Reliability Engineering and System Safety 213 (2021) 107751

Contents lists available at ScienceDirect

Reliability Engineering and System Safety

journal homepage: www.elsevier.com/locate/ress

A C++ class for multi-state algebraic reliability computations

A.M. Bigatti [a], P. Pascual-Ortigosa [b], E. Sáenz-de-Cabezón [b,*]

[a] *Universitá degli Studi Genova, Italy*
[b] *Universidad de La Rioja, Spain*

ABSTRACT

We present the design and implementation of a C++ class for reliability analysis of multi-state systems using an algebraic approach based on monomial ideals. The class is implemented within the open-source CoCoALib library and provides functions to compute system reliability and bounds. The algorithms we present may be applied to general systems with independent components having identical or non-identical probability distributions.

1. Introduction

The development and implementation of efficient algorithms for system reliability computations is an important task in reliability engineering. Many algorithms exist, and are available to the community in a variety of forms. Some are included in large versatile commercial systems [1–3], others are offered as packages, functions or libraries in mathematical software systems of general purpose languages, for example Matlab [4,5], Python [6] or R [7]. Others still are directly distributed by the authors as stand-alone software, like SHARPE [8].

In this paper we describe the C++ class which implements our algebraic approach to system reliability, and can be integrated in other software systems. In this way it may become available in different forms and toolboxes to researchers, software developers and reliability engineers. The language C++ (now in its version 17, standard ISO/IEC 14882) is a widely used [9] object-oriented general purpose computer language, both for very large systems and for small ad-hoc applications. Among the virtues of C++ is its integrability with other languages and also its high performance, meeting the need for fast and reliable computations. Our class is implemented in the CoCoALib library [10], which is a C++ library for Computations in Commutative Algebra, currently at its version 0.99712 (December 2020). It is open source and free.

The main feature of the C++ class introduced in this paper is that it is applicable to a large variety of systems, with or without a known identifiable structure, and can be used to compute the reliability (and bounds) of systems having independent identical or non-identical components. The performance of this class is good in terms of time requirements, being able to compute the reliability of systems with hundreds of components and tens of thousands of minimal paths or cuts. Even though there exist optimized algorithms for several kinds of systems which are faster than the ones presented here, ours are useful to analyze systems for which no specialized algorithms are known, and to benchmark new algorithms for particular types of systems.

The outline of the paper is the following. Section 2 gives an introduction to the algebraic approach for system reliability which is at the mathematical core of our implementation. We describe the main functions and design of the class in Section 3. Finally, in Section 4, we show some examples of its use and the results of some computer experiments. All the code of the class and the examples are available at http://www.dima.unige.it/~bigatti/data/AlgebraicReliability/.

2. Algebraic reliability of coherent systems

The algebraic approach to system reliability is based on resolutions and Hilbert series of ideals in rings of polynomials in several indeterminates. Although at first it might look like an abstract and theoretical method, it is made practically applicable by the combinatorial nature of monomial ideals and it is supported by strong results and algorithms in the area. Other authors have used algebraic structures in system reliability analysis before, see for instance the seminal works [11,12] or the application of algebraic structures to network reliability [13,14]. In particular, the Universal Generating Function method (UGF), introduced in [15] and described in more detail in books like [16], uses the exponents and coefficients of polynomial-like structures to encode the performance distribution and probabilities of multi-state systems. The UGF method is very flexible and has been applied to several types of multi-state systems, see for instance [17] for a recent example.

In this section we give an overview of the basics of the approach by monomial ideals, that is the backbone of the algorithms in the C++ class we introduce in Section 3.

* Corresponding author.
E-mail addresses: bigatti@dima.unige.it (A.M. Bigatti), papasco@unirioja.es (P. Pascual-Ortigosa), eduardo.saenz-de-cabezon@unirioja.es (E. Sáenz-de-Cabezón).

https://doi.org/10.1016/j.ress.2021.107751
Received 8 October 2020; Received in revised form 4 May 2021; Accepted 5 May 2021
Available online 27 May 2021

2.1. Coherent systems

A system $\mathcal{S}$ consists of n components which are its elementary units, we denote the n system components by c_i with $i \in \{1, \dots, n\}$. At each moment in time the system is in one of a discrete set of levels $S = \{0, 1, \dots, M\}$ indicating growing levels of performance or of failure. Each individual component c_i of the system can be in one of a discrete set of levels $S_i = \{0, \dots, M_i\}$. A state of a component is its level and a state of the system is the n-tuple of its components' states. Given two states $s = (s_1, \dots, s_n)$ and $t = (t_1, \dots, t_n)$ we say that $s \geq t$ if $s_i \geq t_i$ for all $i = 1, \dots, n$ and, conversely, that $s \leq t$ if $s_i \leq t_i$ for all i. The level of performance of the system is determined in terms of the states of the components by a *structure function* $\Phi : S_1 \times \cdots \times S_n \longrightarrow S$. The system $\mathcal{S}$ is said to be *coherent* if Φ is non-decreasing and each component is relevant to the system, *i.e.* for each component c_i there exist a system state $s = (s_1, \dots, s_n)$ and two different levels $j, k \in S_i$ such that $\Phi(s_{i,j}) \neq \Phi(s_{i,k})$, where $s_{i,\ell} = (s_1, \dots, s_{i-1}, \ell, s_{i+1}, \dots, s_n)$.

We distinguish between *working systems*, also called *path systems*, *i.e.* systems described by the working states of their components, and *failure systems* or *cut systems i.e.* systems described by the failure states of their components.[1]

In *path systems* the focus is on working states. In this case the levels of the system, $\{0, \dots, M\}$ indicate growing levels of performance, the system being in level 0 indicates that the system is failing, and level $j > i$ indicates that the system is performing at level j better than at level i. For each component c_i and for each of its levels j, we denote by $p_{i,j}$ the probability that c_i is performing at level $\geq j$. The j-reliability of $\mathcal{S}$, denoted by $R_j(\mathcal{S})$ is the probability that $\mathcal{S}$ is performing at level $\geq j$; conversely, the j-unreliability of $\mathcal{S}$, denoted $U_j(\mathcal{S})$, is $1 - R_j(\mathcal{S})$. A *path system* is given at level j by its set of j-working states *i.e.* those tuples $(s_1, \dots, s_n) \in S_1 \times \cdots \times S_n$ such that $\Phi(s_1, \dots, s_n) \geq j$. We say that a state $(s_1, \dots, s_n) \in S_1 \times \cdots \times S_n$ is a *minimal j-working state* or *minimal j-path* if $\Phi(s_1, \dots, s_n) \geq j$ and $\Phi(t_1, \dots, t_n) < j$ whenever all $t_i \leq s_i$ and at least in one case the inequality is strict. We say that a state $(s_1, \dots, s_n) \in S_1 \times \cdots \times S_n$ is a *minimal j-failure state* or *minimal j-cut* if $\Phi(s_1, \dots, s_n) < j$ and $\Phi(t_1, \dots, t_n) \geq j$ whenever all $t_i \geq s_i$ and at least one of the inequalities is strict. Path systems are usually denoted by :G (for *good*) in the literature.

Example 2.1. Let $\mathcal{S}$ be a path system with 3 components c_1, c_2, c_3 such that $S = S_1 = S_2 = S_3 = \{0, 1, 2\}$. The structure function of $\mathcal{S}$ is given by $\Phi(s_1, s_2, s_3) = \min\{s_1, s_2, s_3\}$ *i.e.* $\mathcal{S}$ is a multi-state *series:G* system, the system works at level $j > 0$ only if all of its components are working at level j or bigger. Let $p_{1,1} = 0.8, p_{1,2} = 0.75, p_{2,1} = 0.9, p_{2,2} = 0.8$ and $p_{3,1} = 0.75, p_{3,2} = 0.7$.

The only minimal 1-working state of $\mathcal{S}$ is $(1, 1, 1)$ and the only minimal 2-working state is $(2, 2, 2)$. The minimal 1-failure states are $(2, 2, 0)$, $(2, 0, 2), (0, 2, 2)$ and the minimal 2-failure states are $(2, 2, 1), (2, 1, 2)$, $(1, 2, 2)$. The 1-reliability of the system is $R_1(\mathcal{S}) = 0.54$ and $U_1(\mathcal{S}) = 0.46$; the 2-reliability is $R_2(\mathcal{S}) = 0.42$ and $U_2(\mathcal{S}) = 0.58$.

In *cut systems* the focus is on the failure of the system, the exact counterpart of path systems. In this case the levels of the system, $\{0, \dots, M\}$, indicate growing levels of failure. Hence being at level 0 indicates that the system is completely functional and level $j > i$ indicates that the system is performing worse at level j than at level i (*i.e.* the higher the level, the higher the intensity of failure). For each state j of each of the components c_i we denote by $q_{i,j}$ the probability that component c_i is failing at level $\geq j$. The j-unreliability $U_j(\mathcal{S})$ of $\mathcal{S}$ is the probability that $\mathcal{S}$ is failing at level $\geq j$; conversely, the j-reliability of $\mathcal{S}$ is $R_j(\mathcal{S}) = 1 - U_j(\mathcal{S})$. A *cut system* is given at level j by its set of j-failing states *i.e.* those tuples $(s_1, \dots, s_n) \in S_1 \times \cdots \times S_n$ such that $\Phi(s_1, \dots, s_n) \geq j$. We say that a state $(s_1, \dots, s_n) \in S_1 \times \cdots \times S_n$ is a *minimal j-failing state* or *minimal j-cut* if $\Phi(s_1, \dots, s_n) \geq j$ and $\Phi(t_1, \dots, t_n) < j$ whenever all $t_i \leq s_i$ and at least in one case the inequality is strict. We say that a state $(s_1, \dots, s_n) \in S_1 \times \cdots \times S_n$ is a *minimal j-working state* or *minimal j-path* if $\Phi(s_1, \dots, s_n) < j$ and $\Phi(t_1, \dots, t_n) \geq j$ whenever all $t_i \geq s_i$ and at least one of the inequalities is strict. Cut systems are usually denoted by :F (for *fail*) in the literature.

