IRIA
labor

Domaine de Voluceau
Rocquencourt
78150 - Le Chesnay
Tél. 954.90.20

laboratoire de recherche
en informatique
et automatique

Graph partitioning and constructing optimal decision trees are polynomial complete problems

L. Hyafil
R.L Rivest

Rapport de Recherche n° 33
octobre 1973

GRAPH PARTITIONING AND CONSTRUCTING OPTIMAL DECISION TREES ARE POLYNOMIAL COMPLETE PROBLEMS

L. Hyafil, R.L. Rivest

Abstract :

The problem of partitioning the nodes of a graph into subsets of bounded size so as to minimize the number of edges connecting nodes from distinct subsets is shown to be polynomial complete. The problem of constructing a decision tree which minimizes the expected number of tests required to identify an object is also shown to be polynomial complete. (Every polynomial complete problem (for example, traveling salesman, graph coloring) can be solved in polynomial time if one of them can.)

Keywords : complexity, polynomial complete, graph partitioning, decision trees.

Nous montrons que le problème de la partition des noeuds d'un graphe en sous ensembles de taille bornée, de façon à minimiser le nombre d'arêtes joignant deux noeuds se trouvant dans des sous ensembles distincts, est polynomial complet. Nous montrons aussi que la construction d'un arbre de décision qui minimise le nombre moyen de recherches nécessaires pour identifier un objet est un problème polynomial complet. (Si un problème polynomial complet peut être résolu en temps polynomial alors tous les problèmes polynomiaux complets (par exemple le problème du voyageur de commerce, ou le problème de la coloration d'un graphe) peuvent être résolus en temps polynomial.)

Mots clés : complexité, polynomial complet, partition d'un graphe, arbre de décision.

In this paper we demonstrate that the problems of graph partitioning and constructing optimal decision trees are polynomial complete. We first present some basic definitions, then the proof for graph partitioning, followed by the proof for decision trees.

Let P (respectively NP) denote the class of functions computable by a deterministic (respectively non-deterministic) Turing machine in a time bounded by a polynomial function of the length of the input. Cook [1] and Karp [5] have shown that P = NP if and only if there exist polynomial-time bounded algorithms for each function in the class PC, the polynomial complete functions, where PC $\subset$ NP. The class PC includes many classic "intractable" problems such as the traveling salesman problem, the knapsack problem, and determining the chromatic number of a graph. The fact that no polynomial-time bounded deterministic algorithm has yet been discovered for any problem in PC leads one to suspect that none exist, that these problems are "inherently" hard, requiring exponential-time algorithms for their solution.

A problem X is polynomial complete if X ϵ NP and there exists a polynomial complete problem Y such that Y is "polynomial reducible" to X (denoted by Y $\propto$ X) (in other words, if Y can be solved in polynomial time by a deterministic Turing machine which has an oracle for solving problem X).

Let G = (V, E) be a graph, where V = $\{v_1, \ldots, v_n\}$ is a finite set of vertices and E is a set of unordered pairs of vertices, the edges of G. A partition $\mathfrak{T}$ of G is a decomposition of the vertices V of G into disjoint nonempty blocks (subsets) $\mathfrak{T} = \{T_1, T_2, \ldots, T_k\}$, so that we have

$$V = \bigcup_{1 \le i \le k} T_i \tag{1}$$

and

$$i \neq j \Rightarrow (T_i \cap T_j) = \emptyset. \tag{2}$$

A partition $\mathfrak{J}$ is said to be r-bounded if no block $T_i \in \mathfrak{J}$ contains more than r vertices. The cost $\mathcal{C}(\mathfrak{J})$ of a partition is defined to be the number of edges joining vertices belonging to distinct blocks. That is,

$$\mathcal{C}(\mathfrak{J}) =_{\text{def}} |\{\{x,y\} \mid (\{x,y\} \in E) \wedge (x \in T_i) \wedge (y \notin T_i)\}| . \quad (3)$$

The graph partitioning problem GP(G,r,w) is to determine whether there exists an r-bounded partition $\mathfrak{J}$ of G such that $\mathcal{C}(\mathfrak{J}) \le w$.

