CIRCUIT PARTITIONING WITH SIZE AND CONNECTION CONSTRAINTS

Yehoshua Perl * and Marc Snir **

DCS-TR-122

* Department of Computer Science
  Rutgers University
  New Brunswick, N.J. 08903

and

Bar-Ilan University
Ramat-Gan, Israel

** Department of Computer Science
Hebrew University
Jerusalem, Israel

December, 1982
CIRCUIT PARTITIONING WITH SIZE AND CONNECTION CONSTRAINTS

by

Yehoshua Perl

Department of Computer Science, Rutgers University, New Brunswick, N.J. 08903
and Bar-Ilan University, Ramat-Gan, Israel

and

Marc Snir

Department of Computer Science, Hebrew University, Jerusalem, Israel

ABSTRACT

The problem of partitioning a circuit into subcomponents with constraints on the size of each subcomponent and the number of external connections is examined. While this problem is shown to be NP complete even for very restricted cases, a pseudo-polynomial dynamic programming algorithm is given for the case where the circuit has a tree structure.
1. INTRODUCTION

An important problem in the realization of circuits is that of circuit packaging. In its simplest form it consists of allocating the vertices of a circuit to distinct boards so that the number of boards is minimized. Two constraints limit such partition: The area of each board restricts the number of vertices a board may contain, and the number of connectors restricts the number of external connections to the vertices on the board. The same problem, arises at the next level of integration, where circuits are packaged into discrete VLSI chips.

Formally the problem can be described as follows: We are given an undirected graph G = \langle V, E \rangle. Each vertex v ∈ V is assigned a weight \( w(v) \in \mathbb{Z}^+ \) (measuring the "size" of the vertex) and each edge e ∈ E is assigned a capacity \( c(e) \in \mathbb{Z}^+ \) (measuring the "thickness" of the line). For each subset \( U \subseteq V \) of vertices \( w(U) \), the weight of \( U \), is defined to be the sum of the weights of the vertices in \( U \), and \( c(U) \), the capacity of \( U \), is defined to be the sum of the capacities of the edges connecting vertices within \( U \) to vertices outside \( U \). Note that if weights and capacities have all unit value then \( w(U) \) is just the number of vertices in \( U \) and \( c(U) \) is the number of edges outgoing from \( U \).

A \((W_0, C_0)\)-partition of \( G \) is a partition of the vertices of \( V \) into components each of weight bounded by \( W_0 \) and capacity bounded by \( C_0 \). The WEIGHT AND CAPACITY CONSTRAINED GRAPH PARTITIONING problem (WCGP) consists of finding, for given graph \( G \) and bounds \( W_0 \) and \( C_0 \), a \((W_0, C_0)\)-partition of \( G \) with the least number of
components. We shall also consider the related WEIGHT AND CAPACITY CONSTRAINED CONNECTED GRAPH PARTITIONING PROBLEM (WCCGP). It is defined as above, with the additional restriction that the components are connected. A simpler problem, which has been considered by several authors, is the MINIMUM CAPACITY GRAPH PARTITIONING problem (MCGP) which consists of finding, for given graph G and bound $W_0$, a $(W_0,\infty)$-partition of G that minimizes the sum of the capacities of the components.

2. NP-COMPLETENESS RESULTS

Not surprisingly, the optimizations problems defined in the last section are not tractable in general. The intractability of the related MCGP problem has been shown in [HR] (see problem ND14 in [GJ]). With each of the two optimization problems we defined in the previous section we associate a decision problem -- that of deciding whether a partition with at most $k$ components can fulfill the constraints. We have

**Lemma 1.** The WCGP and WCCGP decision problems are NP-complete, even when restricted to graphs with unit edge capacities and polynomially bounded vertex weights.

**Proof:** The problem with no capacity constraints reduces to the BIN PACKING PROBLEM (see [GJ], problem SRI). Indeed, let $<U,s,B>$ be an instance of the bin packing problem, where $s(u)$ is the size of item $u \in U$, and $B$ is the bin capacity. Let $G$ be the complete graph with set of vertices $U$, where each vertex $u$ is assigned weight $s(u)$ and each edge is assigned capacity 1. Then the set of items $U$ can be packed into $K$ bins iff there exist a $(B,\infty)$-partition (or $(B,\infty)$ connected partition) of $G$ with $K$ components.
The additional capacity constraints enable us to obtain a much stronger theorem.

