Reprinted from INFORMATION AND CONTROL
All Rights Reserved by Academic Press, New York and London

Vol. 53, No. 3, June 1982
Printed in Belgium

On the Complexity of Designing Distributed Protocols

CHRISTOS H. PAPADIMITRIOU[1] AND JOHN TSITSIKLIS

Department of EECS, M.I.T., Cambridge, Massachusetts 02139

The complexity of two problems of distributed computation and decision-making is studied. It is shown that deciding whether two distant agents can arrive at compatible decisions without any communication can be done in polynomial time if there are two possible decisions for each agent, but is *NP*-complete if one agent has three or more alternatives. It is also shown that minimizing the amount of communication necessary for the distributed computation of a function, when two distant computers receive each a part of the input, is *NP*-complete. This proves a conjecture due to A. Yao.

1. INTRODUCTION

Recently there has been some interest in modeling problems of distributed computation and distributed decision-making as computational problems, and in studying their complexity (Ladner, 1979; Kanellakis, 1980; Kanellakis and Papadimitriou, 1981; Garey *et al.*, 1982; Yao, 1979). The goal of this line of research is to capture the apparent intricacy inherent in distributed computation in terms of the *computational complexity* of the corresponding problem. This paper is a contribution to this research.

Suppose that two distant agents I and II must arrive at a decision based on local observations, and perhaps on some information communicated between them. Agent I's observation, say y_I, comes from a finite set Y_I of possible observations. Similarly $y_{II} \in Y_{II}$. I's decision is an element u_I of the finite set U_I of possible decisions, and likewise for II. Not all combinations of decisions are acceptable, however; the overall decision (u_I, u_{II}) must belong to a particular subset $S(y_I, y_{II})$ of $U_I \times U_{II}$. The question is, given the function S, can we design a protocol whereby the two agents arrive each at their own decision, based only on the respective local information (i.e., without *any* communication), so that the specifications in S are always observed? We call this the *distributed satisficing problem*, after the term introduced by Simon (1969). Notice that this problem can be considered as a simple (finite) special case of the classical *team decision making* problem (Marschak and Radner, 1972; Tsitsiklis, 1983).

[1] Current address: Departments of Operations Research and Computer Science, Stanford University.

0019-9958/82 $2.00

Copyright © 1982 by Academic Press, Inc.
All rights of reproduction in any form reserved.

We show that the complexity of the distributed satisficing problem depends explicitly on the cardinalities of the decision sets U_I and U_{II}. If either set is empty or a singleton, then of course the problem is trivial. If both sets are of cardinality two, then we show that the problem can be solved in polynomial time (Theorem 2). In all other cases, distributed satisficing is *NP*-complete (Theorem 1). For definitions related to *NP*-completeness, the reader is referred to Garey and Johnson (1979), Papadimitriou and Steiglitz (1982).

Suppose that the cardinalities of both U_I and U_{II} are two, say $U_I = U_{II} = \{0, 1\}$, and that communication is indeed necessary; by the discussion above, this is easy to check. How can we minimize the amount of communication that is required in order for the two agents to arrive at satisficing decisions? An important special case of this problem is that of *distributed function evaluation*, which has been recently studied in the computer science literature, in connection with both distributed computation and VLSI (Abelson, 1978; Yao, 1979; Papadimitriou and Sipser, 1982; Lipton and Sedgewick, 1981). Suppose that we wish to compute a function $f: Y_I \times Y_{II} \to \{0, 1\}$, when part of the input (namely, y_I) is available to computer I, and the rest (y_{II}) is available to computer II. It is easy to see that this is a special case of the distributed satisficing problem with $U_I = U_{II} = \{0, 1\}$ and $S(y_I, y_{II}) = \{(0, 0)\}$ or $\{(1, 1)\}$ for all y_I and y_{II}. Obviously, most interesting functions f cannot be computed without any communication, and there has been some interesting recent work concerning lower bounds on the amount of communication (number of bits exchanged) that is required for specific functions f (Abelson, 1978; Yao, 1979; Lipton and Sedgewick, 1981; Papadimitriou and Sipser, 1982). In fact, Yao asked in 1979 whether the problem of minimizing the amount of communication necessary for the distributed computation of a given function f is an *NP*-complete problem. We show that it is (Theorem 3).

