数理解析研究所講究録
1120巻 1999年 120-129

グラフのTutte多項式計算システム

A System for Computing the Tutte Polynomial of a Graph

今井 浩　　関根 京子
Hiroshi IMAI　Kyoko SEKINE

東京大学大学院理学系研究科
〒113-0033 東京都文京区本郷7-3-1
imai@is.s.u-tokyo.ac.jp

Abstract: The invariant polynomials of discrete systems such as graphs, matroids, hyperplane arrangements, and simplicial complexes, have been theoretically investigated actively in recent years. These invariants include the Tutte polynomial of a graph and a matroid, the chromatic polynomial of a graph, the network reliability of a network, the Jones polynomial of a link, the percolation function of a grid, etc. The computational complexity issues of computing these invariants have been studied and most of them are shown to be #P-complete. But, these complexity results do not imply that we cannot solve a given instance of moderate size. To meet large demand of computing these invariants in practice, there have been proposed a framework of computing the invariants by using the binary decision diagrams (BDD for short). This provides mildly exponential algorithms which are useful to solve moderate-size practical problems. This paper surveys the BDD-based approach to computing the invariants, and describe computational resutls of the system which has been developed for practical use.

Keywords: Tutte polynomial, graph, matroid, BDD, #P-complete, mildly exponential algorithm

1 Introduction

This paper concerns computing the invariant polynomial of discrete systems, specifically the Tutte polynomials of graphs and matroids and their variants. The theory of these invariant polynomials was originated around the beginning of this century, and it has been extended to various fields connected with discrete systems [3, 26]. Computational aspects of these invariant polynomials have been a hot topic in these ten years, because its computation is very useful in a variety of fields [26]. This computation problem is #P-complete in general. Recently, the binary decision diagram, BDD, has been used to solve this combinatorial problem efficiently [17]. This paper first describes the theory of these invariant polynomials briefly, and surveys the computational approach in detail.

The Tutte polynomial of a graph is one of fundamental invariants in graph theory, which was proposed by Tutte [23]. As for invariant polynomials of a graph, the chromatic polynomial, which denotes the number of vertex colorings such that no two adjacent vertices have the same color, seems more popular. This might be because the well-known 4-color theorem of a planar graph. In fact, the chromatic polynomial was originally considered to tackle this problem around 1912 (see [25, 26]).

The Tutte polynomial can be naturally defined for matroids. The Tutte polynomial $T(M;x,y)$ of a matroid M is a two-variable polynomial of x and y. This polynomial has many combinatorial meanings. For example, the following invariant polynomial of discrete systems are special cases of the Tutte polynomial.

- the chromatic polynomial and flow polynomial of a graph

- the network reliability of a network
- the partition function of an Ising model and a Q-state Potts model
- the Jones polynomial of an alternating link
- the weight enumerator of a linear code over $GF(q)$
- the shelling polynomial and the characteristic polynomial of a matroid complex

Also, values of the Tutte polynomial $T(M;x,y)$ of a matroid M with two variables x and y at some typical points (x,y) have the following meanings.

- $T(M;1,1)$ is the number of bases of M (spanning trees in the case of a graph)
- $T(M;2,1)$ counts the number of independent sets of M (forests in the case of a graph)
- $T(M;1,2)$ counts the number of spanning sets
- $T(M;2,0)$ is the number of cells of a central arrangement of a linear matroid on reals and is the number of acyclic orientations of a graph

For more details, see [3, 26].

The problem of computing the Tutte polynomial, $T(G;x,y)$, of a graph G is #P-complete in general, except in some special cases such as the number $T(G;1,1)$ of spanning trees. For example, when $x=2$ and $y=1$, it gives the number of forests, and this computation becomes #P-hard. That is, in most cases, it is in a complexity class at least as intractable as NP and therefore seems unlikely to have a polynomial time solution. Recently, Alon, Frieze and Welsh [1] developed fully polynomial time randomized approximation schemes for approximating the value of the Tutte polynomial for any dense graph G, whenever $x,y \geq 1$. This result was extended to a general graph by Karger [10]. Hence this is especially useful for calculating the approximate values of the Tutte polynomials which have special meanings such as the number of forests.

On the other hand, the exact computation of the Tutte polynomial still remains a challenging problem. Although exponential time would be inevitable for the exact computation in view of the #P-completeness, reducing the exponent would enable us to solve moderate-size problems. Mildly exponential algorithms are practically important.

There has been proposed a BDD-based approach to tackle these hard problem. The binary decision diagram, BDD for short, has been used in VLSI CAD for manipulating Boolean functions in an efficient way [2]. A general package of BDD has been developed. It is powerful enough compared with other methods of handling Boolean functions, but such a general approach has apparent limitation to the Tutte polynomial computation. Sekine and Imai [17] propose a top-down construction algorithm of the BDD representing all spanning trees of a graph, and then Imai, Iwata, Sekine and Yoshida [9] the BDD of bases of a binary and ternary matroid. This approach can be generalized to solve related problems, such as computing the Jones polynomial of a link, and the number of ideals of a partially ordered set. Such a relation between the BDD and the Tutte polynomial computation has been recognized in a series of papers [6, 17, 20, 21], from which interesting insights can be obtained from both sides.

