Vol.2010-AL-131 No.3
2010/9/22

確率的な通信時間を持つネットワーク上でのブロードキャスト時間計算

Ei Ando †1 Joseph Peters†2

本稿では，計算機ネットワークでブロードキャストにかかる時間を計算することを考える．ただし，どの通信路を通して通信を行う場合にもランダムな時間（通信時間）がかかるものと考える．本稿では各通信路の通信時間を互いに独立な確率変数と考え，確率変数であるブロードキャスト時間も確率変数の確率分布を求める．この問題と類似の問題が $\#P$-完全に属するため，この問題も厳密に解決することが困難であることが予想される．このため，本稿ではブロードキャスト時間の確率分布関数 $F_B(x)$ を，ある与えられた $w > 0$ よりも小さい x について ϵ 以下の誤差で近似する多項式を求める．本稿では最初に環状のネットワークにおけるブロードキャスト時間分布を求めるアルゴリズムを示す．その後，cactus 構造で一つの閉路のサイズが定数 b 以下であるようなネットワークにおけるブロードキャスト時間分布を求めるアルゴリズムを示す．アルゴリズムの計算時間は分布関数の引数 x の上界 w，ネットワークサイズ n，最大許容誤差 ϵ について多項式である．

Computing the Broadcast Time in Networks with Stochastic Transmission Time

Ei Ando †1 and Joseph Peters†2

In this paper, we consider a problem of computing distribution function of the broadcast time in networks where the transmission time of each communication link is given by a mutually independent random variables. Since computing the exact distribution function of similar problems are proved to be $\#P$-complete, we compute a polynomial $\hat{F}_B(x)$ that approximates the broadcast time distribution function $F_B(x)$ in $0 \le x \le w$ for a given real number $w \ge 0$. We first show an algorithm that computes $\hat{F}_B(x)$ in ring networks. We then show an algorithm that computes $\hat{F}_B(x)$ in cactus networks in which any cycle in it has no more than a constant b vertices. Under some natural assumptions for the transmission time, the running time of our algorithm is polynomial time with respect to the network size n, the upper bound w on the argument x of the distribution function, and the maximum acceptable error ϵ.

1. Introduction

We consider the broadcast time in a network whose topology is given by an undirected graph $G = (V, E)$ of order n with vertex set $V = \{v_1, \ldots, v_n\}$ and edge set E. Each vertex corresponds to a processor that sends and receives messages; the edges corresponds to communication links through which the processors send messages. The *transmission time* X_e of an edge $e \in E$ is the time in which a message is transmitted through a communication link that corresponds to e. The *broadcast time* X_B is the time in which one short message is transmitted from single processor to all processors in the network. We assume that the transmission time X_e is random for every $e \in E$; X_e's are mutually independent random variables. Thus, the broadcast time X_B is also a random variable. We are to compute the distribution function $F_B(x)$ of X_B.

In reality, it is reasonable to treat the transmission times between processors as random variables. In multi-commodity networks such as the Internet, the processors that forwards our messages also forwards the other people's messages. This gives some delay in the transmission. Since we do not know the behavior of the other people, we may model the transmission times as random variables.

Computing the exact distribution function $F_B(x)$ of the broadcast time X_B for general graph seems hard to solve. Consider a network whose topology is given by single-sourced directed acyclic graph (DAG) with unique source s where the broadcast is started from; the problem of computing the broadcast time can be transformed into computing the longest path length in a DAG. Computing the distribution of the longest path length in directed acyclic graphs (DAGs) is proved to be $\#P$-complete by Hagstrom[6]. According to Ball et al.[5], the problem of computing the exact distribution function of the stochastic longest path length is still NP-hard even for the series-parallel DAGs.

To overcome the difficulty, we approximately compute $F_B(x)$ by using Taylor polynomials. There are several results due to Ando et al. for approximately computing

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

Vol.2010-AL-131 No.3
2010/9/22

the distribution function of stochastic optimal solutions' weights[1)–4)]; in particular, the results in this paper is an application of the techniques in[3)] to approximately computing the stochastic broadcast time distribution function. We have three assumptions for the transmission time: We assume (1)that the transmission time is positive; (2)that the Taylor polynomial generated by the distribution function $F_e(x)$ of any transmission time X_e around $x = 0$ converges to $F_e(x)$ itself; (3)that the derivatives of $F_e(x)$ does not exceed 1 if $0 \leq x \leq w$. If we are to deal with a transmission time whose derivative of some order may exceed 1, we suggest that we find a value σ that satisfies, for any $p \geq 0$ and $0 \leq x \leq w$,

$$\left| \left(\frac{d}{dx} \right)^p F_e(x) \right| \leq \sigma^p. \tag{1}$$

Then we have $F'_e(x) = F_e(x/\sigma)$ instead of $F_e(x)$ and the derivatives of $F'_e(x)$ does not exceed 1.

This paper is organized as follows. In Section 2, we show an algorithm that computes a polynomial $\hat{F}_B(x)$ that approximates $F_B(x)$. In Section 3, we show how we can approximately compute $F_B(x)$ as a polynomial $\hat{F}_B(x)$ for cactus networks in which the maximum size of a cycle is at most a constant b. We conclude this paper in Section 4.

