On decorating Christmas trees

Christian Hagemeier*
Bernhard Fuchs Dirk Räbiger Britta Wienand
ZAIK / University of Cologne, Weyertal 80, D–50931 Koeln, Germany

March 24, 2004

Abstract

The following decision problem is regarded in this note: given a tree, decide if it is possible to cover exactly k nodes of the tree with stars (that is with trees of depth 1. We give a proof of the polynomiality of the problem which directly leads to a linear time algorithm.

1 Introduction

A *graph* is a pair $G = (V, E)$, where V is a set of nodes, $|V| = n$ and E a set of pairs (v_1, v_2) of nodes, called edges, $|E| = m$. Two nodes connected by an edge are called *adjacent*. The number of nodes adjacent to v is called degree of v. A sequence of nodes $v_1, ..., v_k \in V$, $\forall i < k : (v_i, v_{i+1}) \in E$, is called *path* iff $\forall i \neq j : v_i \neq v_j$. A *circle* is a path where $v_1 = v_k$. Graphs containing no circles are called *trees*. A *rooted tree* is a tree with a special node r called *root*. Nodes with degree one are called *leaves*, adjacent nodes all having degree 2 or less are called *chains*. The *depth* of a node v in a rooted tree is the length of the path from r to v, the *height* of v is the length of the longest path to a leaf within the subtree induced by v (that is the subgraph containing all nodes and edges beneath v, i.e. the subgraph that is generated by cutting off the first edge on the path from v to the root).

In the context of this note a rooted tree $S = (V, E)$ is called *star* iff the maximal depth of all nodes is 1 and $|E| \geq 2$. In the context of a star the root r is called *center* of the star, c for short, and all other nodes of S are called *sons*. Let v be a node, $S = (V, E)$ a star and c the center of this star; then $S \cup v$ is the star $S^\star = (V \cup \{v\}, E \cup \{(v, c)\})$.

The *Christmas tree problem* may now be stated as a node partitioning problem as follows: Given a tree $T = (V, E)$, does there exist a partition of T in stars $S_i = (V_i, E_i)$, $i = 1, ..., k$, such that $V = \dot{\bigcup}_{1 \leq i \leq k}$.

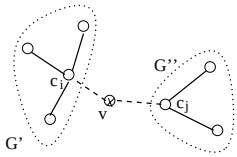

Figure 1: v is an optional node.

Given a tree $T = (V, E)$. A node v is called *optional in respect of* S_1 (or just *optional* if the context is clear), if there are two subgraphs $S_1 = (V_1, E_1)$ and $S_2 = (V_2, E_2)$ with $V_i \subset V$ and $E_i \subset E$, $i \in \{1, 2\}$, such that all graphs $S_1, S_2, S_1 \cup v, S_2 \cup v$ are stars. (See figure 1 for an illustration where the crossed circle illustrates the optional node.)

*corresponding author; email: hagemeier@zaik.de

2 Algorithm

Our algorithm to solve the Christmas tree problem sequentially constructs stars in two phases: Phase one constructs stars at locations within the graph when there is no other possibility of building the stars (namely: at leaves). Phase two creates stars for all other parts of the tree. Each time a star is created, the corresponding nodes and all incident arcs are deleted, resulting in a *reduced graph*.

Proposition 1 *Single optional nodes do not harm during the creation of the star partition of the tree.*

The reason is that an optional node v in respect of star S, which is a leaf in the reduced graph, may be ignored if necessary: $S \cup v$ is as well a star (because of the definition of optional nodes).

Proposition 2 *Chains of length at least three may be reduced modulo three.*

That is since the only possibility to partition those kind of nodes is to build stars with two edges. Therefore we may assume for the remainder of this note that there do not exist any chains longer than two nodes.

Proposition 3 *Not optional leaves in the tree enforce that their fathers are centers of stars.*

2.1 Phase 1: Graph reduction

The goal for our first phase is to build a reduced graph where all leaves are optional nodes. All actions done in this phase are enforced by the tree structure so that we do not have any possibility to choose another arrangement for the stars than demonstrated further on (we will show this in detail at each step taken). The consequence of this is if we result in a reduced graph that may not be partitioned by stars in phase 1, there is no other arrangement of stars that may be able to partition the tree correctly.

For phase 1 we need to choose a root for our tree in advance. Whatever root is chosen, it does not matter with regard to the solution of the Christmas tree problem. During the whole phase this root is not changed.

We iterate on the leaves of the reduced graph which are not (yet) optional, starting from those that have got the maximal depth in the tree, and then successively choosing the next deepest one. The order of nodes with the same depth does not matter. As induction assertion we handle the assumption that all leaves below the current leaf are optional nodes. If this assumption is met through all steps in phase 1, in the end all leaves are optional and thus phase 1 is done.

Before handling the complicated cases, note that two simple fundamental cases are able which may not be partitioned by stars:

The regarded leaf l has got no father: If l is the only node in the tree and is not optional, this tree may not be partitioned by stars.

l and its father f have got degree 1: It is again not possible to partition those two nodes with stars.

