Algorithmica (2000) 27: 316–336
DOI: 10.1007/s004530010022

Algorithmica
© 2000 Springer-Verlag New York Inc.

Evaluating Network Reliability and 2-Edge-Connected Reliability in Linear Time for Bounded Pathwidth Graphs

C. Lucet,[1] J.-F. Manouvrier,[1] and J. Carlier[1]

Abstract. This paper presents a decomposition method for computing the 2-edge-connected reliability of undirected networks. This reliability is defined as the probability that all the vertices of a given graph G are 2-edge-connected, when edges fail independently with known probabilities. The principle of this method was introduced by Rosenthal in 1977 [1]. For the all terminal reliability problem it consists in enumerating specific state classes of some subnetworks. These classes are represented by the partitions of the boundary sets. For the 2-edge-connected reliability problem these classes are represented by labeled forests whose nodes are the partition blocks and some "unidentified" blocks. Our implementation uses a vertex linear ordering. The computational complexity depends on the number of classes, which depends on the vertex separation number of a given vertex linear ordering. Our computational results show the efficiency of this method when the vertex separation number is smaller than 7.

Key Words. Network reliability, 2-Edge-connectivity, Network decomposition, Boundary set, Pathwidth.

1. Introduction.

There is much literature devoted to the problem of all terminal reliability [2]–[5]. When considering this problem, it is supposed that the network is modeled by an undirected graph $G = (V, E)$. Vertices are perfect, but edges can fail with known probabilities. Computing the all terminal reliability means computing the probability that the network remains connected when failures are statistically independent. The aim of this paper is to revisit a decomposition method for the problem of all terminal reliability and to adapt it to 2-edge-connected reliability.

For general networks, all terminal reliability computation is NP-hard [6]. Satyanarayana and Chang [7] and Wood [8] have shown that the factoring algorithm using reductions is more efficient than the classical path or cut enumeration methods for solving it. This is confirmed by the experimental works of Theologou and Carlier [9], but its running time remains prohibitive for large networks. In 1977 Rosenthal presented a decomposition method which is a generalized reduction [1]. It consists in splitting up the network according to a boundary set of vertices and evaluating the probabilities of some subnetwork classes such that the reliability can be achieved by combining some of them. Carlier and Lucet have worked out and tested this method for K-terminal reliability computation with imperfect edges and vertices [10]. Their results show that it is more efficient than factoring using reductions and that it can be applied to real-world-size networks. In this paper we are concerned with another fundamental measure on networks:

[1] UMR CNRS 6599 HEUDIASYC, Université de Technologie de Compiègne, B.P. 20529, 60205 Compiègne cedex, France. {Corinne.Lucet, JF.Manouvrier, Jacques.Carlier}@hds.utc.fr.

Received February 14, 1997; revised February 11, 1998. Communicated by H. L. Bodlaender.

2-edge-connected reliability. This is defined as the probability that any two vertices are connected by at least two edge-disjoint paths. Implementation of the decomposition method [11] has shown that it is very efficient for computing standard K-terminal reliability, even when the vertices of G are subject to failures. We therefore propose that this decomposition method be used for 2-edge-connected reliability evaluation.

This paper is organized as follows. First, in Section 2, we describe the problem and give some definitions. Then, in Section 3, we present the decomposition method for all terminal reliability. Next, in Section 4, we show that it can be used to compute 2-edge-connected reliability by defining some appropriate subnetwork classes. Finally, in Section 5, we examine the computational complexity of the algorithm using enumeration of these classes and show that it can compute all terminal reliability and 2-edge-connected reliability in linear time for graphs with bounded pathwidth.

2. Definitions and Notation

2.1. *Notation.* $\mathbf{G} = (V, E)$ is an undirected graph, with $\mathbf{V}$ its set of vertices, and $\mathbf{E} \subset V \times V$ its set of edges. $(\mathbf{u}, \mathbf{v})$ denotes an edge of E, a nonordered pair with $u \in V$ and $v \in V$. Consequently, (u, v) and (v, u) denote the same edge. $\mathbf{p_e}$ is the reliability of the edge e, and $\mathbf{q_e} = 1 - p_e$. $\mathbf{H}$ and $\mathbf{L}$ are subgraphs of G. $\mathbf{H} \cup \mathbf{L}$ is a graph, i.e., its vertices are the vertices of H and the vertices of L, and its edges are the edges of H and the edges of L. $\mathbf{H} \cap \mathbf{L}$ is a graph, i.e., its vertices are the vertices belonging to both H and L and its edges are the edges belonging to both H and L. $\mathbf{F}$ is the boundary set of H. $[\cdots][\cdots]\cdots[\cdots]$ denotes a partition of F. $\mathcal{R}(G)$ is the all terminal reliability of G and $\mathcal{R}_{2\text{ec}}(G)$ is the all terminal 2-edge-connected reliability of G. $\mathcal{G}_i$ is a state of G and $\mathbf{G}(\mathcal{G}_i)$ its associated partial graph. $|S|$ is the cardinal of any set S.

2.2. *Definitions.* A given graph G is connected if there exists at least one path between any two vertices. It is 2-edge-connected if there exist at least two paths without common edges between any two vertices. Our network model is an undirected stochastic graph $G = (V, E)$. Each edge of E can fail, statistically independently with known probability, but vertices of V are perfectly reliable. The failure probability of the edge $e \in E$ is q_e ($q_e \in [0, 1]$) and its reliability is $p_e = 1 - q_e$. A subgraph of a given graph $G = (V, E)$ is a graph $G' = (V', E')$ such that $V' \subset V$ and $E' = (V' \times V') \cap E$. A partial graph of a given graph $G = (V, E)$ is a graph $G'' = (V, E'')$ such that $E'' \subset E$.

Each edge of the stochastic graph is subject to failure. So, as there are two states for an edge (each edge functions or fails), there are $2^{|E|}$ possible states for the graph. One state $\mathcal{G}_i$ of the stochastic graph $G = (V, E)$ is denoted $\langle s_1, s_2, \ldots, s_{|E|}\rangle$ where s_e stands for the state of edge e, i.e., $s_e = 0$ when edge e fails and $s_e = 1$ when it functions. The associated probability of $\mathcal{G}_i$ is

$$(1) \qquad \text{Prob}(\mathcal{G}_i) = \prod_{e \in E} [s_e \cdot p_e + (1 - s_e) \cdot q_e].$$

With each state $\mathcal{G}_i$ of $G = (V, E)$ is associated a partial graph $G(\mathcal{G}_i) = (V, E'')$, such that $e \in E''$ if and only if $e \in E$ and $s_e = 1$. In the following we also consider states and partial graphs of subgraphs H and L of G. $\mathcal{H}_i = \langle s'_1, s'_2, \ldots, s'_{|E'|}\rangle$ denotes one state of

$H = (V', E')$ and $H(\mathcal{H}_i)$ its associated partial graph. The reliability of $G = (V, E)$ is the probability that G supports a given operation. For the all terminal reliability problem, this operation requires that any two vertices are able to communicate via at least one operational path:

$$\mathcal{R}(G) = \sum_{G(\mathcal{G}_i) \text{ is connected}} \text{Prob}(\mathcal{G}_i). \tag{2}$$

For the all terminal 2-edge-connected reliability problem, it is necessary that any two vertices of V are able to communicate via at least two operational paths with no edges in common. So the 2-edge-connected reliability problem consists in evaluating the probability that a given graph is 2-edge-connected when its edges fail independently. This probability can be obtained by summing all the associated probabilities of states $\mathcal{G}_i$, such that $G(\mathcal{G}_i)$ is 2-edge-connected:

$$\mathcal{R}_{2\text{ec}}(G) = \sum_{G(\mathcal{G}_i) \text{ is 2-edge-connected}} \text{Prob}(\mathcal{G}_i). \tag{3}$$

3. Decomposition Principle for All Terminal Reliability

3.1. *Introduction.* Let $v \in V$ such that its removal leaves G disconnected, i.e., v is an articulation point of G. Therefore, G can be decomposed into two subgraphs H and L, such that the vertex set of $H \cap L$ is $\{v\}$, $H \cup L = G$ (Figure 1), and $\mathcal{R}(G) = \mathcal{R}(H) \cdot \mathcal{R}(L)$.

This technique was extended by Rosenthal in 1977 [1] for the all terminal reliability problem, when the vertex set of $H \cap L$ is F, a separator set of the graph, with $|F| \geq 2$. F is the boundary set of H and L, and its vertices are called the boundary vertices. It is also supposed that H and L have no edges in common.

Rosenthal proposed to associate with H and L some state classes depending on F. One class of H regroups some of its states according to an equivalence relation. For the all terminal problem, these classes can be modeled by the partitions of F. Hence, the total information about H is reduced to these boundary vertex connections.

