The online prize-collecting traveling salesman problem

Giorgio Ausiello [a], Vincenzo Bonifaci [a,b,*,1], Luigi Laura [a]

[a] *Dipartimento di Informatica e Sistemistica, Sapienza Università di Roma. Via Ariosto 25, 00185 Roma, Italy.*

[b] *Dipartimento di Ingegneria Elettrica, Università dell'Aquila. Monteluco di Roio, 67040 L'Aquila, Italy.*

Abstract

We study the online version of the Prize-Collecting Traveling Salesman Problem (PCTSP), a generalization of the Traveling Salesman Problem (TSP). In the TSP, the salesman has to visit a set of cities while minimizing the length of the overall tour. In the PCTSP, each city has a given weight and penalty, and the goal is to collect a given quota of the weights of the cities while minimizing the length of the tour plus the penalties of the cities not in the tour. In the online version, cities are disclosed over time. We give a 7/3-competitive algorithm for the problem, which compares with a lower bound of 2 on the competitive ratio of any deterministic algorithm. We also show how our approach can be combined with an approximation algorithm in order to obtain an $O(1)$-competitive algorithm that runs in polynomial time.

Key words: on-line algorithms, analysis of algorithms, combinatorial problems

1 Introduction

In the well known Traveling Salesman Problem, a salesman has to visit a set of cities to sell his merchandise, and his goal is to minimize the length of the tour.

* Corresponding author.
Email addresses: ausiello@dis.uniroma1.it (Giorgio Ausiello), bonifaci@dis.uniroma1.it (Vincenzo Bonifaci), laura@dis.uniroma1.it (Luigi Laura).

[1] This work was partially supported by the Future and Emerging Technologies Unit of EC (IST priority - 6th FP), under contract no. FP6-021235-2 (project ARRIVAL).

Let us consider the more general case in which each city has both a *penalty* and a *weight* associated with it; now the commitment of the salesman is to collect a given *quota* of weights, by visiting a sufficient number of cities; the final cost will be the length of the tour plus the penalty of every city that was not visited. This problem is known as the *Prize-Collecting Traveling Salesman Problem* (PCTSP) [7]. If all the penalties are equal to zero, then the PCTSP reduces to the special case known as the Quota Traveling Salesman Problem [6, 8], also called sometimes the Quorum-Cast problem [9]. On the other hand, when the quota is zero, i.e. there is no requirement to visit any city at all, the resulting problem is known [2] as the Profitable Tour Problem [10]. Related to the PCTSP is also the so called k-TSP [3] problem, i.e. the problem of finding a tour of minimum length which visits k cities among the given ones. Thus, the PCTSP generalizes a number of interesting routing problems.

In this paper we address the online version of the PCTSP, in which the requests arrive over time in a metric space and a server (the traveling salesman) has to decide which requests to serve and in what order to serve them, without yet knowing the whole sequence of requests; the goal is, as in the offline PCTSP, to collect the quota while minimizing the sum of the time needed to complete the tour and the penalties associated to the requests not in the tour. We study the online PCTSP in the framework of *competitive analysis*, which was already applied to other routing problems [3, 4]. In competitive analysis, the performance of the online algorithm is compared to that of an offline solver that knows all the requests ahead of time, together with their release dates, and serves them optimally. The worst-case ratio between the costs of an online algorithm and the optimum offline solution is called the *competitive ratio* of the online algorithm.

The rest of the paper is structured as follows. We begin by providing a formal definition of the offline problem and an overview of approximation results in Section 2. We introduce the online model of the problem in Section 3. In Section 4, we observe that a lower bound on the competitive ratio of any algorithm for the online PCTSP is 2 and we give a 7/3-competitive algorithm. In fact, we show the stronger result that given a ρ_0-approximation algorithm for the offline PCTSP and a ρ-approximation algorithm for the offline PCTSP with release dates, we can construct a c-competitive algorithm for the online PCTSP that runs in polynomial time, where

$$c = \rho + \frac{\rho}{1 + 2/\rho} + \rho_0.$$

[2] Some authors use the name PCTSP to refer to this special case, which can be a source of confusion.

[3] This traveling salesman problem on k *cities* should not be confused with the traveling salesman problem with k *salesmen*, which is also sometimes referred to as k-TSP.