The paper proceeds as follows. The section 2 introduces the Tutte polynomial of a matroid, and mentions a fundamental result for computing the Tutte polynomial of a graph. Then the section 3 describes the BDD-based paradigm for this computation for graphs. The time and space complexities of the algorithms are analyzed for complete graphs and planar graphs. As a specific example how the computation of some special case of the Tutte polynomial is interesting, the network reliability computation is discussed in the section 4, together with computational results of the real system of comuting the Tutte polynomial.

2 Tutte Polynomial: Definitions and Naïve Algorithm

The Tutte polynomial is defined for a general matroid M, but wee will be mainly concerned with a linear matroid M on a finite set E. For matroids, see [3, 13, 25]. The most typical linear matroid is that over the reals. Given a set E of m vectors $\boldsymbol{a}_1, \boldsymbol{a}_2, \ldots, \boldsymbol{a}_m$ in $\mathbf{R}^n$, linear independence among these vectors induces a linear matroid $M(E)$ of

vectors in E. The rank function $\rho: 2^E \to \mathbf{Z}$ of $M(E)$ is defined by

$$\rho(S) = \dim(\{\boldsymbol{a}_i \mid \boldsymbol{a}_i \in S\}) \quad (S \subseteq E).$$

where the righthand is the dimension of a space spanned by $\boldsymbol{a}_i$ ($\boldsymbol{a}_i \in S$). The linear matroid $M(E)$ of vectors $\boldsymbol{a}_i \in E$ can be regarded as that of the arrangement of hyperplanes $h_i = \{\boldsymbol{x} \mid \boldsymbol{a}_i \cdot \boldsymbol{x} = 0\}$ ($i = 1, \ldots, m$) in the dual $\mathbf{R}^n$.

The Tutte polynomial $T(M; x, y)$ of matroid M on E is a two-variable polynomial of x and y. By the rank function ρ, it is defined by

$$T(M; x, y) = \sum_{S \subseteq E} (x-1)^{\rho(E)-\rho(A)} (y-1)^{|S|-\rho(S)}.$$

The original definition of the Tutte polynomial by Tutte is expressed as the summation over all bases of a matroid. To describe this, we need more definitions. Let B be a base of matroid M. For $e \in E - B$, a minimal dependent set of $B \cup \{e\}$, including e, is uniquely determined, which is called the fundamental circuit of e with respect to B. For $e \in B$, $\{e' \in E \mid (B - \{e\}) \cup \{e'\}$ is a base$\}$ is called the fundamental cutset of e with respect to B. Given an ordering $e_1, e_2, \ldots . e_m$ of elements of E, $e_i \in E - B$ is called externally active if its fundamental circuit with respect to B consists of e_j with $j \leq i$. $e_i \in B$ is called internally active if its fundamental cutset with respect to B consists of e_j with $j \leq i$. Then, for B, the external activity $r(B)$ is the number of external active elements, and the internal activity $s(B)$ is the number of internal active elements. Then, for this ordering, the Tutte polynomial is given by

$$T(M; x, y) = \sum_{B: \text{ bases of } M} x^{r(B)} y^{s(B)}.$$

The Tutte polynomial of matroid M has many meanings. For example, $T(M; 1, 1)$ is the number of bases of M, since it counts the number of subsets S with $|S| = \rho(S) = \rho(E)$. $T(M; 2, 1)$ the number of independent sets of M, and $T(M; 1, 2)$ the number of spanning sets of M (see [3, 26]). With the arrangement of hyperplanes such that all the hyperplanes pass the origin, a linear matroid M over the reals is associated in a straightforward way. An arrangement is central if their hyperplanes have non-empty common intersection, and our arrangement is central. In this case, $T(M; 2, 0)$ gives the number of regions of this central arrangement.

When the Tutte [23] introduced the Tutte polynomial, he also showed it has the recursive formula. This formula holds for matroids, but from here in this section we describe the case of a graph to state a specific complexity of some fundamental algorithm.

Theorem 1 *For $e \in E$, the Tutte polynomial $T(G; x, y)$ is expanded as follows:*

$$\begin{cases} xT(G/e; x, y) & e\text{: coloop} \\ yT(G\backslash e; x, y) & e\text{: loop} \\ T(G\backslash e; x, y) + T(G/e; x, y) & \text{otherwise} \end{cases}$$

Here, for an edge e in E, we denote by $G\backslash e$ the graph obtained by deleting e from G, and by G/e the graph obtained by contracting e from G. A loop is an edge connecting the same vertex, and a coloop is an edge whose removal decreases the rank of the graph by 1. If G is a connected graph a coloop is an edge of G whose removal disconnects G. By definition, the Tutte polynomial of a loop is y and that of a coloop is x. The Tutte polynomial of a graph with no edge is 1. Note that the deletion, contraction, loop, and coloop are all defined for matroids.

