

## The NP-Completeness Column: An Ongoing Guide

DAVID S. JOHNSON

*AT&T Bell Laboratories, Murray Hill, New Jersey 07974*

This is the thirteenth edition of a quarterly column that provides continuing coverage of new developments in the theory of NP-completeness. The presentation is modeled on that used by M. R. Garey and myself in our book "Computers and Intractability: A Guide to the Theory of NP-Completeness," W. H. Freeman & Co., New York, 1979 (hereinafter referred to as "[G&J]"; previous columns will be referred to by their dates). A background equivalent to that provided by [G&J] is assumed, and, when appropriate, cross-references will be given to that book and the list of problems (NP-complete and harder) presented there. Readers who have results they would like mentioned (NP-hardness, PSPACE-hardness, polynomial-time-solvability, etc.), or open problems they would like publicized, should send them to David S. Johnson, Room 2C-355, AT&T Bell Laboratories, Murray Hill, NJ 07974 (CSNET address: dsj.btl@csnet-relay). Please include details, or at least sketches, of any new proofs (full papers are preferred). If the results are unpublished, please state explicitly that you would like them to be mentioned in the column. Comments and corrections are also welcome. For more details on the nature of the column and the form of desired submissions, see the December 1981 issue of this Journal.

### 1. INTRODUCTION

This month I resume the discussion of parallel and distributed computing begun in the [June 1983] and [Sept. 1983] columns. The emphasis this time will be on the complexity issues that arise when parallel processes need to communicate with one another. The levels of communication discussed will range from the primitive semaphores of Dijkstra [9] to such sophisticated mechanisms as the long distance lines of my employer (and its competitors).

Section 2 considers complexity results reminiscent of the results about resource sharing that were discussed in [Sept. 1983]. Here we ask how difficult it is to tell whether a given system of communicating processes will behave properly, or will at least avoid committing such standard *faux pas* as deadlock, lock-out, etc. Section 3 treats what has come to be called "communication complexity." This is a complexity theory where the quantity we wish to minimize is not time (measured in Turing machine steps) or space (measured in tape cells), but communication (measured in transmitted bits). In addition to providing an

attractive source of new NP-hard problems, this theory also provides examples of *provable* exponential gaps between determinism and non-determinism.

In the computational models underlying the problems of Sections 2 and 3, the act of communication is viewed as a system primitive; the mechanics of *how* a message is to be transmitted are ignored. One message delivery system is the *communications network*, with nodes corresponding to the parallel processes (or processors), and edges corresponding to the communication links between them. Section 4 surveys complexity results about the operation of such networks, from the routing and scheduling of messages to the testing of transmitters and links.

I conclude in Section 5 with a brief return to the topic of the previous column (randomized complexity) in order to correct a statement I made about that column's "Open Problem of the Month." Readers knowing of other errors or misstatements in previous columns are urged to communicate them to me. In the near future I hope to correct and update the first dozen or so editions of the NP-Completeness Column and issue them as a book. Watch this space for further details.

## 2. ETIQUETTE FOR COMMUNICATING PROCESSES

There are many aspects to the question of proper comportment for communicating processes. As with other cultural rituals, elaborate patterns of behavior may be required. Readers of my [Sept. 1983] column on the sharing of resources will recall that many pages were devoted just to describing the basic social situations that gave rise to the problems under consideration. In the interests of brevity I shall be a bit sketchier this time, looking for the unifying principles rather than the behavioral minutiae. Readers who are interested in the full details are encouraged to look up the primary sources (or to do field work of their own).

A recurrent theme in what follows is that a little communication ability can yield a large increase in computing power. For instance, consider the deterministic finite automaton, for which most problems (i.e., string acceptance, language equivalence, etc.) are solvable in polynomial time. Suppose we have two such machines, each augmented by an input queue. Based on its current state and the current symbol heading its input queue, each machine can (a) change state, (b) pop the symbol at the head of its queue, (c) send a symbol to the tail of the other machine's queue, or (d) any combination of the above. Given two such machines with initially empty queues, the problem of determining whether one of them will eventually halt in an "accept" state is *undecidable* [8]. This is because the message queues can be used to simulate the tape of an arbitrary Turing machine. The second machine merely sends each symbol it receives back to the first machine, thus creating a loop that can grow arbitrarily large, so long as there is no bound imposed on the queue length. Thus we face the halting problem and undecidability. The level of intractability can be reduced to exponential

space completeness if a bound on queue length is part of the input, and to PSPACE-completeness if the bound is written in unary [8]. The problem does not become polynomial until the bound is finite and fixed, independent of the input.

If we allow the number of automata to vary with the instance, then we face PSPACE-hardness even if all queue lengths are fixed at 1, since now the automata themselves can be used to simulate the tape cells [6]. Moreover, a non-deterministic variant on this finite queue-length model can simulate *alternating* PSPACE (and hence exponential time — see the [Dec. 1983] column) by asking the question of whether “lockout” can be prevented [16]. (A process is *locked out* if there is some way by which the remaining processes can collude to prevent it from making progress, no matter what it does; note the game-like alternation implicit in this definition.)

