On the Complexity of Decentralized Decision Making and Detection Problems

JOHN N. TSITSIKLIS, MEMBER, IEEE, AND MICHAEL ATHANS, FELLOW, IEEE

Abstract—We study the computational complexity of the discrete versions of some simple but basic decentralized decision problems. These problems are variations of the classical "team decision problem" and include the problem of decentralized detection whereby a central processor is to select one of two hypotheses, based on 1-bit messages from two noncommunicating sensors. Our results point to the inherent difficulty of decentralized decision making and suggest that optimality may be an elusive goal.

I. INTRODUCTION

The field of decentralized (distributed) decision making has been an active area of research for more than two decades [4], [11], [12]. In the meantime, it has been realized that decentralized decision problems are qualitatively different from the corresponding decision problems with centralized information. The classical "counterexample" of Witsenhausen [5], [17] in decentralized stochastic control best illustrates this point. It is safe to conjecture that the prohibitive factor in decentralized problems is not so much the inadequacy of the mathematical tools presently been used, but rather the inherent complexity (in the broad sense of the term) of the problems that have usually been formulated. However, there are very few [3], [18] precise mathematical results on the nature of this ever-present complexity. The present paper, which follows the line of research of [10], should be viewed as a contribution in this direction. We focus on finite versions of some simple but fairly typical decentralized decision-making problems and characterize their complexity by using the tools of the theory of computational complexity [2], [9]. Keeping with the tradition of this theory, we consider "easy" those problems that can be solved by a polynomial algorithm, whereas we consider NP-complete (or worse) problems to be "hard." In our opinion, such an approach is: 1) more satisfying intellectually, and 2) given the present state of the theory of decentralized decision making, it will allow us to systematically identify hard problems and redirect research efforts to heuristic and approximate algorithms or possibly easier special cases.

Overview

The main issue of interest in decentralized systems may be loosely phrased as "who should communicate to whom, what, when, etc." From a purely logical point of view, however, there is a question which precedes the above: "Are there any communications necessary?" Section II addresses the difficulty of the problem of deciding whether any communications are necessary for a given decentralized system. We use a formulation of this problem introduced in [10]. We impose some additional structure on this problem, and we are able to determine the boundary between easy and hard cases. In Section III, we formulate the discrete version of the decentralized detection problem and prove that it is, in general, a hard one. In Section IV, we present and discuss a few problems related to the problem of Section II, including the team decision problem. Section V contains our conclusions. All proofs may be found in the Appendix.

II. A PROBLEM OF SILENT COORDINATION

Let \{1, \ldots, M\} be a set of processors (decision makers). Each processor \(i\) obtains an observation \(y_i \in Y_i\), where \(Y_i\) is a finite set. Then processor \(i\) chooses a decision \(u_i\) belonging to a finite set \(U_i\) of possible decisions according to some function \(\gamma_i(y_i)\), where \(\gamma_i\) is some function from \(Y_i\) into \(U_i\). The \(M\) tuple \((y_1, \ldots, y_M)\) is the total information available and may be viewed as the "state of the environment." For each state of the environment, we assume that only certain \(M\) tuples \((u_1, \ldots, u_M)\) of decisions accomplish a given, externally specified goal. More precisely, for each \((y_1, \ldots, y_M)\), we are given a set \(S(y_1, \ldots, y_M) \subset U_1 \times \cdots \times U_M\) of satisfying (using the term introduced by Simon) decisions. So \(S\) may be viewed as a function from \(Y_1 \times \cdots \times Y_M\) into \(2^{U_1 \times \cdots \times U_M}\). We then ask whether there exist decision rules \(\gamma_i: Y_i \to U_i\) (involving no communications) which always lead to satisfying decisions. This problem was first introduced (for the case \(M = 2\)) in [10] and may be formalized as follows.

**Problem DS:** Given finite sets \(Y_1, \ldots, Y_M, U_1, \ldots, U_M\) and a function \(S: Y_1 \times \cdots \times Y_M \to 2^{U_1 \times \cdots \times U_M}\), are there functions \(\gamma_i: Y_i \to U_i, i = 1, \ldots, M\) such that

\[
\forall (y_1, \ldots, y_M) \in S(y_1, \ldots, y_M),
\gamma_1(y_1), \ldots, \gamma_M(y_M) \in S(y_1, \ldots, y_M).
\]

(2.1)

We are assuming above that the function \(S\) is easily computable; for example, it may be given in the form of a table. Also note that the centralized counterpart of DS would be to allow the decision \(u_i\) of each processor to depend on the entire set of observations, in which case the problem would become trivial. Since DS has a trivial centralized counterpart, any difficulty inherent in DS is only caused by the fact that information is decentralized. In this sense, DS captures the essence of coordinated decentralized decision making (silent coordination).

The following were shown in [10].

1) The problem DS with two processors \((M = 2)\) and restricted to instances for which the cardinality of the decision sets is \(2(|U_i| = 2, i = 1, 2)\) may be solved in polynomial time.

2) The problem DS with two processors \((M = 2)\) is NP-complete even if we restrict to instances for which \(|U_i| = 2, i\).

An extension of the above results is the following.

