IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received October 22, 2020, accepted November 10, 2020, date of publication November 16, 2020,
date of current version November 24, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3038116

An Efficient Method to Find All d-MPs in Multistate Two-Terminal Networks

YASSER LAMALEM, KHALID HOUSNI, AND SAMIR MBARKI
MISC Laboratory, Faculty of Science, Ibn Tofail University, Kenitra 14000, Morocco
Corresponding author: Yasser Lamalem (yasserlamalem@gmail.com)

ABSTRACT Many systems in the real-world are multi-state systems composed of multi-state components (nodes and arcs) in which the reliability of the system can be calculated in terms of the lower bound points of level d, called d-Minimal Paths (d-MPs). Such systems (oil/gas production, power transmission and distribution networks, etc.) may be considered as network flows whose arcs have independent and discrete random capacities. In this paper, we present a new algorithm to generate all d-MPs candidates for all d levels. Our algorithm is based on the incremental enumeration of the different d-MPs. In the beginning, we determine the list of all minimal paths (also called 1-MPs). The latter will then be used to generate the 2-MPs. Then from 1-MPS and 2-MPs we generate 3-MPs ...and so on. The greatest contribution of our method is the use of a technique which makes it possible to know, at each level i, which minimal paths can lead to valid (i+1)-MPs without resorting to an additional validation process. This has reduced the execution time compared to the existing algorithms to find all d-MPs for all possible d values.

INDEX TERMS Reliability, d-minimal paths, minimal paths, multi-state network, d-MPs.

I. INTRODUCTION

The problem which arises during the evaluation of network reliability is, when the reliability of the elements that are subject to random failure are known, it is usually not easy to obtain and takes a lot of time when the network is large. This computation time becomes quite important when the network is non series-parallel and this is the case in most real systems.

Various methods that serve for the determination of the reliability of a network have been proposed in the literature,and as examples, we can cite [1]–[11]. These methods can be classified into state enumeration methods [1]–[3] and cut/path set enumeration methods [4]–[10]. Most of these methods require advanced mathematics or can only be applied to either oriented or non oriented graphs.

In many real-life situations, multi-state systems are more reasonable and practical than binary-state systems. A limited-flow network is a multi-state system whose arc capacities can be regarded as independent and discrete random capacities. The reliability evaluation of a multi-state network flow is NP-hard but possible [12], [13]. Several authors have proposed several approaches to evaluate the probability that the system capacity level or its lower bound is d [2], [14]–[18].

The associate editor coordinating the review of this manuscript and approving it for publication was Amjad Gawanmeh.

In a binary-state flow network, the capacity of each arc has two levels 0 and a positive integer. Given the demand d, the system reliability is the probability that the maximum flow of the network is greater than d.

Lin [2] proposed a mathematical formulation with three constraints to obtain the d-MP candidates using all the minimal paths. After generating the d-MP candidates, a recursive comparison is made to verify each d-MP candidate as a result all d-MPs are obtained for a particular level d. Ramirez-Marquez *et al.* [19] used another technique called the information sharing mechanism. It does not require MPs as prior knowledge, and this new technique seems to be able to reduce the search space.

Chen and Lin [16] has proposed an algorithm to search for all d-minimal paths (d-MPs) in a multi-state networks using the pre-calculated minimal paths, the algorithm is using the integer programming problems in the literature [20]. The presented algorithm is reducing a huge number of steps by re-arranging the order of formulas in the algorithm.

Bai *et al.* [14] introduced a new method that says all d-MP candidate can be generated by a combination of MPs, and verifies each d-MP candidate with the maximum capacity for each component (maximum state vector).

Yeh [21] presented an improvement for the Bai's [14] algorithm, by avoiding undesired d-MP candidates with infeasible states. The author also proposed a new method called

logarithmic prime number transfer for removing all duplicate d-MP candidates, which is faster than recursive comparison used in Bai *et al.* [14].

In this paper, we present a new algorithm to generate all d-MPs candidates for all d levels. Our algorithm is based on the incremental enumeration of the different d-MPs. Unlike Yeh's [21], our method is not meant for improving Bai's [14], on the contrary it's a new technique that generates all d-MPS. At the beginning, we determine the list of all minimal paths (also called 1-MPs). The latter will then be used to generate the 2-MPs. Then from 1-MPS and 2-MPs we generate 3-MPs … and so on. The greatest contribution of our method is the use of a technique which makes it possible to know, at each level i, which minimal paths can lead to valid (i+1)-MPs without resorting to an additional validation process. This has reduced the execution time compared to the existing algorithms for finding all d-MPs for all possible d values. This method generates many duplicates like Bai's [14], to eliminate those duplicates we used the same algorithm in [21] that's called logarithmic prime number transfer.

This paper is organized as follows: The next section presents the mathematical preliminaries for the method. In sections 3, we describe the proposed algorithm for finding all d-minimal paths for all d levels in a multi-state two-terminal networks. Section 4 contains an illustrative example. Finally, Section 5 compares the efficiency of the proposed algorithm to the fastest algorithm in the literature.

II. PRELIMINARIES

A. NOTIONS AND ASSUMPTIONS

1) NOTATION

n: the number of components in the network system.

m: the number of binary MP in the network system.