3.2. *Operating and Failure States of a Subgraph.* Let $\mathcal{H}_i$ be a state of H and let $H(\mathcal{H}_i)$ be the partial subgraph of H composed of the surviving edges. Let F-clique be

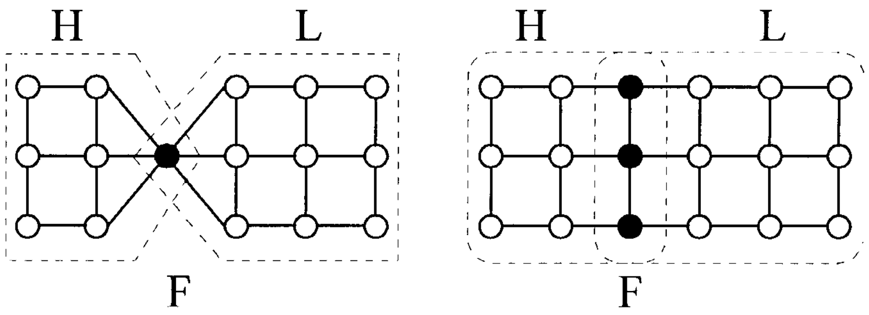


Fig. 1. Decomposition principle. $|F| = 1$ (articulation point) and $|F| = 3$.

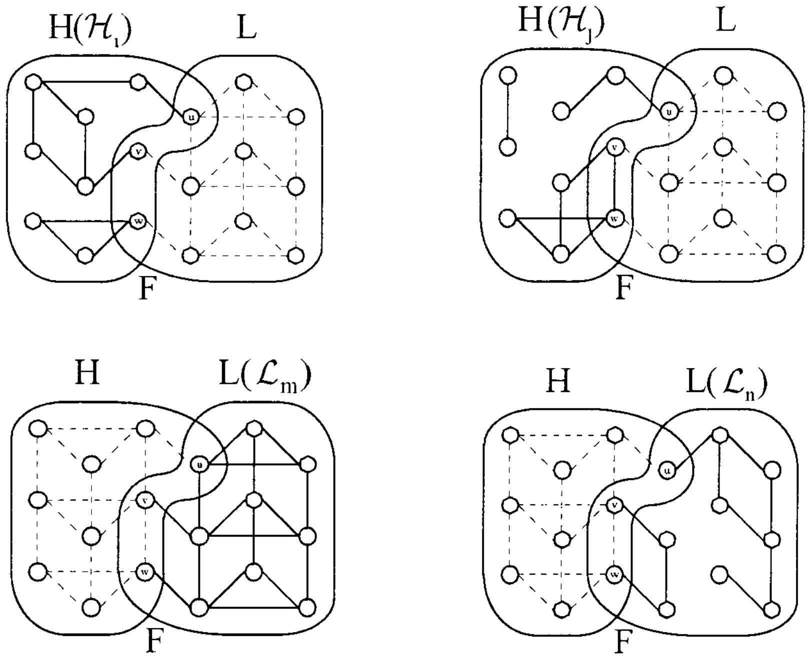


Fig. 2. State examples for all terminal reliability. Only the operating edges of a state are represented.

the completely connected subgraph $(F, F \times F)$. By definition, a state $\mathcal{H}_i$ of H is an operating state if $H(\mathcal{H}_i) \cup (F\text{-clique})$ is a connected graph; that is to say that a state $\mathcal{L}_j$ of L could exist such that $H(\mathcal{H}_i) \cup L(\mathcal{L}_j)$ is connected, i.e., $\mathcal{H}_i \cup \mathcal{L}_j$ is an operating state $\mathcal{G}_k$ of G. Otherwise, if $H(\mathcal{H}_i) \cup (F\text{-clique})$ is not a connected graph, some of the vertices of $H(\mathcal{H}_i)$ are permanently disconnected, whatever the topology of L. In this case $\mathcal{H}_i$ is called a failure state.

In Figure 2 $\mathcal{H}_i$ is an operating state of H, whereas $\mathcal{H}_j$ is a failure state. In this example, the boundary set between H and L is $F = \{u, v, w\}$:

— $H(\mathcal{H}_j)$ consists of three connected components denoted as $\mathcal{O}_{H_j,1}$, $\mathcal{O}_{H_j,2}$, and $\mathcal{O}_{H_j,3}$. One of these connected components is disconnected from the boundary set: $\mathcal{O}_{H_j,1} \cap F = \{u\}$, $\mathcal{O}_{H_j,2} \cap F = \{v, w\}$, $\mathcal{O}_{H_j,3} \cap F = \varnothing$. Therefore there is no state of L that allows the vertices of $\mathcal{O}_{H_j,3}$ to be connected to the others, not even the state $\mathcal{L}_m$, which is the state such that all edges of L function, i.e., $\mathcal{H}_j \cup \mathcal{L}_m$ is a failure state of G. So $\mathcal{H}_j$ is a failure state of H.

— $H(\mathcal{H}_i)$ contains two connected components, $\mathcal{O}_{H_i,1}$ and $\mathcal{O}_{H_i,2}$, connected with F: $\mathcal{O}_{H_i,1} \cap F = \{u, v\}$, $\mathcal{O}_{H_i,2} \cap F = \{w\}$. For an operating state such as $\mathcal{H}_i$, the network has a possibility of being connected and this possibility depends on the state of the subgraph L. If the state of the subgraph L is a failure state, the network cannot be connected. We now consider one operating state of L. $L(\mathcal{L}_n)$ is composed of two connected components $\mathcal{O}_{L_n,1}$ and $\mathcal{O}_{L_n,2}$: $\mathcal{O}_{L_n,1} \cap F = \{u\}$, $\mathcal{O}_{L_n,2} \cap F = \{v, w\}$. $H(\mathcal{H}_i) \cup L(\mathcal{L}_n)$ is connected and therefore $\mathcal{H}_i \cup \mathcal{L}_n$ is an operating state of G.

The formula for the reliability (2) of our graph can be written:

$$\mathcal{R}(G) = \sum_{\mathcal{H}_i,\mathcal{L}_j / H(\mathcal{H}_i)\cup L(\mathcal{L}_j) \text{ is connected}} \text{Prob}(\mathcal{H}_i)\cdot\text{Prob}(\mathcal{L}_j). \tag{4}$$

PROPOSITION 3.1. *The two following statements are equivalent*:

Statement 1. *$H(\mathcal{H}_i) \cup L(\mathcal{L}_j)$ is connected.*
Statement 2. *$\mathcal{H}_i$ and $\mathcal{L}_j$ are both operating states*
and
$(F \cap \mathcal{O}_{H_i,1})$-clique $\cup \cdots \cup$ $(F \cap \mathcal{O}_{H_i,s})$-clique $\cup$ $(F \cap \mathcal{O}_{L_j,1})$-clique $\cup \cdots \cup$ $(F \cap \mathcal{O}_{L_j,t})$-clique is connected.

Proposition 3.1 entails that the information required for providing the functioning state of G is, for two operating states, the manner of connection of the boundary vertices, i.e., their connections via H and via L.

3.3. *Class Definition and Equivalence Relation.* We have seen above that the functioning of G depends on the manner of connection of the boundary vertices, via H and via L. Now, certain operating states of a subgraph give the same connectivity for the boundary vertices via H. Such states are called equivalent states and are grouped in the same class. In Figure 3 the two operating states $\mathcal{H}_i$ and $\mathcal{H}_k$ are equivalent states of the subgraph H, because they provide an identical connectivity of the boundary vertices.

Rosenthal has grouped all the failure states of H into a failure class, denoted DEF(H), and all the operating states $\mathcal{H}_i$ of H into operating classes, according to the manner of connection of the boundary vertices, via the partial subgraph $H(\mathcal{H}_i)$. In the case of all terminal reliability, these operating classes are the partitions of F. One partition is made of several blocks. Each block stands for the intersection between one connected component of $H(\mathcal{H}_i)$ and F, i.e., one block contains vertices of F that belong to the same connected component.

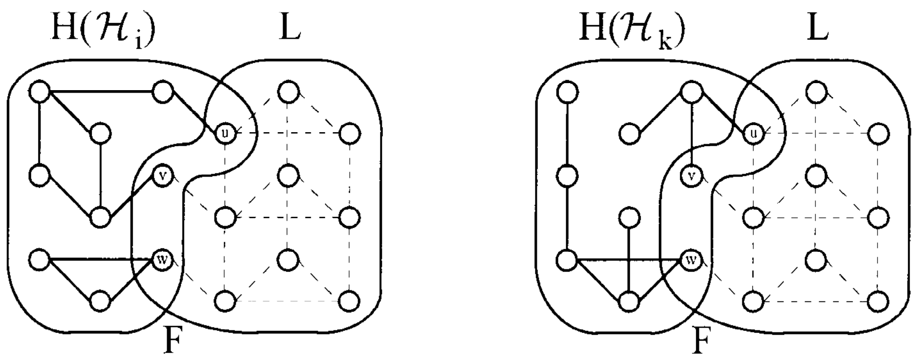


Fig. 3. Two equivalent states for all terminal reliability. $\mathcal{H}_i$ and $\mathcal{H}_k$ are two operating states of H that belong to the same class: $[uv][w]$.

We denote $C_{H,k}$ as the kth class of H. For instance, the classes of H for a boundary set of three vertices u, v, and w can be the following:

$C_{H,1} = [uvw]$,	u, v, and w are connected via H.
$C_{H,2} = [uv][w]$,	u and v are both connected via H, and w is disconnected.
$C_{H,3} = [uw][v]$,	u and w are both connected via H, and v is disconnected.
$C_{H,4} = [u][vw]$,	v and w are both connected via H, and u is disconnected.
$C_{H,5} = [u][v][w]$,	u, v, and w are disconnected via H.

