40 Prize-Collecting Traveling Salesman and Related Problems

Giorgio Ausiello
University of Rome "La Sapienza"

Vincenzo Bonifaci
University of Rome "La Sapienza"

Stefano Leonardi
University of Rome "La Sapienza"

Alberto Marchetti-Spaccamela
University of Rome "La Sapienza"

40.1 Introduction

The most general version of the Prize-Collecting Traveling Salesman Problem (PCTSP) was first introduced by Balas [1]. In this problem, a salesman has to collect a certain amount of prizes (the *quota*) by visiting cities. A known prize can be collected in every city. Furthermore, by not visiting a city, the salesman incurs a pecuniary *penalty*. The goal is to minimize the total travel distance plus the total penalty, while starting from a given city and collecting the quota.

The problem generalizes both the Quota TSP, which is obtained when all the penalties are set to zero, and the Penalty TSP (PTSP) (sometimes unfortunately also called PCTSP), in which there is no required quota, only penalties. A special case of the Quota TSP is the *k*-TSP, in which all prizes are unitary (*k* is the quota). The *k*-TSP is strongly tied to the problem of finding a tree of minimum cost spanning any *k* vertices in a graph, called the *k*-Minimum Spanning Tree (*k*-MST) problem.

The *k*-MST and the *k*-TSP are NP-hard. They have been the subject of several studies for good approximation algorithms [2–6]. A 2-approximation scheme for both the *k*-MST and the *k*-TSP given by Garg [6] is the best known approximation ratio. Interestingly enough, all these algorithms use the primal-dual algorithm of Goemans and Williamson [7] for the Prize-Collecting Steiner Tree as a subroutine.

The Quota TSP was also considered by some researchers. It was considered by Awerbuch et al. [8], who gave an $O(\log^2(\min(Q, n)))$ approximation algorithm for instances with n cities and quota Q. Ausiello et al. [9] give an algorithm with an approximation ratio of 5 for what could be called the Quota MST, by extending some of the ideas of Ref. [5].

The PTSP has apparently a better approximation ratio. Goemans and Williamson [7], as an application of their primal-dual technique, give an approximation algorithm with a ratio of 2 for the PTSP. Their algorithm uses a reduction to the Prize-Collecting Steiner Tree Problem. The running time of the algorithm was reduced from $O(n^2 \log n)$ to $O(n^2)$ by Gabow and Pettie [10].

We finally remark that both the MST and the TSP also admit a *budget* version; in these cases a budget B is specified in the input and the objective is to find the largest k-MST, respectively the k-TSP, whose cost is no more than the given budget.

Awerbuch et al. [8] in the aforementioned work, were the first to give an approximation algorithm for the general PCTSP. Their approximation ratio is again $O(\log^2(\min(Q, n)))$. They achieve this ratio by concatenating the tour found by the Quota TSP algorithm to a tour found by the Goemans–Williamson algorithm. As an application of the 5-approximate algorithm for the Quota MST [9] it follows that the approximation ratio of the PCTSP is constant.

In what follows, we introduce formal details and we provide a review of the main results in the area. In Section 40.2, we present the algorithm by Goemans and Williamson for the Prize-Collecting Steiner Tree, which is a basic building block for all the other algorithms in this chapter and has also an application to the PTSP. In Section 40.3, we analyze Garg's technique showing a 5-approximation for both the k-MST and k-TSP and show how to extend it to the Quota TSP. In Section 40.4, we describe an algorithm for the general PCTSP that builds over the algorithms for Quota TSP and PTSP. In Section 40.5, we show some applications of these algorithms to the minimum latency problem and graph searching.

40.2 The Prize-Collecting Steiner Tree Problem and Penalty Traveling Salesman Problem

40.2.1 Definitions

Prize-Collecting Steiner Tree. Given an undirected graph $G = (V, E)$ with vertex penalties $\pi : V \to \mathbb{Q}^+$, edge costs $c : E \to \mathbb{Q}^+$ and a root node r, the Prize-Collecting Steiner Tree problem asks to find a tree $T = (V_T, E_T)$ including r that minimizes

$$c(T) = \sum_{e \in E_T} c_e + \sum_{v \in V \setminus V_T} \pi_v$$

Penalty Traveling Salesman Problem. Given an undirected graph $G = (V, E)$ with vertex penalties $\pi : V \to \mathbb{Q}^+$, edge costs $c : E \to \mathbb{Q}^+$ satisfying the triangle inequality and a root node r, the PTSP asks to find a tour $T = (V_T, E_T)$ including r that minimizes

$$c(T) = \sum_{e \in E_T} c_e + \sum_{v \in V \setminus V_T} \pi_v$$

40.2.2 History of the Results

The first approximation algorithms for the Prize-Collecting Steiner Tree and PTSP were developed by Bienstock et al. [11]. They gave LP-based algorithms achieving a 3-approximation for the PCST and a 5/2-approximation for the PTSP with triangle inequality. Both bounds were later improved to 2 by Goemans and Williamson [7] with a combinatorial algorithm. The NP-hardness (more precisely, APX-hardness) of the problems follows from that of the Steiner Tree problem [12] and the TSP [13], respectively.

40.2.3 The Primal-Dual Algorithm of Goemans and Williamson

We review the algorithm of Goemans and Williamson [7] for the *Prize-Collecting Steiner Tree* and *PTSP*. We are given an undirected graph $G = (V, E)$, nonnegative edge costs c_e, and nonnegative vertex penalties π_i. The goal in the Prize-Collecting Steiner Tree problem is to minimize the total cost of a Steiner tree and the penalties of the vertices that are not spanned by the Steiner tree. In the PTSP, we aim to minimize the cost of a tour and of the penalties of the vertices that are not included in the tour. In this section we revise the primal-dual algorithm of Ref. [7] that provides a 2-approximation for both problems. We first show a 2-approximation for the Steiner tree version and then show how to obtain a 2-approximation for the TSP version. Subsequently, we use `GW` (Goemans-Williamson) to refer to this algorithm.

`GW` is a *primal-dual* algorithm, that is, the algorithm constructs both a feasible and integral primal and a feasible dual solution for a linear programming formulation of the problem and its dual, respectively. We will consider the version of the problem in which a root vertex r is given and the Steiner tree or the traveling salesman tour will contain r. This is without loss of generality if we can run the algorithm for all possible choices of r.

