3-approximation Algorithm for the Travelling Repairman Problem with Unit Time-windows

Luis Eduardo Urbán Rivero[1], Cynthia A. Rodríguez Villalobos[2], Rafael López Bracho[3], Francisco Javier Zaragoza Martínez[3]

[1] UAM Azcapotzalco, Posgrado en Optimización, Mexico

[2] University of Waterloo, Department of Combinatorics and Optimization, Canada

[3] UAM Azcapotzalco, Departamento de Sistemas, Mexico

lurbanrivero@gmail.com, ca7rodri@uwaterloo.ca, rlb@correo.azc.uam.mx, franz@correo.azc.uam.mx

Abstract. The travelling repairman problem (TRP) is a scheduling problem in which a repairman must visit locations to perform some task. Each location has a time window in which the repairman is allowed to arrive. The objective of this problem is to maximize the number of tasks performed. In this paper, we consider a special case in which all the locations are on a straight line, the tasks have no processing time, and all time-windows are of unit length. We present an improvement on the analysis of the 4-approximation algorithm presented by S. L. Pérez Pérez *et al.* in 2014.

Keywords: TRP, approximation algorithm, unit time-windows.

1 Introduction

In the travelling repairman problem (TRP), we are given a set of locations the repairman can visit and the time needed to travel between any pair of locations. Each location has a set of tasks to be done by the repairman; each with a fixed processing time and a time-window during which the repairman is allowed to arrive at the location to perform that task. The objective of the problem is to find the route which maximizes the number of tasks completed by the repairman.

In 1992, J. Tsitsiklis [5] proved that deciding whether the repairman can complete k tasks within their time-windows is NP-complete, even if the tasks have no processing time.

In 2005, R. Bar-Yehuda, *et al.* [1] considered the special case in which all location are on a line, there are no processing times, and all time-windows are of unit length. They proposed an 8-approximation algorithm with running time

$O(n^2)$. This algorithm has the disadvantage of double-counting some of the tasks performed. Therefore, they also propose a $(4+\epsilon)$-approximation algorithm which avoids double-counting but has computational cost $O(n^{8/\epsilon})$. It is worth noting that the computational complexity of this special case is unknown.

In 2012, G. Frederickson and B. Wittman [2] showed that the TRP with unit time-windows on a tree is NP-hard. Also, in [2], they propose a 3-approximation algorithm for that case with running time $O(n^4)$. This result has a better approximation factor than the algorithm proposed by R. Bar-Yehuda *et al.* but has a high computational cost.

In 2014, S. L. Pérez Pérez *et al.* present a 4-approximation algorithm for the TRP with unit time-windows on a line. The algorithm is based on the algorithm of R. Bar-Yehuda *et al.* [1], and improves the computational cost to $O(n^2)$. The algorithm is also faster and simpler than the algorithm proposed by G. Frederickson and B. Wittman [2].

In this paper, we give a different analysis to the algorithm proposed by S. L. Pérez Pérez *et al.* [3] and show its approximation factor is 3. In Section 2 we formally describe the problem. In Section 3 we present the $O(n^2)$ algorithm by R. Bar-Yehuda *et al.* [1] for the TRP with unit time-windows on a line. In Section 4 we present the algorithm by S. L. Pérez Pérez *et al.* [3] and give our analysis for the algorithm improving its approximation factor.

2 Preliminaries

We define the travelling repairman problem with time windows (TRP-TW) as follows.

Definition 1. *Let (X, d) be a metric space where X is a set of points and $d : X \times X \to \mathbb{R}_+$ is a metric. We are given:*

- *a starting point $x_0 \in X$.*
- *a set of locations $V = \{x_1, x_2, \ldots, x_n\}$, where $x_i \in X$ for all $i \in \{1, \ldots, n\}$,*
- *a set of non-negative processing times $\{t_0 = 0, t_1, t_2, \ldots, t_n\}$,*
- *a set of profits $\{p_0 = 0, p_1, p_2, \ldots, p_n\}$,*
- *a set of time windows $\{[\ell_i, u_i] : 0 \leq \ell_i \leq u_i, i \in \{1, 2, \ldots, n\}\}$*

The objective is to find a route $x_0 r_1 r_2 \ldots r_k$ maximizing $\sum_{i=1}^{k} p_i r_i$ such that $r_i \in V$ for all $i \in \{1, 2, \ldots, k\}$ and $\ell_i \leq \sum_{i=1}^{j} d(r_{i-1}, r_i) + t_{i-1} \leq u_i$ for all $j \in \{1, 2, \ldots, k\}$.

