A Note on the Complexity of Network Reliability Problems

Hans L. Bodlaender

Thomas Wolle

institute of information and computing sciences, utrecht university

technical report UU-CS-2004-001

www.cs.uu.nl

A Note on the Complexity of Network Reliability Problems

Hans L. Bodlaender and Thomas Wolle

Institute of Information and Computing Sciences, Utrecht University
P.O.Box 80.089, 3508 TB Utrecht, The Netherlands
`hansb@cs.uu.nl` `thomasw@cs.uu.nl`

Abstract. Let be given an undirected, simple graph $G = (V, E)$. We associate to each vertex a number in $[0, 1]$ - its reliability, i.e. the probability that it does not fail. Furthermore, let a set $S \subseteq V$ of servers and a set $L \subseteq V$ of clients be given. Vertex failures are independent of each other. The network reliability asks for the probability that the graph induced by the non-failed vertices is connected. We can also ask for the probabilities of connections between vertices of S and between vertices of S and L, e.g. 'What is the probability that all clients are connected to at least one server?' In this note, we consider a list of such problems and prove their $\#P$-completeness.

1 Introduction

Networks are an essential part of current information and communication technology. The reliability of these networks is of great importance. In this note, we look at the problem to determine such reliabilities for networks, modelled in the following way. We model the real world networks as undirected graphs; sites correspond to vertices and links to edges. A number representing the non-failure probability (i.e. the reliability) is associated to each element of the network. The *surviving subgraph* is a subgraph of the graph, where the reliability of an element gives the probability that it is present in the surviving subgraph. We furthermore assume that failures of network elements do not influence each other, i.e., the probabilities that vertices fail are independent. The classical network reliability problems ask for the probability that the surviving subgraph is still connected or for the probability that there is a connection between two distinguished vertices.

Twenty years ago, Provan and Ball [7] and Valiant [9] showed that the classical network reliability problems are $\#P$-hard. A similar result was shown independently by Jerrum [4]. The complexity class $\#P$ was first defined by Valiant [9]. Carlier and Lucet give a decomposition method to compute the reliability of a network [2]. This decomposition method was introduced by Rosenthal [8]. Because of the computational hardness of the problems on general graphs, Arnborg and Proskurowski [1], Mata-Montero [6], Lucet, Manouvrier and Carlier [5] and Wolle [10] give efficient algorithms for graphs of bounded treewidth (pathwidth).

In our model we are given a graph and the reliabilities of its vertices. We assume edges are perfectly reliable. Furthermore, we have a set of servers and a set of clients as distinguished vertices. We extend the classical network reliability problems to questions such as: 'What is the probability that all clients are connected to at least one server?' or 'What is the expected number of components of the graph induced by the non-failed elements that contain a vertex of S?' We prove these and similar questions to be $\#P'$-complete on general graphs. $\#P'$ is an extension of $\#P$ to include rational valued functions.

$\#P'$ was called $\#P$ in [7]. We use the different notation $\#P'$ to avoid confusion with the original class $\#P$. The $\#P'$-completeness proofs in this paper consist, as usual, of two parts: proofs of membership in $\#P'$, and transformations from problems known to be $\#P'$-hard. Unlike as is common for NP-completeness proofs, the membership part is not entirely trivial, although it is not very complex.

The paper is organised as follows. In Section 2, we give preliminary definitions and results: descriptions of the classical network reliability problems, the formal definition of $\#P'$ and some elementary consequences of the definition, a discussion how edge failures can be modelled by vertex failures, and our graph theoretic model. At the end of Section 2, we give the list of network reliability problems that are considered in this note. Section 3 contains proofs of the $\#P'$-membership of two network reliability problems. The membership in $\#P'$ of the other network reliability problems mentioned in Section 2 can be proven in the same way. In Section 4, we prove each of the considered network reliability problems to be $\#P'$-hard. In Section 5, we consider variants of most of the problems which amount to computing the numerator of the probability one is interested in; the denominator is easily computable. These 'numerator variants' are shown to be $\#P$-complete. Section 6 gives some concluding remarks.

2 Preliminaries

2.1 The Classic Network Reliability Problems

The two problems our transformations are based on, are the classical network reliability problems. Let an undirected, simple graph G be given with a number $p_e \in [0, 1]$ associated to each edge e of G. This number p_e is the probability that the edge e is present in the surviving subgraph. We can now consider the following problems:

- $[2 \leftrightarrow t]_e$ (Two Terminal Network Reliability Problem): What is the probability that there is a path between two distinguished vertices s and t?
- $[A \leftrightarrow t]_e$ (All Terminal Network Reliability Problem): What is the probability that all vertices remain connected?

