\(O(n^2 \log n)\) implementation of an approximation for the Prize-Collecting Steiner Tree Problem

Paulo Feofiloff \(^*\)  Cristina G. Fernandes \(^*\)  Carlos E. Ferreira \(^*\)  José Coelho de Pina \(^*\)

February 2002

Abstract

We give a low-level description of an \(O(n^2 \log n)\) implementation of Johnson, Minkoff and Phillips’ approximation algorithm for the Prize-Collecting Steiner Tree Problem.

1 Introduction

The Prize-Collecting Steiner Tree Problem is an extension of the Steiner Tree Problem where each vertex left out of the tree pays a penalty. The goal is to find a tree which minimizes the sum of its edge costs and the penalties for the vertices left out of the tree. Johnson, Minkoff and Phillips [2] presented a 2-approximation for this problem based on the primal-dual scheme. In this manuscript, we describe in details an \(O(n^2 \log n)\) implementation of this algorithm.

We adopt the notation used [1], which is summarized below. We start with a formal definition of the problem. Consider a graph \(G = (V, E)\), a function \(c\) from \(E\) into \(\mathbb{Q}_{\geq 0}\) (non-negative rationals) and a function \(\pi\) from \(V\) into \(\mathbb{Q}_{\geq 0}\). For any subset \(F\) of \(E\) and any subset \(W\) of \(V\), let \(c(F) := \sum_{e \in F} c_e\) and \(\pi(W) := \sum_{w \in W} \pi_w\). The **Prize-Collecting Steiner Tree Problem** (PCST) consists of the following: given \(G\), \(c\), and \(\pi\), find a tree \(T\) in \(G\) such that

\[
c(E_T) + \pi(V \setminus V_T)\]

is minimum.

\((V_H\) and \(E_H\) denote the vertex and edge sets of a graph \(H\).)

An edge is **internal to** a partition \(P\) of \(V\) if both of its ends are in the same element of \(P\). All other edges are **external to** \(P\). For any external edge, there are two elements of \(P\) containing its ends. We call those two elements the **extremes** of the edge in \(P\).

A collection \(\mathcal{L}\) of subsets of \(V\) is **laminar** if, for any two elements \(L_1\) and \(L_2\) of \(\mathcal{L}\), either \(L_1 \cap L_2 = \emptyset\) or \(L_1 \subseteq L_2\) or \(L_2 \subseteq L_1\). The collection of maximal elements of a laminar collection \(\mathcal{L}\) will be denoted by \(\mathcal{L}^\ast\). So, \(\mathcal{L}^\ast\) is a collection of disjoint subsets of \(V\). Let \(\bigcup \mathcal{L}\) denote the union of all sets in \(\mathcal{L}\).

\(^*\)Departamento de Ciência da Computação, Instituto de Matemática e Estatística, Universidade de São Paulo, Rua do Matão 1010, 05508-090 São Paulo/SP, Brazil. E-mail: \{pf, cris, cef, coelho\}@ime.usp.br. Research supported in part by PRONEX/CNPq 664107/1997-4 (Brazil).

\(^1\)Research supported in part by CNPq Proc. 301174/97-0 (Brazil).

\(^2\)Research supported in part by CNPq Proc. 300752/94-6 (Brazil).
For any collection $\mathcal{L}$ of subsets of $V$ and any subset $X$ of $V$, let $\overline{X} := V \setminus X$, $\mathcal{L}^X := \{ L \in \mathcal{L} : L \subseteq X \}$ and $\mathcal{L}_X := \{ L \in \mathcal{L} : L \supseteq X \}$. When $X = \{ v \}$, we write $\mathcal{L}_v$ and $\mathcal{L}_v$ instead, and when $X = V_T$ or $X = \overline{V_T}$, we write $T$ or $\overline{T}$ instead. For any $e$ in $E$, let $\mathcal{L}(e) := \{ L \in \mathcal{L} : e \in \delta_G L \}$, where $\delta_G L$ stands for the set of edges of $G$ with one end in $L$ and the other in $\overline{L}$. For any function $y$ from $\mathcal{L}$ into $\mathbb{Q}_\geq$ and any subcollection $\mathcal{M}$ of $\mathcal{L}$, let $y(\mathcal{M}) := \sum_{L \in \mathcal{M}} y(L)$.

We say that $y$ respects a function $c$ defined on $E$ (relative to $\mathcal{L}$) if
\[ y(\mathcal{L}(e)) \leq c_e \quad \text{for each } e \in E. \] (1)

An edge $e$ is tight for $y$ if equality holds in (1).