By applying the above formula recursively for an edge chosen by any order we can also compute the Tutte polynomial. This computation process corresponds to top-down fashion for an expansion tree (Fig. 1). The root corresponds to the graph G, and each parent has at most two children. For each path from the root to a leaf in the expansion tree, when a coloop is contracted or a loop is deleted, x or y is multiplied, respectively. Then the sum of the leaves is the Tutte polynomial of a given graph G.

Here, for each path from the root to a leaf in the expansion tree, a set of contracted edges corresponds to a spanning tree of G one-to-one. For example, the most left path in Fig. 1 corresponds to the spanning tree $\{e_1, e_2, e_3\}$. Then the number of leaves equals the number of spanning trees. The depth of the expansion tree is $|E|$, Since the depth of the expansion tree is $|E|$, by using this expansion tree in a clever way we obtain the following bound. For more details of existing approaches, see [16].

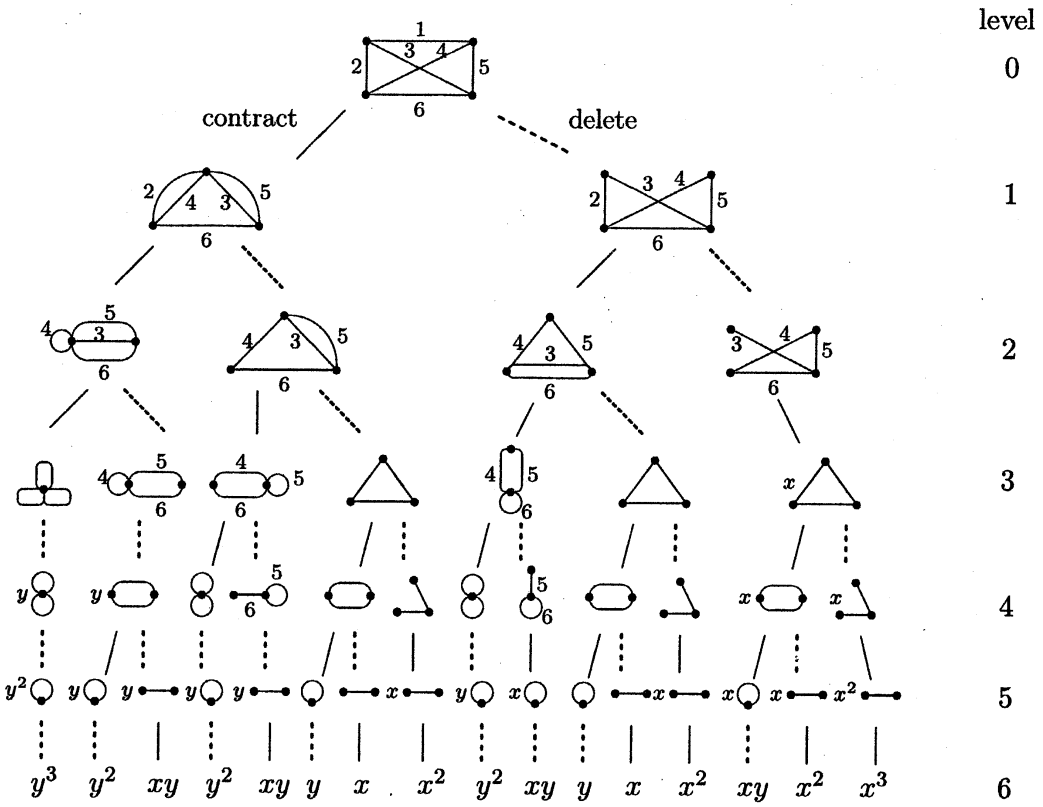

Figure 1: Expansion tree for complete graph K_4

Theorem 2 *Using the recursive formula, the Tutte polynomial of a graph $G = (V, E)$ can be computed in $O(|E|T(G;1,1))$ time.*

3 Tutte Polynomial: BDD-based Algorithms

In this section, a BDD-based algorithm for computing the Tutte polynomial of a graph, which does not take time proportional to the number of spanning trees.

For a given graph G, order the edges $e_1, e_2, \ldots, e_m$ ($m = |E|$). Suppose we apply the recursive formula in the order of $e_1, e_2, \ldots, e_m$ in a top-down fashion as in the expansion tree described in the previous section. A graph obtained from G by deletions and/or contractions of edges is called a minor of G. Nodes in the i-th level in the expansion tree correspond to minors of G with the edge set $\{e_{i+1}, e_{i+2}, \ldots, e_m\}$ (the 0-th level is the root). Since the Tutte polynomial is an invariant for isomorphic graphs, we may represent isomorphic minors among them by one of these members. However, for given two graphs, there is no efficient algorithm to decide whether they are isomorphic or not and finding all isomorphic minors may be difficult.

