Network reliability, performability metrics, rare events and standard Monte Carlo

Gerardo Rubino

Network reliability, performability metrics, rare events and standard Monte Carlo

Gerardo Rubino

Abstract In this chapter, we consider static models in network reliability, that cover a huge family of applications, going way beyond the case of networks of any kind. The analysis of these models is in general #P-complete, and Monte Carlo remains the only effective approach. We underline the interest in moving from the typical binary world where components and systems are either up or down, to a multi-variate one, where the up state is decomposed into several performance levels. This is also called a performability view of the system. The chapter then proposes a different view of Monte Carlo procedures, where instead of trying to reduce the variance of the estimators, we focus on their time complexities. This view allows a first straightforward way of exploring these metrics. The chapter focuses on the resilience, which is the expected number of pairs of nodes that are connected by at least one path in the model. We discuss the ability of the mentioned approach for quickly estimating this metric, together with variations of it. We also discuss another side effect of the sampling technique proposed in the text, the possibility of easily computing the sensitivities of these metrics with respect to the individual reliabilities of the components. We show that this can be done without a significant overhead of the procedure that estimates the resilience metric alone.

Gerardo Rubino
INRIA, Campus de Beaulieu, Rennes e-mail: Gerardo.Rubino@inria.fr

1 Introduction

Network reliability refers to a large family of random graph-based models used in multiple areas of science and technology, for different types of analysis [1]. The reference case, and the one with which we will work in this chapter, is when we use them to analyze a communication network composed of a set $\mathcal{V}$ of nodes connected by a set $\mathcal{E}$ of edges (undirected links, no loops). The graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is called the *underlying graph* in the model. We are then interested in the communications between nodes using paths. Graph $\mathcal{G}$ is assumed to be connected. With each each link i we associate a given fixed probability r_i, and the interpretation is that with probability $1 - r_i$ the link is removed from the graph. More formally, with each link i we associate a Bernoulli or Binary random variable (r.v.) X_i with parameter r_i; this means that $X_i \in \{0, 1\}$ and that $r_i = \mathbb{P}(X_i = 1)$. We call r_i the *(elementary) reliability* of link i and X_i the *state* of link i. Observe that there is no explicit time variable. That is why we call these models *static* [2]. We see the system at a "typical" point in time, or at a specific instant of interest, possibly at infinity, but there is no stochastic process in the model. Denote by $m = |\mathcal{E}|$ the number of edges. Then, and always referring to the reference model, we assume that the m r.v.s $X_1, \ldots, X_m$ are independent.

With this setting, we build a random graph G with values in the set of partial subgraphs of $\mathcal{G}$ (that is, G has the same nodes as $\mathcal{G}$ and a subset of $\mathcal{G}$'s edges). Observe that there are 2^m points in G's space of values. Informally, in a realization of G edge $i \in \mathcal{E}$ exists with probability r_i. More formally, we define the distribution of G by the following expression: for any subset E of $\mathcal{E}$,

$$\mathbb{P}\big(G = (\mathcal{V}, E)\big) = \Pi_{i \in E} r_i \Pi_{j \notin E} (1 - r_j).$$

Using this random object, we can define many *connectivity*-based metrics quantifying different properties of G related to its capability to transport something (information, some fluid, ...) from some nodes to other nodes in the model. By far, the most used one is the so-called *source-to-terminal reliability* defined as follows: two different nodes are fixed, say s and t, and the metric is $R_{s,t} = \mathbb{P}(s \text{ and } t \text{ are connected in } G)$, that is, the probability that s and t belong to the same connected component of G. This is also called the *2-terminal* reliability, or simply the *s-t*-reliability. Another widely used metric is the *all-terminal* reliability, R_{all}, defined by $R_{\text{all}} = \mathbb{P}(\text{all nodes are connected in } G)$, that is, the probability that G is connected (that it has a single connected component). Formally, we usually code the fact that the selected property holds by 1 and that it doesn't hold by 0, and we call *system's state* that random number.

There are many variations around this melody. Instead of looking at the connections between two nodes s and t or between all pairs of nodes, we can define a K-terminal reliability R_K concerning a given subset K of nodes; the corresponding definition is

$$R_K = \mathbb{P}(\text{all nodes in } K \text{ belong to the same connected component of } G).$$

The connectivity criteria can be more complicated. For instance, we can define the *distance-based 2-terminal reliability with parameter d*, $R_{d;s,t}$, where $d \geq 1$ is an integer, by

$$R_{d;s,t} = \mathbb{P}(s \text{ and } t \text{ are connected in } G \text{ by at least a path having length} \leq d),$$

and similarly for R_{all} or R_K [3]. A different extension consists of associating a non-negative integer k_s with node s for all $s \in \mathcal{V}$ and defining a new metric as the probability that there are $\min\{k_s, k_t\}$ elementary edge-disjoint paths between nodes s and t, for all s and t [4]. All these metrics can also be defined in a similar model where the nodes and not the edges, have random states, or both nodes and edges are random, on directed graphs, instead of undirected ones, etc. Recently, some works appeared which remove the independence assumption concerning the state variables [5, 6].

The evaluation of all these reliability metrics is, in general, a computational hard problem [7]. Formally, it belongs to the #*P*–complete class, a family of *NP*–hard problems not known to be in *NP*. A #P-complete problem is equivalent to counting the number of solutions to an *NP*-complete one. For this reason, #*P*–complete problems are, at least, as hard as *NP*–complete ones. Even in very restricted classes of graphs, the computation of the before mentioned metrics remains in this complexity class. For instance, it is shown in [8] that the computation of the source-to-terminal reliability is in the #*P*–complete class even when the graph is planar (in fact, s,t-planar) and has vertex degree at most equal to three. From a practical point of view, this means that a graph with, say, more than one hundred elements (nodes, lines) can not be exactly evaluated (except for special topologies). An efficient alternative is then to use a Monte Carlo method, the topic of this chapter. The standard version consists of generating an N-sample of G, say $G_1, \ldots, G_N$, and to estimate the reliability metric, say $R_{s,t}$, using the standard estimator

$$\widehat{R}_{s,t} = \frac{1}{N} \sum_{n=1}^{N} 1\big(s \text{ and } t \text{ connected in } G_n\big), \tag{1}$$

where $1(P)$ is equal to 1 if the predicate P is true, 0 if it is false. In many cases, this approach can handle medium and large size models, but, of course, it has its own problem, namely the fact that its efficiency is sensitive to the numerical values of data (the r_is). In particular, this standard procedure (sampling N times graph G and checking the property used to define the desired metric on each realization) becomes of no use in the *rare event* case, that is, when the system reliability metric is (very) close to one. Unfortunately, this is often the interesting case.