We will use the abbreviations with brackets as the name of the problem when we refer to network reliability problems in formulas or figures.

The Two Terminal Network Reliability Problem was proven to be $\#P$-hard by Valiant [9]. More precisely, Valiant shows that computing the number of subgraphs or induced subgraphs of a given graph G such that there is a path between given vertices s and t is $\#P$-complete. This implies the $\#P'$-completeness of the Two Terminal Network Reliability Problem for the case that all edges, or all vertices (except possibly s and t), respectively, have the same reliability. Provan and Ball [7] show $\#P$-hardness for the All Terminal Network Reliability Problem. They state that the problem is $\#P$-complete (i.e. in our terms: $\#P'$-complete), but do not give a proof for the membership of the problem in

$\#P'$. Jerrum [4] showed that the probability that the surviving network is connected, viewed as a polynomial in the edge failure probabilities is complete for Valiant's class of P-definable polynomials with respect to P-projections. As the proofs of membership in $\#P'$ of the network reliability problems are not available in currently easily accessible scientific literature, we give such a proof here (Lemma 2), for one of the considered problems; the other problems can be showed to belong to $\#P'$ in the same way.

Provan and Ball [7] also showed that it is $\#P$-hard to approximate the two problems, even for the case that all edges have the same reliability.

2.2 Complexity

Valiant introduced in [9] the notions $\#P$ and $\#P$-completeness to express the hardness of problems that 'count the number of solutions'. More precisely, $\#P$ is the set of integer-valued functions that express the number of accepting computations of a nondeterministic Turing machine of polynomial time complexity. Let FP represent the class of polynomial-time computable functions. Furthermore, let $I\!N$ denote the set of nonnegative integers and let Σ be the finite alphabet of the input and output of the Turing machines considered in this note. The next definition that defines $\#P$ equivalently to Valiant's definition, can be found in [3], which also contains an extensive overview of the theory of counting complexity.

Definition 1. *The class $\#P$ consists of the functions $f : \Sigma^* \Rightarrow I\!N$ such that there exists a nondeterministic polynomial time Turing machine M such that for all inputs $x \in \Sigma^*$, $f(x)$ is the number of accepting paths of M.*

As problems like the network reliability problems are functions to the rational numbers in $[0, 1]$ and not to the integers, we cannot say, using Valiant's definition, that they belongs to $\#P$. Provan and Ball [7] extended Valiant's notion of $\#P$ and $\#P$-completeness to deal with functions with a range of multiple or rational values.

A language (function) g is *polynomially transformable* to a language (function) f, written as $g \leq f$, if there exists a Turing machine that for any input x, computes $g(x)$ with a polynomial (in the size of x) number of elementary operations and evaluations of language (function) f. (The size of x is the number of bits needed to denote x.) A function f is *$\#P$-hard*, if every function g in $\#P$ is polynomially transformable to f. f is *$\#P$-complete*, if f belongs to $\#P$ and f is $\#P$-hard.

Where NP, NP-hardness and NP-completeness deal, roughly said, with verifying and finding *one* solution to a combinatorial problem, $\#P$, $\#P$-hardness and $\#P$-completeness deal, again roughly said, with establishing the *number* of solutions to a combinational problem. As the problem of counting the number of solutions to a problem is at least as hard as determining if there is at least one solution, $\#P$-complete problems are 'as least as hard', but possibly harder than NP-complete problems.

If we would use the terminology of [7], we would indeed be able to prove that our network reliability problems are $\#P$-complete. In order not to depart from most of the current complexity theory literature, we restrict our use of the notion of $\#P$ and $\#P$-completeness to the original definition, and denote the version we use for dealing with another type of functions as $\#P'$. We define $\#P'$ as the class of functions, that can be computed in polynomial time from input x and a value $f(x)$ for some $f \in \#P$.

Definition 2. *The class $\#P'$ is defined as follows. Hardness and completeness for this class are defined as usual.*

$$\begin{aligned}\#P' = \{\ & h|h : \Sigma^* \Rightarrow \Sigma^*, \\ & \exists f \in \#P,\ f : \Sigma^* \Rightarrow I\!N, \\ & \exists g \in FP,\ g : I\!N \times \Sigma^* \Rightarrow \Sigma^*, \\ & \forall x \in \Sigma^* : h(x) = g(f(x), x)\ \}\end{aligned}$$

Lemma 1. *Let $h' : \Sigma^* \Rightarrow \Sigma^*$ be a function.*

1. $\#P \subseteq \#P'$
2. h' is $\#P$-hard $\Longrightarrow$ h' is $\#P'$-hard
3. h' is $\#P$-complete $\Longrightarrow$ h' is $\#P'$-complete

Proof. 1. This follows trivially from the definition by choosing g to be the projection on the first argument.