The set of locations V represent tasks to be performed by the repairman. Note that a point in (X, d) can have multiple locations associated if there are multiple tasks to be performed there. The time-windows give the time span during which the repairman is allowed to arrive to a location for a task.

We consider a special case (Line-TRP-UTW) where X is a line, all processing times are 0, all profits are 1, and all time windows are unitary ($u_i - \ell_i = 1$ for all $i \in \{1, 2, \ldots, n\}$).

3 8-approximation Algorithm for Line-TRP-UTW

We describe the algorithm proposed by R. Bar-Yehuda *et al.* [1]. First, the position of each location in V and the time window of the associated task on the position vs time plane. In this representation, each task becomes a vertical line segment of unit length. Note that a curve C on the plane will be a feasible route if and only if it is monotone with respect to time and the tangents to any point on C have angle between 45 and 135 degrees.

Then, the plane is rotated clockwise 45 degrees. Because of this, the representations of the tasks become line segments of length 1 with slope of 45 degrees. and a curve C on the plane is a feasible route if and only if it is monotone with respect to both axes. The Line-TRP-UTW is then equivalent to finding a monotone curve on this plane maximizing the number of intersected line segments.

Afterwards, we overlay a (finite) grid with square size $1/\sqrt{2}$ on the plane such that it does not intersect the extreme point of any of the slanted line segments and all slanted segments lie inside the grid. Note that every slanted segment will intersect the grid at exactly one horizontal line and one vertical line. We then transform the grid to a directed acyclic graph (DAG) $\bar{G} = (V, A)$ by setting all the intersections of grid lines as vertices and orienting all the horizontal edges rightwards and all the vertical edges upwards. Finally, we associate a weight $w(a)$ to every arc $a \in A$ corresponding to the number of line segments intersecting that arc.

The size of $\bar{G}$ is polynomial in $\max\{u_i : i \in \{1, \ldots, n\}\} - \min\{u_i : i \in \{1, \ldots, n\}\}$ and $\max\{x_i, i \in \{0, 1, \ldots, n\}\} - \min\{x_i, i \in \{0, 1, \ldots, n\}\}$ but it is not strongly polynomial in the size of the input. However, it is possible to reduce the grid to a strongly polynomial size.

Theorem 1. *The DAG $\bar{G} = (V, A)$ can be reduced in polynomial time to a DAG G of size $3n \times 3n$.*

Proof. We will construct the DAG G by generating at most $3n$ columns and $3n$ rows. Consider the endpoints of the n slanted line segments given by (x_i, y_i) and $(x_i + 1, y_i + 1)$. Order the segments in increasing x-order in $O(n \log n)$ time. For each $1 \leq i \leq n$, add columns labeled $\lfloor x_i \rfloor$, $\lfloor x_i \rfloor + 1$, and $\lfloor x_i \rfloor + 2$. Repeat this process on the y coordinate, adding at most $3n$ rows, and yielding a grid of size at most $3n \times 3n$. Note that no slanted lines intersected any of the removed arcs and therefore any curve in the reduced grid can be mapped to a curve in the original grid intersecting the same number of slanted segments. □

Finally, we use the longest path algorithm on G to find a curve C over the grid lines intersecting the most slanted segments. This route is feasible for the repairman and can be computed in $O(n^2)$ time by using dynamic programming [4, pp. 661-666].

In Figure 1 we show an example of the reduced DAG G with the slanted task segments and associated arc weights (left) as well as the solution corresponding to the longest path on G.