The example in Figure 3 shows two equivalent states that belong to the class $[uv][w]$. The class $C_{H,2}$ is a factorization of all states $\mathcal{H}_i$ composed of two connected components $\mathcal{O}_{H_i,1}$ and $\mathcal{O}_{H_i,2}$ with $\mathcal{O}_{H_i,1} \cap F = \{u, v\}$ and $\mathcal{O}_{H_i,2} \cap F = \{w\}$.

3.4. *The Decomposition Principle.* The decomposition algorithm consists in enumerating the operating classes of H and L (omitting the failure classes DEF(H) and DEF(L)), and in computing their associated probabilities. The associated probability of the class $C_{H,k}$ is

$$\text{Prob}(C_{H,k}) = \sum_{\mathcal{H}_i / \mathcal{H}_i \in C_{H,k}} \text{Prob}(\mathcal{H}_i). \tag{5}$$

The reliability of G is computed by combining the compatible classes of H and L. Two classes $C_{H,x}$ and $C_{L,y}$ are compatible if the connectivity of the boundary set given by $C_{H,x}$ and the connectivity of the boundary set given by $C_{L,y}$ provide the connectivity of the whole graph G.

We report here some of the possible combinations between the classes of two subgraphs H and L, separated by a boundary set of three vertices, and we specify the compatible classes. The set of possible classes is the set of partitions, that is $\{[uvw], [uv][w], [uw][v], [u][vw], [u][v][w]\}$.

$C_{H,1} = [uvw]$	and	$C_{L,5} = [u][v][w]$	are compatible classes.
$C_{H,2} = [uv][w]$	and	$C_{L,5} = [u][v][w]$	are not compatible classes.
$C_{H,3} = [uw][v]$	and	$C_{L,3} = [uw][v]$	are not compatible classes.
$C_{H,3} = [uw][v]$	and	$C_{L,4} = [u][vw]$	are compatible classes.

So there is a factorization of (4) that becomes

$$\mathcal{R}(G) = \sum_{C_{H,x}, C_{L,y} / C_{H,x} \text{ and } C_{L,y} \text{ are compatible}} \text{Prob}(C_{H,x}) \cdot \text{Prob}(C_{L,y}), \tag{6}$$

which is in fact

$$\mathcal{R}(G) = \sum_{C_{H,x}, C_{L,y}, \text{ compatible}} \left[\sum_{\mathcal{H}_i \in C_{H,x}} \text{Prob}(\mathcal{H}_i) \cdot \sum_{\mathcal{L}_j \in C_{L,y}} \text{Prob}(\mathcal{L}_j) \right].$$

Formula (6) is more efficient than (4), because it reduces the number of multiplications.

Table 1. The total number of partitions according to the size of F.

$\|F\|$	1	2	3	4	5	6	7	8	9	10
Nb_classes	1	2	5	15	52	203	877	4140	21,147	115,975

Rosenthal's algorithm [1] uses the recurrence formula:

$$(7)\quad \text{Prob}(C_{H3,z}) = \sum_{\substack{C_{H1,x},\, C_{H2,y}/C_{H1,x} \text{ and } C_{H2,y} \\ \text{provide the connectivity of } C_{H3,z}}} \text{Prob}(C_{H1,x}) \cdot \text{Prob}(C_{H2,y})$$

$$\text{with} \quad H3 = H1 \cup H2$$

Algorithm [1]

By definition, a subgraph is resolved if the probabilities of all its classes have been computed. Repeat 1 and 2 until $H3 = G$.

1. Choose two resolved subgraphs denoted as $H1$ and $H2$ such that $H3 = H1 \cup H2$.
2. Use (7) to resolve the subgraph $H3$.

3.5. *Number of Classes*. The subgraphs H and L have an equal number of classes, which is the number of partitions of F. We denote the Stirling number of the second kind by $A_{i,j}$, which is the number of partitions with j blocks for a set of i elements. This number grows exponentially with i, consequently the number of classes grows exponentially with the size of the boundary set F (see Table 1). We have the recurrent formulae:

$$A_{i,j} = 1 \quad \text{if} \quad j = 1, \qquad A_{i,j} = 0 \quad \text{if} \quad 0 < i < j,$$

$$A_{i,j} = j \cdot A_{i-1,j} + A_{i-1,j-1} \quad \text{if} \quad 1 < j \leq i.$$

$$\text{Nb\_classes} = \sum_{j=1}^{|F|} A_{|F|,j}.$$

The 2-edge-connected reliability can be computed in a similar way, but in this case the classes are not simply partitions of the boundary set F. We describe these classes in the following section.

4. Classes for All Terminal 2-Edge-Connected Reliability

4.1. *Introduction*. Our aim now is to use the same decomposition principle for 2-edge-connected (2ec) reliability. Decomposition with an articulation point gives a similar formula to that in Section 3.1: $\mathcal{R}_{2ec}(G) = \mathcal{R}_{2ec}(H) \cdot \mathcal{R}_{2ec}(L)$. In this section we extend this principle for a set F of boundary vertices between two subgraphs H and L ($H \cup L = G$) with $|F| \geq 2$. H and L are assumed to have no edges in common (Figure 1).

The main difficulty here is to define appropriate classes to stand for the sets of equivalent states of a subgraph. Previously, the information contained in a class for reliability computation was the connectivity of the boundary vertices via H. Now the information required to specify a class for 2ec reliability computation will be the connectivity and the 2-edge-connectivity of the boundary vertices via H.

4.2. *Operating and Failure States of a Subgraph.* To define operating and failure states, we introduce the notion of a 2-edge-connected-clique. By definition, a subgraph $G' = (V', E')$ of a graph $G = (V, E)$ is a 2-edge-connected-clique (2ec-clique) if and only if $|V'| \neq 2$ and G' is a clique, or $|V'| = 2$ and G' is the multigraph $G' = (\{u, v\}, \{(u, v), (u, v)\})$. If a subgraph $G' = (V', E')$ of a graph $G = (V, E)$ is a 2ec-clique, then there exist two edge-disjoint paths between any two vertices of V'.

With regard to 2ec reliability, a state $\mathcal{H}_i$ is by definition an operating state if $H(\mathcal{H}_i) \cup (F\text{-2ec-clique})$ is 2-edge-connected. So a failure state $\mathcal{H}_j$ is a state which does not permit the whole graph to be 2-edge-connected, whatever the state of L. Figure 4 summarizes the conditions for operating and nonoperating states for the two problems under discussion.

We again consider the examples of Figure 2. The state $\mathcal{L}_n$ is a failure state for 2-edge-connectivity although it is an operating state for the connectivity. As we have shown in Section 3.2, a state $\mathcal{G}_k$ of G is composed of two states in the two subgraphs H and

Reliability :

Operating states	Failure states
$H(\mathcal{H}_i) \cup$ (F-clique) is connected $\equiv$ $\forall v \in (H(\mathcal{H}_i) - F)$, $\exists$ a path from v to F.	$H(\mathcal{H}_i) \cup$ (F-clique) is disconnected $\equiv$ $\exists v \in (H(\mathcal{H}_i) - F)$, $\nexists$ a path from v to F.

2-edge-connected reliability :

Operating states	Failure states
$H(\mathcal{H}_i) \cup$ (F-2ec-clique) is 2-edge-connected $\equiv$ $\forall v \in (H(\mathcal{H}_i) - F)$, $\exists$ two edge-disjoint paths from v to F.	$H(\mathcal{H}_i) \cup$ (F-2ec-clique) is not 2-edge-connected $\equiv$ $\exists v \in (H(\mathcal{H}_i) - F)$, $\nexists$ two edge-disjoint paths from v to F.

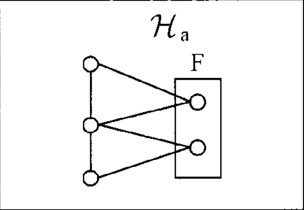
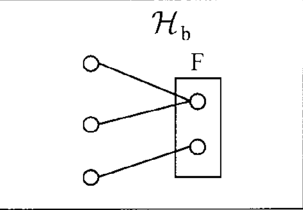
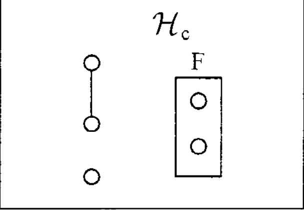
		

Fig. 4. Operating and failure states.

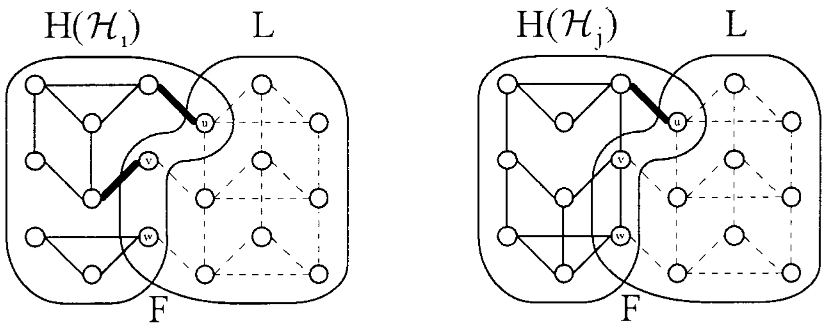


Fig. 5. State examples for 2ec reliability.