We say $y$ respects a function $\pi$ defined on $V$ (relative to $\mathcal{L}$) if
\[ y(\mathcal{L}^L) \leq \pi(L) \quad \text{for each } L \in \mathcal{L}. \] (2)

(3)

2 Johnson, Minkoff and Phillips’ algorithm

In its high-level description below, we refer to an algorithm PRUNING whose high-level description we omit. It corresponds to the second phase of the primal-dual scheme, where edges are deleted from the tree produced in the first phase.

Johnson, Minkoff and Phillips’ algorithm receives $G$, $c$, $\pi$ and returns a tree $T$ in $G$ such that $c(E_T) + \pi(V_T) \leq 2 \text{opt}(\text{PCST}(G, c, \pi))$. Each iteration starts with a spanning forest $F$ in $G$, a laminar collection $\mathcal{L}$ of subsets of $V$ with $\bigcup \mathcal{L} = V$, a subcollection $\mathcal{S}$ of $\mathcal{L}$, and a function $y$ from $\mathcal{L}$ into $\mathbb{Q}_\geq$. The first iteration starts with $F = (V, \emptyset)$, $\mathcal{L} = \{ \{ v \} : v \in V \}$, $\mathcal{S} = \emptyset$, and $y = 0$. Each iteration consists of the following:

Case 1: $|\mathcal{L}^* \setminus \mathcal{S}| > 1$.

Let $\varepsilon$ be the largest number in $\mathbb{Q}_\geq$ such that the function $y'$ defined by
\[ y'_L = \begin{cases} y_L + \varepsilon, & \text{if } L \in \mathcal{L}^* \setminus \mathcal{S} \\ y_L, & \text{otherwise} \end{cases} \]
respects $c$ and $\pi$.

Subcase 1A: some edge $e$ external to $\mathcal{L}^*$ is tight for $y'$.

Let $L_1$ and $L_2$ be the extremes of $e$ in $\mathcal{L}^*$. Set $y'_{L_1 \cup L_2} := 0$ and start a new iteration with $F + e$, $\mathcal{L} \cup \{ L_1 \cup L_2 \}$, $\mathcal{S}$, $y'$ in the roles of $F$, $\mathcal{L}$, $\mathcal{S}$, $y$ respectively.

Subcase 1B: some element $L$ of $\mathcal{L}^* \setminus \mathcal{S}$ is tight for $y'$.

Start a new iteration with $F$, $\mathcal{L}$, $\mathcal{S} \cup \{ L \}$, $y'$ in the roles of $F$, $\mathcal{L}$, $\mathcal{S}$, $y$ respectively.

Case 2: $|\mathcal{L}^* \setminus \mathcal{S}| = 1$.

Let $M$ be the only element of $\mathcal{L}^* \setminus \mathcal{S}$. Call subalgorithm PRUNING with arguments $F \cap M$, $\mathcal{L}^M$, and $\mathcal{S}^M$. The subalgorithm returns a subcollection $\mathcal{Z}$ of $\mathcal{S}^M$. Return $T := (F \cap M) - \bigcup \mathcal{Z}$ and stop.
3 Data structures and basic functions

Here is the list of variables and functions used by the algorithm:

1. \( L_1, \ldots, L_N \) are nonempty subsets of \( V_G \) such that \( L_1 \cup \cdots \cup L_N = V_G \) and, for each pair \( i < j \), either \( L_i \subseteq L_j \) or \( L_i \cap L_j = \emptyset \), whence \( N < 2n \), where \( n := |V_G| \). Each \( L_i \) is represented by a bit vector as well as by a linked list. (In the high-level version of the algorithm given in [1], \( \{L_1, \ldots, L_N\} \) is denoted by \( \mathcal{L} \).

2. A subset \( F \) of \( E_G \), represented as a doubly-linked list (a bit vector would be too long). Since \((V_G, F)\) is a forest, \(|F| < n\).

3. A bit vector \( \mu[1 \ldots N] \) such that \( \mu[i] = 1 \) iff \( L_i \) is a maximal element of \( \{L_1, \ldots, L_N\} \). (In the high-level version of the algorithm, this set of maximal elements is denoted by \( \mathcal{L}^* \).)

4. An array \( d \) indexed by \( V_G \) with values in \( \mathbb{Q} \). (In terms of the high-level notation, \( d[v] := \mathcal{L}_v \equiv \sum_{L_i \subseteq L: v \in L_i} y_L \) for each vertex \( v \).)