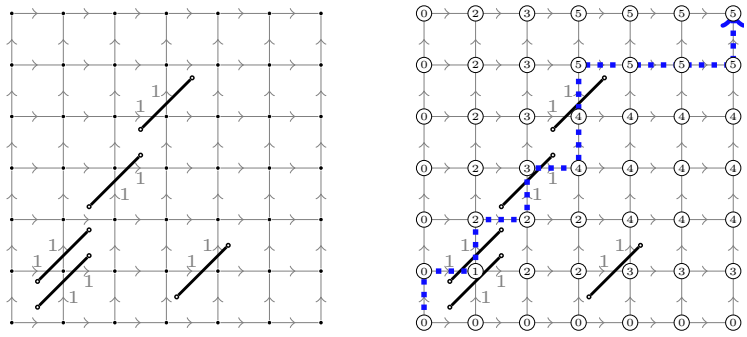

Fig. 1. Example of a reduced DAG G with the slanted segments and arc weights (left). The monotone curve attaining maximum weight on G (right).

Theorem 2 (Bar-Yehuda, Even, Shahar). *The described algorithm has approximation factor* 8.

The proof of Theorem 2 can be found in [1]

4 3-approximation Algorithm for Line-TRP-UTW

The algorithm described in Section 3 sometimes counts slanted segments twice. For example the solution of the example given in 1 gives a route of weight 5 but only intersects 3 slanted segments. S. L. Pérez Pérez *et al.* [3] proposed a modification to the algorithm which eliminates double counting, which we describe briefly.

We take the reduced DAG G and build an auxiliary DAG G' as follows:

1. We create a vertex for each horizontal arc in G and denote the set of all such vertices as V_h. Similarly, we obtain V_v from the vertical arcs in G.
2. We add an arc joining two vertices in V_h rightwards if the corresponding arcs were adjacent in G and call the set of all such arcs A_h. Analogously, we obtain A_v from joining vertices in V_v.
3. We add an arc from a vertex in V_h to a vertex in V_v if the corresponding arcs were adjacent in G and call the set of these arcs A_{hv}. Likewise, we obtain A_{vh} from arcs from a vertex in V_v to a vertex in V_h.

Then $G' = (V_h \cup V_v, A_h \cup A_v \cup A_{hv} \cup A_{vh})$. A section of G' is shown in Figure 2.

Lemma 1. *G' can be constructed in $O(n^2)$ time.*

Proof. By Theorem 1, the grid G has at most $9n^2$ vertices and every vertex has out-degree 2. Therefore, $|V(G')| = |A(G)| = 18n^2$. Every vertex in $V(G')$ also has out-degree at most 2, so $A(G') \leq 36n^2$. Assuming the creation of vertices and edges take constant time, constructing G takes $O(n^2)$ steps. □

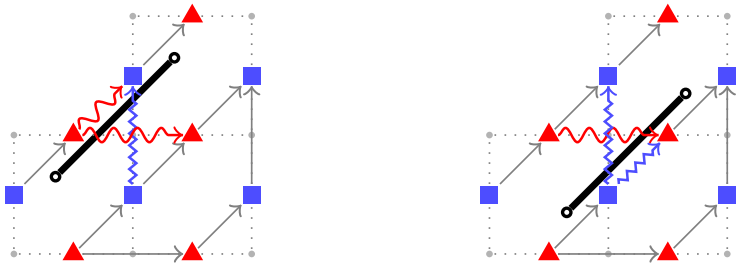

Fig. 2. A section of G': vertices of V_h are triangles and vertices of V_v are squares. The edges that get weight 1 when a double counting slanted segment intersects the original grid are highlighted.

The new algorithm then becomes:

1. We create the auxiliary DAG G' in $O(n^2)$ time.
2. We assign the appropriate weights to each arc in $A(G')$ in $O(n^2)$ time.
3. We compute the longest path in G' by dynamic programming in $O(|V(G')+A(G'))\in O(n^2)$ time.

In [3], S. L. Pérez Pérez *et al.* showed that this algorithm avoids the problem with double counting and has approximation factor 4. Its running time is $O(n^2)$.

