Acta Cybernetica, Vol. 11, No. 3, Szeged, 1994

Partitioning Graphs into Two Trees*

Ulrich Pferschy [†] Gerhard J. Woeginger[‡] En-Yu Yao[§]

Abstract

We investigate the problem of partitioning the edges of a graph into two trees of equal size. We prove that this problem is NP-complete in general, but can be solved in polynomial time on series-parallel graphs.

1 Introduction

In this note, we will examine the partitioning problem PG2T defined as follows.

PARTITIONING GRAPHS INTO TWO TREES (PG2T)

Input. A graph $G = (V, E)$.

Question. Does there exist a partition of $E = E_1 \cup E_2$ with $|E_1| = |E_2|$, $V_1, V_2 \subseteq V$, such that the two edge-induced subgraphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ of G both are trees ?

If the trees G_1 and G_2 are required to be *spanning* trees of G, the problem can be solved in polynomial time by matroid partitioning techniques, see Lawler [4]. In contrast to this polynomial time result, we will show that detecting a partitioning into two arbitrary (not necessarily spanning) equal-sized trees is NP-complete. Our reduction is done from the Hamiltonian Path problem in cubic graphs (Garey and Johnson [2]). To simplify the presentation, we will introduce an intermediate problem TCT (defined below) and prove that it is also NP-complete.

On the positive side, we will show that PG2T is polynomial time solvable for the class of series-parallel graphs.

The paper is organized as follows: Section 2 presents the NP-completeness result, Section 3 gives the polynomial time algorithm for series-parallel graphs and Section 4 finishes with the discussion.

*This research was partially supported by the Christian Doppler Laboratorium für Diskrete Optimierung and by the Fonds zur Förderung der wissenschaftlichen Forschung, Project P8971-PHY.

[†]TU Graz, Institut für Mathematik B, Kopernikusgasse 24, A-8010 Graz, Austria

[‡]TU Graz, Institut für Theoretische Informatik, Klosterwiesgasse 32/II, A-8010 Graz, Austria

[§]Mathematical Department, Zhejiang University, Hangzhou, People's Republic of China

2 Why the problem is NP-complete

In this section, we prove that PG2T is NP-complete. The proof is done by a two-step reduction from the following special case of the Hamiltonian Path problem.

HAMILTONIAN PATH IN CUBIC GRAPHS (HP3)

Input. A graph $G' = (V', E')$ such that all vertices in V' are of degree three with the exception of the degree one vertices s and t.

Question. Does there exist a Hamiltonian Path of G' that starts in s and ends in t ?

TWO COVERING TREES PROBLEM (TCT)

Input. A graph $G'' = (V'', E'')$, two disjoint subsets F_1 and F_2 of E''.

Question. Do there exist two edge-disjoint trees T_1 and T_2 in G'' such that T_i contains all edges in F_i, $i = 1, 2$?

To be precise, we will show that HP3 is polynomial time reducible to TCT, and then that TCT is polynomial time reducible to PG2T. This clearly establishes the NP-completeness result claimed above.

Hence, let us consider some instance $G' = (V', E')$, $s, t \in V'$ of HP3. We will construct a corresponding instance of TCT that is solvable if and only if HP3 is solvable. This is done in three steps as follows.

(i) First, we subdivide every edge $e = (u, v) \in E'$ into two subedges $(u, e(m))$ and $(e(m), v)$ by introducing a new vertex $e(m)$. Furthermore we introduce a single new vertex c. Vertex c is connected to all vertices $e(m)$ by an edge which is put into F_2.

(ii) We perform the following construction for every $v \in V'$ of degree three: Two new vertices v^* and $\tilde{v}$ are introduced together with the two edges (v, v^*) and $(v, \tilde{v})$. The edge (v, v^*) is put into F_1, the edge $(v, \tilde{v})$ into F_2.

(iii) Finally, we introduce two new vertices s^* and t^* and two edges (s, s^*) and (t, t^*) that are both put into F_1.

We claim that the designed instance of TCT is solvable if and only if G' has a Hamiltonian Path.

(If): Assume, a Hamiltonian Path exists. Our tree T_1 simply consists of all edges in F_1 together with all subdivided edges of the Hamiltonian Path (i.e. if the edge $e = (u, v)$ is in the Hamiltonian Path, we put the two edges $(u, e(m))$, $(e(m), v)$ into the tree). It is trivial to check that this edge set is connected, without cycles and contains all edges in F_1.

