Vol.2012-AL-138 No.8
2012/1/28

確率的な枝重みつき無向グラフ上の二点間最短路長さ分布の近似計算手法

Ei Ando †1 Joseph Peters†2

本稿では，無向グラフ G における二点間の確率的な最短路長さの分布関数 $B_G(x)$ に対する近似アルゴリズムを提案する．確率的な最短路問題は厳密に解く事が困難なことが多く，枝長さが二つの離散的な値を取り得るような場合，最短路長さの確率分布を求める問題は $\#P$-完全である．一方で，枝長さの分布とグラフの構造の両方に良い性質がある場合には確率分布関数 $B_G(x)$ を効率的に計算できる．本稿では，二点間の確率的な最短路問題の分布関数を求める問題について，まず枝長さが二値分布に従う場合にこの問題が $\#P$-完全であることを示す．次に，連続的な分布に従う枝長さを考え，それらがある自然な条件を満たし，かつグラフ G の treewidth が定数 k 以下である場合を考える．このとき x の多項式 $\tilde{B}_G(x)$ として $B_G(x)$ を近似することを考え，ある正の実数 ε, w が与えられるとき，$|B_G(x) - \tilde{B}_G(x)| \leq \varepsilon$ を $0 \leq x \leq w$ の範囲で満たすような x を，x の上限 w，枝の数 m，許容誤差の逆数 $1/\varepsilon$ の多項式時間で $B_G(x)$ を計算できる事を示す．

Approximating the Stochastic Shortest Path Length Between Two Vertices

Ei Ando †1 and Joseph Peters†2

In this paper, we propose an approximation algorithm for computing the distribution function $B_G(x)$ of the stochastic shortest path length between two vertices in undirected graph G. The stochastic shortest path problem is hard to solve; computing the exact stochastic shortest path lengths' distribution function is $\#P$-complete if we assume that the edge lengths can take two discrete values. We show that, however, if we consider some continuously distributed edge lengths satisfying some natural conditions, we can approximately compute the distribution function $B_G(x)$ of the stochastic shortest path length of G. Our approximation algorithm outputs a polynomial $\tilde{B}_G(x)$ that approximates $B_G(x)$ for $0 \leq x \leq w$ and satisfies that $|B_G(x) - \tilde{B}_G(x)| \leq \varepsilon$, where w and ε are positive values. The running time of our algorithm is polynomial in the graph size, w and $1/\varepsilon$ if G has a constant treewidth k.

1. Introduction

Let $G = (V, E)$ be a graph with vertex set $V = \{v_0, \ldots, v_{n-1}\}$ and edge set $E \subseteq \{\{u, v\} | u, v \in V\}$. We associate an *edge length* X_e for each edge $e \in E$; here we assume that X_e's are mutually independent random variables. We consider the problem of computing the shortest path length between two vertices $s, t \in V$ in G; here we use the random edge lengths X_e for each $e \in E$. Note that any $s - t$ path between G can be the shortest path because the shortest path can vary depending on the realization of the random edge lengths. Since we cannot determine the shortest path as a single path in the stochastic version of the problem, there can be at least two kinds of approaches: (1) By re-defining the shortest path including the probability, try to find a single shortest path; or (2) we do not specify which is the shortest path but consider the 'shortest path length' as a random variable and compute its distribution function. We take the latter approach, that is, we are to compute the distribution function $B_G(x)$ of the shortest path length.

The problem of computing the distribution function of this kind is, however, usually hard to solve. Hagstrom[8] proved, using the transportation graph, that the problem of computing the stochastic longest path length in directed acyclic graphs is $\#P$-complete if the edge lengths can take 0 or 1. Ball et al. writes about the same problem in[3] that the problem is NP-hard for series-parallel graphs when the edge lengths can take two different values. In this paper, we give a simple $\#P$-completeness proof for computing the distribution function of the sum of the mutually independent random variables that can take 0 or a positive integer. Our proof shows that if we consider a stochastic version of the optimization problem such as the shortest path problem, the minimum spanning tree problem, maximum matching problem, etc., the problem of computing the distribution function of the optimal solutions' weight is $\#P$-complete even for a path graph.

†1 Faculty of Computer and Information Science, Sojo University, 4-22-1, Ikeda, Kumamoto.
†2 School of Computing Science, Simon Fraser University, Burnaby, Canada.

On the contrary, by defining a parameter k of a graph, we showed, in[1], that the problem of computing the shortest path length's distribution can be solved in polynomial time in the graph size, if the parameter k is bounded by a constant, and the edge lengths obey the horizontal shifts of the exponential distribution with expectation 1. In addition, we show, in the paper, that there is an FPTAS for the problem of computing the distribution of the shortest path length if the given graph has treewidth less than a constant k and the random edge lengths obeys continuous distributions satisfying some natural conditions.

Here we make some notes about the treewidth. According to[7], the treewidth is defined as follows, using the tree decomposition.