**Theorem 2.** The WCGP and WCCGP decision problems are NP-complete, even when restricted to graphs with unit weights and capacities and to fixed weight and capacity bounds.

**Proof:** We transform the VERTEX COVER problem ([GJ], problem GT1) for cubic graphs (i.e., regular graphs of degree three) to the WCGP (WCCGP) problem. A proof that VERTEX COVER is still NP-complete when restricted to cubic graphs is given in [5n].

Let $G = (V, E)$ be a cubic graph with $n$ vertices. Replace each edge $e = uv$ of $E$ with a distinct copy $G_e$ of a complete graph on 9 vertices, of which two are connected to $u$ and two others are connected to $v$. Append to each vertex $v \in V$ a complete graph $D_v$ on 27 vertices, by connecting one of its vertices to $v$. Figure 2 illustrates the graph obtained by applying this transformation to the graph represented in Figure 1.

Let $G'$ be the graph thus obtained. We assign to each vertex weight one and to each edge capacity one. We claim that the following three assertions are equivalent.

(i) $G$ has a vertex cover with no more than $k$ vertices.

(ii) $G'$ has a $(28,7)$-partition into no more than $k+n$ connected components.

(iii) $G'$ has a $(28,7)$-partition into no more than $k+n$ (not necessarily connected) components.
Indeed, let \( C \) be a set of \( k \) vertices covering all the edges of \( G \). We partition arbitrarily the edges of \( G \) into \( k \) disjoint subsets \( E_c \), so that the edges in \( E_c \) are incident with the vertex \( c \in C \). Define the following induced subgraphs in \( G' \):

1. \( S_v = \) if \( v \in C \) then \( D_v \) else \( D_v \cup \{v\} \), for each \( v \in V \).
2. \( S' = \{c\} \cup \{ O_e : c \in E_c \} \), for each \( c \in C \).

The \( k+n \) sets \( S_v, S' \) form a \((28,7)\)-partition of \( G' \) into connected components. Thus (i)+(ii). We also clearly have (ii)+(iii).

Let now \( \{S_i\} \) be a \((28,7)\)-partition of \( G' \) into \( k+n \) (not necessarily connected) components. The weight and capacity restrictions on the subsets of the partition impose the following facts:

1. Each subgraph \( D_v \) is completely contained in one component of the partition.
2. A component of the partition that contains a subgraph \( D_v \) may contain either the subgraph \( D_v \) only or the subgraph \( D_v \) with the addition of the vertex \( v \).
3. Each subgraph \( O_e \) is completely contained in one component of the partition.
4. If a component contains the subgraph \( O_e \), where \( e = uv \), and additional vertices, then it contains either \( u \) or \( v \), but not both.

It follows that the components \( S_i \) are of one of the following types:

1. \( D_v \), where \( v \in V \);
2. \( \{v\} \cup D_v \), where \( v \in V \);
3. \( O_e \), where \( ee \in E \);
4. \( \{v\} \cup \{E_v: e \in E_v\} \), where \( v \in V \) and \( E_v \) is a subset of the set of edges incident in \( G \) to \( v \).

If the partition has a minimal number of components then clearly no set containing only a subgraph \( O_v \) will occur in the partition. The partition will contain \( n \) components of the first or second type (one for each vertex \( v \in V \)) and \( k \) components of the last type. The set of \( k \) vertices in \( V \) corresponding to components of the last type forms a vertex cover for \( G \). Thus \((iii) \Rightarrow (i)\).

The weight and capacity bounds 28 and 7 can be decreased in the last theorem at the price of a somewhat more complex argument.

