Pergamon

Computers Math. Applic. Vol. 29, No. 4, pp. 93–99, 1995

0898-1221/95 $9.50 + 0.00

0898-1221(94)00242-8

Reliability Optimization of Distributed Computing Systems Subject to Capacity Constraints

RUEY-SHUN CHEN,* DENG-JYI CHEN AND Y. S. YEH
Institute of Computer Science and Information Engineering
National Chiao Tung University, Hsinchu, Taiwan, R.O.C.
Rschen@cc.nctu.edu.tw

(Received and accepted August 1993)

Abstract—In this paper, we propose a simple, easily programmed exact method for obtaining the optimal design of a distributed computing system in terms of maximizing reliability subject to memory capacity constraints. We assume that a given amount of resources are available for linking the distributed computing system. The method is based on the partial order relation. To speed up the procedure, some rules are proposed to indicate conditions under which certain vectors in the numerical ordering that do not satisfy the capacity constraints can be skipped over. Simulation results show that the proposed algorithm requires less time and space than exhaustive method.

Keywords—Distributed computer system, Reliability, Exact approach, Capacity constraint.

1. INTRODUCTION

A distributed computer system (DCS) has been defined as a collection of nodes at which reside computing resources that communicate with each other via a set of links [1]. Large scale distributed computer systems are coming into use primarily because of the economy achieved through resource sharing [2]. The main objective of a DCS is to provide efficient communication among various nodes in order to increase their utility and to make their service available to more users [3]. One of the fundamental considerations in designing such systems is that of system reliability, which strongly depends on the topological layout of the communication links [4]. Reliability is a very good measure of DCS performance if all the needed network users are to be connected with each other, i.e., it is desired to establish a communication path between all the K-available nodes at one time. A DCS may be modelled by a graph in which the nodes correspond to the file servers and the edges to the communication links.

Several heuristic methods [5–7] have been proposed for obtaining an optimal network topology that gives maximum overall reliability of a given computer communication network, but there is no method that provides an exact solution. All of the proposed methods find an approximate solution, because as the number of links increases, the number of possible layouts of the links grows faster than exponentially. However, an exact optimal solution is important where the topology will be used for an extended time. To date, the problem of maximizing the reliability of a DCS under memory constraints through exact methods does not appear to have been considered. In

*Author to whom all correspondence should be addressed.
This research was partially supported by the National Science Council of the Republic of China under contract NSC81-0301-E009-507.

this paper, we propose an exact method for obtaining an optimal DCS using the partial order relation of solving discrete optimization problems [8]. The method is simple, easy to understand, and easy to program. To speed up the procedure, rules indicating conditions under which certain vectors in the numerical ordering do not satisfy the capacity constraints can be skipped over. This reduces the evaluation count and execution time.

The organization of the rest of this paper is as follows. In Section 2, assumptions, notation and definitions that will be used throughout this paper are given. Section 3 presents the mathematical formulation of the problem and an algorithm with a flow chart of the solution procedure. Examples are used to illustrate the method and simulation results are obtained in Section 4. Section 5 concludes the paper.

2. ASSUMPTIONS, NOTATION AND DEFINITIONS

ASSUMPTIONS. The method suggested here makes the following assumptions.

1. The locations of the various computers and the possible locations of the connecting links are known.
2. The reliability of every link is known.
3. The capacity of every node installed is specified.
4. Each node is perfectly reliable.
5. Every link is either in the working (ON) state or failed (OFF) state.
6. The total capacity constraint on the DCS is known.

NOTATION. The notation and definitions used in the rest of paper are summarized here.

$G = (N, L)$	an undirected DCS graph in which the set of nodes N represents the PEs and the links L represent the communication links.
N_i	a node representing a processing element i.
L_i	an edge representing a communication link i.
$L = \{L_1, L_2, \ldots, L_n\}$	the set of all allowable links.
$X^\wedge$	denotes the vector which has current optimal solution $R(G_x)$.
X_i	a decision node, $X_i = 1$ if i is selected, else $X_i = 0$.
$X = \{x_1, x_2, x_3, \ldots, x_n\}$	the set of decision nodes, $X_i = 0$ or 1, $i = 1, 2, \ldots, n$.
C_i	capacity of the i^{th} node.
C_s	memory capacity constraint in system.
G_k	denotes the graph G with K-node specified.
$R^\wedge$	denotes the current optimal reliability solution.
$R(G_x)$	the reliability of X-node solution of the DCS graph G.
X^*	first vector following X in the numerical ordering that has the property $X \not\leq X^*$.
$X < Y$	X is less than Y with numerical ordering.
$X \leq Y$	X is less than Y with vector partial ordering.

