Proceedings
of the
1982 Conference
on
INFORMATION SCIENCES AND SYSTEMS

Department of Electrical Engineering and Computer Science
Princeton University
Princeton, New Jersey 08544
CIRCUIT PARTITIONING WITH SIZE AND CONNECTION CONSTRAINTS
(Extended Abstract)
by
Yehoshua Perl
Department of Mathematics and Computer Science,
Bar-Ilan university, Ramat-Gan, Israel
and
Department of Computer Science, Queens College, New York
and
Marc Snir
Department of Computer Science, New York University, New York

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 of 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 $G = (V, E)$. Each vertex $v \in V$ is assigned a weight $w(v) \geq 0$ (measuring the "size" of the vertex and each edge $e \in E$ is assigned a capacity $c(e) \geq 0$ (measuring the "thickness" of the line). For each subset $U \subseteq V$ of vertices $w(U)$ is defined to be the sum of the weights of the vertices in $U$, and $c(U)$ 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 PARTITION problem (WGCP) 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 PARTITION problem (WCSCP).

2. NP-COMPLETENESS RESULTS

Not surprisingly, the optimizations defined in the last section are not tractable in general. The intractability of related problems has been shown by [HR] and [GJS] (see problems ND14 and ND17 in [GJ]). With each of these 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 WGCP and WCSCP decision problems are NP-complete, even when restricted to graphs with unit capacities and polynomially bounded weights.

**Proof:** The problem with no capacity constraints reduces to the BIN PACKING PROBLEM (see [GJ] for definition). Indeed, let $<S,B>$ be an instance of the bin packing problem, where $s(u)$ is the size of item $u$, and $B$ is the bin capacity. Let $G$ be the complete graph with set $V$ of vertices, 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 if and only if there exists a $(B,\infty)$-partition of $G$.

The additional capacity constraints enable us to obtain a much stronger theorem.

**Theorem 2.** The WGCP and WCSCP decision problems are NP-complete, even when restricted to graphs with bounded 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 WGCP (WCSCP) problem. A proof that VERTEX COVER is still NP-complete when restricted to cubic graphs is given in [Sn].

Let $G = (V, E)$ be a cubic graph with $n$ vertices. Replace each edge $e = uv$ of $E$ with a distinct copy $0_e$ 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 7 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 $k+m$ connected components.

(iii) $G$ has a $(28,7)$-partition into no more than $k+m$ (not necessarily connected) components.

Indeed, let $G$ be a set of $k$ vertices covering all the edges of $G$. We partition arbitrarily the edges of $G$ into $k$ disjoint subsets $E_i$, so that the edges in $E_i$ are incident with the vertex $c_i$. Define the following sets in $G'$:

1. $S_v = \{v\} \cup \{ U_e: e \in E_v \}$, for each $v \in V$.
2. $S_e = \{v\} \cup \{ U_e: e \in E_v \}$, for each $c_i$.

The $k+m$ sets $S_v$, $S_e$ form a $(28,7)$-partition of $G'$ into connected components. Thus (i)$\implies$(ii).

We also clearly have (ii)$\implies$(iii).

Let now $\{S_i\}$ be a $(28,7)$-partition of $G'$ into $k+m$ (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 $e \in E$;
4. $\{v\} \cup \{ U_e: e \in E_v \}$, where $v \in V$ and $E_v$ is a subset of the set of edges incident in $G$ to $v$.

If the partition is minimal then clearly no set containing only a subgraph $O_e$ 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)$\implies$(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 $\langle A, s, B \rangle$ be an instance of the 3-PARTITION problem, where $A$ is a set of $3m$ elements, $s(a)$ is the size of $aaA$, where $B/4 < s(a) < B/2$, and $s(a) = s$. 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 $aaA$. Define a tree $G$, as illustrated in Figure 3: $G$ consists of a path containing the elements of $A$ as vertices, where each vertex $aaA$ has $s(a)-1$ additional auxiliary vertices attached to it. We claim that $G$ has a $(B,6)$-partition into $m$ sets iff the original instance admits a solution, that is a partition of $A$ into $m$ disjoint subsets $A_1$,..,$A_m$, where for each $A_i$

$s(a) < B$.

Indeed, if such solution exists then let $U_1$,..,$U_m$ be a $(B,6)$-partition of $G$ and let $A_1 = A \cap U_1$. Out of the auxiliary vertices associated with vertices in $A_1$ at least $s-5$ are in $U_1$ (otherwise $c(U_1) > 6$). We can actually prove more:

**Lemma 4.** The WCGP decision problem restricted to trees is NP-complete.

**Proof:** We shall prove that the problem of deciding whether there exist a feasible solution to the WCGP problem is NP-complete. The proof uses a reduction to the KNAPSACK problem ([GJ], problem MP9). Let $\langle U, s, v, B, K \rangle$ 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 $uuU$, $B$ 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.o.g. that
\[ s(u) < B \text{ and } \sum_{u \in U} v(u) > K \text{ for any } u \in U. \]

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
\[ \sum_{u \in V} v(u) < B+1 \text{ and } \sum_{u \in U \setminus V} v(u) > K. \]

Let $V' = V \cup \{r\}$. Then $v(V') < B+1$ and $C = \sum v(u) - \sum_{u \in U \setminus V} v(u) = 1 + \sum u \not \in V v(u) - K = C$. Thus there exist 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 containing $r$, and singleton sets containing vertices of $U$ not in $V'$, where the set $V' \cap U$ 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 are polynomially bounded. Indeed, we show in the next section that the WCCGP problem can be solved in pseudo-polynomial time in either cases. The last 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 3. 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 a related partitioning problem. We shall describe the dynamic programming procedure which is polynomial in $n$, the tree size, and $C_0$. A similar procedure exists which is polynomial in $n$ and $W_0$. Thus, either $C_0$ or $W_0$ are polynomially bounded problem can be solved in polynomial time. Two cases cover most applications.