An integer programming formulation for the Steiner tree problem has a binary variable x_e for all edges $e \in E$: x_e has value 1 if edge e is part of the resulting forest and 0 otherwise. Let us denote by $\mathscr{S}$ all subsets of $V/\{r\}$. The integer programming formulation has a binary variable x_U for each set $U \in \mathscr{S}$. The cost of set U is $\sum_{v \in U} \pi_v$. For a subset $U \in \mathscr{S}$ we define $\delta(U)$ to be the set of all edges that have exactly one endpoint in U. Let T be the set of edges with $x_e = 1$ and let A be the union of sets U for which $x_U = 1$. For any $U \in \mathscr{S}$, any feasible solution must *cross* U at least once, that is, $|\delta(U) \cap T| \geq 1$, or U must be included in the set of vertices that are not spanned, that is, $U \subset A$.

This gives rise to the following integer programming formulation for the Prize-Collecting Steiner Tree problem:

$$\begin{aligned} \texttt{opt}_{\mathrm{IP}} = \min \quad & \sum_{e \in E} c_e \cdot x_e + \sum_{U \in \mathscr{S}} x_U \cdot \sum_{i \in U} \pi_i && \text{(IP)} \\ \text{s.t.} \quad & \sum_{e \in \delta(U)} x_e + \sum_{U' : U \subseteq U'} x_{U'} \geq 1 \quad \forall U \in \mathscr{S} && (40.1) \\ & x_e, x_U \in \{0, 1\} \quad \forall e \in E, \forall U \in \mathscr{S} \end{aligned}$$

It is easy to observe that any solution to the Prize-Collecting Steiner Tree problem is an integral solution of this integer linear program: We set $x_e = 1$ for all edges of the Steiner tree T that connects the spanned vertices to the root r. We set $x_U = 1$ for the set U of vertices not spanned by the tree. In the linear programming relaxation of IP we drop the integrality constraints on variables x_e and x_U.

The dual (D) of the linear programming relaxation (LP) of (IP) has a variable y_U for all sets $U \in \mathscr{S}$. There is a constraint for each edge $e \in E$ that limits the total dual assigned to sets $U \in \mathscr{S}$ that contain exactly one endpoint of e to be at most the cost c_e of the edge. There is a constraint for every $U \in \mathscr{S}$ that limits the total dual from subsets of U by at most the total penalty from vertices in U.

$$\begin{aligned} \texttt{opt}_{\mathrm{D}} = \max \quad & \sum_{U \in \mathscr{S}} y_U && \text{(D)} \\ \text{s.t.} \quad & \sum_{U \in \mathscr{S} : e \in \delta(U)} y_U \leq c_e \quad \forall e \in E && (40.2) \\ & \sum_{U' \subseteq U} y_{U'} \leq \sum_{i \in U} \pi_i \quad \forall U \in \mathscr{S} && (40.3) \\ & y_U \geq 0 \quad \forall U \in \mathscr{S} \end{aligned}$$

Algorithm `GW` constructs a primal solution for (IP) and a dual solution for D. The algorithm starts with an infeasible primal solution and reduces the degree of infeasibility as it progresses. At the same time, it creates a dual feasible packing of sets of largest possible total value. The algorithm raises dual variables of certain subsets of vertices. The final dual solution is maximal in the sense that no single set can be raised without violating a constraint of type (40.2) or (40.3).

We can think of an execution of `GW` as a process over time. Let x^τ and y^τ, respectively, be the primal incidence vector and feasible dual solution at time τ. Initially, $x_e^0 = 0$ for all $e \in E$, $x_U^0 = 0$ for all $U \in \mathscr{S}$, and $y_U^0 = 0$ for all $U \in \mathscr{S}$. In the following we say that an edge $e \in E$ is *tight* if the corresponding constraint (40.2) holds with equality, and that a set U is tight if the corresponding constraint (40.3) holds with equality. We use F^τ to denote the forest formed by the collection of tight edges corresponding to x^τ, and and $\mathcal{S}^\tau$ to denote the collection of tight sets corresponding to x^τ. Assume that the solution x^τ at time τ is infeasible. A set $U \in \mathcal{S}$ is *active* at time τ if it is spanned by a connected component in forest F^τ, there is no tight edge $e \in \delta(U)$, and the corresponding constraint (40.3) is not tight. Let $\mathcal{A}^\tau$ be the collection of sets that are active at time τ. GW raises the dual variables for all sets in $\mathcal{A}^\tau$ uniformly at all times $\tau \geq 0$.

Two kind of events are possible. (i) An edge $e \in \delta(U)$ becomes tight for a set $U \in \mathcal{A}^\tau$. We set $x_e = 1$ and update $\mathcal{A}^\tau$. (Observe that if the tight edge connects U to a component containing r, the newly formed connected component of F^τ is part of $\mathcal{A}^\tau$. If the tight edge connects set U to a component of F^τ that does not contain the root (either active or inactive), the newly formed component is part of $\mathcal{A}^\tau$.) (ii) Constraint (40.3) becomes tight for a set $U \in \mathcal{A}^\tau$. We set $x_U = 1$. GW ends with a reverse pruning phase applied to all sets $U \in \mathcal{S}$ with $x_U = 1$. They are analyzed in order of decreasing time at which the corresponding constraint (40.3) became tight. If the vertices of U are actually connected to the root via tight edges in the current solution, we set $x_U = 0$. The algorithm ends with a tree T connecting a set of vertices to the root r and a set A of vertices for which the penalties are paid. Denote by $c(T, A)$ the total cost of the solution, that is, $c(T, A) = \sum_{e\in T} c_e + \sum_{i\in A} \pi_i$.

