Next $\mathcal{M}$ contains discrete metric spaces representable by an underlying graph with all edges having unit length. The vertices are the points of the metric space. Working on such spaces time needs to be discretized, i.e., the times t_i at which requests are made are non-negative integers, and the server determines its strategy at integer points in time. At each integer time, the server is at some point in the metric space (vertex in the graph) and either remains there or moves in one time step to a neighboring point in the metric space.

Thus, an example of a model that we do not consider here is one in which the server moves on a road network of freeways and a request can arrive while he is moving between two exits and he has to proceed to the next exit before being able to change his strategy. In our model the server would be allowed to do a U-turn and return to the previous exit.

For any path T in M, let $|T|$ denote its length. Note that if T is a path from x to y, we must have $|T| \geq d(x, y)$ by the triangle inequality.

As mentioned in the Introduction we consider two versions of the problem:

The *Nomadic On-line Travelling Salesman Problem - (N-OLTSP)*, defined as minimizing the completion time required to serve all presented requests;

The *Homing On-line Travelling Salesman Problem - (H-OLTSP)*, defined as minimizing the completion time required to serve all presented requests and return to the origin o.

On-line algorithms for the problems N-OLTSP and H-OLTSP determine the behavior of the server at a certain moment t as a function of all requests $< t_i, p_i >$ such that $t_i \leq t$.

We will denote the completion time of the solution produced by an on-line algorithm OL by Z^{OL} and that of the optimal (off-line) solution by Z^*. An on-line algorithm for OLTSP is ρ-competitive if for any sequence of requests $Z^{OL} \leq \rho Z^*$. Let $p^{OL}(t)$ and $p^*(t)$ respectively denote the positions of OL's server and the optimal off-line server at time t. At time 0 the server is located at the origin o, $p^{OL}(0) = p^*(0) = o$.

3 Lower bounds

In this section we derive lower bounds on the competitive ratio of any on-line strategy for serving the requests in the versions of the problem.

3.1 A lower bound for N-OLTSP

We show that no on-line algorithm can achieve a competitive ratio smaller than 2 for N-OLTSP. With this aim, we provide a sequence of requests for which no algorithm can finish within less than twice the optimal off-line time.

Theorem 3.1 *Any ρ-competitive algorithm for N-OLTSP has $\rho \geq 2$. The lower bound is achieved on the real line.*

Proof: The proof is derived from the following simple argument. Consider the problem on the real line with the abscissa 0 as the origin. An adversary gives a request at time 1 in either 1 or -1, depending on whether at time 1 the on-line server is in a negative or a positive position, respectively. Thus, the adversary has completed at time 1, whereas the on-line server needs at least 2, with 2 sufficing when it is at 0 at time 1. □

We observe that the former proof can be easily adapted to show that the same lower bound holds for randomized algorithms against an oblivious adversary. The same simple sequence can be used replacing the "position" of the on-line server by "expected position". For the definition of oblivious adversary we refer to [4].

3.2 A lower bound for H-OLTSP

In this section we show a lower bound on the competitive ratio of any algorithm for H-OLTSP on metric spaces belonging to $\mathcal{M}$.

Theorem 3.2 *For any $\epsilon > 0$, any ρ-competitive algorithm for H-OLTSP for metric spaces belonging to $\mathcal{M}$ has $\rho \geq 2 - \epsilon$.*

Proof: Take as the metric space the boundary of the unit square $[0,1]^2$. We denote by (x, y) a point of abscissa x and ordinate y. Given two points of the square (x_1, y_1) and (x_2, y_2), we denote by $[(x_1, y_1), (x_2, y_2)]$ the segment that is obtained by traversing the square in clockwise direction from (x_1, y_1) to (x_2, y_2). The distance between two points is defined as the length of the shorter of the two segments of the boundary of the unit square between the two points. As the origin we take the point $(0, 0)$. At time 0, for a fixed $n \geq 1$, requests are given at points of the square $\{(0, i/n), i = 0, \ldots, n\} \cup \{(1, i/n), i = 0, \ldots, n\} \cup \{(i/n, 0), i = 1, \ldots, n-1\} \cup \{(i/n, 1), i = 1, \ldots, n-1\}$. Thus $(0, 0)$, $(1, 0)$, $(0, 1)$ and $(1, 1)$ belong to this set of points. Notice that these requests can be served optimally in time $Z^* = 4$.

We first show that for some δ, with $0 \leq \delta \leq 2$, at time $2 + \delta$ any on-line server must be in one of the two points at distance $2 - \delta$ from the origin (not necessarily requested points). For this purpose define the function $f : [0, 2] \to [0, 2]$ as the distance from the point $(1, 1)$ at time $2 + x$. Then the function $g(x) = f(x) - x$ has the property that $g(0) \geq 0$ and $g(2) \leq 0$. Since g is continuous there must be at least one point δ with $0 \leq \delta \leq 2$ with $g(\delta) = 0$.

Take the smallest value of δ for which this holds. Without loss of generality we assume that this point $p^{OL}(2+\delta)$ is on the path between $(0, 0)$ and $(1, 1)$ that passes through $(0, 1)$. At time $2 + \delta$ the server has served the requests on the segment $T_1 = [(0, 0), p^{OL}(2 + \delta)]$. Additionally, it may have visited requests on a segment $S_1 = [(x_1, y_1), (0, 0)]$ and requests on a segment $S_2 = [p^{OL}(2 + \delta), (x_2, y_2)]$ (see Figure 1). The total length of these latter two segments is no more than δ since the server is at a distance $2 - \delta$ from the origin and must have travelled each of these segments at least twice. Thus, $|T_1| + |S_1| + |S_2| \leq 2$ This implies that the on-line algorithm has not touched any requested point of a segment $T_2 = [(x_2, y_2), (x_1, 0)]$ of length at least 2.

Now, at time $2 + \delta$, a new set of requests is given in each of the points on the segment $T_1 = [(0, 0), p^{OL}(2 + \delta)]$ of length $2 - \delta$, that were requested before and visited by the on-line server. This new set of requests ends the sequence. The optimal completion time

for the whole sequence is still $Z^* = 4$ since an anti-clockwise tour of the square visits any request not earlier than its release time. Given the situation of the on-line server at time $2 + \delta$ one of the two following options will give the best possible completion time:

1. Traverse twice the segment $[p^{OL}(2 + \delta), (x_1, y_1)]$ with the exception of a segment $[(x_1, y_1), (x_3, y_3)]$ of size at most $1/n$. The traversed segment is therefore of size at least $2 - 1/n$. After this traverse once the segment T_1. The cost of the algorithm is in this case $Z^{OL} \geq (2 + \delta) + (4 - 2/n) + (2 - \delta) = 8 - 2/n$.

2. Traverse first twice the segment $T_1[(0, 0), (0, 1/n)]$ and then once the segment $[p^{OL}(2+\delta), (0, 0)]$. The cost of the algorithm is in this case $Z^{OL} = (2+\delta) + (4 - 2\delta - 2/n) + (2 + \delta) = 8 - 2/n$.

Therefore, for any arbitrarily small $\epsilon > 0$, the ratio between the on-line server's completion time and the optimal completion time can be made $2-\epsilon$ by choosing a sufficiently large value for n.

□

We emphasize that this theorem says that *some metric spaces* in $\mathcal{M}$ can induce any algorithm for H-OLTSP to be no less than 2-competitive. Therefore, better competitive ratios may be possible for particular metric spaces, for instance for the line, as we shall see in what follows.

3.3 A lower bound for H-OLTSP on the line

In this subsection we present a lower bound on the competitive ratio of algorithms for H-OLTSP defined on the real line. We study this case separately so as to compare the lower bound with the competitive ratio of an algorithm for the problem on the real line presented in Section 6.2.