The isomorphism between two graphs whose edges have an identity map can be determined in linear time. For this reason, we may restrict ourselves just to finding isomorphic minors whose corresponding edges have the same order in the original graph G. By merging the isomorphic minors with the same edge ordering, the expansion tree becomes an acyclic graph (an edge is directed from a parent to a child). See an example of the complete graph K_4 in Fig.2. This acyclic graph has a single source (the original graph G) and the m-th level may be regarded as a single sink.

Rigorously, the acyclic graph representing the computation process can be constructed as the following algorithm, where S_i is the set of minors in the i-th level.

$S_0 := \{G\}$;
for $i := 1$ **to** m **do**
  begin
    $S_i := \emptyset$;
    for each minor $\tilde{G}$ in S_{i-1} **do**

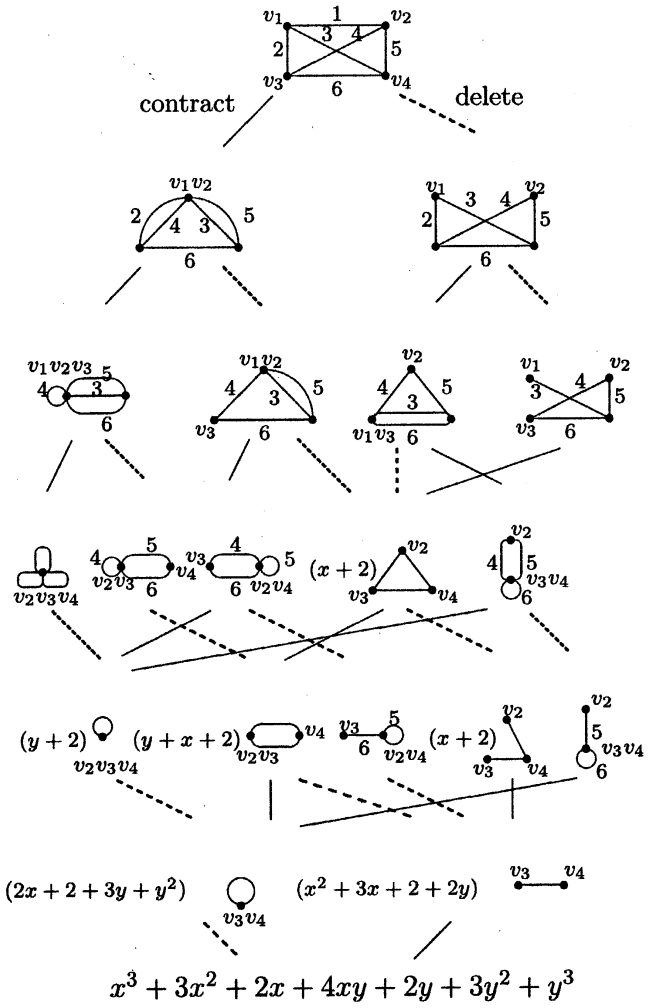

Figure 2: Computation process of $T(K_4; x, y)$

begin
  if e_i is a loop in $\widetilde{G}$ **then**
    child($\widetilde{G}$) := $\{\widetilde{G}\backslash e_i\}$
  else if e_i is a coloop in $\widetilde{G}$ **then**
    child($\widetilde{G}$) := $\{\widetilde{G}/e_i\}$
  else (comment: e_i is neither a loop nor a coloop) child($\widetilde{G}$) := $\{\widetilde{G}\backslash e_i, \widetilde{G}/e_i\}$;
  for each minor $\widetilde{G}_{e_i}$ in child($\widetilde{G}$) **do**
    begin
      check if there is an isomorphic graph with the same edge ordering in S_i;
      if there is such an isomorphic graph $\hat{G}$ in S_i then construct an edge from the node representing $\widetilde{G}$ to the node representing $\hat{G}$;
      otherwise, add $\widetilde{G}_{e_i}$ to S_i and construct an edge from the node representing $\widetilde{G}$ to the node of $\widetilde{G}_{e_i}$;
    end
  end
end;

Via the above computation process, the Tutte polynomial can be computed as follows. The next algorithm shows the Tutte polynomial can be computed by top-down fashion and need not by bottom-up fashion. Here a two-variable polynomial $t(v; x, y)$ is associated with each minor v in the computation process.

t(source; x, y) := 1;
for i := 1 **to** m **do**
  begin
    for all nodes u in S_i **do** $t(u; x, y) := 0$;
    for each node v in S_{i-1} **do**
      begin
        if v has two children u, w **then**
          begin
            $t(u; x, y) := t(u; x, y) + t(v; x, y)$;
            $t(w; x, y) := t(w; x, y) + t(v; x, y)$
          end
        else (comment: v has only one child u)
        if e_i is a loop **then**
          $t(u; x, y) := t(u; x, y) + yt(v; x, y)$
        else (comment: e_i is a coloop)
          $t(u; x, y) := t(u; x, y) + xt(v; x, y)$;
      end
  end;
t(sink; x, y) is $T(G; x, y)$.