P_i: the vector form of ith P, i = 1,2,..,m.

a_i: the ith component in the network system, i = 1,2,..,n.

X: System-state vector $X^i = \{x_1, x_2, x_3, \ldots, x_n\}$ where x_i denotes the state of component a_i.

X^{Max}: Maximum capacity $X^{Max} = (M^1, \ldots, M^n)$ of each arc a_i.

C_v: the v^{th} cycle in the network, $C_v = (x_i, x_j, \ldots, x_K)$, v = 1,2,…,c.

N: The number of d-MPs candidates founded in the moment of checking.

P_r: The probability that the maximum flow is not less than d.

R_d: The system reliability such as the probability that the maximum flow is not less than d.

$V(X)$: The sum of the flows of all the minimal paths in the network under a capacity vector $X = (x_i, x_j, \ldots, x_k)$

2) DEFINITION

G(A,N,M): stochastic-flow network where $A = (a_1, \ldots, a_n)$ represents the set of all arcs, $N = (n_1, \ldots, n_l)$ represents the set of all nodes and $X^{Max} = (M^1, \ldots, M^n)$.

Path: is a set of components, which connect the source node s and sink node t.

Minimal paths: is a sequence of components from source node s to sink node t, which contains no cycle.

Paths matrix (P_m): Let G(A,N) be a network, P_m is a matrix where the columns are the arcs a_i and the rows are all the MPs of the network. If an arc a_i is presented in the MP then the value is 1, otherwise 0.

Reduced matrix (P_r): it is the matrix obtained after the removal of some rows from the Paths matrix.

3) FUNCTIONS

check_cycle(X, C_v): Accepts as an argument a state capacity vector and all cycles in the graph, and returns TRUE if there exists at least one component for X that has capacity equal to 0 for each cycle $c \in C_v$:

$$\min(c) = \min(x_i, x_j, \ldots, x_k) = 0.$$

4) ASSUMPTIONS

The network is assumed to satisfy the following assumptions:

- The nodes are perfectly reliable.
- The states in arcs are statistically independent from each other.
- Flow in the network represented by G(A,N,M) satisfies the flow-conservation law [22].
- The capacity of each component a_i is an integer-valued random variable which takes values $0<1<2\ldots<M^n$ according to a given distribution.
- The network is oriented. For a non oriented network, an approach reported in [23] can be used to transform it into a directed network.

B. STOCHASTIC-FLOW NETWORK MODEL

1) NETWORK RELIABILITY EVALUATION

As it is already presented in the introduction, in a binary-state flow network, the capacity of each arc has two levels 0 and a positive integer. Given the demand d, the system reliability is the probability that the maximum flow of the network is greater than d, i.e., $R_d = Pr\{X|V(X) \geq d\}$. However, in a multi-state network, as it is introduced in [2], the enumeration of all possible cases such that the maximum flow of the network is greater than or equal to d is not wise. To overcome this difficulty, the author in [2] shows that the reliability of a network can be evaluated according to the lower boundary points for d which can be defined as a any minimal vector Y in the set $\{X|V(X) \geq d\}$ such that for any capacity vector $Z < Y$, $V(Z) < d$. In the case of a multi-state network, since the capacity of each link varies between 0 and M^{max}, the condition " for any capacity vector $Z < Y$, $V(Z) < d$ " is always checked for cases where $\{X|V(X) = d\}$.

2) STOCHASTIC-FLOW NETWORK MODEL

Let $p_1, p_2, \ldots, p_m$ be the minimal paths from the source node s to the sink node t. The stochastic-flow network can

be described in terms of two vectors: the flow vector F = $(f_1, f_2, \ldots f_m)$ and the capacity vector X = $(x_1, x_2, \ldots, x_n)$, where f_i denotes the current flow of a minimal path p_i and x_i denotes the current capacity of an arc a_i. The vector F is feasible if and only if:

$$\sum_{j=1}^{m} \{f_j | a_i \in p_j\} \leq M_i, \quad i = 1, 2, \ldots n \tag{1}$$

$$\sum_{j=1}^{m} f_j = d \tag{2}$$

$$f_j \leq min\{M^i | a_i \in p_j\}, \quad j = 1, 2, \ldots m \tag{3}$$

The constraint (1) describes that the total flow through a_i under F cannot exceed the maximal capacity of a_i.

The constraint (2) says that the total flow is equal to the demand given d.

Constraint (3) says that the flow on each p_j cannot exceed the maximum capacity of p_j

Let $\mathbf{F} = \{F \mid$ satisfies the constrains (1) and (2)$\}$

If $X = (x_1, x_2, \ldots, x_n)$ is a d-MPs for d, then there is a $F \in F$, such that:

$$x_i = \sum_{j=1}^{m} \{f_j | a_i \in p_j\}, \quad i = 1, 2, \ldots n \tag{4}$$

III. ALGORITHM

A. BASIC IDEA

All d-MPs for multistate networks can be found using the pre-calculated minimal paths, according to Lin [24] there are three constraints to obtain all d-MPs candidates:

- the summation of the flow on all the MPs equal to d as constraint (2) says.
- the total flow on each MP is smaller than or equal to the maximum capacity on that MP as constraint (1) says.
- the total flow going through each component is less than or equal to its maximum capacity as constraint (3) says.