Definition1 A tree decomposition of $G = (V, E)$ is a pair $(\{U_i \mid i \in I\}, T)$, where $\{U_i \mid i \in I\}$ is a family of subsets of V and T is a tree with vertex set I such that

(1) $\bigcup_{i\in I} U_i = V$,

(2) for all edges $(x, y) \in E$, there is an element $i \in I$ such that $x, y \in U_i$,

(3) for all triples $i, j, k \in I$, if j is on the path from i to k in T, then $U_i \cap U_k \subseteq U_j$.

The width of a graph decomposition $(\{U_i \mid i \in I\}, T)$ is given by $\max_{i\in I}\{|U_i| - 1\}$. The treewidth of a graph G is defined as the minimum width taken all over tree decompositions of G.

According to Arnborg et al.[2], computing the exact treewidth is an NP-hard problem and corresponding decision problem is NP-complete. However, checking if a graph G has treewidth less than a fixed integer k can be answered efficiently as long as k is not included in the problem instance[5]. It is also shown in[5] that once a graph G is known to be treewidth k graph, we can construct a tree decomposition efficiently. There is an approximation algorithm that is proposed by Bouchitté et al.[4]. Their algorithm outputs an upper bound on the graph k and has approximation ratio $O(\log k)$ for the treewidth k graph.

The paper is organized as follows. In Section 2, we show our proof for the $\#P$-completeness of the computing the distribution of the sum of discrete random variables. In Section 3, we show our exact algorithm for computing the shortest path length's distribution function. In Section 4, we show our approximation algorithm. The paper is concluded in Section 5.

2. #P-Completeness of Computing a Sum's Distribution

We here explain the $\#P$-completeness of computing the distribution function of a sum of discrete random variables. The arguments in this section apply to the problem of computing the distribution function of many stochastic version of the optimization problems. We say a random variable X obeys a *two-values distribution* if X can take value 0 with probability 1/2 and value a value $c \neq 0$ otherwise. Here we consider the following problem.

Definition2 Let $X_1, \ldots, X_n$ be n mutually independent random variables, where X_i's obey two-values distributions; X_i can take 0 or a positive integer c_i with probability 1/2 for each values. In problem SUM-PDF, we are to compute the distribution function $F(x)$ of the sum $X = \sum_{i=1,\ldots,n} X_i$. That is, we compute the probability that X is less than or equal to a given integer x.

Then we can prove the following theorem.

Theorem1 SUM-PDF is $\#P$-complete.

To prove the theorem, we need to show the following Lemmas 1.

Lemma1 SUM-PDF is in $\#P$.

Proof The distribution function $F(x)$ is given by counting the number $N(x)$ of combinations of $X_1, \ldots, X_n$ where the sum X is less than x; we have that $F(x) = N(x)/2^n$. □

Lemma2 SUM-PDF is $\#P$-hard.

Proof Let SUM-PMF be another problem in which we are to compute the probability mass function $F'(x)$ of the sum $X = \sum_{i=1,\ldots,n} X_i$. That is, given an integer x, we are to compute the probability that X is equal to x. Then, since the probability mass function can be computed by $F'(x) = F(x) - F(x-1)$, SUM-PMF can be solved in polynomial time if we could solve SUM-PDF.

We can see that SUM-PMF is actually equivalent to the counting version of the following SUBSET SUM[6], which is proved to be a $\#P$-complete problem by Simon[11] using Karp's reduction in[10]. In SUBSET SUM problem, we are given a set $A = \{a_1, \ldots, a_n\}$ of positive integers and one more integer B. Since we are considering counting version of the problem, we are to compute the number of subsets A' of A satisfying $\sum_{a\in A'} a = B$. Then, if we could solve SUM-PMF in polynomial time, we can solve SUBSET SUM. We set all $c_i = a_i$ for all $i = 1, \ldots, n$. Then if we could compute $F'(x)$, we have the answer of the counting version of SUBSET SUM, that is, $F'(x)2^n$.

Therefore, we can solve the counting version of SUBSET SUM, if we could solve SUM-PDF, which proves this lemma. □

Then we have the proof of Theorem 1.

Proof Since SUM-PDF is in $\#P$ and $\#P$-hard by Lemma 1 and 2, we have Theorem 1. □

Now let us consider another problem for comparison.

Definition3 Let $X_1, \ldots, X_n$ be n mutually independent random variables. We here assume that X_i for $i = 1, \ldots, n$ obey the horizontal shifts of the exponential distribution with expectation 1, that is,

$$P(X_i \le x) = H(x - c_i)\exp(-x + c_i), \tag{1}$$

where $H(x-c_i)$ is a step function satisfying $H(x-c_i) = 0$ for $x \le c_i$ and $H(x-c_i) = 1$ for $x > c_i$. Then, SUM-PDF-EXP is a problem in which we are to compute the distribution function $F(x)$ of the sum $X = \sum_{i=1,\ldots,n} X_i$.