An argument similar to that used for the lower bound for N-OLTSP (Theorem 3.1) could be used to obtain a 3/2 lower bound for H-OLTSP, both for deterministic and randomized algorithms. However, a stronger lower bound for deterministic algorithms is proved below.

Theorem 3.3 *Any ρ-competitive algorithm for H-OLTSP on the real line has $\rho \geq (9 + \sqrt{17})/8 \approx 1.64$.*

Proof: Suppose OL is a ρ-competitive on-line algorithm for H-OLTSP with $\rho < (9 + \sqrt{17})/8$. An adversary could proceed as follows. Before time $t = 1$ no requests are presented. At that moment, the position $p^{OL}(1)$ of the server of the ρ-competitive on-line algorithm OL must be inside the interval $[-(2\rho - 3), (2\rho - 3)]$, and note that $2\rho - 3 < 1$ since $\rho < 2$. notice that if $p^{OL}(1) > (2\rho - 3)$ the first (and unique) request of the sequence would be at point -1, giving $Z^{OL} > 1 + (2\rho - 3) + 2 = 2\rho$, because OL has to travel from its current position to -1 and back to 0. On the other hand, for this sequence $Z^* = 2$, and therefore the algorithm would not be ρ-competitive. The case in which $p^{OL}(1) < -(2\rho - 3)$ is symmetric.

Thus, suppose that $p^{OL}(1) \in [-(2\rho - 3), (2\rho - 3)]$. Now, at time $t = 1$, the adversary presents two simultaneous requests at points -1 and 1. At time $t = 3$, the on-line server cannot have served *both* requests. Suppose, without loss of generality, that it has not served the request in -1.

We now show that if $-(7-4\rho) < p^{OL}(3) < (7-4\rho)$, then OL can not be ρ-competitive. Note that $7 - 4\rho < 1$ since $\rho < 7/4$. In this case the adversary could be in $p^*(3) = 1$, present a new request in $+1$ and return to the origin with a completion time $Z^* = 4$. OL, however, would still have to serve requests in both extremes, and hence $Z^{OL} > 3 + 1 - (7 - 4\rho) + 3 = 4\rho$, since starting at time $t = 3$ it would have to go to one of the extremes and then to the other and back to 0.

Note that since $\rho < (9+\sqrt{17})/8$, we have that the interval $[-(7-4\rho), (7-4\rho)]$ strictly contains the interval $[-(2\rho - 3), (2\rho - 3)]$.

Thus, we are left with two cases to be considered.

1. At time $t = 3$ the on-line server has not yet served $+1$, and $-1 \leq p^{OL}(3) \leq -(7-4\rho)$ or $(7 - 4\rho) \leq p^{OL}(3) \leq 1$.

2. At time $t = 3$ the on-line server has served $+1$, and $(7 - 4\rho) \leq p^{OL}(3) \leq 1$. The server cannot be to the left of $-(7-4\rho)$, since it started to move toward $+1$ after time 1 from a position not to the right of $(2\rho-3)$, and $1+(1-(2\rho-3))+(1+(2\rho-3)) = 3$.

We notice that in both cases the following situation occurs: the on-line server is within distance $1 - (7 - 4\rho)$ of the extreme on one side and has not served the extreme on the other side. This property is sufficient for the rest of the proof, where we will suppose that the on-line server is near 1 and has not served the request in -1 (the other case is symmetric).

In this case the adversary has so far served -1, is at position $p^*(3) = +1$ and finishes with $Z^* = 4$. Then, any ρ-competitive on-line algorithm has to pass point 0 no later than $4\rho - 2$. Let us denote the time at which the on-line server crosses the origin as $3 + q$. Therefore we have

$$q \leq 4\rho - 5 \tag{1}$$

At time $(3 + q)$ the adversary can be in position $(1 + q)$ and place a request at that point and return to 0. For this sequence we have that $Z^{OL} = 7 + 3q$ and $Z^* = 4 + 2q$, and therefore $\frac{Z^{OL}}{Z^*} = \frac{7+3q}{4+2q}$. By hypothesis OL is ρ-competitive, so that

$$\rho \geq \frac{7 + 3q}{4 + 2q}$$

This is a monotonously decreasing function of q, and by inequality (1) we get

$$\rho \geq \frac{7 + 3(4\rho - 5)}{4 + 2(4\rho - 5)} \tag{2}$$

The least value of ρ that satisfies inequality (2) is the value that achieves equality, that is $\rho = \frac{9+\sqrt{17}}{8}$. □

4 Algorithms for metric spaces in $\mathcal{M}$

In this section we will present competitive algorithms for metric spaces belonging to $\mathcal{M}$. The first algorithm we will analyze is based on a greedy strategy. Essentially it follows at each time the shortest route that serves all the requests with, for H-OLTSP, the additional constraint of terminating at the origin of the metric space. For H-OLTSP we will also present a more complicated algorithm that attains the best possible competitive ratio by following different rules for requests "close" to the origin and for requests "far" from the origin. The reason is that requests close to the origin can be served when the server is on the way back to the origin, the endpoint of his route. Clearly, these considerations do not hold for N-OLTSP, where the server can end his work in any position of the metric space.

The above mentioned strategies use *super polynomial time*, assuming that P≠NP, since they need to compute an optimal path or an optimal tour over a set of points. We will also present polynomial-time strategies (see Section 5) with worse competitive ratios, based on polynomial approximation algorithms to compute a path or a tour over a set of points.

4.1 An algorithm for N-OLTSP

For N-OLTSP we first analyze an algorithm based on a greedy strategy that follows a shortest Hamiltonian path on the set of requests that has not been visited yet. The route is re-computed each time a new request arrives. Clearly, each computation of a shortest Hamiltonian path may take exponential time. We will refer to this as the Greedy algorithm. The algorithm is defined for any metric space belonging to the class $\mathcal{M}$.

The algorithm is described completely by stating the action taken at any moment t, when a new request arrives. Let $\mathcal{S}$ be the set of all requests presented until t, including the new one and the origin o.

In order to simplify matters, we restrict the greedy server to move only on the shortest path between pairs of points in $\mathcal{S}$, and therefore we call the algorithm GTR, for "Greedily Travelling between Requests". Assume that at time t, when a new request is presented, the on-line server's position, $p^{GTR}(t)$, is on the shortest path between x and y in $\mathcal{S}$. Then the algorithm computes and follows the shortest route that first visits either x or y and then the yet unserved requests.

GTR achieves a competitive ratio of 5/2, as we establish in the following theorem.

Theorem 4.1 *GTR is a 5/2-competitive algorithm for N-OLTSP, and the ratio is tight.*

Proof: Let time t be the time at which the last request is presented. Let us first state two lower bounds on the optimal completion time required. First, $Z^* \geq t$ since also in the optimal solution a request cannot be served before the time at which it is presented. For the second lower bound we define $\mathcal{T}$ as the optimal Hamiltonian path on the set $\mathcal{S}$, constrained to have o as one of the 2 extreme points. Notice that $\mathcal{T}$ does not take the release times of the requests into account. Then, $Z^* \geq |\mathcal{T}|$ since any algorithm must visit all points in $\mathcal{S}$. Thus, proving $Z^{GTR} \leq t + (3/2)|\mathcal{T}|$ proves the theorem.

Let a be the endpoint of $\mathcal{T}$, the starting point is o. Observe that $p^{GTR}(t)$, the position of GTR at time t, is somewhere on the shortest path between two points of $\mathcal{S}$, say x and y. Assume that following $\mathcal{T}$ from o to a, x is visited before y. Then, $\min\{d(p^{GTR}(t), x) + d(x, o), d(p^{GTR}(t), y) + d(y, a)\} \leq (1/2)|\mathcal{T}|$. Without loss of generality assume that the first term is smaller than the second one. Consider the route that goes from $p^{GTR}(t)$ to x, then to o and finally follows $\mathcal{T}$ until a. Its length is at most $(3/2)|\mathcal{T}|$ and is also an upper bound on the length of the route followed by GTR starting at time t, and hence the on-line completion time is bounded from above by $t + (3/2)|\mathcal{T}|$ proving the theorem.