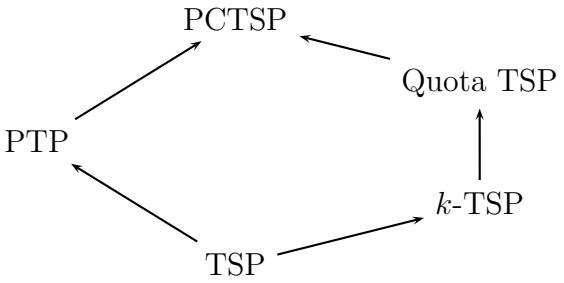

Figure 1. Relations between TSP, k-TSP, Quota TSP, PTP and PCTSP.

Finally, we give further results and conclusions in Section 5.

2 The offline PCTSP

We begin by formally defining the offline PCTSP.

Definition 1 *An instance of the* Prize-Collecting Traveling Salesman Problem*, is given by a metric d over a finite space $\{1,\ldots,n\}$. Moreover, the instance specifies $2n+1$ numbers $Q, w_1,\ldots,w_n,\pi_1,\ldots,\pi_n \in \mathbb{Q}_+$. A feasible solution is given by a* tour*, that is a cyclic permutation φ on some set $S \subseteq \{1,\ldots,n\}$ such that $1 \in S$ and $\sum_{i\in S} w_i \geq Q$. The cost of the solution is $c(\varphi) = \ell(\varphi) + \pi(\varphi)$ where $\ell(\varphi) = \sum_{i\in S} d_{i\varphi(i)}$ and $\pi(\varphi) = \sum_{i\notin S} \pi_i$.*

Several other problems can be defined in terms of the PCTSP by restricting some of the input parameters (the set of feasible solutions and the costs are defined exactly as in the PCTSP):

- *Profitable Tour Problem* (PTP): set $Q = 0$ and $w_i = 0$ for $i = 1,\ldots,n$;
- *Quota TSP*: set $\pi_i = 0$ for $i = 1,\ldots,n$;
- *k-TSP*: set $Q = k$ and $\pi_i = 0$, $w_i = 1$ for $i = 1,\ldots,n$.

Notice that the standard Traveling Salesman Problem is a special case of both the PTP (by letting every penalty be sufficiently high) and of the k-TSP (when $k = n$). The lattice of relations between all these problems is given in Figure 1, where an arrow from problem A to problem B means that A is a special case of B.

The best results known so far for PCTSP and its variants are summarized by the following theorem.

Theorem 2 ([2, 6, 11, 12]) PCTSP *admits an $O(1)$-approximation algorithm. In particular,*

(1) PTP *admits a 2-approximation algorithm;*
(2) k-TSP *admits a 2-approximation algorithm;*

(3) Quota TSP *admits a* 10*-approximation algorithm;*
(4) PCTSP *admits a* 12*-approximation algorithm.*

PROOF. Parts (1), (2) and (3) follow from Goemans and Williamson [12], Garg [11] and Ausiello et al. [2] respectively. Part (4) follows from applying the techniques of Awerbuch et al. [6] to the more recent results (1–3).

□

3 The online model

In this section we formally define the online PCTSP. Let $\mathbb{M}$ be a metric space, with a distinguished point o called the *origin*. As in previous works [3], we only consider so-called *path metric* spaces, in which the distance $d(x, y)$ between two points x and y is equal to the length of the shortest path between x and y. We also require the space to be *continuous*, in the sense that for all $x, y \in \mathbb{M}$ and for all $a \in [0, 1]$ there is $z \in \mathbb{M}$ such that $d(x, z) = ad(x, y)$ and $d(z, y) = (1 - a)d(x, y)$. A discrete space, like a weighted graph, can be extended to a continuous path metric space in the natural way; the continuous space thus obtained is said to be *induced* by the original space.