**Proposition 2.1:** The problem DS with three or more processors \((M \geq 3)\) is NP-complete even if we restrict to instances for which \(|U_i| = 2, i\).

\footnote{For any finite set \(A\), we let \(|A|\) denote its cardinality.}
We may therefore conclude that the problem DS is, in general, a hard combinatorial problem, except for the special case in which there are only two processors and each one has to make a binary decision. It should be noted that the difficulty is not caused by an attempt to optimize with respect to a cost function because no cost function is introduced.

We now turn to special cases of DS by introducing some more structure (reflecting the nature of typical practical problems) with the aim of determining the dividing line between easy and hard special cases. Moreover, we restrict to the case of only two processors \((M = 2)\). (Certainly, problems with \(M > 2\) cannot be easier.)

In the formulation of DS that we introduced earlier, all pairs \((y_1, y_2) \in Y_1 \times Y_2\) are likely to occur. So the information of different processors is completely unrelated; their coupling is caused only by the structure of the satisfying sets \(S(y_1, y_2)\). In most practical situations, however, information is not completely unstructured: when processor 1 observes \(y_1\), he is often able to make certain inferences about the value of the observation \(y_2\) of the other processor and exclude certain values. We now formalize these ideas.

**Definition:** An information structure \(I\) is a subset of \(Y_1 \times Y_2\). We say that an information structure \(I\) has degree \((D_1, D_2)\), \((D_1, D_2, 1)\), and \((D_1, D_2, 2)\) are positive integers if

1. for each \(y_1 \in Y_1\), there exist at most \(D_1\) distinct elements of \(Y_2\) such that \((y_1, y_2) \in I\)
2. for each \(y_2 \in Y_2\), there exist at most \(D_2\) distinct elements of \(Y_1\) such that \((y_1, y_2) \in I\)
3. \((D_1, D_2)\) are the smallest integers satisfying 1) and 2).

We now interpret this definition. The information structure \(I\) is the set of pairs \((y_1, y_2)\) of observations that may occur together. If \(I\) has degree \((D_1, D_2)\), processor 1 may use his observation \(y_1\) to decide which elements of \(Y_2\) may have been observed by processor 2. In particular, he may exclude all elements except for (at most) \(D_1\) of them. The situation faced by processor 2 is symmetrical.

If \(D_1 = 1\) and processor 1 observes \(y_1\), there is only one possible value for \(y_2\). So processor 1 knows the observation of processor 2. (The converse is true when \(D_2 = 1\).) We could call this a nested information structure because the information of one processor contains the information of the other. When \(D_1 = D_2 = 1\), each processor knows the observation of the other; so their information is essentially shared.

By restricting our attention only to pairs \((y_1, y_2) \in I\), we obtain the following version of DS.

**Problem DS:** Given a finite set \(Y_1, Y_2, U_1, U_2, I \subset Y_1 \times Y_2\) and a function \(S: I \rightarrow 2^{U_1 \times U_2}\), are there functions \(\gamma_i: Y_i \rightarrow U_i, i = 1, 2\) such that

\[
(\gamma_1(y_1), \gamma_2(y_2)) \in S(y_1, y_2), \quad \forall (y_1, y_2) \in I.
\] (2.2)

Note that any instance of DS is equivalent to an instance of DS in which

\[
S(y_1, y_2) = U_1 \times U_2, \quad \forall (y_1, y_2) \in I.
\]

That is, no compatibility restrictions are placed on the decisions of the two processors for those \((y_1, y_2)\) that cannot occur. We now proceed to the main result of this section.

**Proposition 2.2:**

1. The problem DS restricted to instances satisfying any of the following:
   a) one or more of \(|U_1|, |U_2|, D_1, D_2\) is equal to 1
   b) \(|U_1| = |U_2|\)
   c) \(D_1 = D_2 = 2\)
   d) \(D_1 = |U_2|, D_2 = |U_1| = 2\) may be solved in polynomial time.

2. The problem DS is NP-complete even if we restrict to instances for which \(|U_1| = D_3 = 3, |U_2| = D_2 = 2\). Note that the above result draws a precise boundary between polynomial and NP-complete special cases in terms of \(D_1, |U_i|, i = 1, 2\). We have seen that, in general, DS is NP-complete even if \(D_1, D_2\) are fixed. There is, however, a special case of DS, with \(D_1, D_2\) constant, for which an efficient algorithm of the dynamic programming type is possible.

**Proposition 2.3:** Let \(Y_1 = Y_2 = \{1, 2, \ldots, n\}\). Let \(D\) be a positive integer constant. Consider those instances of DS for which \((i, j) \in I\) implies either \(|i - j| \leq D\) or \(|i - j| \geq n - D\). Then DS may be solved in time which is polynomial in \(n\).

A condition of the type \(|i - j| \leq D\) or \(|i - j| \geq n - D\) for all \((i, j) \in I\) is fairly natural in certain applications. For example, suppose that the observations \(y_1\) and \(y_2\) are noisy measurements of an unknown variable \(x(y_1, y_2)\) where the noises \(w_i\) are bounded: \(|w_i| < D/2\). Similarly, \(|i - j| \leq D\) or \(|i - j| \geq n - D\), \((i, j) \in I\) arises if the observations \(y_1, y_2\) are noisy measurements of an unknown quantized angle: \(y_1 = \theta + w_i\) (mod \(2\pi\)) where the noises \(w_i\) are again bounded by \(D/2\).