Theorem 40.1 (Goemans and Williamson [7])

Suppose that algorithm GW *outputs a tree T, a set of vertices A and a feasible dual solution $\{y_U\}_{U\in\mathcal{S}}$. Then*

$$c(T, A) \leq 2 \cdot \sum_{U\in\mathcal{S}} y_U \leq 2 \cdot \mathtt{opt}$$

where opt *is the minimum-cost solution for the Prize-Collecting Steiner Tree problem.*

The proof of the above theorem [7] is along the following lines. The total dual of subsets of $U \in \mathcal{S}$ with $x_U = 1$ will pay for the penalties of vertices in A, that is, $\sum_{U:x_U=1} \sum_{U'\subseteq U} y_{U'} \leq \sum_{i\in A} \pi_i$, since constraints (40.3) are tight for these sets. Due to the reverse pruning phase, subsets of $U \in \mathcal{S}$ with $x_U = 1$ do not contribute to make tight any edge in T. The cost of T is then paid by twice the total dual of sets $U \in \mathcal{S}$ loading edges of T, that is, those that contribute to making the corresponding constraints (40.2) tight.

GW also provides a 2-approximation for the PTSP when edge costs obey the triangle inequality. First, we run the PCST algorithm with halved penalties. Then, the resulting tree is converted to a tour by doubling every edge and shortcutting the resulting Eulerian tour. For the proof of 2-approximation, we observe that the following integer linear program is a formulation for the problem.

$$\begin{aligned}
\mathtt{opt}_{\mathrm{IP}} = \min\ & \sum_{e\in E} c_e \cdot x_e + \sum_{U\in\mathcal{S}} x_U \sum_{i\in U} \frac{\pi_i}{2} && \text{(IP)}\\
\text{s.t.}\ & \sum_{e\in\delta(U)} x_e + \sum_{U':U\subseteq U'} x_{U'} \geq 2, \quad \forall U \in \mathcal{S} && (40.4)\\
& x_e \in \{0, 1\}, \quad \forall e \in E\\
& x_U \in \{0, 2\}, \quad \forall U \in \mathcal{S}
\end{aligned}$$

Constraints (40.4) impose that each subset must me crossed at least twice unless we pay the penalties for all the vertices of the subset. The dual of the corresponding relaxation is

$$\begin{aligned}
\mathtt{opt}_{\mathrm{D}} = \max\ & 2 \cdot \sum_{U\in\mathcal{S}} y_U && \text{(D)}\\
\text{s.t.}\ & \sum_{U\in\mathcal{S}\,:\,e\in\delta(U)} y_U \leq c_e \quad \forall e \in E && (40.5)\\
& \sum_{U'\subseteq U} y_{U'} \leq \sum_{i\in U} \frac{\pi_i}{2} \quad \forall U \in \mathcal{S} && (40.6)\\
& y_U \geq 0 \quad \forall U \in \mathcal{S}
\end{aligned}$$

The cost of the solution provided by `GW` is given by at most twice the cost of tree T and the total penalty of vertices in A. Since twice the total dual collected by the algorithm is a lower bound to the optimal solution, we conclude from the following theorem.

Theorem 40.2 (**Goemans and Williamson** [7])
Suppose that algorithm `GW` *outputs a cycle C, a set of vertices A and a feasible dual solution $\{y_U\}_{U\in\mathscr{S}}$. Then*

$$c(C, A) \leq 2 \cdot \left(2 \cdot \sum_{U\in\mathscr{S}} y_U\right) \leq 2 \cdot \texttt{opt}$$

where `opt` *is the minimum-cost solution for the PTSP.*

40.3 The *k*-Minimum Spanning Tree, *k*-TSP and Quota Traveling Salesman Problem

40.3.1 Definitions

k-Minimum Spanning Tree Problem. Given an undirected graph $G = (V, E)$, a tree on G spanning exactly k nodes is called a *k-tree.* Given such a graph with edge costs $c : E \to \mathbb{Q}^+$ and a positive integer k, the (*unrooted*) k-MST problem asks to find a k-tree of minimum total cost. In the *rooted* version of the problem, the k-tree has to include a given root node r.

k-Traveling Salesman Problem. Given an undirected graph $G = (V, E)$, a cycle of G spanning exactly k nodes is called a *k-tour.* Given such a graph with edge costs $c : E \to \mathbb{Q}^+$ satisfying the triangle inequality, a positive integer k and a root node r, the k-TSP asks to find a k-tour including r of minimum total cost.

Quota Traveling Salesman Problem. Given an undirected graph $G = (V, E)$ with vertex weights $w : V \to \mathbb{Z}^+$ and a nonnegative integer Q, a cycle C of G such that $\sum_{v\in C} w(v) \geq Q$ is called a *quota Q-tour.* Given such a graph with edge costs $c : E \to \mathbb{Q}^+$ satisfying the triangle inequality and a root node r, the Quota TSP asks to find a quota Q-tour including r of minimum total cost.

40.3.2 History of the Results

The k-MST problem is known to be an NP-hard problem [14]. Heuristics were given by Cheung and Kumar [15], who studied the problem in the context of communication networks. The first approximation algorithms were considered by Ravi et al. [16], who gave an algorithm achieving an approximation ratio of $O(\sqrt{k})$. Later, this ratio was improved to $O(\log^2 k)$ by Awerbuch et al. [8]. The first constant-ratio algorithm was given by Blum et al. [17]. Subsequently, Garg [5] gave a simple 5-approximation algorithm and a more complicated 3-approximation algorithm, while a 2.5-approximation algorithm for the unrooted case was found by Arya and Ramesh [3]. Arora and Karakostas gave a $(2 + \varepsilon)$-approximation scheme for the rooted version. A 2-approximation by Garg [6] is the current best bound.

We observe that the rooted and the unrooted versions of the k-MST are equivalent with respect to the approximation ratio. In fact, given a c-approximation algorithm for the rooted case it is sufficient to run n times the k-MST algorithm with all possible choices for the root node r, and return the cheapest k-tree found to obtain a c-approximation algorithm for the rooted case . Garg [6] observed that a c-approximation algorithm for the unrooted case gives a c-approximation algorithm for the rooted case.

Some of these works also addressed the k-TSP and Quota TSP. The algorithms for the k-MST by Garg, as well as the scheme by Arora and Karakostas, extend to the k-TSP, thus giving a 2-approximation algorithm for this problem as the current best bound. Finally, as an application of their $O(\log^2 k)$-approximation algorithm for k-MST, Awerbuch et al. give a $O(\log^2(\min(Q, n)))$-approximation algorithm for the Quota TSP, where n is the number of nodes of the graph.