This problem has applications in circuit layout and program segmentation problems. For circuit layouts one wishes to place the n components on circuit boards so as to minimize the number of interboard connections. Similarly one might want to segment a program for placement on the pages of a secondary storage device so as to minimize the number of interpage references.

Simple generalizations of this problem would include weights for each vertex and edge; one then attempts to minimize the weighted sum of the edges joining vertices in distinct blocks of the partition, subject to the constraint that the weighted sum of the nodes in each block may not exceed r. These more general problems are a fortiori also members of PC, as a consequence of the result shown here.

It is interesting to note that when $r = 2$, the graph partitioning problem is just the problem of finding a maximum matching in a graph, and when $r = n - 1$ it is equivalent to finding a minimal cut set of the graph. Both of these problems have polynomial-time bounded algorithms (see [2], [3]).

Theorem GP(G,r,w) $\in$ PC.

Proof : First of all, it is clear that GP $\in$ NP, since a nondeterministic Turing machine can merely guess a partition $\mathfrak{J}$ and then check to see if $\mathcal{C}(\mathfrak{J}) \le w$ and if $\mathfrak{J}$ is r-bounded.

Let S3 be the problem of "satisfiability with exactly three literals per clause". That is we have a set of clauses $D = \{D_1,\ldots,D_r\}$ where each D_i contains exactly 3 literals from the set $L = \{x_1,x_2,\ldots,x_m\} \cup \{\overline{x}_1, \overline{x}_2,\ldots,\overline{x}_m\}$. The set D is satisfiable if there exists a set $L^* \subset L$ of literals such that L^* does not contain any pair of complementary literals and $L^* \cap D_i \neq \emptyset$ for $1 \le i \le r$. Karp [5] has shown that satisfiability with <u>at most</u> three literals per clause to be polynomial complete. As a consequence, $S3 \in PC$ as well since any clause with only one or two literals can be expanded. For example,

$$(a \vee b) \equiv (a \vee b \vee c) \wedge (a \vee b \vee \overline{c}). \tag{4}$$

We will demonstrate that $S3 \propto GP(G,r,w)$. We will first construct the graph G from the set of clauses $D = \{D_1,\ldots,D_r\}$ in two steps. First we consider the graph G' having 3r points which has an r-clique if and only if D is satisfiable (this construction is due to Karp [5]). The graph G' is defined by

$$V' = \{\langle\sigma,i\rangle \mid \sigma \in D_i\}, \text{ and} \tag{5}$$

$$E' = \{\{\langle\sigma,i\rangle,\langle\delta,j\rangle\} \mid i \neq j \text{ and } \sigma \neq \overline{\delta}\}.$$

It is simple to verify that G' has the desired property, since the edges in E' only join vertices corresponding to non-complementary literals in distinct clauses.

We now modify G' slightly to obtain our graph G. To each set of 3 points in G' corresponding to a single clause in D we add a new (r-2)-clique and attach every point of the clique to each of the three points. Thus we have

$$V = V' \cup \{v_{ij} \mid 1 \le i \le r \text{ and } 1 \le j \le (r-2)\}, \text{ and} \tag{6}$$

$$E = E' \cup \{\{v_{ij}, v_{ik}\} \mid 1 \le i \le r,\ 1 \le j \le r-2,\ 1 \le k \le r-2,\ j \neq k\}$$

$$\cup \{\{v_{ij},\langle\sigma,i\rangle\} \mid 1 \le i \le r,\ 1 \le j \le r-2,\ \sigma \in D_i\}.$$

It is easy to see that G contains an r-clique if and only if G' does (that is, if and only if D is satisfiable). We now assert that the problem GP(G,r,$|E|-(r+1)\binom{r}{2}+r$) has a solution if and only if D is satisfiable. Furthermore, the optimal partition will contain the r-clique as a block by itself, if one exists.