### III. Decentralized Detection

A basic problem in decentralized signal processing, which has attracted much attention recently [1], [6], [7], [13]–[15] is the decentralized detection (hypothesis testing) problem. In this section, we consider the discrete version of this problem.

Two processors (sensors) \(S_1\) and \(S_2\) receive measurements \(y_i \in Y_i, i = 1, 2\) where \(Y_i\) are finite sets (Fig. 1). There are two hypotheses \(H_0\) and \(H_1\) on the state of the environment with prior probabilities \(p_0\) and \(p_1\), respectively. For each hypothesis \(H_i\), we are also given the joint probability distribution \(P(y_1, y_2|H_i)\) of the observations, conditioned on the event that \(H_i\) is true. Upon receipt of \(y_i\), processor \(S_i\) evaluates a binary message \(u_i \in \{0, 1\}\) according to a rule \(u_i = \gamma_i(y_i)\) where \(\gamma_i: Y_i \rightarrow \{0, 1\}\). Then \(u_1\) and \(u_2\) are transmitted to a central processor (fusion center) which evaluates \(u_0 = u_1u_2\) and declares hypothesis \(H_0\) to be true if \(u_0 = 0, H_1\) if \(u_0 = 1\). (So we essentially have a voting scheme.) The problem is to select the functions \(\gamma_1, \gamma_2\) so as to minimize the probability of accepting the wrong hypothesis. (More general performance criteria may be also considered.)

Most available results assume that

\[
P(y_1, y_2|H_i) = p(y_1|H_i)p(y_2|H_i), \quad i = 1, 2\]

which states that the observations of the two processors are independent when conditioned on either hypothesis. In particular, it has been shown [15] that if (3.1) holds, then the optimal decision rules \(\gamma_i\) are given in terms of thresholds for the likelihood ratio \(P(y_i|H_0)/P(y_i|H_1)\). The optimal thresholds for the two sensors are coupled through a system of equations which gives necessary conditions of optimality. (These equations are just the person-by-person optimality conditions.) Few analytical results are available when the conditional independence assumption is removed [7]. The purpose of this section is precisely to explain this status of affairs.

Suppose that (3.1) holds, and let \(N_i\) denote the cardinality of \(Y_i\), Given the results of [15], there are only \(N_i + 1\) decision rules \(\gamma_i\) which are candidates for being optimal. We may evaluate the cost associated to each pair of candidate decision rules and select a pair with least cost. This corresponds to a polynomial algorithm, and shows that under condition (3.1), decentralized detection is an easy problem. Without the conditional independence assumption (3.1), however, there is no guarantee that optimal decision rules can be defined in terms of thresholds for the likelihood ratio. Accordingly, a solution by exhaustive enumeration could require

---

2 Such an assumption is reasonable in problems of detection of a known signal in independent noise, but is typically violated in problems of detection of an unknown signal.
the examination of as many as \(2^N\) pairs of decision rules. One might expect that a substantially faster (i.e., polynomial) algorithm is possible. However, the main result of this section (Proposition 3.1 below) states that decentralized detection is NP-complete even if we restrict to instances for which perfect detection (zero probability of error) is possible for the corresponding centralized detection problem.

We now present formally a suitable version of the problem.

### Decentralized Detection (DD):
We are given finite sets \(Y_1, Y_2\); a rational number \(K\); a rational probability mass function \(p: Y_1 \times Y_2 \rightarrow Q\); and a partition \(\{A_0, A_1\}\) of \(Y_1 \times Y_2\). Do there exist decision rules \(y_i: Y_i \rightarrow \{0, 1\}, i = 1, 2\) such that \(J(y_1, y_2) = K\)? Here

\[
J(y_1, y_2) = \sum_{(y_1, y_2) \in A_0} p(y_1, y_2)\gamma_1(y_1)\gamma_2(y_2) + \sum_{(y_1, y_2) \in A_1} p(y_1, y_2)[1 - \gamma_1(y_1)\gamma_2(y_2)].
\] (3.2)

**Remarks:**

1) In the above definition of DD, think of \(H_i\) as being the hypothesis that \(y_i, y_2 \in A_i\). Then it is easy to see that \(J(y_1, y_2)\) corresponds to the probability of error associated to the decision rules \(y_1, y_2\). Note that if a single processor knew both \(y_1\) and \(y_2\) (centralized information), he could make the correct decision with certainty. Consequently, the above-defined problem corresponds to the special case of decentralized detection for which perfect centralization is possible.

2) If we let \(K = 0\), then DD is a special case of problem DS with \(|U_1| = |U_2| = 2\), and is therefore polynomially solvable.

**Proposition 3.1:** DD is NP-complete.

It should be pointed out that Proposition 3.1 remains valid if the problem is modified so the fusion center uses some other rule for combining the messages it receives (e.g., \(u_0 = u_t(1 - u_t)\)) or if the combining rule is unconstrained and the fusion center is supposed to find and use an optimal combining rule.