Communication can enhance still simpler computational models, as can be seen in the next two results concerning the use of communication to coordinate processes in an asynchronous environment. In the context of the synchronization mechanism embodied by Dijkstra’s semaphores and “P” and “V” primitives [9], DeMillo and Miller show that a system of straightline processes can simulate an arbitrary polynomial time nondeterministic Turing machine (the machine accepts if and only if there is a way for some pair of processes in the system to violate a prespecified prohibition on mutual state pairs) [7]. In the context of the synchronization mechanism known as “rendezvous” in the language Ada, Taylor [25] shows that many problems are NP-hard even if all the processes are finite-state and acyclic (have no loops in their control structures).

There is, however, a limit on how simple one can make the machines (and the property to be verified), while still retaining NP-hardness. Suppose the machines are all simple loop programs (an initialization sequence followed by an infinite loop) and operate synchronously, communicating by either sending a symbol to a particular neighboring machine or reading a symbol sent by a particular neighbor. The problem of telling whether the system’s behavior is *strongly coordinated* (i.e., corresponding sends and reads always occur during the same time step) can be solved in polynomial time, as can a problem involving a slightly weaker notion of coordination. Also polynomial-time solvable are extensions of these problems to a more general machine model that shares the loop program’s inability to use the symbols it reads to direct the flow of its control [6]. The key observation here is that under these restrictions the problem decomposes into a collection of independent problems, one for each pair of machines that communicate with each other. (The problem recomposes itself if arbitrary machines are allowed, and becomes PSPACE-complete [6].)

The above results suggest that, if we want to find tractable aspects of the general problem of communicating processes, we would do well to concentrate initially on the two-machine case. One result of such concentration is the new form of complexity theory discussed in the next section.

### 3. COMMUNICATION COMPLEXITY

It has long been recognized that, in distributed systems, communication is expensive. In addition to the direct cost that may be associated with each message sent (as with the charges assessed to users of a telephone network), there is also the indirect cost associated with the possibility that if too many messages are simultaneously in transit, they may clog the system and degrade its overall performance. Thus theoreticians studying distributed systems have been interested in minimizing the amount of communication as well as (or instead of) minimizing such standard quantities as running time and space.

For instance, communication is the main complexity measure considered in recent papers [12,14] on the problem of electing a leader in a ring of communicating processes (a problem mentioned in the [Sept. 1984] column). Such an emphasis on minimizing communication is not surprising for problems like this one, which makes sense only in a distributed environment. Given the rudimentary nature of the individual processes in many models, space (the amount of memory associated with the processes) may not be a crucial resource, and, especially in asynchronous models, there may be no realistic way to define “running time.” A more recent development is the attempt to use communication as a complexity measure for the more traditional type of computational problem in which one is given an input  $x$  and asked for the value of a function  $f$  on that input.

When researchers first considered the problem of computing such functions using parallel processors, they were mainly interested in the speedups that could be obtained, and communication costs were either ignored or included in the running time. (Some of these issues were covered in the [June 1983] column.) However, when attention was turned toward implementing the parallel computation of functions on VLSI chips, it became clear that communication costs were more than just a component of running time. As first observed by Thompson [26,27], the necessity that information be communicated from the inputs to the outputs of a chip imposes a trade-off between its speed and area. To be specific, if a chip  $C$  computes a function  $f$ , the communication implicit in  $f$  imposes a lower bound on the product  $AT^2$ , where  $A$  is the area of the chip and  $T$  is its running time.

Here “communication” is measured in terms of a simple model involving two communicating processes. As with the Boolean “circuit complexity” described in the [June 1983] column, this is a measure that is defined separately for each input size  $n$ , without requiring that the processes for different values of  $n$  be related. For a given  $n$ , let  $f_n$  be the restriction of  $f$  to  $n$ -bit inputs and suppose that each process is to be provided with half the input bits and that the second is to output the answer. For a given protocol  $P$  that enables the processes to compute  $f_n$  and a given partition  $\pi$  of the input bits, denote by  $c(f_n, P, \pi)$  the maximum number of “communication” bits that are transmitted between the

processes, over all  $n$ -bit inputs  $x$ . Then let  $c(f_n, \pi)$  be the minimum of this quantity over all such  $P$ , and  $c(f_n)$  the minimum of  $c(f_n, \pi)$  over all partitions of the input bits into two equal-size sets. The lower bound on  $AT^2$  is then  $AT^2 = \Omega(c(f_n)^2)$  [18,35].

