Proceeding Series of the Brazilian Society of Applied and Computational Mathematics, Vol. 3, N. 2, 2015.
Trabalho apresentado no III CMAC - SE, Vitória-ES, 2015.

Proceeding Series of the Brazilian Society of Computational and Applied Mathematics

A Simple Novel Algorithm for a Special Diophantine System Appeared in the System Reliability Problem

Majid Forghani-elahabad[1]
Center for Engineering, Modeling and Applied Social Sciences (CECS), UFABC, Santo Andre, SP
Luiz Henrique Bonani[2]
Center for Engineering, Modeling and Applied Social Sciences (CECS), UFABC, Santo Andre, SP

Abstract. System reliability of a stochastic-flow network (SFN) can be computed in terms of either upper boundary points for demand level d called d-minimal cuts (d-MCs) or lower boundary points for demand level d called d-minimal paths (d-MPs). A number of different algorithms have been proposed in the literature to search for and determine all the d-MCs or d-MPs in an SFN. Majority of those algorithms need to solve a special Diophantine system. Here, we consider the Diophantine system appeared in d-MC and d-MP problems and propose a novel efficient algorithm to solve it. The proposed algorithm finds all the system solutions in a increasing lexicographic order. We illustrate the algorithm through an example and show its time complexity to be linear according to the number of its solutions.

Keywords. Diophantine system, System reliability, Algorithm, d-MP, d-MC

1 Introduction

Network reliability theory has extensively been applied to a variety of real-world systems such as transportation, power transmission and distribution, computer and communication, and etc. [3]- [10]. System reliability, the probability that the maximum flow of the network is more than or equal to a demand level d, can be computed in terms of either upper boundary points for demand level d, called d-minimal cuts (d-MCs) [6] or lower boundary points for demand level d, called d-minimal paths (d-MPs) [5, 8, 9]. Once all the d-MCs or d-MPs are determined, the system reliability can be computed by some methods such as sum of disjoint product [3]. Jane et al. [7] proposed an algorithm to first obtain all the d-MC candidates and then specify the d-MCs. Presenting new results as well as a new data structure, Forghani and Mahdavi-Amiri [6] proposed an efficient algorithm and showed the algorithm to be more efficient than the one given in [7]. Lin et al. [8] introduced the notion of d-MP candidate and proposed an algorithm that uses a comparative method to solve d-MP problem. Evaluating the system reliability is an NP-hard problem [4] and the problem continues to be interesting to investigate. To our

[1]m.forghani@ufabc.edu.br
[2]luiz.bonani@ufabc.edu.br

knowledge, majority of the proposed algorithms in the literature [5]- [8] need to solve a special Diophantine system whereas the authors have not payed attention to it. Although several algorithms are available to solve a Diophantine system [2], the desired system is a special case and can be efficiently solved via a particular approach. Here, we consider this special system and present a novel efficient linear time algorithm to solve it. We also show that the proposed algorithm finds all the system solutions in a increasing lexicographic order that can be helpful when it employed in *d-MC* or *d-MP* problem. In the sequence, Section 2 describes the required notations, nomenclature and assumptions, and states briefly the *d-MC* and *d-MP* problems. In Section 3, a simple novel efficient algorithm is proposed to find all the solutions of a special Diophantine system appeared in *d-MC* and *d-MP* problems. Then, the proposed algorithm is shown to be correct.

2 Model Building

Here, we first state the required definitions and assumptions. Then, we explain briefly the *d-MC* and *d-MP* problems to meet the desired Diophantine system (for more details on the *d-MC* and *d-MP* problems, see [5]- [9]).