The following example shows that the ratio of 5/2 is asymptotically tight. Note that if all the requests are located on the real line, GTR boils down to going always to the nearest extreme of the smallest interval containing the requests that are yet to be visited. Ties are broken in an arbitrary way. This is not a limitation since any choice can be enforced by displacing requests a negligible distance. The example is illustrated in Figure 2.

Let 0 be the origin. Consider a sequence starting at time 1 with two requests, one in -1 and one in 1, and suppose without loss of generality that GTR first goes towards 1. At time 3 it will be back in 0 and the adversary may put a request at point 1 again. Let $t_0 = 3, p_0 = 1$ and in general $t_i = (5/3)t_{i-1} - 2/3$, and $p_i = t_i - 2$, for $i = 1, 2, 3, \ldots, n$. The sequence continues with a request in point p_i at time t_i. The adversary's completion time for this sequence will be exactly t_n, since it can start going to -1 and then going always to the right arriving at each point p_i when the request is presented. As for the completion time of GTR, we will show that at time t_n it is in the middle of the interval $[-1, p_n]$, and hence it must still travel 3/2 times the length of the interval, as it has not yet served the requests in the extremes. The total time needed will then be equal to $t_n + (3/2)(p_n + 1) = t_n + (3/2)(t_n - 1)$, and hence the ratio between the solution value of GTR and that of the adversary tends to 5/2 as n tends to infinity.

Let us show that for every i, at time t_i the position of GTR is $(t_i - 3)/2$, that is, the center of the interval it still has to visit. This is obvious for $i = 0$: $p^{GTR}(3) = 0$. Assuming the hypothesis is true for i we will prove it for $i + 1$. At time t_i it leaves $(t_i - 3)/2$ towards the right extreme, where it arrives at time $t_i + (t_i - 3)/2 + 1 = (3/2)t_i - 1/2$. Then, it turns back towards -1, and at time $t_{i+1} = (5/3)t_i - 2/3$ will be at point $t_i - 2 - [((5/3)t_i - 2/3) - ((3/2)t_i - 1/2)] = (t_{i+1} - 3)/2$, the center of the new interval. □

4.2 An algorithm for H-OLTSP

The greedy algorithm GTR presented for N-OLTSP, can be transformed in a direct way into a greedy algorithm for H-OLTSP, by replacing "paths" by "paths finishing in the origin o". A similar analysis shows that GTR is 5/2-competitive for the H-OLTSP. It is not sure however that this ratio is tight in this case for the metric spaces belonging to $\mathcal{M}$. For the real line it can be shown that its competitive ratio is precisely 2, although better algorithms for this case exist as we shall see in Section 6.

However, for H-OLTSP we can exploit the requirement of having to return to the origin, within a greedy framework. This is done by making a difference between requests that

are relatively close and those that are relatively far from the origin, where we postpone serving the former set of requests. The algorithm that we devise, and which we call PAH (for Plan-At-Home) achieves a competitive ratio of 2 for any metric space belonging to $\mathcal{M}$. We emphasize that this is equal to the lower bound derived in Section 3.2. We abbreviate the position $p^{PAH}(t)$ of the PAH algorithm by p.

1. Whenever the server is at the origin, it starts to follow an optimal route that serves all the requests yet to be served and goes back to the origin.

2. If at time t a new request is presented at point x, then it takes one of two actions depending on its current position p:

   2a. If $d(x,o) > d(p,o)$, then the server goes back to the origin (following the shortest path from p) where it appears in a Case 1 situation.

   2b. If $d(x,o) \leq d(p,o)$ then the server ignores it until it arrives at the origin, where again it reenters Case 1.

Notice that requests that are presented between an occurrence of Case 2a and the arrival at the origin will not make the server deviate from his current shortest path back to the origin.

Theorem 4.2 *PAH is 2-competitive.*

Proof: Let t be the time of the last request, and let the position of this request be x. We show that in each of the three Cases 1, 2a and 2b, PAH is 2-competitive.

Let T^* be the optimal tour that starts at the origin, serves all the requests presented, and ends at the origin. Clearly, $Z^* \geq t$ since no algorithm can finish before the last request is presented. Also, trivially, $Z^* \geq |T^*|$.

1. In Case 1 PAH is at the origin at time t. Then it starts an optimal tour that serves all the unserved requests and goes back to the origin. The time needed by PAH is $Z^{PAH} \leq t + |T^*| \leq 2\ Z^*$.

If, when the new request arrives, PAH is not at the origin, we can distinguish two cases, corresponding to Cases 2a and 2b.

2a. $d(o,x) > d(o,p)$. Then PAH goes back to o, where it will arrive before time $t+d(o,x)$. After this, PAH computes and follows an optimal tour through all the unserved requests. Therefore, $Z^{PAH} < t + d(o,x) + |T^*|$.

Notice that $Z^* \geq t + d(o,x)$, since from the time t the request is presented every algorithm has to travel at least the distance from the request to the origin. This, together with $Z^* \geq |T^*|$ implies that $Z^{PAH} < 2\ Z^*$.

2b. $d(o,x) \leq d(o,p)$. Suppose PAH is following a route $\mathcal{R}$ that has been computed the last time it was at the origin. Let $\mathcal{Q}$ be the set of requests that have been temporarily ignored since the last time PAH left the origin. Let q be the location of the first request in $\mathcal{Q}$ served by the adversary, and t_q the time at which q was presented. Let $P^*_{\mathcal{Q}}$ be the shortest *path* that starts at q, visits all the points in $\mathcal{Q}$ and ends at o. Clearly, $Z^* \geq t_q + |P^*_{\mathcal{Q}}|$.

At time t_q, the distance that PAH still has to travel on the route $\mathcal{R}$ before arriving at o is at most $|\mathcal{R}| - d(o,q)$, since $d(o,p(t_q)) \geq d(o,q)$ implies that PAH has travelled on the route $\mathcal{R}$ a distance not less than $d(o,q)$. Therefore, it will arrive at o before time $t_q + |\mathcal{R}| - d(o,q)$. After that it will follow an optimal tour $T^*_{\mathcal{Q}}$ that covers the set $\mathcal{Q}$ of yet unserved requests. Hence, the total time to completion will be $Z^{PAH} \leq t_q + |\mathcal{R}| - d(o,q) + |T^*_{\mathcal{Q}}|$. Because $|T^*_{\mathcal{Q}}| \leq d(o,q) + |P^*_{\mathcal{Q}}|$, we have $Z^{PAH} \leq t_q + |\mathcal{R}| - d(o,q) + d(o,q) + |P^*_{\mathcal{Q}}| = t_q + |\mathcal{R}| + |P^*_{\mathcal{Q}}|$. Since, obviously, $Z^* \geq |\mathcal{R}|$ and, as established before, $Z^* \geq t_q + |P^*_{\mathcal{Q}}|$ we have that $Z^{PAH} \leq 2\ Z^*$.

□

The competitive ratio of 2 achieved by PAH is the best possible for metric spaces belonging to the class $\mathcal{M}$ (see Section 3.2). It is tight even for the real line as will be seen from the following instance where as usual point 0 is taken as the origin o. The sequence of requests starts with a request at time 1 at position +1. PAH remains at the origin until time 1 when it leaves towards 1. Then a sequence of requests is presented, one each time PAH arrives at point $+\epsilon$, at a point "slightly" to the left of $-\epsilon$, in such a way that PAH always turns back to 0. This goes on until time $2+\epsilon$. The optimal strategy consists of serving first the request in +1 and then all the requests to the left of 0 yielding a completion time arbitrarily close to $2+2\epsilon$, while PAH will be to the left of $+\epsilon$ at least until time $2+\epsilon$, yielding a completion time of at least 4. Making ϵ arbitrarily small gives a ratio of 2. In Section 6.2 we will give a better algorithm for the real line.