DEFINITION 1. *K-node reliability is defined as the probability of successful communication, i.e., all K-nodes in G_x are connected by working edges within the given memory capacity constraint.*

DEFINITION 2. *An K-node DCS reliability problem is the problem of computing $R(G_x)$. The problem is a member of the class of number K-complete problems that is a class of NP-complete problems.*

DEFINITION 3. *An evaluation count is the number of computation of $R(G_x)$ that are needed to satisfy the capacity constraint.*

3. MATHEMATICAL FORMULATION OF THE PROBLEM

The problem considered in this paper may be stated as follows: determine an optimal DCS that gives maximum reliability within the given memory capacity constraint. In other words,

we are to find a set of X-nodes from the given set N which constitutes an optimal DCS in that X-node reliability is maximized and the total memory capacity satisfies the capacity constraint. The main problem can be stated mathematically as follows:

$$\begin{aligned} &\text{Maximize} && R(G_x), \\ &\text{subject to:} && \sum_{Xi \in K} C_i \geq C_s. \end{aligned}$$

In this section, we discuss some preliminaries that are essential for the description of the algorithm. We consider vectors X of the form $X = (x_1, x_2, \ldots, x_n)$ which are binary in the sense that each X_j is either 0 or 1. We say that $X \leq Y$ if and only if $X_j \leq Y_j$, for $j = 1, 2, \ldots, n$, e.g., $X \leq Y$ where $X = (0\,1\,0)$ and $Y = (0\,1\,1)$. This is the vector partial ordering. Note that the numerical ordering is a refinement of the vector partial ordering, i.e., $X \leq Y$ implies $n(x) \leq n(y)$, but $n(x) \leq n(y)$ does not imply $X \leq Y$, where $n(x)$, $n(y)$ are numerical order. Suppose all binary n-vectors are listed in numerical order, i.e.,

$$(0, 0, 0, 0), (0, 0, 0, 1), (0, 0, 1, 0), (0, 0, 1, 1), \ldots, (1, 1, 1, 1), \text{ etc.}$$

Immediately following an arbitrary vector X, there may (or may not) be a number of vectors X' with the property that $X \leq X'$. Roughly speaking, these are vectors that differ from X only in that they have 1's in place of one or more of the rightmost 0's of X, for example, immediately following $X = (0, 1, 0, 0)$, $(0, 1, 0, 1)$, $(0, 1, 1, 0)$, and $(0, 1, 1, 1)$, each of which is greater than X in the vector partial ordering.

We let X^* denote the first vector following X in the numerical ordering that has the property that $X \not\leq X^*$. For any given X, the vector X^* is easily calculated on a computer as follows. Treat X as a binary number:

(1) Subtract 1 from X,
(2) Logically 'or' X and $X - 1$ to obtain $X^* - 1$,
(3) Add 1 to obtain X^*.

An example:

Let $X = 0101100$,
(l) $X - l = 0101011$,
(2) $X^* - 1 = 0101111$,
(3) $X^* = 0110000$.

Note that $X^* - 1$ is greater than each of X, $X + 1, \ldots, X^* - 2$, in the vector partial ordering. Consider the following simple optimal DCS problem:

$$\begin{aligned} &\text{Maximize} && R(G_x), \\ &\text{subject to:} && \sum_{Xi \in K} C_i \geq C_s. \end{aligned}$$

We can solve this problem by examining each of the 2^n possible solution vectors in numerical order, beginning with $X = (0, 0, \ldots, 0)$ and ending with $(1, 1, \ldots, 1)$. However, this process can be shortened considerably by invoking certain rules, which are stated below.

As we proceed through the list of vectors, we keep a record of the least capacity solution found to date. Let $X^\wedge$ denote current optimal solution vector. Let X denote the vector that is currently being examined. The following rules indicate conditions under which certain vectors in the numerical ordering can be skipped over.

RULE 1. *If* $\sum_{Xi \in K} C_i(X^* - 1)_i < C_s$, *then skip to* X^*.

JUSTIFICATION. X is monotone nondecreasing, and $X \leq X+1 \leq X+2 \leq, \ldots, \leq X^*-2 \leq X^*-1$, thus, $Ci(X) \leq Ci(X+1) \leq Ci(X+2) \leq, \ldots, Ci(X^*-2) \leq Ci(X^*-1) \leq Cs$.

RULE 2. *If the vector partial ordering* $X \geq X^\wedge$, *then skip to* X^*.

