A note on the traveling salesman reoptimization problem under vertex insertion

Jerome Monnot

1 CNRS UMR 7243
2 PSL, Université Paris-Dauphine, LAMSADE, France. monnot@lamsade.dauphine.fr

Abstract. We propose a $4/3$-approximation in linear time for the metric traveling salesman reoptimization problem under vertex insertion. This constitutes an improvement of the time complexity of the best known existing bound since in (G. Ausiello, B. Escoffier, J. Monnot and V. Th. Paschos: Reoptimization of minimum and maximum traveling salesman’s tours. J. Discrete Algorithms 7(4): 453-463, 2009) a $4/3$-approximation is already given with a complexity $O(n^3)$. Our algorithm is of independent interest because it does not use a matching to build the approximate solution.

Key words. Approximation Algorithms; Reoptimization under vertex insertion; TSP

1 Introduction

The traveling salesman problem (in short TSP) is one of the most popular optimization problems. It has been widely studied in the operational research community (see, for example, the books [26, 28, 30]) and consists of finding a cycle of minimum weight which spans all vertices of a complete weighted graph $(K_n, d)$ on $n$ vertices. As is well known, TSP is NP-hard and it is NP-hard to get a polynomial-time approximation within approximation ratio of $2^P(n)$ for any polynomial $P(n)$ [25]. In the metric case, i.e., when the graph weights satisfy the triangle inequality, TSP admits a $3/2$-approximation known as the famous Christofides’ heuristic [22] and there is no approximation better than $22/21$ unless $P = NP$ [29].

The need to gain a better understanding of the behavior of the optimal solutions subject to some local modification of the instance has led to the study of the field of reoptimization. The goal of the reoptimization consists in, given an instance of the problem together with its optimal solution and a locally modified instance, finding a new optimal solution of the perturbated instance. The questions raised in this context are the following. Does the old optimal solution help finding a new optimal one? How much does it help: in terms of the runtime?, in terms of the performance guarantee? Can we quickly find a good approximate solution on the perturbed instance? More formally, considering an instance $I$ of a given problem $Π$ with a known optimum OPT, and an instance $I'$ which results from a local perturbation of $I$, can the information provided by OPT be used to solve $I'$ in a more efficient way (i.e., with a lower complexity and/or with a better approximation ratio) than if this information was not available?

In general, NP-hardness of a problem implies that an optimal solution for a close instance cannot be found in polynomial time [14]. However, it is possible to develop polynomial approximation algorithms with better approximation performance ratio for the reoptimization version than for the original problem [14]. The exact meaning of local modification is defined in the reoptimization setting. For instance, two versions have been usually analyzed in the literature: adding some inputs or removing some inputs. For graph problems it is usually vertex, edge or cost modifications. Among
the reoptimization problems studied in the literature we can quote: the traveling salesman problem and its variants [1, 5, 12, 15, 7], the rural postman problem [3], the Steiner Tree problem [8, 13, 11, 24], the spanning tree problem [19], the knapsack problem [2], the shortest common superstring problem [9], several covering problems [10], some maximum weight induced hereditary subgraph problems [17, 18], several scheduling problems [32, 6, 16]. Several surveys of combinatorial reoptimization problems are given in [4, 14, 20, 34] and a recent theoretical framework capturing many different dynamic situations can be found in [33]. As indicated in these surveys, the aim is to determine if it is possible to fruitfully use our knowledge on the optimal solution in order to either (i) achieve better approximation ratios, or (ii) devise much faster approximation algorithms for the perturbed instance. In this note, we address point (ii) for the traveling salesman reoptimization problem under vertex insertion.

More precisely, we suppose that a complete graph \((K_n, d)\) of \(n\) vertices \(\{1, \ldots, n\}\) and edge-weighted by \(d\) is given and an optimum solution \(OPT_n\) of TSP for \((K_n, d)\) has already been computed. In the reoptimization problem we deal with, denoted \(TSP^+\) in the sequel, \((K_n, d)\) is transformed into \((K_{n+1}, d)\) by adding a new vertex \(n + 1\) together with all distances between \(n + 1\) and any vertex \(i\). How can we reuse the known optimum solution of TSP for \((K_n, d)\) in order to compute a good approximate solution for \((K_{n+1}, d)\)? An analogous problem denoted \(TSP^-\) consists of reoptimizing TSP when \(n + 1\) in \(K_{n+1}\) is deleted together with all edges incident to it.