Note that the largest lower bound that this measure can provide is  $\Omega(n^2)$ , since  $c(f_n, \pi) = O(n)$  for all  $\pi$ . (Process 1 can simply send all its input bits to Process 2, and let the latter do all the computing.) This maximum lower bound is achieved by a variety of functions, from sorting [27] and the Discrete Fourier Transform [26,27] to such simple decision problems as determining whether a given pattern occurs in a string [18]. There are also, of course, functions with communication complexity substantially less than  $n$ . For instance, determining whether the number of 1-bits in the input is odd requires at most 1 bit of communication, no matter how the input bits are partitioned between processes.

Of course this last observation is irrelevant as far as VLSI lower bounds go. It would provide a lower bound of only  $\Omega(1)$ , whereas any chip that computes a function that depends on  $n$  input bits will have  $AT^2 \geq AT = \Omega(n)$ . (Those input bits arriving in parallel use up area and those arriving in sequence use up time.) The range in possible growth rates for  $c(f_n, \pi)$  as a function of  $n$  suggests, however, that this communication complexity measure can be used to make interesting distinctions. It is such distinctions that I will investigate in the remainder of this section. Readers interested in learning more about area-time trade-offs in VLSI are directed to the substantial literature that now exists on the subject; for a start, see [18,19,26,27,30,33,35].

The idea of studying a complexity measure like  $c(f_n, \pi)$  as an independent topic was first raised (in a different context) by Abelson [1], who was interested in the amount of communication needed when two processes compute a real-valued function of their combined (real-valued) inputs by interchanging real values. Yao [34] adapted this idea to the computation of finite functions by the exchange of bits. In both models, the input partition is fixed. In the discrete case this means that  $f$  can be viewed as a function of two equal-length variables  $x$  and  $y$ , where the bits of  $x$  are assigned to Process 1 and the bits of  $y$  are assigned to Process 2. The two processes proceed by alternately sending each other bits (one per message) until Process 2 has sufficient information to determine the correct answer. (The requirement that transmissions alternate is not a significant restriction, as it can be fulfilled using dummy bits if one process needs to be more loquacious than the other.) The communication complexity  $c(f_{2n})$  is then the minimum number of communication bits that suffice for such a pair of processes to compute  $f(x, y)$  for all pairs of length- $n$  inputs  $x$  and  $y$ .

This definition ignores the amount of *time* needed to compute  $f(x, y)$  (for a study of trade-offs between communication and running time, see [32]). There is, however, one running time question that naturally arises in conjunction with the definition: How difficult is it to compute  $c(f_{2n})$  itself? The answer to this question follows immediately from this month's first "official" NP-

completeness result:

### [1] COMMUNICATION COMPLEXITY

INSTANCE: Two finite sets  $X$  and  $Y$  of strings of length  $n$ , finite function  $f_{2n}: X \times Y \rightarrow \{0,1\}$ , positive integer  $K$ .

QUESTION: Is  $c(f_{2n}) \leq K$ ?

*Reference.* Papadimitriou and Tsitsiklis [22]. Transformation from EXACT COVER.

*Comment.* That this problem is in NP is perhaps more surprising than that it is NP-hard, and follows from a characterization due to Yao [34]. If  $f_{2n}$  is viewed as a 0-1 result matrix with rows labelled by elements of  $X$  and columns labelled by elements of  $Y$ , then  $c(f_{2n}) \leq K$  if and only if the matrix can be partitioned into  $K$  monochromatic submatrices (ones whose entries are either all 0 or all 1). Since such a partition can be guessed and checked in polynomial time, the problem is in NP. Reference [22] also considers a related problem. Suppose the set  $R$  of possible outputs is expanded beyond  $\{0,1\}$  and the two processes are allowed to output different values so long as the pair of output values is “compatible,” where the compatibility relation is given by a function  $r$  from  $X \times Y$  to subsets of  $R \times R$ . Determining communication complexity in this model is also NP-complete, even if we ask whether the communication complexity is 0, i.e., if the outputs can be computed without any communication at all. (This zero-communication problem is, however, solvable in polynomial time if  $|R| \leq 2$ , in which case it is equivalent to 2-SATISFIABILITY.)

Fortunately, the difficulty of computing  $c(f_{2n})$  exactly does not prevent us from obtaining interesting results about communication complexity. Indeed, as we have already seen, it is possible to obtain non-trivial lower bounds on the communication complexity of specific problems in NP, something that is beyond the current state of the art for time and space complexity. Let us now restrict attention to 0-1 valued functions  $f$ , that is, to decision problems. A first result, proved nonconstructively by Papadimitriou and Sipser [21], is an analog of the standard hierarchy results involving space and time. For any function  $t$  such that  $1 < t(n) \leq n$  for all  $n$ , let  $\text{COMM}(t(n))$  denote the set of decision problems  $f$  such that  $c(f_{2n}) \leq t(n)$  for all  $n > 0$ . Then for each such  $t$ ,  $\text{COMM}(t(n) - 1)$  is strictly contained in  $\text{COMM}(t(n))$ .