The algorithm is applying a bottom scanning of the (rooted) tree, calculating each vertex $v$ and integers $k$ and $c$, $0 < k < 0 < c < C_0$, the minimum weight $W_{k,c}(v)$ of top component of capacity $c$ in a legal partition of the subtree rooted at $v$ into $k+1$ components where the top component is the one containing the root $v$. In case no legal partition of the subtree rooted at $v$ satisfy 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 $W_{k,c}(v)$ has already been computed for each $k$ and $c$. Let $B \subseteq \{1,2, \ldots, d\}$ denote the set of indices of the edges $e_i = (v,v_i)$ assigned a cut

Thus, $W_{k,c}(v) = \min(w(v) + \sum W_{k_1,c_1}(v_i), i \in B)$, where the minimum is taken over all subsets $B$ and choices of $k_1$ and $c_1$ such that

1. $\sum c(e_i) + \sum c_i = c$, $i \in B$ and $i \in B$
2. $\sum c(e_i) < C_0$, for $i \in B$

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

$$W_k(r) = \min_{0 \leq c \leq C_0} W_{k,c}(r)$$

and let $k > 0$ be the minimum index such that $W_k(r) \geq W_0$. Then $k+1$ is the minimum number of subtrees in a legal partition of the tree. The partition itself can be reconstructed by the appropriate pointers while computing the weights.

This dynamic programming approach is optimal since it clearly satisfies the principle of optimality (see e.g. [D]). However the complexity of the algorithm may be exponential with respect to $n$, and thus the complexity of the algorithm may 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 $T_v(v)$, $1 \leq i \leq d$, denote the rooted at $v$ containing the children $v_1, \ldots, v_i$ and all their descendants. Denote by $W_{k,c}(v,d)$ the minimum weight of a top component of capacity $c$ in a partition of the subtree $T_v(v)$ into $k$ legal components ($W_{k,c}(v,d) = \infty$ if no partition exists). Note $W_{k,c}(v) = W_{k,c}(v,d)$. We have

1. $W_{k,c}(v,d) = \infty$ if $k < 0$ or $c < 0$, $i = 1,2, \ldots, d$;
2. $W_{0,0}(v) = w(v)$, if $v$ is a leaf;
(iii) $W_{0,0}(v,1) = W_{0,0}(v) + w(v)$;
(iv) $W_{0,0}(v,i) = W_{0,0}(v,i-1) + W_{0,0}(v_1)$, for $i = 2, \ldots, d$;
(v) $W_{k,c}(v,1) = \min \{ W_{k-1,c-c(v,v_1)}(v_1) \}$
then $w(v)$
else $w(v) + W_{k,c}(v_1)$.

For calculating $W_{k,c}(v,1)$, $i=2, \ldots ,d$, we distinguish whether the edge $(v,v_1)$ is assigned a cut or not. In the first case we have

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

where the minimum is taken over all the indices $k_1, c_1$ such that

$0 < k_1 < k$, $0 < c_1 < c - c(v,v_1)$, and $W_{k_1,c_1}(v_1) < W_0$.

In the second case we have

$W''_{k,c}(v,1) = \min \{ W_{k_1,c_1}(v_1) + W_{k-k_1,c-c_1(v_1)} \}$

Finally

$W_{k,c}(v,1) = \min \{ W'_{k,c}(v,1), W''_{k,c}(v,1) \}$.

4. COMPLEXITY ANALYSIS OF THE ALGORITHM

For computing $W_{k,c}(v,1)$ $O(kc)$ operations are required. Let $d(v)$ denote the degree of $v$. Then $O(kc-d(v))$ operations are required to compute $W_{k,c}(v)$, Now $1 < k < n$ and $1 < c < C_0$.

Thus $O(n^2C_0^2-d(v))$ operations are required to compute the weights for the vertex $v$. For the whole tree the complexity is $O(n^2C_0^2 \sum d(v)) = O(n^4C_0^4)$.

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 $O(n^2C_0^2)$ 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 $O(w_0^2C_0^2)$. This algorithm is polynomial only if both $w_0$ and $C_0$ are polynomially bounded, but is 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 $W_{k,c}(v,1)$ simplifies to

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

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

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

Also, part of the computation can be avoided as $W_{k,c}(v) = \infty$ implies $W'_{k,c}(v') = \infty$ for every $k \leq k_1 < k$, $c' < c$, and vertex $v$ on the path from the root to $v$.

The space complexity of the algorithm seems to be $O(n^2C_0d)$, 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 we obtain $O(nC_0 \cdot \text{height}(T))$ space complexity. Note that $W_{k,c}(v,i) = \infty$ whenever $k > |T_i(v)|$. 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 $O(nC_0)$.

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 $1 < i < q$ minimizing $\max w(T_i)$. Let

$W_0 = \min \max w(T_i)$,

where the minimum is over any partition satisfying $c(T_i) < C_0$, $1 < i < q$. 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 $O(n^2C_0^2 \log(w(T)))$, 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.

REFERENCES


[HR] L. HYAFIL and R.L. RIVEST, Graph partitioning and constructing optimal decision trees are polynomial complete


---

Abstr.

The proposed k-d tree method for the linear partitioning of semi-parallel problems provides an efficient and provable solution. It is shown that the algorithm can be implemented in linear time.

Introduction

This paper presents a novel approach to the problem of partitioning semi-parallel problems. The proposed algorithm, based on the k-d tree method, provides an efficient and provable solution. It is shown that the algorithm can be implemented in linear time.

Figure 1

Figure 2

Figure 3