Hence, it remains to show that the set E^* of remaining edges also forms a tree. First, we will argue that E^* is connected. Consider some vertex v of V' and the three incident subedges $(v, e_1(m))$, $(v, e_2(m))$ and $(v, e_3(m))$. The Hamiltonian Path uses exactly two of the edges e_1, e_2 and e_3. Therefore, the edge $(v, \tilde{v})$ is connected to vertex c via the unused edge.

E^* contains all edges $(c, e_i(m))$ incident to c. Some of the vertices $e_i(m)$ are of degree one in E^*, some of them are incident to two edges $(u, e_i(m))$ and $(v, e_i(m))$.

Finally, there are the corresponding edges (u, u^*) and (v, v^*) appended to u respectively v. Hence, E^* is a tree of radius three with center c and the proof of the (If)-part is complete.

(Only if): Now we assume that the TCT-instance is solvable. Consider T_1 and call an edge $e = (u, v)$ in E' *complete* iff both subedges $(u, e(m))$ and $(e(m), v)$ are in T_1. We claim that the complete edges constitute a Hamiltonian Path in G'.

Every degree three vertex v in G' is incident to at most two complete edges (otherwise, the edge $(v, \bar{v})$ in F_2 would be separated from T_2). Vertices s and t are incident to exactly one complete edge.

We remove from T_1 all edges that are neither in F_1 nor subedge of a complete edge. It is easy to check that these removals cannot disconnect T_1. Then we remove all edges in F_1 and replace the remaining subedges by the corresponding complete edges. Since each vertex is of degree at most two and since s and t are of degree one, the resulting graph is a path spanning all vertices in V'. This completes the proof of the (Only If)-part.

What we proved till now suffices to establish the NP-completeness of TCT. However, we are interested in proving the NP-completeness of PG2T, and to this end we need the following lemma.

Lemma 2.1 *Given an instance of* HP3*, we can compute in polynomial time an instance $G'' = (V'', E'')$, F_1, F_2 of* TCT*, such that* HP3 *is solvable iff* TCT *is solvable and such that the following four conditions hold.*

(C1) TCT *is solvable if and only if there exist two edge-disjoint connected subgraphs S_1 and S_2 such that S_i contains all edges in F_i, $i = 1, 2$.*

(C2) If TCT *is solvable, then there exists a solution that uses all edges in E''.*

(C3) $|F_1|$ and $|F_2|$ are two distinct prime numbers.

(C4) F_1 and F_2 both contain at least one edge with one endvertex of degree one.

Proof. To see (C1), we just have to check that in the proof of the (Only If)-part above, we did not exploit the fact that T_1 is a tree but only the connectedness of T_1. (C2) follows from the proof of the (If)-part.

To ensure that (C3) and (C4) hold, we first compute a prime p_1 with $|F_1| < p_1 < 2|F_1|$. Such a prime exists by Chebyshev's theorem. The prime can be computed in polynomial time, since $|F_1|$ is unary encoded by enumerating its elements. By similar arguments, we can find another prime $p_2 \neq p_1$ with $|F_2| < p_2 < 4|F_2|$.

Then for $i = 1, 2$, we take an edge $e_i = (v_i, u_i) \in F_i$, create $p_i - |F_i|$ new vertices for V'' and connect all these new vertices to v_i by new edges that are added to F_i. Obviously, this new instance of TCT fulfills (C3) and (C4), it is solvable if and only if the original instance was solvable, and conditions (C1) and (C2) still hold. □

Now we consider an instance $G'' = (V'', E'')$, $F_1, F_2 \subseteq E$ of TCT as described in the statement of Lemma 2.1. We construct an instance of PG2T that is solvable exactly if TCT is solvable. Our construction is as follows (let $n = 2|E''|$, $p_1 = |F_1|$, $p_2 = |F_2|$).

(i) We subdivide every edge e in E'' by a new vertex $v(e)$. If $e \in F_1$, we append to $v(e)$ a path of length $p_2 n^2$. Similarly, if $e \in F_2$ then we append to its subdividing vertex a path of length $p_1 n^2$.

(ii) Let $e_i = (u_i, v_i) \in F_i$ with degree of v_i equal to one, $i = 1, 2$, denote the two edges that exist by (C4). We connect v_1 and v_2 by a path of length $2|E''|$.