Let $G = G(N, A, M)$ be a stochastic-flow network (SFN), where $N = \{1, 2, \ldots, n\}$ is the set of nodes (so, n is the number of nodes), $A = \{a_i|\ 1 \leq i \leq m\}$ is the set of arcs (so, m is the number of arcs), and $M = (M_1, M_2, \ldots,\ M_m)$ is a vector with M_i denoting the maximum capacity of arc a_i, for $i = 1, 2, \ldots, m$. In network G, node 1 and n are respectively considered as the source and the sink nodes. Denote the current capacity of a_i by x_i, and let $X = (x_1, x_2, \ldots, x_m)$ be a system-state vector representing the current capacity of all the arcs. Let $Z(X) = \{a_i \in A|x_i > 0\}$, $U(X) = \{a_i|x_i < M_i\}$, and $V(X)$ be respectively the set of nonzero-capacity arcs, the set of unsaturated arcs, and the maximum flow of the network from node 1 to node n, under system vector X. Let also $\nu = V(M)$ be the maximum flow of the network from node 1 to node n, and $e_i = 0(a_i)$ be a system-state vector in which the capacity level is 1 for a_i and 0 for other arcs. A system state $X = (x_1, x_2, \ldots, x_m)$ is less than or equal to system state $Y = (y_1, y_2, \ldots, y_m)$, i.e., $X \leq Y$, when $x_i \leq y_i$, for $i = 1, 2, \ldots, m$, and also $X < Y$, when $X \leq Y$ and there exists at least one j, $j = 1, 2, \ldots, m$, such that $x_j < y_j$. A path is a sequence of adjacent arcs from the source node 1 to the sink node n. A minimal path (MP) is a path with no cycle. A cut is a subset of A in which there is no path from the source node to the sink node after the elimination of all its arcs from G. A minimal cut (MC) is a cut so that none of its proper subsets is a cut. Furthermore, let h and f be respectively the number of MPs and MCs of a network, $P_1,\ P_2,\ \ldots,\ P_h$ be all the MPs, and $K_1,\ K_2,\ \ldots,\ K_f$ be all the MCs of the network. Let also $CP_j = \min\{M_i|a_i \in P_j\}$ be the capacity of MP, P_j, for $j = 1, 2, \ldots, h$, and $C_{K_i}(X) = \Sigma_{a_k \in K_i} x_k$ be the capacity of MC, K_i, under system state $X = (x_1, x_2, ..., x_m)$, for $i = 1, 2, \ldots, f$. Finally, let R_d be the system reliability of an SFN for demand level d, the probability of transmitting at least a given demand amount, d units, of data (flow) from source node 1 to sink node n.

Here, we consider the assumptions given below.

1. The capacity of each arc is stochastic with a given probability distribution.

2. Capacity of each arc is statistically independent from the one for any other arc.
3. Flow in G satisfies the flow conservation law [1].
4. Each node is perfectly reliable; i.e., deterministic.
5. d is a non-negative integer-valued flow less than or equal to ν.

2.1 *d-MC* problem

An upper boundary point, called d-minimal cuts (*d-MC*), is a system-state vector, say X, such that $V(X) = d$, and all its unsaturated arcs are sensitive to increase in capacity, that is, for every $Y > X$, we have $V(Y) > d$. To find all the *d-MC*s, Jane et al. [7] introduced the notion of a *d-MC* candidate as given below.

Definition 2.1. *Assuming that K_i is an MC, every integer vector $X = (x_1, x_2, ..., x_m)$ satisfying the following system is a d-MC candidate obtained from MC, K_i:*

$$\begin{array}{lll} (i)\ C_{K_i}(X) = d, & & \\ (ii)\ 0 \leq x_k \leq M_k, & \forall\ a_k \in K_i, \\ (iii)\ x_l = M_l, & \forall\ a_l \notin K_i. \end{array} \tag{1}$$

Therefore, in order to determine all the *d-MC*s in an SFN, one can first determine all the *d-MC* candidates, and then search for the *d-MC*s among them.

Assuming $X^1, X^2, ..., X^q$ as all the *d-MC*s and $X = (x_1, x_2, ..., x_m)$ as a system-state vector, the system reliability for level $d + 1$ using the sum of disjoint products method [3] is given by

$$R_{d+1} = pr(A) = \sum_{i=1}^{q} pr(E_i), \tag{2}$$

where, $A = \cup_{i=1}^{q} A_i = \cup_{i=1}^{q} E_i$, $A_i = \{X | X > X^i\}$, $E_1 = A_1$, $E_i = A_i - \cup_{j=1}^{i-1} A_j$, $i = 2, 3, ..., q$, $pr(E_i) = \sum_{X \in E_i} pr(X)$, and $pr(X) = \prod_{j=1}^{m} pr(x_j)$.

Thus, to compute the system reliability in terms of *d-MC*s, the first step is finding all the *d-MC* candidates. It means, we should solve the given Diophantine system in Definition 2.1, System (1), according to MC, K_i, for $i = 1, 2, \dots, f$. This shows the importance of solving the system.