Thus, the WCGP and WCCGP problems are strongly NP-complete for general graphs, and cannot be solved by pseudo-polynomial algorithms. Quite often problems which are intractable for general graphs can be solved in polynomial time when restricted to graphs with a simple structure, such as trees. However, the proof of Lemma 1 implies that the WCGP (connectivity not required) is NP-complete, even when restricted to trees (or to any reasonable family of graphs). We can actually prove more:

**Lemma 3.** The WCGP problem is NP-complete even when restricted to trees with unit weights and capacities, with a fixed capacity bound.

**Proof:** We transform the 3-PARTITION problem to the WCGP problem ([GJ], problem SP15). Let \( \langle A, a, B \rangle \) be an instance of the 3-PARTITION problem, where \( A \) is a set of 3m elements, \( a(x) \) is the size of \( axA \), where \( B/4 < a(x) < B/2 \), and \( \Sigma a(x) = mB \). Since the 3-PARTITION problem is strongly NP-complete we can assume w.l.o.g. that the
sizes \( s(a) \) are all polynomially bounded. We also assume w.l.o.g. that \( B \) and \( s(a) \) are divisible by 10, for each \( a \in A \). Define a tree \( G \), as illustrated in Figure 3: \( G \) consists of a path containing the elements of \( A \) as vertices, where each vertex \( a \in A \) has \( s(a)-1 \) additional auxiliary vertices attached to it. We claim that \( G \) has a \((B,6)\) partition into \( m \) sets if and only if the original instance admits a solution, that is a partition of \( A \) into \( m \) disjoint subsets \( A_1, \ldots, A_m \), where for each \( A_i \)

\[
\sum_{a \in A_i} s(a) < B.
\]

Indeed, if such solution exists then let \( U_1 \) consist of the vertices in \( A_i \) and the associated auxiliary vertices. Then \( U_1, \ldots, U_m \) form a \((B,6)\)-partition for \( G \). Conversely, let \( U_1, \ldots, U_m \) be a \((B,6)\)-partition of \( G \) and let \( A_i = A \setminus U_i \). Out of the \( s = \sum_{a \in A_i} (s(a)-1) \) auxiliary vertices associated with vertices in \( A_i \) at least \( s-5 \) are in \( U_1 \) (otherwise \( c(U_1) > 6 \)). We thus have

\[
B > w(U_1) > \sum_{a \in A_i} (s(a)-1) - 5.
\]

(2.1)

Since \( s(a) > B/4 \), we have

\[
B > |A_i| \cdot (B/4) - 5, \text{ or } |A_i| < 4(B+5)/B < 5.
\]

Substituting back in Eq. 2.1 we obtain

\[
\sum_{a \in A_i} s(a) < B + 9.
\]

Since 10\( |B \) and 10\( |s(a) \), this implies

\[
B > \sum_{a \in A_i} s(a),
\]

and \( A(1), \ldots, A(m) \) solves the 3-PARTITION problem.

The WCCG problem cannot be solved therefore by a pseudo-polynomial algorithm, even when restricted to trees. Where components are required to be connected subtrees of the tree, the problem becomes simpler. We still have

**Lemma 4.** The WCCG decision problem restricted to trees is NP-complete.
PROOF: We shall prove that the problem of deciding whether there exists a feasible solution to the WCDDP problem is NP-complete. The proof uses a reduction to the KNAPSACK problem ([GJ], problem MP9). Let \( <U,s,v,B,K> \) be an instance of the KNAPSACK problem, where \( U \) is a set of elements, \( s(u) \) and \( v(u) \) are respectively the size and value of the element \( u \in U \), \( B \) is an upper bound on the size of a solution set, and \( K \) is a lower bound on the value of a solution. We can assume w.l.g. that \( s(u) < B \) and \( \sum_{w \neq u} v(w) > K \) for any \( u \in U \). Let \( G \) be the star graph consisting of the set of vertices \( U \) and one new vertex \( r \), where each vertex \( u \in U \) is connected by one edge to \( r \). Assign to \( r \) weight 1, to each \( u \in U \) weight \( s(u) \), and to each edge \( (r,u) \) capacity \( v(u) \). Let \( C = \sum v(u) - K \). Then \( G \) has a \( (B+1,C) \)-connected partition iff the associated knapsack problem admits a solution.