JUSTIFICATION. X is monotone nondecreasing, and $X^\wedge \leq X \leq X+1 \leq X+2 \leq, \ldots, \leq X^*-2 \leq X^*-1$, thus, $R(Gx^\wedge) \geq R(G_x) \geq R(G_{x+1}) \geq R(G_{x+2}) \geq, \ldots, \geq R(G_{x^*-2}) \geq R(G_{x^*-1})$ are all less than $R(G_{x^\wedge})$.

RULE 3. *If* X *is a feasible solution of* $R(G_x) < R^\wedge$, *then skip to* X^*.

JUSTIFICATION. X is monotone nondecreasing, and $X \leq X+1 \leq X+2 \leq, \ldots, \leq X^*-2 \leq X^*-1$, so $R(G_x) \geq R(G_{x+1}) \geq R(G_{x+2}) \geq, \ldots, \geq R(G_{x^*-2}) \geq R(G_{x^*-1})$. Of course, if Rule 3 applies and $R(G_x) > R^\wedge$, then $X^\wedge$ is substituted for X and $R^\wedge$ is substituted for $R(G_k)$.

To find an optimal solution, we do not consider an exhaustive method, since it is too time-consuming. Instead, we apply the Lawler-Bell algorithm [8] to find an optimal solution by skipping unsatisfactory conditions and executing only a portion of all the combinations. The problem of DCS can then be transformed into the following formulation:

$$\begin{aligned} &\text{Maximize} && R(G_x), \\ &\text{subject to:} && \sum_{Xi \in K} C_i \geq C_s. \end{aligned}$$

The exact solution for our problem using the Lawler-Bell technique is described as follows:

1. Node vector:

$$X = (X_n, X_{n-1}, \ldots, X_1), \qquad X_i = 0 \text{ or } 1, \qquad i = 1, 2, \ldots, n.$$

2. Numerical ordering:

$$(0, 0, 0, 0), (0, 0, 0, 1), (0, 0, 1, 0), (0, 0, 1, 1), \ldots, (1, 1, 1, 1).$$

3. Vector partial ordering:

$$X \leq Y \text{ if and only if } X_i \leq Y_i, \qquad \text{for } i = 1, 2, \ldots, n,$$

e.g., if $X = (0, 1, 0, 0, 1)$, $Y = (0, 1, 1, 0, 1)$, then, $X \leq Y$ implies $R(G_x) \geq R(G_y)$ and $Ci(Xi) \leq Ci(Yi)$.

4. X^*: the first vector following X in the numerical ordering that has the property that $X \not\leq X^*$. The formulation is $X^* = (X \text{ bit or } X-1 \text{ bit}) + 1$, e.g.,

$$X = 0101100,$$
$$X^* = 0110000.$$

If X is monotone nondecreasing in each of the vector partial orderings, then

$$X \leq X+1 \leq X+2 \leq, \ldots, \leq X^*-2 \leq X^*-1 \not\leq X,$$

implying that $R(G_x) \geq R(G_{x+1}) \geq R(G_{x+2}) \geq, \ldots, \geq R(G_{x^*-2}) \geq R(G_{x^*-1}) \not\geq R(G_{x^*})$.

ALGORITHM. The algorithm based on the above rules is given below.

STEP 1. Determine whether the count number has overflowed. If $X >$overflow_X then stop; otherwise, go to the next step.

STEP 2. Compare the capacity $\sum_{i=1}^{n} C_i(x^* - 1)_i$ with capacity constraint Cs. If $\sum_{i=1}^{n} C_i(x^* - 1)_i < Cs$, then substitute X for X^* and go to back to Step 1, else go to the next step.

STEP 3. Compare the capacity $\sum_{i=1}^{n} C_i X_i$ with capacity constraint Cs. If $\sum_{i=1}^{n} C_i X_i < Cs$, then $X = X + 1$ and go to back to Step 1, else go to the next step.

STEP 4. Compare the vector partial ordering X with $X^\wedge$. If $X \geq X^\wedge$, then X is substituted for X^* and go to back to Step 1, else go to the next step.

STEP 5. Compute reliability $R(G_x)$ and compare with the current optimal solution $R^\wedge$. If $R(G_x) > R^\wedge$, then substitute $R^\wedge$ for $R(G_x)$ and $X^\wedge$ for X, else substitute X for X^* and go to back to Step 1.

STEP 6. Continue loop until X overflow. Then, last $R(G_{x^\wedge})$ reliability computed with the factoring algorithm [9] is obtained for our X-node reliability and $X^\wedge$ is obtained for our optimal node vector.