2.2 *d-MP* problem

Definition 2.2. *A system state vector $X = (x_1, x_2, \dots, x_m)$ is a lower boundary point for demand level d (or d-MP) when it satisfies the followings:*

$$\begin{array}{ll} (i)\ V(X) = d, & \\ (ii)\ V(X - e_i) < d, & \text{for each } i \text{ that } a_i \in Z(X). \end{array} \tag{3}$$

Definition 2.3. *A vector $F = (f_1, f_2, \dots, f_h)$ is named a feasible flow vector when it satisfies the followings:*

$$\begin{array}{ll} (i)\ f_1 + f_2 + \cdots + f_h = d, & \\ (ii)\ 0 \leq f_j \leq CP_j, & j = 1, 2, \dots, h, \\ (iii)\ \sum_{a_i \in P_j} f_j \leq M_i, & i = 1, 2, \dots, m, \end{array} \tag{4}$$

where h is the number of MPs, CP_j is the capacity of MP, P_j, for $j = 1, 2, \dots, h$, and M_i is the maximal capacity of arc a_i, for $i = 1, 2, \dots, m$.

Note that in a feasible flow vector (FFV) $F = (f_1, f_2, \dots, f_h)$, each component, say f_j, for $j = 1, \dots, h$, denotes the amount of flow passing through MP, P_j. Hence, for each FFV F, we can obtain a system state vector $X_F = (x_1, x_2, \dots, x_m)$ using the following equation:

$$x_i = \sum_{a_i \in P_j} f_j, \qquad \forall i = 1, 2, \dots, m. \tag{5}$$

Lemma 2.1. *[5] Assuming $X_F = (x_1, x_2, \dots, x_m)$ as the obtained system state vector from a feasible solution vector $F = (f_1, f_2, \dots, f_h)$ employing Eq. (5), we have $V(X_F) = d$.*

For each feasible flow vector $F = (f_1, f_2, \dots, f_h)$, the system state vector $X_F = (x_1, x_2, \dots, x_m)$ obtained by Eq. (5) is named a *d-MP* candidate. Lin et al. [8] showed that each *d-MP* is a *d-MP* candidate. Hence, to find all the *d-MP*s, one can first determine all the candidates, and then check each candidate to be a *d-MP* according to Definition 2.2.

Assuming $X^1, X^2, \dots, X^\lambda$ as all the *d-MP*s, let $E_r = \{X | X \geq X^r\}$, for $r = 1, 2, \dots, \lambda$, $H_1 = E_1$, $H_i = E_i - \cup_{r=1}^{i-1} E_r$, $i = 2, 3, ..., \lambda$. In this case, according to sum of disjoint product technique [3], we have

$$R_d = \sum_{r=1}^{\lambda} \Pr(H_r), \tag{6}$$

where $\Pr(H_r) = \sum_{X \in H_r} \Pr(X)$, and $\Pr(X) = \prod_{j=1}^{m} \Pr(x_j)$.

As seen, the first step in evaluating the system reliability in terms of *d-MP*s, is also determination of all the candidates, and this shows the importance of solving System (4).

3 The proposed algorithm

Here, we propose a new simple efficient algorithm to find all the integer solutions of the systems (1) and (4). As seen, the last equation in system (1) is to set some variables, and thus its computing cost can be disregarded. Also, we observe that the first equations in the systems (1) and (4) state that the sum of some corresponding variables must equal d. Moreover, one can first find all the solutions of (i) and (ii) in System (4), and then check them for the third inequality, (iii). Consequently, here, we consider Diophantine system (7) given below because it is the common part in both desired systems (1) and (4).

$$\begin{array}{ll} (i)\ x_1 + x_2 + \ldots + x_h = d, & \\ (ii)\ 0 \leq x_i \leq M_i, & i = 1, 2, \dots, h. \end{array} \tag{7}$$

We can now discuss our algorithm for solving (7). Our proposed algorithm finds solutions of the system as h-tuple vectors, $(x_1, \dots, x_h)$. In the beginning (Step 0), every component of the vector is set to 0. The algorithm first finds the first solution of the system in lexicographical order. Then, in each iteration, it obtains the next solution in lexicographical order using the previous one. The procedure consists of two general stages (steps 1 and 2). In the first stage (Step 1), it distributes the given amount d among the components of solution vector from the right end of the vector to the front. In the second stage (Step 2), the algorithm starts from the right end of the current solution vector searching for the first component whose value can be increased by one unit. This component is named the pivot component. The algorithm sets all the components at the right side of the pivot to zero, increasing d by the sum of the removed values. Then, the pivot value is increased by one unit, reducing d by one unit. Next, transfer is made to the first stage to distribute the current value of d to determine the next solution vector. We are now in a position to state the algorithm.