Example 2.2. Let $\mathcal{S}$ be a cut system with 3 components c_1, c_2, c_3 such that $S = S_1 = S_2 = S_3 = \{0, 1, 2\}$. The structure function of $\mathcal{S}$ is given by $\Phi(s_1, s_2, s_3) = \max\{s_1, s_2, s_3\}$ *i.e.* $\mathcal{S}$ is a multi-state *parallel:F* system, the system fails at level $j > 0$ or bigger whenever any of its components is failing at level j or more. Let $q_{1,1} = 0.25, q_{1,2} = 0.2, q_{2,1} = 0.2, q_{2,2} = 0.1$ and $q_{3,1} = 0.3, q_{3,2} = 0.25$.

The minimal 1-failing states of $\mathcal{S}$ are $(1, 0, 0), (0, 1, 0), (0, 0, 1)$ and the minimal 2-failing states of this system are $(2, 0, 0), (0, 2, 0), (0, 0, 2)$. The 1-unreliability of the system is $U_1(\mathcal{S}) = 0.58$ and $R_1(\mathcal{S}) = 0.42$; the 2-unreliability of the system is $U_2(\mathcal{S}) = 0.46$ and $R_2(\mathcal{S}) = 0.54$.

For clarity, in this paper we refer mainly to path systems. All the computations and considerations may be correspondingly applied to cut systems unless otherwise stated. We assume that the working or failure probabilities of the components in any system are independently distributed, although the method can also be applied to systems with dependent components. Also, we will consider that the probabilities of the systems' components to be in their different levels are constant in time (or equivalently, we consider reliability at a given instant t or for the steady state of the system). The algebraic method can be applied to components' probabilities that vary in time, and to repairable systems and renewal processes. In those cases, we must consider the j-reliability of the system, *i.e.* the probability that the system is performing at level j or better, and also the j-availability of the system at a given instant t or an interval $\mathcal{T}$ of time, *i.e.* the probability that the system is performing at level j or better for all $t \in \mathcal{T}$. Both notions have been extensively studied in the literature of multi-state systems, e.g. [19]. For complete introductions to multi-state system reliability and methods see [18,20, 21].

2.2. Algebraic reliability

The use of commutative algebra (in particular of monomial ideals) in system reliability started in [22,23] in a close relation to improvements in inclusion–exclusion formulas and Bonferroni bounds [24]. The approach we follow was developed in a series of papers, e.g. [25–27]. The main idea is to associate an algebraic object to a coherent system and obtain information about the structure and reliability of the system by investigating the properties of the algebraic object. In this section we make a brief self-contained description of the algebraic concepts involved and refer the interested reader to the cited series of papers for full details and proofs.

Let $\mathcal{S}$ be a path system with n components and let $j \in \{0, \dots, M\}$ be one of the levels of the system. Let $F_j(\mathcal{S})$ be the set of j-working states of the system and $\overline{F}_j(\mathcal{S})$ the subset of minimal j-working states. Let us consider $P = \mathbf{k}[x_1, \dots, x_n]$ a polynomial ring in n indeterminates, one for each component of $\mathcal{S}$; here $\mathbf{k}$ denotes any field of characteristic 0, but for clarity we assume that our coefficients are in $\mathbb{Q}$ or $\mathbb{R}$. To each state $s = (s_1, \dots, s_n) \in S_1 \times \cdots \times S_n$ of $\mathcal{S}$ we associate the monomial $x^s = x_1^{s_1} \cdots x_n^{s_n} \in P$.

We denote by $pr(x^s) = \prod_{i=1}^{n} p_{i,s_i}$ the probability that the system is in a state $\geq s$. In algebraic terms, having a state $t \geq s$ is equivalent to saying that the monomial x^t is a multiple of x^s, *i.e.* x^t is in $I = \langle x^s \rangle$, the ideal in P generated by the monomial x^s. Now we consider the

[1] There are several definitions of multi-state systems in the literature, see the relationships between them in the diagram given by Natvig in [18] Figure 2.1. The algebraic methodology that we use can be applied to multi-state strongly coherent systems, multi-state coherent systems and multi-state weakly coherent systems as defined in [18] Definition 2.4 since the algebraic expressions of reliability are not affected by irrelevant components for each of the levels of performance of the system.

probability that the system is in a state greater then or equal to at least one of the states in $\{\mu_1, \dots, \mu_r\}$: this situation algebraically corresponds to the set of monomials x^t belonging to the ideal $I = \langle x^{\mu_1}, \dots, x^{\mu_r} \rangle$. Thus we denote its probability by $pr(I) = pr(\bigcup_{i=1}^r \langle x^{\mu_i} \rangle)$.

The ideal generated by the j-working states of $\mathcal{S}$ is denoted by $I_j(\mathcal{S})$ and is called the *j-reliability ideal* of $\mathcal{S}$. For a monomial ideal there is a unique minimal monomial generating set, denoted $\mathrm{MinGens}(I)$. Thus we observe that, due to the coherence property of $\mathcal{S}$, we have that $\mathrm{MinGens}(I_j(\mathcal{S}))$ is the set of the monomials corresponding to the minimal j-paths of $\mathcal{S}$

$$I_j(\mathcal{S}) = \langle x^\mu \mid \mu \in \overline{F}_j(\mathcal{S}) \rangle.$$

From these definitions we have that the reliability of $\mathcal{S}$ is given by

$$R_j(\mathcal{S}) = pr(I_j(\mathcal{S})).$$

Remark 2.3. The definition and description of the structure function of a multi-state coherent systems can be based on the sets of minimal j-paths or minimal j-cuts, which extend the notion of minimal paths and minimal cuts from binary systems [28]. These special sets of state vectors are described as lower boundary points and upper boundary points to level j in [28]. Both sets can be formulated in algebraic terms as the minimal generators of the j-reliability ideal (lower boundary points) and maximal standard pairs (upper boundary points). For full details and a complete proof of this correspondence, see [29].

Since $pr(I_j(\mathcal{S}))$ is expressed as the probability of a union, a natural choice for this computation is using the inclusion–exclusion principle, which in this case can be expressed as

$$pr(I_j(\mathcal{S})) = \sum_{i=1}^{r} (-1)^{i+1} \sum_{|\sigma|=i} pr(\mathrm{lcm}(x^{\mu_s} \mid s \in \sigma)), \tag{1}$$

where σ denotes subsets of $\{1, \dots, r\}$ and lcm denotes the least common multiple.

A compact form of Eq. (1) can be obtained by the multigraded Hilbert series of $I_j(\mathcal{S})$. The multigraded Hilbert series of an ideal or module is a very important invariant in commutative algebra and algebraic geometry [30]. It is useful in our context because it provides a compact way to enumerate all monomials in a monomial ideal. The multigraded Hilbert series of an ideal $I \in P$, given by

$$H_I(x_1, \dots, x_n) = \sum_{\mu \in \mathbb{N}^n} [x^\mu \in I] x^\mu,$$

where the symbol $[x^\mu \in I]$ is equal to 1 if x^μ is in I and 0 otherwise. The multigraded Hilbert series is an element of the formal power series ring $\mathbb{Z}[[x_1, \dots, x_n]]$. Observe that in this ring we have the identity $\frac{1}{1-x_i} = 1 + x_i + x_i^2 + \cdots$ and hence one way to enumerate all the monomials in P is to consider the summands of $\prod_{i=1}^n \frac{1}{1-x_i}$. Therefore the Hilbert series of P is given by $H_P(x_1, \dots, x_n) = \prod_{i=1}^n \frac{1}{1-x_i}$. Just by multiplying every monomial by a given $x^\mu \in P$ one obtains that $H_{\langle x^\mu \rangle}(x_1, \dots, x_n) = \prod_{i=1}^n \frac{x^\mu}{1-x_i}$. Now, since the set of monomials in a monomial ideal I generated by $\{x^{\mu_1}, \dots, x^{\mu_r}\}$ is the union of the sets of monomials in each of the ideals $\langle x^{\mu_i} \rangle$ then we have that

$$H_I(x_1, \dots, x_n) = \sum_{i=1}^{r} (-1)^{i+1} \sum_{|\sigma|=i} \frac{\mathrm{lcm}(x^{\mu_s} \mid s \in \sigma)}{\prod_{j=1}^n (1 - x_j)}, \tag{2}$$

which is the algebraic version of Eq. (1). Let $HN_I(x_1, \dots, x_n)$ denote the numerator of the Hilbert series of the ideal I and let $\mathcal{S}$ be a coherent path system as in Section 2.1. Let $pr(HN_{I_j(\mathcal{S})}(x_1, \dots, x_n))$ denote the formal substitution of every x^μ by $pr(x^\mu)$ in the numerator of the multigraded Hilbert series of $I_j(\mathcal{S})$. The direct relation between Eqs. (1) and (2) allows us to establish the fundamental identity of the algebraic approach to system reliability

$$R_j(\mathcal{S}) = pr(I_j(\mathcal{S})) = pr(HN_{I_j(\mathcal{S})}(x_1, \dots, x_n)). \tag{3}$$

Hence any way to obtain $HN_{I_j(\mathcal{S})}(x_1, \dots, x_n)$ gives us a way to compute $R_j(\mathcal{S})$. Of course a direct one, although very redundant in general, is Eq. (2) by means of the inclusion–exclusion principle. Other more efficient and compact ways to obtain $HN_{I_j(\mathcal{S})}(x_1, \dots, x_n)$ are described in [31].