5 Polynomial time algorithms for OLTSP

We now turn the attention to polynomial time competitive algorithms for metric spaces belonging to $\mathcal{M}$. Even though competitiveness and computational complexity are not related concepts, for practical applications it is obviously relevant to make available polynomial time algorithms with a good competitive ratio.

In the following we will present 3-competitive polynomial time algorithms for metric spaces belonging to $\mathcal{M}$ that use two well-known approximation algorithms for the Euclidean TSP as subroutine.

5.1 A polynomial time algorithm for N-OLTSP

One known approximation algorithm for the Travelling Salesman Problem in which distances satisfy the triangular inequality (ΔTSP) is the 2-approximate Minimum Spanning Tree heuristic (e.g., see [14]). The MST heuristic provides a tour whose cost is at most twice the length of a minimum spanning tree. This heuristic also provides a 2-approximation algorithm to the Hamiltonian Path Problem, since the size of a minimum spanning tree is a lower bound on the total length of an optimal Hamiltonian path.

Let X be a set of points of the metric space M. We denote with $MST(X)$ the size of the minimum spanning tree of the complete graph with set of vertices X, every edge between two points x, y in X is weighted with the distance $d(x, y)$.

Before presenting the algorithm, we give a preliminary lemma.

Lemma 5.1 *For every pair of points x and y in a set X, there exists a 2-approximate tour on X in which x and y are adjacent.*

Proof: Consider the minimum spanning tree over X. The MST heuristic consists of doubling all the edges of the tree, which yields an Eulerian graph, i.e., a connected graph in which all vertices have even degree. The total edge length of this graph is $2MST(X)$, which is at most twice the optimal tour length. We then use the fact that for any Eulerian graph and any edge in that graph, a standard short-cutting argument based on the triangle inequality will construct a TSP tour that contains that edge and has length no more than the sum of the graph's edge lengths. Now note that starting from the doubled tree, we will still have an Eulerian graph if we replace a shortest path between x and y by a direct edge between x and y. This is because (a) all vertex degrees remain even (since the degrees of x and y remain unchanged and those of the internal vertices of the path are reduced by

2) and (b) the graph remains connected (one copy of the removed path must still remain in the graph). By the triangle inequality, the edge replacement cannot increase the total edge length of the graph, so by the above observation the lemma follows. □

Let S be the set of requests still to be served. Let Σ be the set containing all presented requests and the origin o, and $\mathcal{T}$ the shortest Hamiltonian path over Σ. Let us assume that the on-line server is at position p on the shortest path from the last served request, x, to an unserved request, y, when a new request is presented. The algorithm that we propose using the MST heuristic is similar to the greedy algorithm described before in Section 4.1. Instead of the optimal route through all the requests that still have to be served, an MST through all these points and the last visited point x is computed. A 2-approximate path starting with the route from x to y is now followed.

Theorem 5.2 *The algorithm that uses the MST heuristic is 3-competitive for N-OLTSP, and the ratio is tight.*

Proof: We first note that

$$MST(\{x\} \cup S) \leq MST(\Sigma) \leq |\mathcal{T}|.$$

Let t be the time the last request is presented. At t, the on-line server is on the path leading from x to a point $y \in S$. By Lemma 5.1 this is a legal start of a 2-approximate path on $\{x\} \cup S$. Hence, the on-line completion time is $Z^{OL} \leq t + 2MST(\{x\} \cup S)$. Since t and $MST(\{x\} \cup S)$ are both lower bounds on the optimal off-line completion time, the total cost of the on-line server is less than 3 times the cost of the optimal off-line algorithm.

The following example on the real line shows that 3 is a tight bound on the competitive ratio. Let the origin o be at point 0. At time 0 a request at point 1 is presented. At time ϵ a request at point 0 is presented. The MST contains the segment $\overline{01}$, and the on-line server is at point ϵ and continues to follow the segment $\overline{01}$ until 1. At time $1 + \epsilon$ the on-line server is at position $1 - \epsilon$ and a new request is given at point 1. Now the on-line server goes towards 0 and afterwards it goes back to 1. The total time is 3, while the optimal solution takes $1 + \epsilon$. □

It is interesting to note that the competitive ratio 3 obtained by this strategy is exactly the sum of the factor 2 of the approximation ratio of the heuristic plus 1, the same that would be obtained following the heuristic path after the last request is presented. Such a strategy can not be considered because no information about which is the last request is given to the on-line algorithm.

5.2 A polynomial time algorithm for H-OLTSP

In this section we present a 3-competitive polynomial algorithm for H-OLTSP that uses the 3/2-approximate polynomial algorithm by Christofides [14] for TSP on metric spaces. Observe that this heuristic has not been used for N-OLTSP since Lemma 5.1 does not hold for the 3/2-approximate algorithm.

Let S be the set of requests that have not yet been served by the on-line algorithm plus the origin o, $CHR(S)$ be the length of a 3/2-approximate tour over the set S, and T be the optimal tour over S. Moreover, let Σ be the set of presented requests plus the origin o, and $\mathcal{T}$ be the optimal tour over Σ.

The algorithm will always move on a shortest route between pairs of points of Σ. Assume that at time t, when a new request is presented, the server is travelling from point x to point y. The algorithm follows the shortest route to the origin o through x or y, and then a 3/2-approximate tour over S.

Theorem 5.3 *The algorithm that uses Christofides' heuristic is 3-competitive for H-OLTSP*

Proof: Let t be the time the last request is presented. Denote with $p^{OL}(t)$ the position of the on-line server that is travelling from point x to point y of Σ. Clearly, both t and $|\mathcal{T}|$ are lower bounds on the optimal off-line completion time. Let $D = d(o,x) + d(x,y) + d(o,y)$. Then we have $Z^* \geq D$ by the triangle inequality, since the points o, x and y must occur in that order in some orientation of the optimal tour. Furthermore, $d(o, p^{OL}(t)) \leq D/2$, again by the triangle inequality. Thus, $Z^{OL} \leq t + d(o, p^{OL}(t)) + CHR(S) \leq Z^* + (1/2)Z^* + (3/2)|\mathcal{T}| \leq 3Z^*$. □

We do not have a proof of the tightness of the competitive ratio for this heuristic.

6 OLTSP on the real line

In this section we consider the particular case in which the metric space is the real line. Clearly all the algorithms presented for metric spaces in $\mathcal{M}$ can also be applied to this case. Notice that the examples of tightness for the analysis of both the 5/2-competitive algorithm for N-OLTSP and the 2-competitive algorithm for H-OLTSP are given on the real line. Hence, there is no hope that those algorithms have better performance on the real line.

In order to obtain a better performance we have to design specific algorithms. In particular, a 7/3-competitive algorithm for N-OLTSP and a 7/4-competitive algorithm for H-OLTSP will be given. These competitive ratios are fairly close to the lower bounds for the problems on the real line presented in Section 3, 2 for N-OLTSP and ≈ 1.64 for H-OLTSP.

6.1 An algorithm for N-OLTSP on the line

In this section we give an algorithm for N-OLTSP on the real line that achieves a competitive ratio of 7/3. As we did before, we will consider the origin o at point 0.

Let I be the smallest interval containing the presented requests not yet served. The algorithm, which we call *ENO*, for "serve Extreme Nearest to the Origin first", consists in visiting I always starting from its extreme that is nearer to the origin.

Theorem 6.1 *Algorithm ENO is 7/3-competitive, and the ratio is tight.*

Proof: As usual let us assume that time t is the time of the last request.