5. A function \( \text{ResidualCost} \) that takes edges into \( \mathbb{Q} \): upon receiving an edge \( uv \), the function returns the number \( c_{uv} - d[u] - d[v] \). Of course this can be implemented to run in \( O(1) \) time. (We do not treat \( \text{ResidualCost} \) as an array because we cannot afford to update \( \text{ResidualCost} \) every time \( d \) changes.)

6. An array \( \Delta[1 \ldots N] \) with values in \( \mathbb{Q} \). (In terms of the high-level notation, \( \Delta[i] = \sum_{v \in L_i} \pi[v] - \sum_{S \subseteq L_i} y_S \).)

7. A bit vector \( \lambda[1 \ldots N] \) such that if \( \lambda[i] = 0 \) then \( \Delta[i] = 0 \). We say that \( L_i \) is active iff \( \lambda[i] = 1 \). (In terms of the high-level notation, \( \lambda[i] = 0 \) iff \( L_i \in \mathcal{S} \).)

8. A variable \( mzActive \) records the cardinality of the set \( \{i : 1 \leq i \leq N, \mu[i] = 1, \lambda[i] = 1\} \).

9. An array \( A[1 \ldots N, 1 \ldots N] \) whose elements are sets of at most one edge each. More specifically, for \( i \neq j \) such that \( \mu(i) = \mu(j) = 1 \),

   if \( \delta(L_i) \cap \delta(L_j) = \emptyset \) then \( A[i, j] = A[j, i] = \emptyset \);
   otherwise, \( A[i, j] = A[j, i] = \{uv\} \) where \( uv \) is an element of \( \delta(L_i) \cap \delta(L_j) \) that minimizes \( \text{ResidualCost}(uv) \).

10. A function \( \text{Key} \) defined on \( \{1, \ldots, N\} \times \{1, \ldots, N\} \) as follows: if \( A[i, j] = \emptyset \) then \( \text{Key}(i, j) = \infty \); else \( \text{Key}(i, j) = \text{ResidualCost}(uv) \), where \( uv \) is the unique edge in \( A[i, j] \). Of course this function can be implemented to run in \( O(1) \) time.

11. For each \( i \) such that \( \mu[i] = 1 \), there are two subsets of \( \{1, \ldots, N\} \) denoted by \( H_0[i] \) and \( H_1[i] \). For each \( h \), the set \( H_h[i] \) consists of all \( j \neq i \) such that \( \mu[j] = 1, \lambda[j] = h, A[i, j] \neq \emptyset \).

Each set \( H_h[i] \) is organized as a min-heap, the key of each element \( j \) being \( \text{Key}(i, j) \). Hence, the first element of \( H_h[i] \) minimizes \( \text{Key}(i, j) \).

---

1 Johnson, Minkoff and Phillips say this is the “surplus” of \( L_i \).

2 Johnson, Minkoff and Phillips say that the key of \( j \) is the “deficit” of the only edge in \( A[i, j] \).
12. For \( h \in \{0, 1\} \), we assume that we can decide in time \( O(1) \) whether or not a statement like "\( p \in H_{h}[i] \)" is true or false. Moreover, if the statement is true, we assume that the deletion of \( p \) from \( H_{h}[i] \) can be carried out in \( O(\log n) \) time. (This is easy to implement: for each \( i \), each \( h \), and each \( p \) in \( \{1, \ldots, N\} \), maintain the location of \( p \) in \( H_{h}[i] \).)

4 Main functions

The core of the algorithm is given by the next functions.

```
PCST-LOW-LEVEL \((G, c, \pi)\)
1   \text{INITIALIZATION}() 
2   \( N \leftarrow mActive \leftarrow n \) 
3   \text{while } mActive > 1 \text{ do } \triangleright \text{ at most } 2n \text{ iterations}
4       \text{ONEITERATION}() 
5   \( (X, F) \leftarrow \text{PRUNING}() \)
6   \text{return } X \text{ and } F
```