2. The Stochastic Broadcast Time in Ring Networks

2.1 Approximately Computing $F_B(x)$

We consider how we can compute the distribution function of the broadcast time in ring networks. After defining the necessary symbols, we show that $F_B(x)$ can be represented by a sum of repeated integrals (i.e., repetition of definite integrals). We then show how we can compute the repeated integral to obtain the approximate value of $F_B(x)$.

Here we define some symbols for the descriptions. We assume that $E = \{\{v_i, v_{i+1}\} \mid i = 1, \ldots, n-1\} \cup \{v_n, v_1\}$; let $e_i = \{v_i, v_{i+1}\} \in E$ for $i = 1, \ldots, n-1$ and $e_n = \{v_n, v_1\}$. We denote the transmission time of e_i by X_i. Without loss of generality, we also assume that the broadcast is always started by v_1.

Let us consider the distribution function $F_B(x)$ of the broadcast time X_B in the ring network. By assuming that the vertices $v_1, \ldots, v_n$ are ordered in clockwise way, We

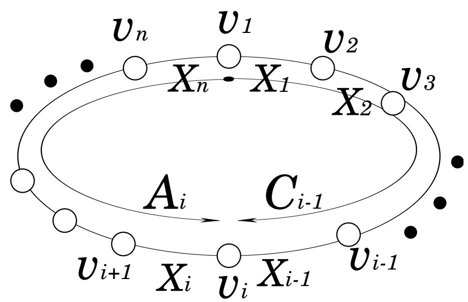

図 1 Definition of X_i, C_i and A_i.

have that $C_i = \sum_{1 \leq j \leq i} X_j$ (resp. $A_i = \sum_{i \leq k \leq n} X_n$) is the time when v_i receives the broadcast message that comes the clockwise way (resp. anticlockwise way). Figure 1 shows the definitions of X_i, C_i and A_i. Since the broadcast finishes when the last vertex receives a broadcast message, we have that

$$X_B = \max_{1 \leq i \leq n} \{\min\{C_{i-1}, A_i\}\}. \tag{2}$$

For computing $F_B(x)$, we are to aggregate all possible situations. We consider the cases where each of $X_1, \ldots, X_n$ takes an actual value $x_1, \ldots, x_n$, respectively. Also, we define $c_i = \sum_{1 \leq j \leq i} x_j$ and $a_i = \sum_{i \leq k \leq n} x_k$. To make the foresight clearer, we consider $2n$ mutually disjoint situations. Let $B_i(x)$ (resp. $D_i(x) \subset \mathbb{R}^n$) be set of situations in which v_i receives the broadcast messages before time x from the clockwise way (resp. from the anticlockwise way). Formally, $B_i(x)$ (resp. $D_i(x)$) is defined as a region of n-dimensional space $\mathbb{R}^n$ such that in case $(X_1, \ldots, X_n) = (x_1, \ldots, x_n)$ is in $B_i(x)$ (resp. $D_i(x)$), the last vertex is v_i and v_i receives the broadcast message from the clockwise way earlier (resp. from the anticlockwise way earlier). We have that

$$F_B(x) = \sum_{1 \leq i \leq n} \left(\underbrace{\int_{B_i(x)} \prod_{1 \leq j \leq n} f_j(x_j) dx_j}_{(A)} + \underbrace{\int_{D_i(x)} \prod_{1 \leq j \leq n} f_j(x_j) dx_j}_{(B)} \right), \tag{3}$$

where $f_i(x)$ is the probability density function of X_i. Since $B_i(x)$ and $D_i(x)$ are symmetric to each other, we concentrate on $B_i(x)$. We can formulate $B_i(x)$ as

$$B_i(x) = \{(x_1,\dots,x_n) \in \mathbb{R}^n \mid \quad c_{i-1} \le a_i \tag{4}$$

$$\wedge\, a_k \le c_{i-1} \quad \text{for} \quad k = i+1,\dots,n \tag{5}$$

$$\wedge\, c_{j-1} \le x \quad \text{for} \quad j = 1,\dots,i-1\}, \tag{6}$$

The inequality (4) is to ensure that the clockwise way message reaches v_i earlier. The inequalities (5) is to ensure that all vertices on the left side of v_i receive the broadcast message earlier than v_i. The inequalities (6) is to ensure that all vertices on the right side of v_i receive the broadcast message before v_i.

We are not going to compute the exact representation of $F_B(x)$; computing the exact representation of (A) is not easy because the description of the resulting form of an integral grows exponentially long with respect to the number of processed integrals.

To avoid the difficulty, we compute the Taylor polynomials of $f_i(x_i)$ for each $i = 1,\dots,n$. In executing the integrals of polynomials, we are not going to have exponentially long description. Let $\hat{f}_i^p(x_i)$ be the Taylor polynomial of order p.

In the following, we show how we compute the integral.