More entertaining results are possible if we introduce the notion of *nondeterministic* communication complexity, with the standard criterion that an input be accepted if there is *any* accepting computation possible. This allows us to phrase an “admittedly far-fetched” [21] analogy of the P versus NP question: Is there a decision problem whose deterministic communication complexity is exponentially greater than its nondeterministic communication complexity? (It is

not difficult to see that no larger gap is possible [21].) Surprisingly, we can do more than exhibit likely candidates for such problems (which is the best we have been able to do in the case of P versus NP). Here we can in certain cases *prove* that the exponential gap exists.

An example is the problem “Given a graph  $G$ , does it contain a triangle?” We assume that each process is given half the entries in the adjacency matrix for  $G$  (this involves  $N(N - 1)/4$  bits if  $G$  has  $N$  vertices). Process 1 need only nondeterministically choose a set of three vertices for which it knows of no missing edges, and transmit their names. If Process 2 also knows of no missing edges, the three vertices induce a triangle in  $G$ , and the processes answer yes. If  $G$  has  $N$  vertices, vertex names will require  $\log N$  bits, so the communication complexity is  $O(\log N)$ . On the other hand, one can prove in a page or so (if “one” is Papadimitriou and Sipser [21]) that this problem has essentially the worst possible deterministic communication complexity, i.e.,  $c(f_{2n}) = \Omega(N^2)$ . This bound holds even for *non-deterministic* protocols, so long as they are computing the *complement* of our original question (checking that *no* triangles exist). Hence  $NP \neq \text{co-NP}$  for communication complexity, an even stronger statement than  $P \neq NP$ .

If one is willing to settle for polynomial rather than exponential gaps, one can also show that, for communication complexity,  $P \neq NP \cap \text{co-NP}$ , or, more precisely, that the deterministic communication complexity of a 0-1 function  $f$  can be strictly greater than the nondeterministic communication complexities of both  $f$  and its complement. Consider the problem where Process 1 and Process 2 are each given a sequence of  $N$  length- $N$  strings and must determine whether there is a  $k$  such that the  $k^{\text{th}}$  entries in the two lists are identical. Mehlhorn and Schmidt [19] show that the deterministic communication complexity of this problem is  $\Omega(n^2)$ , whereas both it and its complement have  $O(n \log n)$  nondeterministic communication complexity.

Nondeterminism does not provide the only interesting variant on our “standard” communication complexity. For instance, several of the protocols I have described have been “1-way” rather than “2-way,” since they require only that information be transmitted from Process 1 to Process 2, and not vice versa. This raises the question of whether 2-way communication is truly more powerful than 1-way communication. As one might expect, it is. Once again we can get an exponential gap between the two communication complexities (although no worse than exponential) [21]. One can also ask about “randomized” communication complexity, in analogy with the randomized computational complexity discussed in the [Sept. 1984] column. An intriguing result here concerns the problem where Processes 1 and 2 are each given an  $n$ -bit number and must decide whether the two numbers are equal. In [23], Paturi and Simon describe a randomized protocol whereby the processes can achieve the correct answer with probability exceeding  $1/2$  merely by the exchange of 2 bits!

Finally, let me discuss some results of Lakshminpathy and Winklmann [17] on

a variant of communication complexity in which one can prove an exponential gap between certain NP-complete problems and their polynomial-time solvable counterparts. Suppose a graph  $G = (V, E)$  has a *vertex separator*  $S$ , i.e.,  $V$  can be partitioned into three sets  $V_1$ ,  $V_2$ , and  $S$ , such that there are no edges in  $G$  joining vertices in  $V_1$  to vertices in  $V_2$ . Suppose further that  $S$  is small relative to  $V$ . This suggests a particularly natural way of partitioning the description of  $G$  between two processes, involving a small amount of possibly useful redundancy. The input to Process 1 is the subgraph of  $G$  induced by  $S \cup V_1$  and the input to Process 2 is the subgraph induced by  $S \cup V_2$ . Each process is told which of its vertices belong to  $S$ , and both know these vertices under the same names. If the properties of  $G$  we wish to determine are sufficiently “local,” we may now be able to get away with far less communication than if each process received an arbitrary portion of  $G$ ’s adjacency matrix.

In order to get at this issue of “local” versus “global” properties, Lakshminpathy and Winklmann choose to examine how the need for communication varies with the size of the separator  $S$ , rather than the size of the input. For a protocol  $P$  that enables the two processes to compute  $f$ , let  $c(f, s, P)$  be the maximum number of bits communicated, taken over all graphs with separators of size  $s$  and over all such separators. The new communication complexity measure for  $f$  is then the minimum of  $c(f, s, P)$  over all such  $P$ , denoted by  $c(f, s)$ . The results in [17] show that for a large collection of problems straddling the border between NP-completeness and polynomial-time solvability (such as GRAPH 3-COLORABILITY and GRAPH 2-COLORABILITY, PARTITION INTO TRIANGLES and PERFECT MATCHING, HAMILTONIAN CIRCUIT and EULER CIRCUIT, etc.), the NP-complete problems have communication complexities that are exponential in  $s$ , while the polynomial-time solvable problems have communication complexities that are polynomial in  $s$ .