1.1 Related work

The concept of reoptimization problems has been introduced by Archetti et al. [1] in 2003, where \(TSP^+\) and \(TSP^-\) have been proved \(NP\)-hard. Moreover they showed that the simple best insertion heuristic for updating the known optimum tour is a linear-time algorithm in \(O(n)\) time which produces a (tight) \(3/2\)-approximate tour for \(TSP^+\) and \(TSP^-\) in the metric case. In 2010, some improvements are proposed in Ausiello et al. [5]: a \(4/3\)-approximation for \(TSP^+\) in the metric case using a mixture of the best insertion heuristic of [1] and Christofides’ heuristic [22] given from scratch. As indicated in [5], the global complexity is dominated by Christofides’ algorithm and is \(O(n^3)\) because the algorithm computes a maximum weighted matching in a complete graph. Finally, let us note that the non-metric TSP reoptimization is as hard to approximate as the non-reoptimization non-metric TSP [5], and that for the metric case also edge cost modifications have been considered in the literature [12, 15, 7]. A survey of existing results for several versions of the TSP reoptimization can be found in [34] (Section 4.4, Table 4.2, page 20).

1.2 Notations

Given a multigraph \(G = (V, E)\), a tour \(\Gamma\) also called a Hamiltonian cycle is an elementary cycle of \(G\) which spans each vertex of \(V\). Alternately, it is a closed walk which starts and ends at the same vertex and includes every vertex once. An Eulerian cycle is a simple cycle of \(G\) which spans each edge of \(E\). Equivalently, it is a closed walk which visits every edge once. Euler’s theorem affirms that \(C\) is an Eulerian cycle iff the subgraph \(G' = (V, C)\) is connected and the degree of all its vertices is even.

Given a complete graph \((K_n, d)\) where \(V(K_n) = \{1, \ldots, n\}\) is the vertex set, the graph weights \(d\) satisfy the triangle inequality if \(\forall i, j, k \in V(K_n)\), \(d(i, j) \leq d(i, k) + d(k, j)\). For a subset \(E'\) of edges of \(G\), let \(d(E') = \sum_{e \in E'} d(e)\). In particular, if \(\Gamma = E'\) is a tour of \((K_n, d)\), then \(d(\Gamma) = \sum_{e \in E} d(e)\) is the total weight of \(\Gamma\) and \(OPT_n = d(OPT_n)\) is the value of an optimal tour \(OPT_n\). When \(d\) satisfies the triangle inequality, it is well known that, given an Eulerian cycle \(C\), we can find within \(O(\lvert C\rvert)\) time a tour \(\Gamma_C\) such that \(d(\Gamma_C) \leq d(\Gamma)\) by applying the famous shortcut method.
2 Reoptimization of the traveling salesman problem under vertex insertion

We propose an algorithm which returns the best of two different tours depending on the value of $\Delta(x, y) = d(x, n + 1) + d(y, n + 1) - d(x, y)$ where $x$ and $y$ are some vertices; Figure 1 proposes an illustration of the Algorithm Approx-TSP$^+$. Either, it is the best insertion heuristic which is applied (Step 3 of Figure 1) or it is the shortcut method applied on a particular Eulerian cycle (Step 4 of Figure 1).

**Input:** A complete graph $(K_{n+1}, d)$ and an optimal tour $OPT_n$ of $(K_n, d)$.

**Output:** An approximate tour $\Gamma$ of $(K_{n+1}, d)$.

1. Compute $\Delta^* = \Delta(\ell_1^*, \ell_2^*) = \min_{1 \leq i, j \leq n} \Delta(\ell_i, \ell_j)$ and $\delta^* = \Delta(\ell_1^*, \ell_2^*) = \min_{\ell_1^*, \ell_2^* \in OPT_n} \Delta(\ell_1^*, \ell_2^*)$ where $\Delta(\ell_1, \ell_2) = d(\ell_1, n + 1) + d(\ell_2, n + 1) - d(\ell_1, \ell_2)$.
2. Compute a minimum spanning tree $T_n$ of $(K_n, d)$ with weight $\text{mst}_n = d(T_n)$.
3. If $\text{opt}_n + \delta^* \leq 2\text{mst}_n + \Delta^*$, do
   (a) Let $\Gamma = \Gamma_1 = (OPT_n \cup \{(\ell_3^*, n + 1), (\ell_4^*, n + 1)\})$ be the tour of $(K_{n+1}, d)$.
4. Else do
   (a) Let $C_2 = T_n \cup (T_n \setminus \mu T_n(\ell_1^*, \ell_2^*)) \cup \{(\ell_3^*, n + 1), (\ell_4^*, n + 1)\}$ be the Eulerian tour formed by the edges of $T_n$ doubled except those in the unique path $\mu T_n(\ell_1^*, \ell_2^*)$ in $T_n$ from $\ell_1^*$ to $\ell_2^*$ plus the two edges $(\ell_3^*, n + 1), (\ell_4^*, n + 1)$;
   (b) Apply the shortcut method to $C_2$ to obtain a tour $\Gamma = \Gamma C_2$ of $(K_{n+1}, d)$.
5. Return the approximate tour $\Gamma$.