Given a leaf l (and its father called f) in the tree, all leaves that are deeper in the tree are optional ones. Now have a look at the father of this leaf and you will get the following different cases:

1. More than one son of f is a leaf (in the reduced graph).
2. The only leaf is l:
   (a) l is the only son of f.
   (b) There is at least one other son of f with height at least 2:
      i. The height of at least one son is 2.
      ii. The height of all sons is bigger than 2.

We now construct the stars containing the regarded leaves:

Case 1.: "More than one son of f is a leaf" From proposition 3 follows that f has to be the center of a star and l a son of it. All other leaves will be included into this star. The other sons and the father of f may be cut off and regarded as optional nodes, since they may be included into this star if needed.

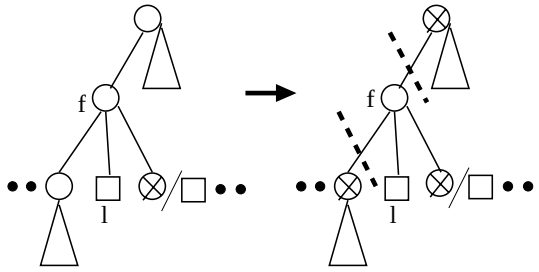

Figure 2: Situation of case 1.

The created star is a feasible one, since it has got at least two sons. Further on in all cut off trees under f are only leaves that are optional nodes (follows by induction). The optionality status of leaves of the remaining tree is not modified. Therefore the induction assertion is still valid.

For an illustration see figure 2. The crossed circles illustrate optional nodes, the triangles illustrate not further specified other parts of the tree (including the empty tree) and the dashed lines mark edges which are cut off.

Case 2.(a) "l is the only son of f": From proposition 3 follows that f has to be the center of a star and l a son of it. Since f has got degree 2, the only possibility to build a correct star with these nodes enforces to use the father of f as second son as well. Thus one has to cut off all other nodes from the father of f. (See figure 3)

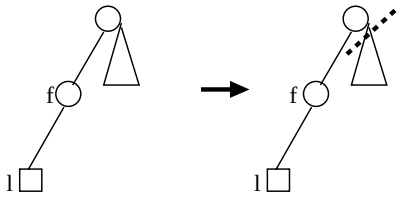

Figure 3: Situation of case 2.(a).

It may now be possible that we have created trees that are not able to be partitioned by stars but we have been forced to build our star in this way. Further on, all possibly new created leaves are above l, thus the induction assertion is still valid.

Case 2.(b)i. "The height of at least one son of f is 2": Again, proposition 3 forces f to be the center of a star. Since there exists at least one son with height 2, called s, all sons of s are leaves. But these leaves have got a higher depth and thus are optional by induction. Therefore all sons of s may be ignored by propositon 1 and s is the second son of the created star. All other subtrees below f may be cut off and the sons of f (except j, s) are marked as optional. The same occurs with the father of f. (See figure 4)

The created star is a feasible one. Further on all cut off trees only consist of optional leaves by induction.

Case 2.(b)ii. "The height of all sons of f is bigger than 2": Proposition 3 forces f and l into a star. Now consider each subtree induced by the sons of f (except l) separately (the sons of f will be called s_j). See figure 5 for illustration. Our Goal is to join one of the s_j

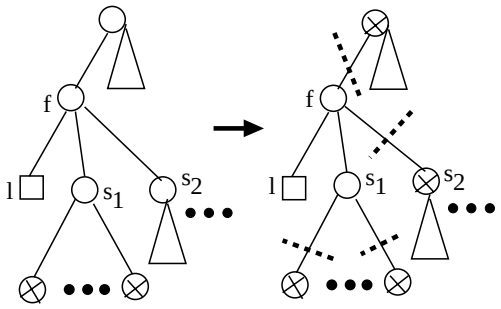

Figure 4: Situation of case 2.(b)i.

to the star centered in f. Now we will regard each s_j separately, starting with s_1. We handle each son of s_j (called t) separately, depending on the size of its induced subgraph by t:

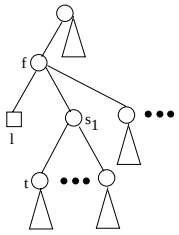

Figure 5: Illustration of case 2.(b)ii.

Subcase "1 node": This optional node may be cut off by proposition 1.

Subcase "2 nodes": In this situation s_j may be cut off from node f, since t enforces s_j either to be the center of a star or a son of the star around t.

Thus we create a new tree $T^\star$, but our induction assumption is still true: if s has got more than one son, all leaves of $T^\star$ are optional. If t is the only son of s_j, $T^\star$ ist a chain of length 3 that builds a single star.

Again this situation was enforced by the structure of the graph.

Subcase "≥ 3 nodes": If t has got degree at least 3, it may be cut off from s_j, because the resulting new tree has got by induction assumption only optional leaves.

Thus it remains that all t's have got degree 2: In this case create a new tree T' by cutting off t from s_j, one by one. Now we have got a new leaf which is not optional, but all other leaves in T' are optional. Now start this algorithm with tree T' and leaf t again.