The number of iterations is \( \leq 2n \) because the sum \( 2 \times mActive + mInactive \), where \( mInactive \) is the cardinality of \( \{i : 1 \leq i \leq N, \mu[i] = 1, \lambda[i] = 0\} \), starts at \( 2n \) and strictly decreases with each iteration.

```
\text{INITIALIZATION}() 
01   n \leftarrow |V_G| 
02   i \leftarrow 0 
03   \text{for each } v \text{ in } V_G \text{ do} 
04       d[v] \leftarrow 0 
05   i \leftarrow i + 1 
06   L_i \leftarrow \{v\} 
07   o[v] \leftarrow i 
08   \mu[i] \leftarrow \lambda[i] \leftarrow 1 
09   \Delta[i] \leftarrow \pi_v 
10   \text{for each } i \text{ in } \{2, \ldots, n\} \text{ do} 
11       \text{for each } j \text{ in } \{1, \ldots, i - 1\} \text{ do} 
12           A[i, j] \leftarrow \emptyset 
13       \text{KEY}(i, j) = \infty 
14   \text{for each } i \text{ in } \{2, \ldots, n\} \text{ do} 
15       \text{for each } uv \text{ in } \delta(L_i) \text{ do} 
16           \text{if } o[u] = i 
17               \text{then } j \leftarrow o[v] 
18           \text{else } j \leftarrow o[u] 
19           \text{if } \text{KEY}(i, j) > \text{RESIDUALCOST}(uv) 
20               \text{then } A[i, j] \leftarrow A[j, i] \leftarrow \{uv\}
```
\begin{verbatim}
21 \( F \leftarrow \emptyset \)
22 \( H_0[i] \leftarrow \emptyset \)
23 \text{for each } i \in \{1, \ldots, n\} \text{ do}
24 \hspace{1em} \( H_1[i] \leftarrow \emptyset \)
25 \text{for each } j \in \{1, \ldots, n\} - \{i\} \text{ do}
26 \hspace{2em} \text{if } A[i, j] \neq 0 \text{ then } H_1[i] \leftarrow H_1[i] \cup \{j\}

The total time spent executing lines 14–20 is \( O(m) = O(n^2) \). The total time spent building the heap \( H_1[i] \) in lines 20–21 is \( O(n) \). The total spent by \textsc{Initialization} is \( O(n^2) \).

\textsc{OneIteration()} \hspace{1em} \triangleright \text{each call takes } O(n \log n) \text{ time}
01 \hspace{1em} \epsilon' \leftarrow \epsilon'' \leftarrow \infty
02 \text{for each } p \in \{1, \ldots, N\} \text{ such that } \mu[p] = \lambda[p] = 1 \text{ do}
03 \hspace{1em} \text{if } \epsilon' > \Delta[p] \text{ then } \epsilon' \leftarrow \Delta[p]
04 \hspace{2em} p' \leftarrow p
05 \text{if } H_0[p] \neq \emptyset \text{ then let } q \text{ be the first element of } H_0[p]
06 \hspace{1em} \text{if } \epsilon'' > \text{KEY}(p, q) \text{ then } \epsilon'' \leftarrow \text{KEY}(p, q)
07 \hspace{2em} p'' \leftarrow p
08 \hspace{2em} q'' \leftarrow q
09 \text{if } H_1[p] \neq \emptyset \text{ then let } q \text{ be the first element of } H_1[p]
10 \hspace{1em} \text{if } \epsilon'' > \frac{1}{2} \text{KEY}(p, q) \text{ then } \epsilon'' \leftarrow \frac{1}{2} \text{KEY}(p, q)
11 \hspace{2em} p'' \leftarrow p
12 \hspace{2em} q'' \leftarrow q
13 \hspace{1em} \epsilon \leftarrow \min(\epsilon', \epsilon'')
14 \text{for each } p \in \{1, \ldots, N\} \text{ such that } \mu[p] = \lambda[p] = 1 \text{ do}
15 \hspace{1em} \Delta[p] \leftarrow \Delta[p] - \epsilon
16 \text{for each } v \text{ in } L_p \text{ do}
17 \hspace{1em} d[v] \leftarrow d[v] + \epsilon
18 \hspace{2em} \triangleright \text{no need to rebuild heaps } H_0 \text{ and } H_1
19 \hspace{1em} \text{if } \epsilon = \epsilon'
20 \hspace{2em} \text{then } \textsc{Subcase1B}(p') \hspace{1em} \triangleright \text{takes time } O(n \log n)
21 \hspace{1em} \text{else } \textsc{Subcase1A}(p'', q'') \hspace{1em} \triangleright \text{takes time } O(n \log n)
\end{verbatim}

Taken together, all executions of line 22 consume \( O(n) \) time. The total spent by \textsc{OneIteration} is \( O(n \log n) \).
\textbf{Subcase1A}(p, q) \triangleright merge \ L_p and \ L_q; takes time \(O(n \log n)\)