Algorithm 1. Finding all the integer solutions of system (7).

```
Step 0 Let $Q = \phi,\ r = 1$, and $x_i = 0$, for $i = 1, 2, \ldots, h$.
Step 1 Distribute the given amount of $d$.
1.1. Let $i = h$.
1.2. Let $x_i = \min\{M_i, d\},\ d = d - x_i$, and $i = i - 1$.
1.3. If $d > 0$ then go to Step 1.2, else let $X^r = (x_1, x_2, \ldots, x_h),\ Q = Q \cup \{X^r\},\ r = r + 1$,
and go to Step 2.
Step 2 Find the pivot component of solution $X^r$ to find next solution as follows:
2.1. Let $i = h$.
2.2. If $i = 1$ then stop ($Q$ is the set of all integer solutions of system (7) sorted in lexico-
graphical order).
2.3. Let $d = d + x_i,\ x_i = 0$, and $i = i - 1$.
2.4. If $x_i = M_i$ or $d = 0$ then go to Step 2.2
else $x_i$ is a pivot: let $x_i = x_i + 1$, $d = d - 1$ and go to Step 1.
```

To expose Algorithm 1, we use an example. Consider $x_1 + x_2 + x_3 = 7$ with $x_1 \leq 2$, $x_2 \leq 4$, and $x_3 \leq 5$. Next, we find its all integer solutions using Algorithm 1.

Step 0. Let $Q = \phi,\ r = 1$, and $x_1 = x_2 = x_3 = 0$.
Step 1. Distribute the given amount of $d = 7$.
1.1. $i = 3$.
1.2. $x_3 = \min\{5, 7\} = 5,\ d = 7 - 5 = 2$, and $i = 2$.
1.3. Since $d = 2 > 0$, transfer is made to Step 1.2.
1.2. $x_2 = \min\{4, 2\} = 2,\ d = 2 - 2 = 0$, and $i = 1$.
1.3. Since $d = 0$, we let $X^1 = (0, 2, 5),\ Q = \{X^1\},\ r = 2$, and transfer is made to Step 2.
Step 2. Find the pivot component of solution X^1 to find the next solution as follows:
2.1. Let $i = 3$.
2.2. $i \neq 1$.
2.3. $d = 0 + 5 = 5,\ x_3 = 0$, and $i = 2$.
2.4. Since $x_2 = 2 \neq M_2$, x_2 is an pivot component.
2.5. $x_2 = 3$ and $d = 5 - 1 = 4$ and transfer is made to Step 1.
⋮

The final set of obtained solutions is: $Q = \{(0,2,5),\ (0,3,4),\ (0,4,3),\ (1,1,5),\ (1,2,4),\ (1,3,3),\ (1,4,2),\ (2,0,5),\ (2,1,4),\ (2,2,3),\ (2,3,2),\ (2,4,1)\}$.

According to Algorithm 1, we see that to generate a solution vector, steps 1 and 2 are iterated once. On the other hand, the time complexity of each iteration of steps 1 and 2 is $O(h)$, where h is the number of the components of the solution vector. Therefore, for obtaining each solution vector, the time complexity of Algorithm 1 is $O(h)$. Consequently, the time complexity of the algorithm is $O(h\gamma)$ where γ is assumed to be the number of solutions of the system. As a result, we have the following result.

Theorem 3.1. *The time complexity of Algorithm 1 is $O(h\gamma)$.*

The following lemma demonstrates the correctness of Algorithm 1.

Lemma 3.1. *If we solve system (7) by Algorithm 1, then the followings hold:*
(1) The solutions obtained by the algorithm are sorted in lexicographical order.
(2) The algorithm obtains all solutions of the system with no duplicates.