Step 1, Integrating w.r.t. x_i: We first focus on that x_i appears only in inequality (4). Since the only inequality that is concerned with x_i is

$$x_i \ge x_1 + \cdots + x_{i-1} - (x_{i+1} + \cdots + x_n) = c_{i-1} - a_i, \tag{7}$$

we compute this definite integral

$$\int_{c_{i-1}-a_i}^{\infty} \prod_{1\le j\le n} f_j(x_j)dx_j, \tag{8}$$

assuming $c_{i-1} - a_{i+1} \ge 0$. Since the condition $c_{i-1} - a_{i+1} \ge 0$ is identical with (5) of $k = i+1$, we do not have to consider the case where $c_{i-1} - a_i < 0$ as long as $B_i(x)$ is concerned. The resulting form of (8) is

$$\int_{B_i'(x)} (\lim_{a\to\infty} F_i(a) - F_i(c_{i-1} - a_i)) \prod_{1\le j\le i-1} f_j(x)dx_j \prod_{i+1\le k\le n} f_k(x)dx_k, \tag{9}$$

where $B_i'(x)$ is a set of $n-1$ dimensional points $(x_1,\dots,x_{i-1},x_{i+1},\dots,x_n) \in \mathbb{R}^{n-1}$ such that there exists an x_i satisfying $(x_1,\dots,x_n) \in B_i(x)$. Since $F_i(x)$ is a cumulative probability distribution function, we know that $\lim_{a\to\infty} F_i(a) = 1$; we approximate $F_i(c_{i-1} - a_i)$ by a Taylor polynomial.

Step 2, Integrating w.r.t. $x_{i+1},\dots,x_n$: We next focus on that x_{i+1} appears only in (5) of $k = i+1$. By (5) and the definition of x_{i+1}, we have

$$0 \le x_{i+1} \le x_1 + \cdots + x_{i-1} - (x_{i+2} + \cdots + x_n) = c_{i-1} - a_{i+2}. \tag{10}$$

We execute an integral in (9) with respect to x_{i+1}. Then we proceed to x_{i+2} since x_{i+2} is appears only in one of the remaining inequalities ((5) of $k = i+2$). This computation continues until we finish the integral with respect to $x_{i+1},\dots,x_n$. An important point for the later proof of the time complexity is that, in executing the integral with respect to x_k $(k = i+1,\dots,n)$, the description of the polynomial needs at most three variables in the integrand: x_k, c_{i-1} and a_{k+1}.

Step 3, Integrating w.r.t. $x_{i-1},\dots,x_1$: Finally, we have $i-1$ inequalities (6). Let us first focus on x_{i-1}. By (6), we have

$$0 \le x_{i-1} \le x - (x_1 + \cdots + x_{i-2}) = x - c_{i-2}, \tag{11}$$

since we assume that x_{i-1} is non-negative. By the similar argument in the previous step, we have inequalities $0 \le x_j \le c_{j-1}$ for $j = i-2, i-3,\dots,1$. We execute the integrals with respect to $x_{i-1}, x_{i-2},\dots,x_1$ in this order. The entire computation of computing $\int_{B_i(x)} \prod_{1\le j\le n} f_j(x_j)dx_j$ finishes when we finish executing the integral with respect to x_1. It is important that, in executing the integral with respect to x_l $(l = i-1,\dots,1)$, we need at most three variables here for the description of the integrand: x_l, c_{l-1} and x.

Since we can compute $\int_{D_i(x)} \prod_{1\le j\le n} f_j(x_j)dx_j$ in a symmetric way, it is clear that we can compute a polynomial that gives an approximation of $F_B(x)$. In the next, we show that the computation is polynomial time with respect to n, w and accepted maximum error ϵ.

2.2 Complexity Our Algorithm

Let us consider the complexity of our algorithm. We assume that we compute the Taylor polynomial of order p for approximating the density function $f_i(x)$. We first count the number of terms after executing the first integral. After that, we estimate how fast the number of terms would increase in each execution of the other integrals. We assume that the computation of each term finishes $O(1)$ time and thus the total number of terms that appears in the computation is the complexity of our algorithm. We have the following theorem for the complexity of our Algorithm.

Theorem1 The value of the distribution function $F_B(x)$ of the broadcast time in ring networks can be approximated within error ϵ in running time $O((w+n\ln nw/\epsilon)^3 n^5)$.

We first formulate the running time by using p, and then we show how large p is sufficient to reduce the error less than ϵ. Let us see the following lemma.

Lemma1 The running time of the algorithm in the previous section is $O(n^5 p^3)$. The proof is based on counting the number of terms that appear in the computation of the repeated integral.

Proof The proof is based on bounding the number of terms that appear in the computation of the repeated integral.