L: $\mathcal{H}_i \cup \mathcal{L}_j = \mathcal{G}_k$. So the formula for the 2ec reliability of our graph can be written:

$$(8) \qquad \mathcal{R}_{2\text{ec}}(G) = \sum_{\mathcal{H}_i, \mathcal{L}_j / H(\mathcal{H}_i) \cup L(\mathcal{L}_j) \text{ is 2-edge-connected}} \text{Prob}(\mathcal{H}_i) \cdot \text{Prob}(\mathcal{L}_j)$$

The subgraph $H(\mathcal{H}_i)$ is composed of the set of the 2ec components $\Phi(\mathcal{H}_i)$ and the set of the cut-edges $\Delta(\mathcal{H}_i)$ that join these 2ec components, $\Delta(\mathcal{H}_i) \subset \Phi(\mathcal{H}_i) \times \Phi(\mathcal{H}_i)$, i.e., there exists an edge between two 2ec components $\mathcal{T}_1$ and $\mathcal{T}_2$ in $\Delta(\mathcal{H}_i)$ if and only if there exists a cut-edge between $u \in \mathcal{T}_1$ and $v \in \mathcal{T}_2$ in $H(\mathcal{H}_i)$. $(\Phi(\mathcal{H}_i), \Delta(\mathcal{H}_i))$ is a forest. We distinguish two kinds of 2ec components in $\Phi(\mathcal{H}_i) = \Phi 1(\mathcal{H}_i) \cup \Phi 2(\mathcal{H}_i)$: $\Phi 1(\mathcal{H}_i)$ denotes the set of 2ec components of $H(\mathcal{H}_i)$ that contain at least one vertex of F (i.e., the boundary 2ec components), and $\Phi 2(\mathcal{H}_i)$ denotes the set of 2ec components of $H(\mathcal{H}_i)$ that have no intersection with F.

To know if the result of $H(\mathcal{H}_i) \cup L(\mathcal{L}_j)$ is 2-edge-connected, we have to consider $\Phi(\mathcal{H}_i)$, $\Phi(\mathcal{L}_j)$, $\Delta(\mathcal{H}_i)$, and $\Delta(\mathcal{L}_j)$.

We illustrate using the examples shown in Figure 5. The boundary set between H and L is $F = \{u, v, w\}$.

— $H(\mathcal{H}_j)$ contains two 2ec components: $\Phi(\mathcal{H}_j) = \Phi 1(\mathcal{H}_j) = \{\mathcal{T}_{H_j,1}, \mathcal{T}_{H_j,2}\}(\mathcal{T}_{H_j,1} \cap F = \{u\}, \mathcal{T}_{H_j,2} \cap F = \{v, w\})$, and the set $\Delta(\mathcal{H}_j)$ contains a single cut-edge between these two components. All the vertices of H are connected to F by at least two paths, which is the condition for the state $\mathcal{H}_j$ to be operating.

— $H(\mathcal{H}_i)$ contains four 2ec components: $\Phi 1(\mathcal{H}_i) = \{\mathcal{T}_{H_i,1}, \mathcal{T}_{H_i,2}, \mathcal{T}_{H_i,3}\}$, $\Phi 2(\mathcal{H}_i) = \{\mathcal{T}_{H_i,4}\}(\mathcal{T}_{H_i,1} \cap F = \{u\}, \mathcal{T}_{H_i,2} \cap F = \{v\}, \mathcal{T}_{H_i,3} \cap F = \{w\}, \mathcal{T}_{H_i,4} \cap F = \varnothing)$, and two cut-edges in $\Delta(\mathcal{H}_i)$ (between $\mathcal{T}_{H_i,1}$ and $\mathcal{T}_{H_i,4}$, and between $\mathcal{T}_{H_i,2}$ and $\mathcal{T}_{H_i,4}$). All of the 2ec components are connected to F by at least two paths (so it is an operating state), nevertheless, the intersection between F and a component could be empty ($\mathcal{T}_{H_i,4}$).

PROPOSITION 4.1. *The two following statements are equivalent*:

Statement 1. *$H(\mathcal{H}_i) \cup L(\mathcal{L}_j)$ is 2-edge-connected.*

Statement 2. *$\mathcal{H}_i$ and $\mathcal{L}_j$ are both operating states and the partial graph*

$$\bigcup_{\mathcal{T}_{H_i,x}\in\Phi1(\mathcal{H}_i)} \{(F\cap\mathcal{T}_{H_i,x})\text{-}2ec\text{-}clique\} \cup \bigcup_{\mathcal{T}_{L_j,y}\in\Phi1(\mathcal{L}_j)} \{(F\cap\mathcal{T}_{L_j,y})\text{-}2ec\text{-}clique\}\cup\Delta'(\mathcal{H}_i)\cup\Delta'(\mathcal{L}_j)$$

is 2-edge-connected, where $\Delta'(\mathcal{H}_i)$ is a transformation of $\Delta(\mathcal{H}_i)$ to make possible this union, i.e., if an endpoint $\mathcal{T}_{H_i,x}$ of an edge of $\Delta(\mathcal{H}_i)$ belongs to $\Phi1(\mathcal{H}_i)$, then this endpoint is replaced by a vertex of $F\cap\mathcal{T}_{H_i,x}$, hence $\Delta'(\mathcal{H}_i)\subset(F\cup\Phi2(\mathcal{H}_i))\times(F\cup\Phi2(\mathcal{H}_i))$.

Proposition 4.1 leads to the following remark. As in the case of reliability, it is not necessary to know the identity of a vertex of $H(\mathcal{H}_i)$ out of the boundary set in order to deduce that $H(\mathcal{H}_i)\cup L(\mathcal{L}_j)$ is 2-edge-connected, because these vertices cannot be linked directly to a vertex of L. Hence, the information required for an operating state $\mathcal{H}_i$ (to calculate (8)) is first the connection of the boundary vertices in 2ec components, and secondly the connections between these 2ec components. These paths, which can go through 2ec components of $\Phi2(\mathcal{H}_i)$, are represented by the set of cut-edges $\Delta(\mathcal{H}_i)$. This information must be used to represent a class, but we reduce it below.

4.3. *Class Definition and Equivalence Relation*

4.3.1. *Introduction.* As in the case of reliability, the function of the 2ec reliability classes is to group all equivalent operating states, in order to factorize (8) with (5):

$$(9)\qquad \mathcal{R}_{2\mathrm{ec}}(G)=\sum_{C_{H,x},C_{L,y}/C_{H,x}\text{ and }C_{L,y}\text{ are compatible}}\mathrm{Prob}(C_{H,x})\cdot\mathrm{Prob}(C_{L,y}).$$

We also have here the failure classes DEF(H) and DEF(L) which group all failure states.

The information provided by a class must be just the information required to combine and find the compatible classes. Indeed, the more restricted the information standing for a class, the greater the number of equivalent states in this same class, and consequently the greater the efficiency of (9). We study in this section the information provided by a state to obtain the required information to define a class.

4.3.2. *State Representation.* The state $\mathcal{H}_i$ can be represented as a forest of 2ec components: $\mathcal{F}(\mathcal{H}_i)=(\Phi1(\mathcal{H}_i)\cup\Phi2(\mathcal{H}_i),\Delta(\mathcal{H}_i))$ where $\Phi1(\mathcal{H}_i)$ and $\Phi2(\mathcal{H}_i)$ are the sets of 2ec components defined in Section 4.2. We now reduce the knowledge provided by the forest $\mathcal{F}(\mathcal{H}_i)$ to obtain a state representation denoted as $\zeta(\mathcal{H}_i)$. We saw with Proposition 4.1, that knowing the identities of the vertices outside of the boundary set is superflous. So the state representation is a forest: $\zeta(\mathcal{H}_i)=(\Theta1(\mathcal{H}_i)\cup\Theta2(\mathcal{H}_i),\Psi(\mathcal{H}_i))$ where $\Theta1(\mathcal{H}_i)$, $\Theta2(\mathcal{H}_i)$, and $\Psi(\mathcal{H}_i)$ correspond respectively to the reductions of $\Phi1(\mathcal{H}_i)$, $\Phi2(\mathcal{H}_i)$, and $\Delta(\mathcal{H}_i)$. $\Theta1(\mathcal{H}_i)$ is a set of blocks of the boundary set F, that is to say a partition of F. $\Theta2(\mathcal{H}_i)$ is a set of empty blocks that we call unidentified blocks and $\Psi(\mathcal{H}_i)$ is the set of edges joining these blocks (see Figure 6).

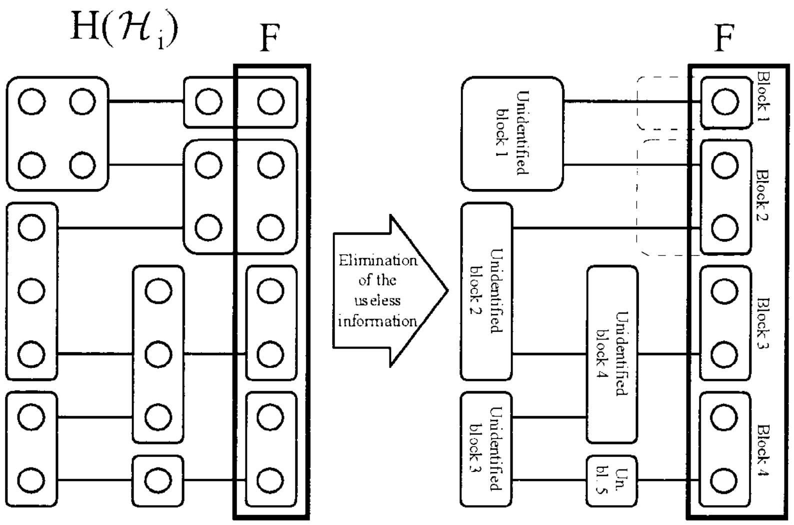


Fig. 6. State representation for 2ec reliability. $H(\mathcal{H}_i)$ is a forest (here a tree) of nine 2ec components.