2. Distributed Satisficing

The distributed satisficing problem can be formalized as follows:

Distributed Satisficing

Given finite sets Y_I, U_I, Y_{II}, U_{II}, and a function $S: Y_I \times Y_{II} \to 2^{U_I \times U_{II}}$, are there functions $\gamma_I: Y_I \to U_I$ and $\gamma_{II}: Y_{II} \to U_{II}$ such that for all $y_I \in Y_I$, $y_{II} \in Y_{II}$ $(\gamma_I(y_I), \gamma_{II}(y_{II})) \in S(y_I, y_{II})$?

In order to study the complexity of the distributed satisficing problem, we shall first point out the close connection between this problem and a family of restricted versions of the *satisfiability* problem for propositional calculus,

which we call k-RSAT ($k > 1$ is an integer). A formula of k-RSAT has the following set of variables, for some $m, n > 0$

$$\{x_1, x_2, \ldots, x_n\} \cup \{y_{ij} : i = 1, \ldots, m; j = 1, \ldots, k\}.$$

The clauses are the following:

(a) One clause for each i between 1 and m, stating that *exactly* one of the variables $y_{i1}, \ldots, y_{ik}$ is true, and

(b) An arbitrary number of clauses of the form $(\neg y_{ij} \vee x_q)$ or $(\neg y_{ij} \vee \neg x_q)$.

LEMMA 1. *k-RSAT is equivalent to distributed satisficing with $U_{\mathrm{I}} = \{0, 1\}$ and $U_{\mathrm{II}} = \{1, 2, \ldots, k\}$.*

Proof. Think of x_i as stating that, if agent I observes the ith element of Y_{I}, then she decides 1 (thus $\neg x_i$ states that she decides 0); and think of y_{ij} as stating that, if agent II observes the ith element of Y_{II}, then she decides j. By using the clauses in (b), we can express the combinations of decisions that are incompatible according to S. ∎

Our first two results follow now from Lemma 1:

THEOREM 1. *3-RSAT is NP-complete. Consequently, the distributed satisficing problem with decision sets of cardinality greater than or equal to two and three, respectively, is NP-complete.*

Proof. We shall reduce to 3-RSAT the *NP*-complete problem of satisfiability of propositional formulae with three literals per clause (3SAT, Garey and Johnson, 1979). Given such a propositional formula, we shall construct an equivalent 3-RSAT formula F. For each variable of the original formula we have in F a new x-variable. For each pair of variables a and b of the original formula, we add to F two *triples of* y-variables y_{abj} and y'_{abj}, $j = 1, 2, 3$, and the corresponding "exactly one is true" clauses. Also, we add to F the x-variable z_{ab}, and the following 10 clauses:

$$(\neg y_{ab1} \vee a), \quad (\neg y_{ab1} \vee b), \quad (\neg y_{ab2} \vee a), \quad (\neg y_{ab2} \vee \neg b),$$
$$(\neg y'_{ab1} \vee \neg a), \quad (\neg y'_{ab1} \vee b), \quad (\neg y'_{ab2} \vee \neg a), \quad (\neg y'_{ab2} \vee \neg b),$$
$$(\neg y_{ab3} \vee z_{ab}), \quad (\neg y'_{ab3} \vee \neg z_{ab}).$$

It is left as an exercise to the reader to verify that these ten clauses force the variables $y_{ab1}, y_{ab2}, y'_{ab1}, y'_{ab2}$ to always take the same values as the *expressions* $(a \wedge b)$, $(a \wedge \neg b)$, $(\neg a \wedge b)$, $(\neg a \wedge \neg b)$, respectively. Using this observation, we can rewrite any three-literal clause of our original formula as a two-literal clause of F. ∎

THEOREM 2. 2-RSAT *is solvable in time* $O(n)$. *Consequently, the distributed satisficing problem with decision sets both of cardinality two is linear-time solvable.*