After all the d-MP candidates are generated, each d-MP candidate is verified and all d-MPs are obtained for a particular level d.

Bai *et al.* [14] uses the constrain in [24] that says: The capacity of each component in X^i is smaller than or equal to the maximum capacity of its corresponding component X^{Max}, that is:

Let $X^i = \{x_1, x_2, x_3, \ldots, x_n\}$,

$$X^i \leq X^{Max} \tag{5}$$

where $x_i \leq M^i$ for i = 1,2,…,n.

Bai *et al.* [14] generates all the d-MPs for level d by combining all pre-calculated MPs. For finding d-MPs for a level d, he adds (d-1)-MPs for level d-1 with 1-MPs for level 1, and all the new d-MPs are verified using the previous constraint (5). For acyclic networks all the d-MPs candidates found are real d-MPs, but for a cyclic network, X_i contains no cycle [17], [24], there exists a least one component for X_i that has a capacity equal to zero for each cycle. For example for finding the d-MPS for level 5, he has to find the d-MPs for level 4 (4-MPs) and add them with the d-MPs for level 1 (1-MPs), and so on. and in each generated d-MPS he removes redundant d-MPs, and compares them with the maximum capacity of its corresponding component. The problem here is the comparison between the d-MPs candidates and the maximum capacity X^{max}, if we have n component a_n and m generated d-MPs, only the complexity of the comparison is $O\{n*m\}$, which take a lot of time.

As it can be observed that each d-MP candidate cannot exceed the maximum capacity of its corresponding component X^{max}. The proposed method takes advantage of this constraint and generating all the d-MPs for all d levels for a multi-state network, without comparing the d-MPS candidates with the maximum capacity X^{max}.

Let X^i be a 1-MP (all the MPs are d-MPs for a level 1), according to the constraint (5), $X^i \leq X^{Max}$, that means:

$X^{Max} - X^i = \{r_1, r_2, \ldots, r_n\}$, where $r_i \geq 0$, for i = 1,2,…,n.

Let X^j be a 1-MP, $X^{Max} - X^i - X^j = \{r_1, r_2, \ldots, r_n\}$, if $r_i \geq 0$, for i = 1,2,…,n. Then the combination of X^i and X^j are a d-MPs for level 2.

This technique is useful to find all the d-MPs for all the d levels, but we can't choose any candidates X_i or X_j, for example:

Let $X^{Max} = \{3,1,2\}$, $X^1 = \{1,1,0\}$, $X^2 = \{0,1,1\}$ and $X^3 = \{1,0,1\}$, below is the corresponding paths matrix:

$$P_m = \begin{bmatrix} & a_1 & a_2 & a_3 \\ P_1 & 1 & 1 & 0 \\ P_2 & 0 & 1 & 1 \\ P_3 & 1 & 0 & 1 \end{bmatrix}$$

$X^{Max} - X^1 = \{2, 0, 2\}$, we have all $r_i \geq 0$, for i = 1,2,3.

But in $X^{Max} - X^1 - X^2 = \{2, -1, 1\}$, we have $r_2 \leq 0$, then the combination of X_1 and X_2 are not a 2-MPs.

The problem here is to choose the right MP, that can lead to a result, so the proposed method can have a big advantage over Bai's [14]. From the previous example $X^{Max} - X^1$, has some components a_i that's equal to 0 ($a_2 = 0$). Therefore, the next MPs that will be used in the algorithm must have in his representation the corresponding $a_i = 0$. To find such MPs, we use a new matrix called the paths matrix P_m.

To find all the minimal paths that have $a_i = 0$ in their representation, we remove all rows from the paths matrix P_m that have $a_i = 1$ to obtain the reduced matrix P_r. The new matrix P_r contains all the MPs that can be used to generate all the next d-MPs for a level d without comparing them with the maximum capacity X^{max}.

Given the previous example, the reduced matrix for all paths that have $a_2 = 0$ is:

$$P_r = \begin{bmatrix} & a_1 & a_2 & a_3 \\ P_3 & 1 & 0 & 1 \end{bmatrix}$$

Then $X^{Max} - X^1 - X^3 = \{1, 0, 1\}$, as a result the combination of X^1 and X^3 are 2-MPs.

The new $X^{Max} = \{1,0,1\}$, have $a_2 = 0$. For finding the d-MPs for level 3, we use the same reduced matrix P_r that we have already found and we remove all rows that have $a_2 = 1$ to obtain the new reduced matrix P_r (The matrix stays the same because P_3 have $a_2 = 0$).

$$P_r = \begin{bmatrix} & a_1 & a_2 & a_3 \\ P_3 & 1 & 0 & 1 \end{bmatrix}$$

Then $X^{Max} - X^1 - X^3 - X^3 = \{0, 0, 0\}$, as a result the combination of X^1, X^3 and X^3 are 3-MPs.

As it can be noticed from this example, all the d-MPs candidates are generated without comparing them with the maximum capacity of its corresponding component X^{Max}, which can reduce a lot of time compared to Bai's [14].