![Fig. 1. Algorithm Approx-TSP$^+$](image)

The running time of this algorithm is $O(n^2)$ and it is linear since the graph is complete. Actually, the time complexity of Step 1 is $O(n^2)$, the computation of $T_n^*$, can be done in $O(m + n \log n)$ time using an implementation of Prim’s algorithm with Fibonacci heaps, where $m$ is the number of edges (cf. Cormen et al., Introduction to Algorithms, 2nd ed., p.573, see [23]), while the other steps require $O(n)$ time. In a complete graph, $m = O(n^2)$; thus the Algorithm Approx-TSP$^+$ runs in linear time in $m$ if we use Fibonacci heaps.

**Theorem 1.** ApproxTSP$^+$ is a $4/3$-approximation algorithm for TSP$^+$ running in $O(n^2)$ time.

**Proof.** Let us show that the approximate tour $\Gamma$ of $(K_{n+1}, d)$ satisfies the following inequality:

$$d(\Gamma) \leq \min\{\text{opt}_n + \delta^*; 2\text{mst}_n + \Delta^*\}. \quad (1)$$

If $\text{opt}_n + \delta^* \leq 2\text{mst}_n + \Delta^*$, then by construction we have $\Gamma = \Gamma_1$ and $d(\Gamma_1) = \text{opt}_n + \delta^*$. Hence, $d(\Gamma) \leq \min\{\text{opt}_n + 2\delta^*; 2\text{mst}_n + \Delta^*\}$. Now, assume $\text{opt}_n + \delta^* > 2\text{mst}_n + \Delta^*$. So, by construction $\Gamma = \Gamma C_2$ and $d(C_2) = 2\text{mst}_n - d(\mu T_n(\ell_1^*, \ell_2^*)) + d(\ell_1^*, n + 1) + d(\ell_2^*, n + 1) \leq 2\text{mst}_n + \Delta(\ell_1^*, \ell_2^*) = 2\text{mst}_n + \Delta^*$ because by the triangle inequality we have $d(\mu T_n(\ell_1^*, \ell_2^*)) \geq d(\ell_1^*, \ell_2^*)$. Thus, by the shortcut method, we have $d(\Gamma) = d(\Gamma C_2) \leq d(C_2) \leq 2\text{mst}_n + \Delta^* = \min\{\text{opt}_n + 2\delta^*; 2\text{mst}_n + \Delta^*\}$.

Let us focus first on the quantity $\text{opt}_n + \delta^*$.

Let $OPT_{n+1}$ be an optimal tour of $(K_{n+1}, d)$ with value $OPT_{n+1}$ and denote by $i, j$ the two neighbors of $n + 1$ in $OPT_{n+1}$. Now, let $(i, i_1) \in OPT_n$ and $(j, j_1) \in OPT_n$. Using triangle
inequality $\delta^* \leq \Delta(i, i_t) \leq 2d(i, n + 1)$ and $\delta^* \leq \Delta(j, j_t) \leq 2d(j, n + 1)$. Thus, $\delta^* \leq d(i, n + 1) + d(j, n + 1)$. Since $(\text{OPT}_{n+1} \setminus \{(i, n + 1), (j, n + 1)\}) \cup \{(i, j)\}$ is a tour of $(K_n, d)$, we obtain $\text{opt}_n + d(i, n + 1) + d(j, n + 1) \leq \text{opt}_{n+1} + d(i, j)$. Hence:

$$\text{opt}_n + \delta^* \leq \text{opt}_{n+1} + d(i, j).$$

Now, let us focus on the second quantity $2\text{mst}_n + \Delta^*$.