We use this representation for the examples of Figure 5:

$$\begin{aligned}
\zeta(\mathcal{H}_i) &= (\Theta1(\mathcal{H}_i) \cup \Theta2(\mathcal{H}_i), \Psi(\mathcal{H}_i)) \\
&\quad \text{with} \quad \Theta1(\mathcal{H}_i) = \{B_1, B_2, B_3\}, \quad \Theta2(\mathcal{H}_i) = \{B_{-1}\}, \\
&\quad\quad \Psi(\mathcal{H}_i) = \{(B_1, B_{-1}), (B_2, B_{-1})\}, \\
&\quad \text{and} \quad B_1 = [u], \quad B_2 = [v], \quad B_3 = [w], \quad B_{-1} = [\,]; \\
\zeta(\mathcal{H}_j) &= (\Theta1(\mathcal{H}_j) \cup \Theta2(\mathcal{H}_j), \Psi(\mathcal{H}_j)) \\
&\quad \text{with} \quad \Theta1(\mathcal{H}_j) = \{B_1, B_2\}, \quad \Theta2(\mathcal{H}_j) = \{\,\}, \quad \Psi(\mathcal{H}_j) = \{(B_1, B_2)\}, \\
&\quad \text{and} \quad B_1 = [u], \quad B_2 = [vw].
\end{aligned}$$

With such a state representation ($\zeta(\mathcal{H}_i)$), the composition of the 2ec components of $\Phi2(\mathcal{H}_i)$, represented by the unidentified blocks of $\Theta2(\mathcal{H}_i)$, is totally unknown (whereas the composition of a boundary 2ec component is partially known with $\Theta1(\mathcal{H}_i)$). This notion of unidentified blocks allows us to represent all the required connection manners of the boundary blocks of the partition $\Theta1(\mathcal{H}_i)$. The single condition for an unidentified block to exist is the presence of edges in $\Psi(\mathcal{H}_j)$ incident to this unidentified block, which represents useful block path information. In the following, we eliminate some unidentified blocks without deleting this essential information.

4.3.3. *From State Representation to Minimal State Representation.* We now remove the useless information of $\zeta(\mathcal{H}_i)$ and obtain the minimal state representation denoted as $\zeta'(\mathcal{H}_i)$, i.e., the bare necessities. A simplification can be made for $\Theta2(\mathcal{H}_i)$. Indeed, if there is an unidentified block $h \in \Theta2(\mathcal{H}_i)$ which is adjacent to only two other blocks, $(h, k_1) \in \Psi(\mathcal{H}_i)$, $(h, k_2) \in \Psi(\mathcal{H}_i)$, then we need retain no information other than the path between k_1 and k_2, i.e., (k_1, k_2) can replace (h, k_1) and (h, k_2) in $\Psi(\mathcal{H}_i)$ enabling

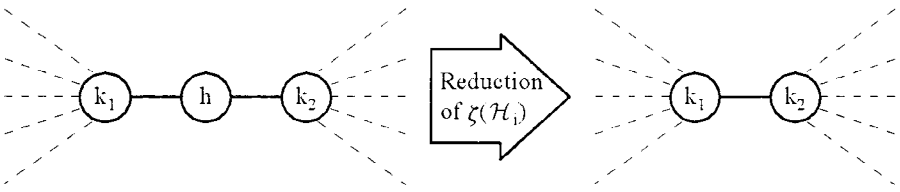


Fig. 7. From state representation to minimal state representation. *h* is an unidentified block of degree two.

the unidentified block *h* to be removed from $\Theta2(\mathcal{H}_i)$ (Figure 7). This is a reduction of $\Theta2(\mathcal{H}_i)$ to $\Theta2'(\mathcal{H}_i)$, and a transformation from $\Psi(\mathcal{H}_i)$ to $\Psi'(\mathcal{H}_i)$.

After the deletion of all the unidentified blocks of degree two in $\Psi(\mathcal{H}_i)$ and $\Theta2(\mathcal{H}_i)$, we obtain $\Psi'(\mathcal{H}_i)$ and $\Theta2'(\mathcal{H}_i)$, and no further simplification of the information for a state is possible (see Figure 8). The minimal representation of a state is composed of a forest whose nodes are blocks of a partition and unidentified blocks: $\zeta'(\mathcal{H}_i) = (\Theta1(\mathcal{H}_i) \cup \Theta2'(\mathcal{H}_i), \Psi'(\mathcal{H}_i))$, such that the unidentified blocks have a degree strictly larger than two. Indeed it must be at least two in order to have an operating state, and it cannot be two because of the reduction from $\Psi(\mathcal{H}_i)$ to $\Psi'(\mathcal{H}_i)$.

PROPERTY 4.2. For any unidentified block *b* of $\Theta2'(\mathcal{H}_i)$, the degree of *b* in the forest $\zeta'(\mathcal{H}_i) = (\Theta1(\mathcal{H}_i) \cup \Theta2'(\mathcal{H}_i), \Psi'(\mathcal{H}_i))$ is strictly larger than two.

Consequently, we have $0 \leq |\Theta2'(\mathcal{H}_i)| \leq \max(0, |\Theta1(\mathcal{H}_i)| - 2)$.

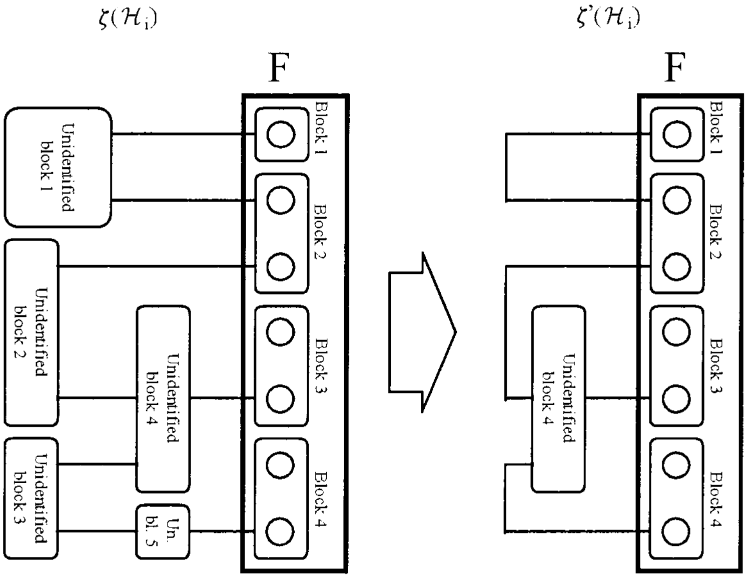


Fig. 8. From state representation to minimal state representation (example). Removal of the unidentified blocks of degree two.

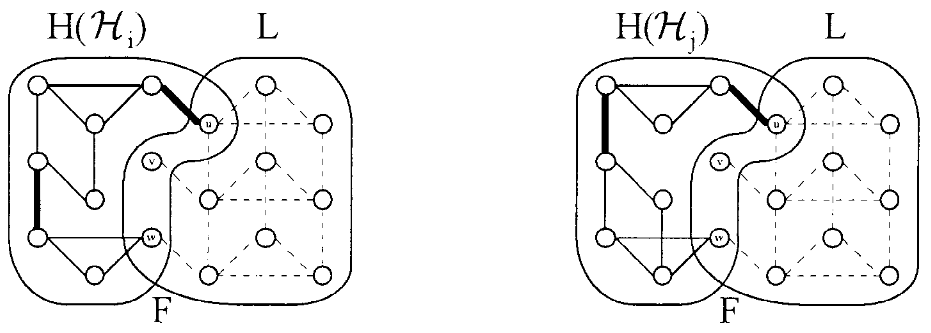


Fig. 9. Equivalent states for 2ec reliability. $\zeta(\mathcal{H}_i) = (\Theta 1(\mathcal{H}_i) \cup \Theta 2(\mathcal{H}_i), \Psi(\mathcal{H}_i)) = (\{B_1, B_2, B_3, B_{-1}\}, \{(B_1, B_{-1}), (B_3, B_{-1})\})$ with $B_1 = [u]$, $B_2 = [v]$, $B_3 = [w]$, $B_{-1} = []$. $\zeta(\mathcal{H}_j) = (\Theta 1(\mathcal{H}_j) \cup \Theta 2(\mathcal{H}_j), \Psi(\mathcal{H}_j)) = \zeta(\mathcal{H}_i) \Rightarrow \mathcal{H}_i$ and $\mathcal{H}_j$ are equivalent states.