Finally, we remark that a dual version of the TSP is known as the *orienteering problem* [18]. In this problem we are given an edge-weighted graph and a budget and the goal consists in visiting as many vertices of the graph as possible and return to the origin without incurring in a cost greater than the allowed budget. In Ref. [19] a 4-approximation algorithm that makes use of min-cost path algorithms of Ref. [20] is presented for the orienteering problem. In Ref. [21] an improved algorithm leading to a 3-approximation bound for the problem is shown in the context of a more general approach to constrained vehicle routing problem. Other budget versions have been defined also for MST and for TSP [6,19,22].

40.3.3 A 5-Approximation Algorithm for *k*-Minimum Spanning Tree and *k*-Traveling Salesman Problem

In this section, we present and discuss the algorithm by Garg achieving a 5-approximation for the rooted *k*-MST and its modification yielding the same approximation for the *k*-TSP. Our analysis follows Chudak et al. [23].

Several assumptions can be made with no loss of generality. First, we can suppose that the edge costs satisfy the triangle inequality, by using well-known techniques [16]. Also, we will assume that the distance from the root to the farthest vertex is a lower bound on the optimum value. It turns out that this is easy to ensure: We can run the algorithm $n-1$ times with all possible choices of a "farthest" vertex, every time disregarding nodes farther than the chosen one, and return the best solution found. The last assumption is that $\mathtt{opt} \geq c_0$, where c_0 is the smallest nonzero edge cost. This is not the case only if $\mathtt{opt} = 0$, meaning that the optimal solution is a connected component containing r of size k in the graph of zero-cost edges, and the existence of such a component can be easily checked in a preprocessing phase.

A possible formulation of the rooted *k*-MST as an integer linear problem is the following:

$$\begin{aligned}
\mathtt{opt} = \min \quad & \sum_{e\in E} c_e x_e && \text{(IP)}\\
\text{subject to:} \quad & \sum_{e\in\delta(S)} x_e + \sum_{T:T\supseteq S} z_T \geq 1 \quad \forall S \subseteq V\setminus\{r\} && (1)\\
& \sum_{S:S\subseteq V\setminus\{r\}} |S| z_S \leq n-k && (2)\\
& x_e \in \{0,1\} \quad \forall e \in E\\
& z_S \in \{0,1\} \quad \forall S \subseteq V\setminus\{r\}
\end{aligned}$$

In the above formulation, r is the root node and $\delta(S)$ the set of edges with exactly one endpoint in S. The variables x_e indicate whether the edge e is included in the tree; the variables z_S indicate whether the set of vertices S is not spanned by the tree. The set of constraints (1) enforces, for each $S \subseteq V\setminus\{r\}$, either some edge of $\delta(S)$ is in the tree or all the vertices in S are not spanned by the tree. Thus, every vertex not in any S such that $z_S = 1$ will be connected to the root r. Constraint (2) enforces at least k vertices to be spanned. Finally, the LP relaxation of this integer program is obtained by replacing the integrality constraints with nonnegativity constraints (in an optimal solution, $x_e \leq 1$ and $z_S \leq 1$ for all e and S).

All the proposed constant approximation algorithms for the *k*-MST problem use as a subroutine the primal-dual 2-approximation algorithm for the Prize-Collecting Steiner Tree of Goemans and Williamson [7]. This is not by chance, because this problem is essentially the Lagrangean relaxation of the *k*-MST. Indeed, if we apply Lagrangean relaxation to constraint (2) of the LP relaxation of the *k*-MST program, we obtain the following:

$$\begin{aligned}
\min \quad & \sum_{e\in E} c_e x_e + \lambda\left(\sum\nolimits_{S\subseteq V\setminus\{r\}} |S| z_S - (n-k)\right) && \text{(LR)}\\
\text{subject to:} \quad & \sum_{e\in\delta(S)} x_e + \sum_{T:T\supseteq S} z_T \geq 1 \quad \forall S \subseteq V\setminus\{r\}\\
& x_e \geq 0 \quad \forall e \in E\\
& z_S \geq 0 \quad \forall S \subseteq V\setminus\{r\}
\end{aligned}$$

where $\lambda \geq 0$ is the Lagrangean variable. Apart from the constant term $-\lambda(n-k)$ in the objective function, this is the same as the LP relaxation of the Prize-Collecting Steiner Tree problem with $\pi_v = \lambda$ for all v. Moreover, any solution feasible for the LP relaxation of k-MST is also feasible for (LR), so the value of this program is a lower bound on the cost of an optimal k-MST.

Before discussing Garg's algorithm, we recall that the primal-dual approximation algorithm for the Prize-Collecting Steiner Tree returns a solution (F, A), where F is a tree including the root r, and A is the set of vertices not spanned by F. The algorithm also constructs a feasible solution y for the dual of the LP relaxation of PCST.

Theorem 40.3 (Goemans and Williamson [7])

The primal solution (F, A) and the dual solution y produced by the prize-collecting algorithm satisfy

$$\sum_{e\in F} c_e + \left(2 - \frac{1}{n-1}\right)\pi(A) \leq \left(2 - \frac{1}{n-1}\right) \sum_{S\subseteq V\setminus\{r\}} y_S$$

where $\pi(A) = \sum_{v\in A} \pi_v$.

A corollary of Theorem 40.3 is that the prize-collecting algorithm has an approximation ratio of 2, by weak duality and the feasibility of y.

We would like to use the prize-collecting algorithm to solve the k-MST problem. Thus suppose that we run the algorithm with $\pi_v = \lambda$ for all $v \in V$, for some value $\lambda \geq 0$. Then by Theorem 40.3, we obtain (F, A) and y such that

$$\sum_{e\in F} c_e + 2|A|\lambda \leq 2 \sum_{S\subseteq V\setminus\{r\}} y_S \tag{40.7}$$

Consider the dual of the Lagrangean relaxation of the k-MST LP:

$$\begin{aligned}
\max \quad & \sum_{S\subseteq V\setminus\{r\}} y_S - (n-k)\lambda && \text{(LR-D)}\\
\text{subject to:} \quad & \sum_{S: e\in\delta(S)} y_S \leq c_e \quad \forall e \in E \\
& \sum_{T: T\subseteq S} y_T \leq |S|\lambda \quad \forall S \subseteq V \setminus \{r\} \\
& y_S \geq 0 \quad \forall S \subseteq V \setminus \{r\}
\end{aligned}$$

Since this dual is, apart from the objective function, the same as the dual of the LP relaxation of the PCST instance, the solution y is feasible for this dual, and the value of the objective function is a lower bound on the cost of an optimal k-MST, by weak duality. Subtracting $2(n-k)\lambda$ from both sides of Eq. (40.7)

$$\sum_{e\in F} c_e + 2\lambda\big(|A| - (n-k)\big) \leq 2\left(\sum_{S\subseteq V\setminus\{r\}} y_S - (n-k)\lambda\right) \leq 2\,\texttt{opt}$$

where opt is the cost of an optimal solution to the k-MST instance.