Here we can see that SUM-PDF-EXP can be solved efficiently: It is well known that the sum of mutually independent and exponentially distributed random variables are given as a gamma distribution. SUM-PDF-EXP can be solved in linear time in the input size, which makes a contrast to the discrete version of the problem.

One may wonder where this difference comes from. We conjecture that the exponential distribution makes the problem easier because the distribution is given by a uniform formula whose density has only one peak. In fact, we could consider some other discrete distributions, including the binomial distribution, that make the problem easy. It is also possible that the two-values distribution of SUM-PDF is one of the hardest distribution to deal with, despite of its simple appearance.

We are now interested in a question: To what extent can the problem be easier when we assume some well-behaved continuous distributions ? This is important because the stochastic network analysis with continuously distributed random variables has many applications. As for the shortest path length's distribution function in a graph with random edge lengths, we answer this question partly in the following.

3. Algorithm for the Case where Edge Lengths are Continuously Distributed

In this section, we briefly introduce our algorithm SPL-PDF that computes the exact distribution function of the stochastic shortest path length between $s = v_0$ and $t = v_{n-1}$, if we could compute the integrals in SPL-PDF. In our previous work[1], we showed that SPL-PDF computes the distribution function of the stochastic shortest path length. Although there is a slight change between the algorithm in the previous work, the correctness of the algorithm does not be spoiled by the change: the difference from the algorithm in[1] is only in the order of processing the edges, which does not change the output of the algorithm. As for the correctness proof of SPL-PDF, see[1].

3.1 Creating the Initial Graph S_0

Given an undirected graph $G = (V, E)$ with n vertices and m edges, we construct a graph $G_J = (V_J, E_J)$ with vertex set $V_J = V \cup \{v_e, u_e \mid e = \{u, v\} \in E \text{ and } u, v \in V\}$ and edge set $E_J = J_1 \cup J_2$, where $J_1 = \{\{u, u_e\}, \{v_e, v\} \mid u, v \in V, e = \{u, v\} \in E\}$ and $J_2 = \{\{u_e, v_e\} \mid u, v \in V, e = \{u, v\} \in E\}$. In other words, we replace each edge $e = \{u, v\}$ of G by a path $\{u, u_e\}, \{u_e, v_e\}, \{v_e, v\}$ containing two new vertices u_e and v_e. We assign a fixed edge length $\epsilon > 0$ that is arbitrary close to 0 to each edge in J_1. Each edge $\{u_e, v_e\} \in J_2$ is assigned the edge length of $\{u, v\} \in E$. We call the vertices $u_e, v_e \in V_J \setminus V$ the *joint vertices*. A joint vertex $v \in V_J \setminus V$ is *open* in subgraph G'_J of G_J if there is some edge e incident on v in G_J but e is missing in G'_J. Fig. 1 shows an example of G_J generated from a graph G. We use $\mathcal{X}_0$ to denote the association of each joint vertex $v \in V_J \setminus V$ with a variable x_v which is the distance between the source $s = v_0$ and v. Thus, $\mathcal{X}_0 = \{(v, x_v) \mid v \in V_J \setminus V\}$. The subgraph S_0 of G_J is obtained by removing the edges in J_2 from G_J. Let $e_1, \ldots, e_m \in J_2$ be ordered according to a subroutine BOTTOMUP-TD that we show later. Then graph S_i for $i = 1, \ldots, m$ is a graph which is given by adding edge $e_1, \ldots, e_i$ to S_0. Also, we consider the correspondence between the dummy variables for the distances from s to each open joint vertices in $S_1, \ldots, S_m$. The set $\mathcal{X}_i$ for $i = 1, \ldots, m$ is given by removing the pairs (v, x_v) from

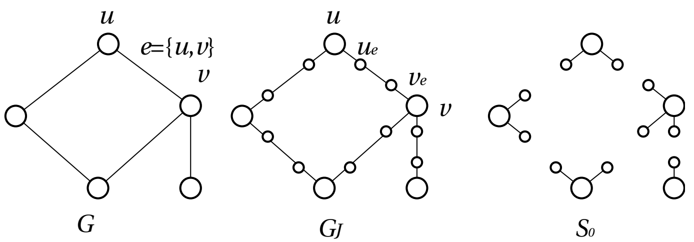

図 1 An example of a graph G (left), G_J (centre), and S_0 (right). Smaller circles are the joint vertices. No joint vertices are open in G_J; all joint vertices are open in S_0.

$\mathcal{X}_0$ where v is a joint vertex that appears in either one of edges $e_1, \dots, e_i$.

The description of the subroutine BOTTOMUP-TD(G) is the following. The idea of BOTTOMUP-TD is to output the edges from the leaves of the tree decomposition to the edges that are closer to the root.