For the reader unfamiliar with the area, let us finally comment that the specific source-to-terminal model has a huge application area in the dependability analysis of many kinds of systems. The general literature is rich in examples. Let us just mention [9], where bounds of the metric $R_{s,t}$ are proposed, applied to the analysis of critical control subsystems of a modern aircraft.

There is a very large literature in the network reliability area. A good review of the main concepts and results is [1], and an older and classical text, [10]. Another already mentioned survey is ours [2]. Specifically in the Monte Carlo side, some papers by Pierre L'Ecuyer and co-authors are [11, 12, 13]. See also [14] on the rare event side, in particular Chapter 7, or [15] for many technical aspects.

In the sequel of this chapter, we explore in Section 2 the interest of using *performability* metrics to quantify the capacity of a network to provide service in spite of failures (or equivalent events) of its components, and we focus on one of them, called *resilience*. Then, in Section 3, we discuss the estimation of the resilience using the naive Monte Carlo approach. After commenting the rare event problem, we show that it is possible to be efficient in the use of the naive, or standard, or crude estimator, by means of a smarter implementation (this extends old work presented in [16]). In Section 4, we illustrate the use of the previously discussed implementation, and we show that it allows exploring other types of metrics, written as variations on the resilience idea. We also show that the same approach makes straightforward to evaluate also the gradient of the resilience with respect to the elementary reliabilities, with almost no overhead (extension to ideas presented in [17]). Section 5 concludes the chapter.

2 Performability metrics and resilience

Quantitative analysis of systems is mainly done following two different viewpoints. Either we assume the system perfect (that is, we ignore its possible failures and repairs) and we focus on measures related to the work it does, on the properties of the service it provides, or we ignore the latter and we focus on those possible failures and repairs, and try to analyze different related properties of the system. Around the 80s appeared another idea, in general a much more complex one, where we look simultaneously, that is, in the same model, at both sides of systems' behavior. Following the pioneering work done by Meyer [18], we call it *performability*. We will consider this view here, based on the reference model presented before.

So, we have an undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ assumed to be connected and with no loops, with n nodes and m edges (the underlying graph), and our standard probabilistic setting parameterized by the elementary reliabilities $r_1, \ldots, r_m$. Consider now the R_{all} metric, widely used to quantify the ability of the network to support communications between all pairs of nodes. It measures a binary property, an all-nothing one, the probability that all nodes can communicate. But for the system manager, or for the users, the fact that, for instance, a single node is isolated, making that among the total number $n(n-1)/2$ of pairs of nodes, only $n-1$ of them can't communicate, is very different than the situation where many links are failed making that just a few pairs of nodes can talk to each other (possible none of them if all links are failed). A way of measuring this is by directly looking at the number of pairs that can communicate.

We thus define the central r.v. of this chapter,

$$NCP = \text{number of pairs of nodes that can communicate in } G.$$

The analysis of this r.v. provides a much richer information about the capacity of the network to provide communications between its nodes. We will focus on its expectation in the sequel.

2.1 The resilience metric

The expectation of *NCP* has been called *resilience* in some works [19], in the same context as in this chapter. It has been explored in a few other works [20, 21], always in the setting discussed here. So, formally, the resilience *Res* of our model is

$$Res = \mathbb{E}(NCP). \tag{2}$$

Observe first that we have the following immediate properties:

- $0 \le NCP \le \binom{n}{2} = \dfrac{n(n-1)}{2}$;
- $\mathbb{P}(NCP = 0) = \Pi_{i\in\mathcal{E}}(1 - r_i)$;
- $\mathbb{P}\left(NCP = \binom{n}{2}\right) = R_{\text{all}}$;
- $Res|_{\forall i \text{ in } \mathcal{E},\, r_i=0} = 0, \qquad Res|_{\forall i \text{ in } \mathcal{E},\, r_i=1} = \binom{n}{2}$.

We can normalize the metric, dividing $\mathbb{E}(NCP)$ by $\binom{n}{2}$, thus leading to an index in $[0, 1]$. The *scaled resilience* of the network, *ResScaled*, is then $ResScaled = 2Res/\big(n(n-1)\big)$.

Let us provide first a few simple examples, assuming, to simplify the presentation, that we are in the i.i.c. case, meaning independent and identical components (here, we refer to the links) with common elementary reliability r.

Bridge. Take first a bridge, as depicted in Fig. 1. Using brute force (listing all 2^5 possible links' states), we get $Res = r\big(5 + 8r - 14r^3 + 7r^4\big)$.

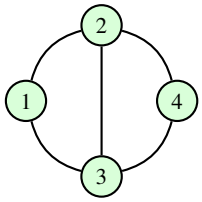

Fig. 1 The bridge topology.

Paths. Consider now a path with n nodes (thus, with $n - 1$ edges), $n \ge 2$. Denote by Res_n its resilience. Conditioning on the state of any of the 2 extreme edges, we

can verify, after some algebra, that

$$Res_n = \frac{r\big[n(1-r) - (1-r^n)\big]}{(1-r)^2} = \frac{r\big[n - 1 - r(n - r^{n-1})\big]}{(1-r)^2}.$$

A few first values: ($Res_1 = 0$,) $Res_2 = r$, $Res_3 = 2r + r^2$, $Res_4 = 3r + 2r^2 + r^3$.

Rings. Last, consider a ring with $n \geq 3$ nodes (or $n \geq 2$ if we accept the multigraph corresponding to the $n = 2$ case). Denote again Res_n its resilience. Conditioning on the state of any of its links, we obtain

$$Res_n = nr\left(\frac{1 - r^{n-1}}{1-r} - \frac{n-1}{2}r^{n-1}\right).$$

A few first values of Res_n: ($Res_1 = 0$,) $Res_2 = 2r - r^2$, $Res_3 = 3r + 3r^2 - 3r^3$, $Res_4 = 4r + 4r^2 + 4r^3 - 6r^4$.