The second advantage of the proposed algorithm, is that we don't have to add all the 1-MPs with (d-1)-MPs to generate d-MPs candidates. For example, in Bai's [14], to generate the 2-MPs candidates for level 2, if we take X^1 as the first MP, we add $X^1+X^1, X^1+X^2, X^1+X^3$ and then compare the result of those 2-MPs candidates with the maximum capacity X^{Max}, but as we have already seen in our method, when we used the path matrix P_m we only use the combination X^1+X^3 (because X^3 has $a_2 = 0$). With this method we are not only saving time for the comparison between the d-MPs candidates and the maximum capacity X^{Max}, but also the time for adding the rest of the (d-1)-MPs with 1-MPs that cannot lead to a result.

B. ELIMINATING DUPLICATES

The proposed algorithm generates many duplicates like Bai's [14], especially when the network is large. This requires additional time and memory to remove duplicates. The complexity of checking if a d-MP candidate is a redundant using recursive comparison is *O(N*m)*. As we can see when the number of d-MPs candidates and the number of arcs grows the complexity of checking one d-MP grows as well.

Yeh's [21] presented an improvement for the Bai's [14] algorithm by avoiding undesired d-MP candidates with infeasible states. For this, the author proposed a new method called logarithmic prime number transfer. The main idea of this method is to transfer each vector X to a value $L(X)$ equal to the multiplication of the logarithm of distinctive prime numbers and then sort these values to find duplicates.

Calculation methods and lemmas presented as follows:

1) Let $X = \{x_1, x_2, x_3, \ldots, x_n\}$ is a state system vector, then $X(e_i) = x_i$ and $L(X) = \sum_{i=1}^{m} x_i log(\lambda_i)$. For example: $X = \{2, 1, 0, 3\}$, $L(X) = 2 * log(2) + 1 * log(3) + 0 * log(5) + 3 * log(7) = 3.614475366$.
2) $L(X) \neq L(Y)$, if and only if $X \neq Y$ for all vectors X and Y.
3) $L(X + Y) = L(X) + L(Y)$
4) If $L(X) < L(Y)$, then $L(X + P_i) < L(Y + P_i)$.

C. PROPOSED ALGORITHM

Below is the proposed algorithm to generate all the d-MPs for all d levels. The algorithm accepts as a first call the path matrix P_m, the maximum state vector X^{Max}, the current demand d, all the cycles C_v (v = 1,12…c) and the vector *Prev* contains the sum of the previous vectors.

Algorithm 1 *DMPS* (P_m, X^{Max},d,C_v,*Prev*,*Num*)

1: **for all** $a_i = 0$ **in** X^{Max} **do**
2: Delete all the corresponding rows that have a_i equal to 1 in P_m to obtain P_r, beginning from path number *Num*.
3: **end for**
4: **for** X^i **in** P_r **do**
5: $X^r = X^{Max} - X^i$.
6: $D\_MP = Prev + X^i$
7: **if** *D_MP* is not a redundant **AND** check_cycle(*D_MP*, C_v) **then**
8: Output: a new d-MPs candidate of level d is generated *D_MP*.
9: **DMPS**(P_r, X^r,d+1,C_v,*D_MP*,i)
10: **end if**
11: **end for**

The reason for using the variable *Num* is to reduce the number of redundant d-MPs candidates generated by the algorithm. Taken the previous example:

$$P_m = \begin{bmatrix} & a_1 & a_2 & a_3 \\ P_1 & 1 & 1 & 0 \\ P_2 & 0 & 1 & 1 \\ P_3 & 1 & 0 & 1 \end{bmatrix}$$

The algorithm in some point will arrive to: $X^{Max} - X^1 - X^2$ and also to $X^{Max} - X^2 - X^1$, the two equations have the same result. Therefore, to reduce the path matrix P_m, we start from the path P_{Num} and remove all the rows from the paths matrix P_m that have $a_i = 0$ in X^{Max}. to obtain the reduced matrix P_r. If we take the previous path matrix P_m and $Num = 2$, the algorithm will start from the path P_2 and will ignore the path P_1.

IV. ILLUSTRATIVE EXAMPLE

Considering the graph in the Fig.1 taking from [14].The algorithm at the first call will take the maximum state vector $X^{Max} = (3, 2, 1, 1, 1, 2)$, cycle C = ($x_3$, x_4), the *Prev* = 0,0,0,0,0,0, d = 0 and the paths matrix P_m is:

$$P_m = \begin{bmatrix} & a_1 & a_2 & a_3 & a_4 & a_5 & a_6 \\ P_1 & 1 & 1 & 0 & 0 & 0 & 0 \\ P_2 & 0 & 0 & 0 & 0 & 1 & 1 \\ P_3 & 1 & 0 & 1 & 0 & 0 & 1 \\ P_4 & 0 & 1 & 0 & 1 & 1 & 0 \end{bmatrix}$$

Call 1: first call of the algorithm:

X^{Max} has no component $a_i = 0$. Therefore $P_r = P_m$.

for $X^1 = \{1, 1, 0, 0, 0, 0\}$

$X^r = X^{Max} - X^1 = \{2, 1, 1, 1, 1, 2\}$, $D\_MP = \{1, 1, 0, 0, 0, 0\}$.

D_MP is not a redundant and check_cycle(*D_MP*,C) return True (min(x_3, x_4) = min(0,0) = 0).