Proof. Simply notice that 2-RSAT can be reduced to the linear-time solvable special case of satisfiability, in which the clauses are restricted to have only two literals (2SAT, Cook, 1971; Garey and Johnson, 1979). To do this, we replace the only clauses that do not conform to the format of 2SAT, namely, the "exactly one true" ones, as follows: "exactly one of y_{i1}, y_{i2} is true" becomes $(y_{i1} \vee y_{i2}) \wedge (\neg y_{i1} \vee \neg y_{i2})$. ∎

3. DISTRIBUTED EVALUATION OF BOOLEAN FUNCTIONS

If I and II wish to cooperate in order to compute in a distributed fashion a function $f: Y_{\mathrm{I}} \times Y_{\mathrm{II}} \to \{0, 1\}$, they must design a protocol for information transfer. How can we model mathematically such protocols, as well as the amount of information transfer that they require? Think of the function f as a table with $|Y_{\mathrm{I}}|$ rows, $|Y_{\mathrm{II}}|$ columns, and $0-1$ entries. Let $B \geqslant 0$. We say that *f can be computed with B bits of communication* if either the table is all zeroes or all ones (in which case no communication is required), or the table can be partitioned horizontally or vertically into subtables, by splitting the set of rows or the set of columns of f, both of which can be computed with $B-1$ bits of communication. This recursive definition also suggests a protocol for achieving this computation. If f can be computed with B bits, and the appropriate partition is a horizontal (resp. vertical) one, then I (resp. II) sends a bit signalling which of the two sets of the partition the current row (resp. column) happens to belong to; after this, the two go on recursively to compute the appropriate restriction of f in one less bit of communication, until a trivial (i.e., constant) function has resulted. This definition of communication protocol is the most natural and liberal one used in the literature, and it corresponds to the *prefix-freeness* property of messages insisted upon in Papadimitriou and Sipser (1982) (i.e., no message of one computer to the other can be a prefix of another message, and thus the two computers know when a message ends). For a discussion of the desirability of this property see Papadimitriou and Sipser (1982). Yao in fact used a slightly different definition, in which the two agents must alternate sending one-bit messages (Yao, 1979). *NP*-completeness can be similarly proved in Yao's model as well.

The problem we are interested in is

Distributed Function Evaluation

Given a function $f: Y_{\mathrm{I}} \times Y_{\mathrm{II}} \to \{0, 1\}$ and an integer B, is there a protocol

for computing this function, which uses a total number of bits less than or equal to B?

The following theorem proves a conjecture due to Yao (1979).

THEOREM 3. *Distributed function evaluation is NP-complete.*

In order to prove this result, we shall first need a lemma concerning the following problem:

Exact Cover

Given a family $F = \{S_1,..., S_m\}$ of subsets of $U = \{u_1,..., u_n\}$, is there a subset C of F containing mutually disjoint sets, whose union is U?

LEMMA 2. EXACT COVER *is NP-complete even if the following conditions are true*:

(a) *All sets in F have cardinality one or three.*

(b) *m is a power of* 2, *and any exact cover must contain exactly half of the sets in F.*

(c) *U can be divided into three subsets V, W, and Y such that*:

(c1) *Each element of V is contained in exactly two sets, both of cardinality three. These sets have two elements from V and one from W.*

(c2) *All singletons in F are subsets of W.*

(c3) *All other sets in F (besides those in* (c1) *and* (c2)) *consist of two elements in Y and one in W.*

(c4) $|V| + |W| = m$.

Proof. The construction is a variation of the reduction given in Garey and Johnson (1979) from 3SAT to THREE-DIMENSIONAL MATCHING. We start from the version of *one-in-three satisfiability*, in which we are given m disjunctive clauses with three literals each from the variables $x_1,..., x_n$, and we are asked whether there is a truth assignment which satisfies *exactly* one literal in each clause. This problem is *NP*-complete (Schaefer, 1978; Garey and Johnson, 1979). In fact, we can assume that no literal appears more than twice in the formula.