The input is given by a pair (Q, σ), where $Q \in \mathbb{Q}_+$ is called the *quota* and $\sigma = \sigma_1 \cdots \sigma_n$ is a sequence of *requests*. Every request σ_i is a quadruple (r_i, x_i, w_i, π_i), where $r_i \in \mathbb{R}_+$ is the *release date* of the request, $x_i \in \mathbb{M}$ its *location*, $w_i \in \mathbb{Q}_+$ its *weight* and $\pi_i \in \mathbb{Q}_+$ its *penalty*. We also assume that the sequence is ordered such that $i < j$ implies $r_i \leq r_j$. All the information about a request, including its existence, becomes known only at its release date. On the other hand, the quota is revealed immediately to the algorithm.

The algorithm controls a single *server* (traveling salesman), initially located at the origin. The server can move around the space at most at unit speed. To *serve* a request, the server must visit the location of the request not earlier than its release date.

A feasible solution for instance (Q, σ) is a *schedule*, that is, a sequence of moves of the server such that the following conditions are satisfied: (1) the server starts in the origin, (2) the total weight of the served requests is at least Q, and (3) the server ends in the origin. Let S be a schedule for (Q, σ). The time at which the server returns permanently at the origin is called the *makespan* of S. The cost of a schedule S is the sum of the makespan of S and of the penalties associated to requests not served in S.

In the following, we denote by $A(Q,\sigma)$ the cost incurred by an algorithm A on input (Q,σ). An online algorithm A is *c-competitive* if, for every Q and σ,

$$A(Q,\sigma) \leq c \cdot \mathrm{OPT}(Q,\sigma),$$

where $\mathrm{OPT}(Q,\sigma)$ is the cost of an optimal solution to the instance (Q,σ).

For a sequence σ, let $\pi(\sigma) = \sum_{\sigma_i \in \sigma} \pi_i$. By $\sigma^{\leq t}$ we denote the subsequence of σ including all the requests having release date less than or equal to t, and similarly $\sigma^{>t}$ is the suffix of σ consisting of all the requests having release date strictly larger than t. When Q and σ are clear from the context, we denote by $A(t)$ the total cost incurred by algorithm A over the sequence $\sigma^{\leq t}$, that is, $A(Q,\sigma^{\leq t})$. Finally, we use $A[t]$ to denote the schedule computed by algorithm A on input $(Q,\sigma^{\leq t})$.

4 A competitive algorithm for the online PCTSP

In this section we give lower and upper bounds on the competitive ratio of the online PCTSP. We start with a lower bound.

Theorem 3 *The competitive ratio of any deterministic online algorithm for the online PCTSP is at least 2, even if $Q = 0$ or if $\pi_i = 0$ for all requests σ_i.*

PROOF. When $Q = 0$, we can give requests with arbitrarily high penalties so that the problem becomes equivalent to the standard online TSP, for which a lower bound of 2 is known [3].

When $\pi_i = 0$ for all requests r_i, the problem is equivalent to the online Quota TSP, for which a lower bound of 2 is also known, even for fixed Q [4].

□

Note that the lower bound above is known to be tight when all penalties are zero (there is a 2-competitive algorithm for Quota TSP [4]) or when all penalties are large enough [3].

In order to get our competitiveness result, we begin by proving a lemma on the properties of $\mathrm{OPT}(t)$ as a function of t. Notice that the fact that a request is served in the optimal solution for a sequence σ does not imply that the request is served in an optimal solution for $\sigma^{\leq t}$; this is why the following lemma is not trivial.

Lemma 4 *Consider an instance (Q, σ) of the online PCTSP and let σ_l be a request with maximum release date among the requests served in an optimal offline solution. Then*

(a) $\mathrm{OPT}(t) + \pi(\sigma^{>t}) = \mathrm{OPT}(r_n)$ *for all* $t \in [r_l, r_n]$*;*
(b) $\mathrm{OPT}(r_l) \geq r_l$*.*

PROOF.

(a) Since a feasible solution for σ is to first serve optimally $\sigma^{\leq t}$ and then pay penalties for all successive requests, we have

$$\mathrm{OPT}(r_n) \leq \mathrm{OPT}(t) + \pi(\sigma^{>t}). \tag{1}$$

On the other hand, a feasible solution for $\sigma^{\leq t}$ is to follow the optimal offline schedule for σ, of course saving on the penalty cost of the requests released after time t, so we have

$$\mathrm{OPT}(t) \leq \mathrm{OPT}(r_n) - \pi(\sigma^{>t}). \tag{2}$$

The claim follows by combining (1) and (2).