Unfortunately (there had to be an “unfortunately,” didn’t there?), there is no hope that this approach can lead to a universal communication gap between P and NP. First, it seems to make sense only in the context of graph problems, and even here there seems to be no sensible way to extend it to problems about weighted graphs, or to problems like VERTEX COVER or MAXIMUM MATCHING that contain a number in the input. (If one wants only to *build* an optimum cover or matching, rather than test its size, the extension of the above results to problems of this type *is* possible [17].) More importantly, however, the approach doesn’t even handle all unweighted (and un-numbered) graph problems correctly. Observe that, as defined,  $c(f, s)$  might well be infinite, and indeed *will* be infinite if the amount of communication needed depends on the size of the graph. (An argument is required to show that the above NP-complete problems have *only* exponential complexity.) One problem where the need for communication grows with the size of the graph is graph isomorphism. Two graphs  $G_1$  and  $G_2$  can be viewed as a single graph  $G$  with a separator of size 0; if  $f$  is the function whose value is 1 if and only if  $G_1$  and  $G_2$  are isomorphic,

then  $c(f,0) = \infty$ . This is true even if both graphs are trees. However, tree isomorphism is solvable in polynomial time, and so there are polynomial-time solvable problems with super-polynomial communication complexity. On the other hand, I haven't been able to find an NP-complete problem that has polynomial communication complexity. The question of whether such a problem exists is thus left as the "Open Problem of the Month."

#### 4. COMMUNICATION NETWORKS

This section surveys recent NP-completeness results about communication networks. In keeping with the theme of this column, I concentrate on problems involving the *operation* of such networks, rather than their design. The first problem models an attempt to minimize telephone bills by coordinating the transfers of files between distant computers, scheduling the transfers to take place late at night during the period when telephone rates are lowest.

##### [2] FILE TRANSFER SCHEDULING

**INSTANCE:** Multigraph  $G = (V,E)$ , a positive integer length  $L(e)$  for each edge  $e \in E$ , a positive integer *port capacity*  $c(v)$  for each vertex  $v \in V$ , and a positive integer deadline  $D$ . Here an edge  $\{u,v\}$  represents a file to be transmitted telephonically between computers  $u$  and  $v$ , and the length of the edge is the amount of time the transfer will take. The port capacity  $c(v)$  is the number of phone lines possessed by computer  $v$ , and hence the maximum number of transfers it can perform simultaneously.

**QUESTION:** Can one schedule the edges of  $G$ , subject to the port capacities, so that all finish before the deadline? More formally, is there a function  $s$  that assigns a non-negative integer start time to each edge such that (i) for each vertex  $v \in V$  and each non-negative time  $t \leq D$ , there are no more than  $c(v)$  edges that have  $v$  as an endpoint and satisfy  $s(e) < t \leq s(e) + L(e)$  (the port capacities are obeyed) and (ii) for each edge  $e \in E$ ,  $s(e) + L(e) \leq D$  (the transfers all finish by the deadline)?

*Reference.* Coffman, Garey, Johnson, and LaPaugh [5]. Transformations from PARTITION, 3-PARTITION, and CHROMATIC INDEX.

*Comment.* NP-complete in the strong sense even if  $|V| = 2$ . For cycles (i.e., our familiar "ring of processors") and unit port capacities, the problem is solvable in polynomial time if the cycle is even and NP-complete (in the ordinary sense) if the cycle is odd. If all edge lengths are equal, the problem looks much like an edge coloring problem, and so is NP-complete for general graphs and solvable in polynomial time for bipartite ones. See [5] for a complete complexity classification, leaving only one subcase with unresolved complexity. A

generalization to scheduling the edges of hypergraphs is studied in [15], where the hyper-edges model the shared execution of tasks by sets of contiguous processors.

In the above problem, communication must go directly from sender to receiver; forwarding is not allowed. The next problem considers networks based on packet switching, where forwarding is a basic operation.

### [3] DEADLOCK EXPOSURE IN PACKET SWITCHING NETWORKS

INSTANCE: Graph  $G = (V, E)$ , a positive integer number  $b(v)$  of *buffers* for each vertex  $v \in V$ , a set  $R$  of *packet routes*, each  $r \in R$  being a simple path in  $G$  from a source vertex  $s(r)$  to a destination vertex  $d(r)$ , and for each ordered pair  $(u, v)$  of vertices that are the source and destination of some packet route  $r \in R$ , a *window size*  $w(u, v)$ .