Observe that $(\text{OPT}_{n+1} \setminus \{(i, n + 1), (j, n + 1)\})$ is a spanning tree of $(K_n, d)$. Hence $\text{mst}_n \leq \text{opt}_{n+1} - (d(i, n + 1) + d(j, n + 1)) \leq \text{opt}_{n+1} - d(i, j)$. We deduce:

$$2\text{mst}_n \leq 2\text{opt}_{n+1} - (d(i, n + 1) + d(j, n + 1) + d(i, j)).$$

Moreover, by construction of $\Delta^*$ we obtain:

$$\Delta^* \leq \Delta(i, j) = d(i, n + 1) + d(j, n + 1) - d(i, j).$$

Adding inequality (3) to inequality (4), we deduce:

$$2\text{mst}_n + \Delta^* \leq 2\text{opt}_{n+1} - 2d(i, j).$$

In conclusion, we obtain:

$$d(T) \leq \min\{\text{opt}_n + \delta^*; 2\text{mst}_n + \Delta^*\} \leq \frac{2}{3}(\text{opt}_n + \delta^*) + \frac{1}{3}(2\text{mst}_n + \Delta^*)$$

$$\leq \frac{2}{3}(\text{opt}_{n+1} + d(i, j)) + \frac{1}{3}(2\text{opt}_{n+1} - 2d(i, j)) \quad \text{(by (2) and (5))}$$

$$\leq \frac{2}{3} + \frac{2}{3}\text{opt}_{n+1} = \frac{4}{3}\text{opt}_{n+1}.$$

Let us conclude this article by some remarks concerning the improvement of the running time of Algorithm Approx-TSP$^+$ in some particular cases:

- The time complexity of Approx-TSP$^+$ strongly depends on the running time of the fastest algorithm for finding a minimum spanning tree in a complete graph (MST in short). A possible approach to improve it consists of using a linear time $(1 + \varepsilon)$-approximation algorithm for MST; suppose $T$ is a spanning tree of $(K_n, d)$ built in linear time in $n$ (or sublinear in $n$) and such that $d(T) \leq (1 + \varepsilon)\text{mst}_n$ for some $\varepsilon > 0$. Replace $T_n^*$ by $T$ everywhere in Algorithm Approx-TSP$^+$ (see Figure 1). Change also condition $\text{opt}_n + \delta^* \leq 2\text{mst}_n + \Delta^*$ of Step 3 for $\text{opt}_n + \delta^* \leq 2d(T) + (1 + \varepsilon)\Delta^*$. We can prove that the new algorithm is $(1 + (1 + 2\varepsilon)(1 - \frac{2(1+\varepsilon)}{1+2\varepsilon}))$-approximate for TSP$^+$ and its running time is linear in $n$ (under the assumption that $\Delta^*$ is given in linear or constant time; for instance with the initial instance and OPT$_n$). In particular, if $\varepsilon < 1/2$, then we obtain an approximation ratio strictly better than $3/2$, which is the best ratio known so far for approximating TSP$^+$ under linear time in $n$ [1]. Unfortunately, this approach were only used to approximate MST in a probabilistic way [21].

- On the other hand, if we do not suppose that $\Delta^*$ is given with the initial instance, we need to consume $O(n^2)$ time to compute it. We can reduce this time by computing $\min_{\ell_1, \ell_2 \in F} \{d(\ell_1, n + 1) + d(\ell_2, n + 1) - d(\mu_{T_n^*}(\ell_1), \ell_2)\}$ where $F$ are the set of leafs of $T_n^*$ and $\mu_{T_n^*}(\ell_1, \ell_2)$ is the unique path in $T_n^*$ between $\ell_1$ and $\ell_2$. It is equivalent to compute the diameter of $T_n^*$ between leaves where we modify the distance from the leafs to their fathers (i.e., $d'(\ell, p(\ell)) = d(\ell, p(\ell) = \frac{2}{3}d' + \frac{1}{3}d,$ where $d'$ is the distance from the leafs to their fathers and $d$ is the initial distance).
\[ d(\ell, p(\ell)) - d(\ell, n + 1) \] where \( p(\ell) \) is the father of the leaf \( \ell \) in the tree) and then it is linear in \( n \) using dynamic programming [27]. Assume that this quantity is reached for leaves \( \ell^*_1 \) and \( \ell^*_2 \). Thus, by taking the minimum between \( d(\Gamma^*_1) \) and \( d(\Gamma^*_2) \), we maintain the approximation ratio of \( 4/3 \). Moreover, this algorithm is linear in \( n \) if and only if MST is solvable in linear time (in \( n \) where we recall that the graph is complete). However, the existence of a linear time algorithm for MST is still open [31].

Acknowledgements

I would like to thank the anonymous reviewers for their valuable comments and suggestions.

References