3.1 Decision of Isomorphic Minors

The size of the computation process is defined as the number of minors which occur in computing the Tutte polynomial by the algorithm. The width is defined as the maximum among the numbers of minors of the computation process at each level. The depth of the computation process is the number of edges. Hence the width of the computation process is relevant.

Suppose that $E_i = \{e_1, e_2, \ldots, e_i\}$, and $\overline{E_i} = \{e_{i+1}, e_{i+2}, \ldots, e_m\}$. Then the minors of G in the i-th level have the edge set $\overline{E_i}$. For $i = 1, \ldots, m$, define the i-th level elimination front $\widetilde{V}_i$ to be a vertex subset consisting of vertices v such that v is incident to some edges in E_i and some edges in $\overline{E_i}$. By the edges contracted in this process, we can define an equivalence relation on $\widetilde{V}_i$ such that

two vertices are in the same equivalence class if and only if, in the process of obtaining the minor, they are unified into one vertex by the contractions. Then consider a partition of $\tilde{V}_i$ into the equivalence classes by this relation. We call this partition the i-th level elimination partition of the minor. For example, in Fig. 2 the third level elimination front is $\{v_2, v_3, v_4\}$, since all incident edges of v_1 are contracted or deleted. When e_1 and e_2 are contracted and e_3 is deleted, v_2 and v_3 are unified into one vertex. In this case, the elimination partition of this minor is $\{\{v_2, v_3\}, \{v_4\}\}$. By using these definitions, we can derive the following.

Theorem 3 *Let H_1 and H_2 be two minors of G with the same edge set $\overline{E_i}$. H_1 and H_2 are isomorphic with the same edge ordering if and only if their i-th level elimination partitions are identical.*

This theorem can be used not only to check the isomorphism more easily but also to analyze the size of the computation process. Furthermore, checking whether two partitions are identical can be done very easily.

The Tutte polynomial is an invariant for 2-isomorphic graphs which is related to isomorphism of matroids. If two graphs G_1 and G_2 are isomorphic then they are also 2-isomorphic, although we can merge only isomorphic minors with the same edge ordering by using Theorem 3.

For a given connected graph if the edge ordering has a connectedness property, all 2-isomorphic minors with the same edge ordering are isomorphic minors with the same edge ordering. Here, the edge ordering is assumed to have a connectedness property if for $i = 1, \ldots, m$, all subgraphs of G on $\overline{E_i}$ are connected.

Theorem 4 *Suppose that the edge ordering has the connectedness property for a given connected graph G. Let G_1 and G_2 be two minors of G on the same edge set $\overline{E_i}$. Then G_1 and G_2 are 2-isomorphic with the same edge ordering. if and only if G_1 and G_2 are isomorphic with the same edge ordering.*

For proofs of these theorems, see [19].

Table 1: The size of computation process of K_n

n	width	Bell number B_{n-2}	number of spanning trees
2	1	–	1
3	2	(1)	3
4	5	(2)	16
5	14	(5)	125
6	42	(15)	1296
7	130	(52)	16807
8	406	(203)	262144
9	1266	(877)	4782969
10	3926	4140	10^8
11	15106	21147	$\approx 2.36 \times 10^9$
12	65232	115975	$\approx 6.20 \times 10^{10}$
13	279982	678570	$\approx 1.79 \times 10^{12}$
14	1191236	4213597	$\approx 5.67 \times 10^{13}$

3.1.1 The Complexity of a Complete Graph

We consider the size of the computation process of a complete graph K_n of n vertices, since it is the upper bound for the other simple connected graphs.

For the complete graph K_n of n vertices, order the vertices from 1 to n. Then, represent each edge by a tuple (u, v) where u and v are numbers attached to their endpoints and $u < v$, and order edges in the increasing lexicographic order of (u, v). This ordering is called a canonical edge ordering of a complete graph.

Let $L(G, i)$ be the number of minors in the i-th level of the computation process for the canonical edge ordering. Since each parent has at most two children, $L(G, i) \leq 2^i$. More precisely, for the complete graph K_n, the following theorem holds. Here the Bell number B_n is the number of partitions of a set of n elements.

Theorem 5 *$L(K_n, i) = 2^{O(i)}$ for $i \leq 2n-3$, and $L(K_n, i) \leq B_j$ for $2n-3 < i$.*

Corollary 1 *For $n \geq 10$, the width of the computation process of K_n for the canonical edge ordering is bounded by B_{n-2}*

This bound is not so tight. Table 1 gives the width and the size of the computation process of K_n up to $n = 14$. It also shows the width can be bounded by B_{n-2} for $n \geq 10$ and much smaller than the number of spanning trees.

Theorem 6 *For any simple connected graph G with n vertices, there exists an edge ordering such that the size of the computation process of G is less than or equal to the size of the computation process of the complete graph K_n with respect to the canonical edge ordering.*

Note that, in computing the Tutte polynomial of a graph with n vertices ($n \geq 10$) via the computation process, the space complexity is also bounded by the width of the computation process and hence by B_{n-2}. This is another advantage of this algorithm.