An important feature of the inclusion–exclusion formulas is that they can be truncated to obtain the so called Bonferroni bounds [24]. More precisely, we have that

$$\begin{aligned} pr(I_j(\mathcal{S})) &\le \sum_{i=1}^{t} (-1)^{i+1} \sum_{|\sigma|=i} pr(\mathrm{lcm}(x^{\mu_s} \mid s \in \sigma)) \text{ for } t \le r \text{ odd}, \\ pr(I_j(\mathcal{S})) &\ge \sum_{i=1}^{t} (-1)^{i+1} \sum_{|\sigma|=i} pr(\mathrm{lcm}(x^{\mu_s} \mid s \in \sigma)) \text{ for } t \le r \text{ even}. \end{aligned} \tag{4}$$

One way to obtain the multigraded Hilbert series of a monomial ideal I is by constructing a multigraded free resolution of I and read $HN(I)$ from the data in the resolution. Every ideal $I \subseteq P = \mathbf{k}[x_1, \dots, x_n]$ can be described as a module in terms of what is called a *free resolution*, which is a series of free modules and morphisms among them. A free module is a direct sum of copies of P with the usual grading shifted by some degree $d \in \mathbb{N}$ denoted by $P(-d)$. In the case of monomial ideals we can also have multigraded resolutions, in which the degree shifts are given by multidegrees $(d_1, \dots, d_n) \in \mathbb{N}^n$ and the shifted copies of P are denoted by $P(\mu)$. A multigraded free resolution of a monomial ideal I is of the form

$$0 \longrightarrow \bigoplus_{j=1}^{r_d} P(-\mu_{d,j}) \xrightarrow{\partial_d} \cdots \xrightarrow{\partial_2} \bigoplus_{j=1}^{r_1} P(-\mu_{1,j}) \xrightarrow{\partial_1} P/I \longrightarrow 0,$$

where the ∂_i are graded module morphisms (here called *differentials*), d is the length of the resolution, the r_i are called *ranks* of the resolution and the $\mu_{i,j}$ for each i are the multidegrees of the ith module of the resolution. Given an ideal I one can build different resolutions and among them there is a distinguished one called the *minimal free resolution* which is unique up to isomorphisms and is characterized by having smallest ranks among all the possible resolutions of I. The ranks of the minimal free resolution of I are called the *Betti numbers* of I and are a fundamental invariant of I [30].

Now, given any multigraded free resolution of a monomial ideal I we have the following expression for $HN_I(x_1, \dots, x_n)$

$$HN_I(x_1, \dots, x_n) = \sum_{i=1}^{d} (-1)^i \sum_{j=1}^{r_i} x^{\mu_{i,j}}. \tag{5}$$

This expression can be truncated as in (4) and produces the following bounds for $R_j(\mathcal{S})$, see [25]

$$\begin{aligned} R_j(\mathcal{S}) &\le \sum_{i=1}^{t} (-1)^{i+1} \sum_{j=1}^{r_i} pr(x^{\mu_{i,j}}) \text{ for } t \le r \text{ odd}, \\ R_j(\mathcal{S}) &\ge \sum_{i=1}^{t} (-1)^{i+1} \sum_{j=1}^{r_i} pr(x^{\mu_{i,j}}) \text{ for } t \le r \text{ even}. \end{aligned} \tag{6}$$

Among this type of bounds, those given by the minimal multigraded free resolution of $I_j(\mathcal{S})$ are the tightest, cf. [25].

Example 2.4. Consider the series:G system $\mathcal{S}$ studied in Example 2.1. The j-reliability ideals of $\mathcal{S}$ are $I_1(\mathcal{S}) = \langle x_1x_2x_3 \rangle$ and $I_2(\mathcal{S}) = \langle x_1^2x_2^2x_3^2 \rangle$. The minimal free resolution of $I_1(\mathcal{S})$ has length 1 and the only free module of this resolution has multidegree $(1, 1, 1)$, hence $R_1(\mathcal{S}) = pr(x_1x_2x_3) = 0.54$. An equivalent computation shows that $R_2(\mathcal{S}) = pr(x_1^2x_2^2x_3^2) = 0.42$.

Example 2.5. Consider now the parallel:F system $\mathcal{S}$ in Example 2.2. The j-unreliability ideals of $\mathcal{S}$ are $I_1(\mathcal{S}) = \langle x_1, x_2, x_3 \rangle$ and $I_2(\mathcal{S}) = \langle x_1^2, x_2^2, x_3^2 \rangle$. The minimal free resolution of $I_2(\mathcal{S})$ has length 3 and the multidegrees of its modules are $\mu_{1,1} = (2,0,0), \mu_{1,2} = (0,2,0), \mu_{1,3} =$

$(0,0,2), \mu_{2,1} = (2,2,0), \mu_{2,2} = (2,0,2), \mu_{2,3} = (0,2,2)$ and $\mu_{3,1} = (2,2,2)$ hence the 2-unreliability of $\mathcal{S}$ is given by

$$U_2(\mathcal{S}) = pr(x_1^2) + pr(x_2^2) + pr(x_3^2) - (pr(x_1^2x_2^2) + pr(x_1^2x_3^2) + pr(x_2^2x_3^2)) + pr(x_1^2x_2^2x_3^2) = 0.55 - 0.095 + 0.005 = 0.46.$$

Observe that truncating this expression we obtain a first upper bound of 0.55 and a first lower bound of 0.455 for $U_2(\mathcal{S})$.

The algebraic methodology for reliability computation using monomial ideals is based on two main principles. The first one is to avoid as much redundancy as possible when enumerating the states needed for the final reliability computation. This is provided by the possibility of using different resolutions to express the numerator of the Hilbert series of the system's ideals. In this respect, a fast computation of the minimal resolution or close-to-minimal resolutions is the main component of our approach. The second principle is that this methodology can be approached as a recursive procedure, computing the Hilbert series of an ideal in terms of the Hilbert series of smaller ideals. Recursion is usually very efficient in reliability computations and is used in other methodologies, such as the UGF method [16], the decomposition and factoring methods [20,21] or ad-hoc methods for particular systems, see [32] for instance.

To achieve the aforementioned two principles, we use in our algorithm the Hilbert series expression given by ranks of resolutions computed by Mayer–Vietoris trees, described in [33]. They are a fundamental tool in our implementation of the algebraic method for system reliability, for they are a fast and efficient recursive algorithm and their output is a resolution that is in most cases minimal or very close to minimal. This is the main ingredient of the efficiency of the class described in Section 3 which is demonstrated in some examples in Section 4.

Remark 2.6. The principles of avoiding redundant computations and using recurrence are the keys of the UGF method, which also uses an algebraic formalism. They are however used in a slightly different way compared to our approach. In the first place, the UGF method avoids the redundancy by collecting like terms during the recursive computation. In our approach we reduce redundant terms by checking divisibility of the involved monomials during the computation and hence not using terms that would be canceled in the final evaluation of the expression. With respect to the recursive methodology, the recursions used in the UGF method needs a certain structure for the base cases (e.g. series or parallel systems as base case to stop the recursion) and the recursion is applied by blocks or subsystems. In our case the recursion is applied by selecting a single generator of the ideal in each step. This reduces the need of previous knowledge of the system's structure and is therefore very general, but has the disadvantage of eventually producing larger recursion trees.

These observations suggest that both methods could mutually benefit from considering the stronger points of the other to improve their own performance. A full comparison is certainly worth considering, but it is beyond the scope of this paper.

Example 2.7. Consider the double bridge binary system in Fig. 1, taken from [24]. It has 5 nodes and 8 connections and we assume that only its connections are subject to failure. The minimal cuts of the system are 123, 1258, 13 457, 1478, 2346, 24 568, 3567 and 678. The cut ideal corresponding to this network is then

$$I = \langle x_1x_2x_3,\ x_1x_2x_5x_8,\ x_1x_3x_4x_5x_7,\ x_1x_4x_7x_8,\ x_2x_3x_4x_6,\ x_2x_4x_5x_6x_8,\ x_3x_5x_6x_7,\ x_6x_7x_8\rangle.$$

To compute the (un)reliability of this system we can use the Hilbert series of its cut ideal, which has 8 generators. Using the Taylor resolution corresponds to the inclusion–exclusion method, and uses in this case 255 summands for the Hilbert series. Another resolution is

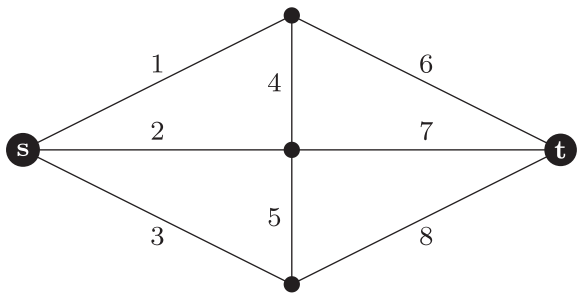

Fig. 1. Double bridge network.

the Scarf resolution, that is equivalent to apply the method of *abstract tubes* [23,24], which in this case is much less redundant, giving an expression for the Hilbert series that uses only 51 summands. Using the minimal free resolution gives an expression using 43 summands and is the most compact form achievable with this methodology. If we use the expression given by the inclusion–exclusion method and Bonferroni bounds obtained by truncation, we have that the eighth bound is sharp, while in the case of the Scarf or minimal resolutions already the fourth bound is sharp.

Remark 2.8. The network in Example 2.7 can be handled efficiently using the factoring algorithm. The domination invariant [34] of this system is 4, which implies that the reliability function can be expressed very easily as the sum of just four terms. Observe that this is the same number of terms in the minimal resolution of the reliability ideal of this system.