Proof. (1) Since in the beginning, the algorithm sets the maximum possible value to the components from the right end of the $h-$tuple vector, it is readily seen that the first solution obtained by the algorithm is the first one in lexicographical order. Suppose that Q is the solution set of the system in lexicographical order and X and Y are two arbitrary successive solutions of the system in set Q with X appearing immediately before Y. Also, assume that if X is obtained by the algorithm, the next solution obtained by the algorithm is Z. We show that $Z = Y$. The way the algorithm operates, Z is clearly after X in lexicographical order. Hence, since Y is immediately after X, the solution Z is equal to Y or comes after Y, i.e., $Z \geq Y$. Now, assume that the first difference between X and Z is in the jth component and the first difference between X and Y is in the rth component. Note that in accordance with the algorithm, it is clear that the first difference between X and Z appears at the pivot component selected in the second step of the algorithm, and so $z_j = x_j + 1$. Since the vectors Y and Z in Q both appear after X, their components are greater than the corresponding components of X in their first occurrence of a difference. Moreover, since Z is equal to or greater than Y, we have $r \geq j$. On the other hand, according to the way algorithm selects the pivot component, we know that $r \leq j$. Therefore, we have $r = j$. Now, since $z_j = x_j + 1$ and Z cannot be before Y, it is deduced that $z_j = y_j$. So, the first j components of both Y and Z are the same. Thus, the sum of their last $h - j$ components are the same as well. Since the algorithm constructs Z by filling in from the right end component with a maximum possible value, it is concluded that Z cannot be after Y, and consequently $Z = Y$.

(2) According to (1), we see that if Q is the solutions set of the system in lexicographical order, the algorithm find all the solutions of the system one by one exactly in the order they are set in Q. Hence, the proof is complete. □

For Diophantine system (7), the existing algorithms in the literature usually give the solutions by a specific solution, X^0, and a set of linearly independent spanning vectors, $X^1, X^2, \ldots,$ and X^α [2]. Hence, even with having this kind of solutions, to obtain each solution exception of X^0, we need at least $O(h)$ time to construct the solution according to

$X^1, X^2, \ldots$, and X^α (remind each solution is h-tuple). Therefor, since γ is assumed to be exactly the number of system solutions, not an upper bound for it, the best possible time complexity to have all the system solutions at hand is $O(h\gamma)$ where is the time complexity of Algorithm 1. Moreover, another remarkable point is that the algorithm finds all the solutions in a increasing lexicographic order that can be very useful in the *d-MC* and *d-MP* problems with budget or time constraints [9].

Acknowledgment

The authors thank Universidade Federal do ABC-UFABC for its support.

References

[1] R.K. Ahuja, T.L. Magnanti and J.B. Orlin, Network Flows: Theory, Algorithms, and Applications, Englewood Cliffs, NJ, Prentice-Hall International, (1993).

[2] T. Andreescu, D. Andrica, and I. Cucurezeanu, An Introduction to Diophantine Equations, A Problem-Based Approach, NY, Dordrecht Heidelberg, London, Springer, (2010).

[3] A.O. Balan and A. Traldi, Preprocessing minpaths for sum of disjoint products, IEEE Trans. Reliab., vol. 52(3), 289–95, (2003).

[4] M.O. Ball, Computational complexity of network reliability analysis: an overview, IEEE Trans. Reliab., vol. 35(3), 230–239, (1986).

[5] M. Forghani-elahabad and N. Mahdavi-Amiri, A Simple Algorithm to Find All Minimal Path Vectors to Demand Level d in a Stochastic-Flow Network, In Proceeding of the 5th Iranian Conference on Applied Mathematics, Iran, September 2-4, 532–535, (2013).

[6] M. Forghani-elahabad and N. Mahdavi-Amiri, A new efficient approach to search for all multi-state minimal cuts, IEEE Trans. Reliab., vol. 63(1), 154–66, (2014).

[7] C.C. Jane, J.S. Lin and J. Yuan, Reliability evaluation of a limited-flow network in terms of minimal cutsets, IEEE Trans. Reliab., vol. 42(3), 354–361, (1993).

[8] J.S. Lin, C.C. Jane and J. Yuan, On reliability evaluation of a capacitated-flow network in terms of minimal path sets, Networks, vol.3, 131–38, (1995).

[9] Y.K. Lin and C.T. Yeh, Multistate components assignment problem with optimal network reliability subject to assignment budget, Appl. Math. Comput., vol. 217, 10074-86, (2011).

[10] W.W. Wu, A. Ning and X.X. Ning, Evaluation of the reliability of transport networks based on the stochastic flow of moving objects, Reliab. Eng. Syst. Saf., vol. 93, 838–844, (2008).