Matemática Contemporânea, Vol. 42, 39–48
http://doi.org/10.21711/231766362012/rmc425

Design of connection networks with bounded number of non-terminal vertices

Mitre C. Dourado **Rodolfo A. Oliveira**
Fábio Protti **Uéverton S. Souza**

Abstract

The SPANNING TREE PROBLEM and the STEINER TREE PROBLEM aim at obtaining an acyclic subgraph connecting a set of terminal points and satisfying some properties. Both problems have several important network applications. Let G be a graph. A *connection tree* of a subset $W \subseteq V(G)$ is an acyclic, connected subgraph T of G such that $W \subseteq V(T)$ and all leaves of T are in W. In a connection tree, there are three types of vertices: (1) *terminal vertices*, i.e., those belonging to W; (2) non-terminal vertices with degree two in T, called *linkers*; (3) non-terminal vertices with degree at least three in T, called *routers*. Motivated by its large potential applicability, we propose a new problem in graphs, called TERMINAL CONNECTION PROBLEM (TCP), where the number of non-terminal vertices (linkers and/or routers) is bounded by a constant value. In this work, we prove NP-complete and polynomial cases for variants of the TCP.

2000 AMS Subject Classification: 60K35, 60F05, 60K37.
Key Words and Phrases: Computational Complexity, Graph Algorithms, Network Design, NP-completeness.
*Partially Supported by CNPq and FAPERJ.

1 Introduction

In [5], Prim discusses the importance of connecting terminal points with shortest possible network direct links, i.e., for a given set of terminal points, the problem consists of connecting all terminals by a network of direct terminal-to-terminal links having the smallest possible total length (sum of the link lengths). This problem is called the SPANNING TREE PROBLEM and has several applications in network design, particularly in communication, distribution, and transportation networks.

Another important problem is the STEINER TREE PROBLEM [1, 4]: given a connected, weighted graph G and a subset $W \subseteq V(G)$, find a minimum cost subgraph containing W, where the use of additional non-terminal points, called *Steiner vertices*, is allowed. For unweighted graphs, a Steiner tree is a connected subgraph T of G such that $W \subseteq V(T)$ and $|E(T)|$ is minimum.

A *connection tree* for a subset $W \subseteq V(G)$ is an acyclic, connected subgraph T of G, where $W \subseteq V(T)$ and all leaves of T are in W. Clearly, every Steiner tree for W is a connection tree. In a connection tree, there are three types of vertices: (1) the vertices of W, called *terminals*; (2) the vertices in $V(T)\backslash W$ with degree two in T, called *linkers*; (3) the vertices in $V(T)\backslash W$ with degree at least three in T, called *routers*.

Motivated by a large potential applicability, we study the complexity of determining whether an unweighted graph G admits a connection tree T for a subset $W \subseteq V(G)$ satisfying the following conditions: (*i*) $V(T)$ contains at most r routers; (*ii*) $V(T)$ contains at most ℓ linkers. The idea of limiting the number of non-terminal vertices has arisen from some questions in network information security, since non-terminal vertices used for establishing the connection may try to access private information shared only by terminals. In addition, in some situations, these limits represent the real scenario when designing different connecting structures, finding ways to build roads or railways to connect a set of locations, or deciding routing policies over the internet for multicast traffic.

Note that determining a minimum connection tree satisfying (i) and (ii) is equivalent to determining a spanning tree when $W = V(G)$, $r = 0$, and $\ell = 0$; and is equivalent to determining a Steiner tree when $r = \ell = |V(G)\backslash W|$.

In this paper, we prove that deciding whether there exists a connection tree T satisfying (i) and (ii) is NP-complete, even when: (a) the parameter ℓ is a fixed constant value; (b) the parameter r is a fixed constant value. On the other hand, we show that when both parameters are fixed the problem can be solved in polynomial time.