Algorithm BOTTOMUP-TD(G)

1. Compute a tree decomposition $(\{U_i \mid i \in I = \{1, \dots, r\}\}, T)$ of G_J so that $s \in U_1$ and $\max_{i \in I}\{|U_i| - 1\} \leq k$;
2. Let A be $\emptyset$;
3. For each U_i $i \in I$ in the BFS order from U_i;
4. For each edge $e = \{u, v\} \subseteq U_i$;
5. Add edge e to A at the tail;
6. Output the reverse of A.

3.2 Labelling variables and Label Choosing Operations

We introduce *labelling variables* and *label choosing operations*. The labelling variables and label choosing operations are for simplifying the description of the algorithm. For each $v \in V$, there may be two labelling variables v^o, v^i and two label choosing operations R_v^o, R_v^i. The label choosing operation R_v^o (resp. R_v^i) is used in the algorithm (in the next subsection) to compute the sum of the terms that include the labelling variable v^o (resp. v^i) as a factor.

3.3 Algorithm Description

In our algorithm, we consider a probability $B_i(S_i, \mathcal{X}_i, x)$ for $i = 0, \dots, m$ of the event where each variables in $\mathcal{X}_i$ is greater than the the shortest path lengths between s and the corresponding open joint vertices in $\mathcal{X}_i$. The following is the definition of $B_0(S_i, \mathcal{X}_i, x)$.

Definition4

$$B_s(S_0, \mathcal{X}_0) = \prod_{v \in V_s} v^o H(x_u) \tag{2}$$

$$B_t(S_0, \mathcal{X}_0, x) = \sum_{v \in V_t} v^i H(x - x_v) \prod_{w \in V_t \setminus \{v\}} w^o H(x_w - x_v) \tag{3}$$

$$B_u(S_0, \mathcal{X}_0) = \sum_{v \in V} v^i \prod_{w \in V_u \setminus \{v\}} w^o H(x_w - x_v) \tag{4}$$

$$B_0(S_0, \mathcal{X}_0, x) = B_s(S_0, \mathcal{X}_0) B_t(S_0, \mathcal{X}_0, x) \prod_{u \in V \setminus \{s,t\}} B_u(S_0, \mathcal{X}_0). \tag{5}$$

By $V_v \subseteq V$, we denote the set of vertices that are adjacent to $v \in V$. We note that $H(x)$ is a step function; $H(x) = 0$ if $x \leq 0$ and $H(x) = 1$ $x > 0$. Now the following is the description of SPL-PDF.

Algorithm SPL-PDF(G)

1. Construct S_0 from G;
2. Set i:=0;
3. Compute $B_0(S_0, \mathcal{X}_0, x)$ using Definition 4;
4. For each $e = \{u, v\} \in J_2$ in order given by BOTTOMUP-TD(G) do
5. Let S_{i+1} be the graph that is obtained by adding e to S_i;
6. Let $\mathcal{X}_{i+1} = \mathcal{X}_i \setminus \{(u, x_u), (v, x_v)\}$, where x_u and x_u are the distance from u, v to s, respectively;
7. Compute $B_{i+1}(S_{i+1}, \mathcal{X}_{i+1}, x)$ by the following;

$$B_{i+1}(S_{i+1}, \mathcal{X}_{i+1}, x) = P_1(S_i, \mathcal{X}_{i+1}, x; e) + P_2(S_i, \mathcal{X}_{i+1}, x; e) + P_3(S_i, \mathcal{X}_{i+1}, x; e), \tag{6}$$

$$P_1(S_i, \mathcal{X}_{i+1}, x; e) = \int_{\mathbb{R}^2} \left(\frac{\partial}{\partial x_v} R_v^o(R_u^i(B_i(S_i, \mathcal{X}_i, x))) \right) f_e(x_u - x_v) dx_u dx_v, \tag{7}$$

$$P_2(S_i, \mathcal{X}_{i+1}, x; e) = \int_{\mathbb{R}^2} \left(\frac{\partial}{\partial x_u} R_v^i(R_u^o(B_i(S_i, \mathcal{X}_i, x))) \right) f_e(x_v - x_u) dx_v dx_u, \tag{8}$$

$$P_3(S_i, \mathcal{X}_{i+1}, x; e) = \int_{\mathbb{R}^2} \left(\frac{\partial}{\partial x_u} \frac{\partial}{\partial x_v} R_v^o(R_u^o(B_i(S_i, \mathcal{X}_i, x))) \right) (1 - F_e(|x_v - x_u|)) dx_u dx_v; \tag{9}$$

8. Replace the labelling variables u^i, u^o, v^i, and v^o by 1 in $B_{i+1}(S_{i+1}, \mathcal{X}_{i+1}, x)$;
9. Set i:=$i + 1$;
10. Output $B_m(S_m, \mathcal{X}_m, x)$ as $B_G(x)$, where $S_m = G_J$ and $\mathcal{X}_m = \emptyset$ at this point.

To make our algorithm SPL-PDF work, we need some implementation for executing the integrals. In the next section, we consider approximately executing the integrals by using Taylor approximation.