Let us now interpret Proposition 3.1. Although it is, in a sense, a negative result, it can be useful in suggesting meaningful directions for future research: instead of looking for efficient exact algorithms, the focus should be on approximate ones. (In fact, it is an interesting research question whether polynomial approximate algorithms for DD exist.) Proposition 3.1 also shows that any necessary conditions to be developed for problem DD will be deficient in one of two respects:

1) either there will be a very large number of decision rules satisfying these conditions or
2) it will be hard to find decision rules satisfying these conditions.

Another consequence of Proposition 3.1 is that optimal decision rules are not given, in general, in terms of thresholds for the likelihood ratio because in that case, an efficient algorithm could be obtained. Of course, this fact can be also verified directly by constructing appropriate examples. When the condition (3.1) holds and decision rules are given in terms of thresholds, the decision rule of a processor can be viewed as a tentative local decision, submitted to the fusion center. In general, however, optimal decision rules are not threshold rules and this interpretation is no more valid. Rather, DD should be viewed as a problem of optimal quantization of the observation of each processor.

In that respect, Proposition 3.1 is reminiscent of the result of [3], namely, that the general problem of minimum distortion quantization is NP-complete.

### IV. RELATED PROBLEMS

The best known static decentralized problem is the team decision problem [8], [11] which admits an easy and elegant solution under linear quadratic assumptions. Its discrete version is the following.

**Team Decision Problem (TDP):** Given finite sets \(Y_1, Y_2, U_1, U_2\); a rational probability mass function \(p: Y_1 \times Y_2 \rightarrow Q\); and an integer cost function \(c: Y_1 \times Y_2 \times U_1 \times U_2 \rightarrow N\), find decision rules \(\gamma_i: Y_i \rightarrow U_i, i = 1, 2\) which minimize the expected cost

\[
J(y_1, y_2) = \sum_{y_1 \in Y_1, y_2 \in Y_2} c(y_1, y_2, \gamma_1(y_1), \gamma_2(y_2))p(y_1, y_2).
\]

Another problem related to DS is the following.

**Maximize Probability of Satifying (MPS):** Given finite sets \(Y_1, Y_2, U_1, U_2\); a rational probability mass function \(p: Y_1 \times Y_2 \rightarrow Q\); and a function \(S: Y_1 \times Y_2 \rightarrow 2^{U_1 \times U_2}\), find decision rules \(\gamma_i: Y_i \rightarrow U_i, i = 1, 2\) which maximize

\[
J(y_1, y_2) = P(\gamma_1(y_1), \gamma_2(y_2)) \in S(y_1, y_2)
\]

(which is the probability of making a satisfactory decision).

Given an instance of TDP, let

\[
S(y_1, y_2) = \{(u_1, u_2): c(y_1, y_2, u_1, u_2) = 0\}.
\]

If we solve TDP, we also effectively answer the question of whether \(J(y_1, y_2) = 0\). But this is equivalent to solving the instance of DS associated with the above definition of \(S(y_1, y_2)\). Therefore, TDP cannot be easier than DS. The same argument is also valid for MPS. It then follows from Proposition 2.2 that TDP and MPS are NP-hard (that is, NP-complete or worse) even if we restrict to instances for which \(|U_1| = |U_2| = 2\). However, even more is true: it suffices to notice that the problem DD of the previous section is a special case of both TDP and MPS, with \(|U_1| = |U_2| = 2\). Using Proposition 3.2, we obtain the following.

**Corollary 4.1:** TDP and MPS are NP-hard even if we restrict to instances for which \(|U_1| = |U_2| = 2\). This is true even if the cost function \(c\) associated to TDP is restricted to take only the values 0 and 1.

These results show that, unlike the linear quadratic case, the team decision problem is, in general, a hard combinatorial problem. The reason for this difference is the following: in the linear quadratic problem, Radner's theorem [11] guarantees that, as a consequence of the convex structure of the problem, a person-by-person optimal decision rule is also team optimal. This is no longer true for nonconvex (for example, discrete) team problems. While it may be argued that finding person-by-person optimal decision rules is relatively easy, there is no simple criterion for deciding whether a person-by-person optimal decision rule is also team optimal, and this is really the source of the difficulty. Let us also stress here that difficulties arising from the possibility of multiple person-by-person optima are equally relevant to team decision problems formulated on continuous decision and probability spaces, as is commonly done. In other words, the difficulties do not arise because we discretize an otherwise easy problem, but they are of a more fundamental nature. These issues will be discussed elsewhere in more detail.
V. Conclusions

The general conclusion of this paper is that even the simplest problems of decentralized decision making are hard from an algorithmic viewpoint. Moreover, this difficulty does not arise by an attempt to optimize with respect to a cost function, but persists even in the face of a qualitative (zero-one) cost criterion. Our results refer to discrete versions of such problems, but the continuous counterparts cannot be any easier, unless we are dealing with special, more structured cases. Of course, there are many objections to the idea that NP-completeness is an unequivocal measure of the difficulty of a problem because it is based on a worst case analysis, whereas the average performance of an algorithm may be a more relevant measure from a pragmatic point of view. However, such measures are hard to analyze, and an NP-completeness result is often a useful guide on what types of research questions should be pursued.