Now if the term $|A| - (n-k)$ is zero, we can conclude that the tree F is a k-tree and has cost no more than twice optimal. Unfortunately, if $|A| - (n-k)$ is positive then the tree F is not feasible, while if it is negative we cannot conclude anything about the approximation ratio. However, it turns out that it is possible to find values of λ such that even if these cases occur, they can be taken care of, although at the cost of resulting in an approximation ratio higher than two.

What Garg's algorithm does is indeed a binary search for these critical values of λ, through a sequence of calls to the prize-collecting algorithm. Notice that if the prize-collecting algorithm is called with $\lambda = 0$, it will return the empty tree spanning only r as a solution, while for $\lambda = \sum_{e\in E} c_e$ it will return a tree spanning all vertices. Thus the initial interval of the binary search will be $[0, \sum_{e\in E} c_e]$, and at every iteration, if the current interval is $[\lambda_1, \lambda_2]$, the prize-collecting algorithm is run with $\lambda = \frac{1}{2}(\lambda_1 + \lambda_2)$. If the returned tree has less than k vertices, we update λ_1 to λ; if it has more than k vertices, we update λ_2 to

λ. Notice that in the lucky event that, at any point, a tree with exactly k vertices is returned, we can stop, since by the above discussion that must be within a factor 2 of optimal. So assume that this event does not happen. We will stop when we have found two values λ_1, λ_2 such that:

(1) $\lambda_2 - \lambda_1 \leq \frac{c_0}{2n(n+1)}$ (recall that c_0 is the smallest nonzero edge cost);
(2) for $i = 1, 2$, the prize-collecting algorithm run with λ set to λ_i returns a primal solution (F_i, A_i) spanning k_i vertices and a dual solution $y^{(i)}$, with $k_1 < k < k_2$.

Note that these two values will be found at most after $O(\log \frac{n^2 \sum_e c_e}{c_0})$ calls to the prize-collecting algorithm. The final step of the algorithm is combining the two solutions (F_1, A_1) and (F_2, A_2) into a single k-tree. Solution (F_1, A_1) is within a factor of 2 of optimal, but infeasible, while solution (F_2, A_2) can be easily made feasible but not within a factor of 2 of optimal. More precisely, as a consequence of Theorem 40.3,

$$\sum_{e \in F_1} c_e \leq \left(2 - \frac{1}{n}\right)\left(\sum_{S \subseteq V \setminus \{r\}} y_S^{(1)} - |A_1|\lambda_1\right)$$
$$\sum_{e \in F_2} c_e \leq \left(2 - \frac{1}{n}\right)\left(\sum_{S \subseteq V \setminus \{r\}} y_S^{(2)} - |A_2|\lambda_2\right)$$

To get a bound on the cost of F_1 and F_2 in terms of opt, let

$$\alpha_1 = \frac{n - k - |A_2|}{|A_1| - |A_2|} \text{ and } \alpha_2 = \frac{|A_1| - (n - k)}{|A_1| - |A_2|}.$$

Then $\alpha_1|A_1| + \alpha_2|A_2| = n - k$ and $\alpha_1 + \alpha_2 = 1$, and after defining, for all $S \subseteq V \setminus \{r\}$, $y_S = \alpha_1 y_S^{(1)} + \alpha_2 y_S^{(2)}$, it is possible to prove the following lemma.

Lemma 40.1 (Chudak et al. [23])

$$\alpha_1 \sum_{e \in F_1} c_e + \alpha_2 \sum_{e \in F_2} c_e < 2\texttt{opt}$$

Proof
We omit the proof for brevity; the reader can find it in the overview by Chudak et al. [23]. □

We now show how to obtain a 5-approximation algorithm by choosing one of two solutions. First, if $\alpha_2 \geq \frac{1}{2}$, the tree F_2, besides spanning more than k vertices, satisfies

$$\sum_{e \in F_2} c_e \leq 2\alpha_2 \sum_{e \in F_2} c_e \leq 4\texttt{opt}$$

by Lemma 40.1. If instead $\alpha_2 < \frac{1}{2}$, the solution is constructed by extending F_1 with nodes from F_2. Let $\ell \geq k_2 - k_1$ be the number of nodes spanned by F_2 but not by F_1. Then we can obtain a path on $k - k_1$ vertices by doubling the tree F_2, shortcutting the corresponding Eulerian tour to a simple tour of the ℓ nodes spanned only by F_2, and choosing the cheapest path of $k - k_1$ vertices from this tour. The resulting path has cost at most

$$2\frac{k - k_1}{k_2 - k_1} \sum_{e \in F_2} c_e$$

Notice that this path is disconnected from F_1. However, we can connect it by adding an edge from the root to any node of the set, costing at most opt by one of the assumptions at the beginning of the section. Since

$$\frac{k - k_1}{k_2 - k_1} = \frac{n - k_1 - (n - k)}{n - k_1 - (n - k_2)} = \frac{|A_1| - (n - k)}{|A_1| - |A_2|} = \alpha_2$$

the total cost of the produced solution is bounded by

$$\sum_{e\in F_1} c_e + 2\alpha_2 \sum_{e\in F_2} c_e + \texttt{opt} \le 2\left(\alpha_1 \sum_{e\in F_1} c_e + \alpha_2 \sum_{e\in F_2} c_e\right) + \texttt{opt} \le 4\texttt{opt} + \texttt{opt}$$

using Lemma 40.1 and the fact that $\alpha_2 < \frac{1}{2}$ implies $\alpha_1 > \frac{1}{2}$.