2.2 Some properties of resilience

The "first moment principle" says that if $X \geq 0$ is an integer r.v., then it holds that $\mathbb{P}(X > 0) \leq \mathbb{E}(X)$. Applied to $\mathbb{E}\big(NCP\big)$, this means that $1 - \Pi_{i\in\mathcal{E}}(1 - r_i) \leq Res$. However, this is not very good, because Res is, in general, > 1, and $1 - \Pi_{i\in\mathcal{E}}(1 - r_i)$ is ≤ 1.

Now, simply by definition, $\binom{n}{2}R_{\text{all}} \leq Res \leq \binom{n}{2}$, which are better bounds, especially in the rare event case. For the scaled resilience, we have $R_{\text{all}} \leq ResScaled \leq 1$.

Now, define the r.v.

$$Y_{s,t} = 1\big(\text{there is a path connecting nodes } s \text{ and } t\big).$$

We have $R_{s,t} = \mathbb{P}(Y_{s,t} = 1) = \mathbb{E}(Y_{s,t})$ and also

$$NCP = \sum_{\text{all nodes } s,t,\ s<t} Y_{s,t},$$

from which, taking expectations, we have the following important relation, making the connection between this performability approach and the classical binary metrics:

$$Res = \sum_{\text{all nodes } s,t,\ s<t} R_{s,t}. \tag{3}$$

Conditioning with respect to the state X_i of any edge i, we can write

$$Res(\mathcal{G}) = \mathbb{E}(NCP(\mathcal{G}) \mid X_i = 1)r_i + Res(\mathcal{G}_i^d)(1 - r_i), \tag{4}$$

where we use the notation $Res(H)$ to make explicit the fact that we define the metric on graph H, and where $\mathcal{G}_i^d$ is the graph obtained by removing edge i from $\mathcal{G}$. If

$\mathcal{G}_i^c$ denotes the graph obtained by contracting edge i in $\mathcal{G}$, here we don't have the usual fact that $Res(\mathcal{G}_i^c)$ coincides with $\mathbb{E}(NCP(\mathcal{G}) \mid X_i = 1)$, a relation that holds for classical metrics such as $R_{s,t}$. This makes that Relation (4) is less useful than for standard metrics, where it is the basis of the most powerful approach for their exact numerical analysis. This is because of the more complex structure of $Res(\mathcal{G}_i^c)$ related to $Res(\mathcal{G})$. This issue is out of the scope of this paper, so, we will not pursue commenting it here.

In the rest of this paper, we will look at the evaluation of this expectation using Monte Carlo.

3 Using standard Monte Carlo for resilience-based analysis

First of all, consider reliability metrics such as $R_{s,t}$ or R_{all} (or any of their many variations). Since these metrics are typically close to 1, it is better to work with the complementary events and target the corresponding unreliability values, which we will denote here by γ (for instance, $1 - R_{s,t}$, or $1 - R_{\text{all}}$). The fact that now the target γ is close to 0 makes clear that the correct way to look at the quality of these estimations is through the analysis of relative errors and not absolute ones.

3.1 The standard estimator

Estimating an unreliability metric such as $1-R_{s,t}$ or $1-R_{\text{all}}$, say generically $\gamma = 1-R$, and using the standard or crude estimator, means sampling the following r.v.:

$$\widehat{\gamma} = \frac{1}{N} \sum_{n=1}^{N} Y^{(n)},$$

where $Y^{(1)}, \ldots, Y^{(N)}$ are N independent copies of the Binary r.v. Y equal to 1 if the considered connectivity property doesn't hold, and to 0 otherwise (Y is the complementary of what we called the system's state). We have $\mathbb{E}(\widehat{\gamma}) = \mathbb{E}(Y) = \gamma$ (that is, $\widehat{\gamma}$ is unbiased). A pseudo-code corresponding to this estimator is as follows:

```
Algorithm A // naive implementation of standard Monte Carlo
            // goal: network unreliability
// the connectivity property used is denoted here CP
set counter to 0;
execute N times:
   sample G;
   if the CP doesn't hold: counter++
return counter/N
```

Using the states of the links $X_1, \ldots, X_m$, this means that we repeat N times the following: for each link $i \in \mathcal{E}$, we sample a Binary r.v. having parameter r_i, and we put the result in x_i; then, we remove from the underlying graph $\mathcal{G}$ those links j such that $x_j = 0$; next step is to execute a procedure to check the connectivity property used, for instance, to check if the obtained partial graph of $\mathcal{G}$ is connected or not; at the end, we return the number of times the graph wasn't connected divided by N, which is an estimator of $\gamma = 1 - R_{\text{all}}$.

Observing first that we have $\mathbb{V}(\widehat{\gamma}) = \gamma(1-\gamma)/N$, the corresponding standard confidence interval is $\left(\widehat{\gamma} \mp z(\varepsilon)\sqrt{\widehat{\gamma}(1-\widehat{\gamma})/(N-1)}\right)$, where $1-\varepsilon$ is the confidence level, or ideal coverage probability, with

$$z(\varepsilon) = \Phi^{-1}\left(1 - \frac{\varepsilon}{2}\right), \qquad \Phi(x) = \frac{1}{\sqrt{2\pi}} \int_0^x e^{-t^2/2} \mathrm{d}t$$

(e.g., $z(0.05) \approx 1.96$ for a confidence level of 95%). The quantity $\widehat{\gamma}(1-\widehat{\gamma})/(N-1)$ is proportional to half the confidence interval length (the proportionality constant being $z(\varepsilon)$), and can thus be interpreted as an absolute error. The ratio

$$\frac{\sqrt{\dfrac{\widehat{\gamma}(1-\widehat{\gamma})}{N-1}}}{\widehat{\gamma}} = \sqrt{\frac{1-\widehat{\gamma}}{(N-1)\widehat{\gamma}}} \approx \frac{1}{\sqrt{N\widehat{\gamma}}}$$

can then be seen as an estimation of the (statistical) relative error we have, and we conclude by saying that it goes to ∞ as the rarity of the event of interest increases, that is, as $\gamma \to 0$. This formally show that as the system's failure becomes rarer, the estimation procedure becomes less accurate, and this happens without any bound.

The point to be made next is that the estimator $\widehat{\gamma}$ can be computed in a more efficient way, especially in the frequent case where the r_is are all close to 1, and this leads to efficient estimations, because what matters is to have a small product "mean running time × variance of the estimator" [22]. This is the topic developed in next subsection.