Clearly, the size of the new graph $G = (V, E)$ is polynomial in the size of G'', and the construction can be performed in polynomial time. The total number $|E|$ of edges in the new graph is $2p_1p_2n^2 + 4|E''|$. We claim that the designed instance of PG2T is solvable if and only if TCT has a solution.

(If): Let T_1 and T_2 constitute a solution of TCT that uses all edges in E''. Let n_1 denote twice the number of edges in T_1, $0 < n_1 < 2|E''|$. We put into E_1 all the edges corresponding to T_1, i.e. subdivided edges of G'' and the corresponding appended paths. Moreover, we put into E_1 the $2|E''| - n_1$ edges of the path defined in (ii) that are nearest to v_1.

Thus, E_1 contains p_1 appended paths with p_2n^2 edges per path, together with a number n_1 of subdivided edges from G'', together with $2|E''| - n_1$ edges from the path defined in (ii). This gives a total number of $p_1p_2n^2 + 2|E''| = |E|/2$ edges in E_1. It is easy to see that E_1 is cyclefree and connected, since T_1 is cyclefree and connected. The same holds for $E - E_1$.

(Only If): Assume, the PG2T-instance has a solution E_1, E_2. Each of the appended paths defined in (i) is contained as a whole either in E_1 or in E_2.

We claim that all p_1 paths of length p_2n^2 are in one of the E_i, and all p_2 paths of length p_1n^2 are in E_{3-i}. Suppose otherwise: Let E_1 contain x_1 paths of length p_1n^2 $(0 < x_1 \leq p_2)$ and x_2 paths of length p_2n^2 $(0 < x_2 \leq p_1)$. Then the facts $0 < x_1 \leq p_2$ and $0 < x_2 \leq p_1$ imply $x_1p_1n^2 + x_2p_2n^2 \neq p_1p_2n^2$. W.l.o.g. E_1 contains at least as many edges of the appended paths as E_2. This yields a contradiction, since

$$|E_1| \geq p_1p_2n^2 + n^2 > p_1p_2n^2 + 2|E''| = |E|/2.$$

To complete the proof, we show that there exists a connected subgraph S_1 of G'' that covers F_1. Using a symmetric argument for F_2 and condition (C1) of Lemma 2.1, this implies the existence of a solution to the TCT-instance. W.l.o.g. let E_1 connect all appended paths corresponding to edges in F_1. We define S_1 to contain all edges in F_1 together with all edges in E'' for which *both* subedges are in E_1 (if only one of the subedges is in E_1, it cannot contribute to the connectivity of E_1). It is easy to check that S_1 is connected.

Summarizing, we have proved the following theorem.

Theorem 2.2 *The problem* PG2T *is NP-complete.* □

3 Series-parallel graphs are easy to treat

The class of series-parallel graphs is a well-known model of series-parallel electrical networks. Many difficult combinatorial problems for graphs become easy when restricted to series-parallel graphs, see e.g. Tamkamizawa, Nishizeki and Saito [5]. In this section, we show that the same holds for the partitioning problem of a graph into two trees, i.e. we will give a polynomial time algorithm for this problem on series-parallel graphs.

One possibility to define series-parallel graphs is via two-terminal graphs, cf. Duffin [1]. A *two-terminal graph* $G = (V, E)$ is a graph with two special vertices