4.1 Our Analysis of the New Algorithm

We start by giving an alternate proof that the new algorithm has approximation factor 4.

Lemma 2. *The new algorithm has approximation factor* 4.

Proof. Let P be the optimal route and let z_{OPT} be its cost. P can be covered by horizontal and vertical blocks on the grid as shown in Figure 4.1. Consider the four periodic routes r_1, r_2, r_3, r_4 show in Figure 4.1, with costs z_1, z_2, z_3, z_4 respectively, and the solution of the new algorithm A with cost z_A. Note that r_1, r_2, r_3, r_4 are all feasible paths for G' and together they intersect all edges intersected by P at least once. Therefore, $z_A \geq \max\{z_1, z_2, z_3, z_4\} \geq \frac{z_1+z_2+z_3+z_4}{4} \geq \frac{z_{\mathrm{OPT}}}{4}$ □

We use a similar idea to show that the approximation factor is at most 3.

Lemma 3. *The new algorithm has approximation factor* 3.

Proof. Let P be the optimal route and let z_{OPT} be its cost. P can be covered by horizontal and vertical blocks on the grid as shown in Figure 4.1. Consider the three periodic routes r_1, r_2, r_3 show in Figure 4.1, with costs z_1, z_2, z_3 respectively, and the solution of the new algorithm A with cost z_A. Note that r_1, r_2, r_3 are all feasible paths for G' and together they intersect all edges intersected by P at least once. Therefore, $z_A \geq \max\{z_1, z_2, z_3\} \geq \frac{z_1+z_2+z_3}{3} \geq \frac{z_{\mathrm{OPT}}}{3}$ □

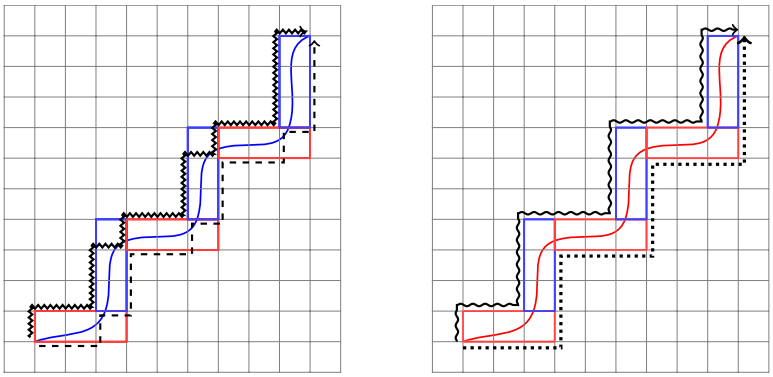

Fig. 3. Block cover of an optimal solution and the four periodic routes.

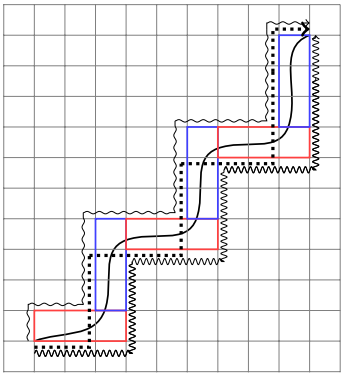

Fig. 4. Block cover of an optimal solution and the three periodic routes.

References

1. Bar-Yehuda, R., Even, G., Shahar, S.M.: On approximating a geometric prize-collecting traveling salesman problem with time windows. Journal of Algorithms 55(1), 76–92 (2005)
2. Frederickson, G.N., Wittman, B.: Approximation algorithms for the traveling repairman and speeding deliveryman problems. Algorithmica 62(3-4), 1198–1221 (2012)
3. Pérez Pérez, S.L., Urbán Rivero, L.E., López Bracho, R., Zaragoza Martínez, F.J.: A fast 4-approximation algorithm for the traveling repairman problem on a line. In: Proceedings of CCE. pp. 268–271 (2014)
4. Sedgewick, R., Wayne, K.: Algorithms. Pearson Education (2011)
5. Tsitsiklis, J.N.: Special cases of traveling salesman and repairman problems with time windows. Networks 22(3), 263–282 (1992)