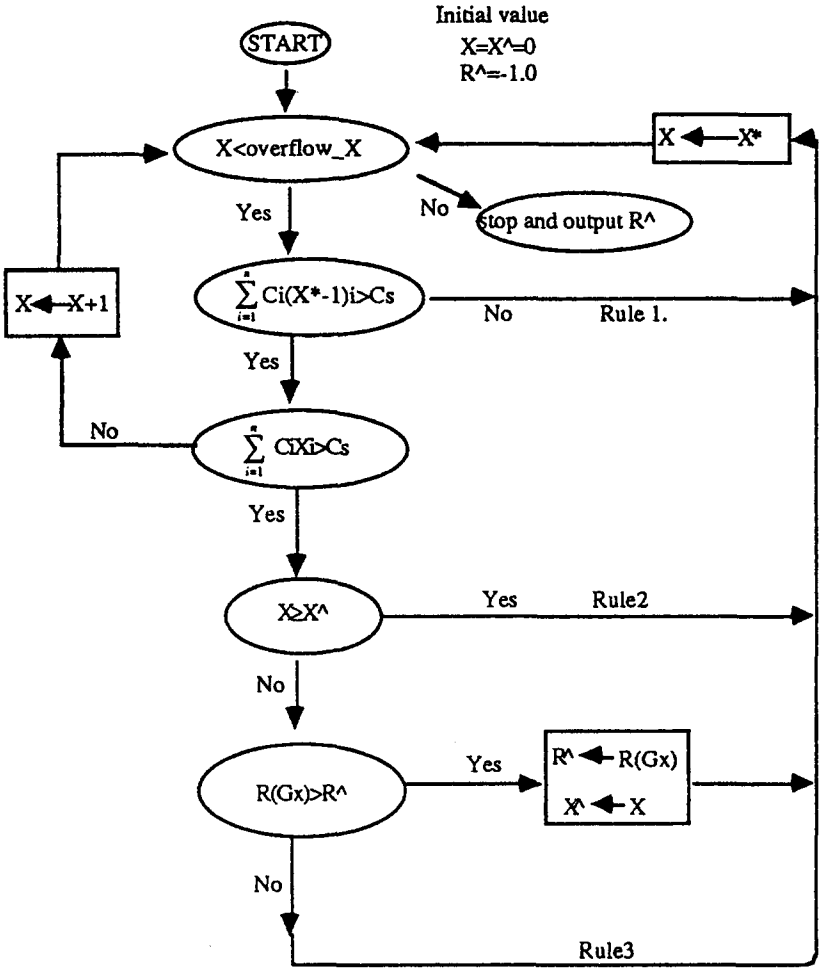

Figure 1. Flow chart of the actual solution procedure.

The actual solution algorithm followed is a variation of that given by Lawler and Bell [8], modified for the sake of computational simplicity. The algorithm is illustrated with the flow chart in Figure 1. We determine the X-node reliability of a DCS iff the corresponding node capacity satisfies the given capacity constraints.

4. EXAMPLES AND RESULTS

The exact method is illustrated below by means of two examples.

4.1. Example 1

Consider the six node DCS with eight links depicted in Figure 2. Here, our problem is to determine an optimal DCS which includes some of the nodes $X_1, X_2, \ldots, X_6$, whose total capacity exceeds the memory capacity constraint of 70 units. The optimization can be formulated as the following mathematical problem:

$$\begin{aligned} &\text{Maximize} && R(G_z), \\ &\text{subject to:} && \sum_{Xi \in K} C_i \geq 70. \end{aligned}$$

Using a C computer program based on the proposed algorithm and the flow chart in Figure 1, an optimal DCS was obtained by an Intel-486 personal computer in 2 clock cycle execution time. The optimal K-DCS topology with node vector (X_1, X_2, X_3, X_5)was found in the DCS with maximum reliability of 0.7628 and memory capacity of 70 units. The evaluation count was only 17, compared with a count of 32 for the exhaustive method. This is a rather modest reduction in computing, but much greater saving will be made in problems with a larger DCS.

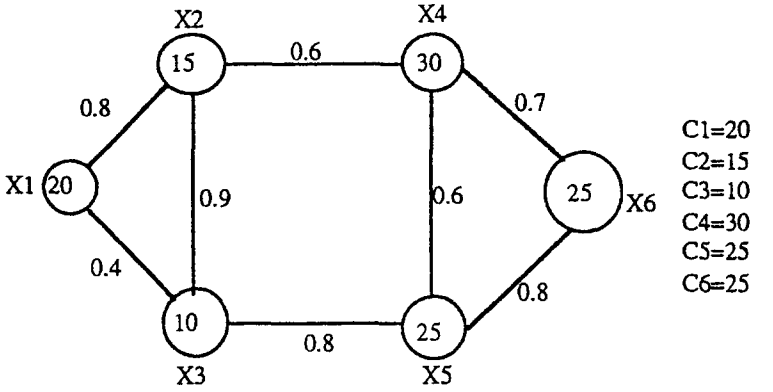

Figure 2. A six node distributed computing system.