2.3. Duality

Given a structure function Φ its dual Φ^D with respect to $\mathbf{t} \in \mathbb{N}^n$ is given by (cf. [19])

$$\Phi^D(s_1, \dots, s_n) = M - \Phi((t_1 - s_1, \dots, t_n - s_n)). \tag{7}$$

Example 2.9. Consider a binary series:G system $\mathcal{S}$ with three components where $\Phi(s_1, s_2, s_3) = \min\{s_1, s_2, s_3\}$. We have $\Phi^D(s_1, s_2, s_3) = 0$ if and only if $(s_1, s_2, s_3) = (0,0,0)$ hence the minimal working states of the dual system are $(1,0,0), (0,1,0)$ and $(0,0,1)$, which correspond to a parallel system. The dual of a series system is always a parallel system and vice-versa.

There is a notion of duality in monomial ideals, called *Alexander duality* [35]. To describe it we use the following notation. Given a vector $\mu \in \mathbb{N}^n$, we denote by $\mathfrak{m}^\mu$ the monomial ideal

$$\mathfrak{m}^\mu = \langle x_i^{\mu_i} \mid \mu_i \geq 1\rangle.$$

Given two vectors μ and ν in $\mathbb{N}^n$ let $\mu\backslash\nu$ the vector whose i'th coordinate is $\mu_i + 1 - \nu_i$ if $\nu_i \geq 1$ and 0 otherwise.

Definition 2.10. Let $I \subset \mathbf{k}[x_1, \dots, x_n]$ be a monomial ideal, $\mathrm{MinGens}(I)$ its minimal set of monomial generators, and $x^\nu = \mathrm{lcm}(\mathrm{MinGens}(I))$. The Alexander dual of I is the intersection

$$I^D = \bigcap_{x^\mu \in \mathrm{MinGens}(I)} \mathfrak{m}^{\nu\backslash\mu},$$

where $\mathfrak{m}^{(s_1,\dots,s_n)}$ denotes the monomial ideal $\langle x_i^{s_i} \mid s_i \geq 1\rangle$

Remark 2.11. Given a coherent system $\mathcal{S}$ its j-reliability ideal $I_j(\mathcal{S})$ is generated by the monomials corresponding to its minimal j-paths. The ideal of its dual system $I_j(\mathcal{S}^D)$ is generated by the monomials corresponding to maximal j-cuts of $\mathcal{S}$ and may be seen as the ideal generated by the maximal standard pairs of $I_j(\mathcal{S})$ [29]. These can be computed using the Alexander dual of the artinian ideal $I_j(\mathcal{S}) + \langle x_i^{M_1+1}, \dots, x_n^{M_n+1}\rangle$ [36].

We can use the dual ideal of a system to compute its reliability in the following way. Let $\overline{pr}(x^\mu) = \prod_{i=1}^n (1 - p_{i,\mu_i+1})$ *i.e.* the product of the probabilities that each component i is in a state less than or equal to μ_i. We denote by $\nu = (M_1, \dots, M_n)$ the vector of maximal possible levels of the components. Let $\overline{I}_j(\mathcal{S})$ be the ideal generated by the monomials $\{x^{\overline{\mu}} \mid x^\mu \text{ is a generator of } I_j(\mathcal{S})\}$. We consider the ideal $\overline{I}_j(\mathcal{S})^D$ and compute $H_{\overline{I}_j(\mathcal{S})^D}(x_1, \dots, x_n)$. We obtain $U_j(\mathcal{S}) = 1 - R_j(\mathcal{S})$ by formally substituting each monomial x^μ in $HN_{\overline{I}_j(\mathcal{S})^D}(x_1, \dots, x_n)$ by $\overline{pr}(\frac{x^\nu}{x^\mu})$.

Example 2.12. Consider the system in Example 2.9. We have that $I_1(\mathcal{S}) = \langle x_1x_2x_3\rangle$, then $I_1(\mathcal{S})^D = \langle x_1, x_2, x_3\rangle$ and $\overline{I}_1(\mathcal{S})^D = \langle x_1, x_2, x_3\rangle$. By using the minimal free resolution of $\overline{I}_1(\mathcal{S})^D = \langle x_1, x_2, x_3\rangle$ we have that $HN_{\overline{I}_1(\mathcal{S})^D}(x_1, \dots, x_n) = (x_1 + x_2 + x_3) - (x_2x_3 + x_1x_3 + x_1x_2) + x_1x_2x_3$. Hence, if we set the probabilities $p_{1,1} = 0.8$, $p_{2,1} = 0.9$ and $p_{3,1} = 0.75$, we get

$$\begin{aligned} U_1(\mathcal{S}) &= \overline{pr}(x_1x_2) + \overline{pr}(x_1x_3) + \overline{pr}(x_2x_3) - \overline{pr}(x_1) - \overline{pr}(x_2) - \overline{pr}(x_3) + \overline{pr}(1) \\ &= 0.25 + 0.1 + 0.2 - (0.025 + 0.05 + 0.02) + 0.005 = 0.46, \end{aligned}$$

and we obtain $R_1(\mathcal{S}) = 0.54$. Observe that in the equality above, $\overline{pr}(1) = \overline{pr}(x_1^0x_2^0x_3^0) = pr(x_1 \le 0)pr(x_2 \le 0)pr(x_3 \le 0) = 0.005$.

3. Algebraic reliability class in `CoCoALib`

The good performance of an algorithm depends also on the efficiency of its implementation. In this section we give the interested reader some technical details on the implementation of our algorithms and some of the decisions we made, like the choice of data types and the structure of the algorithms. These decisions contribute to the actual performance of the algorithms in terms of memory usage and CPU time. Also, we describe the `CoCoALib` library, which provides convenient implementations of the main algebraic structures we need. We hope these descriptions, although not fully detailed, make it easier for engineers and reliability practitioners to practically use these algorithms or incorporate them into their own software, and also make it easy to reproduce our results, experiments and benchmarks.

3.1. CoCoALib

`CoCoALib`, for Computations in Commutative Algebra, is an open source `C++` software library principally based on multi-variate monomials and polynomials and devoted to algebraic geometry. It is the computational core of the CoCoA software system [37]. A crucial aspect of CoCoALib is that it was designed from the outset to be an open-source software library. This initial decision, together with the desire to help the software prosper, has many implications: *e.g.* designing a particularly clean interface for all functions with comprehensive documentation. This cleanliness makes it easy to integrate CoCoALib into other software in a trouble-free manner. The library is fully documented, and also comes with about 100 illustrative example programs. CoCoALib reports errors using `C++` exceptions, while the library itself is exception-safe and thread-safe. The current source code follows the `C++14` standard. The main features of the design of CoCoALib are:

- it is well-documented, free and open source `C++` code (under the GPL v.3 licence);
- the design is inspired by, and respects, the underlying mathematical structures;
- the source code is clean and portable;
- the user function interface is natural for mathematicians, and easy to memorize;
- execution speed is good with robust error detection.

The design of the library (and its openness) was chosen to facilitate and encourage “outsiders” to contribute. There are two categories of contribution: code written specifically to become part of CoCoALib, and stand-alone code written without considering its integration into CoCoALib. The library has combined some of the features of various external libraries into CoCoALib. Such as *Frobby* (see [38]) which is specialized for operations on monomial ideals. Other integrations are with *Normaliz* library for computing with affine monoids or rational cones and *GFanLib* which is a `C++` software library for computing Gröbner fans and tropical varieties.

3.2. The class description

We have implemented within CoCoALib a set of `C++` classes for making computations in algebraic reliability. The UML class diagram is depicted in Fig. 7 in Appendix. Our main class is the *abstract class* `CoherentSystem` which consists of a series of levels and a matrix of probabilities. The levels are stored in a `std::vector` (an efficient structure of the `C++` language) in which each component is an instance of the class `CoherentSystemLevel`, and the probabilities are given by a `vector` of `vectors` of type `double` where the j'th entry of the i'th vector corresponds to $p_{i,j} = p(s_i \ge j)$, the probability that the level of the i'th component of the system is bigger than or equal to j. Each instance of the class `CoherentSystemLevel` consists basically of an ideal and its dual, which are objects of the CoCoALib class `ideal`. Also, we store as member fields their Mayer–Vietoris trees, which play the role of multigraded free resolutions optimized for monomial ideals.

The *concrete classes* inheriting from the class `CoherentSystem` are `CoherentSystemPath` and `CoherentSystemCuts` which respectively represent :G systems in which the levels and probabilities denote working states, and :F systems in which the levels and probabilities represent failures, as seen in Section 2.1. For any instance of these two concrete classes, and hence of the abstract class `CoherentSystem` we can call the following member functions:

myMinimalPaths Receives a level and gives a vector of vectors of type `long`. Each of these vectors is a minimal path of the system at the given level.

myMinimalCuts Receives a level and gives a vector of vectors of type `long`. Each of these vectors is a minimal cut of the system at the given level.

myReliability Receives a level j and computes $R_j(\mathcal{S})$.

myUnreliability Receives a level j and computes $U_j(\mathcal{S})$.

myReliabilityBounds Receives a level j and computes bounds for $R_j(\mathcal{S})$ given by the resolution obtained by the Mayer–Vietoris tree of $I_j(\mathcal{S})$ as given in Eq. (6).

myUnreliabilityBounds Receives a level j and computes bounds for $U_j(\mathcal{S})$ given by the resolution obtained by the Mayer–Vietoris tree of $I_j(\mathcal{S})$ computed from the bounds for $R_j(\mathcal{S})$ given in Eq. (6).