We first show that we have $O(p^3)$ terms in the resulting form after executing the integral with respect to x_i. Notice that we are *not* discussing about variables $x_1, \ldots, x_n$, but one specific variable x_i for integrating in a region $B_i(x)$. When we execute the integral with respect to x_i, we have (9); the number of its terms is one (the term $\lim_{a\to\infty} F_i(a) = 1$) plus the number of terms in the Taylor polynomial $\hat{F}_i(x)$ generated by $F_i(c_{i-1} - a_i)$. We replace a_i by $a_i = x_i + a_{i+1}$; we leave c_{i-1} as it is. After we replace x in $\hat{F}_i(x)$ by $c_{i-1} - (x_i + a_{i+1})$, we expand the representation into a sum of products (e.g., $Cx_i^p c_{i-1}^p a_{i+1}^p + C' x_i^{p-1} c_{i-1}^p a_{i+1}^p + \cdots$). Then, the terms in the resulting form can be represented by $Cx_i^\alpha c_{i-1}^\beta a_{i+1}^\gamma$, where C is a constant; α, β and γ are non-negative integers up to p; C, α, β and γ are defined for each terms. This leads to that we have at most $(p+1)^3$ terms in the representation of $\hat{F}_i(c_{i-1} - (x_i + a_{i+1}))$.

For considering the each step of executing the remaining integrals with respect to x_j $(j \neq i)$, we define some symbols. Let $s = (i, i+1, \ldots, n, 1, \ldots, 1)$ be a sequence of subscription numbers of variables x_i's; the order of s is the order the algorithm processes the variables $x_1, \ldots, x_n$. Let $s(j)$ be the j-th number in s; for example, $f_{s(1)}(x_{s(1)})$ is equivalent to $f_i(x_i)$. In processing the j-th integral with respect to $x_{s(j)}$, let the resulting form of the previous integral can be represented by

$$\int_{B_i(x)} I(x_{s(j)}, x_{s(j+1)}, \ldots, x_1) f_{s(j)}(x_{s(j)}) dx_{s(j)} \cdots f_1(x_1) dx_1, \qquad (12)$$

where $I(x_{s(j)}, \ldots, x_1)$ is a polynomial of the remaining variables.

We show that we have $O(j^3 p^3)$ terms in the execution of each integral with respect to $s(j)$. We expand $I(x_{s(j)}, \ldots, x_1) f_{s(j)}(x_{s(j)})$ into a sum of products as we did in executing the integral with respect to x_i. If $s(j)$ is earlier than $i-1$ in the order of s, any term in the resulting form can be represented by $Cx_{s(j)}^\alpha c_{i-1}^\beta a_{s(j)+1}^\gamma$, where the constants C, α, β and γ are defined for each terms; C is a constant real number; α, β and γ are non-negative integers up to $jp+j$. If $s(j)$ is later than $i-1$ in the order of s, each term in the resulting form can be represented by $Cx_{s(j)}^\alpha c_{s(j)-1}^\beta$, where C, α and β are similarly defined. By counting all possible terms, the number of the terms in the resulting form of each integral with respect to $s(j)$ is at most $(jp+j)^3$.

Now we have the total complexity of our algorithm. At this point, we have $O(j^3 p^3)$ terms in every step of executing n integrals. This leads to that, in total, $O(n^4 p^3)$ terms appear in the approximation of a repeated integral that is in $B_i(x)$, which proves this lemma. □

As for estimating how large p is sufficient to bound the approximation error less than a constant ϵ, we have the following lemma.

Lemma2 To have a polynomial $\hat{F}_B(x)$ that approximate $F_B(x)$ by running the algorithm in the previous section within error less than ϵ for $0 \le x \le w$, we have that $p = O(w + n \ln nw/\epsilon)$ is sufficient.

Proof In the proof we consider the upper bound on the error between $F_B(x)$ and the output $\hat{F}_B(x)$ of our algorithm. The idea of the proof is based on that the Taylor polynomial has $O(\ln 1/\epsilon)$ running time for approximating a function within error ϵ.

We first show that the entire error can be reduced quickly by increasing p. It is well known that the error of the Taylor polynomial $\hat{f}(x)$ of order p generated by $f(x)$ is bounded by

$$|f(x) - \hat{f}(x)| \le \frac{w^p}{(p+1)!} = \delta. \qquad (13)$$

(see e.g.,[7]) Let $\epsilon_j = f_j(x) - \hat{f}_j(x)$ for $j \neq i$ be the error that is created when approximating $f_j(x)$ by $\hat{f}_j(x)$. To have the approximation convergent, we approximate $1 - F_i(x)$ by $1 - \hat{F}_i(x)$ for $j = i$. Then the approximation is given by

$$\int_{B_i(x)} (1 - F_i(x) + \epsilon_i) \prod_{1 \le j \le i-1} (f_j(x_j) + \epsilon_j) dx_j \prod_{i+1 \le k \le n} (f_k(x_k) + \epsilon_k) dx_k. \qquad (14)$$

Vol.2010-AL-131 No.3
2010/9/22

By using δ the error is bounded by

$$\int_{B'_i(x)} (1 - F_i(x) + \delta) \prod_{1\le j\le i-1} (f_j(x_j) + \delta)dx_j \prod_{i+1\le k\le n} (f_k(x_k) + \delta)dx_k$$
$$- \int_{B'_i(x)} (1 - F_i(x)) \prod_{1\le j\le i-1} f_j(x_j)dx_j \prod_{i+1\le k\le n} f_k(x_k)dx_k. \qquad (15)$$