Indeed, Let \( V \) be a solution to the knapsack problem, so that

\[
\sum_{u \in V} s(u) < B \quad \text{and} \quad \sum_{u \in V} v(u) > K.
\]

Let \( V' = V \cup \{r\} \). Then \( w(V') < B+1 \) and

\[
c(V') = \sum_{u \in V} v(u) = \sum_{u \in V} v(u) - \sum_{u \in V} \leq \sum_{u \in V} v(u) - K = C.
\]

Thus there exists a \( (B+1,C) \)-connected partition of \( G \), consisting of \( V' \) and the one singleton set for each remaining element of \( U \). Conversely, every connected partition of \( G \) consists of one set \( V' \) containing \( r \), and singleton sets containing vertices of \( U \) not in \( V' \), where the set \( V' \cup U \) is a solution to the knapsack problem.

The knapsack problem can be solved in pseudo-polynomial time by dynamic programming, and above result is not valid anymore if either the weight constraint or the capacity constraint is polynomially bounded. Indeed, we show in the next
section that the WCCGP problem can be solved in polynomial time in either case.
The result is therefore the strongest possible one.

3. DYNAMIC PROGRAMMING PROCEDURE FOR TREE PARTITIONING

In this section we consider partitioning a tree into the least number of
subtrees satisfying given weight and capacity constraints.

Note that transforming the tree into a rooted tree does not change the
problem. Thus we may assume that a rooted tree is given.

Where only weight restriction is imposed then the problem can be solved in
linear time by a bottom-up scanning algorithm (see [KM]). This approach is not
applicable to our problem with two restrictions, as demonstrated by the
NP-completeness proof in Lemma 4. Indeed, an attempt to optimize with respect to
weight may conflict with optimizing with respect to the capacity constraint. The
two conflicting constraints can, however, be handled by a dynamic programming
procedure, similar to that used to obtain a pseudo-polynomial algorithm for the
knapsack problem. This procedure processes the vertices of the tree in end order,
computing optimal partitions for the subtree rooted at each vertex, where the
partitions associated with the subtree rooted at v depends only on the partitions
associated with the subtrees rooted at the children of v. A similar approach has
been used by Lukes [Lu] to solve the simpler MINIMUM CAPACITY TREE PARTITIONING
problem. We shall describe a dynamic programming procedure which is polynomial in
n, and C_0. A similar procedure exists which is polynomial in n and
Thus, if either $C_0$ or $W_0$ are polynomially bounded the problem can be solved in polynomial time. These two cases cover most applications.

The algorithm applies a bottom-up scanning algorithm to the (rooted) tree, calculating for each vertex $v$ and integers $k$ and $c$, $0 < k < n$, and $0 < c < C_0$, the minimum weight $W_k,c(v)$ of a top component of capacity $c$ in a legal partition of the subtree rooted at $v$ into $k+1$ components. The top component is the component containing the root $v$. In case no legal partition of the subtree rooted at $v$ satisfies these requirements we define $W_k,c(v) = \infty$. A partition of the tree into $k+1$ components is obtained by assigning cuts to $k$ edges of the tree.

Let $v$ have $d$ children $v_1, \ldots, v_d$ and assume $W_k,c(v_i)$ has already been computed for each $k, c$ and $i$. Let $B \subseteq \{1,2,\ldots,d\}$ denote the set of indices of the edges $e_i = (v,v_i)$ assigned a cut. We use also $c(v,v_i)$ for $c(e_i)$. Then

$$W_k,c(v) = \min\{w(v) + \sum_{i \in B} W_{k_i,c_i}(v_i)\},$$

where the minimum is taken over all subsets $B$ and and choices of $k_i$ and $c_i$ such that

(i) $\sum_{i \in B} c(e_i) + \sum_{i \in B} c_i = c,$
(ii) $|B| + \sum_{i=1}^d k_i = k,$ and
(iii) $W_{k_i,c_i}(v_i) \leq W_0$ for $i \in B$.