Without loss of generality we suppose that of the two extreme requests not yet served at time t the leftmost one is nearest to the origin, and that the rightmost one, the one furthest from the origin, has positive abscissa.

At time t the interval $I = [x, X]$ is still to be served, with $X > 0$ and $|x| \leq X$, where $|x|$ denotes in this proof the absolute value of x. Observe that if $x > 0$ then I does not include the origin. Moreover, let $\mathcal{X}$ be the rightmost request in the past and $-\mathcal{Y}$ be either the leftmost request in the past or 0 in case the leftmost request has positive abscissa. Let $p^{ENO}(t)$ be the position of ENO at time t. Clearly, at time t the following holds: $t \leq Z^*$, $x \geq -\mathcal{Y}$, $X \leq \mathcal{X}$ and $-\mathcal{Y} \leq p^{ENO}(t) \leq \mathcal{X}$. We consider three cases depending on the position p^{ENO}:

1. $-\mathcal{Y} \leq p^{ENO}(t) \leq x$. ENO is to the left of x and will finish its work visiting once the interval that lies to its right. Since $p^{ENO}(t) \geq -\mathcal{Y}$ the total time needed by ENO is $Z^{ENO} \leq t + \mathcal{Y} + X$. To serve the whole set of requests, the whole interval $[-\mathcal{Y}, \mathcal{X}]$ must be travelled at least once, whence $Z^* \geq \mathcal{Y} + \mathcal{X}$. Therefore, the ratio in this case is

$$\frac{Z^{ENO}}{Z^*} \leq \frac{(t + \mathcal{Y} + X)}{Z^*} \leq 1 + \frac{\mathcal{Y} + X}{\mathcal{Y} + \mathcal{X}} \leq 2,$$

since $Z^* \geq t$ and $X \leq \mathcal{X}$.

2. $x \leq p^{ENO}(t) \leq |x|$, with $x < 0$ (this case coincides with the previous one if $x > 0$). In the worst case ENO is in position $|x|$ and must visit first the leftmost extreme in position x. The time needed by ENO is $Z^{ENO} \leq t + 3|x| + X$. The optimal time is at least $Z^* \geq 2|x| + \mathcal{X}$. From this and the assumption that $|x| \leq X \leq \mathcal{X}$ we have

$$\frac{Z^{ENO}}{Z^*} \leq \frac{(t + 3|x| + X)}{Z^*} \leq 1 + \frac{(3|x| + X)}{(2|x| + \mathcal{X})} \leq 7/3.$$

3. $|x| < p^{ENO}(t) \leq \mathcal{X}$. We consider two different cases:

   - In the optimal solution x is visited after $\mathcal{X}$. Then we have $Z^* \geq 2\mathcal{X} - x$. At time t ENO must cover at most twice the interval $[x, \mathcal{X}]$, in case it is very close to the rightmost extreme $\mathcal{X}$. Then it will finish by $Z^{ENO} \leq t - 2x + 2\mathcal{X}$ time and the ratio is

   $$\frac{Z^{ENO}}{Z^*} \leq 1 + \frac{(-2x + 2\mathcal{X})}{(-x + 2\mathcal{X})} \leq 7/3.$$

   - In the optimal solution $\mathcal{X}$ is visited after x. Suppose that the optimal off-line algorithm visits x at time d for the last time (with $d \geq |x|$, obviously). Then, at time d it still has to travel at least from x to $\mathcal{X}$. Thus, $Z^* \geq d - x + \mathcal{X}$. If $d \geq t$ we have that

   $$\frac{Z^{ENO}}{Z^*} \leq \frac{(d - 2x + 2\mathcal{X})}{(d - x + \mathcal{X})} \leq 2.$$

   Otherwise, if $d < t$, the following two claims hold:

   Claim 6.2 *At every time t', $d \leq t' \leq t$, $p^{ENO}(t') \geq |x|$.*

   Proof: We prove this by contradiction. The time at which the request in position x is presented is less than d since we assumed that at time d the optimal off-line algorithm has served x for the last time. Suppose ENO is in $p^{ENO}(t') \leq x$ at time t'. This implies that at time t the request in x has already been served since $p^{ENO}(t) \geq x$, which is a contradiction. This proves the claim for $x \geq 0$. For $x < 0$, suppose $x < p^{ENO}(t') < |x|$. Because x remains unserved at time t, ENO must have remained to the right of x until that time. Thus, since x is the leftmost unvisited request at time t, it also must have been so at time d. For ENO to end up to the right of $|x|$ at time t, as we are assuming in this case, it would have to travel away from x. However, this could have

happened only so long as the rightmost unvisited request at the time was less than or equal to $|x|$, and so could not have caused ENO to be at the right of $|x|$ at time t, a contradiction. □

Claim 6.3 *Starting at time d ENO moves to the left until time t.*

Proof: From the previous claim we know that between time d and time t ENO is always to the right of $|x|$. We notice that at any time during this period the extreme point of the interval of yet unserved requests that is nearest to the origin must be inside $[-|x|, |x|]$, implying that ENO always moves to the left during this period. This can be readily seen from the fact that, given the previous claim, during the whole period x always remains the leftmost point not yet served, and the on-line server is to the right of $|x|$ at time t. □

Thus, at time d ENO starts from a position $p^{ENO}(d) \geq |x|$ to travel to the left and at time t it is still to the right of $|x|$. Therefore, $Z^{ENO} \leq d + 2\mathcal{X} - 2x$ yielding the ratio

$$\frac{Z^{ENO}}{Z^*} \leq \frac{(d - 2x + 2\mathcal{X})}{(d - x + \mathcal{X})} \leq 2.$$

We finally prove that there is a sequence of requests for which ENO achieves a ratio of 7/3. This case is illustrated in Figure 3.

At time 1 two requests in -1 and 1/2 are presented. At time 1 ENO leaves 0 towards 1/2 and arrives at the origin at time 2 when a new request is presented in $1 - \epsilon$. Again, ENO goes to the right and arrives in $1 - \epsilon$ at time $3 - \epsilon$. At that time a new request is given in $1 + \epsilon$ and ENO goes to the left since the extreme point in -1 is nearer to the origin than $1 + \epsilon$. Altogether ENO takes time $7 - \epsilon$ to serve the requests, while the optimal off-line solution needs $3 + \epsilon$. The ratio tends to 7/3 as ϵ tends to 0. □

6.2 An algorithm for H-OLTSP on the line

In this section we present an algorithm for H-OLTSP on the real line whose competitive ratio is 7/4. We call this algorithm PQR (for Possibly-Queue-Requests). As in PAH, the 2-competitive algorithm for any metric space in $\mathcal{M}$ (see Section 4), PQR is based on the idea of postponing requests close to the origin.

At any point in time let $\mathcal{S}$ be the set of requests unserved by PQR and let $\mathcal{Q}$, the *queue*, be the subset of $\mathcal{S}$ containing the requests that are temporarily ignored. PQR

always follows the shortest tour from its current position through the set $\mathcal{P} = \mathcal{S} \backslash \mathcal{Q}$ finishing at the origin, followed by the shortest tour serving the requests in $\mathcal{Q}$.

PQR works in phases. The first phase starts with the first request, and each successive phase starts when a new request is presented that is not on the currently scheduled tour and whose absolute value is bigger than that of any other unserved request.

At the beginning of a phase, PQR schedules the shortest route that, starting from its current position $p^{PQR}(t)$, serves all the unserved requests and goes back to the origin. We call this route the *greedy route*. Requests may be presented *during* the phase. Some of them may call for computation of a new greedy route, while others are simply added to $\mathcal{Q}$ and cause a recomputation of the shortest tour through $\mathcal{Q}$.