2. For all $f \in \#P$, we have: $f \leq h'$, since h' is $\#P$-hard. Hence, there is a deterministic Turing machine M that when given input $x \in \Sigma^*$ computes $f(x)$, using polynomial time and a polynomial number of evaluations of h'. Let $h \in \#P'$, i.e. $h(x) = g(f(x), x)$ for $f \in \#P$, $g \in FP$ and all $x \in \Sigma^*$. We describe a deterministic Turing machine M' that for input $x \in \Sigma^*$ computes $h(x)$, using polynomial time and a polynomial number of evaluations of h'. At input x, M' computes $f(x)$ by simulating machine M. After that, M' computes deterministically in polynomial time $h(x) = g(f(x), x)$, since $g \in FP$. Hence we have: $h \leq h'$, for all $h \in \#P'$.

3. This follows directly from 1. and 2. □

2.3 Edge Failures vs. Vertex Failures

In the classical network reliability problems only links between sites can fail; or translated into graph theory: only edges of a graph can be non-present. In this note, however, we consider network reliability problems where only vertices can go down (and hence its incident edges as well), but edges between two present vertices will always be present in the graph (i.e. edges are perfectly reliable).

This model with only vertex failures is not a restriction, since edge failures can be simulated by vertex failures in the following way. Place on each edge $\{u, w\}$ of the graph a new vertex v, see Figure 1. The reliability $p(v)$ of v is equal to the reliability p_{uw} of the edge $\{u, w\}$: $p(v) = p_{uw}$. All edges of the resulting graph will be defined to be perfectly reliable. The size of the new graph is polynomially bounded in the size of the original graph. Let $G = (V, E)$ be the original graph and $G' = (V', E')$ be the graph after the subdivisions. Then we have: $|V'| = |V| + |E|$ and $|E'| = 2 \cdot |E|$.

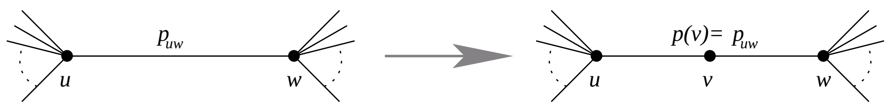

Fig. 1. Simulating edge failures by introducing new vertices.

2.4 Our Graph-Theoretical Model

For all our problems, we are given an undirected, simple graph $G = (V, E)$ with each vertex $v \in V$ a rational number $p(v) \in [0, 1]$ associated to it:

$$p(v) = \frac{a_v}{b_v}, \quad a_v, b_v \in I\!N, \quad a_v \leq b_v, \quad b_v \neq 0, \quad \text{and}$$

$$a_v, b_v \text{ have no common devisor greater than } 1$$

For each $v \in V$, the number $p(v)$ is the *reliability* of v, and models the probability that the network element represented by v is properly functioning. We are interested in the subgraph of G obtained as follows: each $v \in V$ is in the subgraph with probability $p(v)$, and each edge is in the subgraph if and only if both its endpoints are in the subgraph. Such a model is often called a *stochastic graph*. The subgraph resulting from this experiment is called the *surviving subgraph*. The vertices in the surviving subgraph are said to be *up*, while the others not in the surviving subgraph are said to be *down*.

Furthermore, we are given a set $S \subseteq V$ of servers and a set $L \subseteq V$ of clients, $n_S = |S|$ and $n_L = |L|$. The problems we consider ask for the probabilities that there are certain connections between vertices in S, and also that there are certain connections between vertices of S and vertices of L. These are listed in the next Section 2.5.

2.5 Network Reliability Problems

This section lists all network reliability problems that are proven to be $\#P'$-complete in this note. In the same way, as for the classic network reliability problems, we will use the abbreviations to refer to a problem.

- $[2 \leftrightarrow t]$: What is the probability that there is a path in the surviving subgraph between two distinguished vertices u and v?
- $[S \leftrightarrow t]$: What is the probability that all servers (S) are connected in the surviving subgraph?
- $[L \leftrightarrow \geq 1S]$: What is the probability that each client (L) is connected to at least one server (S) in the surviving subgraph?
- $[\geq xS]$: What is the probability that at least x servers are connected to each other in the surviving subgraph?
- $[Ec \geq 1S]$: What is the expected number of components of the surviving subgraph with at least one server?