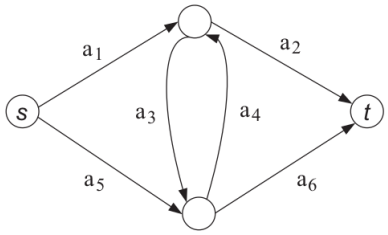

FIGURE 1. A network example.

A new 1-MPs candidate of level 1 is generated {1, 1, 0, 0, 0, 0}.

Then we call the algorithm with the new arguments.

Recursive call 1.1:

X^{Max} = {2, 1, 1, 1, 1, 2}, cycle C = (x_3, x_4), the *Prev* = {1, 1, 0, 0, 0, 0}, d = 2.

X^{Max} has no component $a_i = 0$. Therefore $P_r = P_m$.

for X^1 = {1, 1, 0, 0, 0, 0}

$X^r = X^{Max} - X^1$ = {1, 0, 1, 1, 1, 2}, *D_MP* = {2, 2, 0, 0, 0, 0}.

D_MP is not a redundant and check_cycle(*D_MP*,C) return True (min(x_3, x_4)= min(0,0) = 0).

A new 2-MPs candidate of level 2 is generated {2, 2, 0, 0, 0, 0}.

Then we call the algorithm with the new arguments.

Recursive call 1.1.1: X^{Max} = {1, 0, 1, 1, 1, 2}, cycle C = (x_3, x_4), the *Prev* = {2, 2, 0, 0, 0, 0}, d = 3.

X^{Max} have $a_2 = 0$. Therefore we remove the rows that have the value 1 from P_m to obtain P_r:

$$P_m = \begin{bmatrix} & a_1 & a_2 & a_3 & a_4 & a_5 & a_6 \\ P_2 & 0 & 0 & 0 & 0 & 1 & 1 \\ P_3 & 1 & 0 & 1 & 0 & 0 & 1 \end{bmatrix}$$

for X^2 = {0, 0, 0, 0, 1, 1}

$X^r = X^{Max} - X^2$ = {1, 0, 1, 1, 0, 1}, *D_MP* = {2, 2, 0, 0, 1, 1}.

D_MP is not a redundant and check_cycle(*D_MP*,C) return True (min(x_3, x_4) = min(0,0) = 0).

A new 3-MPs candidate of level 3 is generated {2, 2, 0, 0, 1, 1}.

Then we call the algorithm with the new arguments.

Recursive call 1.1.1.1:

X^{Max} = {1, 0, 1, 1, 0, 1}, cycle C = (x_3, x_4), the *Prev* = {2, 2, 0, 0, 1, 1}, d = 4.

X^{Max} has $a_2 = 0$ and $a_5 = 0$. Therefore we remove the rows that have the value 1 from P_m to obtain P_r:

$$P_m = \begin{bmatrix} & a_1 & a_2 & a_3 & a_4 & a_5 & a_6 \\ P_3 & 1 & 0 & 1 & 0 & 0 & 1 \end{bmatrix}$$

for X^3 = {1, 0, 1, 0, 0, 1}

$X^r = X^{Max} - X^3$ = {0, 0, 0, 1, 0, 0}, *D_MP* = {3, 2, 1, 0, 1, 2}.

D_MP is not a redundant and check_cycle(*D_MP*,C) return True (min(x_3, x_4)=min(0,1)=0).

A new 4-MPs candidate of level 4 is generated {3, 2, 1, 0, 1, 2}.

Then we call the algorithm with the new arguments.

Recursive call 1.1.1.1.1:

X^{Max} = {0, 0, 0, 1, 0, 0}, cycle C = (x_3, x_4), the *Prev* = {3, 2, 1, 0, 1, 2}, d = 5.

X^{Max} has all $a_i = 0$ except a_4. Therefore we remove the rows that have the value 1 from P_m to obtain P_r, P_r = {∅}, exit the branch.

Recursive call 1.1.2:

for X^3 = {1,0, 1, 0, 0, 1}

$X^r = X^{Max} - X^3$ = {0, 0, 0, 1, 1, 1}, *D_MP* = {3, 2, 1, 0, 0, 1}.

D_MP is not a redundant and check_cycle(*D_MP*,C) return True (min(x_3, x_4) = min(1,0) = 0).

A new 3-MPs candidate of level 3 is generated {3, 2, 1, 0, 0, 1}.

Then we call the algorithm with the new arguments.

Recursive call 1.1.2.1:

X^{Max} = {0, 0, 0, 1, 1, 1}, cycle C = (x_3, x_4), the *Prev* = {3, 2, 1, 0, 0, 1}, d = 4.

X^{Max} has $a_1 = 0, a_2 = 0$ and $a_5 = 0$. Therefore we remove the rows that have the value 1 from P_m to obtain P_r:

$$P_m = \begin{bmatrix} & a_1 & a_2 & a_3 & a_4 & a_5 & a_6 \\ P_2 & 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix}$$

for X^2 = {0, 0, 0, 0, 1, 1}

$X^r = X^{Max} - X^3$ = {0, 0, 0, 1, 0, 0}, *D_MP* = {3, 2, 1, 0, 1, 2}.