QUESTION: Is there a possible “deadlock state” for  $G$  under the “individual end-to-end window” flow control discipline? This says that for every route  $r$ , a new packet can be started out along  $r$  only if the network already contains fewer than  $w(s(r), d(r))$  packets with source  $s(r)$  and destination  $d(r)$ . A “deadlock state” consists of a collection  $P$  of packets, each packet  $p$  with an assigned route  $r(p) \in R$  and a current location in a buffer at one of the vertices of  $r(p)$ . The assignments must be such that no buffer contains more than one packet, all the window size bounds are obeyed, and no legal move of the network is possible. A “legal move” can be one of the following: (i) If there is a route  $r$  such that fewer than  $w(s(r), d(r))$  packets are currently in transit from  $s(r)$  to  $d(r)$ , and if  $s(r)$  has an empty buffer, then a new packet  $p$  can be generated and assigned route  $r$  and an empty buffer at  $s(r)$ , (ii) If a packet  $p$  has reached the buffer of its destination vertex  $d(r(p))$ , it can be consumed and removed from that buffer, and (iii) A packet  $p$  that has not yet reached its destination can be transferred from its current buffer to a buffer of the next vertex  $v$  along the path  $r(p)$ , provided that  $v$  has an empty buffer.

*Reference.* Toueg and Steiglitz [29]. Transformation from 3SAT.

*Comment.* Remains NP-complete even if all window sizes are the same and all buffer sizes are the same. Solvable in polynomial time if all window sizes are infinite (“unrestricted” flow control), and in this case even if there is an overall bound imposed on the number of packets allowed in the system at one time (“isarithmic” flow control). Similar results hold if we also require that the deadlock state be “reachable” from an initially empty network, although now the isarithmic case is open and the windowed case is not known to be in NP (reaching a state might require an exponential number of packet generations). The problem of determining, given  $G$  and a set of source-destination pairs, whether a set of routes exists that will make deadlock impossible is NP-hard in

all the above cases [29]. For results about flow control methods that resist deadlock and, unlike the windowing and isarithmic methods above, use only local information, see [4,28].

So far we have considered problems about the *use* of communication networks. Our final two problems concern the *testing* of such networks, first the transmission links (the edges) and then the transmitters and receivers (the vertices).

#### [4] COMMUNICATION LINK TESTING

INSTANCE: Graph  $G = (V, E)$ , positive integer  $K$ .

QUESTION: Can all the edges of  $G$  be tested in both directions in  $K$  or fewer phases? An edge  $\{u, v\}$  is tested in direction  $(u, v)$  by assigning  $u$  to be a transmitter and  $v$  to be a receiver and attempting to transmit a signal from  $u$  to  $v$ . During a “phase,” no vertex can be both a receiver and a transmitter, and no receiver can receive along more than one edge.

*Reference.* Even, Goldreich, Moran, and Tong [11]. Transformation from GRAPH 4-COLORABILITY.

*Comment.* Remains NP-complete even if  $G$  is bipartite, although it becomes polynomial time solvable in that case if no transmitter is allowed to transmit along more than one edge during a phase. This latter case returns to NP-completeness if we add the constraint (imposed in practice by interference problems) that no vertex that is receiving during a phase can have links to two vertices that are transmitting, even if those vertices are transmitting to different destinations. The problem of determining the maximum number of edges that can be tested in a phase is also NP-complete in many situations. For a breakdown on both problems, see [11]. As with FILE TRANSFER SCHEDULING (which overlaps this problem slightly), the complexity classification is complete except for one subcase.

#### [5] COMMUNICATION NODE TESTING

INSTANCE: Directed graph  $G = (V, A)$ , a “test result”  $r(a) \in \{0, 1\}$  for each arc  $a \in A$ , and a positive integer  $t$ . Here the arc  $(u, v)$  represents a test of the head vertex  $v$  by the tail vertex  $u$ . If  $u$  is faulty, the result can be either 0 or 1, but if  $u$  is fault-free, then the result is 0 if and only if  $v$  is also fault-free.

QUESTION: Could these test results be consistent with the hypothesis that  $t$  or fewer of the vertices are faulty?

*Reference.* Maheshwari and Hakimi [20]. Transformation from

## INDEPENDENT SET.

*Comment.* If no set of test results for  $G$  is consistent with more than one set of  $t$  or fewer faulty vertices, we say that  $G$  is “ $t$ -diagnosable.” A polynomial-time algorithm for testing  $t$ -diagnosability has recently been discovered, using network flow techniques [24]. When  $G$  is not  $t$ -diagnosable, we may still be able to proceed (under the assumption of  $t$  or fewer faults) if there is some vertex  $v$  that is present in every consistent set of  $t$  or fewer faulty vertices. Such a vertex must be faulty under the assumption of  $t$  or fewer faults, so we can simply repair it and then repeat the tests. Unfortunately, telling whether a given vertex has this property for a given set of test results is itself NP-hard (unless  $t = |V|$ , in which case it is polynomial-time solvable) [13].