2 Computational Complexity Results

As it is well-known, a spanning tree can be found in polynomial time [5], but finding a Steiner tree is NP-hard [3]. In [2], Dreyfus and Wagner presented a dynamic programming algorithm that obtains a Steiner tree in $O(n^3 + n^2 2^{k-1} + n 3^{k-1})$ time, where k is the number of terminals; this result (obtained independently by Levin [1]) implies a polynomial time algorithm when k is bounded by a constant or is a linear function of $\log n$.

We define the TERMINAL CONNECTION PROBLEM as follows:

TERMINAL CONNECTION PROBLEM - TCP

Instance: A connected graph G, a subset $W \subseteq V(G)$, and two nonnegative
integers ℓ and r.

Question: Does G contain a connection tree T with at most ℓ linkers and
r routers?

The TCP is clearly in NP, because it is easy to check whether a tree T connects W using at most ℓ linkers and r routers. By setting constant values to some parameters of the TCP, three variants are formulated:

- TCP(ℓ) - the version of the TCP with ℓ bounded by a constant;
- TCP(r) - the version of the TCP with r bounded by a constant;

- TCP(ℓ, r) - the version of the TCP with ℓ and r bounded by constants.

Note that TCP$(0,0)$ is equivalent to the SPANNING TREE PROBLEM.

Theorem 2.1. TCP(ℓ) is NP-complete.

Proof. The proof uses a reduction from the problem 3-SAT. We show that given a boolean formula F with m clauses and n variables, there is some truth assignment of `true` and `false` values to the variables in F if and only if in the associated graph G there is a connection tree for a specific subset $W \subseteq V(G)$ with $r = 2n+1$ routers and $\ell = 2$ linkers. (For other values of ℓ, even for $\ell = 0$, the proof can be easily adapted.)

Given a boolean formula F with m clauses and n variables where each clause contains exactly 3 literals, construct an associated graph G as follows:

- for each clause C_j of F, create three vertices c_j^1, c_j^2, c_j^3 in G;
- for each variable X_i of F, create a gadget g_i consisting of the vertices x_i, t_{x_i}, f_{x_i}, $w_{x_i}^1$, $w_{x_i}^2$ and the edges $(w_{x_i}^1, x_i)$, (x_i, t_{x_i}), $(t_{x_i}, w_{x_i}^2)$, $(w_{x_i}^2, f_{x_i})$, (f_{x_i}, x_i);
- create the gadget g_f consisting of the vertices f, w_f^1, w_f^2, l^1, l^2 and the edges (f, l_1),(f, l_2),(l_1, w_f^1), (l_2, w_f^2);
- for all i, add an edge (f, x_i) to G;
- add the edges (t_{x_i}, c_j^1),(t_{x_i}, c_j^2) and (t_{x_i}, c_j^3) if and only if the clause C_j contains the literal X_i;
- add the edges (f_{x_i}, c_j^1),(f_{x_i}, c_j^2) and (f_{x_i}, c_j^3) if and only if the clause C_j contains the literal $\overline{X_i}$;
- include in W the vertices w_f^1, w_f^2, $w_{x_i}^1$, $w_{x_i}^2$, c_j^1, c_j^2, and c_j^3 (for all i, j).

First, we will prove that if F is satisfiable then the associated graph G contains a connection tree T with $r = 2n+1$ routers and $\ell = 2$ linkers. By construction, the gadget g_f must belong to T. For all i we add to T edges (f, x_i) and $(x_i, w^1_{x_i})$. Let A be a truth assignment for F. Assume that every variable makes at least one clause of F true (it is easy to see that such an assignment always exists if F is satisfiable). Add edge (x_i, t_{x_i}) to T if $X_i =$`true` in A, otherwise add edge (x_i, f_{x_i}). If t_{x_i} (or f_{x_i}) belongs to T then add edge $(t_{x_i}, w^2_{x_i})$ (or $(f_{x_i}, w^2_{x_i})$) to T. For each vertex t_{x_i} (or f_{x_i}) in T, choose a new vertex c^k_j and insert an edge between t_{x_i} (or f_{x_i}) and c^k_j in T. (Vertices c^k_j are leaves of T.) Finally, for each vertex c^k_j not yet included in T, insert in T an edge of G connecting c^k_j to some vertex t_{x_i} or f_{x_i} already in T. At this point, T is a connection tree for W containing the two linkers in g_f, and having as routers vertices $f, x_1, x_2, \ldots, x_n$ and t_{x_i} or f_{x_i} for each x_i.