4. Approximation Algorithm for the Shortest Path Length's Distribution

In this section, we show an algorithm for approximately computing the distribution function of the shortest path length. We give the approximation algorithm by slightly changing SPL-PDF. The idea of our approximation algorithm is to approximately compute the integrals in SPL-PDF by using the Taylor polynomials. Our approximation algorithm can be applied if the edge lengths are mutually independent and their distribution functions have the following three properties:

(1) For the distribution function $F_e(x)$ of the edge length of e is 0 if $x \leq 0$;
(2) the Taylor series of $F_e(x)$ generated at $x = 0$ converges to $F_e(x)$ for any $x > 0$;
(3) given an upper bound w on x, for any nonnegative integer i satisfying $0 \leq i \leq p$, the i-th derivative $\left(\frac{d}{dx}\right)^i F_e(x)$ is less than 1 for all $0 \leq x \leq w$.

The following is the approximation algorithm APPROX-SPL-PDF that computes the approximating polynomial of the shortest path length's distribution function $B_G(x)$. The idea of APPROX-SPL-PDF is that it computes the approximation $A_i(S_i, \mathcal{X}_i, x)$ of $B_i(S_i, \mathcal{X}_i, x)$ that is in algorithm SPL-PDF. APPROX-SPL-PDF accepts three inputs: the input graph G, a positive integer p, and a real number $w \geq 0$. The second input p is the degree of Taylor polynomial with which we use to approximate the ongoing computation of the integrals. To compute an approximating polynomial $\tilde{B}_G(x)$ of $B_G(x)$ so that the difference between the $B_G(x)$ and $\tilde{B}_G(x)$ is less than ε for $0 \leq x \leq w$, our algorithm finishes in a polynomial time in n, m, ε and w.

Algorithm APPROX-SPL-PDF(G, p, w)

1. Construct S_0 from G;
2. Set i:=0;
3. Compute $A_0(S_0, \mathcal{X}_0, x) = B_0(S_0, \mathcal{X}_0, x)$ using Definition 4;
4. For each $e = \{u, v\} \in J_2$ in order given by BOTTOMUP-TD(G) do
5. Let S_{i+1} be the graph that is obtained by adding e to S_i;
6. Let $\mathcal{X}_{i+1} = \mathcal{X}_i \setminus \{(u, x_u), (v, x_v)\}$, where x_u and x_u are the distance from u, v to s, respectively;
7. Compute $A_{i+1}(S_{i+1}, \mathcal{X}_{i+1}, x)$ by the following;
$A_{i+1}(S_{i+1}, \mathcal{X}_{i+1}, x) = Q_1(S_i, \mathcal{X}_{i+1}, x; e) + Q_2(S_i, \mathcal{X}_{i+1}, x; e) + Q_3(S_i, \mathcal{X}_{i+1}, x; e)$,
where $Q_j(S, \mathcal{X}', x; e)$ is the output of APPROX-INTEGRAL($P_j(S, \mathcal{X}, x; e), x_u, x_v$) for $j = 1, 2, 3$ and

$$P_1(S_i, \mathcal{X}_{i+1}, x; e) = \int_{\mathbb{R}^2} \left(\frac{\partial}{\partial x_v} R_v^o(R_u^i(A_i(S_i, \mathcal{X}_i, x)))\right) f_e(x_u - x_v) dx_u dx_v, \tag{10}$$

$$P_2(S_i, \mathcal{X}_{i+1}, x; e) = \int_{\mathbb{R}^2} \left(\frac{\partial}{\partial x_u} R_v^i(R_u^o(A_i(S_i, \mathcal{X}_i, x)))\right) f_e(x_v - x_u) dx_v dx_u, \tag{11}$$

$$P_3(S_i, \mathcal{X}_{i+1}, x; e) = \int_{\mathbb{R}^2} \left(\frac{\partial}{\partial x_u}\frac{\partial}{\partial x_v} R_v^o(R_u^o(A_i(S_i, \mathcal{X}_i, x)))\right)(1 - F_e(|x_u - x_v|)) dx_u dx_v; \tag{12}$$

8. Replace the labelling variables u^i, u^o, v^i, and v^o by 1 in $A_{i+1}(S_{i+1}, \mathcal{X}_{i+1}, x)$;
9. Set i:=$i + 1$;
10. Output $A_m(S_m, \mathcal{X}_m, x)$ as $\tilde{B}_G(x)$, where $S_m = G_J$ and $\mathcal{X}_m = \emptyset$ at this point.

The subroutine APPROX-INTEGRAL is used in step 7. of APPROX-SPL-PDF. APPROX-INTEGRAL works for a double definite integral $D(\mathcal{X}, x)$ that has, as its integrand, a product of sum-of-products-form polynomials, label removing operations, a edge length's distribution (or density) function $F_e(x_v - x_u)$, and the differentiation symbol $\partial/\partial x_u$, $\partial/\partial x_v$.