3.2 The standard estimator efficiently implemented in the rare event case

Imagine we implement the standard estimator of γ as follows. Instead of sampling G a huge number N of times and checking the connectivity property on each sample, we sample N times each X_i and put the N results in column i of a big table. The table has $m+1$ columns, one for each link plus a last one for the complement of the system's or network's state, and N rows. Then, we read the table by rows and we fill the last $(m+1)$th column checking the system's state. Since, of course, we will not implement such a table, we will call it the *virtual* table from now on.

Now, assume for a moment that $N = \infty$, and consider column i. Since r_i is usually close to 1, the column has a majority of 1s. The position of its first 0 is a geometric r.v. F_i with values in the strictly positive integers and distribution given by $\mathbb{P}(F_i = h) = r_i^{h-1}(1 - r_h)$, $h \geq 1$. The discrete point process defined by the positions of the different 0s in that column is a Bernoulli process, where the distance between consecutive 0s (and from the first 0 and an auxiliary initial and empty row 0 in the table) are independent copies of F_i. Now, let $F = \min\{F_1, \ldots, F_m\}$ be the first row where there is at least one 0 in the first m columns. We have that the r.v. F is also geometric, with values on the integers ≥ 1, and law given by $\mathbb{P}(F = \ell) = q^{\ell-1}(1 - q)$, $\ell \geq 1$, where $q = r_1 r_2 \cdots r_m$. And in the same way as for column i, if we look at those rows in the infinite virtual table where at least one link is down, the distance between consecutive rows with that property are the points of a discrete Bernoulli process with parameter q (that is, if U_k is the kth such row, then the sequence $(U_1, U_2 - U_1, U_3 - U_2, \ldots)$ is i.i.d. with the law of F. The interest of this type of description of the estimation procedure is that the statistical properties of the estimator used are those of the standard one, in particular, the variance.

This suggests a different way of implementing the standard estimator of γ. Sample F and add the obtained value to a first register *rows*. In the first $F - 1$ rows of the table, we have only 1s in the first m columns, so, the r.v.s $Y_1, \ldots, Y_{F-1}$ are 0 (technically, we assume *coherent* systems [2], so, they are always up if all the components are up). It remains to see what happens with the Fth row. For that purpose, we must sample the X_is, but now knowing that there is at least one zero in the vector of links' states. Consider the natural ordering $1, 2, \ldots, m$ of the network's links, denote by Z the r.v. "number of zeros in the vector of links' states", and by J the index of the first zero in that vector. We have then the following immediate result: for $j = 1, 2, \ldots, m$,

$$\mathbb{P}(J = j \mid Z \geq 1) = \frac{r_1 r_2 \cdots r_{j-1}(1 - r_j)}{1 - r}. \tag{5}$$

Then, we sample from this distribution, obtaining some $j \in \mathcal{E}$, we set the states of links $1, 2, \ldots, j - 1$ to 1 and that of link j to 0, and we independently sample the states of links $j + 1, \ldots, m$ using their original Binary distributions with parameters $r_{j+1}, \ldots, r_m$ respectively. Once known the states of all the links, we check if the connectivity property used holds or not. In the latter case, we increase a second register *counter*. We repeat the procedure, sampling a new value from F's law, obtaining some integer $f \geq 1$, adding it to *rows*, and repeating the conditional sampling of the links knowing that there is at least a 0 in the vector of links' states, until updating *counter*. We stop the process when we reach the condition *rows* $\geq N$, and we return the estimation *counter*$/N$. In pseudo-code, this gives Algorithm B.

```
Algorithm B // 2nd implementation of standard Monte Carlo
            // goal: network unreliability
// the connectivity property used is denoted here CP
rows = 0; counter = 0;
while rows < N: // main loop
   f = sampleFrom(F);
```

```
   rows += f;
   if rows ≤ N:
      conditionally sample vector of links' states;
      if CP doesn't hold: counter++
   // end of main loop
return counter/N
```

Recall that the instruction "`sample vector of links' states`" means sampling first J using (5), obtaining some value j, then setting $X_1 = X_2 = \cdots = X_{j-1} = 1$, $X_j = 0$, and sampling each remaining state variable X_k, for $k = j+1, \ldots, m$, according to the Binary law with parameter r_k, and independently from each other.

Complexity analysis. Consider first Algorithm A. Sampling G needs $\Theta(m)$ in computation time. Testing the connectivity property used has a cost $O(n+m)$ in most of the cases mentioned in the chapter, in particular, for the resilience (see below) and since $m \geq n-1$ because the underlying graph is connected, we can also write it $O(m)$. Since we iterate N times, the total mean computational cost is $O(Nm)$.

In Algorithm B, we will sample from F, for large N (almost always the case), on average, $N/\mathbb{E}(F)$ times. We have $\mathbb{E}(F) = (1-q)^{-1}$. The conditional sampling procedure of (5) plus the test of the selected connectivity property costs $O(m)$, so, the total average computational cost is $O(N(1-q)m)$. The time reduction in B with respect to the naive implementation of A is, then,

$$\frac{Nm}{N(1-q)m} = \frac{1}{1-q}.$$

This approach alone is modest in time reduction, but it can be improved (see below, in Subsection 3.4).

Remark 1 Observe that what we are doing is somehow inspired by a conditional Monte Carlo technique. The interest of the type of implementation of the estimation procedure followed here is that we can see it as a different algorithm for the evaluation of the standard estimator, and this can be useful, as shown below. It will be exploited for quickly exploring other possible metrics of the resilience type, and also to estimate sensitivities (gradients). The central interest in the conditioning we used is that its effect can be seen as a significant reduction in the sampling process of the standard estimator, but not on the estimator's variance, that is, in the execution time instead.

Let us come back first to the estimation of the resilience.