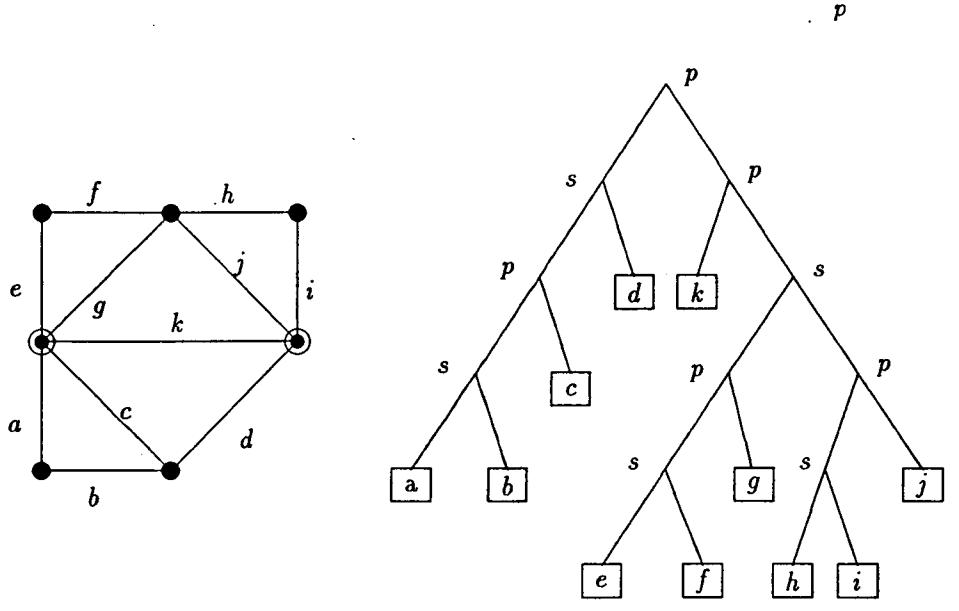

Figure 1: A TTSP graph and its binary decomposition tree

that are called the *left terminal* t_l and the *right terminal* t_r. For two-terminal graphs $G_i = (V_i, E_i)$ with terminals t_l^i and t_r^i, $1 \leq i \leq 2$, we define the following two operations.

- The *series connection* $G_s = G_1 * G_2$ of G_1 and G_2 results from identifying the right terminal of G_1 with the left terminal of G_2. The obtained graph G_s is regarded as a two-terminal graph with with left terminal t_l^1 and right terminal t_r^2.
- The *parallel connection* $G_p = G_1 // G_2$ of G_1 and G_2 results from identifying both right terminals with each other and both left terminals with each other. The terminal vertices of G_p simply are the identified terminals.

Now a *two-terminal series-parallel graph* (TTSP) is defined as follows:

(i) The graph consisting of two terminals connected by a single edge is a TTSP.

(ii) If G_1 and G_2 are TTSPs, then $G_1 * G_2$ and $G_1 // G_2$ are TTSPs.

(iii) No other graphs than those defined by (i) and (ii) are TTSPs.

Finally, a graph is a *series-parallel graph* iff it is the underlying graph of a TTSP (i.e. the terminals are considered as ordinary vertices).

It is well-known that decomposing a series-parallel graph into its atomic parts according to the series and parallel operations can be done in linear time. Essentially, such a decomposition corresponds to a binary tree where all interior vertices are labeled by s or p (series or parallel connection) and where all leaves correspond to edges of the graph (see Figure 1 for an illustration). We associate with every interior vertex v of the decomposition tree the series-parallel graph $G(v)$ defined by the subtree rooted in v.

The usual way to deal with problems on series-parallel graphs is dynamic programming via the decomposition tree, and this approach also works in our case.

Let us consider a TTSP graph $G = (V, E)$, and one of the TTSP components $G(v)$ of G associated with one of the vertices v of the decomposition tree of G, and let t_l and t_r denote the terminals of $G(v)$. Let T be a subtree of G, and let T' denote the edge-induced subgraph of T induced by the edges in $T \cap G(v)$. We distinguish five combinatorial types for T'.

(T1) T' consists of two connected components, one containing terminal t_l and the other one containing t_r.

(T2) T' is connected and contains both terminals t_l and t_r.

(T3) T' is connected and contains only terminal t_l but not t_r.

(T4) T' is connected and contains only terminal t_r but not t_l.

(T5) T' is connected and contains neither t_r nor t_l.

Clearly, type (T1) covers the only possibility of not connected T' (In this case, T can only be connected via some path going from t_l to t_r outside of $G(v)$). The remaining four types (T2), (T3), (T4), and (T5) cover all possibilities for connected graphs T'. Note that a T' of type (T1) consists of exactly two trees, and a T' of one of the other types is a tree itself.

We introduce twenty-five two-dimensional boolean arrays $A_{ij}[v, m]$, $1 \leq i, j \leq 5$. The first index v runs through all vertices of the binary decomposition tree, the second index m runs from 1 to $|V|$. $A_{ij}[v, m]$ will be set to TRUE if and only if

there exists a partition of $G(v)$ into two edge-disjoint subgraphs T_1' and T_2' such that T_1' is of type (Ti) and T_2' is of type (Tj) with respect to $G(v)$, and such that T_1' has exactly m edges.

If we compute the truthvalues of all entries of all arrays $A_{i,j}[*,*]$, we solve the PG2T-problem as a by-product: The root r of the decomposition tree corresponds to the graph $G = (V, E)$ itself. The problem PG2T has a solution if and only if $|E|$ is even and at least one of the sixteen entries $A_{ij}[r, |E|/2]$ with $2 \leq i, j \leq 5$ is set to true.