The basic idea of APPROX-INTEGRAL is the following 3 steps: (1) Expand a part of the integrand into a sum of products; (2) Approximate the part of the integrand by the Taylor polynomial of degree p; (3) Execute the integral. Note that, to keep the integrand's expression as short as possible, we do not expand the entire integrand in the step (1). We say that a *clause* of a polynomial $D(\mathcal{X}, x)$ is a factor of $D(\mathcal{X}, x)$ that may be a constant, a variable, a function, or a sum of products of these items. Before we approximate the integrand, we expand the product of clauses that includes the variables x_u and x_v into a sum of products, making a new clause. Note that we do not expand the integrand any further as long as the dummy variable of the integral does not appear in multiple clauses. We note that any clause corresponds to a connected component of S_i throughout the execution of the algorithm APPROX-SPL-PDF when we expand the clauses in this way. For example, at the beginning of APPROX-SPL-PDF, we consider the factors of $B_0(S_0, \mathcal{X}_0, x)$ int Definition 4 as the clauses: $B_s(S_0, \mathcal{X}_0)$, $B_t(S_0, \mathcal{X}_0, x)$, and $B_u(S_0, \mathcal{X}_0, x)$ for each $u \in V \setminus \{s, t\}$. It is clear that these factors corresponds to the connected components of S_0. Then it is easy to see that multiplying a function that includes two variables x_u and x_v, and expanding the two clauses and this function into a sum of products makes a new clause that corresponds to a connected component in S_1 given by adding edge $\{u, v\} \in J_2$ to S_0. Then we can execute the double integral in the clause with respect to x_u and x_v after approximating the clause by a Taylor polynomial including x_u and x_v. Also, note that the three polynomials $Q_1(S_i, \mathcal{X}_{i+1}, x), Q_2(S_i, \mathcal{X}_{i+1}, x)$ and $Q_3(S_i, \mathcal{X}_{i+1}, x)$ have the other clauses in common, which means that $A_{i+1}(S_{i+1}, \mathcal{X}_{i+1}, x)$ can be easily factorized into a product of clauses.

Algorithm APPROX-INTEGRAL$(D(\mathcal{X}, x), x_u, x_v)$

1. Process the partial differentiations in the integrand of $D(\mathcal{X}, x)$;
2. Let $U(\mathcal{X}, x)$ be a polynomial that consists in the step function, the labelling variables, and variables in $\mathcal{X}$ satisfying

$$D(\mathcal{X}, x) = \iint_{\mathbb{R}^2} U(\mathcal{X}, x) F_e(x_u - x_v) dx_u dx_v; \tag{13}$$

4. Expand the product of clauses of $D(\mathcal{X}, x)$ including the variables x_u and x_v;
5. Let $D'(\mathcal{X}, x)$ be given by replacing the clause including x_u and x_v in the integrand of $D(\mathcal{X}, x)$ by Taylor approximation of degree p at the point where all variables are equal to 0;
6. Let $Q(\mathcal{X}', x)$ be the resulting form of executing the integrals of $D'(\mathcal{X}, x)$ with respect to x_u and x_v;
7. Let $Q'(\mathcal{X}', x) = Q(\mathcal{X}', x)/\tau$ where $\tau = 1 + (k+1)^p w^{p+2}/(p+1)!$;
8. Output the resulting form $Q'(\mathcal{X}', x)$.

We execute step 7. of APPROX-INTEGRAL in order to bound the approximation error in the proof.

By using p as a parameter, we prove the following theorem.

Theorem2 Let G be a treewidth k graph and p be a positive integer. Algorithm APPROX-SPL-PDF finishes in $O(4^{4(k+2)^2}(p+2)^{2(k+1)}4^{2k}m)$ time.

Proof We prove the running time by bounding the number of terms in the description of $B_i(S_i, \mathcal{X}_i, x)$. The point of the proof is that we have at most $k+1$ variables (i.e., k variables for the distances from s to k vertices and the shortest path length x) for the distribution function of one connected component in S_i.

Let us define some necessary symbols and words. We assume that the order of the edges $e_1, e_2, \ldots, e_m \in E$ is given by the BOTTOMUP-TD(G). Let $C_{i,v}$ be the connected component of S at the i-th execution of APPROX-SPL-PDF's loop. Remember that we expand the clause that correspond to $C_{i,v}$ into a sum of products after computing the integrals at step 7. The factors of each term in the clause can be separated into the step functions and the others; we call the earlier *the step function part* and the latter *the elementary function part*. In the following we bound the number of possible step function parts and elementary function parts.

Here we prove that the running time of processing the step function parts is $O(2^{4(k+2)^2})$ per clause.