3.1.2 The Complexity of a Planar Graph

Next, we will see that the proposed algorithm solves the problem of computing the Tutte polynomial of a planar graph, which itself is still #P-hard, very efficiently.

First, to examine its efficiency for a planar graph, we will consider its computational complexity of a lattice graph. The (square) lattice graph $L_{m,n}$ is a graph which has $m \times n$ vertices located at the points (x, y) of the 2-dimensional grid with edges joining neighbours on the grid. The lattice graph is extremely important for a number of problems in statistical physics.

For a $k \times k$ lattice graph $L_{k,k}$ with $n = k^2$ vertices, Theorem 7 shows there is an edge ordering such that the size of any elimination front (the maximum number of vertices in any elimination front) can be bounded by $k = \sqrt{n}$, that is, the algorithm works very efficiently.

For a $k \times k$ lattice graph $L_{k,k}$ order the vertices in a row-major order i.e., from the top row to the bottom row, and for each row from left to right. Then, a canonical edge ordering of a lattice graph is defined by the same way for a complete graph. In addition, the Catalan number C_{k+1} is defined to be $\frac{1}{k+1}\binom{2k}{k}$.

Theorem 7 *(i)* $L(L_{k,k}, i) \leq C_{k+1}$. *Equality holds for* $\lfloor \frac{k}{2} \rfloor (2k-1) \leq i \leq 2(k^2-k)-k$.
(ii) $L\left(L_{k,k}, 2(k^2-k)-j\right) = C_{j+2}$ *for* $0 \leq j < k$.

Again, the sizes of the computation process of $L_{k,k}$ up to $k = 12$, i.e., up to 144 vertices and 264 edges, have been computed by the algorithm proposed here (Table 2). The width is bounded by C_{k+1}, though the number of spanning trees becomes huge even for small k.

Next we will see the size of the computation process of general planar graphs. In general, the size of the computation process depends on the ordering of edges. For a planar graph, by using the planar separator theorem, we can see that an appropriate edge ordering exists and the Tutte polynomial can be computed efficiently as follows.

Let G be a planar graph with n vertices. Here, the planar separator theorem [12] is that the vertices of G can be divided into three sets A, B, C such that the following conditions hold.

- There is no edge whose one end belongs to A and the other end belongs to B.
- A and B do not include more than $\frac{2}{3}n$ vertices.
- C does not include more than $2\sqrt{2}\sqrt{n}$ vertices.

The set C is called separtor. Ordering edges by using the planar separator theorem recursively and the vertex ordering $A \prec B \prec C$, we obtain the following.

Lemma 1 *For a simple connected planar graph G of n vertices, there exists an edge ordering such that any elimination front consists of $O(\sqrt{n})$ vertices, and such an edge ordering can be found in $O(n \log n)$ time.*

Lemma 1 can be extended for graphs with good separators:

Lemma 2 *For a class of graphs having a separator of $O(n^\alpha)$ (n: the number of vertices), there exists an edge ordering such that any elimination front consists of $O(n^\alpha)$ vertices, and such an edge ordering can be found in $O(n \log n)$ time.*

Theorem 8 *The width of the computation process of an n^α-separable graph $\mathcal{G}_\alpha$ ($0 < \alpha < 1$) with n vertices is bounded by $2^{O(n^\alpha \log n)}$.*

For planar graphs, we can derive a more tight bound.

Theorem 9 *For a connected, simple planar graph with n vertices, there exists an elimination ordering of edges such that any elimination partition consists of at most $O(2^{O(\sqrt{n})})$. Such an elimination ordering can be found in $O(n \log n)$ time.*

Table 2: The size of computation process of $k \times k$ lattice graphs $L_{k,k}$

k	$\|V\|$	$\|E\|$	width ($= C_{k+1}$)	number of spanning sets
2	4	4	2	5
3	9	12		431
4	16	24	14	555195
5	25	40	42	10286937043
6	36	60	132	2692324030864335
7	49	84	429	9852929684161379901975
8	64	112	1430	501079193080617800221189943995
9	81	144	4862	352690403996687922642590703716802346343
10	100	180	16796	342629768051375876407570610261504079066783230441 5
11	121	220	58786	458508006189588425325361635000918336126387961057365005349963
12	144	264	208012	84398514547184481668012694247654470155019747743857323680567642196192347 5

Note that the BDD-based algorithm has high parallelism as reported in a preliminary report [15].

4 Network Reliability

To analyze network reliability against probabilistic failures of links and sites, simple theoretical models have been proposed. The simplest model is concerned with link failures, and considers the probability that the network remains connected when each edge e becomes open (fail, disappear) with some probability p_e independently (and hence edge e survives with probability $1-p_e$) [4]. This is called the all-terminal network reliability, and, when each p_e is a constant p, it is called the canonical all-terminal network reliability. For example, the all-terminal reliability function involve information on the number of minimum cuts, etc., of networks. In fact, the size (the number of edges) of minimum cuts as well as the number of minimum cuts is used as criteria for network reliability in papers concerned with graph connectivity, etc., and these are represented implicitly in our network reliability model.