Assuming that the r-clique exists, let $\mathcal{J}^*$ be the partition which has the r-clique as one block and r other blocks, each containing one (r-2)-clique and two of the three vertices connected to that clique. (Remember, the r-clique contains exactly one point from each group of three corresponding to a clause). The cost of this partition is just

$$\begin{aligned} \mathcal{C}(\mathcal{J}^*) &= |E| - \binom{r}{2} - r[\binom{r}{2} - 1] \\ &= |E| - (r+1)\binom{r}{2} + r \end{aligned} \tag{7}$$

If no r-clique exists in G, then the average number of edges internal to a block, per node of that block, can be at most $[\binom{r}{2} - 1]/r$, as this is the average number of edges per node in an r-clique lacking one edge. But then we have

$$\begin{aligned} \mathcal{C}(\mathcal{J}) &\geq |E| - (r+1)[\binom{r}{2} - 1] \\ &> \mathcal{C}(\mathcal{J}^*). \end{aligned} \tag{8}$$

Thus the optimal partition contains an r-clique of G as a separate block, if it exists. (Note that any other partition $\mathcal{J}'$ such that $\mathcal{C}(\mathcal{J}') \leq \mathcal{C}(\mathcal{J}^*)$ would also have to contain an r-clique as a block, since the number of internal edges is sufficiently high.)

Thus we have shown that GP $\in$ NP and S3 $\propto$ GP (the construction of G from D takes polynomial time), so that GP is polynomial complete. □

We now turn our attention to the problem of constructing optimal decision trees. Let $X = \{x_1,\ldots,x_n\}$ be a finite set of <u>objects</u> and let $\mathcal{Q} = \{T_1,\ldots,T_t\}$ be a finite set of <u>tests</u>. For each test T_i, $1\leq i\leq t$, and object x_j, $1\leq j\leq n$, we either have $T_i(x_j)$ = true or $T_i(x_j)$ = false. Without fear of confusion, we shall also let T_i denote the set $\{x\in X \mid T_i(x) = \text{true}\}$.

The problem is to construct an identification procedure for the objects in X such that the expected number of tests required to completely identify an element of X is minimal. An identification procedure is essentially a binary decision tree; at the root and all other internal nodes a test is specified, and the terminal nodes specify objects in X. To apply the identification procedure one first applies the test specified at the root to the unknown object; if it is false one takes the left branch, otherwise the right. This procedure is repeated at the root of each successive subtree until one reaches a terminal node which names the unknown object. Let $p(x_i)$ be the length of the path from the root of the tree to the terminal node naming x_i that is, the number of tests required to identify x_i. Then the cost of this tree is merely the external path length, that is, $\sum_{x_i \in X} p(x_i)$.

The decision tree problem $DT(Q,X,w)$ is to determine whether there exists a decision tree with cost less than or equal to w.

This problem has many applications, most of them straightforward identification problems. The problem of compiling decision tables is one such application. As before, this problem can be generalized by the addition of costs for each test $T_i \in Q$ and probabilities for each $x_i \in X$, but a fortiori those are polynomial complete problems as well.

Theorem $DT(Q,X,w) \in PC$.

Proof: Clearly $DT \in NP$, since a non-deterministic Turing machine can guess the decision tree and then see if its weight exceeds w.

To show that DT is complete, we show that $EC3 \propto DT$, where EC3 is the problem of finding an exact cover for a set X, where each of the subsets available for use contains exactly 3 elements. More precisely, we are given a set $X = \{x_1,\ldots,x_n\}$ and a family $Q = \{T_1,\ldots,T_t\}$ of subsets of X, such that $|T_i| = 3$ for $1 \le i \le t$, and we wish to find a sequence of indices $i_1, i_2, \ldots, i_k$ such that

$$\bigcup_{1 \le j \le k} T_{ik} = X, \qquad (9)$$

and

$$(j \neq \ell) \Rightarrow [(T_{jk} \cap T_{i\ell}) = \emptyset]. \qquad (10)$$

The exact cover problem EC (where there is no restriction on the size of each T_i), is known to be in PC (see Karp [5]).

To show that EC3 is complete, we show that 3DM $\propto$ EC3, where 3DM is the problem of finding a "three dimensional matching". That is, we are given a set $U \subseteq B \times B \times B$ and we wish to find a subset W of U such that $|W| = |B|$ and no two elements of W agree in any coordinate. Karp [5] shows that 3DM $\in$ PC. But 3DM becomes EC3 if we let U consist of a set of triples chosen from $B_1 \cup B_2 \cup B_3$ where B_1, B_2, B_3 are distinct sets of size $|B|$ and each element of U contains one element from each B_i. Thus EC3 $\in$ PC.