Let us first see that the coefficients of the variables in the step functions' arguments may be 1, -1 and 0. At the beginning, all the coefficients of the variables that appears in the step functions of $B_0(S_0, \mathcal{X}_0, x)$ are one of 1, -1 or 0. Then, it is easy to see that neither multiplying the step functions nor differentiating the step functions does not make any change to none of the coefficients of the variables in the step functions' arguments. Then, since executing an integral of a term with respect to a variable x_v in APPROX-SPL-PDF causes only the replacement of x_v by another single variable, we can see that execution of the integral does make coefficients of the variables other than 1, -1 or 0. Therefore, at any point of the loop of APPROX-SPL-PDF, the variables in the arguments of step functions are one of 1, -1 or 0.

We next prove that there are exactly two variables in the step functions' arguments. It is clear that there are exactly two variables in the arguments of every step functions of $B_i(S_i, \mathcal{X}_i, x)$ at the beginning of APPROX-SPL-PDF. Then, again, neither multiplying nor differentiating the step functions does not change the number of the variables in the step functions' arguments. Let us assume that all step functions' arguments in $B_i(S_i, \mathcal{X}_i, x)$ have two variables each at a i-th execution of the loop of APPROX-SPL-PDF. Suppose that we are going to integrate a term with respect to a variable x_v. Since the step function part of a term may define the upper limit or the lower limit of the definite integral of x_v, executing an integral replaces x_v by another single variable, which means that we still have that all arguments consists of exactly two variable at the $(i+1)$-th execution of the loop of APPROX-SPL-PDF. Therefore, there are exactly two variables in any step function's argument.

Let us see that the two variables in the step functions' arguments are chosen from at most $k+2$ variables in each factor of $B_i(S_i, \mathcal{X}_i, x)$. Remember that we use the output of procedure BOTTOMUP-TD(G) as the order of edges. Since the given graph G and its joint graph G_J has treewidth k, there are at most k open joint vertices in connected component $C_{i,v}$. Then we have one variable x that is the broadcast time and k variables that corresponds to the distance from s to the open joint vertices.

Now we are ready to see that there are at most $2^{4(k+2)^2}$ step function parts per one clause that corresponds to a connected component of S. Since there are at most $k+2$ variables, there are at most $(2(k+2))^2$ step function factors. Then the number of possible combinations of the step function factors in a clause is bounded by $2^{4(k+2)^2}$.

We proceed to bounding the number of elementary function part of a clause.

Let S_i have ℓ open joint vertices at the i-th execution of the loop of APPROX-SPL-PDF. Since we approximate the edge length's distribution functions by a p-th Taylor polynomial, the elementary function part of the terms a clause that corresponds to a connected component can be expanded into a sum of the following form

$$C x_1^{a_1} x_2^{a_2} \dots x_\ell^{a_\ell}, \tag{14}$$

where C is a constant that is given for each term, $x_1, x_2, \dots, x_\ell$ are the distance from s to ℓ joints, $a_1, \dots, a_\ell$ are nonnegative integers. Especially, we have that $0 \le a_j \le p+2$ for all $j = 1, \dots, \ell$. It amounts to that there are at most $(p+2)^{k+1}$ elementary function parts in a closure that corresponds to a connected component of S_i.

Since there can be 2^{2k} labelling variables' combinations per one term, we have that there are $2^{4(k+2)^2}(p+2)^{k+1}2^{2k}$ combinations of step function parts elementary function parts, and labelling variables in a closure that corresponds to a connected component of S.

The running time of processing the product of two clauses before executing an integral is the largest part of the running time. Since we expand the product of two clauses into a sum of products, the running time to process the expansion may be the square of the number of possible terms. That is, since we may have $2^{4(k+2)^2}(p+2)^{k+1}2^{2k}$ terms in one clause, the running time to process the product of two clauses is $4^{4(k+2)^2}(p+2)^{2(k+1)}4^{2k}$.

Note that the running time of computing the Taylor approximation and executing the integrals is relatively smaller and thus does not appear in the asymptotic evaluation of the running time.

Now multiplying the running time of processing the product of two clauses, and the number m of loop executions proves the theorem. □

Then, we show how large p is sufficient for having ε as the upper bound on the difference between our approximation and actual broadcast time distribution function for $0 \le x \le w$.

Theorem3 Let G be a treewidth k graph. Running APPROX-SPL-PDF with $p = O(k + w + \ln m + \ln 1/\varepsilon)$ is large enough for having the difference between $B_G(x)$ and the output $\tilde{B}_G(x)$ of APPROX-SPL-PDF less than ε for $0 \le x \le w$.

Proof Here we bound the difference between $B_G(x)$ and the output $\tilde{B}_G(x)$ of APPROX-SPL-PDF by using m, w and p.