3.3 Estimating the resilience

Recall that the resilience is defined as the expected number of pairs of nodes that can communicate, in the random graph G, $\mathbb{E}(NCP)$ (Relation (2)). Observe that if the graph is connected, we have $NCP = n(n-1)/2$. Suppose now that it is not, and

let NCC be the number of connected components of G. Denote the associated node sets as $\mathcal{S}_1, \ldots, \mathcal{S}_{NCC}$ and their cardinalities by $C_1, \ldots, C_{NCC}$. We then obviously have

$$NCP = \sum_{h=1}^{NCC} \binom{C_h}{2} \tag{6}$$

where $\binom{h}{k} = 0$ when $h < k$. Since computing NCC and the cardinalities $C_1, \ldots, C_{NCC}$ needs only a DFS (Depth First Search) or a BFS (Birth First Search) [23] on G, the cost of computing NCP is $O(m)$. So, the procedure for estimating Res is basically the same in both Algorithms A and B: the only change to do is that instead of checking one of the considered connectivity properties, we have just to decompose G into its connected components and use (6). For instance, the resilience version of Algorithm B is as follows.

```
Algorithm Bbis // 2nd implementation of standard Monte Carlo
               // goal: network resilience
rows = 0; counter = 0;  sum = 0;
while rows < N: // main loop
   f = sampleFrom(F);
   if rows + f ≤ N:
      sum += (f − 1)n(n − 1)/2;
      conditionally sample vector of links' states with (5);
      find the connected components of G;
      sum += NCP of G using (6)
   else
      sum += (N − f)n(n − 1)/2
   rows += f
   // end of main loop
return counter/N
```

Formally, we are using the standard estimator

$$\widehat{Res} = \frac{1}{N} \sum_{n=1}^{N} Res^{(n)},$$

where $Res^{(n)}$ is $Res(G_n)$. A confidence interval for Res with confidence level of, say, 95%, is then given by $(\widehat{Res} \mp 1.96\, S)$, where S^2 is the standard estimator of the variance of $\widehat{Res}$. We have

$$S^2 = \frac{1}{N(N-1)} \sum_{n=1}^{N} NCP^{(n)\,2} - \frac{1}{N-1} \widehat{Res}^{\,2}.$$

3.4 Improving Algorithm B

Assume you know, for instance, the *breadth* b of the underlying graph, that is, the size of a mincut of minimal size (a set of edges whose removal disconnect the graph [23]). This number can be computed using a max flow algorithm such as Edmonds-Karp's, and its cost is polynomial in (n, m). For instance, suppose that $b = 2$. This means that to transform $\mathcal{G}$ into an unconnected graph by removing edges, you must remove at least 2 edges. This means that instead of checking the connectivity property used when there is at least one zero in the row of the virtual table, or instead of decomposing G into its connected components in the resilience case, we can first just count how many zeros we have, and if the number is less than b, we know that the current realization of G is connected, so, *NCP* is equal to $n(n-1)/2$. This will make a supplementary gain in performance, needing only to compute the graph breadth, which is a one shot polynomial task, executed only at the beginning. The corresponding algorithm is presented again in pseudo-code, this time for estimating the resilience.

```
Algorithm C // 3rd implementation of standard Monte Carlo
            // goal: network resilience
b = breadthOf(G);
rows = 0; counter = 0; sum = 0;
while rows < N: // main loop
   f = sampleFrom(F);
   if rows + f ≤ N then
      sum += (f − 1)n(n − 1)/2;
      conditionally sample vector of links' states with (5);
      if its # of zeros is ≤ b then
         sum += n(n − 1)/2
      else
         find the connected components of G;
         sum += NCP of G computed using (6)
   else
      sum += (N − f)n(n − 1)/2
   // end of main loop
return sum/N
```

For the improvement in reducing the computing time of the estimation process, let us wait until next section, where some numerical illustrations are given.

Comment. The way we presented Algorithm C is to underline that it improves version B (or Bbis). But it can be simplified, by redefining the r.v. F. Instead of being the first row of the virtual table with at least 1 zero, we can define it as the first row of the virtual table with at least b zeros. This means then to use again Algorithm B but with this new conditioning. There are other possibilities, using a covering tree of $\mathcal{G}$, or several ones, etc. To discuss the idea here we will just keep the very simple

idea of counting the 0s in the vector of links' states, that is enough to describe the methodology. More on this in Section 4, but let us look now at sensitivities.

3.5 Sensitivity analysis

The last element of the discussion given in this chapter concerns the sensitivity analysis of resilience. This means computing the gradient of *Res* seen as a function of the N variables $r_1, \ldots, r_m$, that is, the partial derivatives $\partial Res/\partial r_i$ for all $i \in \mathcal{E}$. This subsections builds on the results presented in [17], extended to handle the resilience concept.

Define

$$\sigma_i = \frac{\partial Res}{\partial r_i} \qquad \text{and} \qquad \sigma_{s,t;i} = \frac{\partial R_{s,t}}{\partial r_i}.$$

Denote, similarly as for the reliability metrics,

$$Res_i^c = Res(\mathcal{G} \mid X_i = 1) \quad (\neq Res(\mathcal{G}_i^c))$$

and

$$Res_i^d = Res(\mathcal{G} \mid X_i = 0) = Res(\mathcal{G}_i^d).$$

Then, the next result is the first step toward the estimation of the gradient of the resilience.

Proposition 1 *We have*

$$\sigma_i = \frac{Res_i^c - Res}{1 - r_i} = \frac{Res - Res_i^d}{r_i}.$$

Proof Let us show the steps to prove the first equality. The second one is similar. In (3) we stated the basic identity

$$Res = \sum_{\text{all nodes } s,t,\ s<t} R_{s,t}.$$

Taking partial derivatives with respect to r_i, we get

$$\sigma_i = \sum_{\text{all nodes } s,t,\ s<t} \sigma_{s,t;i}.$$

In [17], we proved that

$$\sigma_{s,t;i} = \frac{R_{s,t;i}^c - R_{s,t}}{1 - r_i} = \frac{R_{s,t} - R_{s,t;i}^d}{r_i}. \tag{7}$$

Using the first equality, we have

$$\begin{aligned}\sigma_i &= \sum_{\text{all nodes } s,t,\ s<t} \sigma_{s,t;i} \\ &= \sum_{\text{all nodes } s,t,\ s<t} \frac{R^c_{s,t;i} - R_{s,t}}{1-r_i} \\ &= \frac{1}{1-r_i}\left(\sum_{\text{all nodes } s,t,\ s<t} R^c_{s,t;i} \ - \sum_{\text{all nodes } s,t,\ s<t} R_{s,t} \right) \\ &= \frac{Res^c_i - Res}{1-r_i}.\end{aligned}$$

The second equality follows when using the second one in (7). □

Now, in the same paper [17], it was shown that

$$\widehat{\sigma}_{s,t;i} = \frac{X_i - r_i}{r_i(1-r_i)} Y_{s,t}$$

is an unbiased estimator of $\sigma_{s,t;i}$.