In addition, for :G systems given by its sets of paths, we have implemented two more bounds, described by Gåsemyr and Natvig in [39]:

GNMaxMinPathBound Let $\mathbf{y^m}$, $m = 1, \dots, M_p$ the minimal paths of $\mathcal{S}$ for level j, the following lower bound for $R_j(\mathcal{S})$ is given in [40]:

$$l'^j(\mathbf{p}) = \max_{1\le m\le M_p} \left(\prod_{i=1}^n p_i^{y_i^m} \right)$$

GNCoproductMinCutsBound Let $\mathbf{z^m}$, $m = 1, \dots, M_c$ the set of minimal cut vectors of $\mathcal{S}$ for level j, then we have the following lower minimal bound for $R_j(\mathcal{S})$ [40]:

$$l^{**j}(\mathbf{p}) = \prod_{m=1}^{M_c} \coprod_{i=1}^{n} p_i^{z_i^m+1}$$

where for $p_i \in [0, 1]$ we define $\coprod_{i=1}^n = 1 - \prod_{i=1}^n (1 - p_i)$.

When computing the functions `myReliability`, `myUnreliability`, `myReliabilityBounds` or `myUnreliabilityBounds` the object checks its ideal and its dual ideal, and chooses whichever of them has a smallest number of minimal generators to perform the actual computation. To compute duals of ideals we use the `Frobby` library, in particular the function `FrbAlexanderDual` which is in general a fast computation. Once the ideal is chosen, we check whether the system has already computed its Mayer–Vietoris tree. If it is not yet computed, it is computed and stored in the corresponding class member field. Then the Mayer–Vietoris tree is used to retrieve the required value or bounds for reliability or unreliability.

4. Examples of use

In this section we apply our C++ class to some examples of reliability computations. We use binary networks and multi-state systems. We consider systems in which their components have independent identically distributed probabilities as well as systems in which the components' probabilities are independent but not identically distributed. All the computations in this section have been implemented by the authors and executed in an HP Z-book laptop.[2]

4.1. Test examples

First, we validate our algorithms with a set of diverse examples of multi-state systems found in the literature. We selected systems of different nature so that we can test our algorithms with examples featuring different characteristics. Table 1 shows the results of these tests. The first column of the table indicates the name of each example (see description below), n indicates the number of variables and M the number of levels of the system (not counting the complete failure level or level 0). Column M_i indicates the number of levels of each component and column $gens(I_j)$ indicates the number of minimal generators of the j-reliability ideal for each level $j = 1, \dots, M$. The set of test examples consists of the following:

- `Army Battle Plan` is taken from the classical paper [28]. It is a customer-driven multi-state system with 5 different states and 4 components (two binary components and two three-level components), the probabilities of the different components are independent but not identical.
- `Bin.S-P` is a binary Series–Parallel system taken from [41] (Example 4.5) which has seven independent not identical components and two levels.
- `MAX+MIN,TIMES` is a multi-state system with 5 components and 7 levels whose structure function is given by

$$\Phi(x_1, \dots, x_5) = \left(\max\{x_1, x_2\} + \min\{x_3, x_4\}\right) \times x_5,$$

  and the details on components' and system's levels and probabilities (not i.i.d) are given in [42], Example 4.7.
- `Bridge Flow Network` is a multi-state network with 5 edges with different weights considered as flows. The states of the system are given by the possible flows through the network. The example considers the probability of a total flow of at least three units (*i.e.* the system is in level $j = 3$). The details on states and probabilities are given in [43] Example 4, see also [21] Example 5.14.
- `Dominant MS binary-imaged` system is a multi-state system with three i.i.d. components. Both the system and components have four different states. It is presented as Example 12.21 in [20] to illustrate the concept of multi-state dominant binary-imaged system.

[2] CPU: intel i7-4810MQ, 2.80 GHz. RAM: 16 Gb.

Table 1
Test examples of multi-state systems.

Example	n	M	M_i	$gens(I_j)$
`Army Battle Plan`	4	4	2,2,3,3	2,4,5,5
`Bin.S-P`	7	1	1 $\forall i$	3
`MAX+MIN,TIMES`	5	6	3,2,2,3,2	4,3,4,3,2,1
`Bridge Flow Network`	5	3	3,1,2,1,2	3 for $j = 3$
`Dominant MS binary-imaged system`	3	3	3,3,3	3,2,1
`MS Cons. k-out-of-n`	3	3	3,3,3	1,2,1

Table 2
Reliability computations for the GARR 2008 network.

S-node	T-node	# Minpaths	i.i.d probabilities			Non i.i.d.
			0.9	0.95	0.99	
TO	CT	212	0.977344	0.994704	0.999798	0.994352
TS1	NA	223	0.985203	0.996890	0.999895	0.993917
TO*	CT*	196	0.977428	0.994713	0.999797	–
TS1*	NA*	168	0.975771	0.994486	0.999795	–

- `MS Cons.k-out-of-n` is a multi-state consecutive k-out-of-n system with 3 components and three levels. It is example 12.18 in [20].

4.2. Source to terminal networks

One of the main problems in reliability engineering is Network Reliability, see for instance [21,44] for a comprehensive account and [45] for a recent algorithm. In this problem we consider a network in which one vertex is selected as *source vertex* and one or more vertices are selected as *target vertices*. Each of the connections in the network has a certain probability to be working, and the problem is to compute the probability that there exists at least one source-to-target path composed by operational connections. Usually the networks are binary *i.e.* the system and all of its components have only two possible levels, although the multi-state version has also been considered [46,47].

4.2.1. GARR: Italian research and education network

Our first example is the GARR Italian network. The motivation to use this example is to show the performance of our algorithms in a real-life system that has already been studied in the literature. Fig. 2 shows the official 2008 map of the backbone of the GARR network in Italy, which interconnects universities, research centers, libraries, museums, schools and other education, science, culture and innovation facilities, see http//:www.garr.it. The network was at the time formed by 41 nodes and 52 connections. Table 2 shows the results of some reliability computations in this network. First, we use `TO` as source node and `CT` as terminal node, and then we use `TS1` as source node and `NA` as terminal node. In both cases we consider an identical independent probability p for all the connections and compute the source to terminal reliability for $p = 0.9, 0.95$ and 0.99. The last two rows in Table 2 correspond to the same computations in [21] (Example 5.7), observe that the differences are due to the fact that the authors in [21] use a slightly different network which has 42 nodes and hence some different connections and a different number of minimal paths in each example. Since our algorithms can also treat the case of non-identical probabilities, we assigned probability 0.99 to all 10 Gbps connections (4 connections), 0.95 to all 2.5 Gbps connections (14 connections) and 0.9 to the rest of the connections. The results are shown in the last column of the table. All our computations in this table took less than one second.

4.2.2. Random networks

Our second example is a randomly generated set of networks. This is a convenient way to generate a big number of examples not having a regular structure (like for instance series–parallel systems or k-out-of-n systems and variants), and therefore represents a good set of

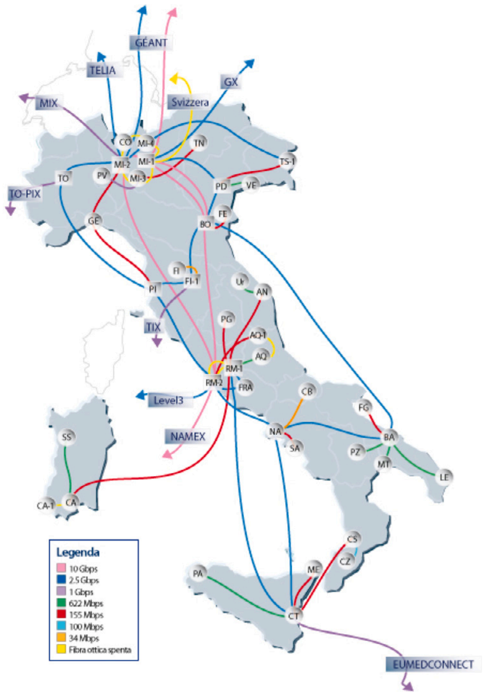

Fig. 2. Map of the GARR network in 2008.

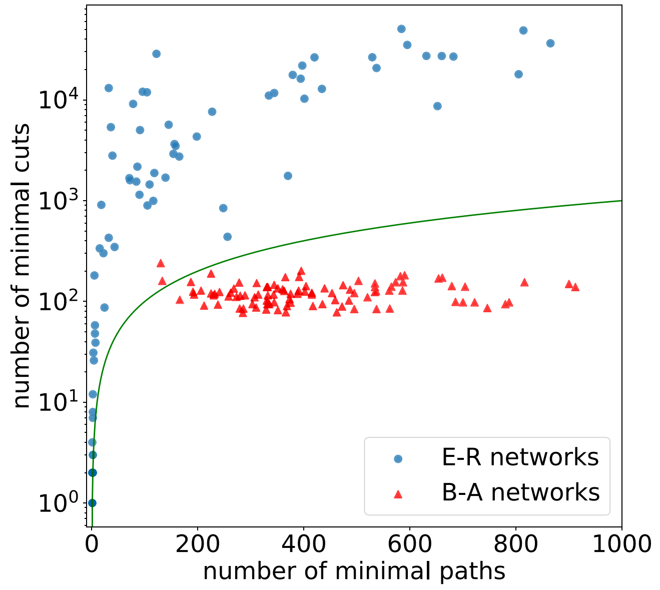

Fig. 3. Number of minimal paths and minimal cuts in Erdős–Rényi and Barabasi–Albert networks. The line indicates number of minimal paths equals number of minimal cuts.

benchmarks for the application to general systems. We demonstrate our algorithms' performance in several random networks generated following the Erdős–Rényi model $ER(n,p)$ [48] and Barabasi–Albert model $BA(n,m)$ [49]. These models generate networks with different characteristics such as degree distribution, modularity, etc. We compute the reliability of 100 random Erdős–Rényi networks with $n = 40$, $p = 0.05$ and 100 Barabasi–Albert networks with $n = 10$ and $m = 4$; we chose randomly one source and one terminal node in each case. The number of minimal paths varies between 100 and 1000 in both cases. However, the relation between the number of minimal paths and minimal cuts is significantly different in the two types of networks. Erdős–Rényi networks tend to have many more minimal cuts with respect to the number of minimal paths, while the situation is the opposite for Barabasi–Albert networks, see Fig. 3. In the case of Barabasi–Albert networks our algorithms compute the reliability of the network using the dual ideal, since it is smaller in most cases. The reliability of the Erdős–Rényi examples was always computed using the minimal path ideal. Times for the computation of the reliability of these networks are shown in Fig. 4. The figures show that the times depend greatly on the number of minimal paths, but the topology of the network influences the algebraic characteristics of the ideals. Observe that there are two cases of Barabasi–Albert graphs in which the number of minimal paths is smaller than the number of minimal cuts and hence the path ideal was used for the computation, which results in higher computation times compared with the cases in which the dual was used. The resolutions of these networks ideals are much shorter in the dual case and hence the results.