If it is not possible to make all leaves optional, s_j has to be part of this tree T'. Thus s_j may not be used in the star around t. Since s_j has to be put into T' and is needed there, it may be marked as optional without harm. Thus all leaves in this tree are optional. Now check the other t's under s_j. If one of them needs s_j as well, it is not possible, thus the whole tree T is not able to be partitioned by stars. If no one else needs s_j, this situation is still possible, but s_j may not be used in the star around f. Thus we have to look at the other sons of f for another son for the new star around f.

On the other hand, if it is possible for all subtrees T' to make all new leaves t optional, s_j is not needed in any of them, thus it may be used as the second son of the star around f, and we are done. Cut off all other sons of f and mark them as optional.

If all other sons of f except l have to be cut off, using the argumentation from above, we have to use the father of f as the second son. Thus we cut off all nodes adjacent to the father of f.

Thus, again, we have shown that all leaves deeper than l (including the new cut off trees) are optional by induction and our chosen stars were enforced.

One remark to the running time of this step: If (in the last subcase) the algorithm is started again, this is done at most once for each subtree; after that, the whole subtree contains only optional leaves (and thus is not regarded any more in the first phase).

The last fact to show for phase 1 is that it always terminates in polynomial time: Since each case concerning leaf l runs in $O(n(l))$, with $n(l)$ the number of nodes involved in this case, and reduces the number of nodes by $n(l)$, and each leaf l is only concerned once it follows:

Proposition 4 *Phase 1 runs in linear time with respect to the number of nodes of T.*

2.2 Phase 2: Partitioning

It is easy to see that the choice of the root of a tree does not interfere with the solution of the Christmas tree problem. Thus we are able to change the root node during phase 2.

Since phase 1 produced trees where all leaves are optional, we use this fact by assuring that after each operation in phase 2 we are able to reconstruct this fact. And since we are able to do so and further on we reduce the size of our trees in each step, it is clear that phase 2 may always produce a correct partitioning in stars.

1. Among all nodes, choose one with at least degree 3 as root (called r). If there is none, go to step 6.
2. As long as there exist sons with degree not 2 and at least three sons of r are available, cut off son by son (called s) as follows:
   (a) If $\deg(s) = 1$, s is an optional node and thus may be cut off.
   (b) If $\deg(s) > 2$, s again has at least 2 sons and thus if it is cut off s is no leaf. Thus, cut off s.
   (c) The case $\deg(s) = 2$ will be regarded later on.

   Both cases result in new trees where all leaves are optional.
3. If $\deg(r) = 2$ go to step 1.
4. Otherwise look for the sons of r (called s_j) which build chains of length exactly 2 (i.e. with s_j and its son are the only nodes of this subtree):
   (a) There exists exactly one chain of length 2:
   This results in the fact that s_j will be the center of a star, together with its (optional) son and r as sons of the star. Cut off the remaining sons of r. All those nodes are optional, since if these nodes have to be included into the former created star, use r as center, s_j and the new node as sons of the star, and cut off the optional leaf under s_j.
   Thus all created new trees only have got optional leaves.
   (b) There are at least two chains of length 2:

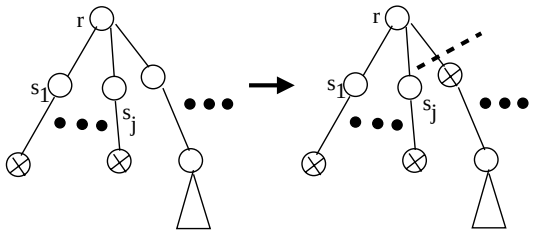

Figure 6: There exist at least two chains of length 2.

r will be the center of the star and the nodes called s_j will be the according sons. All sons of the nodes s_j are optional leaves and thus may be cut off. All other sons

of r may be cut off and marked as optional (since they may be included in the star around r).
Thus all created new trees only have got optional leaves.

(c) There does not exist any chain of length 2: I.e. the only son of s must have degree at least 3 or s is a member of a chain of length 3 (since longer chains would have been shrunken):

i. There is at most one s_j that is not part of a chain of length 3: cut off so many sons that are part of chains of length 3 so that r has got only 2 sons. Since all subtrees cut off are chains of length 3, they are stars and since all leaves in the remaining tree are optional, go to step 1.

ii. There is more than one s_j whose son has got degree at least 3: Cut off all chains of length 3. Cut off all sons of the remaining nodes s; since they still have got degree of at least 2, they are no leaves, so that all leaves of these new trees are optional. The remaining nodes s (at least 2) together with r as center are a star.

Thus all created new trees only have got optional leaves.

5. If still nodes with degree at least 3 exist, go to 1.
6. All resulting trees $T_i = (V_i, E_i)$ are chains with its leaves as optional nodes. After shrinking those chains, there are three possibilities:

$|V_i| = 1$**:** This single optional node may be included into its according star.

$|V_i| = 2$**:** Since both nodes are optional, both will be included in their corresponding stars.

$|V_i| = 3$**:** These three nodes will form a separate star.

This algorithm terminates, because it reduces the number of nodes with degree 3 in each iteration by exaclty one. Furthermore after each iteration the assumption is met that all leaves are optional.