Remark 2 Observe that these estimators fit precisely the idea of having at our disposal the virtual table filled with the values of the states of all links and the corresponding value of *NCP*, at each iteration of the standard method. Actually, the idea of seeing a conditional Monte Carlo approach as described here appeared at the same time that we developed the estimators of sensitivities of classical network reliability metrics of the type presented in the chapter.

Observe that evaluating $\widehat{\sigma}_{s,t;i}$ adds almost no overhead to the estimation of the reliability metrics considered (classical or of the performability type).

Based on previous result, we have the following one.

Theorem 1 *The expression*

$$\widehat{\sigma}_i = \frac{X_i - r_i}{r_i(1-r_i)} NCP$$

defines an unbiased estimator of σ_i.

Proof In a straightforward manner,

$$\begin{aligned}\widehat{\sigma}_i &= \frac{X_i - r_i}{r_i(1-r_i)} \sum_{\text{all nodes } s,t,\ s<t} Y_{s,t} \\ &= \frac{1}{r_i(1-r_i)} \left(\sum_{\text{all nodes } s,t,\ s<t} X_i Y_{s,t} \ - \ r_i \sum_{\text{all nodes } s,t,\ s<t} Y_{s,t} \right).\end{aligned}$$

Taking expectations, and using the fact that $\mathbb{E}(X_i Y_{s,t}) = r_i R^c_{s,t;i}$, we get

$$\mathbb{E}(\widehat{\sigma}_i) = \frac{1}{r_i(1-r_i)} \left(\sum_{\text{all nodes } s,t,\ s<t} r_i R^c_{s,t;i} \quad - \quad r_i Res \right) = \frac{Res^c_i - Res}{1-r_i} = \sigma_i,$$

by means of the result of Proposition 1. □

As we can see, the estimator $\widehat{\sigma}_i$ consists basically of sampling simultaneously the states of the links and the *NCP* variable, something explicit in the virtual table. So, estimating the sensitivities of the resilience with respect to the elementary reliabilities in the system is straightforward following the same approach as for the resilience itself. Next section gives a few examples and discusses other possible resilience-like metrics easily analyzed using the procedure we have described, including the sensitivities.

4 Examples and discussions

In this section, to simplify the presentation we will only consider the i.i.c. case (independent and identical components) in the models, that is, the case of homogeneous links, all sharing the same elementary reliability, denoted here by r.

To provide a few illustrations of previous results, let us consider the model depicted by its underlying graph in Figure 2.

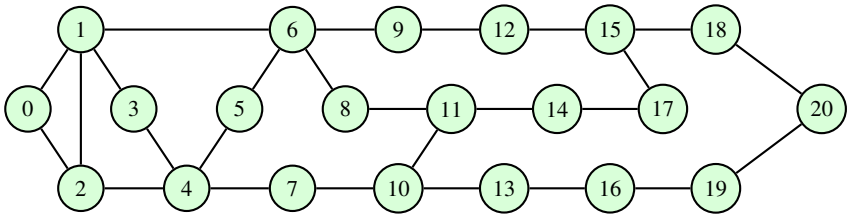

Fig. 2 A widely used Arpanet topology in network reliability, from the history of this famous communication network.

This network has $n = 21$ nodes and $m = 26$ links. Its breadth is $b = 2$. The expected reduction in the execution time, using the basic Algorithm B and written as a factor, is $1 - q$, $q = r^m = r^{26}$. Taking $r = 0.9999$ we obtain a time reduction denominator of ≈ 385. Using the breadth (Algorithm C), we have $b = 2$. The supplementary factor to add is $\mathbb{P}(Z \geq 2 \,|\, Z \geq 1)$, where Z denotes the number of zeros in a row, a Binomial r.v. with $\mathbb{P}(Z = z) = \binom{m}{z}(1-r)^z r^{m-z}$. The supplementary factor is then $\mathbb{P}(Z \geq 2 \,|\, Z \geq 1) = \mathbb{P}(Z \geq 2)/\mathbb{P}(Z \geq 1)$, and $\mathbb{P}(Z \geq 1) = 1 - \mathbb{P}(Z = 0) = 1 - q$, so, the new global average time reduction factor becomes $\mathbb{P}(Z \geq 2) = 1 - r^m - mr^{m-1}(1-r) \approx 3.2448 \cdot 10^{-6}$, which corresponds to a denominator of about $300,000$. This shows that the time reduction can be considerable even when using such a simple conditioning. The accelerations we observed in practice have

been lower than the theoretical factors, probably due to the implementations of the algorithms we had, but in all cases they were important. Using only Algorithm Bbis, in the case of $r = 0.999$, we got accelerations of several hundreds, depending on the tools used (`C` code, `python` alone, `python` with `numpy`). The accelerations were higher using the breadth, but there are other possibilities in the same spirit and still more powerful. Let us give another example here.

Assume we computed a covering tree of the underlying graph, that is, a tree sharing with the underlying graph all its nodes. If we denote the tree as $\mathcal{T} = (\mathcal{V}, \mathcal{E}')$ the probability that all its links are up is $\prod_{i \in \mathcal{E}'} r_i$. In case of that event, we know that $NCP = n(n-1)/2$, so, we can use the event to get a better reduction factor than $1 - q = 1 - r^m$. The sampling of the links' states conditioned to the fact that all links in the covering tree are working is straightforward, only the links out of the tree must be sampled. In the Arpanet model given in Figure 2, the gain is modest. See first a covering tree of this graph in Figure 3.

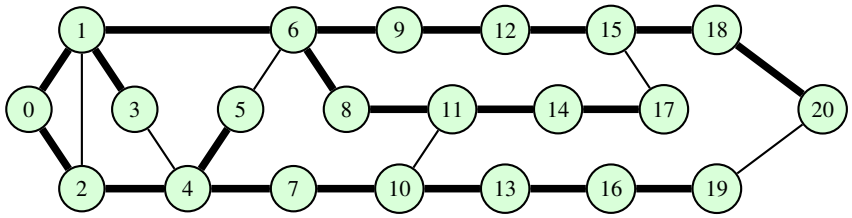

Fig. 3 A covering tree on the Arpanet model of Figure 2.

In the case of Algorithm B with $r = 0.9999$, we had a time reduction factor of about 385. Using a covering tree, always in the i.i.c. case, the factor changes to about $1 - r^{20} \approx 500$. If the graph is more dense, we can have several edge-disjoint covering trees and considering their union we can reach much higher reductions in time. We can also use mincuts with the same purpose. This will not be explored further here, we intend to show only the general principles.