- $[\geq x_1 L \not\leftrightarrow \geq 1S]$: What is the probability that at least x_1 clients are not connected to a server in the surviving subgraph?
- $[\leq x_2 L \leftrightarrow \geq 1S]$: What is the probability that at most x_2 clients are connected to at least one server in the surviving subgraph?
- $[< x_3 L \not\leftrightarrow \geq 1S]$: What is the probability that less than x_3 clients are not connected to a server in the surviving subgraph?
- $[> x_4 L \leftrightarrow \geq 1S]$: What is the probability that more than x_4 clients are connected to at least one server in the surviving subgraph?
- $[1 - [< x_3 L \not\leftrightarrow \geq 1S]]$: What is the probability that it is not the case that less than x_3 clients are not connected to a server in the surviving subgraph?
- $[\leq yS \not\leftrightarrow L]$: What is the probability that at most y servers are not connected to a client in the surviving subgraph?
- $[\geq xL \leftrightarrow S \wedge \geq yS \leftrightarrow L]$: What is the probability that in the surviving subgraph at least x clients are connected to at least one server, while at least y servers are connected to at least one client?
- $[\geq 1L \leftrightarrow \geq xS]$: What is the probability that in the surviving subgraph at least one client is connected to at least x servers?

3 $\#P'$-membership

As already mentioned, for showing the completeness of a problem for a complexity class, we have to show the hardness of the problem and the membership to this complexity class. Using Valiant's original definition of $\#P$ [9], we cannot prove the membership of our network reliability problems. Instead, we prove that these problems belong to $\#P'$. However, we will only have a close look at two problems ($[2 \leftrightarrow t]$ and $[Ec \geq 1S]$), since the description of the machines is very similar for the other problems.

Often, scientists describing nondeterministic Turing machines restrict themselves to machines that can branch in only 2 (or any other constant) computation paths at each step. However, to ease the description of our machines, we will use machines that branch in b_v computation paths at step for vertex v. In our case that is allowed, because such a branch in b_v paths can be simulated by $\lceil \log b_v \rceil$ binary branches, see Figure 2. Note that we

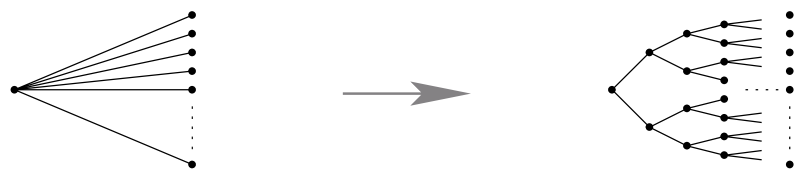

Fig. 2. Simulating one b_v-branch by $\lceil \log b_v \rceil$ binary branches.

need $\lceil \log b_v \rceil$ bits to represent b_v in the input x. Hence, we see easily that the first n steps of our machines can be simulated by $\sum_{v \in V} \lceil \log b_v \rceil$ steps of a machine with only binary branches. This is linear in the input size. If b_v is not a power of 2, in some branches, we stop branching at an earlier time. Thus, the altogether length of a computation path after the 'blowup' due to a restriction to binary branches is still bounded by a polynomial.

Lemma 2. $[2 \leftrightarrow t] \in \#P'$

Proof. To prove the membership in $\#P'$, we specify a function $h \in \#P'$ by describing functions $f \in \#P$ and $g \in FP$ with $h(x) = g(f(x), x)$ for all $x \in \Sigma^*$. Let $\#M(x)$ be the number of accepting computation paths of M with input x.

To describe f, we give a $\#P$ Turing machine M with $f(x) = \#M(x)$. M is constructed as follows. During the first $n = |V|$ steps, M considers each vertex of G, one at a time. In the step for vertex v, M branches b_v times, where in a_v branches the vertex v is present in the graph G and in $b_v - a_v$ branches, vertex v is not present in G. After considering each vertex, M decides deterministically on each computation path, whether the corresponding property is fulfilled. In our case, that is checking, whether u and w are connected in the surviving subgraph corresponding to the considered computation path of M. If this is the case, M accepts on this path, and rejects otherwise.

The probability that a vertex v is present in the graph is encoded in the total number of branches b_v at the step for vertex v, of which in exactly a_v branches vertex v is present in the graph. After fixing the state of each vertex, i.e. after n steps, each path represents exactly one surviving subgraph. Note, however, that each surviving subgraph G' might be represented by more than one such path, depending on the occurrence probability of G'. Hence, it is easy to see, that the probability $h(x)$ that u and w are connected is:

$$h(x) = \frac{\#M(x)}{\text{total number of paths of } M(x)}$$

The number of accepting paths $\#M(x)$ is given as $f(x)$, but the total number of paths of M is not given. However, this number can be computed in polynomial time by function g as the product over all vertices v of b_v:

$$h(x) = g(f(x), x) = \frac{f(x)}{\prod_{v \in V} b_v}$$

□

The method used above works also for all network reliability problems considered in this note that have the form: 'What is the probability that the surviving subgraph has property X?' However, problem $[Ec \geq 1S]$ has a different form: 'What is the expected number of components of the surviving subgraph that have property Y?' and hence, we use a slightly different Turing machine and different functions to prove its $\#P'$-membership.