Conversely, if G contains a connection tree for W using at most $r = 2n + 1$ routers and $\ell = 2$ linkers, then by construction T uses l_1 and l_2 as linkers. Since x_i is the only neighbor of $w^1_{x_i}$ in G, x_i is a router in T (for all i). Every vertex x_i in T is incident to only one of the edges (x_i, t_{x_i}) and (x_i, f_{x_i}), for otherwise this would imply the existence of more than $2n + 1$ routers. In other words, if it is stated that either t_{x_i} or f_{x_i} is a router for every i because of $w^2_{x_i}$, which added to f and the vertices x_i give at least $2n + 1$ routers, directly implying that f_{x_i} is not in T if t_{x_i} is in T (or vice versa). Consequently, every x_i has degree 3 and is adjacent to f. Every vertex c^k_j in T is adjacent to exactly one vertex, for otherwise T would contain a cycle. Thus, we can construct an assignment A for F as follows: set $X_i =$`true` if and only if t_{x_i} belongs to T. Since every c^k_j is adjacent to a vertex t_{x_i} or f_{x_i}, by construction A is a truth assignment.
□

Figure 1 shows, in (a), the graph G constructed from the formula $F = (x_1 + x_2 + x_3).(\overline{x_1} + \overline{x_2} + \overline{x_3}).(x_1 + \overline{x_2} + x_3)$, and in (b) a connection tree T of G for W using $\ell = 2$ linkers and $r = 7$ routers.

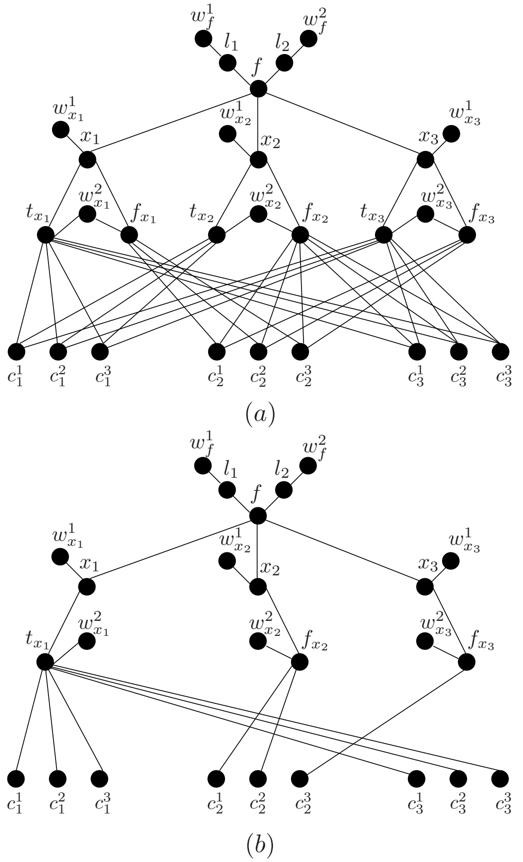

Figure 1: (a) The graph G associated to formula F; (b) a connection tree T for W.

Theorem 2.2. TCP(r) is NP-complete.

Proof. This proof uses a reduction from a very interesting problem described in [3]:

DEGREE CONSTRAINED SPANNING TREE - DCST

Instance: A connected graph H and a positive integer k.

Question: Is there a spanning tree for H in which no vertex has degree larger than k?

According to [3], DCST remains NP-complete for any fixed $k \geq 2$. Denote by DCST(2) the version of DCST for $k = 2$, which is equivalent to the HAMILTONIAN PATH PROBLEM [3].