4.3.4. *Class Representation.* Several states have the same minimal representation. These states are equivalent and belong to the same class (see Figure 9). The modeling of a class $C_{H,k}$ is therefore a forest $C_{H,k} = (\Theta 1(C_{H,k}) \cup \Theta 2'(C_{H,k}), \Psi'(C_{H,k}))$, where:

— $\Theta 1(C_{H,k})$ is a partition of the boundary set F of $|\Theta 1(C_{H,k})|$ blocks.
— $\Theta 2'(C_{H,k})$ is a set of unidentified blocks which satisfy Property 4.2.
— $\Psi'(C_{H,k})$ is a set of edges whose endpoints belong to $\Theta 1(C_{H,k}) \cup \Theta 2'(C_{H,k})$.

All the states whose minimal representation is $C_{H,k}$ belong to this class.

4.4. *The Decomposition Principle.* The principle of decomposition for 2ec reliability is similar to that for all terminal reliability:

— consider two subgraphs H and L and their boundary set F,
— enumerate the operating classes of H and L (omitting the failure classes DEF(H) and DEF(L)),
— compute the associated probabilities of these classes (with (5)),
— search for all compatible classes and compute the 2ec reliability with (9).

2ec reliability of G is computed by combining the compatible classes of H and L (9). Two classes $C_{H,x}$ and $C_{L,y}$ are compatible if their union provides the 2-edge-connectivity of the boundary set F, and can ensure the 2-edge-connectivity of the whole graph G.

Figure 10 presents the set of possible classes for a boundary set of three vertices. Note: the blocks of the partitions are denoted B_i with $i > 0$ and the unidentified blocks are denoted B_j with $j < 0$. We list here some combinations between the classes of the two subgraphs H and L, separated by a boundary set of three vertices, that are compatible

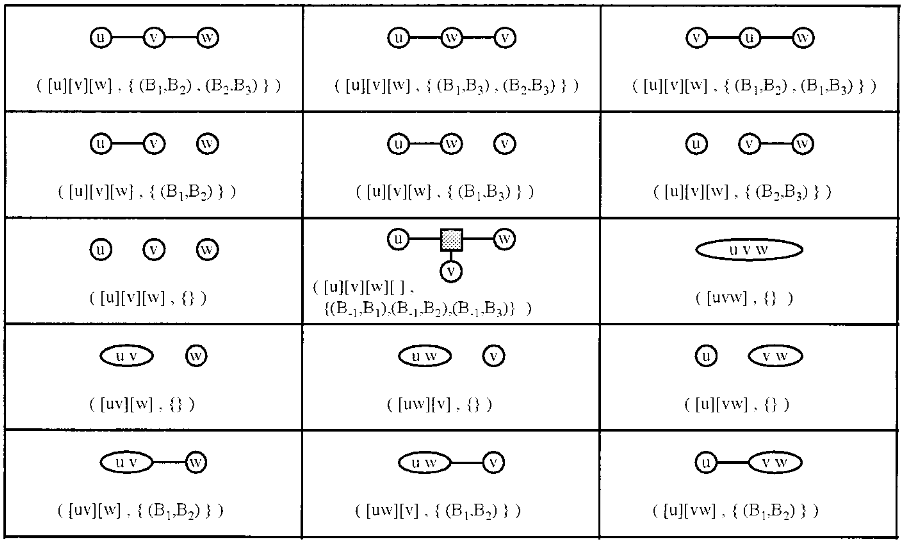

([u][v][w] , { (B1,B2) , (B2,B3) })
([u][v][w] , { (B1,B3) , (B2,B3) })
([u][v][w] , { (B1,B2) , (B1,B3) })
([u][v][w] , { (B1,B2) })
([u][v][w] , { (B1,B3) })
([u][v][w] , { (B2,B3) })
([u][v][w] , {})
([u][v][w][ ] , {(B-1,B1),(B-1,B2),(B-1,B3)})
([uvw] , {})
([uv][w] , {})
([uw][v] , {})
([u][vw] , {})
([uv][w] , { (B1,B2) })
([uw][v] , { (B1,B2) })
([u][vw] , { (B1,B2) })

Fig. 10. The 15 possible classes for a boundary set of three vertices.

for 2ec reliability:

$$\begin{array}{llll}
C_{H,1} &= ([uvw], \{\}) & \text{and } C_{L,i}, \forall i, 1 \le i \le 15. \\
C_{H,3} &= ([uv][w], \{(1,2)\}) & \text{and } C_{L,3} &= ([uv][w], \{(1,2)\}). \\
C_{H,5} &= ([uw][v], \{(1,2)\}) & \text{and } C_{L,3} &= ([uv][w], \{(1,2)\}). \\
C_{H,5} &= ([uw][v], \{(1,2)\}) & \text{and } C_{L,6} &= ([u][vw], \{\}). \\
C_{H,6} &= ([u][vw], \{\}) & \text{and } C_{L,1} &= ([uvw], \{\}). \\
C_{H,8} &= ([u][v][w], \{\}) & \text{and } C_{L,1} &= ([uvw], \{\}). \\
C_{H,12} &= ([u][v][w], \{(1,2), (1,3\}) & \text{and } C_{L,14} &= ([u][v][w], \{(1,3), (2,3\}). \\
C_{H,12} &= ([u][v][w], \{(1,2), (1,3\}) & \text{and } C_{L,15} &= ([u][v][w][], \{(-1,1), (-1,2), (-1,3)\}). \\
C_{H,15} &= ([u][v][w][]\{(-1,1), (-1,2), (-1,3)\}), & \text{and } C_{L,15} &= ([u][v][w][], \{(-1,1), (-1,2), (-1,3)\}).
\end{array}$$

Rosenthal’s algorithm described in Section 3.4 for all terminal reliability can be applied for 2ec reliability without any change.

4.5. *Number of Classes.* The number of classes depends on the number of vertices in the boundary set: $|F|$ (e.g., the class enumeration for $|F| = 3$ is shown in Figure 10). A class is composed of a partition of F with k blocks and of a forest on these blocks, and possibly of unidentified blocks of degree strictly greater than two (see Property 4.2). So the number of classes for a given boundary set F is

$$\text{Nb\_classes}_{2\text{ec}} = \sum_{j=1}^{|F|} (A_{|F|,j} \cdot \text{NFU}(j)),$$

where $A_{|F|,j}$ is the Stirling number of the second kind (see Section 3.5) and NFU(j) stands for the number of these particular labeled forests of j nodes with unidentified nodes (these unidentified nodes are not labeled and have a degree strictly greater than two). These numbers are given in Tables 2 and 3.

Table 2. Number of labeled forests with unidentified nodes against the number of nodes N.

N	1	2	3	4	5	6	7
NFU(N)	1	2	8	58	662	10,584	219,004

5. Linear Time Algorithms. We have seen above the decomposition principle which consists in considering two subgraphs H and L of G and combining their classes to compute reliability (see Sections 3 and 4). Now we look at the implementation of this principle with an algorithm belonging to the table-based reduction algorithm family [12], more effective than Rosenthal's algorithm which we presented in Section 3.4.

5.1. *Preliminary Definitions.* We first define the notions of linear ordering and vertex separation number used in our algorithm, and the notions of pathwidth and treewidth introduced by Robertson and Seymour [13], [14] and Bodlaender [15].

By definition, a **linear ordering** (denoted $\mathcal{N}$) of $G = (V, E)$ is a bijection: $\mathcal{N}\colon V \to \{1, \dots, |V|\}$.

In the following, the vertex $\mathcal{N}^{-1}(i)$, $i \in \{1, \dots, |V|\}$, will be denoted as v_i.

We denote $F_i = \{v_j \in V / \exists (v_j, v_k) \in E$ such that $j \le i \le k\}$, $\forall i \in \{1, \dots, |V|\}$.

The **vertex separation number** (denoted $F_{\max}$) of a linear ordering is $F_{\max}(\mathcal{N}) = \max_{1 \le i \le |V|}(|Fi|)$.

The vertex separation number of a graph is $F_{\max}(G) = \min_{\mathcal{N}}(F_{\max}(\mathcal{N}))$.

A **path-decomposition** (denoted as $\mathcal{D}_p$) of $G = (V, E)$ is a sequence of subsets of V: $\mathcal{D}_p = (X_1, \dots, X_r)$, such that:

— $\cup_{1 \le i \le r} X_i = V$,
— for every edge $(v, w) \in E$, there is a subset X_i, $1 \le i \le r$, with $v \in X_i$ and $w \in X_i$,
— for all $i, j, k \in \{1, \dots, r\}$, if $i \le j \le k$, then $X_i \cap X_k \subseteq X_j$.

By definition, the **pathwidth** of a path-decomposition is pathwidth($\mathcal{D}_p$) $= \max_{1 \le i \le r}(|X_i| - 1)$, and the pathwidth of a graph is pathwidth(G) $= \min_{\mathcal{D}_p}$(pathwidth($\mathcal{D}_p$)).

A **tree-decomposition** (denoted as $\mathcal{D}_t$) of $G = (V, E)$ is a couple composed of a family of subsets of V and a tree: $\mathcal{D}_t = (\{Xi / i \in I\}, T = (I, F))$, such that:

— $\bigcup_{i \in I} X_i = V$,
— for every edge $(v, w) \in E$, there is a subset X_i, $i \in I$, with $v \in X_i$ and $w \in X_i$,
— for all $i, j, k \in I$, if j is on the path from i to k in T, then $X_i \cap X_k \subseteq X_j$.