We denote the current route remaining to be traversed as $\mathcal{R}$, with $\mathcal{R}$ being the concatenation of $\mathcal{G}$, the part of the most recently computed greedy tour that remains to be traversed (and will visit all the cities in $\mathcal{P}$) followed by $\mathcal{H}$, the optimal tour for $\mathcal{Q}$.

During a phase, we refer to the *long side* as the half line from 0 on which the request whose presentation caused the start of the phase is located. The other side is then referred to as the *short side*.

When a phase starts the set $\mathcal{Q}$ is empty. By the construction of our algorithm, $\mathcal{Q}$ will only contain requests on the short side. Requests are removed from $\mathcal{Q}$ as soon as they are served.

PQR is described completely by its behavior when a new request is presented, say at time t.

1. If the new request is on route $\mathcal{R}$ then proceed following $\mathcal{R}$, add the request to $\mathcal{P}$ or $\mathcal{Q}$, depending on whether it is first visited in $\mathcal{G}$ or $\mathcal{H}$, and serve the request when visited; else

2. If the new request is on the long side, then empty the set $\mathcal{Q}$ and redefine $\mathcal{R}$ as the newly computed greedy route. If the new request is further from the origin than any unserved request then also a *new phase* starts; else

3. If the new request is on the short side and it is further from the origin than any unserved request then a *new phase* starts, empty the set $\mathcal{Q}$ and redefine $\mathcal{R}$ as the newly computed greedy route; else

4. The request is on the short side but no new phase starts. Insert the request in $\mathcal{Q}$, redefine $\mathcal{H}$ as the shortest tour that starts at the origin, visits all of $\mathcal{Q}$, and returns.

Theorem 6.4 *PQR is 7/4-competitive, and the ratio is tight.*

Proof:

We show that PQR is 7/4 competitive. Suppose that the last request is presented at time t. Without loss of generality, we will suppose that at time t the long side is the right side, and the short side is the left side. Moreover, let $-\mathcal{Y}$ and $\mathcal{X}$ be, respectively, the leftmost and the rightmost request *ever* presented. When the time t is clear from the context we abbreviate $p^{PQR}(t)$ with p.

There are four cases, depending on which rule the algorithm applies. The proof is by induction on the number of requests, so we may assume that if this last request never arrived, PQR would be 7/4 competitive. The induction hypothesis trivially holds in case of no or 1 requests.

1. In the first case, the new request is on the currently followed route $\mathcal{R}$. Z^{PQR} does not increase, and hence PQR remains 7/4 competitive.

2. In case 2 the new request is on the right of the origin, at position X. After its presentation, X is the rightmost unserved request and $p \leq X$, since otherwise we would have had a Case 1 situation. Let $-x$ be the leftmost unserved request. If there is no unserved request left of 0, we set $-x = 0$. Two cases are distinguished:

   - In the first case $-x < p$. Since PQR follows the newly computed route starting at time t from p, we have that $Z^{PQR} \leq t + 2x + 2X$. For the optimal algorithm $Z^* \geq t + X$ since the new request in X cannot be served before time t, and the algorithm must end the tour at the origin. We also have $Z^* \geq 2x + 2\mathcal{X}$. Moreover, $x \leq \mathcal{X}$ since either the current request in X or a previous request on the right of the origin has started a new phase. We then conclude:

$$\frac{Z^{PQR}}{Z^*} \leq \frac{t + 2x + 2X}{Z^*} = \frac{t + X}{Z^*} + \frac{2x + X}{Z^*} \leq 1 + \frac{2x + X}{2x + 2\mathcal{X}} \leq \frac{7}{4}.$$

   - In the second case $p \leq -x$. The time needed by PQR to serve all the requests is then $Z^{PQR} \leq t + |p| + 2X$. As before we have $Z^* \geq t + X$. Moreover, the position p of PQR implies that there must have been a request at distance at least $|p|$ left from the origin, i.e., $\mathcal{Y} \geq |p|$. Obviously, $Z^* \geq 2\mathcal{Y} + 2X$. This implies that

$$\frac{Z^{PQR}}{Z^*} \leq \frac{t + |p| + 2X}{Z^*} = \frac{t + X}{Z^*} + \frac{|p| + X}{Z^*} \leq 1 + \frac{|p| + X}{2\mathcal{Y} + 2X} \leq \frac{3}{2}.$$

3. In this case the new request is on the left of the origin at position $-x$. The new request is further from the origin than any other unserved request. Let X be the rightmost unserved request. Clearly, $X < x$. If there is no unserved request to the right of 0, we set $X = 0$. At time t, $p > -x$, since otherwise we would have had a Case 1 situation. We distinguish two cases.

   - In the first case we have $-x < p < X$. Since PQR recomputes an optimal greedy route at time t, we have $Z^{PQR} \leq t+2x+2X$. For the optimal algorithm $Z^* \geq t + x$ since the new request at position $-x$ cannot be served before time t. Moreover, $Z^* \geq 2x + 2X$. We then obtain that

$$\frac{Z^{PQR}}{Z^*} \leq \frac{t + 2x + 2X}{Z^*} = \frac{t + x}{Z^*} + \frac{x + 2X}{Z^*} \leq 1 + \frac{x + 2X}{2x + 2X} \leq \frac{7}{4}.$$

   - In the second case we have that $p \geq X$. The time needed by PQR to serve all the requests and return to the origin is $Z^{PQR} \leq t + |p| + 2x$, with $|p| \leq \mathcal{X}$. For the optimal solution we have $Z^* \geq t + x$ and $Z^* \geq 2x + 2\mathcal{X}$. We conclude that

$$\frac{Z^{PQR}}{Z^*} \leq \frac{t + |p| + 2x}{Z^*} = \frac{t + x}{Z^*} + \frac{x + |p|}{Z^*} \leq 1 + \frac{x + \mathcal{X}}{2x + 2\mathcal{X}} \leq \frac{3}{2}.$$

4. In this case the request inserted into $\mathcal{Q}$ is further from the origin than any of the other unserved requests on the short side, but closer to the origin than the furthest unserved request on the long side. Let $-x''$ be the position of this request, $-x'$ and X' the leftmost and the rightmost unserved requests when the greedy tour was last computed, and $-x$ and X the leftmost and rightmost unserved requests when the current phase started and so the current long side was declared to be such. Note that we must have $x'', x', X' \leq X$. We consider two subcases:

   - In the optimal solution X' is served before $-x''$. Let t' be the time at which the request in X' was presented, i.e., the time at which the current greedy route was computed. For the optimal solution we get $Z^* \geq t' + X' + 2x''$.

     Another two subcases are distinguished, depending on $p^{PQR}(t')$, i.e., the position of PQR at time t', relative to the interval $I = [-x', X']$:

     - PQR was inside I at time t'. Notice that, at time t PQR is still working on the greedy tour that was computed last since the request at t did not cause a new phase. Therefore, $Z^{PQR} \leq t' + 2x' + 2X' + 2x''$, while $Z^* \geq 2x' + 2X$.

In this case the ratio is

$$\begin{aligned}\frac{Z^{PQR}}{Z^*} &\leq \frac{t' + 2x' + 2X' + 2x''}{Z^*} = \frac{t' + X' + 2x''}{Z^*} + \frac{2x' + X'}{Z^*} \\ &\leq 1 + \frac{2x' + X'}{2x' + 2X} \leq \frac{7}{4},\end{aligned}$$

since $x', X' \leq X$.