From the theory of computational complexity, even in such simple models it is hard to obtain exact network reliability values. That is, computing the network reliability is a #P-complete problem [14, 24], and is believed hard to solve if the problem size is large. Hence, there have been proposed many approximation algorithms. Recently, randomized fully polynomial-time approximation schemes for computing the network reliability have been developed by Alon, Frieze, Welsh [1] and Karger [10]. Karger and Tai [11] report implementations of those algorithms, and show that the network of moderate size up to 50 to 60 vertices can be analyzed approximately by their methods. For the whole network reliability research, see [4, 5, 7, 8, 22].

The BDD-based approach can be applied to this problem to yield a mildly exponential time algorithm (Sekine, Imai [17, 18]). This outperform other exponential-time algorithms based on the recursive formula. With this approach, networks of moderate size can be analyzed. Furthermore, this approach yields a polynomial-time algorithm for complete graphs, whose reliability provides a natural upper bound for simple networks, and also leads to an effective method for computing the dominant part of the reliability function when the failure probability is sufficiently small.

This section reports computational results of the new approach of analyzing network reliability against probabilistic link failures. Computational results for complete graphs and the case with small failure probability are also reported.

4.1 All-Terminal Network Reliability

Let $G = (V, E)$ be a simple connected undirected graph with vertex set V and edge set E. Consider a network (graph) $G = (V, E)$. The *canonical all-terminal network reliability* $R(G; p)$ is defined as the probability that G remains connected after each edge is deleted with the same probability p. It is known that $R(G; p)$ can be computed by enumerating the spanning sets of G by using the concept of external and internal activity. As for the external activity, its definition is given at the

end of this subsection.

Let $p(e)$ be a given deletion probability of an edge $e \in E$. Then, the *all-terminal network reliability* is defined as the probability that the graph remains connected after each edge e is deleted with the probability $p(e)$. This reliability will be simply denoted by $R(G)$.

In this general case, the following edge deletion/contraction formula holds.

Lemma 3 *For an edge e, $R(G)$ is expanded as follows.*

$$\begin{cases} (1-p(e))R(G/e) & e\text{: coloop} \\ R(G\backslash e) & e\text{: loop} \\ p(e)R(G\backslash e) + (1-p(e))R(G/e) & \text{otherwise} \end{cases}$$

This is essentially equivalent to the recursive formula of the Tutte polynomial. In fact, when all $p(e)$ are identical, the following holds.

Theorem 10

$$R(G;p) = p^{|E|-\rho(E)}(1-p)^{\rho(E)}T(G;1,1/(1-p))$$

It is readily seen that the BDD-based approach can be applied to the general case such that the edge deletion probabilities are distinct. For this network reliability problem, we report computational results for some typical cases.

4.2 Computational Results

s for test networks, we considered a $k \times k$ lattice graph (or, grid graph) of $n = k^2$ vertices and $2k(k-1)$ edges. Computations are done by using SUN workstations. For large-size problems, we use SUN Ultra 60 with 2GB memory, where our programs only use at most around 500MB memory. As is seen in Imai, Sekine and Imai [8], we can solve a graph having some planar proximity relations of up to $50 \sim 60$ vertices and $150 \sim 180$ edges.

We show in Fig. 3 graphs of the reliability polynomials of $L_{k,k}$ for $k = 2$ to 10. Note that $L_{10,10}$ has 100 vertices and 180 edges. Its size may not be large, but it is definitely of moderate size. Since the algorithm in section 4.1 is a mildly exponential algorithm and the lattice graph has a nice ordering with small elimination front (size at most k), we can solve such moderate-size problem in practice.

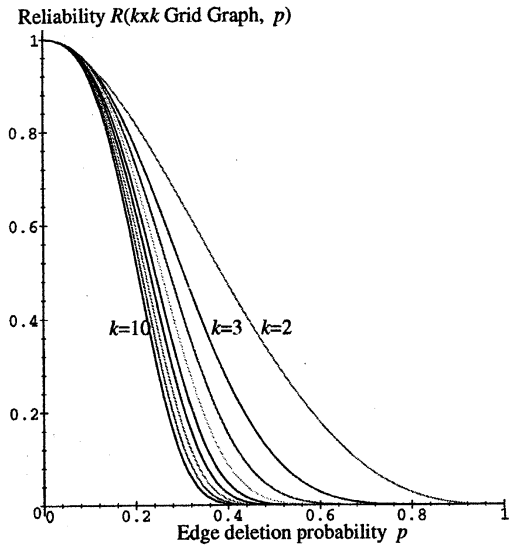

Figure 3: $R(L_{k,k};p)$ $(k = 2,\ldots,10)$

It is observed that the reliability is monotonically decreasing as k increases for the lattice graphs.