As for the k-TSP, it suffices to run the Prize-Collecting subroutine with halved penalties and then shortcut the Eulerian walk obtained after doubling the k-tree found, in the same way as we went from the Prize-Collecting Steiner Tree to the PTSP.

40.3.4 From the *k*-MST to the Quota Traveling Salesman Problem

In this section we will describe a 5-approximation algorithm for the Quota TSP. However, the discussion will be easier if we consider the following problem first.

Quota Minimum Spanning Tree Problem

Given an undirected graph $G = (V, E)$ with vertex weights $w : V \to \mathbb{Z}^+$ and a positive integer Q, a tree F of G such that $\sum_{v\in F} w(v) \ge Q$ is called a *quota Q-tree.* Given such a graph with edge costs $c : E \to \mathbb{Q}^+$ and a root node r, the Quota MST Problem asks to find a quota Q-tree including r of minimum total cost.

Theorem 40.4 (Ausiello et al. [9])

There is a 5-approximation algorithm for the Quota MST Problem.

The idea behind the theorem is that we can run the 5-approximation algorithm for the k-MST by Garg, but instead of setting uniformly the penalties to λ, we set $\pi_v = \lambda \cdot w_v$ when calling the Prize-Collecting Steiner Tree subroutine. The two solutions obtained at the end of the binary search phase can then be patched essentially as before.

Now, we can obtain an algorithm for the Quota TSP in the same way as we went from the Prize-Collecting Steiner Tree to the PTSP in Section 40.2. That is, it is sufficient to run the Prize-Collecting subroutine with $\pi_v = \frac{1}{2}\lambda w_v$. The analysis remains the same.

40.4 The Prize-Collecting Traveling Salesman Problem

40.4.1 Definitions

Prize-Collecting Traveling Salesman Problem

Given an undirected graph $G = (V, E)$ with vertex weights $w : V \to \mathbb{Z}^+$, vertex penalties $\pi : V \to \mathbb{Q}^+$, and a nonnegative integer Q, a cycle C of G such that $\sum_{v\in C} w(v) \ge Q$ is called a *quota Q-tour.* Given such a graph with edge costs $c : E \to \mathbb{Q}^+$ satisfying the triangle inequality and a root node r, the PCTSP asks to find a quota Q-tour $T = (V_T, E_T)$ including r that minimizes

$$c(T) = \sum_{e\in E_T} c_e + \sum_{v\in V\setminus V_T} \pi_v$$

40.4.2 History of the Results

In the general form given here, the PCTSP was first formulated by Balas [1,24], who gave structural properties of the PCTS polytope as well as heuristics. The problem arose during the task of developing daily schedules for a steel rolling mill.

The only results on guaranteed heuristics for the PCTSP are due to Awerbuch et al. [8]. They give polynomial-time algorithm with an approximation ratio of $O(\log^2(\min(Q, n)))$, where n is the number of vertices of the graph and Q is the required vertex weight to be visited. However, the PCTSP contains as special cases both the PTSP and the k-TSP, which received more attention in the literature (for the history of results on these problems, the reader can refer to the previous sections). From some recent results on these problems, we derive a constant-approximation algorithm for the general PCTSP in the following section.

40.4.3 A Constant-Factor Approximation Algorithm for PCTSP

A simple idea exploited by the algorithm of Awerbuch et al. [8] is that, for a given instance I of the PCTSP, the following quantities constitute lower bounds on the cost $\mathtt{opt}$ of an optimal solution:

(1) the cost $\mathtt{optp}$ of an optimal solution to a PTSP instance I_p defined on the same graph and having the same penalties as in the PCTSP instance (since a feasible solution to I is also feasible for I_p and has the same cost);
(2) the cost $\mathtt{optq}$ of an optimal solution to a Quota TSP instance I_q defined on the same graph and having the same weights and quota as in I (since every feasible solution to I can be turned into a feasible solution for I_q of at most the same cost).

Thus, to approximate an optimal solution to the PCTSP instance I we can:

(1) run an α-approximation algorithm for PTSP on I_p to obtain a tour T_p such that $c(T_p) \leq \alpha \cdot \mathtt{optp}$;
(2) run a β-approximation algorithm for Quota TSP on I_q to obtain a tour T_q such that $c(T_q) \leq \beta \cdot \mathtt{optq}$;
(3) concatenate T_p and T_q to obtain a tour T feasible for the PCTSP instance I of cost

$$c(T) \leq c(T_p) + c(T_q) \leq \alpha \cdot \mathtt{optp} + \beta \cdot \mathtt{optq} \leq (\alpha + \beta)\mathtt{opt}$$

This means that, by using the best algorithms currently known for PTSP and Quota TSP, we can obtain a constant-factor approximation to the PCTSP.

40.5 The Minimum Latency Problem and Graph Searching

Suppose that a plumber receives calls from various customers and decides to organize a tour of the customers for the subsequent day; for sake of simplicity let us also assume that, at each visit, the time the plumber needs to fix the customer's problem is constant. A selfish plumber would decide to schedule his tour in such way as to minimize the overall time he takes to serve all customers and come back home; such approach would require the solution of an instance of TSP. Alternatively, a nonselfish plumber would decide to schedule his tour in such a way to minimize the average time customers have to wait for his visit the day after. In this case he will have to solve an instance of the so-called *traveling repairman problem* (TRP).

The TRP problem is more frequently known in the literature as *Minimum Latency Problem* (MLP) [25], but it is also known as *school-bus driver problem* [26] and the *delivery man problem* [27,28]. Strictly related to MLP is the so-called *graph searching problem* (GSP) [29]. In such problem we assume that a single prize is hidden in a vertex of an edge-weighted graph and the vertices are labeled with the probability that the prize is stored in the vertex. The goal is to minimize the expected cost to find the prize by exploring all vertices of the graph. The relationship between MLP and GSP is discussed in Ref. [9].

40.5.1 Definitions

Minimum Latency Problem

Given an undirected graph $G = (V, E)$, with edge costs $c : E \to \mathbb{Q}^+$ satisfying the triangle inequality, let T be a tour that visits the vertices in some order. The *latency* $l_{v_i,T}$ of a vertex $v_i \in T$ is the cost of the prefix of T ending in v_i. The MLP asks to find a tour T such that the sum of the latencies of all vertices along T is minimum.