Lemma 3. $[Ec \geq 1S] \in \#P'$

Proof. Similar to the proof of Lemma 2, we give functions f (i.e. a Turing machine M, with $f(x) = \#M(x)$) and g.

M is constructed as follows. The first n steps are the same as of the machine described in the proof of Lemma 2. However, after these steps, (i.e. after the surviving subgraph on each computation path so far, is completely defined), machine M works differently. Note that for the expected number $h(x)$ of components of the surviving subgraph with at least one server we have:

$$1 \leq h(x) \leq n = |V|$$

Furthermore, note that for each fixed surviving subgraph G', the number $c(G')$ of components with at least one server is a number in $\mathbb{N}$. After the first n steps M branches on each computation path into exactly $c(G')$ accepting paths, where G' is the surviving subgraph defined by the corresponding computation path. If $c(G') = 0$ then M does not branch any further but rejects on this path.

We use the same function g as in the proof of Lemma 2.

$$h(x) = g(f(x), x) = \frac{f(x)}{\prod_{v \in V} b_v}$$

We can see in the following way that this is indeed correct. When we consider the computation tree of M as a probability distribution, we see that after the first n steps every path has the same probability. The different probabilities of the surviving subgraphs G' is expressed by an appropriate number of paths that define G'. This automatically assigns the correct weights to $c(G')$ for all G', when computing the expected number of components with at least one server. □

Due to the last two lemmas and the similarity of the problems in the list in Section 2.5, the correctness of the following corollary is easy to see.

Corollary 1. *The following network reliability problems are in* $\#P'$*:* $[2 \leftrightarrow t]$, $[S \leftrightarrow t]$, $[L \leftrightarrow \geq 1S]$, $[\geq xS]$, $[Ec \geq 1S]$, $[\geq x_1 L \not\leftrightarrow \geq 1S]$, $[\leq x_2 L \leftrightarrow \geq 1S]$, $[< x_3 L \not\leftrightarrow \geq 1S]$, $[> x_4 L \leftrightarrow \geq 1S]$, $[1 - [< x_3 L \not\leftrightarrow \geq 1S]]$, $[\leq yS \not\leftrightarrow L]$, $[\geq xL \leftrightarrow S \wedge \geq yS \leftrightarrow L]$ *and* $[\geq 1L \leftrightarrow \geq xS]$.

When we want to show membership in $\#P'$ for network reliability problems where *edges* can fail, the proof of Lemma 2 is changed as follows: The first $n = |V|$ branching steps which make ‘decisions’ for every vertex, would be replaced by $m = |E|$ branching steps making the decisions whether this edge is ‘up’ or ‘down’ for every edge on a certain computation path.

4 $\#P$-hardness

In this section, we prove the $\#P$-hardness of the problems given in the list of network reliability problems in Section 2.5. For each problem we give a simple transformation from a known $\#P$-hard problem, thus proving the considered problem to be $\#P$-hard. The $\#P'$-hardness of the problems then follows directly from Lemma 1.

Lemma 4. *The following network reliability problems are #P-hard:* $[2 \leftrightarrow t]$, $[S \leftrightarrow t]$, $[L \leftrightarrow \geq 1S]$, $[\geq xS]$, $[Ec \geq 1S]$, $[\geq x_1 L \not\leftrightarrow \geq 1S]$, $[\leq x_2 L \leftrightarrow \geq 1S]$, $[< x_3 L \not\leftrightarrow \geq 1S]$, $[> x_4 L \leftrightarrow \geq 1S]$, $[1 - [< x_3 L \not\leftrightarrow \geq 1S]]$, $[\leq yS \not\leftrightarrow L]$, $[\geq xL \leftrightarrow S \wedge \geq yS \leftrightarrow L]$ *and* $[\geq 1L \leftrightarrow \geq xS]$.

Proof. For each of the problems, we give a transformation from a problem that is known to be $\#P$-hard.