On a more specific level, since no simple algorithm could solve DS, and given that communications between processors are certainly required for those instances of DS which are "no" instances (i.e., there are no satisfying decision rules), it would not be a great loss if we allowed exchange of messages even for some instances for which this is not necessary. Even if these extra communications—being redundant—do not lead to better decisions, they may greatly facilitate the decision process and—from a practical point of view—remove some of the computational load.

Concerning the problem of decentralized detection, we have shown that it becomes hard, once the simplifying assumption of conditional independence is removed; this explains why little progress on the general version of this problem has followed the work of Tenney and Sandell [15].

For those decentralized decision problems for which communications are necessary, there arises naturally the problem of designing a communications protocol. Again, the problem of designing an "optimal" protocol that guarantees satisfying while minimizing the number of bits transmitted is intractable [10] even in simple cases (for \(|U_1| \geq 2, |U_2| \geq 3\), this follows from Proposition 2.2). Trying to centralize information in a "most efficient" manner, e.g., by using the smallest possible set of symbols, also leads to intractable problems [18]. Therefore, practical communications protocols can only be designed on a "good" heuristic or ad hoc basis and should not be expected to be optimal; approximate optimality is probably a more meaningful goal. Again, allowing some redundancy in on-line communications may lead to substantial savings in off-line computations.

Appendix

We start with a brief intuitive discussion of the essence of our proofs, so that the readers who are not interested in the technical aspects may still gain some general understanding.

The general methodology for proving that a certain problem \(P_l\) is NP-complete is the following [2], [9]. We choose a problem \(P_2\) which is known to be NP-complete (e.g., the Traveling Salesman Problem; a large bank of such problems is available [2]), and then we show that \(P_1\) is "harder" than \(P_2\), that is, \(P_2\) can be reduced in an efficient manner to \(P_1\).

For the specific type of problems considered in this paper, we make the following observation: the decisions of agents may be represented by a set of Boolean variables (at least one such variable is needed to represent the decision of an agent for each possible observation); each assignment of truth values (zero or one) to each set of Boolean variables corresponds to specifying a particular decision rule. Then questions of the type: "Is there a decision rule with a certain property?" are equivalent to questions of the form: "Is there an assignment of truth values to the Boolean variables such that a certain set of Boolean expressions are all true?" We then exploit the fact that certain problems of propositional calculus are known to be NP-complete. However, for our proof to be complete, we need to demonstrate that we can start from an arbitrary instance of the problem of propositional calculus under consideration and construct, in polynomial time, an instance of the decision problem under consideration. Carrying out this construction is usually the core of the proof.

Proof of Proposition 2.1: We will reduce to DS (with \(|U| = 2, M = 3\)) the satisfaction problem of propositional calculus with three literals per clause (3SAT) which is a known NP-complete problem [2]. Given an instance of 3SAT, let \(P\) be the set of literals and \(C\) the set of clauses. We construct an instance of DS as follows. Let

\[ Y_l = \{1, 2, \ldots, |Y|\}, \quad U_l = \{0, 1\}, \quad i = 1, 2, 3. \]

Let

\[ S(k, k) = \{(0, 0, 0), (1, 1, 1)\}, \quad k = 1, 2, \ldots, |Y|. \]

Finally, we interpret each clause in \(C\) as stating that a certain triple of decisions is not in the satisficing set. [For example, the clause \((x_1 \lor \neg x_2 \lor x_3)\) translates to the statement that the triple of decisions \((0, 1, 0)\) does not belong to \(S(i, j, k)\).] It is then easy to see that a satisfying assignment for the variables in \(V\) exists if and only if the above constructed instance of DS is a "yes" instance.

Proof of Proposition 2.2: 1)

a) If \(U_1 = 1\) or \(U_2 = 1\), the problem is trivial. If \(D_2 = 1\), a satisfying decision rule exists if and only if

\[ \bigcap_{(y_1, y_2) \in f} \pi_i(S(y_1, y_2)) = \emptyset, \quad \forall y_1 \in Y_1 \]

where \(\pi_i(u, u') \equiv u_i\). The above condition can be clearly tested in polynomial time (in fact, \(O(|Y_1|, |Y_2|)\) time).

b) This is the result of [10] mentioned in Section II.

c) Possibly by renaming, let \(Y_1, Y_2\) be disjoint sets. Consider the graph \(G = (Y_1 \cup Y_2, I)\). (Here \(Y_1 \cup Y_2\) is the set of nodes and \(I\) is the set of undirected edges.) Since \(D_1 = D_2 = 2\), each node of \(G\) has degree at most 2. Therefore, the connected components of \(G\) are either isolated nodes, chains, or cycles. Each connected component of \(G\) defines a subproblem and these subproblems are decoupled. So, without loss of generality, we may assume that \(G\) consists of a single connected component. We may also assume that \(G\) is a cycle (Fig. 2). (If it is a chain, we can introduce an addition edge to make it a cycle; this will not make the problem any easier because a new edge simply allows the presence of some more constraints.) In that case, \(Y_1\) and \(Y_2\) have the same number \(n\) of elements. Possibly by renumbering (see Fig. 2) we may assume that

\[ I = \{(i, i) : i = 1, \ldots, n\} \cup \{(i, i-1) : i = 2, \ldots, n\} \cup \{(1, n)\}. \]