Denote by $r$ the root of the tree. Let

$$W^*_k(r) = \min_{0 < c < C_0} W_k,c(r)$$

and let $k > 0$ be the minimum index such that $W^*_k(r) < W_0$. Then $k+1$ is the minimum number of subtrees in a legal partition of the tree. The partition itself can be reconstructed by keeping appropriate pointers while computing the weights.
This dynamic programming approach is valid since it clearly satisfies the
principle of optimality (see e.g. [DL]). However the number of subsets \( B \) may be
exponential with the degree \( d \) and thus the complexity of the algorithm may not be
polynomial with \( n \). In order to obtain a pseudo-polynomial algorithm we shall
replace the straightforward computation of \( W_{k,c}(v) \) by a propagation process.

Let \( T_i(v) \), \( 1 \leq i \leq d \), denote the subtree rooted at \( v \) containing the children
\( v_1, \ldots, v_i \) of \( v \) and all their descendants. Denote by \( W_{k,c}(v,i) \) the minimum weight
of a top component of capacity \( c \) in a partition of the subtree \( T_i(v) \) into \( k+1 \)
legal components (if no such partition exists). Note that
\[
W_{k,c}(v) = W_{k,c}(v,d).
\]
We have
\[
(i) \quad W_{k,c}(v,i) = \infty \quad \text{if} \quad k < 0 \quad \text{or} \quad c < 0, \quad i=1,2,\ldots,d;
(ii) \quad W_{0,0}(v) = w(v), \quad \text{if} \quad v \quad \text{is a leaf};
(iii) \quad W_{0,0}(v,1) = W_{0,0}(v_1) + w(v);
(iv) \quad W_{0,0}(v,i) = W_{0,0}(v,i-1) + W_{0,0}(v_1), \quad i=2,\ldots,d;
(v) \quad W_{k,c}(v,i) = \begin{cases} \min \{ W_{k-1,c_1}(v_i) + W_{k-1,c_0-c}(v,v_1) \} & \text{if} \quad W_{k-1,c_1}(v_i) + W_{k-1,c_0-c}(v,v_1) < W_0 \quad \text{(If a cut may be assigned to} \quad (v,v_1) \quad \text{)} \\
\infty & \text{otherwise} \end{cases}
\]
For calculating \( W_{k,c}(v,i) \), \( i=2,\ldots,d \), we distinguish whether the edge \( (v,v_1) \)
is assigned a cut or not. In the first case we have
\[
W_{k,c}(v,i) = \min \{ W_{k-1,c_1}(v_i) + W_{k-1,c_0-c}(v,v_1) \},
\]
where the minimum is taken over all the indices \( k_1,c_1 \) such that
\[
0 < k_1 < k, \quad 0 < c_1 < c_0 - c(v,v_1), \quad \text{and} \quad W_{k_1,c_1}(v_1) < W_0.
\]
In the second case we have

\[ W_{k_1,c}(v,i) = \min_{k_1,c_1} \{ W_{k_1,c_1}(v,i) + W_{k-k_1,c-c_1}(v_i) \} \]

where \( 0 \leq k_1 \leq k \), \( 0 \leq c_1 \leq c \).

Finally

\[ W_{k,c}(v,i) = \min \{ W_{k_1,c_1}(v,i), W_{k_1,c}(v,i) \} \] .

4. COMPLEXITY ANALYSIS OF THE ALGORITHM

For computing \( W_{k,c}(v,i) \) \( O(kc) \) operations are required. Let \( d(v) \) denote the degree of \( v \). Then \( O(k \cdot c \cdot d(v)) \) operations are required to compute \( W_{k,c}(v) \). Note \( 1 \leq k \leq n \) and \( 1 \leq c \leq C_0 \). Thus \( O(n^2 \cdot C_0 \cdot c \cdot d(v)) \) operations are required to compute the weights for the vertex \( v \). For the whole tree the complexity is \( O(n^2 \cdot C_0 \cdot \sum_{v \in T} d(v)) \) \( O(n^2 C_0^2) \).