Since the term that does not have δ as a factor cancels in (15), we can have an upper bound on the error that has δ as a factor. By assuming that $0 < \delta < 1$, we can bound $\prod_{1\le j\le i-1}(f_j(x_j) + \delta)\prod_{i+1\le k\le n}(f_k(x_k) + \delta)$ by

$$\delta \prod_{1\le j\le i-1} (f_j(x_j) + 1) \prod_{i+1\le k\le n} (f_k(x_k) + 1) - \prod_{1\le j\le i-1} f_j(x_j) \prod_{i+1\le k\le n} f_k(x_k). \qquad (16)$$

Therefore, the error is bounded from above by

$$\int_{B'_i(x)} \delta(1 - F_i(x) + 1) \prod_{1\le j\le i-1} (f_j(x_j) + 1)dx_j \prod_{i+1\le k\le n} (f_k(x_k) + 1)dx_k. \qquad (17)$$

Now we remove the integrals from the upper bound to estimate the error of the entire computation. Remember that $F_i(x_i)$ and $f_j(x_j)$ is no more than 1 for all j. We have an upper bound on the error as

$$\int_{B'_i(x)} 2\delta \prod_{1\le j\le i-1} 2dx_j \prod_{i+1\le k\le n} 2dx_k, \qquad (18)$$

which is bounded from above by $\delta 2^n w^{n-1}$ because each of $x_1, \ldots, x_n$ but x_i is less than w.

Since we compute $B_i(x)$ and $D_i(x)$ for $i = 1, \ldots, n$, the error multiplies by $2n$. To have the total error $\delta 2^{n+1} w^{n-1} n$ less than ϵ, we have that $p = O(w + n \ln nw/\epsilon)$ is sufficient. □

3. The Stochastic Broadcast Time in Cactus Networks

3.1 Approximately Computing $F_B(x)$

In this section, we consider how we can approximately compute the distribution function of the broadcast time in a cactus network whose maximum number of in a cycle is a constant b. A cactus is a graph in which any two cycle share at most one vertex. Let $G = (V, E)$ be a cactus. We call the source v of the broadcast as the *root* of G. The algorithm computes from the leaf to the root. The first key of the algorithm is replacing each sub-cactus S by an edge with transmission time that is equal to the broadcast time of the original sub-cactus S. Then the sub-cactuses and a cycle can be treated as a ring network with pendant vertices (see Fig. 2). By applying the approximation for the ring with pendant vertices from the leaf to the root, we can approximate the broadcast time distribution function $F_B(x)$ of the entire cactus.

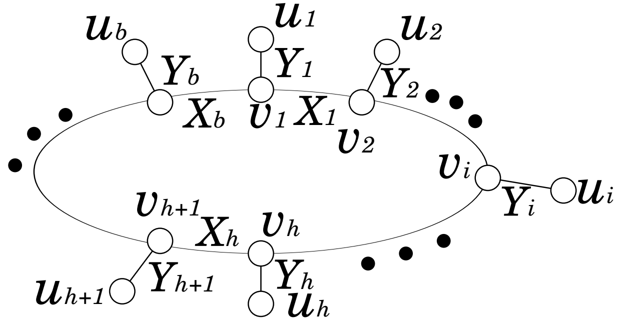

図 2 An example of a ring with pendant vertex.

We first compute the distribution function of the broadcast time in each leaf cycle L by the way we explained in the previous section; a *leaf cycle* is a cycle in a cactus that has no branching vertex (i.e., vertex whose degree is more than 2) except for the vertex that is the closest to the root in the cycle. Let the distribution function of the broadcast time in a leaf cycle be denoted by $F_L(x)$, which is approximated by a polynomial $\hat{F}_L(x)$ of degree p.

The essential point in the cactus graph is in a ring with pendant vertices. *A pendant vertex* here is a vertex that is connected to the other part of the graph by one edge. After computing $\hat{F}_L(x)$ for every leaf cycle L, we replace every leaf ring by a pendant vertex u_L that is connected by an edge e_L; the transmission time of e_L has distribution function $F_L(x)$, which is approximated by $\hat{F}_L(x)$ in the computation. Then the cycles that are one step closer to the root are considered to be ring networks with pendant vertices.

We prepare some notations here. Let $v_1, \ldots, v_b$ be the vertices in the ring and $u_1, \ldots, u_b$ be the pendant vertices.(We have $2b$ vertices here.) We have edges $e_1 = \{v_1, v_2\}, e_2 = \{v_2, v_3\}, \ldots, e_{b-1}\{v_{b-1}, v_b\}$ and $e_b = \{v_b, v_1\}$. In addition, we have edges

$e'_j = \{v_j, u_j\}$ that connects a pendant vertex to a ring vertex. As in the previous section, the (random) transmission time of an edge e_j is denoted by X_j, whose distribution function (resp. density function) is denoted by $F_j(x)$ (resp. $f_j(x)$). We assume that all the distribution function $G_j(x)$ of the broadcast time in each sub-cactus S_j rooted at a vertex v_j are approximately computed as $\hat{G}_j(x)$; we transform a cactus into a ring network with pendant vertices. The (random) transmission time of e'_j is denoted by Y_j; the distribution function (resp. density function) of Y_j is denoted by $G_j(x)$ (resp. $g_j(x)$).