40.5.2 History of the Results

The problem has been shown to be NP-hard by Sahni and Gonzalez [30]. In Ref. [31] Afrati et al. showed that the problem can be solved in polynomial time on trees with bounded number of leaves. Recently, Sitters [32] has shown that the problem is NP-hard also in the case of general weighted trees.

From the approximability point of view, the MLP is, clearly, as the TSP, hard to approximate for any given constant ratio on general graphs [30] (i.e., when the triangle inequality does not hold), while in the case of metric spaces it can be shown to be APX-complete, that is, it allows approximation algorithms but does not allow approximation schemes.

The first constant-factor approximation algorithm for the MLP on general metric spaces has been presented by Blum et al. [25] who show that given a c-approximate algorithm for the k-MST then there exists a $8c$-approximation ratio for the MLP. Subsequently, Goemans and Kleinberg [33] showed that the constant 8 above can be lowered to 3.59, thus implying a 7.18-approximation algorithm for MLP. The best current bound is 3.59 is given by Chaudhuri et al. in Ref. [20].

Arora and Karakostas [34] showed the existence of a quasi-polynomial-time approximation algorithm when the input graph is a tree; to compute a $(1+\varepsilon)$-approximation the algorithm requires time $n^{O(\log n/\varepsilon)}$ time.

We finally remark that the problem has also been extended to the case of k repairmen. Namely, Fakcharoenphol et al. [35] showed the first constant approximation algorithm for the problem. This result has been improved to 8.49-approximation by Chaudhuri et al. in Ref. [20].

40.5.3 A 3.59-Approximation Algorithm for the Minimum Latency Problem

We first present the algorithm proposed by Goemans and Kleinberg in Ref. [33] that gives a 7.18-approximation algorithm. The procedure proposed by the authors computes, for every $j = 1, 2, \ldots, n$ the tour T_j of minimum length that visits j vertices. Then we have to concatenate a subsequence of the tours to form the desired tour. Clearly, the goal is to select those values $j_1, \ldots, j_m$ such that the latency of the final tour obtained by stitching together tours $T_{j_1} \ldots T_{j_m}$ is minimized.

Let d_{j_i} and p_i be the length of tour T_{j_i} and the number of new vertices visited during the same tour, respectively. It is simple to show that the following claim holds:

$$\sum_{i=1}^{m} p_i d_{j_i} \le \sum_{i=1}^{m} (j_i - j_{i-1}) d_{j_i}$$

Note that if, for every i, the tours T_{j_i} and $T_{j_{i-1}}$ were nested, the above inequality would be trivially satisfied, but a careful analysis of the contributions involved on the right and the left-hand sides of the inequality may convince the reader that such inequality is also true when the tours are not nested. It follows that for a number of vertices equal to $\sum_{k=1}^{i} p_k - j_i$ we sum a contribution at most d_{j_k} on the left-hand side of the equation while a contribution larger than d_{j_k} on the right-hand side of the equation. Moreover, each tour T_{j_i} is traversed in the direction that minimizes the total latency of the vertices discovered during tour T_{j_i}. This allows to rewrite the total latency of the tour obtained by concatenating $T_{j_1}, \ldots, T_{j_m}$ as

$$\sum_i \left(n - \sum_{k=1}^{i} p_k \right) d_{j_i} + \frac{1}{2} \sum_i p_i d_{j_i}$$

$$\le \sum_i (n - j_i) d_{j_i} + \frac{1}{2} \sum_i (j_i - j_{i-1}) d_{j_i}$$

$$= \sum_i \left(n - \frac{j_{i-1} + j_i}{2} \right) d_{j_i}$$

The formula above allows to rewrite the total latency of the algorithm only in terms of the indices j_i and of the length d_{j_i}, independently from the number of new vertices discovered during each tour. A complete digraph of $n+1$ vertices is then constructed in the following way. Arc $(i,\ j)$ goes from $min(i,\ j)$ to $max(i,\ j)$ and has length $(n - \frac{i+j}{2})d_{j_i}$. The algorithm computes a shortest path from node 0 to node n. Assume that the path goes through nodes $0 = j_0 < j_1 < \cdots < j_m = n$. The tour is then obtained by concatenating tours $T_{j_1}, \ldots, T_{j_m}$.

The obtained solution is compared against the following lower bound $OPT \geq \sum_{k=1}^{n} \frac{d_k}{2}$. This lower bound follows from the observation that the kth vertex cannot be visited before $d_k/2$ in any optimum tour. The approximation ratio of the algorithm is determined by bounding the maximum over all the possible set of distances $d_1, \ldots, d_n$ of the ratio between the shortest path in G_n and the lower bound on the optimum solution. This value results to be smaller than 3.59.

Theorem 40.5 (Goemans and Kleinberg [33])

Given a c-approximation algorithm for the problem of finding an a tour of minimum length spanning at least k vertices on a specific metric space, then there exists a 3.59c*-approximation algorithm for the* MLP *on the same metric space.*

Again by making use of the 2-approximation algorithm of [6] for k-MST and k-TSP we may achieve a ratio 7.18 for MLP.

With respect to the results we have seen so far, a remarkable step forward has been achieved by Chaudhuri et al. in Ref. [20]. Using techniques from Garg [5], Arora and Karakostas [2], and Archer et al. [36], the authors are able to find a k-MST whose cost is no more than $(1+\varepsilon)$ the cost of the minimum path visiting k vertices. Since such a cost is a lower bound on the latency of a k tour the result implies the following theorem.

Theorem 40.6 (Chaudhuri et al. [20])

There exists a 3.59*-approximation algorithm for the* MLP *on general metric spaces.*

By the arguments provided in Ref. [9] the same approximation bound also holds for GSP.

References

[1] Balas, E., The prize collecting traveling salesman problem, *Networks*, 19, 621, 1989.

[2] Arora, S. and Karakostas, G., A $2+\varepsilon$ approximation algorithm for the k-MST problem, *Proc. SODA*, 2000, p. 754.

[3] Arya, S. and Ramesh, H., A 2.5-factor approximation algorithm for the k-MST problem, *Inf. Proc. Lett.*, 65(3), 117, 1998.