Given such an instance, we shall construct an instance $\{S_1,..., S_m\}$ of EXACT COVER. For each variable x we have four elements $v_1(x),..., v_4(x)$, and four more $w_1(x)$, $w_2(x)$, $w_1(\neg x)$, $w_2(\neg x)$, and the sets $S_1(x) = \{v_1(x), v_2(x), w_1(x)\}$, $S_1(\neg x) = \{v_2(x), v_3(x), w_1(\neg x)\}$, $S_2(x) = \{v_3(x), v_4(x), w_2(x)\}$, $S_2(\neg x) = \{v_4(x), v_1(x), w_2(\neg x)\}$. These are the only sets which involve the $v_i(x)$ nodes, and so any exact cover will have to include either both $S_1(x)$ and $S_2(x)$, or both $S_1(\neg x)$ and $S_2(\neg x)$ for each variable x (and thus it will

imply a truth assignment for the variables). Now we turn to the clauses. For each clause C_j we have two elements $y_1(C_j)$ and $y_2(C_j)$, and the sets $\{y_1(C_j), y_2(C_j), w_i(\lambda)\}$, whenever the ith occurrence of the literal λ is in the clause C_j. If $w_2(\lambda)$ does not occur in any such set (because λ occurs only once in the formula), then we add the singleton $\{w_2(\lambda)\}$. This completes the construction.

We argue that the resulting instance of EXACT COVER has a solution iff the given formula has a satisfying truth assignment. If there is a solution, then it must define a truth assignment by picking both sets corresponding to either x or $\neg x$ for each variable. Among the remaining $w_i(\lambda)$ elements, those which correspond to "unused" occurrences of literals must be picked up by the singletons, whereas the remaining ones are picked up by the clauses in which they occur. It follows that the literals left out by the $S_j(\lambda)$ sets define a truth assignment which satisfies exactly one literal in each clause. Thus, if this instance of EXACT COVER has a solution, then there is a satisfying truth assignment for the given formula which satisfies one literal in each clause, as required. The converse follows easily.

It remains to check that the conditions of the lemma are all satisfied. Condition (a) is already satisfied. For (c), just take V, W, and Y to be the sets of the v, w, and y-elements in our construction. (c1)–(c3) are easily checked, and (c4) follows by arithmetic: If there are n variables, then $|V| = |W| = m/2 = 4n$. The second part of (b) now follows immediately (an exact cover has always $(|W| + |V|)/2$ sets), and the first part can be guaranteed by a variety of padding arguments. ∎

Proof of Theorem 3. We shall reduce to this problem the EXACT COVER problem. Given an instance of EXACT COVER as described in the lemma, we construct an instance of DISTRIBUTED FUNCTION EVALUATION, as follows: Y_{I} is the set $F \cup \{U\}$, and Y_{II} is the set U. The function f is defined as follows: $f(y_{\text{I}}, y_{\text{II}})$ is 1 if $y_{\text{II}} \in y_{\text{I}}$. Otherwise, $f(y_{\text{I}}, y_{\text{II}}) = 0$. The bound on the number of bits that can be exchanged is $B = \log_2 m + 1$.

We claim that f can be computed within this bound of communication iff the given instance of EXACT COVER has a solution. Suppose that there is a protocol for computing f, which involves at most B bits of communication. Thus, the protocol succeeds in finally dividing the table of f into $2^B = 2m$ disjoint smaller tables that are either all zeroes or all ones (we call these tables the *boxes*). Let us consider f with its rows restricted to the sets referred to in (c3) and (c2) of the lemma; it is easy to see that the corresponding table is an identity matrix (the columns are in the set W) followed by several other columns. Each one in the identity matrix must be by itself in a separate box. Furthermore, since both dimensions of the box are 1, for each such box we can find a box with zeroes from the same row or the same column. It follows then that the optimum way to partition an identity matrix into boxes

takes twice as many boxes as rows, and thus $2 \cdot |W|$ distinct boxes are required for this part. In fact, because of the presence of the other columns, this bound can be achieved only if the boxes are partitions of the individual rows into their parts that contain zeros and ones. Also, if we look at the remaining $|V|$ rows of f, except for the U row, we notice that we have a block diagonal matrix with blocks consisting of the adjacency matrices of cycles; this is, followed by other columns, such that each row contains three ones. It can be argued similarly that $2 \cdot |V|$ boxes are required for this part of f. In fact this can be achieved simultaneously with the previous bound only if all these boxes are the rows of the table, each split into its zero and one parts. Adding up, we conclude that $2 \cdot (|V| + |W|) = 2^B$ boxes are absolutely necessary for computing f, and in fact that this bound is achievable only if these boxes are simply partitions of the rows into their zero and one parts.