– PQR was outside I at time t'. Observe that it could not be to the right of X', otherwise a Case 1 situation would have occurred. Therefore, it was necessarily to the left of $-x'$. This implies that $Z^{PQR} \leq t' + |p^{PQR}(t')| + 2X' + 2x''$. Obviously, $|p^{PQR}(t')| \leq \mathcal{Y}$, and $Z^* \geq 2\mathcal{Y} + 2X$. Therefore,

$$\begin{aligned}\frac{Z^{PQR}}{Z^*} &\leq \frac{t' + |p^{PQR}(t')| + 2X' + 2x''}{Z^*} = \frac{t' + X' + 2x''}{Z^*} + \frac{|p^{PQR}(t')| + X'}{Z^*} \\ &\leq 1 + \frac{|p^{PQR}(t')| + X'}{2\mathcal{Y} + 2X} \leq \frac{3}{2}.\end{aligned}$$

- In the optimal solution $-x''$ is served before X'. Then, for the optimal solution we have $Z^* \geq t + x'' + 2X'$. Again, two subcases:

  – PQR is to the right of $-x'$ at time t (it is certainly to the left of X'). We consider two more subcases:

    * $x' < x''$. Then $Z^{PQR} \leq t + 2x' + 2X' + 2x''$ while $Z^* \geq 2x'' + 2X$. In this case the ratio is

$$\begin{aligned}\frac{Z^{PQR}}{Z^*} &\leq \frac{t + 2x' + 2X' + 2x''}{Z^*} = \frac{t + 2X' + x''}{Z^*} + \frac{2x' + x''}{Z^*} \\ &\leq 1 + \frac{2x' + x''}{2x'' + 2X} \leq \frac{7}{4}.\end{aligned}$$

    * $x' \geq x''$. Since the request in $-x''$ is not visited on the current route, PQR has already visited x' at time t and $p > -x''$. Then, $Z^{PQR} < t + x'' + 2X' + 2x''$ whereas $Z^* \geq 2x'' + 2X$. The ratio is

$$\begin{aligned}\frac{Z^{PQR}}{Z^*} &\leq \frac{t + x'' + 2X' + 2x''}{Z^*} = \frac{t + 2X' + x''}{Z^*} + \frac{2x''}{Z^*} \\ &\leq 1 + \frac{2x''}{2x'' + 2X} \leq 3/2,\end{aligned}$$

      since $x'' \leq X$.

  – PQR is to the left of $-x'$ at time t. p cannot be to the left of $-x''$ because in that case $-x''$ would be on the current route, and therefore we have

that $x' < x''$ and p is inside the interval $[-x'', -x']$. Thus, $Z^{PQR} \leq t + x'' + 2X' + 2x''$. Since $Z^* \geq t + x'' + 2X'$ and $Z^* \geq 2x'' + 2X$ we have

$$\frac{Z^{PQR}}{Z^*} \leq \frac{t + x'' + 2X' + 2x''}{Z^*} = \frac{t + 2X' + x''}{Z^*} + \frac{2x''}{Z^*} \leq \frac{3}{2},$$

since $x'' \leq X$.

That the bound on the competitive ratio of 7/4 is asymptotically tight for PQR is proved by the following example. At time 1 requests are presented in both $+1$ and -1. PQR is at the origin until time 1, when it starts a greedy tour that without loss of generality serves $+1$ at time 2, -1 at time 4 and returns at the origin at time 5. However, at time 3, when PQR crosses the origin, a new request is presented at position $1+\epsilon$. This new request starts a new phase since it is the furthest request not yet served. Then a new greedy route is computed that starts from the origin at time 3, serves the two requests in -1 and $1+\epsilon$ and returns back at the origin at time $7+2\,\epsilon$. As for the optimal algorithm it serves -1 at time 1, $+1$ at time 3, $1+\epsilon$ at time $3+\epsilon$ and returns at the origin at time $4+2\epsilon$. Making ϵ arbitrarily small, the ratio is arbitrarily close to 7/4. □

7 Concluding Remarks

We have studied a classical routing problem from a new point of view that is natural and realistic, given the great quantity of applications in which travelling must be started before having complete information about the requests to be served.

Vehicle routing problems have been often considered with time window constraints, i.e., each request has to be served between a given release time and a given deadline. With regard to release times observe that our lower and upper bounds hold also if we change the problem by allowing a release time different from the time in which the request is presented. Then the i-th request is specified by a triple $< t_i, p_i, r_i >$, and the relation $t_i \leq r_i$ holds for every i, with the meaning that requests may be presented at any moment not later than their release times. In that case our algorithms will simply ignore the requests until their release times arrive, obtaining the same upper bounds.

It is obviously an open problem to close the remaining gaps between lower and upper bounds: 5/2 vs. 2 for N-OLTSP on general metric spaces in the class $\mathcal{M}$, and for the real line 7/3 vs. 2 for N-OLTSP and 7/4 vs. ≈ 1.64 for H-OLTSP.

It would also be interesting to study other particular metric spaces (such as trees, cycles, half-lines, etc.), to see if better bounds can be obtained (as we did for the real

line).

An interesting extension of OLTSP is to *crew scheduling* in which more than one server is used to serve the requests. In this case a 2.5-competitive algorithm can be easily obtained for the Homing version of this on-line problem by the following strategy working in phases: Each time a new request is presented, all the servers return to the origin and plan an optimal tour for covering all the unserved requests, (and eventually end to the origin). However, improvements are certainly possible.

Other possible extensions of the problems considered in this paper may take into account different objective functions. For instance consider the sum (or the average) over all the requests of the individual service times, defined as the time at which the request is served, or of the individual service delays, defined as the interval between the time at which the request is presented and the time at which the request is served.

A second class of problems that may be considered is provided by the *dial-a-ride* scenario, in which each request consists of moving an item located at a certain position in the metric space to a second position in the metric space. Examples of such problems (with multiple servers) are a system of taxis in a city or a system of elevators. Notice that in those cases one might wish to impose an extra constraint covering the capacity of the server.

Acknowledgments

We are grateful to David Johnson for his extensive comments on an earlier version of this paper.

References

[1] H. Alborzi, E. Torng, P. Uthaisombut and S. Wagner, The k-client problem, *Proc. Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, (1997), pp. 73-82.

[2] M. Atallah and S. Kosaraju, Efficient solutions for some transportation problems with application to minimizing robot arm travel, *SIAM J. on Computing*, 17 (1988), pp. 849-869.

[3] A. Borodin and R. El-Yaniv,*On-line computation and competitive analysis*, Cambridge University Press, Cambridge UK, 1998.

[4] S. Ben-David, A. Borodin, R.M. Karp, G. Tardos and A. Widgerson, On the power of randomization in on-line algorithms. *Proc of the 22nd Annual ACM Symposium on Theory of Computing*, pp. 379-386, 1990.

[5] R. El-Yaniv, A. Fiat, R.M. Karp and G. Turpin, Competitive analysis of financial games, *Proc. 33rd Annual Symposium on Foundations of Computer Science* (1992), pp. 327-333.

[6] G. Frederickson, A note on the complexity of a simple transportation problem, *SIAM J. on Computing*, 22-1 (1993), pp. 57-61.

[7] G. Frederickson and D. Guan, Preemptive ensemble motion planning on a tree, *SIAM J. on Computing*, 21-6 (1992), pp. 1130-1152.

[8] G. Frederickson, M. Hecht and C. Kim, Approximation algorithms for some routing problems, *SIAM J. on Computing*, 7-2 (1978), pp. 178-193.

[9] M. Garey, and D. Johnson, *Computers and intractability: a guide to the theory of NP-completeness*, Freeman, San Francisco (1979).

[10] R.L. Graham, "Bounds for certain multiprocessing anomalies", *Bell System Technical Journal* 45 (1966), pp. 1563-1581.

[11] D.S. Johnson. *Near optimal bin-packing algorithms.* Doctoral thesis, MIT, 1973.

[12] Y. Karuno, H. Nagamochi and T. Ibaraki, Vehicle scheduling on a tree with release times and handling times, *Proc. 4th. International Symposium on Algorithms and Computation ISAAC'93*, LNCS 762 (1993), Springer-Verlag, pp. 486-495.