01 let \(uv\) be the unique element of \(A[p, q]\)
02 \(F \leftarrow F \cup \{uv\}\)
03 \(L_{N+1} \leftarrow L_p \cup L_q\)
04 \(\mu[p] \leftarrow \mu[q] = 0 \triangleright\) \(L_p\) and \(L_q\) are no longer maximal
05 \(\mu[N+1] \leftarrow 1 \triangleright\) now \(L_{N+1}\) is maximal
06 if \(\lambda[q] = 1\) then \(\text{mzActive} \leftarrow \text{mzActive} - 1\)
07 \(\Delta[N+1] \leftarrow \Delta[p] + \Delta[q]\)
08 \(\lambda[N+1] \leftarrow 1 \triangleright\) now \(L_{N+1}\) is active
09 for each \(i\) in \(\{1, \ldots, N\}\) such that \(\mu[i] = 1\) do
10 \hspace{1em} if \(\text{Key}(p, i) \leq \text{Key}(q, i)\)
11 \hspace{2em} then \(A[N+1, i] \leftarrow A[i, N+1] \leftarrow A[p, i]\)
12 \hspace{2em} else \(A[N+1, i] \leftarrow A[i, N+1] \leftarrow A[q, i]\)
13 for each \(h\) in \(\{0, 1\}\) do
14 \hspace{1em} \(H_h[N+1] \leftarrow H_h[p] \triangleright\) time \(O(1)\)
15 \hspace{1em} \(H_h[N+1] \leftarrow H_h[N+1] - \{q\} \triangleright\) time \(O(\log n)\)
16 for each \(i\) in \(H_h[q]\) do
17 \hspace{2em} if \(i \notin H_h[N+1]\)
18 \hspace{3em} then \(H_h[N+1] \leftarrow H_h[N+1] \cup \{i\} \triangleright\) time \(O(\log n)\)
19 \hspace{2em} if \(i \in H_h[N+1]\) and \(\text{Key}(N+1, i) > \text{Key}(q, i)\)
20 \hspace{3em} then \(\text{DECREASE-KEY}(H_h[N+1], i, \text{Key}(q, i))\)
21 for each \(i\) in \(\{1, \ldots, N\}\) such that \(\mu[i] = 1\) do
22 \hspace{1em} \(H_0[i] \leftarrow H_0[i] - \{q\} \triangleright\) time \(O(\log n)\)
23 \hspace{1em} \(H_1[i] \leftarrow H_1[i] - \{p, q\} \triangleright\) time \(O(\log n)\)
24 \hspace{1em} if \(\text{Key}(i, N+1) < \infty\)
25 \hspace{2em} then \(H_1[i] \leftarrow H_1[i] \cup \{N+1\} \triangleright\) time \(O(\log n)\)
26 \(N \leftarrow N+1\)

\textbf{Subcase1B}(p) \triangleright deactivate \ L_p

01 \(\lambda[p] \leftarrow 0\)
02 \(\text{mzActive} \leftarrow \text{mzActive} - 1\)
03 for each \(i\) in \(\{1, \ldots, N\} - \{p\}\) do
04 \hspace{1em} if \(p \in H_1[i]\)
05 \hspace{2em} then \(H_1[i] \leftarrow H_1[i] - \{p\} \triangleright\) time \(O(\log n)\)
06 \hspace{2em} \(H_0[i] \leftarrow H_0[i] \cup \{p\} \triangleright\) time \(O(\log n)\)

6
PRUNING () $\triangleright O(n^2)$ time
01 for $i \leftarrow N$ down to 1 do $\triangleright$ “reverse delete”
02 if $\lambda[i] = 0$ $\triangleright L_i$ is inactive
03 then $\text{degree} \leftarrow 0$
04 for each $uv$ in $F$ do $\triangleright O(n)$ time
05 if $|\{u, v\} \cap L_i| = 1$ $\triangleright O(1)$ time
06 then $\text{degree} \leftarrow \text{degree} + 1$
07 $\triangleright \text{degree}$ is the cardinality of $F \cap \delta(L_i)$
08 if $\text{degree} \leq 1$
09 then for each $uv$ in $F$ do
10 if $|\{u, v\} \cap L_i| \geq 1$
11 then $F \leftarrow F - \{uv\}$
12 if $F \neq \emptyset$
13 then let $X$ be the set of vertices of $G[F]$
14 else let $x$ be a vertex that maximizes $\pi_x$
15 $X \leftarrow \{x\}$
16 return $(X, F)$

References