[4] Blum, A., Ravi, R., and Vempala, S., A constant-factor approximation algorithm for the k-MST problem, *Proc. of STOC*, 1996, p. 442.

[5] Garg, N., A 3-approximation for the minimum tree spanning k vertices, *Proc. FOCS*, 1996, p. 302.

[6] Garg, N., Saving an epsilon: A 2-approximation for the k-MST problem in graphs, *Proc. STOC*, 2005, p. 396.

[7] Goemans, M. and Williamson, D. P., A general approximation technique for constrained forest problems, *SIAM J. Comput.*, 24(2), 296, 1995.

[8] Awerbuch, B., Azar, Y., Blum, A., and Vempala, S., New approximation guarantees for minimum-weight k-trees and prize-collecting salesmen, *SIAM J. Comput.*, 28(1), 254, 1999.

[9] Ausiello, G., Leonardi, S., and Marchetti-Spaccamela, A., On salesmen, repairmen, spiders, and other traveling agents, *Proc. 4th Italian Conf. on Algorithms and Complexity*, Lecture Notes in Computer Science, Vol. 1767, Springer, Berlin, 2000, p. 1.

[10] Gabow, H. N. and Pettie, S., The dynamic vertex minimum problem and its application to clustering-type approximation algorithms, *Proc. 8th Scandinavian Workshop on Algorithm Theory*, Lecture Notes in Computer Science, Vol. 2368, Springer, Berlin, 2002, p. 190.

[11] Bienstock, D., Goemans, M. X., Simchi-Levi, D., and Williamson, D., A note on the prize collecting traveling salesman problem, *Math. Prog.*, 59(3), 413, 1993.

[12] Bern, M. W. and Plassmann, P. E., The Steiner problem with edge lengths 1 and 2, *Inf. Proc. Lett.*, 32(4), 171, 1989.

[13] Papadimitriou, C. H. and Yannakakis, M., The traveling salesman problem with distances one and two, *Math. Oper. Res.*, 18(1), 1, 1993.

[14] Fischetti, M., Hamacher, H. W., Jørnsten, K., and Maffioli, F., Weighted k-cardinality trees: complexity and polyhedral structure, *Networks*, 24(1), 11, 1994.

[15] Cheung, S. Y. and Kumar, A., Efficient quorumcast routing algorithms, *Proc. of INFOCOM*, 2, 1994, 840.

[16] Ravi, R., Sundaram, R., Marathe, M. V., Rosenkrantz, D. J., and Ravi, S. S., Spanning trees short or small, *SIAM J. Disc. Math.*, 9(2), 178, 1996.

[17] Blum, A., Ravi, R., and Vempala, S., A constant-factor approximation algorithm for the k-MST problem, *JCSS*, 58(1), 101, 1999.

[18] Golden, B. L., Levy, L., and Vohra, R., The orienteering problem, *Nav. Res. Logistics*, 34, 307, 1987.

[19] Blum, A., Chawla, S., Karger, D. R., Lane, T., Meyerson, A., and Minkoff, M., Approximation algorithms for orienteering and discounted-reward TSP, *Proc. FOCS*, 2003, p. 46.

[20] Chaudhuri, K., Godfrey, B., Rao, S., and Talwar, K., Paths, trees, and minimum latency tours, *Proc. FOCS*, 2003, p. 36.

[21] Bansal, N., Blum, A., Chalasani, P., and Meyerson, A., Approximation algorithms for deadline-tsp and vehicle routing with time-windows, *Proc. STOC*, 2004, p. 166.

[22] Johnson, D. S., Minkoff, M., and Phillips, S., The prize collecting steiner tree problem: theory and practice, *Proc. SODA*, 2000, p. 760.

[23] Chudak, F. A., Roughgarden, T., and Williamson, D. P., Approximate k-MSTs and k-Steiner trees via the primal-dual method and Lagrangean relaxation, *Math. Prog.*, 100(2), 411, 2004.

[24] Balas, E., The prize collecting traveling salesman problem: II. Polyhedral results, *Networks*, 25, 199, 1995.

[25] Blum, A., Chalasani, P., Coppersmith, D., Pulleyblank, W. R., Raghavan, P., and Sudan, M., The minimum latency problem, *Proc. of STOC*, 1994, p. 163.

[26] Will, T. G., Extremal Results and Algorithms for Degree Sequences of Graphs, Ph.D. thesis, University of Illinois at Urbana-Champaign, 1993.

[27] Minieka, E., The deliver man problem on a tree network, *Ann. Oper. Res.*, 18, 261, 1989.

[28] Fischetti, M., Laporte, G., and Martello, S., The delivery man problem and cumulative matroids, *Oper. Res.*, 41(6), 1055, 1993.

[29] Koutsoupias, E., Papadimitriou, C. H., and Yannakakis, M., Searching a fixed graph, *Proc. of ICALP*, Lecture Notes in Computer Science, Vol. 1099, Springer, 1996, p. 280.

[30] Sahni, S. and Gonzalez, T. F., P-complete approximation problems, *JACM*, 23(3), 555, 1976.

[31] Afrati, F. N., Cosmadakis, S. S., Papadimitriou, C. H., Papageorgiou, G., and Papakostantinou, N., The complexity of the travelling repairman problem, *Informatique Théorique Appl.*, 20(1), 79, 1986.

[32] Sitters, R., The minimum latency problem is NP-hard for weighted trees, *Proc. 9th Integer Programming and Combinatorial Optimization Conf.*, 2002, p. 230.

[33] Goemans, M. and Kleinberg, J., An improved approximation ratio for the minimum latency problem, *Math. Prog.*, 82(1), 111, 1998.

[34] Arora, S. and Karakostas, G., Approximation schemes for minimum latency problems, *SIAM J. Comput.*, 32(5), 1317, 2003.

[35] Fakcharoenphol, J., Harrelson, C., and Rao, S., The k-traveling repairman problem, *Proc. SODA*, 2003, p. 655.

[36] Archer, A., Levin, A., and Williamson, D. P., A Faster, Better Approximation Algorithm for the Minimum Latency Problem, Technical report 1362, School of Operations Research and Industrial Engineering, Cornell University, Ithaca, NY, 2003.