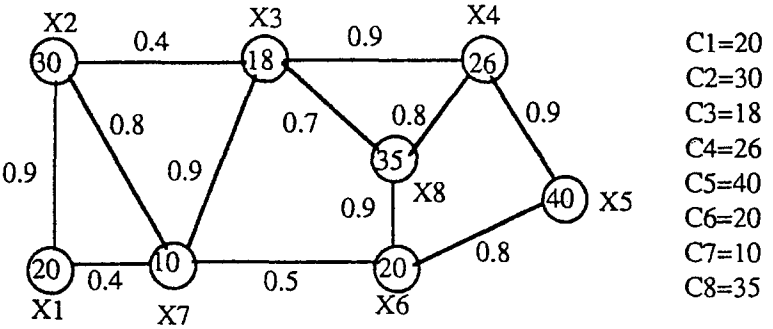

Figure 3. An ARPA-net distributed computing system.

4.2. Example 2

Consider a simplified version of the well-known ARPA network having eight nodes and 12 links, as depicted in Figure 3. Here, our problem is to determine an optimal DCS which includes some of the nodes X_1, $X_2, \ldots, X_8$, the total capacity of which exceeds the memory capacity constraint of 100 units. The problem can be stated mathematically as follows:

$$\begin{aligned} &\text{Maximize} && R(G_z), \\ &\text{subject to:} && \sum_{Xi \in K} C_i \geq 100. \end{aligned}$$

Using a C computer program based on the proposed algorithm and the flow chart in Figure 1, an optimal DCS was obtained by an Intel-486 personal computer in 54 clock cycle execution time. The optimal K-DCS with node vector (X_1, X_2, X_3, X_4, X_7) was found in the DCS with maximum reliability of 0.8245 and memory capacity of 104 units. The evaluation count was only 78, as compared with a count of 128 for the exhaustive method. Although 78 evaluation counts need to be examined, only a small portion of the 78 need to use factoring algorithm [9] to compute reliability $R(G_x)$. Therefore, rules can reduce execution time effectively.

5. CONCLUSION

The present method for the determining of the optimal X-node in a DCS so as to maximize the K-node reliability is based on the partial order relation. However, the proposed algorithm requires less execution time and space than the exhaustive method, because we apply several rules to discard most of the infeasible evaluation count before computing the reliability $R(G_x)$ and reduce the computation time. Thus, the rules are very effective. The algorithm presented is straightforward and easily transformed into a program.

REFERENCES

1. J.A. Stankovic, A perspective on distributed computer systems, *IEEE Trans. Comput.* **33**, 1102–1115 (1984).
2. D.J. Chen and T.H. Huang, Reliability analysis of distributed systems based on a fast reliability algorithm, *IEEE Trans. on Parallel and Distributed Systems* **3** (2) (1992).
3. L. Fratta and U.G. Montanari, Synthesis of available networks, *IEEE Trans. Reliab.* **R-25**, 81–87 (1976).
4. R.S. Wilkov, Design of computer networks based on a reliability measure, In *Proceedings of the Symposium on Computer Communication, Networks and Teletraffics*, pp. 371–384, (1986).
5. K.K. Aggarwal, Y.C. Chopra and J.S. Bajwa, Topological layout of links for optimizing the S-T reliability in a computer communication system, *Microelectron. Reliab.* **22**, 341–345 (1982).
6. K.K. Aggarwal, Y.C. Chopra and J.S. Bajwa, Topological layout or links for optimizing the overall reliability in a computer communication system, *Microelectron. Reliab.* **22**, 347–351 (1982).
7. Y.C. Chpra, B.S. Sohi and K.K. Aggarwal, Network topological for maximizing the terminal 1 reliability in a computer communication system, *Microelectron. Reliab.* **24** (5), 911–913 (1984).
8. E.L. Lawler and M.D. Bell, A method for solving discrete optimization problems, *Ops. Res.* **14**, 1098–1111 (1966).
9. K.R. Wood, Factoring algorithms for computing K-terminal network reliability, *IEEE Trans. Reliability* **35**, 269–278 (1989).