$[2 \leftrightarrow t]_e \leq [2 \leftrightarrow t]$*:* Given an instance of $[2 \leftrightarrow t]_e$, we construct an instance of $[2 \leftrightarrow t]$ by simulating edges as described in Section 2.3. All vertices v that do not simulate an edge are perfectly reliable, i.e. they have $p(v) = 1$. It is easy to see that this is a correct transformation. The same transformation can be used to show that $[2 \leftrightarrow t]$ is $\#P$-hard when we additionally require that both distinguished vertices are perfectly reliable.

$[2 \leftrightarrow t] \leq [S \leftrightarrow t]$*:* Suppose we are given an instance of $[2 \leftrightarrow t]$ with two distinguished vertices u and v. We turn this into an instance of $[S \leftrightarrow t]$ by simply letting u and v be the only servers. Again, as the transformation does not change the value of the requested probability, we have a correct transformation from $[2 \leftrightarrow t]$ to $[S \leftrightarrow t]$.

$[A \leftrightarrow t]_e \leq [S \leftrightarrow t]$*:* Given an instance of $[A \leftrightarrow t]_e$, we construct a $[S \leftrightarrow t]$-instance by simulating edges as described in Section 2.3. We let the set of servers be the set of vertices that do not simulate an edge. Note that these are perfectly reliable. This transformation does not change the value of the requested probability.

$[S \leftrightarrow t] \leq [L \leftrightarrow \geq 1S]$*:* Let an instance of $[S \leftrightarrow t]$ be given. We assume that we do not have clients, and if we have, we delete their 'client'-flag, turning them to non-clients. Now, we choose an arbitrary server. This is again a server in the new to be formed instance. All other servers in the $[S \leftrightarrow t]$-instance become clients in the new instance. Thus, we have formed an instance of $[L \leftrightarrow \geq 1S]$. Now, one can note that all servers are connected in the $[S \leftrightarrow t]$-instance, if and only if each client is connected to at least one server in the $[L \leftrightarrow \geq 1S]$-instance. Hence, this transformation does not change the value of the requested probability.

$[2 \leftrightarrow t] \leq [\geq xS]$*:* Suppose we are given an instance of $[2 \leftrightarrow t]$. Taking the two distinguished vertices as the only two servers and setting $x = 2$ gives an instance of $[\geq xS]$ with the same probability value.

$[2 \leftrightarrow t] \leq [Ec \geq 1S]$*:* More precisely, we give a transformation from the version of $[2 \leftrightarrow t]$ where we assume that both distinguished vertices are perfectly reliable. We have argued above that this version is also $\#P$-hard, so this is not a restriction. Let a $[2 \leftrightarrow t]$-instance G be given, where the two distinguished vertices are perfectly reliable. Mark the two distinguished vertices u and v as the only servers, which gives us a $[Ec \geq 1S]$-instance G'. Let P be the probability that there is a path between u and v in the surviving subgraph $\bar{G}$ of G, and let E be the expected number of components with at least one server in the surviving subgraph $\bar{G}'$ of G'. E is the sum of the probability that the two servers are in one component in the surviving subgraph and two times the probability that the two servers are in two different components in the surviving subgraph. This equals the sum of the probability that there is a path between u and v in the surviving subgraph and two times the probability that there is no path between u and v in the surviving subgraph. Thus, we have $E = 2 - P$, and the transformation is easy to see.

$[\geq x_1 L \not\leftrightarrow \geq 1S] \equiv [\leq x_2 L \leftrightarrow \geq 1S]$*:* This is easy to see, because at least x_1 clients are not connected to a server, if and only if at most $x_2 := n_c - x_1$ clients are connected to at least one server in the surviving subgraph.

$[< x_3 L \not\leftrightarrow \geq 1S] \equiv [> x_4 L \leftrightarrow \geq 1S]$*:* This is also easy to see, since less than x_3 clients are not connected to a server, if and only if more than $x_4 := n_c - x_3$ clients are connected to at least one server in the surviving subgraph.

In the previous two transformations, the symbol '$\equiv$' does not only mean equivalence concerning the hardness of the problems, but it could also be understood as equality of the corresponding solutions, i.e. the solutions of the equivalent problems are equal, respectively, and hence, the transformations follow trivially. In the next transformation, we use '$\equiv$' in the same way, however with a complementary problem:

For a problem $[Z]$ in our list, we define the complementary problem $[1 - [Z]]$. For all instances σ of $[Z]$, we have $[Z]\sigma = 1 - [1 - [Z]]\sigma$, i.e. $[1 - [Z]]$ maps an instance to 1 minus the value $[Z]$ maps the instance to. So, when $[Z]$ gives us the probability that a certain property holds for the surviving subgraph, $[1 - [Z]]$ gives the probability that this property does *not* hold for the surviving subgraph. Concerning the complexity of the problems, clearly $[Z] \equiv [1 - [Z]]$.