From an instance H of DCST(2), create an instance G of TCP(r) for $r = 0$ as follows. (The proof can be easily adapted for other values of r).

(1) Initially, set $G = H$.
(2) For each vertex $v_i \in V(H)$ do: (a) create two vertices v_i' and v_i'' in G, with the same neighborhood as v_i in the current graph G; (b) remove every edge incident to v_i in the current graph G; (c) add edges (v_i, v_i') and (v_i, v_i'') to G.
(3) set $\ell = 2n$, $r = 0$, and $W = \{v_i \mid 1 \leq i \leq n\}$.

Note that all the vertices in W have degree two in G.

If H contains a spanning tree T' in which no vertex has degree larger than 2, then we can construct a connection tree T for W using at most $\ell = 2n$ linkers and $r = 0$ routers. For each edge (v_i, v_j) in T', it suffices to add edges $(v_i, v_i'), (v_i', v_j'), (v_j', v_j)$ to T. (If v_i' or v_j' already belongs to T, replace it by v_i'' or v_j'', respectively).

Conversely, from a connection tree T for W using at most $2n$ linkers and no routers, we can construct a spanning tree T' of H (which is a Hamiltonian path) by adding an edge (v_i, v_j) to T' if and only if there is a path in T between v_i and v_j whose internal vertices are all linkers. $\square$

Theorem 2.3. TCP(ℓ, r) can be solved in polynomial time.

Proof. The proof is based on the following simple algorithm: for each pair L, R of subsets of the input graph such that $L, R \subseteq V(G) \backslash W$, $L \cap R = \emptyset$, $|L| \leq \ell$, and $|R| \leq r$, perform the steps below:

(1) $G' = G[W \cup L \cup R]$

(2) **for each** spanning forest T of G' obtained by choosing two edges of G' incident on v for every $v \in L$ and three edges of G' incident on v for every $v \in R$ and such that $deg_T(v) = 2$ for every $v \in L$ **do**

if T is connected **then** return T **else**

let $S = E(G')\setminus(E(T) \cup E(G'[L]))$;

while T is not connected or S is not empty **do**

remove an edge e of S and insert e in T, provided that $T + e$

contains no cycle

if T is connected **then** return T

The above algorithm returns a connection tree for W using the vertices in L as linkers and the vertices in R as routers, if any. The algorithm considers $O(n^\ell n^r)$ pairs of subsets L, R. Line (2) considers $O(n^{2\ell} n^{3r})$ subgraphs T, where an $O(n)$ time is needed to check whether each T is spanning and acyclic. The remaining operations are easily done in $O(n + m)$ time. The overall complexity is therefore $O(n^{3\ell+4r+1}m)$.

References

[1] A. Y. Levin. Algorithm for shortest connection of a group of graph vertices. *Sov. Math. Dokl.* 12 (1971) 1477–1481.

[2] S. E. Dreyfus and R. A. Wagner. The Steiner problem in graphs. *Networks* 1 (1972) 195–207.

[3] M. R. Garey and D. S. Johnson, *Computers and Intractability- A Guide to the Theory of NP-Completeness*, Bell Telephone Laboratories Incorporated, 1979.

[4] S. L. Hakimi. Steiner's problem in graph and its implications. *Networks* 1 (1971) 113–133.

[5] R. C. Prim. Shortest connection networks and some generalizations. Bell System Technology Journal 36 (1957) 1389–1401.

Mitre C. Dourado Rodolfo A. Oliveira
Universidade Federal do Rio de Janeiro,
Brazil.
mitre@dcc.ufrj.br;
inurao@yahoo.com.br

Fábio Protti
Universidade Federal Fluminense,
Brazil.
fabio@ic.uff.br

Uéverton S. Souza
Centro Federal de Educação Tecnológica Celso Suckow da Fonseca,
Brazil.
usouza@ic.uff.br