4.3. *Multi-state generalized k-out-of-n systems*

Our final example is multi-state generalized k-out-of-n systems. We include this example since they are one of the most important types of systems studied in the reliability engineering literature, both in their binary and multi-state versions. A binary k-out-of-n:G system is a system with n components that is in a working state whenever at least k of its components are working. The multi-state version of this kind of systems has received several definitions in the literature, see [29] for a review. A general definition is that of *generalized multi-state k-out-of-n systems*, see [50]:

Definition 4.1. An n-component systems is called a generalized k-out-of-n:G system if $\phi(s_1,\dots,s_n) > j$, $1 \le j \le M$ whenever there exists an integer value l, $(j \le l \le M)$ such that at least k_l components are in state l or above.

If we denote by N_j the number of components of the system that are in state j or above then this definition can be rephrased by saying that $\phi(S) \ge j$ if

$$\begin{aligned} N_j &\ge k_j \\ N_{j+1} &\ge k_{j+1} \\ &\vdots \\ N_M &\ge k_M. \end{aligned}$$

We have used our C++ class to compute the reliability of several generalized k-out-of-n systems. Since each of these systems is given by a vector $(k_1,\dots,k_M)$ we generated randomly 100 vectors for systems with four levels, and 10 variables. Fig. 5(a) and (b) show the number of minimal paths and minimal cuts of these systems and the computing time of these examples vs. the number of generators used for its computation in each case. All systems considered have components with independent, non-identical working probabilities. The figures show that most of these systems have a smaller number of minimal paths compared to the number of its minimal cuts, and that the computation time depends greatly on the structure of the system. Let us denote by k the maximum of the integers k_l for $l \in \{1,\dots,M\}$. Fig. 6(a) and (b) show the number of minimal paths and cuts for systems with 12 components, 4 levels and $k = 4$, $k = 6$. The number of minimal cuts and paths of multi-state generalized k-out-of-n grows as $\binom{n}{k}$. The performance of our algorithms depend greatly on the number of minimal paths or minimal cuts, as can be seen in Fig. 6(c). There exist specialized algorithms for this kind of systems that are recursive on M, see [29,51] or based on Decision Diagrams [32].

4.4. *Computational complexity*

The algebraic method is (in its general form) an enumerative method, similar to the inclusion–exclusion approach but less redundant. The compact form of the Hilbert series provided by our algorithms

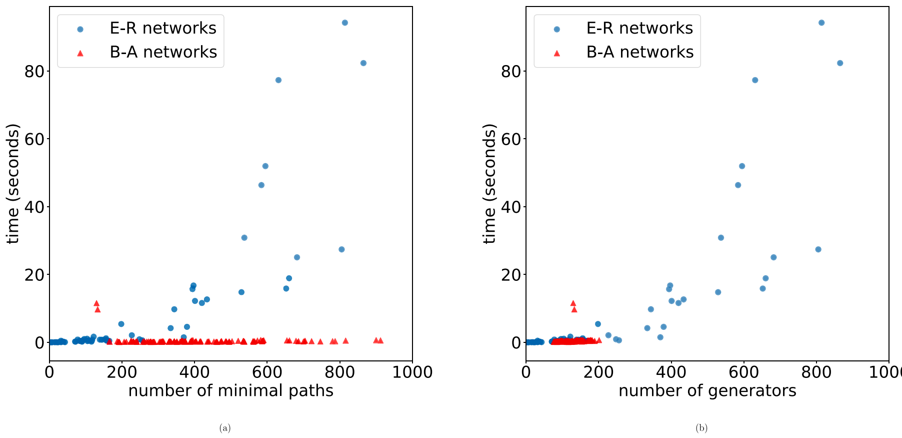

Fig. 4. Times for reliability computation on Erdős–Rényi and Barabasi–Albert graphs.

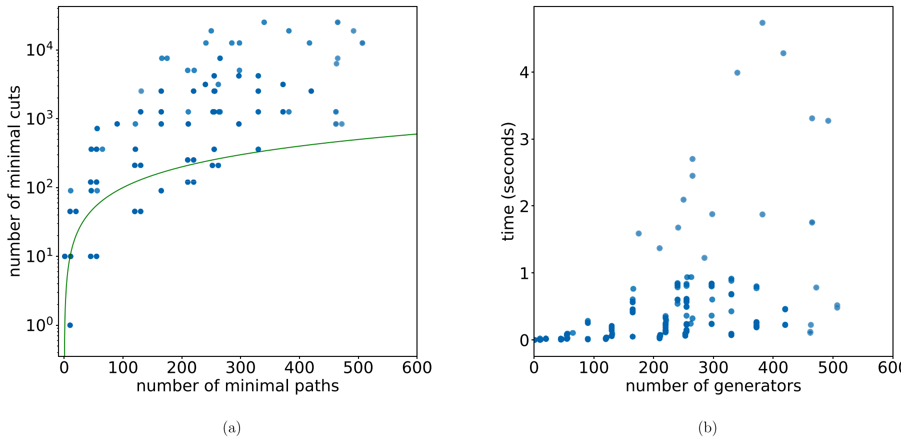

Fig. 5. Number of minimal paths vs. number of minimal cuts and computing time for generalized multi-state k-out-of-n systems with 10 components, 4 levels, and non-identical probabilities.

gives some computational advantages, but there exist certain intrinsic limitations due to the complexity of the problem. The computation of network reliability (either k-terminal, 2-terminal or all-terminal) is a $\#P$-hard problem [52] and hence there is no hope of finding an efficient algorithm for computing the reliability of general systems unless $P = NP$, even in the binary case. The algebraic method in which our algorithms are based shows that the problem of computing the reliability of a multi-state system can be polynomially reduced to the computation of the multigraded Hilbert series of a monomial ideal. This problem belongs to the class of $\#P$-hard problems and there exist several subproblems of it that are known to be $\#P$-complete or NP-complete. In particular, the problem of computing the Euler characteristic of an abstract simplicial complex is equivalent to the computation of the coefficient of the monomial $x_1 \cdots x_n$ in the multigraded Hilbert series of a (square-free) monomial ideal, and this problem belongs to the $\#P$-complete complexity class [53].

There are two complementary directions to follow for finding satisfactory solutions for these problems. One is to develop specialized polynomial algorithms for particular families of systems. The other is to find algorithms showing good heuristic behavior when applied to general problems. In these two directions it is of paramount importance to develop good implementations in terms of data types, memory management, etc. that make the algorithms applicable to practical problems.

The algebraic method for system reliability contributes to both of the directions described above. On the one hand, the study of the structure of the ideals of particular classes of systems can provide efficient algorithms or even formulas (explicit or recursive) for their Hilbert series, see [54,55] for k-out-of-n and consecutive k-out-of-n binary systems, and [29] for the multi-state version. As an example, the formulas for k-out-of-n systems have complexity $O(n^2)$ which is quadratic, but not optimal when restricted to systems with statistically independent components, for which the algorithm in [56], based on the Fast Fourier Transform, runs in complexity $O(n(\log n)^2)$. On the other hand, for the general case we used efficient algorithms for computing the multi-graded Hilbert series of monomial ideals and Alexander duals. These algorithms avoid much of the redundancy that shows up in reliability computation of general systems, when we have no evident structure to take advantage of. Besides, they make use of the recursive nature of the problem, which has also been used in other approaches like the Universal Generating Function method. However, there is still room for improvement. As the UGF and other methods show, it is important, for the sake of efficiency, to identify good base cases for the recursion, and for simplification techniques. The algorithms provided in this paper use only *algebraic* base cases and simplifications, and hence it is expected to gain efficiency by exploring other base cases that arise from the knowledge of system reliability. This is beyond the scope of this paper and is left as future work. Finally, as it is common in computer algebra, implementations of general algorithms which are good enough for NP-hard or $\#P$-hard problems offer good performance in practice. A paradigmatic example of this are the good algorithms for Gröbner bases, a problem whose complexity is known to be doubly exponential. This is the case of the class presented in this paper, in which we took advantage of the data types and optimized routines provided by `CoCoALib` together with good implementations for the Hilbert series and Alexander dual algorithms. This allows us to