For a given problem EC3($\mathcal{Q}$,X) we will construct a corresponding problem DT($\mathcal{Q}'$,X',w) such that the optimal solution to DT embodies the solution to EC3 if it exists, thus showing that EC3 $\propto$ DT. We define

$$X' =_{def} X \cup \{a,b,c\}, \qquad (11)$$

where a,b,c are three elements not in X, and

$$\mathcal{Q}' =_{def} \mathcal{Q} \cup \{\{x\} | x \in X'\}. \qquad (12)$$

That is, $\mathcal{Q}'$ contains all the triples in $\mathcal{Q}$ plus all the elements in X' as singleton sets.

We wish to show that an optimal decision tree has the following form :

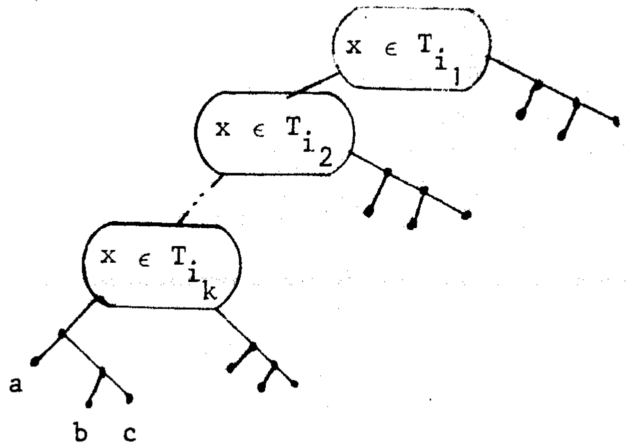

Figure 1

where the sets $T_{i_1}, \ldots, T_{i_k}$ form an exact cover of X by triples from $\mathcal{Q}$, and the singleton sets (tests) are used to distinguish elements within each triple. While this may be intuitively true to the reader, we give a rigorous proof.

Let f(n) denote the minimal cost (path length) of a decision tree on an n-element set X, where all one, two, and three element subsets of X are available as tests. The following table gives the first few values of f(n).

n	1	2	3	4	5	6	7	8
f(n)	0	2	5	8	12	16	20	25

We wish to show that the root of the optimal decision tree always selects a 3-element subset of X as a test, for $n \geq 7$. By definition we have

$$f(n) = \min_{1 \leq i \leq 3} [f(n-i) + f(i) + n], \text{ for } n \geq 4, \tag{13}$$

where i represents the size of the subset chosen as the test at the root. But if $f(m) - f(m-1) > 3$ for $n-3 \leq m \leq n-1$, then $i = 3$ must be chosen to minimize f(n), and $f(n) - f(n-1) = f(n-3) - f(n-4) + 1 > 3$ as well. Thus a 3 element subset must be chosen at the root for all $n \geq 7$.

Since the decision tree of Figure 1 acheives a cost of $f(|X'|)$ exactly, it is optimal. Furthermore, any optimal decision tree must embody the solution to the corresponding EC3 problem, because only in this way can the root of each sufficiently large subtree be a test on a three element subset of X. Thus EC3 $\propto$ DT so that DT $\in$ PC. □

Conclusions

If it is shown that $P \neq NP$, then these problems require exponential-time algorithms for their solution, so that one may have to be content with good heuristic methods, such as those of Kernighan and Lin [4], or Pollack [6].

References

[1] Cook, S. "The complexity of theorem-proving procedures", Third ACM Symposium on Theory of Computing, (1970), 151-158.

[2] Edmonds, J. "Path, Trees, and Flowers", Candian J. Math. 18(1965), 449-467.

[3] Edmonds, J. and R. Karp "Theoretical Improvements in the Algorithmic Efficiency of Network Problems", JACM 19 (1972), 248-264.

[4] Kernighan, B.W. and S. Lin "The partitioning of graphs", Eng. Cybern. 7(1969), 76-82.

[5] Karp, Richard M. "Reducibility among combinatorial problems", IBM Symposium on computational complexity (1973), 85-103.

[6] Pollack, S.L. "Conversion of limited-entry decision tables to computer programs," CACM 8(1965), 667-672.