$[\geq x_1 L \not\leftrightarrow \geq 1S] \equiv [1 - [< x_3 L \not\leftrightarrow \geq 1S]]$*:* This is again not difficult to see: At least x_1 clients are not connected to at least one server, if and only if it is not the case that less than $x_3 := x_1$ clients are not connected to at least one server in the surviving subgraph.

$[L \leftrightarrow \geq 1S] \leq [> x_4 L \leftrightarrow \geq 1S]$*:* This transformation is easy, since we do not have to modify the instance. We simply solve the $[> x_4 L \leftrightarrow \geq 1S]$-problem for $x_4 := n_c - 1$ and get immediately an answer for the $[L \leftrightarrow \geq 1S]$-problem.

$[< x_3 L \not\leftrightarrow \geq 1S] \leq [\leq yS \not\leftrightarrow L]$*:* We interchange the roles of servers and clients and we choose $y := x_3 - 1$.

$[L \leftrightarrow \geq 1S] \leq [\geq xL \leftrightarrow S \wedge \geq yS \leftrightarrow L]$*:* The $[\geq xL \leftrightarrow S \wedge \geq yS \leftrightarrow L]$-problem contains the $[L \leftrightarrow \geq 1S]$-problem as a special case: $x := n_c$ and $y = 0$.

$[\geq xS] \leq [\geq 1L \leftrightarrow \geq xS]$*:* Given a $[\geq xS]$-instance without any clients. We add to each server a pendant which is a private client for this server. This results in a $[\geq 1L \leftrightarrow \geq xS]$-instance. Now, it is easy to see, that this is a correct transformation. □

5 Computing the Numerators of Probabilities

In this section, we look at the problems from our list that compute the value of a probability, (i.e., all problems except $[Ec \geq 1S]$). Each of these probabilities is a rational number, which can be expressed as a fraction with an easily computable denominator and a numerator that is $\#P$-complete to compute.

As before, we assume that each vertex $v \in V$ has a reliability $p(v) = a_v/b_v$, with a_v a nonnegative integer, and b_v a positive integer. Let

$$D = \prod_{v \in V} b_v$$

be the product of all denominators of the reliabilities of the vertices. The following lemma can be easily be observed.

Lemma 5. *For each induced subgraph G' of G, the probability that the surviving subgraph equals G' is an integer multiple of $1/D$.*

Given a function $[X]$ (assumed to be one of the considered network reliability problems), we define the function $[D * [X]]$ by taking for each input ϕ, $[D * [X]]\phi = D * [X]\phi$. For instance, $[D * [S \leftrightarrow t]]$ asks for D times the probability that all servers are connected in the surviving subgraph.

Theorem 1. *The following problems are functions to the integers and $\#P$-complete: $[D * [2 \leftrightarrow t]]$, $[D * [S \leftrightarrow t]]$, $[D * [L \leftrightarrow \geq 1S]]$, $[D * [\geq xS]]$, $[D * [\geq x_1 L \not\leftrightarrow \geq 1S]]$, $[D * [\leq x_2 L \leftrightarrow \geq 1S]]$, $[D * [< x_3 L \not\leftrightarrow \geq 1S]]$, $[D * [> x_4 L \leftrightarrow \geq 1S]]$, $[D * [1 - [< x_3 L \not\leftrightarrow \geq 1S]]]$, $[D * [\leq yS \not\leftrightarrow L]]$, $[D * [\geq xL \leftrightarrow S \wedge \geq yS \leftrightarrow L]]$ and $[D * [\geq 1L \leftrightarrow \geq xS]]$.*

Proof. Consider some property Y.The probability that the surviving subgraph has this property Y is the sum over all induced subgraphs G' of G with property P of the probability that the surviving subgraph equals G'. Thus, by Lemma 5, the probability of Y is an integer multiple of $1/D$. Hence, we have that $[D * [Y]]$ is a function to the integers.

The proof of membership in $\#P$ of $[D * [2 \leftrightarrow t]]$ is similar, and even slightly easier than that of Lemma 2: note that the requested value precisely equals the number of accepting paths $\#M(x)$. For the other problems, membership in $\#P$ can be derived in the same way.

$\#P$-hardness for the problems can be shown with basically the same proof as for Lemma 4. □

So, we have for each of the considered network reliability problems apart from $[Ec \geq 1S]$ that these are functions to the rational numbers, which can be written as the quotient of a $\#P$-complete function (a $[D * \cdots]$ variant of the problem) and a function in P (the product of all values b_v). The problems of the type $[D * \cdots]$ hence can be seen as *numerator*-versions of the original problems.