Let us now consider the U row. How can its ones be covered exactly (i.e., without overlaps) together with some of the already considered boxes which contain ones (so that 2^B boxes are also sufficient)? It is not hard to see that the only way is to choose an exact cover among the rows, and merge the corresponding boxes of ones with the corresponding ones of the U row. Hence, F must have an exact cover.

Conversely, if F has an exact cover, then we can compute f with B bits as follows: First I sends a bit to II, telling her whether the row which I sees is a set *not* belonging to the cover. If this is the case, I uses all but one of the remaining bits to completely describe the row (there are $m/2 = 2^{B-2}$ such rows), and then II tells I with the last bit whether the column II sees has a one in the describe row or not. Otherwise, if the row is in the cover, then II uses all but the last bit to tell I to which of the sets in the exact cover the element corresponding to II's column belongs. Then I tells II whether the row she sees is the U row or the set described by II, in which cases the answer is one. In all other cases, the answer is 0. ∎

ACKNOWLEDGMENTS

This research was supported in part by the Office of Naval Research under Grant ONR/N00014–77–C–0532(NR041–519), and by a grant from the National Science Foundation.

REFERENCES

ABELSON, H. (1978), Lower bounds on information transfer in distributed computations. *in* "Proceedings, 19th FOCS Symposium."

COOK S. A. (1971) The complexity of theorem-proving procedures, *in* "Proceedings, 3rd STOC," pp. 151–158.

GAREY, M. R. AND JOHNSON, D. S. (1979), "Computers and Intractability: A Guide to the Theory of *NP*-completeness," Freeman, San Francisco.

GAREY, M. R., JOHNSON, D.S., AND WITSENHAUSEN, H. S. (1982) The complexity of the generalized Lloyd–Max problem, *IEEE Trans. Inform. Theory* IT-28 2, 255–256.

KANELLAKIS, P. (1980), "The Complexity of Distributed Concurrency Control." Ph. D. Dissertation, Lab. for Computer Science, M.I.T., Cambridge, Mass.

KANELLAKIS, P., AND PAPADIMITRIOU, C. H. (1981), The complexity of distributed concurrency control, *in* "Proceedings, 22nd FOCS Symposium," pp. 185–197.

LADNER, R. E., (1979), The complexity of problems in systems of communicating sequential processes, in "Proceedings, 11th STOC," pp. 214–223.

LIPTON R. J. AND SEDGEWICK, B. (1981), Lower bounds for VLSI, *in* "Proceedings, 13th STOC," pp. 300–307.

MARSCHAK, J., AND RADNER, R. (1972), "The Economic Theory of Teams," Yale Univ. Press, New Haven, Conn.

PAPADIMITRIOU, C. H., AND STEIGLITZ, K. (1982), "Combinatorial Optimization: Algorithms and Complexity," Prentice–Hall, Englewood Cliffs, N. J.

PAPADIMITRIOU, C. H. AND SIPSER, M. (1982), Communication complexity, *in* "Proceedings, 14th STOC," pp. 196–200.

SCHAEFER, T. J. (1978), The complexity of satisfiability problems, *in* Proceedings 10th STOC," pp. 216–226.

SIMON, H. (1969), "The Sciences of the Artificial," M.I.T. Press, Cambridge, Mass.

TSITSIKLIS, J. (1983), "Problems in Distributed Decision-Making," Ph. D. Dissertation, M.I.T., Cambridge, Mass.

YAO, A.C.-C. (1979), Some complexity questions related to distributive computing, *in* "Proceedings, 11th STOC," pp. 209–213.

Printed by the St. Catherine Press Ltd., Tempelhof 41, Bruges, Belgium