D_MP is a redundant then exit the branch.

In the end, we obtain all the d-MPs for all d levels, as the result shows:

1-MPs {1, 1, 0, 0, 0, 0}, {0, 0, 0, 0, 1, 1}, {1, 0, 1, 0, 0, 1},{0, 1, 0, 1, 1,0};

2-MPs {2, 2, 0, 0, 0, 0}, {1, 1, 0, 0, 1, 1}, {2, 1, 1, 0, 0, 1}, {1, 2, 0, 1, 1, 0}, {1, 0, 1, 0, 1, 2};

3-MPs {2, 2, 0, 0, 1, 1}, {3, 2, 1, 0, 0, 1}, {2, 1, 1, 0, 1, 2};

4-MPs {3, 2, 1, 0, 1, 2}.

V. BENCHMARKS AND TEST

A. FIRST TEST

To evaluate the efficiency of the proposed algorithm, we have compared it with the Bai's algorithm [14] using the graph in the Fig.1, each time we increase the maximum capacity of the component. In this test we have used the same recursive comparison in Bai's [14] for deleting redundant d-MPS candidates.

The benchmarks used to evaluate the efficiency of our algorithm is the same as that used by Bai *et al.* [14] and Yeh [21].

In this test, we used HP core i7 second generation 8 GB Ram. As for the implementation language, we used C++.

As a result of the test (see Figure 2 and table 1), the proposed algorithm is more efficient than Guanghan Bai's algorithm [14] in terms of the execution time. The ratio, which is defined as the ratio of the CPU time of the Bai's algorithm to the proposed algorithm, is about 1.5 (see figure 3),

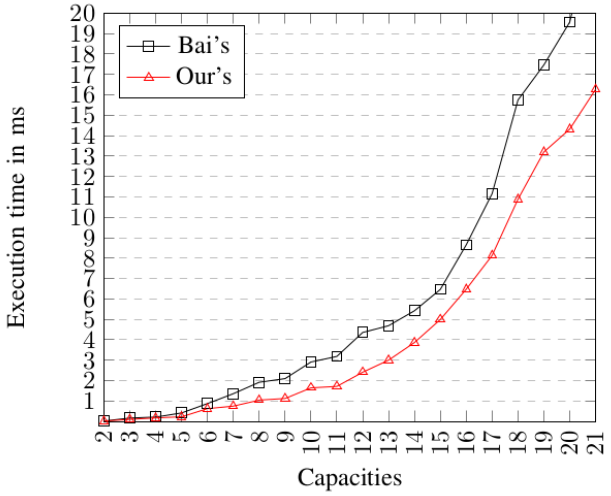

FIGURE 2. Execution time of Bai's [14] vs Execution time of the proposed algorithm.

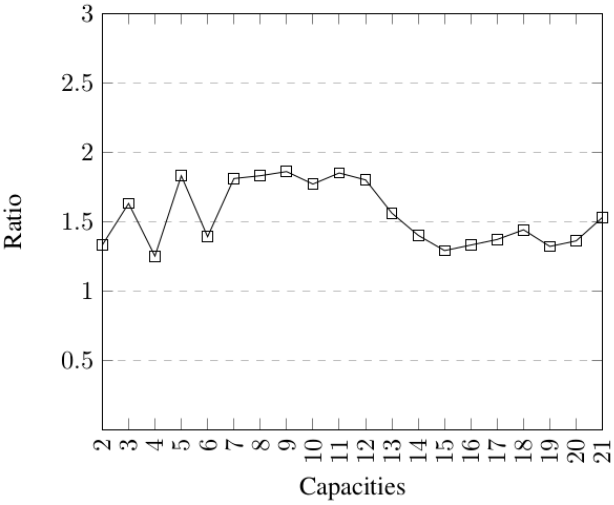

FIGURE 3. Ratio of the CPU time of the Bai's algorithm to the proposed algorithm.

indicating that the proposed algorithm is 1.5 times faster than Bai's algorithm in finding all the d-MPs of the benchmark network.

B. SECOND TEST

In the first test, to eliminate all the redundant d-MPs candidates, we used the same recursive comparison as was used by Bai *et al.* [14]. The reason for that is to prove that the basic idea of our algorithm is more efficient and faster than Bai's [14].

To improve our algorithm, we propose the same method "logarithmic prime number transfer" used in Yeh's [21] to remove all the redundant d-MPs candidates, instead of the recursive comparison. To test this improvement we have compared the proposed algorithm with the Yeh's algorithm [21]. To carry out this test we used the graph in the Fig.1, and at each time, we increase the maximum capacity of the component. The implementation is done in C++ and

TABLE 1. Comparison result between the 2 algorithms in terms of execution time in milliseconds.

M	Bai [14]	Our method	M	Bai [14]	Our method
2	0.02	0.015	3	0.163	0.1
4	0.213	0.170	5	0.414	0.226
6	0.872	0.624	7	1.357	0.747
8	1.917	1.043	9	2.095	1.121
10	2.91	1.643	11	3.183	1.714
12	4.357	2.409	13	4.689	3.001
14	5.421	3.865	15	6.468	4.993
16	8.651	6.475	17	11.158	8.132
18	15.748	10.885	19	17.466	13.201
20	19.568	14.322	21	24.859	16.272

TABLE 2. Comparison result between the 2 algorithms in terms of execution time in milliseconds.