Table 3. Total number of classes (2ec reliability) against the size of F.

$\|F\|$	1	2	3	4	5	6	7
Nb_classes$_{2ec}$	1	3	15	121	1473	25,067	556,783

By definition, the **treewidth** of a tree-decomposition is treewidth$(\mathcal{D}_t) = \max_{i \in I}(|X_i| - 1)$, and the treewidth of a graph is treewidth$(G) = \min_{\mathcal{D}_t}(\text{treewidth}(\mathcal{D}_t))$.

The following propositions have been demonstrated: Let $G = (V, E)$ be a given graph. The pathwidth of a graph G is at least its treewidth. In fact, a path-decomposition of G can be written as a tree-decomposition of G, with the same width. The pathwidth of a graph G is equal to its vertex separation number [16]. Moreover, finding a linear ordering of G with a minimum vertex separation number is equivalent to finding a path-decomposition with the smallest pathwidth.

The pathwidth and treewidth problems (given a graph, find a tree-decomposition or a path-decomposition with the smallest width) are NP-hard [17]. Nevertheless, for fixed parameters k, the problems of finding a path-decomposition or tree-decomposition of width at most k can be solved in linear time, but using algorithms with a rather high constant factor [18], [19].

5.2. *The Dynamic Programming Algorithm for Reliability and 2ec Reliability Computations.* We now present our algorithm based on the decomposition principle (see Sections 3 and 4). We consider a resolved subgraph H whose classes are known, and we enlarge this resolved subgraph by vertex insertion until we have resolved the whole graph.

0. A linear ordering of the vertices is required.
1. $H_0 = \{\}$, $L_0 = G$.
2. For each vertex v_i (in the order given by the linear ordering, from v_1 to $v_{|V|}$):
   - 2.1. Remove v_i from the subgraph L_{i-1} and add v_i to the subgraph H_{i-1} to obtain L_i and H_i.
   - 2.2. Find the boundary set F_i between H_i and L_i.
   - 2.3. Compute the classes of the boundary set F_i in the subgraph H_i and their probabilities.
3. The reliability (or 2ec reliability) of G is the probability of the single class of the last boundary set.

The vertex separation number, $F_{\max}$, is the size of the boundary set F_i that contains the maximal number of vertices during the algorithm.

Figure 11 shows the boundary set evolution and the growth of the resolved subgraph (H_i) during the algorithm. At each step of the algorithm, the classes and their probabilities are stored in a table. The table of each step is computed using the table of the previous one. For each class we need to enumerate the possible states of the new elements introduced in the resolved graph H_i, i.e., the new vertex and the new edges added to the new resolved subgraph H_i (see Figure 12).

5.3. *Complexity and Vertex Separation Number*

5.3.1. *Complexity of All Terminal Reliability Computation.* The results of this algorithm [11] prove that the decomposition method is extremely powerful. In practice, for the all terminal reliability problem, it can handle any network with $F_{\max} \leq 12$. Indeed, the resolution time and required memory grow exponentially in function of $F_{\max}$. The

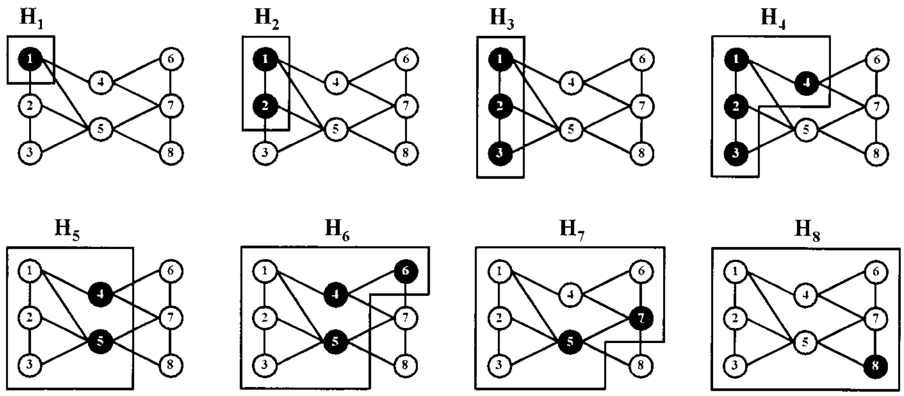
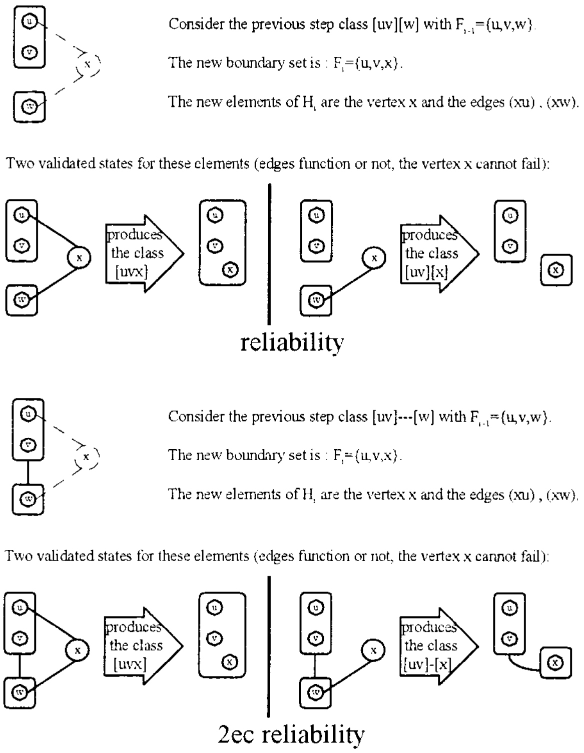


Fig. 11. Evolution of the resolved subgraph during the algorithm. The vertices belonging to F_i are shaded. $F_{max} = 4$.


Consider the previous step class [uv][w] with F_{i-1}={u,v,w}.

The new boundary set is : F_i={u,v,x}.

The new elements of H_i are the vertex x and the edges (xu), (xw).

Two validated states for these elements (edges function or not, the vertex x cannot fail):

Consider the previous step class [uv]---[w] with F_{i-1}={u,v,w}.

The new boundary set is : F_i={u,v,x}.

The new elements of H_i are the vertex x and the edges (xu), (xw).

Two validated states for these elements (edges function or not, the vertex x cannot fail):

Fig. 12. Construction of the new step classes.

Table 4. Reliability computation times (in seconds) with the decomposition algorithm.

	$\lvert V\rvert$									
	10	20	30	40	50	60	70	80	90	100
$F_{\max} = 6$	0.18	0.66	1.16	1.66	2.16	2.64	3.12	3.66	4.12	4.62
$F_{\max} = 7$	0.87	4.23	7.61	10.98	14.39	17.90	21.20	24.79	27.94	31.40
$F_{\max} = 8$	3.36	28.57	53.45	79.08	103.94	128.77	154.09	179.12	204.30	230.21
$F_{\max} = 9$	5.75	202.53	399.74	597.73	793.49	991.21	1186.84	1384.47	1585.97	1775.92

complexity is

$$O\left((|V|)\cdot\sum_{j=1}^{F_{\max}}(A_{F_{\max},j}\cdot 2^j)\cdot(F_{\max})^2\right),$$

where $A_{i,j}$ is the Stirling number of the second kind, and $F_{\max}$ is the vertex separation number. We now give a brief explanation of this complexity: $|V|$ stands for the number of main iterations, i.e., the insertion of a new vertex v_i in H_i. $\sum A_{F_{\max},j}$ is the number of possible classes for a boundary set of size $F_{\max}$. 2^j represents the number of possible states to consider for the edges added to H_i. $(F_{\max})^2$ stands for the class construction and identification in the new resolved subgraph H_{i+1}.

The exponential factor of this method is $F_{\max}$, whereas this factor is the size of the graph for the factoring-reduction method [9], i.e., this algorithm can compute the reliability of large networks whereas the factoring-reduction algorithm is limited to small networks.

Table 4 presents the results of the decomposition algorithm in CPU time, running on a 167 megahertz machine (UltraSparc), for large density graphs with 10–100 vertices, for $F_{\max}$ from 6 to 9. These graphs are $F_{\max}$-paths, which means that the addition of an edge in such a graph increases its $F_{\max}$.

5.3.2. *Complexity of* 2*ec Reliability Computation*. As in the case of reliability, for all terminal 2ec reliability we use a table-based reduction algorithm. The 2ec reliability algorithm requires more classes than the reliability algorithm, which explains why $F_{\max} \leq 7$ in practice. Moreover, these classes are more costly to compute and handle in memory.

The complexity of this algorithm is

$$O\left((|V|)\cdot\sum_{j=1}^{F_{\max}}(A_{F_{max},j}\cdot \mathrm{NFU}(j))\cdot 3^j)\cdot(F_{\max})\cdot \mathrm{NFU}(F_{\max})\right),$$

where $A_{i,j}$ is the Stirling number of the second kind, $F_{\max}$ the vertex separation number, and $\mathrm{NFU}(j)$ is the number of labeled forests with j nodes and k unidentified nodes with $0 \leq k \leq \max(0, j-2)$ (see Property 4.2). We now give a brief explanation for this complexity: $|V|$ stands for the number of main iterations, i.e., the insertion of a new vertex v_i in H_i. $\sum(A_{F_{\max},j}\cdot \mathrm{NFU}(j))$ is the number of possible classes for a boundary set of size $F_{\max}$. 3^j represents the number of possible states to consider for the edges added to H_i. $F_{\max}\cdot \mathrm{NFU}(F_{\max})$ stands for the class construction and identification in the new resolved subgraph H_{i+1}.