6 Conclusions

In this note we have given a formal definition of an extension $\#P'$ of $\#P$ to include rational valued functions. Our main result can be stated as the following theorem, which follows directly from Corollary 1 and Lemma 4.

Theorem 2. *The following network reliability problems are* $\#P'$*-complete:* $[2 \leftrightarrow t]$, $[S \leftrightarrow t]$, $[L \leftrightarrow \geq 1S]$, $[\geq xS]$, $[Ec \geq 1S]$, $[\geq x_1 L \not\leftrightarrow \geq 1S]$, $[\leq x_2 L \leftrightarrow \geq 1S]$, $[< x_3 L \not\leftrightarrow \geq 1S]$, $[> x_4 L \leftrightarrow \geq 1S]$, $[1 - [< x_3 L \not\leftrightarrow \geq 1S]]$, $[\leq yS \not\leftrightarrow L]$, $[\geq xL \leftrightarrow S \wedge \geq yS \leftrightarrow L]$ *and* $[\geq 1L \leftrightarrow \geq xS]$.

The hardness proofs were obtained through simple transformations from known $\#P$-hard problems. Figure 3 gives an overview which transformations were used. A path from left to right corresponds to a chain of transformations and vertical lines represent transformations showing the hardness equivalence of the corresponding problems.

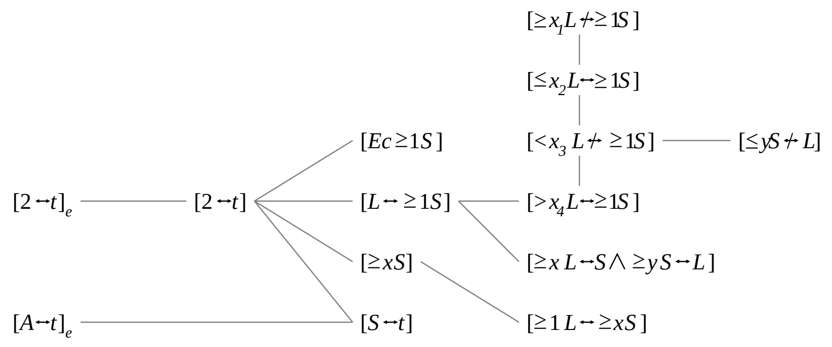

Fig. 3. The transformations in this note.

We showed the $\#P$-hardness of some network reliability problems on general graphs. This justifies to look at graphs with a special structure to enable efficient algorithms. The list given in Section 2.5 is obviously not complete. However, it is not hard to show the same type of results for other network reliability problems of a similar 'flavour'.

Acknowledgements

We thank Mark Jerrum, Jan van Leeuwen, Gerard Tel, and Peter Verbaan for very useful discussions and comments.

References

1. S. Arnborg, A. Proskurowski: *Linear time algorithms for NP-hard problems restricted to partial k-trees.* Discrete Appl. Math. 23, (1989), 11-24.
2. J. Carlier, C. Lucet: *A decomposition algorithm for network reliability evaluation.* Discrete Appl. Math. 65, (1996), 141-156.

3. L. Fortnow: *Counting complexity.* In L. A. Hemaspaandra and A. Selman, editors: *Complexity Theory Retrospective II.* Springer-Verlag, Berlin, (1997), 81-107.
4. M. Jerrum: *On the complexity of evaluating multivariate polynomials.* Ph.D. Thesis, Technical report: CST-11-81, Dept. Computer Science, University Edinburgh, (1981).
5. C. Lucet, J.-F. Manouvrier, J. Carlier: *Evaluating network reliability and 2-edge-connected reliability in linear time for bounded pathwidth graphs.* Algorithmica 27, (2000), 316-336.
6. E. Mata-Montero: *Reliability of Partial k-Tree Networks.* Ph.D. Thesis, Technical report: CIS-TR-90-14, University of Oregon, (1990).
7. J. S. Provan, M. O. Ball: *The complexity of counting cuts and of computing the probability that a graph is connected.* SIAM J. Comput. 12/4, (1983), 777-788.
8. A. Rosenthal: *Computing the reliability of complex networks.* SIAM J. Appl. Math. 32, (1977), 384-393.
9. L. G. Valiant: *The complexity of enumeration and reliability problems.* SIAM J. Comput. 8, (1979), 410-421.
10. T. Wolle: *A framework for network reliability problems on graphs of bounded treewidth.* In P. Bose and P. Morin, editors: *Algorithms and Computation (ISAAC 2002).* LNCS 2518, Springer-Verlag, Berlin, (2002), 137-149.