## 5. RANDOMIZED ERRATA

My previous column dealt with algorithms that obtained faster running times by tolerating a small probability of error. In my rush to meet the deadline for that column, I appear to have used a similar approach, and unfortunately that “small probability of error” caught up with me.

In particular, the “Open Problem of the Month,” ENCODING BY TM, has not been shown to be  $R$ -complete for NP; it has been shown to be “ $PR$ -complete.” It turns out that there are a few more types of “randomized” reductions than I had bargained for. After defining  $R$ -reduction (“ $R$ ” for “random”) in [2], Adleman and Manders defined a *second* kind of randomized reduction in [3], which they called a  $UR$ -reduction, where “ $UR$ ” stands for “unfaithfully random.” Vazirani and Vazirani then altered the definition slightly to make it more general, called the result a  $PR$ -reduction (“ $PR$ ” for “probabilistic”), and used this type of reduction in their paper on ENCODING BY TM [31]. As with  $\gamma$ -reducibility and  $R$ -reducibility,  $PR$ -reducibility is defined in terms of a polynomial time nondeterministic Turing machine  $M$  with outputs. To  $PR$ -reduce a problem  $A$  to a problem  $B$ ,  $M$  must behave as follows:

- (1) If the input  $a$  is a yes-instance of  $A$ , then *all* outputs  $b$  that  $M$  can produce must be yes-instances of  $B$ .
- (2) There is a fixed constant  $c > 0$  such that for all inputs  $a$  that are no-instances of  $A$ , at least a fraction  $c$  of the computation paths of  $M$  must yield outputs that are no-instances of  $B$ . (In the original definition of “ $UR$ ”-reducibility,  $c$  was  $1/2$ .)

$PR$ -reducibility is “unfaithful” because for some instances (namely no-instances), it is possible to produce an output  $b$  yielding an answer *different* from that yielded by the input  $a$ . Surprisingly,  $PR$ -completeness provides the same basic guarantee of intractability as does  $R$ -completeness: If a problem  $B$  is  $PR$ -complete for NP, then  $B$  can be solved in polynomial time only if  $NP = ZPP$ .

As with  $R$ -completeness, the only applications of  $PR$ -completeness (before the results of the Vazirani brothers) were in the number-theoretic domain, where on occasion it had been more effective for proving “intractability” than NP-completeness. (The problem of determining whether the equation  $x^2 - ay^2 = c$  has integer solutions, given  $a$  and  $c$ , is shown in [3] to be  $PR$ -complete for NP, whereas this problem is not known to be NP-complete and is not even known to be  $R$ -complete unless the extended Riemann hypothesis holds.) ENCODING BY TM provides a first non-number-theoretic problem where  $PR$ -completeness, but not NP-completeness, is known to hold. Non-number-theoretic examples for  $R$ -completeness (and for  $\gamma$ -completeness) are not yet known.

#### REFERENCES

1. H. ABELSON, Lower bounds on information transfer in distributed computations, *J. Assoc. Comput. Mach.* **27** (1980), 384-392.
2. L. M. ADLEMAN AND K. MANDERS, Reducibility, randomness and intractability, in “Proceedings 9th Ann. ACM Symp. on Theory of Computing,” pp. 151-163, Association for Computing Machinery, New York, 1977.
3. L. M. ADLEMAN AND K. MANDERS, Reductions that lie, in “Proceedings 20th Ann. Symp. on Foundations of Computer Science,” pp. 397-410, IEEE Computer Society, Los Angeles, 1979.
4. J. BLAZEWICZ, D. P. BOVET, AND G. GAMBOSI, Deadlock-resistant flow control procedures for store-and-forward networks, *IEEE Trans. Communication* **COM-32** (1984), 884-887.
5. E. G. COFFMAN, JR, M. R. GAREY, D. S. JOHNSON, AND A. S. LAPAUGH, Scheduling file transfers in a distributed network, in “Proceedings 2nd ACM SIGACT-SIGOPS Symp. on Principles of Distributed Computing,” pp. 254-266, Association for Computing Machinery, New York, 1983.
6. J. E. CUNY AND L. SNYDER, Testing the coordination predicate, *IEEE Trans. Computers* **C-33** (1984), 201-208.
7. R. A. DEMILLO AND R. E. MILLER, Implicit computation of synchronization primitives, *Inform. Process. Lett.* **9** (1979), 35-38.
8. J. DETREVILLE, On finding deadlocks in protocols, manuscript (1982).
9. E. W. DIJKSTRA, Cooperating sequential processes, in “Programming Languages,” pp. 43-112, F. Genuys (Ed.), Academic Press, 1968.
10. P. DURIS, Z. GALIL, AND G. SCHNITGER, Lower bounds on communication complexity, in “Proceedings 16th Ann. ACM Symp. on Theory of Computing,” pp. 81-91, Association for Computing Machinery, New York, 1984.
11. S. EVEN, O. GOLDRICH, S. MORAN, AND P. TONG, On the NP-completeness of certain network testing problems, *Networks* **14** (1984), 1-24.
12. G. N. FREDERICKSON AND N. A. LYNCH, The impact of synchronous communication on the problem of electing a leader in a ring, in “Proceedings 16th Ann. ACM Symp. on Theory of Computing,” pp. 493-503, Association for Computing Machinery, New York, 1984.
13. H. FUJIWARA AND K. KINOSHITA, On the computational complexity of system diagnosis, *IEEE Trans. Computers* **C-27** (1978), 881-885.
14. A. ITAI AND M. RODEH, Symmetry breaking in distributed networks, in “Proceedings 22nd Ann. Symp. on Foundations of Computer Science,” pp. 150-158, IEEE Computer Society, Los Angeles, 1981.
15. D. S. JOHNSON AND C. L. MONMA, Scheduling with simultaneous resource requirements, manuscript (1984).