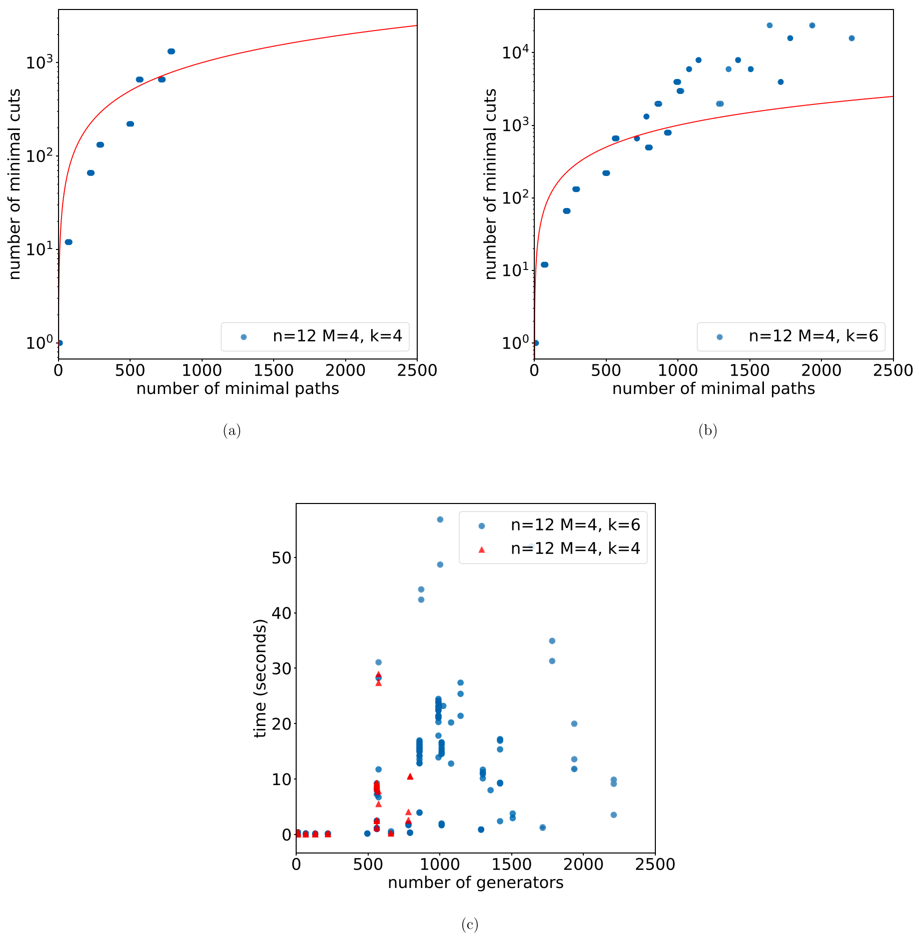

Fig. 6. Number of minimal paths vs. number of minimal cuts and computing time for generalized multi-state k-out-of-n systems with 12 components, 4 levels, $k = 4, 6$, and non-identical probabilities.

efficiently compute the reliability of general systems of big size with an affordable use of time and memory resources.

5. Conclusions and further work

We have presented a C++ class that computes the reliability of multi-state coherent systems. The class is included and distributed with the computer algebra library CoCoALib as free software. The algorithms in this class are based on the algebraic approach to system reliability analysis developed in the last decade. The main advantage of these algorithms is that they can be applied to general systems, that they provide bounds and that can be applied without modification to systems with i.i.d probabilities and systems in which the probabilities are not identical. The main drawback is that this approach is enumerative, in the sense that relies on the enumeration of minimal paths or cuts, which may be impractical for big systems.

Specialized algorithms for particular systems are not easy to find but are very efficient in practice, see for example the algorithms based on Multi-Valued Decision Diagrams for multi-state k-out-of-n systems [32] or the linear algorithm for networks with small treewidth [45]. Our future work includes the design of specialized algebraic algorithms for particular kinds of systems. The structure of particular systems induces a particular structure in the associated ideals which can be studied using algebraic and combinatorial tools allowing the design of more efficient algorithms, as described for instance in [26,27]. Another direction of improvement is to adapt the algorithm for systems with non-independent components. The algebraic theory is exactly the same and only the probability assignment to the computed monomials need to be changed. This would give a wider flexibility to our C++ class.

Finally, further tuning and optimization of the existing algorithms will likely improve their efficiency and reduce the computing times, in particular optimizations coming from the comparison and strong points of other methods, like the UGF method.

The fact that our approach is general makes it useful as one of the default algorithms to try in the cases for which no specific algorithms are known yet, and also as a tool to benchmark new specific algorithms for such problems.

CRediT authorship contribution statement

A.M. Bigatti: Methodology, Software. **P. Pascual-Ortigosa:** Conceptualization, Methodology. **E. Sáenz-de-Cabezón:** Supervision, Conceptualization, Methodology, Software.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

The first author was partially supported by the “National Group for Algebraic and Geometric Structures, and Their Applications” (GNSAGA, INdAM, Italy). The last two authors have been partially supported by grant MTM2017-88804-P from Ministerio de Economía, Industria y Competitividad (Spain).

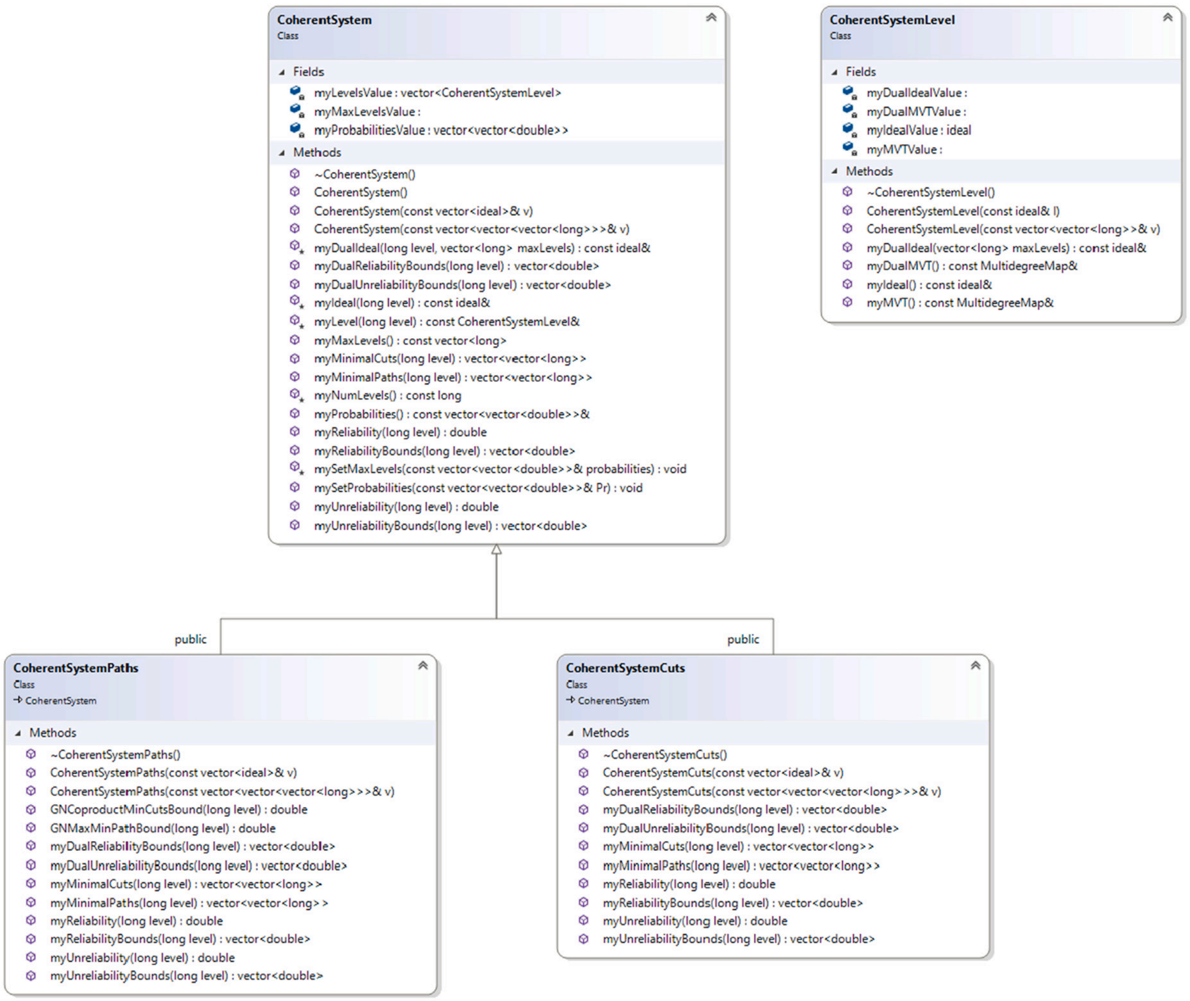
CoherentSystem
Class
Fields
myLevelsValue : vector<CoherentSystemLevel>
myMaxLevelsValue :
myProbabilitiesValue : vector<vector<double>>
Methods
~CoherentSystem()
CoherentSystem()
CoherentSystem(const vector<ideal>& v)
CoherentSystem(const vector<vector<vector<long>>>& v)
myDualIdeal(long level, vector<long> maxLevels) : const ideal&
myDualReliabilityBounds(long level) : vector<double>
myDualUnreliabilityBounds(long level) : vector<double>
myIdeal(long level) : const ideal&
myLevel(long level) : const CoherentSystemLevel&
myMaxLevels() : const vector<long>
myMinimalCuts(long level) : vector<vector<long>>
myMinimalPaths(long level) : vector<vector<long>>
myNumLevels() : const long
myProbabilities() : const vector<vector<double>>&
myReliability(long level) : double
myReliabilityBounds(long level) : vector<double>
mySetMaxLevels(const vector<vector<double>>& probabilities) : void
mySetProbabilities(const vector<vector<double>>& Pr) : void
myUnreliability(long level) : double
myUnreliabilityBounds(long level) : vector<double>
CoherentSystemLevel
Class
Fields
myDualIdealValue :
myDualMVTValue :
myIdealValue : ideal
myMVTValue :
Methods
~CoherentSystemLevel()
CoherentSystemLevel(const ideal& I)
CoherentSystemLevel(const vector<vector<long>>& v)
myDualIdeal(vector<long> maxLevels) : const ideal&
myDualMVT() : const MultidegreeMap&
myIdeal() : const ideal&
myMVT() : const MultidegreeMap&
public
public
CoherentSystemPaths
Class
→ CoherentSystem
Methods
~CoherentSystemPaths()
CoherentSystemPaths(const vector<ideal>& v)
CoherentSystemPaths(const vector<vector<vector<long>>>& v)
GNCoproductMinCutsBound(long level) : double
GNMaxMinPathBound(long level) : double
myDualReliabilityBounds(long level) : vector<double>
myDualUnreliabilityBounds(long level) : vector<double>
myMinimalCuts(long level) : vector<vector<long>>
myMinimalPaths(long level) : vector<vector<long>>
myReliability(long level) : double
myReliabilityBounds(long level) : vector<double>
myUnreliability(long level) : double
myUnreliabilityBounds(long level) : vector<double>
CoherentSystemCuts
Class
→ CoherentSystem
Methods
~CoherentSystemCuts()
CoherentSystemCuts(const vector<ideal>& v)
CoherentSystemCuts(const vector<vector<vector<long>>>& v)
myDualReliabilityBounds(long level) : vector<double>
myDualUnreliabilityBounds(long level) : vector<double>
myMinimalCuts(long level) : vector<vector<long>>
myMinimalPaths(long level) : vector<vector<long>>
myReliability(long level) : double
myReliabilityBounds(long level) : vector<double>
myUnreliability(long level) : double
myUnreliabilityBounds(long level) : vector<double>