In the following, we show how we compute the broadcast time in the cactus networks.

As we did in the previous section, we first show a representation of the distribution function $F_B(x)$ of the broadcast time. Assume that $X_1, \ldots, X_b$ and $Y_1, \ldots, Y_b$ take $x_1, \ldots, x_b$ and $y_1, \ldots, y_b$, respectively. We divide the space $\mathbb{R}^{2b}$ into $2b^2$ regions according to the following three items.

(1) v_h is the last vertex in the ring. $(h = 1, \ldots, b)$

(2) v_h first receives the broadcast message that came from either clockwise or anti-clockwise way.

(3) u_i is the last vertex of all. $(i = 1, \ldots, b)$

Note that u_i can be the last vertex even though v_i is not the last in the ring if Y_i is large enough. Here we denote by $B_{h,i}(x)$ the subset of $\mathbb{R}^{2b}$ where $(x_1, \ldots, x_n, y_1, \ldots, y_b) \in B_{h,i}(x)$ is the case that; the broadcast finishes before time x at the time u_i receives the broadcast message; v_i receives the broadcast message that came from the clockwise way earlier. We define $D_{h,i}(x)$ symmetrically; $(x_1, \ldots, x_b, y_1, \ldots, y_b) \in D_{h,i}(x)$ is the case that u_i is the last one of the broadcast, which finishes before time x; the broadcast message reaches v_i through the anti-clockwise way earlier. Now we have that $F_B(x)$ can be represented by

$$F_B(x) = \sum_{1 \le h \le b} \sum_{1 \le i \le b} \left(\mathcal{B}_{h,i}(x) + \mathcal{D}_{h,i}(x)\right), \tag{19}$$

where

$$\mathcal{B}_{h,i}(x) = \int_{B_{h,i}(x)} \prod_{1 \le j \le b} f_j(x_j) g_j(y_j) dx_j dy_j, \tag{20}$$

$$\mathcal{D}_{h,i}(x) = \int_{D_{h,i}(x)} \prod_{1 \le j \le b} f_j(x_j) g_j(y_j) dx_j dy_j. \tag{21}$$

For approximation, we use $\hat{g}_j(y_j)$ instead of $g_j(y_j)$. Since $\mathcal{B}_{h,i}(x)$ and $\mathcal{D}_{h,i}(x)$ are symmetric to each other, we concentrate on $\mathcal{B}_{h,i}(x)$. For $1 \le i \le h$ we formulate the region $B_{h,i}(x)$ as

$$B_{h,i}(x) = \{(x_1, \ldots, x_b, y_1, \ldots, y_b) \in \mathbb{R}^{2b} \mid$$

$$c_{h-1} \le a_h \tag{22}$$

$$\wedge\, a_{h+1} \le c_{h-1} \tag{23}$$

$$\wedge\, c_{j-1} + y_j \le c_{i-1} + y_i \quad \text{for} \quad j = 1, \ldots, h \tag{24}$$

$$\wedge\, a_k + y_k \le c_{i-1} + y_i \quad \text{for} \quad k = h+1, \ldots, b \tag{25}$$

$$\wedge\, c_{j-1} \le c_{i-1} + y_i \quad \text{for} \quad j = 1, \ldots, b \tag{26}$$

$$\wedge\, a_k \le c_{i-1} + y_i \quad \text{for} \quad k = h+1, \ldots, b \tag{27}$$

$$\wedge\, c_{i-1} + y_i \le x\}, \tag{28}$$

where $c_{i-1} = x_1 + \cdots + x_{i-1}$ and $a_i = x_i + \cdots + x_b$ as in the previous section. The inequality (22) is for the condition that v_h receives the broadcast message that comes from the clockwise way; (23) is to ensure that v_h is the last vertex in the ring. By (24,25, 26,27), we have that u_i is the last vertex in the entire network. The last inequality (28) is for the condition that the broadcast time is less than x. Since the case $h+1 \le i \le n$ is symmetric, we concentrate on the case $1 \le i \le h$.

Step 1, Integrating w.r.t. x_h: Firstly, we focus on that x_h is present only in (22). We have

$$c_{h-1} - a_{h+1} \le x_h. \tag{29}$$

The left-hand side of the inequality (29) is always positive as long as (23) is satisfied. Since, in the integrand, x_h is present only in a factor $f_h(x_h)$, we have $(1 - F_h(c_{h-1} - a_{h+1}))$ as a corresponding factor of the resulting form and the other factors of the integrand remains the same in the resulting form. We approximate $(1 - F_h(c_{h-1} - a_{h+1}))$ by a $2b$-variable Taylor polynomial of degree p. To make the analysis simpler, we divide the Taylor polynomial by

$$1+\underbrace{\frac{(2bw)^{p+1}}{(p+1)!}}_{(A)}. \tag{30}$$

The value of (A) in (30) is an upper bound on the error that is created by the Taylor polynomial; the value of (A) can be bounded by a constant when $p = O(w)$. Dividing the integrand by (30) assures that the approximation of $(1 - F_h(c_{h-1} - a_{h+1})$ is less than 1 if $x \le w$.