Hence, our goal is to compute all entries of the array. This is done in a bottom-up fashion according to the decomposition tree: We start with the entries corresponding to leaves of the decomposition tree, and move up towards the root. The entries corresponding to some vertex v of the decomposition tree are calculated only if all entries corresponding to both sons have already been computed.

The initialization step is trivial, since the leaves of the decomposition tree correspond to TTSPs consisting of a single edge.

The computation of entries corresponding to interior vertices v of the decomposition tree is a little bit more complicated and depends on whether v is labeled s or labeled p. We just sketch two of the 50 possible cases and leave the other cases to the reader as an exercise. (Some combinations like $A_{55}[*,*]$ will only have entries set to FALSE).

(1) Computation of $A_{11}[v, m]$ if v is labeled s: Let v_1 and v_2 denote the right and left son of v. In this case, T_1 may consist of (i) a not-connected part of type (T1) in $G(v_1)$ and a connected part of type (T2) in $G(v_2)$ (or the symmetric possibility with $G(v_1)$ and $G(v_2)$ exchanged), or (ii) of a part of type (T2) or (T3) in $G(v_1)$ and a part of type (T4) in $G(v_2)$ (or again some symmetric possibilities). The same possibilities hold for T_2.

We just check whether there exist corresponding true entries $A_{ij}[v_1, m_1]$ and $A_{kl}[v_2, m_2]$, where m_1, m_2 denote two non-negative integers with $m_1 + m_2 = m$ and i, j, k, l correspond to appropriate types as explained above.

(2) Computation of $A_{52}[v, m]$ if v is labeled p: Again, let v_1 and v_2 denote the right and left son of v. In this case, T_1 must consist of a part of type (T5) in $G(v_1)$ and $G(v_2)$ and of an empty part in the other subgraph. T_2 must consist of a part of type (T2) in $G(v_1)$ and a part that is not of type (T5) in $G(v_2)$ (or vice versa). Similarly as above, $A_{25}[v, m]$ can be computed by investigating appropriate entries $A_{ij}[v_1, m_1]$ and $A_{kl}[v_2, m_2]$, with numbers $\{m_1, m_2\} = \{0, m\}$.

Since all the operations used in the computations of the $A_{ij}[v, m]$ can be performed in polynomial time, we may formulate the following summarizing theorem.

Theorem 3.1 *The problem* PG2T *is solvable in polynomial time if the graph under consideration is series-parallel.* □

4 Discussion

In this paper, we proved that the problem of partitioning a graph into two trees is NP-complete in general, and that the problem is polynomial time solvable for the class of series-parallel graphs.

A similar but simpler version of the dynamic programming approach used for series-parallel graphs in Section 3 succeeds to show that the problem can be solved in polynomial time for trees.

The problem is also polynomial time solvable on the classes of interval graphs, cographs, circular arc graphs, chordal graphs and split graphs (see Johnson [3] for definitions). These results are rather easy to see: The graphs in these graph classes tend to be rather dense and to contain large cliques, whereas a graph $G = (V, E)$ that is partitionable into two trees must fulfill $|E| \leq 2|V| - 2$ and cannot contain cliques of size greater or equal to five. Consequently, most of the graphs in these classes may be a priori disregarded, whereas the remaining 'reasonable' graphs possess a rather rigid and primitive structure. (E.g. a 'reasonable' split graph consists of a clique C with at most four vertices, an independent set I and some edges between C and I).

We do not elaborate on these questions. The surprising part of our results is not that the problem is easy on specially structured graphs, but that the problem is hard in general.

References

[1] R.J.Duffin, Topology of series-parallel networks, *J. Math. Applic.* **10**, 1965, 303–318.

[2] M.R.Garey and D.S.Johnson, *Computers and Intractability, A guide to the theory of NP-completeness*, Freeman, San Francisco, 1979.

[3] D.S.Johnson, The NP-completeness column: an ongoing guide, *J. Algorithms* **6**, 1985, 434–451.

[4] E.Lawler, *Combinatorial Optimization, Networks and Matroids*, Holt, Rinehart and Winston, New York, 1976.

[5] K.Tamkamizawa, T.Nishizeki and N.Saito, Linear-time computability of combinatorial problems on series-parallel graphs, *J. Assoc. Comput. Mach.* **29**, 1982, 623–641.

Received August 30, 1993