In the general heterogeneous case, when the elementary reliabilities are not necessarily all equal, we can look for covering trees with minimal total weight, where the weight of the tree is *the product* of the reliabilities of its links. For that purpose, we can use standard graph algorithms such as Prim's or Kruskal's (see any good reference in the area, such as [23]). Both have complexity $O(m \ln m)$ where m is the number of edges in the graph.

Concerning sensitivities, we only illustrate the main interest in their evaluation, a point underlined already in [17]. Consider the example in Figure 4.

We see that, from the connectivity point of view and when we have an i.i.c. model (homogenous links), the link $\{4, 5\}$ is the most important one, the first one to improve if possible. For example, in the case of $r = 0.999$, we have the estimations given in Table 1.

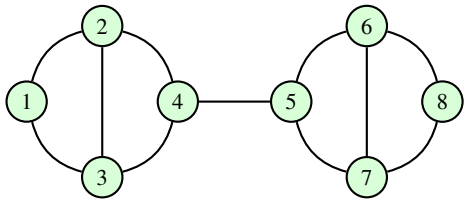

Fig. 4 Two bridges connected by a "bridge" link.

Table 1 A few sensitivities of the resilience, on the model depicted in Figure 4. In the estimations, the absolute error is less than half the last significant digit (confidence level: 95%).

edge	$\{1,2\}$	$\{2,3\}$	$\{4,5\}$
sensitivity	$7.01\ 10^{-3}$	$3.79\ 10^{-5}$	16.0

This illustrates well the fact that the order induced on the links by the gradient of the resilience is a much more robust result than the metric itself. Of course, the example has been chosen to amplify the effect of having links playing a more critical role than the others, but the phenomenon is visible in any model.

Let us come back now at our resilience estimation. Considering the Arpanet model previously shown, the resilience gets pretty close to its maximal value as r gets close to 1. Some examples are given in Table 2.

Table 2 Some values of resilience as r increases. Same accuracy as in Table 1.

r	0.91	0.95	0.99	0.995	0.999	0.9999
Res	199.8	207.2	209.9	209.977	209.9991	209.999989

For the Arpanet model used, since we have $n = 21$ nodes, the maximal possible resilience is $\binom{n}{2} = 210$. We observe that the resilience gets very close to that value, which is expected (the rare event effect). This leads to consider other ideas for differentiating situations. For instance, a way of measuring the way a network reacts face to failures, is seeing what happens when there is no more total connectivity in the model. This means looking at the mean number of communicating pairs when the graph is unconnected. If we denote by *NCC* the number of connected components of our random graph G, observe first that the best situation is when there is only one component, a connected graph, and the worst is when no link is working and we have as many connected components as there are nodes, that is, $NCC = n$. So, we have $NCC \in \{1, 2, \ldots, n\}$, and on the borders, $\mathbb{P}(NCC = 1) = \mathbb{P}(NCP = \binom{n}{2} = R_{\text{all}}$ and $\mathbb{P}(NCC = n) = \mathbb{P}(NCP = 0) = \prod_{i \in \mathcal{E}}(1 - r_i)$.

Now, a metric capturing what happens when the graph is not connected is the conditional resilience

$$\mathbb{E}(NCP \,|\, NCC \geq 2).$$

Let us illustrate this with some numerical examples. In Table 3 we show the mean number of connected components $\mathbb{E}(NCC)$, and the two resiliences just discussed.

Table 3 Three metrics on the Arpanet, i.i.c. case (homogeneous links). All the shown digits are correct (confidence level: 95%).

r	0.91	0.95	0.99	0.995	0.999
$\mathbb{E}(NCC)$	1.251	1.073	1.0027	1.00066	1.000026
$\mathbb{E}(NCP)$	199.8	207.2	209.9	209.977	209.9991
$\mathbb{E}(NCP \mid NCC \geq 2)$	160.3	167.6	174.5	175.2	175.5

The values of the conditional resilience are more separated than those of the unconditional one. But our tools allow us to make other trials. For example, take as central variable the number of pairs of nodes that can communicate using at least two edge-disjoint paths. This is related to bi-connectivity in graphs, and we will not developed the point here, but evaluating this variable needs to decompose the graph G into its bi-connected components (also a linear computation, using a DFS or a BFS, see again [23], for example). Denoting our variable as $NCP2$, we can look at the metric $\mathbb{E}(NCP2 \mid NCC \geq 2)$. We will not pursue these illustrations here. The goal is to show that evaluating these types of metrics is pretty straightforward, and only needs to be able to compute the corresponding graph analysis.

In the Appendix we provide analytical expressions to all the quantities described here in the case of a bridge, that can be useful to check algorithms and to observe behaviors.

5 Conclusions

In this chapter we first discuss the interest of moving to performability-like metrics in network reliability. This allows to distinguish several performance levels when different components fail, capturing the capacity of the system to continue to work but possibly in degraded modes. We center the chapter around one of these metrics, called resilience, that has not received yet much attention in the literature. In telecommunications, the word is being used with the same spirit but in other contexts and associated with it, there are other technologically-oriented definitions.

Then, we considered exploring these metrics with Monte Carlo. We proposed to see the standard or crude Monte Carlo procedure differently, leading to a way of implementing it with much less computational effort when components, and then systems, are highly reliable. This is just a different look at a conditional Monte Carlo approach, but it allows testing in a straightforward manner estimators for these performability metrics. Moreover, and this is how this viewpoint on standard Monte Carlo developed, we can evaluate also the sensitivities of all these metrics with

respect to the (elementary) reliabilities of the systems' components, and with a very small overhead with respect to the cost of the metric evaluation itself.

In future work, we will explore appropriate ways of conditioning, depending on the metric considered, and the properties of the types of metrics we discuss here, as well as their possible uses in the area.