(b) Assume by contradiction that $\mathrm{OPT}(r_l) < r_l$. By (a),

$$\begin{aligned} \mathrm{OPT}(r_n) &= \mathrm{OPT}(r_l) + \pi(\sigma^{>r_l}) \\ &< \quad r_l + \pi(\sigma^{>r_l}). \end{aligned}$$

But since there is an optimal schedule for σ that serves σ_l and no request with a release date later than r_l,

$$\mathrm{OPT}(r_n) \geq r_l + \pi(\sigma^{>r_l})$$

which gives a contradiction.

□

In order to give an algorithm for the online PCTSP, we need to be able to solve the offline PCTSP. We assume that we have black-box access to two algorithms. The first, algorithm A_0, gives ρ_0-approximate solutions to PCTSP instances (as in Definition 1). A second algorithm A constructs ρ-approximate solutions to the PCTSP with release dates. The next lemma shows that given A_0, there is an A such that $\rho \leq 1 + \rho_0$.

Lemma 5 *Let A_0 be a ρ_0-approximation algorithm for the offline* PCTSP *(without release dates). Then there is a $(1 + \rho_0)$-approximation algorithm A for the offline* PCTSP *with release dates.*

PROOF. On input (Q, σ), where $\sigma = \sigma_1 \cdots \sigma_n$, A calls A_0 on the $n+1$ inputs (Q, ϵ), (Q, σ_1), $(Q, \sigma_1\sigma_2)$, ..., $(Q, \sigma_1 \ldots \sigma_n)$. Let the corresponding outputs be $S_0, \ldots, S_n$. For every such solution S_i, A considers the following schedule: wait until all requests served by S_i are released, then follow S_i. The schedule output by A will be the cheapest among the $n+1$ schedules thus constructed.

More precisely, let $r(S_i)$ be the maximum release date of a request served in S_i. Define

$$j \in \operatorname*{argmin}_{0 \le i \le n} r(S_i) + A_0(Q, \sigma_1 \cdots \sigma_i) + \pi(\sigma_{i+1} \cdots \sigma_n).$$

The schedule constructed by A waits until time $r(S_j)$ and then executes S_j. To see that this is a $(1+\rho_0)$-approximation, let j^* be the index of the request having maximum release date among the requests that are served in the optimal schedule. Then $\mathrm{OPT}(Q, \sigma) \ge r_{j^*} \ge r(S_{j^*})$. Also,

$$\mathrm{OPT}(Q, \sigma) = \mathrm{OPT}(\sigma_1 \cdots \sigma_{j^*}) + \pi(\sigma_{j^*+1} \cdots \sigma_n)$$

by Lemma 4(a). Thus by definition of A,

$$\begin{aligned} A(Q, \sigma) &\le r(S_{j^*}) + A_0(Q, \sigma_1 \cdots \sigma_{j^*}) + \pi(\sigma_{j^*+1} \cdots \sigma_n) \\ &\le \mathrm{OPT}(Q, \sigma) + \rho_0 \mathrm{OPT}(Q, \sigma_1 \cdots \sigma_{j^*}) + \pi(\sigma_{j^*+1} \cdots \sigma_n) \\ &\le \mathrm{OPT}(Q, \sigma) + \rho_0 \mathrm{OPT}(Q, \sigma) \\ &= (1+\rho_0)\mathrm{OPT}(Q, \sigma). \end{aligned}$$

□

Algorithm 1 Wait and Go with Restart (WGR)

The algorithm has two states, WAIT and GO. The initial state is WAIT.

- WAIT. Wait until a time t such that $A(t) \in [t, \rho t]$, then enter state GO.
- GO. Let t_g be the time this state was entered. Return the server to the origin at full speed, then start following schedule $A_0[t_g]$; when done, return to state WAIT. Meanwhile, if a new request arrives at time t, compute $A(t)$. If $A(t) \ge t$, make a *restart*, that is, stop the server at its current location and return to state WAIT.

The intuition behind our online algorithm (Algorithm 1) is the following. In order to be competitive, the algorithm tries to guess which requests are ignored and which are served by the optimal offline server. The condition $A(t) \ge t$ is used to ensure that if a restart occurs, only a short time has elapsed, compared to the optimal cost, while if a restart does not occur, the new requests can be safely ignored.