[13] B. Kalyanasundaram and K.R. Pruhs, Constructing competitive tours from local information, *Proc. 20th International Colloquium on Automata, Languages and Programming*, LNCS 700 (1993), Springer-Verlag.

[14] E.L. Lawler, J.K. Lenstra, A.H.G. Rinnooy Kan and D.B. Shmoys (eds.), *The travelling salesman problem: a guided tour of combinatorial optimization*, Wiley, Chichester (1985).

[15] M. Manasse, L.A. McGeoch and D. Sleator, Competitive algorithms for server problems, *Journal of Algorithms* 11 (1990), pp. 208-230.

[16] H. Psaraftis, M. Solomon, T. Magnanti and T. Kim, Routing and scheduling on a shoreline with release times, *Management Science* 36-2 (1990), pp. 212-223.

[17] D.B. Shmoys, J. Wein, D.P. Williamson, "Scheduling parallel machines on-line", *Proc. of the 32nd Annual Symposium on Foundations of Computer Science*, 1991.

[18] D. Sleator, R. Tarjan, Amortized efficiency of list update and paging algorithms, *Comm. ACM* 28 (1985), pp. 202-208.

[19] D. Sleator, R. Tarjan, Self-adjusting binary search trees, *Journal of the ACM*, 32 (1985), pp. 652-686.

[20] M. Solomon, "Algorithms for the vehicle routing and scheduling problem with time window constraints", *Operations Research*, 35-2 (1987), pp. 254-265.

[21] M. Solomon and J. Desrosiers, "Time window constrained routing and scheduling problems: a survey", *Transportation Science*, 22 (1988), pp. 1-13.

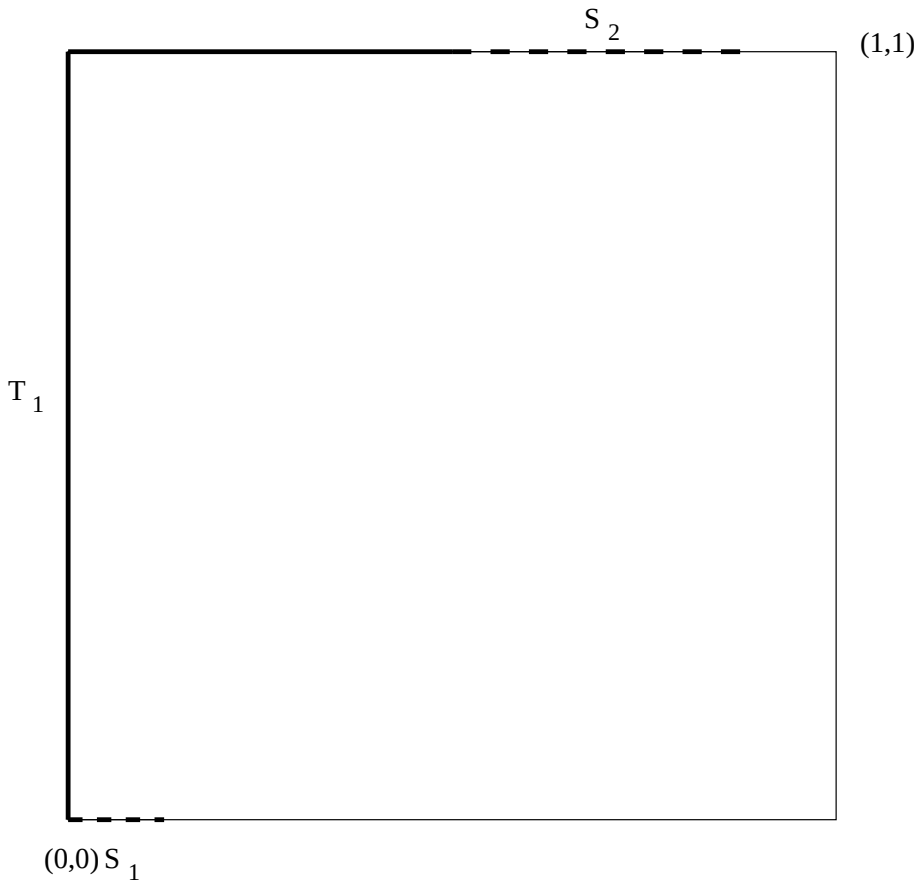

Figure 1: The lower bound for HOLTSP

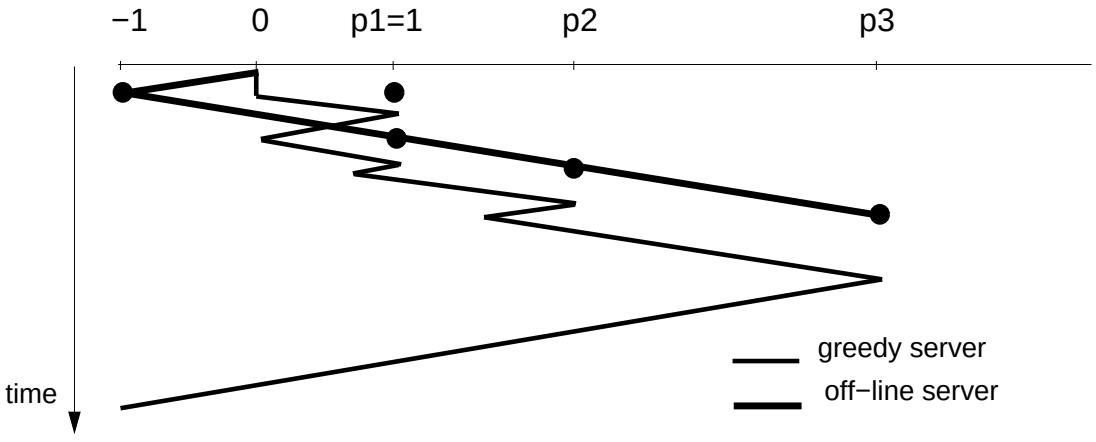

Figure 2: A worst-case sequence for GTR

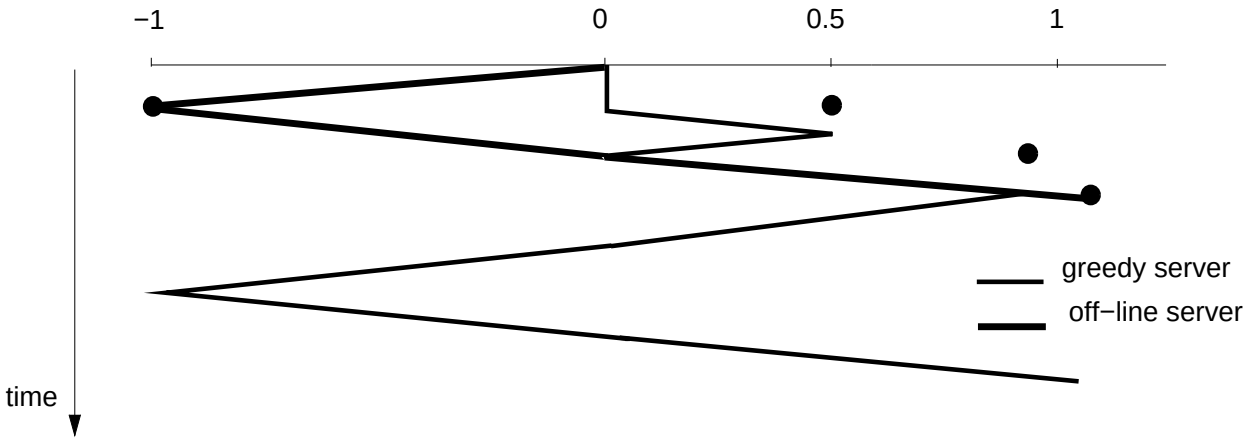

Figure 3: A worst-case sequence for the ENO algorithm