Step 2, Integrating w.r.t. $y_1, \ldots, y_{i-1}, y_{i+1}, \ldots, y_h$: Secondly, we focus on that y_j for $j = 1, \ldots, h$ appears only in (24). By (24) and our assumption that every transmission time is positive, we have

$$0 \le y_j \le c_{i-1} + y_i - c_{j-1}. \tag{31}$$

Since we assume that all the transmission time is positive, the rightmost part is always positive. We execute the integrals with respect to y_j for $j = 1, \ldots, h-1$ at this step. Since y_j appears only in a factor $\hat{g}_j(y_j)$, the corresponding factor in the resulting form is $\hat{G}_j(c_{i-1} + y_i - c_{j-1})$; the other factors in the integrand remains the same in the resulting form.

Step 3, Integrating w.r.t. $y_{h+1}, \ldots, y_n$: Next, we focus on that y_k for $k = h+1, \ldots, b$ appears only in (25). By the similar argument in the previous step, we have

$$0 \le y_k \le c_{i-1} + y_i - a_k. \tag{32}$$

The integrals with respect to y_k are non-zero only if $c_{i-1} + y_i - a_k \ge 0$, which always holds as long as (27)is satisfied. We can execute the integral with respect to y_k for $k = i+1, \ldots, b$. Since $\hat{g}_k(x_k)$ is the only y_k-dependent factor in the integrand, we have $\hat{G}_k(c_{i-1} + y_i - a_k)$ as the corresponding factor in the resulting form. The resulting form after this step is

$$\int_{B'_{h,i}(x)} (1 - \hat{F}_h(c_{h-1} - a_{h+1}))\hat{g}_i(y_i) \prod_{j=1,\ldots,i-1,i+1,\ldots,h} \hat{G}_j(c_{i-1} + y_i - c_{j-1}) \prod_{k=h+1,\ldots,n} \hat{G}_k(c_{i-1} + y_i - a_k) \prod_{l=1,\ldots,h-1,h+1,\ldots,n} F_l(x_l) dx_l dy_i, \tag{33}$$

where $B'_{h,i}(x)$ is a set of b-dimensional points $q = (y_i, x_1, \ldots, x_{h-1}, x_{h+1}, \ldots, x_b) \in \mathbb{R}^b$ such that if $q \in B'_{h,i}(x)$ then $(x_1, \ldots, x_b, y_1, \ldots, y_b) \in B_{h,i}(x)$ as long as $y_1, \ldots, y_{i-1}, y_{i+1}, \ldots, y_b$ and x_h satisfies the inequalities (29, 31, 32).

Step 4, Integrating w.r.t. y_i: We proceed to the integral with respect to y_i. By (26) with $j = h$ and (27) with $k = h+1$, we have that

$$\max\{a_{h+1}, c_{h-1}\} \le c_{i-1} + y_i, \tag{34}$$

whose left-hand side is always equal to c_{h-1} as long as (23) holds. Together with (28), we have that

$$c_{h-1} - c_{i-1} \le y_i \le x - c_{i-1}. \tag{35}$$

To have this interval of y_i non-empty, we must have

$$c_{h-1} \le x, \tag{36}$$

which is always satisfied as long as 28 holds. Before executing the integral with respect to y_i, we expand the integrand into a sum of products. We compute the polynomial of y_i. The number of y_i-dependent factors is b: we have $\hat{G}_j(c_{i-1} + y_i - c_{j-1})$ for $j = 1, \ldots, h$, $\hat{G}_k(c_{i-1} + y_i - a_k)$ for $k = h+1, \ldots, b$ and $\hat{g}(y_i)$.

Step 5, Integrating w.r.t. $x_{h+1}, \ldots, x_n$: Since (23) is the only inequality left that consists of x_{h+1}, we execute the integral with respect to x_{h+1} in the interval

$$0 \le x_{h+1} \le c_{h-1} - a_{h+2}. \tag{37}$$

By the condition that the rightmost part of (37) is positive, we have that

$$0 \le c_{h-1} - a_{h+2}, \tag{38}$$

which gives the interval of integrating with respect to x_{h+2}; this sequence of inequalities appears until we finish integrating with respect to $x_{h+2}, \ldots, x_b$. Before executing the integral with respect to x_l $(l = h+1, \ldots, b)$, we approximate $f_l(x_l)$ by a Taylor polynomial of degree p generated at $x_l = 0$. For the simplicity of the analysis, we divide the integrand by (30). Then we expand the x_l-dependent factors into a sum of products before we execute the integrand with respect to x_l.