To move from intuition to proof, we need the following important property of WGR: after the algorithm restarts, it eventually enters state GO another time.

This property is non-trivial, because $A(t)$ can be a discontinuous function of t. We formalize the property in the following lemma.

Lemma 6 *Let σ_l be a request with maximum release date among the requests served by an optimal solution. If* WGR *is in state WAIT at a time $t_w \geq r_l$ and $\mathrm{OPT}(t_w) \geq t_w$, then at some time $t_g \geq t_w$* WGR *enters state GO.*

PROOF. Consider the sequence $\sigma^{>t_w}$. Since no request in $\sigma^{>t_w}$ is served in an optimal solution for σ, by Lemma 4(a),

$$\mathrm{OPT}(t) = \mathrm{OPT}(t_w) + \pi(\sigma^{>t_w}) - \pi(\sigma^{>t})$$

for any $t \geq t_w$. Thus, for $t \geq t_w$, $\mathrm{OPT}(t)$ cannot decrease as a function of t, thus at some time t_g it must intersect the identity function, since σ has finite length. Then $\mathrm{OPT}(t_g) = t_g$, and $A(t_g) \in [t_g, \rho t_g]$.

□

We need a last ingredient in order to prove the competitiveness of WGR.

Lemma 7 *At any time t, the server controlled by* WGR *is at most at distance $t/(1 + 2/\rho)$ from the origin.*

PROOF. We prove the claim by induction on the sequence of states entered by the algorithm. The claim is trivially true at time 0. Also, by induction, it is true while the algorithm is in state WAIT since in that state the server does not move.

When the algorithm enters state GO, the server is, by inductive hypothesis, at some distance $d_g \in [0, t_g/(1 + 2/\rho)]$ from the origin. WGR first has to move the server to the origin; if it changes state before arriving, then the claim is true. Otherwise, at time $t_g + d_g$, WGR starts following schedule $A_0[t_g]$, which takes time at most $A(t_g) \leq \rho t_g$, so at any later moment $t_g + d_g + \Delta$ the server cannot be at a distance greater than $\min\{\Delta, \rho t_g/2\}$ from the origin. Now

$$\min\{\Delta, \rho t_g/2\} \leq \frac{t_g + \Delta}{1 + 2/\rho} \leq \frac{t_g + d_g + \Delta}{1 + 2/\rho},$$

which proves the lemma.

□

Theorem 8 WGR *is c-competitive for online PCTSP, where*

$$c = \rho + \frac{\rho}{1 + 2/\rho} + \rho_0.$$

PROOF. Let t_g be the last time state GO was entered (if state GO is never entered, one can use Lemmas 6 and 4(b) to show that WGR is optimal). According to Lemma 7, the online server is at distance at most $t_g/(1+2/\rho)$ from the origin. Then the algorithm will pay, for the sequence $\sigma^{\leq t_g}$, at most

$$\begin{aligned} & t_g + \frac{1}{1+2/\rho} t_g + A_0(t_g) \\ \leq\, & A(t_g) + \frac{1}{1+2/\rho} A(t_g) + \rho_0 \mathrm{OPT}(t_g) \\ \leq\, & \left(\rho + \frac{\rho}{1+2/\rho} + \rho_0 \right) \mathrm{OPT}(t_g). \end{aligned}$$

Notice that this already includes the penalties of all requests in $\sigma^{\leq t_g}$ that were not served by $A_0[t_g]$. Moreover, the request σ_l with latest release date among those served by an optimal offline solution must have been released at a time $r_l \leq t_g$, otherwise by Lemma 4(b) $\mathrm{OPT}(r_l) \geq r_l$, implying $A(r_l) \geq r_l$. Thus, WGR would have restarted and by Lemma 6 state GO would have been entered one more time, contradicting our initial assumption.

The online server also pays the penalties of all the requests released after time t_g, but notice that these penalties are also paid by the optimal solution. More specifically,

$$\mathrm{WGR}(\sigma) \leq c \cdot \mathrm{OPT}(t_g) + \pi(\sigma^{>t_g})$$

while by Lemma 4(a),

$$\mathrm{OPT}(\sigma) = \mathrm{OPT}(t_g) + \pi(\sigma^{>t_g}).$$

□

From Theorem 8 we can easily derive the following corollaries.