Let us define

\[ S'(1, n-1) = \{(u_1, u_{n-1}) : (u_1, u_{n-1}) \in U_1 \times U_2 : \exists(u_n, u'_n) \in U_1 \times U_2 \]

such that

\[ (u_n, u'_n) \in S(n, n-1), \quad (u_i, u'_i) \in S(n, i-1), \]

and note that \(S'(1, n-1)\) may be evaluated in \(O(|U|^2)\) time. We now have the following. An instance of DS is a "yes" instance if

\[ \exists(u_1, u_2, \ldots, u_n) \in S(i, i), \quad i = 1, \ldots, n \]

and

\[ \exists(u_n, u'_n) \in S(1, n) \]


3 Throughout, we use the symbols \(\wedge, \lor, \neg\) to denote the Boolean operations logical AND, logical OR, negation, respectively.
Moreover, there are at most \( \min(j) \) solutions of \((A.1)\) for each \(j\), which is equivalent to
\[
\sum_{u_1, \ldots, u_n} \prod_{j=1}^m \left( \prod_{i=1}^n \left( 1 + \prod_{y \in Y_j} x_1(y, i) \right) \right) \leq \prod_{j=1}^m \left( \prod_{i=1}^n \left( 1 + \prod_{y \in Y_j} x_1(y, i) \right) \right).
\]

Consider also the statement
\[
\sum_{u_1, \ldots, u_n} \prod_{j=1}^m \left( \prod_{i=1}^n \left( 1 + \prod_{y \in Y_j} x_1(y, i) \right) \right) \leq \prod_{j=1}^m \left( \prod_{i=1}^n \left( 1 + \prod_{y \in Y_j} x_1(y, i) \right) \right).
\]

which is equivalent to
\[
\sum_{u_1, \ldots, u_n} \prod_{j=1}^m \left( \prod_{i=1}^n \left( 1 + \prod_{y \in Y_j} x_1(y, i) \right) \right) \leq \prod_{j=1}^m \left( \prod_{i=1}^n \left( 1 + \prod_{y \in Y_j} x_1(y, i) \right) \right).
\]

Therefore, we only need to show that the truth of \((A.2)\) can be decided in polynomial time. Note that, for each \((i, j)\), the set \(S'\) may be constructed in time \(O(Y_j | Y_i)\).

Moreover, there are at most \( \min |Y_j| |Y_j| \) pairs to be considered; so the sets \(S'\) may be constructed in time \(O(Y_j | Y_i) \min |Y_j|^2, |Y_i|)\). One \(S'\) is constructed, the statement \((u_i, u'_j) \in S'(i, j)\) may be expressed as a set of clauses with two literals per clause (the literals are the Boolean variables \(u_i, u'_j\); this is similar to the proof of part b); see [10]). Therefore, deciding the truth of \((A.3)\) is a special case of the satisfiability problem with two literals per clause \((2SAT)\), which can be solved in linear time [9].

2) Consider the following restricted version of the satisfiability problem for propositional calculus with three literals per clause \((3SAT)\). An instance of this restricted problem (which we call \(RSA\) that consists of the following: a set of Boolean variables \(F = F_1 \cup F_2\) where \(F_1 = \{y_j: i = 1, m, j = 1, 2, 3\}, F_2 = \{x_i: i = 1, n\}\).

Conversely, assume \((A.2)\) holds. Then it is easy to see that \((A.2)\) holds as well,
\[
\sum_{u_1, \ldots, u_n} \prod_{j=1}^m \left( \prod_{i=1}^n \left( 1 + \prod_{y \in Y_j} x_1(y, i) \right) \right) \leq \prod_{j=1}^m \left( \prod_{i=1}^n \left( 1 + \prod_{y \in Y_j} x_1(y, i) \right) \right).
\]

For this reason, it only remains to prove the following.

**Lemma 4.1:** \(RSA\) is NP-complete even if it is restricted to instances for which \(H_1 = 3, H_2 = 2\).

Proof: Given that \(RSA\) is NP-complete [10], it is sufficient to start with a general instance of \(RSA\) and reduce it to a new instance for which \(H_1 = 3, H_2 = 2\). This is accomplished by creating multiple copies of each variable so that, instead of having the same variable appear in many clauses, distinct copies of it are used. Of course, some clauses will be needed to create the multiple copies, but these can be kept to a small number. The idea of the proof is best shown diagrammatically, as in Fig. 3. (A more formal argument may be found in [16], but the present one is easier to visualize.)