Table 5. 2ec reliability computation times (in seconds) with the decomposition algorithm.

	$\|V\|$									
	10	20	30	40	50	60	70	80	90	100
$F_{\max} = 5$	7.60	30.31	53.18	76.09	97.54	158.29	141.41	164.54	197.85	209.47
$F_{\max} = 6$	182.16	1306.62	2336.50	3491.10	4560.01	5711.74	6634.53	7703.86	8769.99	9857.80

Table 5 presents the results of the decomposition algorithm in CPU time, running on a 167 megahertz machine (UltraSparc), for large density graphs with 10–100 vertices, for $F_{\max} = 5$ and 6. These graphs are $F_{\max}$-paths (see Section 5.3.1).

Table 6 presents our results in CPU time, running on a 100 megahertz machine, for small density graphs (grid networks) with 3×2 to 3×9 vertices, for $F_{\max} = 3$, and compares them with an algorithm using a classical state enumeration.

5.3.3. *Linear Time and Vertex Separation Number.* We have seen in Sections 5.3.1 and 5.3.2 that the main factor of the complexity for the all terminal reliability and 2ec reliability algorithms is $F_{\max}$, the size of the maximum boundary set. Moreover, for a given bounded $F_{\max}$, the complexity is linear in $O(|V|)$ (all other factors of the complexity are constant). The linearity of our results can be verified in Tables 4 and 5.

The largest size of the boundary set during our algorithms, $F_{\max}$, depends on the initial vertex linear ordering and so does the complexity. Figure 13 shows another linear ordering for the graph of Figure 11, with $F_{\max} = 3$ instead of $F_{\max} = 4$.

A basic problem is therefore to find a linear ordering for a given graph such that $F_{\max}$ is minimal. This problem is equivalent to the problem of the pathwidth (see Section 5.1). To solve it, we use a heuristic method described in [11].

5.4. *Dynamic Programming Algorithms to Solve* NP-*Hard Problems*. Our algorithm can solve the network reliability problem, which is NP-hard, in linear time for a bounded $F_{\max}$. It has been proved that some classes of graph problems can be solved in polynomial (or linear time) with such dynamic programming algorithms using a tree-decomposition with a bounded width [20]–[23]. The principle of these algorithms is to use the graph tree topology in order to expand a resolved subgraph, until the whole graph is resolved, and to store in memory all partial solutions standing for the resolved elements, i.e., the information required to compute the final solution.

The algorithm presented in this article (Section 5.2) uses a path-decomposition (which is a particular tree-decomposition) produced by the linear ordering of the vertices. The

Table 6. 2ec reliability computation times (in seconds) with the decomposition method (DEC) and with an enumeration method (ENU).

	$\|V\|$							
	6	9	12	15	18	21	24	27
DEC	0.84	0.80	0.76	0.95	0.65	0.67	0.67	0.66
ENU	0.01	0.03	0.14	1.12	9.24	77.59	651.87	5452.46

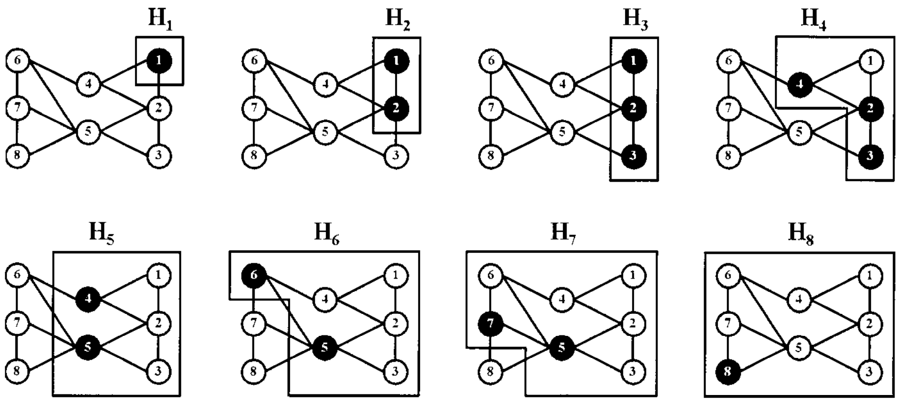


Fig. 13. Another vertex linear ordering. The vertices belonging to F_i are shaded. $F_{\max} = 3$.

vertex separation number ($F_{\max}$) of the linear ordering corresponds to the pathwidth of the path-decomposition [16].

6. Conclusion. We have presented algorithms to compute all terminal reliability and 2ec reliability in linear time for graphs with bounded pathwidths. These algorithms belong to the family of dynamic programming algorithms which solve some NP-hard problems in polynomial (or linear time) with a given small width tree-decomposition. The implementation of our algorithms uses a vertex linear ordering of the graph with a vertex separation number equivalent to the correspondent pathwidth. For greater efficiency, a similar algorithm could be implemented using a tree-decomposition instead of a path-decomposition.

The main difficulty of such dynamic programming algorithms for a given problem is to find and handle the classes. Here we have described the definition of these classes for all terminal reliability and all terminal 2ec reliability with perfect vertices. The classes for K-terminal reliability with imperfect vertices are defined in [11]. With regard to 2-connected reliability, the huge number of classes given in [24] does not allow the decomposition method to be as effective as with reliability and 2ec reliability.

References

[1] A. Rosenthal, Computing the reliability of complex networks, *SIAM J. Appl. Math.*, **32** (1977), 384–393.

[2] C. J. Colbourn and D. D. Harms, Bounding all terminal reliability in computer networks, *Networks*, **18** (1988), 1–12.

[3] M. O. Ball, Computing network reliability, *Oper. Res.*, **27** (1979), 823–838.

[4] O. Frank and W. Gaul, On reliability in stochastic graphs, *Networks*, **12** (1982), 119–126.

[5] O. Theologou, Contribution à l'évaluation de la fiabilité des réseaux, Ph.D. Thesis, Université de technologie de Compiègne, 1990.

[6] M. O. Ball, Complexity of network reliability computation, *Networks*, **10** (1980), 153–165.

[7] A. Satyanarayana and M. Chang, Network reliability and the factoring theorem, *Networks*, **13** (1983), 107–120.

[8] R. K. Wood, A factoring algorithm using polygon-to-chain reductions for computing K-terminal network reliability, *Networks*, **15** (1985), 173–190.

[9] O. Theologou and J. Carlier, Factoring and reductions for networks with imperfect vertices, *IEEE Trans. Reliability*, **40** (1991), 210–217.

[10] J. Carlier and C. Lucet, A decomposition algorithm for network reliability evaluation, *Discrete Appl. Math.*, **65** (1996), 141–156.

[11] C. Lucet, Méthode de décomposition pour l'évaluation de la fiabilité des réseaux, Ph.D. Thesis, Université de technologie de Compiègne, 1993.

[12] S. Arnborg, Efficient algorithms for combinatorial problems on graphs with bounded decomposability—a survey, *BIT*, **25** (1985), 2–23.

[13] N. Robertson and P. D. Seymour, Graph minors. I. Excluding a forest, *J. Combin. Theory Ser. B*, **35** (1983), 39–61.

[14] N. Robertson and P. D. Seymour, Graph minors. II. Algorithmic aspects of tree-width, *J. Algorithms*, **7** (1986), 309–322.

[15] H. L. Bodlaender, A tourist guide through treewidth, *Acta Cybernet.*, **11** (1993), 1–23.

[16] N. G. Kinnersley, The vertex separation number of a graph equals its path width, *Inform. Process Lett.*, **42** (1992), 345–350.

[17] S. Arnborg, D. G. Corneil, and A. Proskurowski, Complexity of finding embeddings in a k-tree, *SIAM J. Algebraic Discrete Methods*, **8** (1987), 277–284.

[18] H. L. Bodlaender, A linear time algorithm for finding tree-decompositions of small treewidth, *SIAM J. Comput.*, **25** (1996), 1305–1317.

[19] H. L. Bodlaender and T. Kloks, Efficient and constructive algorithms for the pathwidth and treewidth of graphs, *J. Algorithms*, **21** (1996), 358–402.

[20] S. Arnborg, J. Lagergren, and D. Seese, Easy problems for tree-decomposable graphs, *J. Algorithms*, **12** (1991), 308–340.

[21] S. Arnborg and A. Proskurowski, Linear time algorithms for NP-hard problems on graphs embedded in k-trees, *Discrete Appl. Math.*, **23** (1989), 11–24.

[22] H. L. Bodlaender, Dynamic programming on graphs with bounded treewidth, *Proc.* 15*th ICALP* '88, LNCS 317, Springer-Verlag, Berlin, 1988, pp. 105–118.

[23] B. Courcelle and M. Mosbah, Monadic second-order evaluations on tree-decomposable graphs, *Theoret. Comput. Sci.*, **109** (1993), 49–82.

[24] J. F. Manouvrier, Méthode de décomposition pour resoudre des problèmes combinatoires sur les graphes, Ph.D. Thesis, Université de technologie de Compiègne, 1998.