Corollary 9 *There is a* 7/3*-competitive online algorithm for the* PCTSP.

Corollary 10 *There is a* 34/5*-competitive polynomial time algorithm for the online* PTP.

PROOF. Follows from Theorem 8, the existence of a 2-approximation for the PTP [12], and Lemma 5.

□

Corollary 11 *There is an $O(1)$-competitive polynomial time algorithm for the* PCTSP.

PROOF. Follows from Theorem 8, the existence of an $O(1)$-approximation for the offline PCTSP (Theorem 2), and Lemma 5.

□

5 Further results and conclusions

We formulated an online version of the Prize-Collecting Traveling Salesman Problem. We gave a 7/3-competitive algorithm which is not far from best possible, the competitive ratio of any online algorithm being at least 2. We also discussed an $O(1)$-competitive algorithm running in polynomial time.

In the paper we only considered general metric spaces. For special metric spaces, the results can be improved. For example, in the case of the real halfline, it is possible to prove a lower bound of 1.89 and an upper bound of 2 on the competitive ratio of the problem [5].

Although some of our techniques are reminiscent of other works [1, 4], to our knowledge the idea of monitoring the optimal cost of the input sequence in order to decide when to restart the schedule is new and may have some uses in other online problems as well.

References

[1] N. Ascheuer, S. O. Krumke, and J. Rambau. Online dial-a-ride problems: Minimizing the completion time. In H. Reichel and S. Tison, editors, *Proc. 17th Symp. on Theoretical Aspects of Computer Science*, volume 1770 of *Lecture Notes in Computer Science*, pages 639–650. Springer-Verlag, 2000.

[2] G. Ausiello, S. Leonardi, and A. Marchetti-Spaccamela. On salesmen, repairmen, spiders, and other traveling agents. In G. Bongiovanni, G. Gambosi, and R. Petreschi, editors, *Proc. 4th Italian Conf. on Algorithms and Complexity*, volume 1767 of *Lecture Notes in Computer Science*, pages 1–16. Springer-Verlag, 2000.

[3] G. Ausiello, E. Feuerstein, S. Leonardi, L. Stougie, and M. Talamo. Algorithms for the on-line travelling salesman. *Algorithmica*, 29(4):560–581, 2001.

[4] G. Ausiello, M. Demange, L. Laura, and V. Paschos. Algorithms for the on-line quota traveling salesman problem. *Information Processing Letters*, 92(2):89–94, 2004.

[5] G. Ausiello, V. Bonifaci, and L. Laura. The online prize-collecting traveling salesman problem. Technical Report TR 08-06, Department of Computer and Systems Science, University of Rome "La Sapienza", Rome, Italy, 2006.

[6] B. Awerbuch, Y. Azar, A. Blum, and S. Vempala. New approximation guarantees for minimum-weight k-trees and prize-collecting salesmen. *SIAM Journal on Computing*, 28(1):254–262, 1998.

[7] E. Balas. The prize collecting traveling salesman problem. *Networks*, 19: 621–636, 1989.

[8] A. Blum, R. Ravi, and S. Vempala. A constant-factor approximation algorithm for the k-MST problem. In *Proc. 28th Symp. on Theory of Computing*, pages 442–448, 1996.

[9] S. Y. Cheung and A. Kumar. Efficient quorumcast routing algorithms. In *Proc. of INFOCOM '94*, volume 2, pages 840–847, 1994.

[10] M. Dell'Amico, F. Maffioli, and P. Värbrand. On prize-collecting tours and the asymmetric travelling salesman problem. *Int. Transactions in Operational Research*, 2(3):297–308, 1995.

[11] N. Garg. Saving an epsilon: a 2-approximation for the k-MST problem in graphs. In *Proc. 37th Symp. on Theory of Computing*, pages 396–402, 2005.

[12] M. Goemans and D. P. Williamson. A general approximation technique for constrained forest problems. *SIAM Journal on Computing*, 24(2): 296–317, 1995.