References

1. M.O. Ball, C Colbourn, and J.S Provan. Network reliability. In M. Ball, T. Magnanti, C. Monma, and G. Nemhauser, editors, *Handbook of Operations Research and Management Science: Network Models*, pages 673–762. Elsevier, Amsterdam, 1995.
2. G Rubino. Network reliability evaluation. In K. Bagchi and J. Walrand, editors, *State-of-the art in performance modeling and simulation*, pages 275–302. Gordon and Breach Books, 1998.
3. H. Cancela and L. Petingi. 2004. Reliability of communication networks with delay constraints: computational complexity and complete topologies. *Int. J. of Math. and Mathematical Sci.*, 29–32, pp. 1551–1562.
4. M. Grötschel, C.L. Monma and M. Stoer. 1995. Design of survivable networks. Chapter 10 in Vol. 7 of *Handbooks in Operations Research and Management Science*, pp. 617–672. Elsevier.
5. N. D. Singpurwalla. 2002. Dependence in network reliability. In *Proceedings of the 5th International Conference on Information Fusion (FUSION 2002*.
6. O. Matus, J. Barrera, E. Moreno and G. Rubino. 2019. On the Marshall-Olkin Copula Model for Network Reliability Under Dependent Failures. *IEEE Trans. Reliab.*, 68(2), pp. 451–461.
7. Valiant, L. 1979. The complexity of enumeration and reliability problems. *SIAM J. Comput.* 8:410–421.
8. J. S. Provan. The complexity of reliability computations in planar and acyclic graphs. *SIAM Journal of Computing*, 15(3), Aug. 1986.
9. S. Sebastio, K.S. Trivedi, D. Wang and X. Yin. Fast computation of bounds for two-terminal network reliability. *EJOR*, 238:810–823, 2014.
10. Locks, M., and A. Satyarayana. 1986. Network reliability – the state of the art. *IEEE Trans. Reliab.* R-35 (3).
11. P. L'Ecuyer, G. Rubino, S. Saggadi, and B. Tuffin. 2011. Approximate Zero-Variance Importance Sampling for Static Network Reliability Estimation. *IEEE Trans. Reliab.* R-8 (4), pp. 590–604.
12. Z. Botev, P. L'Ecuyer, G. Rubino, R. Simard, and B. Tuffin. 2013. Static Network Reliability Estimation via Generalized Splitting. *INFORMS Journal on Computing* 25, 1, 56–71.
13. H. Cancela, P. L'Ecuyer, G. Rubino, and B. Tuffin. 2010. Combination of Conditional Monte Carlo and Approximate Zero-Variance Importance Sampling for Network Reliability Estimation. Proceedings of the 2010 Winter Simulation Conference, 1263–1274.
14. Rare event simulation. Edited by G. Rubino and B. Tuffin. Wiley 2009.
15. C. Colbourn. The Combinatorics of Network Reliability. Oxford University Press, Inc., 1987.
16. M. El Khadiri and G. Rubino. Accelerating the standard Monte Carlo evaluation of highly reliable binary systems. In *Second International Conference on Monte Carlo and Quasi-Monte Carlo Methods in Scientific Computing*. University of Salzburg, Austria, July 1996.
17. G. Rubino. 2005. Sensitivity analysis of network reliability using Monte Carlo. In *Proceedings of the Winter Simulation Conference*.
18. J. Meyer. On evaluating the performability of degradable computing systems. IEEE Transactions on computers 29 (8), pages 720–731, 1980.
19. R.M. Van Slyke and H. Franck. Network reliability analysis I. *Networks*, 1:279–290, 1972.
20. Charles J. Colbourn. Network resilience. *Siam J. Alg. Disc. Meth.*, 8(3):404–409, July 1987.

21. A.T. Amin, K.T. Siegrist and P.J. Slater. The expected number of pairs of connected nodes: Pair-connected reliability. *Mathl. Comput. Modelling*, 17(11), 1993.
22. H. Cancela, G. Rubino and B. Tuffin. 2005. New measures of robustness in rare event simulation. In *Proceedings of the Winter Simulation Conference*.
23. T.H. Cormen, C.E. Leiserson, R.L. Rivest and C. Stein. Introduction to Algorithms, 3rd Edition. MIT Press, 2009.

Appendix

For checking purposes, we put here analytical expressions of all the objects considered in Section 4 of the chapter, in the case of the bridge model. They are obtained basically by brute force, given the small size of the model.

- On the number of connected components NCC, $NCC \in \{1, 2, 3, 4\}$:
  - $\mathbb{P}(NCC = 1) = R_{\text{all}} = r^3(8 - 11r + 4r^2)$
  - $\mathbb{P}(NCC = 2) = 2r^2(1 - r)^2(5 - 4r) = 10r^2 - 28r^3 + 26r^4 - 8r^5$
  - $\mathbb{P}(NCC = 3) = 5r(1 - r)^4$
  - $\mathbb{P}(NCC = 4) = (1 - r)^5$
  - $\mathbb{E}(NCC) = 4 - 5r + 2r^3 + r^4 - r^5$
- On the number of communicating pairs NCP, $NCP \in \{0, 1, 2, 3, 6\}$:
  - $\mathbb{P}(NCP = 0) = (1 - r)^5$
  - $\mathbb{P}(NCP = 1) = 5r(1 - r)^4$
  - $\mathbb{P}(NCP = 2) = 2r^2(1 - r)^3$
  - $\mathbb{P}(NCP = 3) = 2r^2(1 - r)^2(4 - 3r)$
  - $\mathbb{P}(NCP = 6) = R_{\text{all}} = r^3(8 - 11r + 4r^2)$
  - $\mathbb{E}(NCP) = 5r + 8r^2 - 14r^4 + 7r^5$
  - $\mathbb{E}(NCP \,|\, NCC \geq 2) = \dfrac{r(5 + 18r - 17r^2)}{1 + 2r + 3r^2 - 4r^3}$
- On the number of pairs communicating through at least 2 edge-disjoint paths:
  - $NCP2 \in \{0, 3, 6\}$
  - $\mathbb{P}(NCP2 = 0) = (1 - r)(1 + r - r^2 + 3r^3 - 2r^4)$
  - $\mathbb{P}(NCP2 = 3) = 2r^2(1 - r)^2$
  - $\mathbb{P}(NCP2 = 6) = r^4(3 - 2r)$
  - $\mathbb{E}(NCP2) = 6r^2(1 - 2r + 4r^2 - 2r^3)$
  - $\mathbb{E}(NCP2 \,|\, NCC \geq 2) = \dfrac{6r^2}{1 + 2r + 3r^2 - 4r^3}$
- Sensitivities of the resilience:
  - W.r.t. links $\{1, 2\}$, or $\{1, 3\}$, or $\{2, 4\}$, or $\{3, 4\}$: $1 + 3r + r^2 - 12r^3 + 7r^4$;
  - w.r.t. link $\{2, 3\}$: $1 + 4r - 4r^2 - 8r^3 + 7r^4$.