16. R. E. LADNER, The complexity of problems in systems of communicating sequential processes, *J. Comput. System Sci.* **21** (1980), 179-194.
17. N. LAKSHMIPATHY AND K. WINKLMANN, "'Global' graph problems tend to be intractable," Report No. TR84-7, Department of Computing Science, University of Alberta, Edmonton, Alberta, Canada, 1984.
18. R. J. LIPTON AND R. SEDGEWICK, Lower bounds for VLSI, in "Proceedings 13th Ann. ACM Symp. on Theory of Computing," pp. 300-307, Association for Computing Machinery, New York, 1981.
19. K. MEHLHORN AND E. M. SCHMIDT, Las Vegas is better than determinism in VLSI and distributed computing, in "Proceedings 14th Ann. ACM Symp. on Theory of Computing," pp. 330-337, Association for Computing Machinery, New York, 1982.
20. S. N. MAHESHWARI AND S. L. HAKIMI, On models for diagnosable systems and probabilistic fault diagnosis, *IEEE Trans. Computers* **C-25** (1976), 228-236.
21. C. H. PAPADIMITRIOU AND M. SIPSER, Communication complexity, *J. Comput. System Sci.* **28** (1984), 260-269.
22. C. H. PAPADIMITRIOU AND J. TSITSIKLIS, On the complexity of designing distributed protocols, *Information and Control* **53** (1982), 211-218.
23. R. PATURI AND J. SIMON, Probabilistic communication complexity, in "Proceedings 25th Ann. Symp. on Foundations of Computer Science," IEEE Computer Society, Los Angeles, 1984.
24. G. SULLIVAN, A polynomial time algorithm for fault diagnosability, in "Proceedings 25th Ann. Symp. on Foundations of Computer Science," IEEE Computer Society, Los Angeles, 1984.
25. R. N. TAYLOR, Complexity of analyzing the synchronization structure of concurrent programs, *Acta Inform.* **19** (1983), 57-84.
26. C. D. THOMPSON, Area-time complexity for VLSI, in "Proceedings 11th Ann. ACM Symp. on Theory of Computing," pp. 81-88, Association for Computing Machinery, New York, 1979.
27. C. D. THOMPSON, "A Complexity Theory for VLSI," Ph.D. Thesis, Computer Science Department, Carnegie-Mellon University, Pittsburgh, Penn., 1980.
28. S. TOUEG AND K. STEIGLITZ, Deadlock-free packet switching networks, in "Proceedings 11th Ann. ACM Symp. on Theory of Computing," pp. 89-97, Association for Computing Machinery, New York, 1979.
29. S. TOUEG AND K. STEIGLITZ, Some complexity results in the design of deadlock-free packet switching networks, *SIAM J. Comput.* **10** (1981), 702-712.
30. J. D. ULLMAN, "Computational Aspects of VLSI," Computer Science Press, Rockville, Maryland, 1984.
31. U. V. VAZIRANI AND V. V. VAZIRANI, A natural encoding scheme proved probabilistic polynomial complete, *Theor. Comput. Sci.* **24** (1983), 291-300.
32. U. VISHKIN AND A. WIGDERSON, Trade-offs between depth and width in parallel computation, in "Proceedings 24th Ann. Symp. on Foundations of Computer Science," pp. 146-153, IEEE Computer Society, Los Angeles, 1983.
33. J. VUILLEMIN, A combinatorial limit to the computing power of V.L.S.I. circuits, in "Proceedings 21st Ann. Symp. on Foundations of Computer Science," pp. 294-300, IEEE Computer Society, Los Angeles, 1980.
34. A. C.-C. YAO, Some complexity questions related to distributive computing, in "Proceedings 11th Ann. ACM Symp. on Theory of Computing," pp. 209-213, Association for Computing Machinery, New York, 1979.
35. A. C.-C. YAO, The entropic limitations on VLSI computations, in "Proceedings 13th Ann. ACM Symp. on Theory of Computing," pp. 308-311, Association for Computing Machinery, New York, 1981.