5 Concluding Remarks

This paper emphasizes computational aspects of the Tutte polynomial. For the deep theory of the Tutte polynomial from the viewpoint of discrete mathematics, see [3, 26]. The computational approach described here has potential to solve computationally hard problems rigorously in practice when it is of moderate size. There still seem much more applications of this approach.

Acknowledgment

Part of this work of the authors was supported by the Grant-in-Aid on Priority Area 'Algorithm Engineering' of the Ministry of Education, Science, Sports and Culture of Japan.

References

[1] N. Alon, A. Frieze, and D. Welsh, Polynomial time randomized approximation schemes for Tutte-Grothendieck invariants: the dense case, Random Structures Algorithms, vol.6, no.4, pp.459–478, 1995.

[2] R. E. Bryant, Graph based algorithms for Boolean function manipulation, IEEE Trans. on Computers, vol.C-35, pp.677–691, 1986.

[3] T. Brylawski and J. Oxley, The Tutte Polynomial and Its Applications, in Matroid Applications, ed. N. White, Cambridge University Press, pp.123–225, 1992.

[4] C. J. Colbourn, The Combinatorics of Network Reliability, Oxford University Press, 1987.

[5] D. D. Harms, M. Kraetzl, C. J. Colbourn, and J. S. Devitt, Network Reliability: Experiments with a Symbolic Algebra Environment, CRC Press, Inc., 1995.

[6] K. Hayase and H. Imai, OBDDs of a monotone function and of its prime implicants," Theory of Computing Systems, vol.31, pp.579–591, 1998.

[7] H. Imai, Network Reliability Computation and Related Issues — New Trends in Combinatorial Enumeration/Counting (in Japanese), in Discrete Structures and Algorithms V, ed. S. Fujishige, Kindai-Kagaku-sha, pp.1–50, 1998.

[8] H. Imai, K. Sekine and K. Imai, Computational investigations of all-terminal network reliability via BDDs. IEICE Trans. Fundamentals, vol.E82-A, no.5, pp.714–721, 1999.

[9] H. Imai, S. Iwata, K. Sekine and K. Yoshida, Combinatorial and geometric approaches to counting problems on linear matroids, graphic arrangements and partial orders, Lecture Notes in Computer Science, vol.1090, Springer-Verlag, 1996, pp.68–80.

[10] D. R. Karger, A randomized fully polynomial time approximation scheme for the all terminal network reliability problem, Proc. of the 27th Annual ACM Symp. on Theory of Computing, pp.11–17, 1995.

[11] D. Karger and R. P. Tai, Implementing a fully polynomial time approximation scheme for all terminal network reliability, Proc. of the SIAM-ACM Symp. on Discrete Algorithms, pp.334–343, 1997.

[12] R. J. Lipton and R. E. Tarjan, A separator theorem for planar graphs, SIAM J. on Appl. Math., vol.36, no.2, pp. 177–189, 1979.

[13] J. Oxley, Matroid Theory, Oxford University Press, Oxford, 1992.

[14] J. S. Provan, The complexity of reliability computations in planar and acyclic graphs, SIAM J. Comput., vol.15, no.3, pp.694–702, 1986.

[15] K. Sadakane, K. Hayase and H. Imai, A parallel top-down algorithm to construct binary decision diagrams (in Japanese), IPSJ SIG Notes SIGAL-48-11, IPSJ, 1995.

[16] K. Sekine, Algorithm for Computing the Tutte Polynomial and Its Applications, Doctoral Thesis, Department of Information Science, University of Tokyo, 1997.

[17] K. Sekine and H. Imai, A unified approach via BDD to the network reliability and path numbers, Tech. Rep. 95-09, Department of Information Science, University of Tokyo, 1995.

[18] K. Sekine and H. Imai, Counting the number of paths in a graph via BDDs," IEICE Trans. Fundamentals, vol.E80-A, no.4, pp.682–688, 1997,

[19] K. Sekine and H. Imai, A mildly exponential algorithm for computing the Tutte polynomial of a graph, submitted, 1999.

[20] K. Sekine, H. Imai and K. Imai, Computation of the Jones polynomial (in Japanese), Trans. JSIAM, vol.8, no.3, pp.341–354, 1998.

[21] K. Sekine, H. Imai, and S. Tani, Computing the Tutte polynomial of a graph of moderate size, Lecture Notes in Computer Science, vol.1004, pp.224–233, 1995.

[22] D. R. Shier, Network Reliability and Algebraic Structures, Oxford University Press, 1991.

[23] W. T. Tutte, A contribution to the theory of chromatic polynomials, Canadian J. Math., vol.6, pp.80–91, 1954.

[24] L. G. Valiant, The complexity of enumeration and reliability problems, SIAM J. Comput., vol.8, no.3, pp.410–421, 1979.

[25] D. J. A. Welsh, Matroid Theory, Academic Press, London, 1976.

[26] D. J. A. Welsh, Complexity: Knots, Colourings and Counting, Cambridge University Press, 1993.