Remember that, by assumption, the maximum value of $|\left(\frac{d}{dx}\right)^p F_e(x)|$ is less than 1 for $0 \le x \le w$. Consider the difference between $D(\mathcal{X}, x)$ and $D'(\mathcal{X}, x)$ in APPROX-INTEGRAL. Since G is treewidth k graph, there can be at most k open joints in the connected component of S_i that includes u and v; hence there are $k+1$ variables in the corresponding clause of $D(\mathcal{X}, x)$. Therefore, the difference between the value of $D(\mathcal{X}, x)$ and $D'(\mathcal{X}, x)$ can occur as the error of $k+1$ variables Taylor approximation, which is bounded by

$$|D(\mathcal{X}, x) - D'(\mathcal{X}, x)| \le \frac{(k+1)^p w^{p+2}}{(p+1)!}. \tag{15}$$

Then, since we divide the resulting form by $\tau = 1 + (k+1)^p w^p/(p+1)!$ in APPROX-INTEGRAL, we have that our approximation $A_i(S_i, \mathcal{X}_i, x)$ in step 7. of APPROX-SPL-

PDF does not get larger than $B_i(S_i, \mathcal{X}_i, x)$; however, $A_i(S_i\mathcal{X}_i, x)$ may get

$$\left(1 - \frac{(k+1)^p w^{p+2}}{(p+1)!}\right)/\tau \tag{16}$$

times smaller than $B_i(S_i, \mathcal{X}_i, x)$. Then, remembering that $B_i(S_i, \mathcal{X}_i, x)$ is less than 1 and that $1/(1+x) \geq 1 - x$ for $x > 0$, we have that the difference between $A_i(S_i, \mathcal{X}_i, x)$ and $B_i(S_i, \mathcal{X}_i, x)$ grows no more than $2(k+1)^p w^{p+2}/(p+1)!$ per one execution of the loop of APPROX-SPL-PDF.

Now we can consider the overall error of the approximation. Since we repeat this approximation for all edges, we have that the difference between the exact $B_G(x)$ and the output $\tilde{B}_G(x)$ of APPROX-SPL-PDF satisfies

$$|B_G(x) - \tilde{B}_G(x)| \leq \frac{2m(k+1)^p w^{p+2}}{(p+1)!}. \tag{17}$$

To make this smaller than a positive value ε, we have that $p = O(k + w + \ln m + \ln 1/\varepsilon)$ is large enough. □

Now we have the following corollary.

Corollary1 The problem of computing the value of distribution function of the stochastic shortest path length's distribution function has an FPTAS if the given graph G has treewidth less than a constant k.

5. Conclusions

In this paper, we proved that the problem of computing the distribution function of the sum of the discrete random variables is $\#P$-complete if the random variables obey the two-values distribution. It shows that, in many optimization problem with random weights, including the stochastic shortest path problem, computing the distribution function of the optimal solution's weight is $\#P$-complete if the weights can take two values. Then, we showed that there is an FPTAS for the problem of computing the shortest path length's distribution function if the given graph has treewidth less than a constant k and the edge lengths obey the continuous distributions with some conditions that allows us to use the Taylor approximation.

参　考　文　献

1) E.Ando and J. Peters, Computing the Shortest Path Length Distribution Between Two Vertices in Graphs with Random Edge Lengths, *LA Symposium*, pp.18-1 – 18-8, Jul, 2011.
2) S. Arnborg, D. Corneil, A. Proskurowski, Complexity of Finding Embeddings in a k-tree, *SIAM Journal of Algebraic Discrete Methods*, Vol. 8, No. 2, pp.277–284, 1987.
3) M. Ball, C. Colbourn, J. Provan, Network Reliability, Handbooks in Operations Research and Management Science, Vol. 7: Network Models, M. Ball, T. Magnanti, C. Monma, G. Nemhauser (eds.), Elsevier Science B.V., pp.673–762, 1995.
4) V. Bouchitté, D. Kratsch, H. Müller, I. Todinca, On Treewidth Approximations, *Discrete Applied Mathematics*, Vol. 136, pp.183–196, 2004.
5) H. Bodlaender, A Linear-Time Algorithm For Finding Tree-Decompositions of Small Treewidth, *SIAM Journal on Computing*, Vol. 25, No. 6, pp.1305–1317, 1996.
6) M. Garey, D. Johnson, Computers and Intractability, W. H. FREEMAN AND COMPANY, New York, 1979.
7) J. Gross, J. Yellen (eds.), Handbook of Graph Theory, CRC Press, 2003.
8) J. N. Hagstrom, Computational Complexity of PERT Problems, *NETWORKS*, Vol. 18, pp.139–147, 1988.
9) M. Jerrum, A. Sinclair, Conductance and the Rapid Mixing Property for Markov Chain: the Approximation of the Permanent Resolved, *STOC'88 proc. of the twentieth annual ACM symposium on Theory of computing*, pp.235–244, 1988.
10) R. Karp, Reducibility Among Combinatorial Problems, in R. E. Miller and J. W. Thatcher (eds.), *Complexity of Computer Computations*, Plenum Press, New York, pp,85–103, 1972.
11) J. Simon, On the Difference Between One and Many, Lecture Notes in Computer Science, 1977, Volume 57/1977, pp.480–491, DOI: 10.1007/3-540-08342-1_37.