Fig. 7. UML diagram of the `CoherentSystem` class.

Appendix. UML diagram of the algebraic reliability C++ class

See Fig. 7.

References

[1] Reliasoft. Available at http://www.reliasoft.com.
[2] ALD. Available at http://reliability-analysis-software.com/.
[3] ITEM software. Available at http://www.itemsoft.com.
[4] Li H-S, Cao Z-J. Matlab codes of subset simulation for reliability analysis and structural optimization. Struct Multidiscip Optim 2016;54:391–410.
[5] Rao S. Reliability engineering. Pearson; 2015.
[6] Reid M. Reliability. A python library for reliability engineering. 2020, Available at http://github.com/MatthewReid854/reliability.
[7] Shaffer LB, Young TM, Guess FM, Bensmail H, León RV. Using R software for reliability data analysis. Int J Reliab Appl 2008;9:53–70.
[8] Sahner R, Trivedi KS, Pullafito A. SHARPE, (Symbolic Hierarchical Automated Reliability and Performance Evaluator), Available at http://sharpe.pratt.duke.edu.
[9] TIOBE. Available at http://www.tiobe.com/tiobe-index.
[10] Abbott J, Bigatti AM. CoCoALib: a C++ library for doing Computations in Commutative Algebra. Available at http://cocoa.dima.unige.it/cocoalib.
[11] Ball M, Provan J. Computing network reliability in time polynomial in the number of cuts. Oper Res 1988;32:516–26.
[12] Ball M, Provan J. Disjoint products and efficient computations of reliability. Oper Res 1988;36:703–15.
[13] Shier D. Network reliability and algebraic structures. Clarendon Press; 1991.
[14] Harms DD, Kratzel M, Colbourn CJ, Devitt JS. Network reliability. Experiments with a symbolic algebra environment. CRC Press; 1995.
[15] Ushakov I. Optimal standby problem and a universal generating function. Sov J Comput Syst Sci 1987;25:61–73.
[16] Levitin G. The universal generating function in reliability analysis and optimization. Springer; 2005.
[17] Lu S, Shi D, Xiao H. Reliability of sliding window systems with two failure modes. Reliab Eng Syst Saf 2019;188:366–76.
[18] Natvig B. Multi-state systems reliability theory with applications. John Wiley & sons; 2011.
[19] El-Neweihi E, Proschan F, Sethurman J. Multi-state coherent systems. J Appl Probab 1978;15:675–88.
[20] Kuo W, Zuo M. Optimal reliability modelling: Principles and applications. John Wiley & sons; 2003.
[21] Trivedi K, Bobbio A. Reliability and availability engineering. Cambridge University Press; 2017.
[22] Giglio B, Naiman DQ, Wynn HP. Gröbner bases, abstract tubes, and inclusion–exclusion reliability bounds. IEEE Trans Reliab 2002;51:358–66.
[23] Giglio B, Wynn HP. Monomial ideals and the Scarf complex for coherent systems in reliability theory. Ann Statist 2004;32:1289–311.
[24] Dohmen K. Improved Bonferroni inequalities via abstract tubes. Springer; 2003.
[25] Sáenz-de-Cabezón E, Wynn HP. Betti numbers and minimal free resolutions for multi-state system reliability bounds. J Symbolic Comput 2009;44:1311–25.
[26] Sáenz-de-Cabezón E, Wynn HP. Hilbert functions for design in reliability. IEEE Trans Reliab 2015;64:83–93.
[27] Mohammadi F, Pascual-Ortigosa P, Sáenz-de-Cabezón E, Wynn H. Polarization and depolarization of monomial ideals with application to multi-state system reliability. J Algebraic Combin 2020;51:617–39.
[28] Boedigheimer RA, Kapur KC. Customer-driven reliability models for multistate coherent systems. IEEE Trans Reliab 1994;43:46–50.
[29] Pascual-Ortigosa P, Sáenz-de-Cabezón E, Wynn H. Algebraic reliability of multi-state k-out-of-n systems. Probab Engrg Inform Sci 2020. http://dx.doi.org/10.1017/S0269964820000224.
[30] Eisenbud D. Commutative algebra with a view towards algebraic geometry. Springer; 1995.
[31] Bigatti AM. Computation of Hilbert-Poincaré series. J Pure Appl Algebra 1997;119:237–53.

[32] Mo Y, Liudong X, Amari SV, Dugan JB. Efficient analysis of multi-state k-out-of-n systems. Reliab Eng Syst Saf 2015;133:95–105.
[33] Sáenz-de-Cabezón E. Multigraded Betti numbers without computing minimal free resolutions. Appl Algebra Eng Commun Comput 2009;20:481–95.
[34] Satyanarayana A, Chang MK. Network reliability and the factoring theorem. Networks 1983;13:107–20.
[35] Miller E, Sturmfels B. Combinatorial commutative algebra. Springer; 2004.
[36] Bigatti AM, Sáenz-de-Cabezón. Computation of the (n-1)-st koszul homology of monomial ideals and related algorithms. In: International symposium on symbolic and algebraic computation, ISSAC'09, Seoul, South Korea, 2009. ACM; 2009, p. 31–7.
[37] Abbott J, Bigatti AM, Robbiano L. CoCoA: a system for doing Computations in Commutative Algebra. Available at http://cocoa.dima.unige.it.
[38] Roune BH. Frobby. Available at http://www.broune.com/frobby.
[39] Gåsemyr J, Natvig B. Improved availability bounds for binary and multistate monotone systems with independent component processes. J Appl Probab 2017;54:750–62.
[40] Funnemark E, Natvig B. Bounds for the availabilities in a fixed time interval for multistate monotone systems. Adv Appl Probab 1985;17:638–65.
[41] Lisnianski A, Levitin G. Multi-state system reliability. World Scientific; 2003.
[42] Lisnianski A, Frenkel I, Ding Y. Multi-state system reliability analysis and optimization for engineers and industrial managers. Springer; 2010.
[43] Shrestha A, Xing L, Dai Y. Decision diagram based methods and complexity analysis for multi-state systems. IEEE Trans Reliab 2010;59:145–60.
[44] Ball M. Computing network reliability. Oper Res 1979;27:823–38.
[45] Goharshady AK, Mohammadi F. Improved availability bounds for binary and multistate monotone systems with independent component processes. Reliab Eng Syst Saf 2020;193.
[46] Bai G, Tian Z, Zuo M. Reliability evaluation of multistate networks: An improved algorithm using state space decomposition and experimental comparison. IISE Trans 2018;50:407–18.
[47] Zhang C, Liu T, Bai G. An improved algorithm for reliability bounds of multistate networks. Comm Statist Theory Methods 2020;49:3772–91.
[48] Erdős P, Rényi A. On random graphs I. Publ Math Debrecen 1959;6:290–7.
[49] Barabási A-L, Albert R. Emergence of scaling in random networks. Science 1999;286:509–12.
[50] Huang J, Zuo MJ, Wu Y. Generalized multi-state k-out-of-n:G systems. IEEE Trans Reliab 2000;49:105–11.
[51] Zuo MJ, Tian Z. Performance evaluation of generalized multi-state k-out-of-n systems. IEEE Trans Reliab 2006;55:319–27.
[52] Ball M. Computational complexity of network reliability analysis: An overview. IEEE Trans Reliab 1986;R-35:230–9.
[53] Roune BH, Sáenz-de-Cabezón E. Complexity and algorithms for the Euler characteristic of simplicial complexes. J Symbolic Comput 2013;50:170–96.
[54] Sáenz-de-Cabezón E, Wynn HP. Betti numbers and minimal free resolutions for multi-state system reliability bounds. Appl Algebra Eng Commun Comput 2010;21:443–57.
[55] Sáenz-de-Cabezón E, Wynn HP. Computational algebraic algorithms for the reliability of generalized k-out-of-n and related systems. Math Comput Simulation 2011;82:68–78.
[56] Belfore L. An $O(n(\log n)^2)$ algorithm for computing the reliability of k-out-of-n : g and & k-to$l$-out-of-n : g systems. IEEE Trans Reliab 1995;R-44:132–6.