Let \(y_1, y_2, y_3\) be variables in a given instance of \(RSA\). We introduce some new \(z\) variables, as well as some clauses which guarantee, for example (see Fig. 3), that \(z_1 = y_1, z_2 = y_2, z_3 = z_4 = z_5 = y_{13}\). (For example, the clauses
\[
(1 - z_2) V (1 - z_3) \wedge (1 - z_2) V (1 - z_3) \wedge (1 - z_2) V (1 - z_3),
\]

together with a requirement that exactly one of the variables \(z_2, z_3, z_4\) is true, imply \(z_5 = z_6 = 1\). In this way, we can effectively create an arbitrary number of copies of the \(y\) variables, and the same procedure works for the \(x\) variables as well. Note that in doing so, we have respected the requirement that \(H_1 = 3, H_2 = 2\). Finally, for each clause in the original instance of \(RSA\), we may introduce a clause involving appropriate copies, and it is easy to see that the requirement \(H_1 = 3, H_2 = 2\) will still be respected, as long as we use a different copy each time. Moreover, since an arbitrary number of copies may be created by the above

\[\text{The arcs in Fig. 3 indicate the variables that have a common clause; solid lines indicate that two variables are constrained to be equal; a curve encircling a triple of variables in } F_2 \text{ indicates that these are to be merged to a single node.} \]
such that evaluate to compute \( r(n) \) in time we have a "yes" instance of obtain has at most \( r(k) \). Therefore, given \( u_k-D \)

Thus handled. The construction of copies of the original variables.

**Remark:** The algorithm in the proof of Proposition 2.3 does not find a satisfying decision rule; it only determines whether one exists. However, satisfying decision rules may be computed by keeping in the memory some of the intermediate results produced by the algorithm.

**Proof of Proposition 3.1:** Consider the following problem of propositional calculus which we call \( P \).

**Problem P:** We are given two sets \( X = \{x_1, \ldots, x_n\} \), \( Z = \{z_1, \ldots, z_m\} \) of Boolean variables, a set \( D \) of (distinct) clauses of the form \( x_i \land z_j \) or \( \neg(x_i \land z_j) \) (we assume that for any pair \((i,j)\) at most one of the above clauses is in \( D \)), a collection \( \{q_i: i \in \{1, \ldots, n\}, j \in \{1, \ldots, m\}\} \), \( n \in \{1, \ldots, n\} \). Suppose that \( X \), \( Z \) is there a truth assignment for \( X \) and \( Z \) such that \( J \subseteq K \) where

\[
J = \sum \{q_i: i \in \{1, \ldots, n\}, j \in \{1, \ldots, m\}\} \quad (A.5)
\]

**Lemma A.2:** Problem \( P \) is equivalent to \( DD \).

**Proof:** Think of \( X, Z \) as being the sets of observations of processors \( S_1, S_2 \), respectively. A truth assignment to \( X, Z \) corresponds to a choice as to what binary message to transmit to the fusion center, given each processor’s observation. Let \( H_0 \) (respectively, \( H_1 \)) be the hypothesis that \((i,j) \in A_0 \) (respectively, \( A_1 \)). Finally, view \( q_{ij} \) as the (unnormalized) probability that the pair \((i,j)\) of observations is obtained by the two processors. Pairs \((i,j)\) that belong to neither \( A_0 \) nor \( A_1 \) may be viewed as having zero probability and are, therefore, of no concern. Then it is easy to verify that \( J, \) as defined by (A.5), is precisely the (unnormalized) probability of error.

In order to complete the proof of the proposition, we need to show that \( P \) is NP-complete. This will be accomplished by reducing to \( P \) the following (maximum 2-satisfiability) problem of propositional calculus which is known to be NP-complete [2].

**MAX-2-SAT:** Given a set \( U \) of Boolean variables, a collection \( C \) of (distinct) clauses over \( U \), such that each clause \( c \in C \) has exactly two variables and an integer \( K \leq |C| \), is there a truth assignment for \( U \) which simultaneously satisfies at least \( K \) of the clauses in \( C \)? (Without loss of generality, we assume that if a clause is in \( C \), then its negation is not in \( C \).)

Suppose that we are given an instance \((U, C, K)\) of \( MAX-2-SAT \). We construct an instance of \( P \) as follows. Suppose that \( U = \{u_1, \ldots, u_n\} \). Then, let \( X = \{x_1, x_2, x_3: i = 1, \ldots, n\} \) and \( Z = \{z_1, z_2, z_3: i = 1, \ldots, n\} \). For each \( i \in \{1, \ldots, n\} \), introduce the set \( D_i \) of clauses:

\[
\neg(x_i \land z_1), \ (x_2 \land z_2), \ \neg(x_2 \land z_3), \ (x_3 \land z_1), \ \neg(x_2 \land z_1), \ (x_3 \land z_1), \ \neg(x_1 \land z_2), \ (x_2 \land z_2), \ \neg(x_1 \land z_3), \ (x_3 \land z_1).
\]

To these clauses, we assign the weights (\( L \) is a large integer to be determined later):

\[
q_{i_1,a} = 30L, \ q_{2,a} = 15L, \ q_{a_1} = 4L, \ q_{a_2} = 20L
\]

\[
q_{a_3} = 8L, \ q_{a_4} = 2L, \ q_{a_5} = 2L, \ q_{a_6} = 100L.
\]

Next, for each clause \((u_i \land u_j), \ (\neg u_i \land u_j), \ (\neg u_i \land \neg u_j), \ (u_i \land \neg u_j), \ (\neg u_i \land \neg u_j)\) in \( C \), introduce clauses \((x_1 \land z_1), \ (x_2 \land z_2), \ (x_3 \land z_3), \ (\neg x_1 \land z_2), \ (\neg x_1 \land z_3), \ (\neg x_2 \land z_1), \ (\neg x_2 \land z_3)\) in \( D_0 \) and assign to each one unit weight. We now let \( D = D_0 \) and observe that \( X, Z, D, \{q_i\} \) define an instance of \( P \).