Hence the algorithm is pseudo-polynomial and in case \( C_0 \) is polynomial with \( n \), for example if edges have unit capacity, we obtain an algorithm of polynomial complexity. A similar pseudo-polynomial algorithm of complexity \( O(n^2 W_0^2) \) exists. Such an algorithm can be used when \( C_0 \) is not polynomially bounded but \( W_0 \) is. Note that the straightforward approach for a pseudo-polynomial algorithm consists of computing for each vertex \( v \), and each \( w \) and \( c \) the minimum number of components \( K_{w,c}(v) \) in a legal partition of the subtree rooted at \( v \) with a top component of weight \( w \) and capacity \( c \). This approach yields an algorithm with complexity \( O(n W_0^2 C_0^2) \). This algorithm is polynomial only if both \( W_0 \) and \( C_0 \) are both polynomially bounded, and is even more efficient if \( W_0 < n \) and \( C_0 < n \). For example if both \( W_0 \) and \( C_0 \) are fixed (a realistic assumption for many applications) we obtain a linear algorithm.
A variation of the algorithm is obtained by defining $W_{k,c}(v)$ to be the minimum weight of the top component in a legal partition with $k+1$ or less components and capacity $c$ or less at the top component. This variation has the same asymptotic complexity but saves part of the computations. The equation for $W_{k,c}(v,i)$ simplifies to

$$W'_{k,c}(v,i) = \min_{k_1} \{ W_{k-k_1-1,c-c(v,v_1)}(v,i-1) \},$$

where the minimum is taken over all indices $k_1$ such that

$$W_{k_1,C_0-c(v,v_1)}(v_1) < W_0.$$  

Also, part of the computation can be avoided as $W_{k,c}(v) = \infty$ implies $W'_{k,c}(v') = \infty$ for every $k' < k$, $c' < c$, and vertex $v'$ on the path from the root to $v$.

An upper bound on the space complexity of the algorithm is given by $O(n^2 C_0 d)$, where $d$ is the maximum degree in the tree. However only two sets of values $W_{k,c}(v,i)$ for two consecutive $i$'s are used simultaneously. Using also the end-order nature of the processing of the vertices of the tree it is sufficient to keep simultaneously the information for only one vertex for each level of the tree. Thus we obtain

$O(n C_0 \text{height}(T))$ space complexity. Note that $W_{k,c}(v,i) = \infty$ whenever $k \geq |T_i(v)|$. Thus, the number of values stored for $W_{k,c}(v,i)$ can be restricted to $C_0 |T_i(v)|$. Using this last remark we can further reduce the space complexity of the algorithm to $O(n C_0)$; since the relevant $T_i(v)$ trees are disjoint and $\sum |T_i(v)| < n$.

A related partitioning problem is that of finding a capacity constrained min-max weight partition of a tree, namely: given a tree $T$ and numbers $q$ and $C_0$, find a partition of $T$ into $q$ subtrees $T_1, \ldots, T_q$ such that $c(T_i) < C_0$ for $1 \leq i \leq q$ minimizing $\max_{1 \leq i \leq q} w(T_i)$. Let

$$W_0 = \min_{T_1, \ldots, T_q} \max_{1 \leq i \leq q} w(T_i),$$

where the minimum is over any partition satisfying $c(T_i) < C_0$, $1 \leq i \leq q$. Combining a binary search for the value of $W_0$ with applications of our algorithm for finding a legal partition into the minimum number of components, yields a
pseudo-polynomial algorithm for this problem. The time complexity of this algorithm is $O(n^3C_0 \cdot \log(w(T)))$, where $w(T)$ is the weight of the tree, that is the sum of the weights of the vertices. This algorithm is polynomial if $C_0$ is polynomially bounded.

A similar pseudo-polynomial algorithm can solve the size-constrained min-max weight partitioning problem shown in [ABP] to be $NP$-hard. On the other hand a polynomial shifting algorithm is presented there for the height constrained min-max weight partitioning problem.