Step 6, Integrating w.r.t. $x_{h-1}, \ldots, x_1$: Finally by the only remaining inequality (36), we have

$$0 \le x_{h-1} \le x - c_{h-2}. \tag{39}$$

After executing the integral with respect to x_{h-1}, we have, as the condition that the rightmost part of (39) is non-negative,

$$0 \le x_{h-2} \le x - c_{h-3}. \tag{40}$$

Similarly for x_l $(l = h-2, h-3, \ldots, 1)$, we have $0 \le x_l \le x - c_{l-1}$. We can execute the integral with respect to $x_{h-2}, x_{h-3}, \ldots, x_1$ by the same way. At this point, we finish all integration with respect to $x_1, \ldots, x_b, y_1, \ldots, y_b$ and the resulting form gives the approximation of $F_B(x)$. In executing the integral with respect to x_l ($l = i, i-1, \ldots, 2, 1$). We first expand the x_l-dependent factors into a sum of products. In these steps, we use x_l, c_{l-1}, x and no other variables for describing the factor. Then we expand the x_l-dependent factors into a sum of products; we also divide the integrand by (30) before we execute the integrals with respect to x_l's.

3.2 Complexity of Our Algorithm

We prove the complexity of our algorithm here. As in the previous section, we first formulate the running time of our algorithm using p. Then we show how large p is sufficient to bound the error less than a constant ϵ in $0 \leq x \leq w$. We can prove the following theorem.

Theorem2 The time complexity of computing the broadcast time distribution function $F_B(x)$ of cactus networks within error ϵ is polynomial with respect to the network size, the upper bound w on x and ϵ if the maximum size of a cycle is bounded by a constant b.

We prove this theorem by the similar way as we proved the time complexity of computing the broadcast time distribution function of ring networks.

We first formulate the running time of our algorithm by using p as a parameter.

Lemma3 The running time of the approximating $F_B(x)$ in a cactus network with cycles whose size are most b is $O(2n^2p(p+1)^{2b+1})$

The following lemma shows how large p is sufficient to bound the error of the approximation less than ϵ in $0 \leq x \leq w$.

Lemma4 To bound the error of the polynomial that is computed by the algorithm in the previous section, we have that $p = O(b \ln w + w + n \ln b + \ln 1/\epsilon)$ is sufficient.

We omit the proof of the lemmas due to the space limit.

4. Conclusion

In this paper, we showed that we can, in polynomial time with respect to w, n, and ϵ, compute the distribution function of the stochastic broadcast time in ring networks and cactus networks; so far, the size of a cycle in a cactus network needs to be bounded by a constant b, whereas ring networks does not have such constraint.

The difference between ring networks and cactus networks is in the cost of expanding the integrand in each step of executing the integral. In ring networks, we could assume that there are at most three variables in executing an integral with respect to x_j: x_j, a_{j+1} and c_{i-1} for $j = i+1, \ldots, n$, or x_j, c_{j-1} and x for $j = i-1, i-2, \ldots, 1$. On contrast, we have as many as $2b$ symbols in ring networks. If we have any way for coping with this number of symbols or for reducing the number of symbols that appears at a time, we may be able to compute the $F_B(x)$ for general cactus networks.

For future work, there are two ways to go. One way is to reduce the time complexity. There are good chances for reducing the time complexity for approximately computing $F_B(x)$. The other way is to challenge the other kinds of networks. We next intend to analyze the chorded ring networks and more complex networks.

参 考 文 献

1) E. Ando, T. Nakata, M. Yamashita, Approximating the longest path length of a stochastic DAG by a normal distribution in linear time, *Journal of Discrete Algorithms*, 7 pp. 420–438 (2009)
2) E. Ando, H. Ono, K. Sadakane, M. Yamashita, A Counting-Based Approximation the Distribution Function of the Longest Path Length in Directed Acyclic Graphs, *Proceedings of the 7th Forum on Information Technology (FIT2008)*, pp. 7–8 (2008)
3) E. Ando, H. Ono, K. Sadakane, M. Yamashita, Computing the Exact Distribution Function of the Stochastic Longest Path Length in a DAG, *Proceedings of the 6th conference on Theory and Application of the Models of Computation (TAMC2009)*, LNCS, 5532, pp. 98–107 (2009)
4) E. Ando, H. Ono, K. Sadakane, M. Yamashita, A Generic Algorithm for Approximately Solving Stochastic Graph Optimization Problems, *Proceedinsg of the 5th Symposium on Stochastic Algorithms, Foundations and Applications (SAGA 2009)*, LNCS, 5792, pp. 89–103, 2009
5) M. O. Ball, C. J. Colbourn, and J. S. Proban, Network Reliability, Handbooks in Operations Research and Management Science, Vol 7: Network Models, M. O. Ball, T. L. Magnanti, C. L. Monma, and G. L. Nemhauser (eds.), Elsevier Science B. V., pp. 673–762 (1995)
6) J. N. Hagstrom, Computational Complexity of PERT Problems, *NETWORKS*, Vol. 18, pp. 139–147 (1988)
7) G. B. Thomas, Jr., Thomas' Calculus International Edition, Pearson Education, pp. 965–1066 (2005)