M	Yeh [21]	Our method	M	Yeh [21]	Our method
2	0.012	0.009	3	0.089	0.041
4	0.15	0.095	5	0.226	0.151
6	0.308	0.227	7	0.391	0.247
8	0.531	0.342	9	0.747	0.564
10	0.968	0.668	11	1.551	0.8
12	1.695	0.915	13	1.932	1.11
14	2.291	1.443	15	2.943	1.894
16	3.438	2.619	17	3.561	1.958
18	4.049	2.661	19	4.928	2.017
20	4.186	2.488	21	5.054	2.962

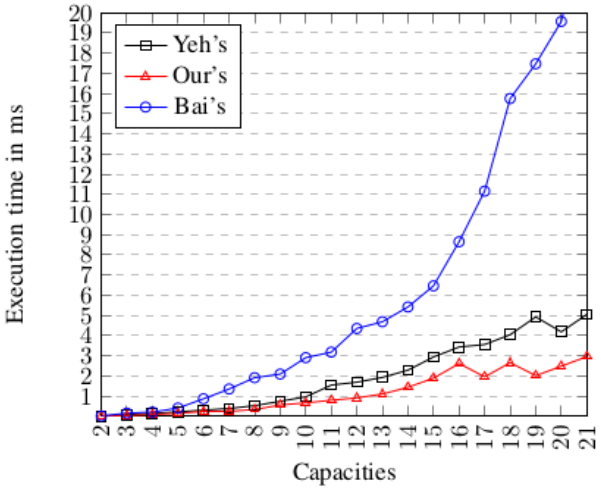

FIGURE 4. Execution time of Yeh's [21] and Bai's [14] vs Execution time of the proposed algorithm.

the experimental computer configuration is intel i7 second generation 8 GB Ram.

The experimental results are listed in Table 2, including the CPU run times in milliseconds consumed by our algorithm and Yeh's algorithm [21]. The result of the test shows that the proposed algorithm with the new improvement is more efficient than Yeh's [21] as the maximum state increases.

In Figure 4, we compare the efficiency of our improved algorithm with Yeh's [21] and Bai's [14] in terms of the execution time. As the result shows, the more we increase the maximum state the difference between the three algorithm increases.

The ratio, which is defined as the ratio of the CPU time of the Yeh's algorithm [21] to the proposed algorithm, is about 1.5 as in the first test (see figure 5), indicating that the

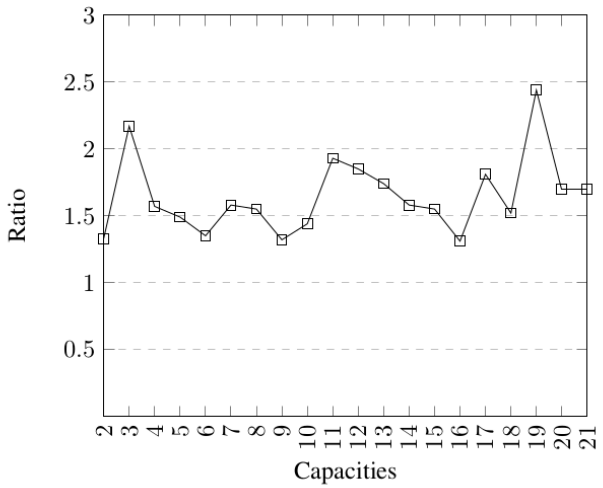

FIGURE 5. Ratio of the CPU time of the Yes's algorithm to the proposed algorithm.

proposed algorithm is 1.5 times faster than Yeh's algorithm [21] in finding all the d-MPs of the benchmark network.

VI. CONCLUSION

In this study, we develop a recursive algorithm based on deep-first search to find all the d-MPs in a multi-state network. The proposed method consists of taking advantage of the constraint that each d-MP candidate cannot exceed the maximum capacity of its corresponding component X^{max}. This constraint allows us to generate all the d-MPs for all d levels for a multi-state network, and as the tests show, the proposed algorithm is more efficient than Guanghan Bai's algorithm [14] and Yeh's [21] as we added more nodes to the graph.