5. LEFT-RIGHT DYNAMIC PROGRAMMING PROCEDURE

FOR TREE PARTITIONING

In a recent paper of Johnson and Niemi [JN] a non-standard approach called left-right dynamic programming is introduced. In this approach the vertices of the tree are ordered in depth-first search order and the dynamic programming procedure processes the vertices in an order which is a combination of the depth-first order and the bottom-up order. Johnson and Niemi apply this approach to the MINIMUM CAPACITY GRAPH PARTITIONING problem. Whereas the bottom-up dynamic programming procedure of Lueks for this problem had a complexity of $O(W_0^2n)$, the solution of Johnson and Niemi has a complexity of $O(W_0^2n^2)$.

The tree partitioning problem considered in this paper is more complex than the problem considered in [Lu] and [JN], as two constraints are involved. Nevertheless, the left-right dynamic programming approach can be applied to obtain
an $O(C_0^2n^2)$ procedure rather than the $O(C_0^2n^3)$ complexity of the previous bottom-up procedure, an improvement in case $C_0 > n$.

Let $v$ be a descendent of $u$ in the tree $T$. We define $T_i(u,v)$ to be the tree consisting of $T_i(v)$ together with all the nodes in $T(u)$ that precede $v$ in depth-first order. Denote by $W_{k,c}(u,v,i)$ the minimum weight of a component of capacity $c$ that contains both $u$ and $v$ in a partition of $T_i(u,v)$ into $k+1$ legal components ($W_{k,c}(u,v,i) = \infty$ if no such partition exists). Note that $W_{k,c}(u,0) = W_{k,c}(u,u,0)$ so that a solution for the optimization problem can be obtained once the values $W_{k,c}(r,r,d(r))$ have been computed for the root $r$ of the tree. We have

a. If $i=0$ then

(i) $W_{k,c}(u,u,0) = \begin{cases} w(u) & \text{if } k=0 \\ \infty & \text{else} \end{cases}$

(ii) If $v$ is the $j$-th child of a descendent $u'$ of $u$ then

$W_{k,c}(u,v,0) = W_{k,c}(u,u',j-1) + w(v)$.

b. If $i \neq 0$ and $v_1$ is the $i$-th child of $v$ then we have two cases, according as the edge $(v,v_1)$ is assigned a cut or not. In the first case we have

$W'_{k,c}(u,v,i) = \min\{ W_{k-k'-1,c-c(v,v_1)}(u,v,i-1) \}$,

where the minimum is taken over all the indices $k'$ such that

$W'_{k,c}(uv_1,v_1,d(v_1)) < W_0$.

In the second case we have

$W'_{k,c}(u,v,i) = W_{k,c}(u,v_1,d(v_1))$.

Finally,

$W_{k,c}(u,v,i) = \min\{ W'_{k,c}(u,v,i), W'_{k,c}(u,v,1) \}$.
Note that the values $w_{k,c}(u,v,i)$ are computed by handling the nodes $u$ in DFS order, and for each $u$, handling the nodes $v$ in the subtree rooted at $u$ in bottom-up order. A detailed analysis shows that this algorithm has time complexity $O(n^3 \cdot \text{height}(T) \cdot C_0)$ and space complexity $O(nC_0)$.

CONCLUDING REMARKS

In this paper we have considered the WEIGHT AND CAPACITY CONSTRAINED GRAPH PARTITIONING problem. We have shown this problem, and the related one of partitioning into connected components, to be strongly NP-complete, and gave a pseudo-polynomial time optimization algorithm for the latter problem in the case when the underlying graph is a tree. Although the problems we considered are related to the simpler MINIMUM CAPACITY GRAPH PARTITIONING problem, there does not seem to be a reduction from one to another. Moreover, there is one fundamental difference which is worth pointing at: The pseudo-polynomial time optimization algorithm that solves the MINIMUM CAPACITY GRAPH PARTITIONING PROBLEM can be used to derive a fully polynomial time approximation scheme for that problem; this is done using the "rounding" scheme [JN]. On the other hand the WCGP optimization problem can be solved in pseudo-polynomial time, but does not admit a polynomial approximation scheme. Indeed, as shown in Lemma 4, there is no polynomial algorithm to decide if the problem has a feasible solution, unless $P=NP$. 
REFERENCES