Note that for any assignment for \( X, Z \), the corresponding cost
where \( J_l, l \in \{0, 1, \ldots, n\} \) is the sum of the weights \( q_l \) of the clauses in \( D_l \) which are not satisfied.

**Lemma A.3:** For any \( i \in \{1, \ldots, n\} \), we have \( J_i = 35L \) if and only if at least \( n \) clauses in \( D_i \) are not satisfied, which is equivalent to the original set \( C \).

**Proof:** The proof is by direct evaluation of \( J \) for each possible assignment. See [16].

In view of Lemma A.3, the clauses in \( D_i \) and their associated weights have the following interpretation: the variable \( x_i \) may be freely assigned, but the remaining variables must be assigned so that \( x_{i1} = x_{i2} = \cdots = x_{i4} = 1 \). For this reason, the clauses in \( D_i \) are effectively the same as the original set \( C \) of clauses.

**Lemma A.4:** Let \( L \) be large enough so that \( |C| < L \). Then there exists a truth assignment for \( U \) for which at least \( K \) clauses in \( C \) are satisfied if and only if there exists a truth assignment for \( X, Z \) such that the resulting cost \( J \) is less than or equal to \( 35nL + |C| - K \).

**Proof:** a) Given an assignment for \( U \), with at least \( K \) clauses satisfied, assign the variables in \( X, Z \) as follows:

\[
\begin{align*}
x_{i1} &= z_{i1} = 1, \\
x_{i2} &= z_{i2} = -1, \\
x_{i3} &= z_{i3} = 0.
\end{align*}
\]

Using Lemma A.3 and the identity (A.6), the resulting cost is \( 35nL \) (i.e., \( 35L \) from each collection \( D_i, i = 1, \ldots, n \)) plus the number of clauses in \( D_k \) which are not satisfied (since these carry unit weight). The latter number is identical to the number of clauses in \( C \) which are not satisfied, which is less than or equal to \( |C| - K \).

b) Conversely, given an assignment for \( X, Z \) such that \( J \leq 35nL + |C| - K \), suppose that for some \( i \in \{1, \ldots, n\} \), \( J_i \geq 37L \). Using Lemma A.3 and the inequality \( |C| < L \), we obtain

\[
J_i \geq 35nL + 2L > 35nL + |C| - K
\]

which is a contradiction and shows that \( J_i = 35L \), \( \forall i \). Consequently, \( \{x_{i1}, x_{i2}, z_{i1}, z_{i2}\} \) have been assigned values in one of the two ways suggested by Lemma A.3. We now assign truth values for \( U \) by setting \( u_i = x_{i1} \). Then \( J_0 \) is the number of clauses in \( C \) which are not satisfied. Moreover, since \( J_0 = 35L, i \in \{1, \ldots, n\} \), it follows that \( J_0 \leq |C| - K \), which implies that at least \( K \) clauses in \( C \) are satisfied. This completes the proof of Lemma A.4.

It is easy to see that the above reduction of \( \text{MAX}-2\text{-SAT} \) to \( P \) is polynomial. Therefore, \( P \) is NP-complete and so is \( DD \), thus completing the proof of the proposition.

**References**


John N. Tsitsiklis (S’80-M’81) was born in Thessaloniki, Greece, on February 18, 1958. He received the B.S. degrees in electrical engineering and mathematics in 1980, the M.S. degree in electrical engineering in 1981, and the Ph.D. degree in electrical engineering in 1984, all from the Massachusetts Institute of Technology, Cambridge.

He spent the 1983-1984 academic year at Stanford University, Stanford, CA, as an Assistant Professor of Electrical Engineering. He is currently an Assistant Professor of Electrical Engineering at the Massachusetts Institute of Technology. His research interests are in decentralized decision making, estimation, and computation, as well as stochastic control.

Dr. Tsitsiklis is a member of Phi Beta Kappa and Sigma Xi.

Michael Athans (S’58-M’61-SM’69-F’73) received the Ph.D. degree in electrical engineering in 1961 from the University of California, Berkeley.

From 1961 to 1964 he was with the M.I.T. Lincoln Laboratory, Lexington. Since 1964 he has been a faculty member in the M.I.T. Department of Electrical Engineering and Computer Science, Cambridge, where he currently holds the rank of Professor of Systems Science and Engineering. While at M.I.T., he also served as Director of the M.I.T. Laboratory for Information and Decision Systems from 1974 to 1981. He is a co-founder of ALPHATECH, Inc., Burlington, MA. He has also consulted for several other industrial organizations and government panels.