REFERENCES

[1] W. Dotson and J. Gobien, "A new analysis technique for probabilistic graphs," *IEEE Trans. Circuits Syst.*, vol. 26, no. 10, pp. 855–865, Oct. 1979.
[2] Y.-K. Lin, "A simple algorithm for reliability evaluation of a stochastic-flow network with node failure," *Comput. Oper. Res.*, vol. 28, no. 13, pp. 1277–1285, Nov. 2001.
[3] A. C. Nelson, J. R. Batts, and R. L. Beadles, "A computer program for approximating system reliability," *IEEE Trans. Rel.*, vols. R–19, no. 2, pp. 61–65, May 1970.
[4] G. Bai, Z. Tian, and M. J. Zuo, "An improved algorithm for finding all minimal paths in a network," *Rel. Eng. Syst. Saf.*, vol. 150, pp. 1–10, Jun. 2016.
[5] S.-G. Chen and Y.-K. Lin, "Search for all minimal paths in a general large flow network," *IEEE Trans. Rel.*, vol. 61, no. 4, pp. 949–956, Dec. 2012.
[6] C.-C. Jane and Y.-W. Laih, "A practical algorithm for computing multi-state two-terminal reliability," *IEEE Trans. Rel.*, vol. 57, no. 2, pp. 295–302, Jun. 2008.
[7] G. B. Jasmon and O. S. Kai, "A new technique in minimal path and cutset evaluation," *IEEE Trans. Rel.*, vols. R–34, no. 2, pp. 136–143, Jun. 1985.
[8] A. M. Al-Ghanim, "A heuristic technique for generating minimal path and cutsets of a general network," *Comput. Ind. Eng.*, vol. 36, no. 1, pp. 45–55, Jan. 1999.
[9] C. J. Colbourn, *The Combinatorics of Network Reliability*. New York, NY, USA: Ox-Ford Univ. Press, 1987.
[10] Y. Lamalem, K. Housni, and S. Mbarki, "New and fast algorithm to minimal cutsets enumeration based on necessary minimal paths," in *Proc. 5th Int. Symp. Innov. Inf. Commun. Technol. (ISIICT)*, Oct. 2018, pp. 1–5.
[11] Y. Lamalem and K. Housni, "New and efficient method to find all minimal paths," in *Proc. 3rd Int. Conf. Adv. Commun. Technol. Netw. (CommNet)*, Sep. 2020, pp. 1–4.
[12] B. Jl, "Evaluation of the unavailability of a multistate-component system using a binary model," *Rel. Eng. Syst. Saf.*, vol. 64, pp. 7–13, Dec. 1999.
[13] P. Js and B. Mo, "The complexity of counting cuts and of computing the probability that a graph is connected," *SIAM J. Comput.*, vol. 12, pp. 77–88, Dec. 1983.
[14] G. Bai, M. J. Zuo, and Z. Tian, "Search for all d-MPs for all d levels in multistate two-terminal networks," *Rel. Eng. Syst. Saf.*, vol. 142, pp. 300–309, Oct. 2015.
[15] J. Lin, C. Jane, and J. Yuan, "A simple algorithm to search for all d-mps with unreliable nodes," *Rel. Eval. Capacitated-Flow Netw. Terms Minimal Pathsets*, vol. 25, pp. 131–138, May 1995.
[16] S.-G. Chen and Y.-K. Lin, "Searching for d-MPs with fast enumeration," *J. Comput. Sci.*, vol. 17, pp. 139–147, Nov. 2016.
[17] W.-C. Yeh, "A simple algorithm to search for all d-MPs with unreliable nodes," *Rel. Eng. Syst. Saf.*, vol. 73, no. 1, pp. 49–54, Jul. 2001.
[18] M. J. Zuo, Z. Tian, and H.-Z. Huang, "An efficient method for reliability evaluation of multistate networks given all minimal path vectors," *IIE Trans.*, vol. 39, no. 8, pp. 811–817, May 2007.
[19] J. Ramirez-Marquez, D. Coit, and M. Tortorella, *IIE Trans*, vol. 38, pp. 477–488, May 2006.
[20] S. G. Chen, "Efficiency improvement in explicit enumeration for integer programming problems," in *Proc. IEEE Int. Conf. Ind. Eng. Eng. Manage.*, Dec. 2013, pp. 98–100.
[21] W.-C. Yeh, "Fast algorithm for searching d -MPs for all possible d," *IEEE Trans. Rel.*, vol. 67, no. 1, pp. 308–315, Mar. 2018.
[22] L. Ford and D. Fulkerson, *Flows in Networks*. Princeton, NJ, USA: Princeton Univ. Press, 1962.
[23] H. JN, "Note on independence of arcs in antiparallel for network flow problems," *Networks*, vol. 4, pp. 567–570, Dec. 1984.
[24] J. Lin, C. Jane, and J. Yuan, "On reliability evaluation of a capacitated—Flow network in terms of minimal pathsets," *Networks*, vol. 25, pp. 131–138, Jun. 1995.

YASSER LAMALEM was born in Kenitra, Morocco. He received the master's degree in computer science from Ibn Tofail University, in 2016, where he is currently pursuing the Ph.D. degree in network reliability. His current research interests include multi-state networks reliability and binary-state networks reliability.

KHALID HOUSNI received the Master of Advanced Study degree in applied mathematics and computer science, and the Ph.D. degree in computer science from the Ibn Zohr University of Agadir, Morocco, in 2008 and 2012, respectively. He joined the Department of Computer Science, University Ibn Tofail of Kenitra, Morocco, in 2014, where he has been involved in several projects in video analysis and networks reliability. His current research interests include image/video processing, computer vision, machine learning, artificial intelligence, pattern recognition, and networks reliability.

SAMIR MBARKI received the B.S. degree in applied mathematics and the Doctorate of High Graduate Studies degree in computer sciences from Mohammed V University, Morocco, in 1992 and 1997, respectively, and the HDR degree in computer science from Ibn Tofail University, in 2010. He joined the Faculty of Science, Ibn Tofail University, Morocco, in 1995, where he is currently a Professor with the Department of Computer Science. His research interests include software engineering, model driven architecture, software metrics, and software tests.