3 B－4

A Method to Calculate Legal Sequence Number for Extended Series-Parallel Digraphs

Naomi Yoshioka and Qi-Wei Ge
Faculty of Education, Yamaguchi University, Japan

1. Introduction

The problem of topological sorting is, given with a directed acyclic graph $G = (V, E)$, to find a total ordering of the vertices such that if $(u, v) \in E$ then u is ordered before v [1]. Instead of finding total orderings, we wish to find out how many total orderings exist in a given directed acyclic graph $G = (V, E)$. We call a total ordering of the vertices as legal sequence and the problem as *LSN* (*legal sequence number*) problem. In this paper we study how to calculate *LSN* for a given directed acyclic graph $G = (V, E)$. First we give the fundamental theorems for calculating *LSN* of general directed acyclic graphs. Then we give a way to calculate *LSN* for series-parallel digraphs [2][3] and methods to transform a class of extended series-parallel digraphs into series-parallel ones. Finally we calculate *LSN* for the class of graphs, which are extended from series-parallel digraphs by adding so called bridge edges and bridge-path between specific paths.

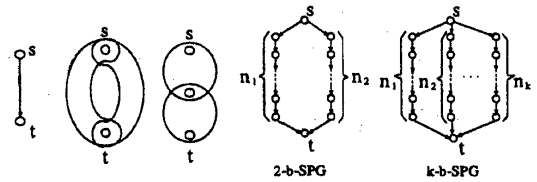
2. Definitions

Let $G = (V, E)$ be a directed acyclic graph denoted as *DAG*. G is called two terminal *DAG* denoted as *st-DAG*, if there exist a single source s and a single sink t in G. An *st-DAG* $G = (V, E)$ is called series-parallel graph (or digraph) denoted as *SPG*, if G can be generated by iteratively applying the following compositions [2]:

· A single edge is series-parallel graph;

· A graph G composed of two *SPG*s in parallel by identifying the sources and the sinks respectively is series-parallel graph as shown in Fig.1 (1) (parallel composition);

· A graph G composed of two *SPG*s in series by identifying the sink of one graph with the source of the another as shown in Fig.1 (1) (series composition).



(1) Composition of SPGs (2) Basic SPGs

Fig.1 : Compositions of SPGs and examples of basic SPGs.

An *st-DAG* $G = (V, E)$ is called basic series-parallel graph denoted as *b-SPG*, if each vertex $v \in V - \{s, t\}$ possesses exact one input and one output edges. A *b-SPG*, G with λ paths from the source to the sink, is denoted as *λ-b-SPG*. Fig.1 (2) shows a *2-b-SPG* and a *k-b-SPG*.

Let $G = (V, E)$ be a *DAG*. For vertices, $u, v \in V$, if there is a directed path (path for short) from u to v, then it is denoted as $u \prec v$ or $v \succ u$. A vertex sequence $\sigma = v_1 v_2 ... v_l$ is called ordered sequence if $v_j \prec v_i$ does not hold for $i < j$. An ordered sequence $\gamma = v_1 v_2 ... v_n$ is called legal sequence if all vertices of G are included in γ. We denote the set of the total legal sequences $\{\gamma_1, \gamma_2, ...\}$ as $\Gamma(G)$ and the legal sequence number $|\Gamma(G)|$ as $LSN(G)$.

3. Fundamental Theorems for *LSN*

In this section, we propose basic fundamental theorems for calculating *LSN* of general directed acyclic graphs. We restrict graphs to *st-DAG*s, otherwise we need only to add two vertices, s and t, by drawing edges from s to the sources and from the sinks to t.

Let $G = (V, E)$ be a *DAG*. An edge $(u, v) \in E$ of G is called redundant if there exists a path from u to v which avoids (u, v). G is minimum if it consists of no redundant edges.

[Theorem 1] Let $G = (V, E)$ be a directed graph. (1) If G contains a directed circuit, $\Gamma(G) = \phi$ and $LSN(G) = 0$ hold; (2) If G_m is minimum graph of G, $\Gamma(G) = \Gamma(G_m)$ and $LSN(G) = LSN(G_m)$ hold. □

Theorem 1 tells us that a graph is equivalent to its minimum one with respect to *LSN*. The following theorem shows an equivalent transformation between graphs for calculating *LSN*.

[Theorem 2] Let μ_1 and μ_2 be any two vertices of *st-DAG* $G = (V, E)$, G_1 and G_2 be two directed graphs generated from G as $G_1 = (V, E \cup \{(\mu_1, \mu_2)\})$ and $G_2 = (V, E \cup \{(\mu_2, \mu_1)\})$ as shown in Fig.2. Then $\Gamma(G) = \Gamma(G_1) \cup \Gamma(G_2)$ and $\Gamma(G_1) \cap \Gamma(G_2) = \phi$ hold. Further $LSN(G) = LSN(G_1) + LSN(G_2)$ holds. □

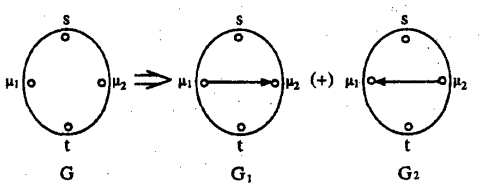


Fig.2 : Illustrations for Theorem 2

4 . Calculation of LSN for extended SPGs

In this section, we are to use the fundamental theorems to calculate LSN of SPGs and extended series-parallel graphs. At first two theorems are given to show how to obtain LSN for SPGs.

[Theorem 3] Let $G = (V, E)$ be λ-b-SPG and $n_1, n_2, ..., n_\lambda$ be the numbers of the vertices included in the λ paths, except the source and the sink. Then

$$LSN(G) = \frac{(n_1 + n_2 + ... + n_\lambda)!}{n_1! n_2! ... n_\lambda!} = C(n_1, n_2, ..., n_\lambda).$$

□

[Theorem 4] Let $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ be two st-DAGs. If G_s is an SPG composed of G_1 and G_2 in series, then $LSN(G_s) = LSN(G_1) \times LSN(G_2)$. If G_p is an SPG composed of G_1 and G_2 in parallel, then $LSN(G_p) = (LSN(G_1) \times LSN(G_2)) \times C(|V_1| - 2, |V_2| - 2)$. □

Let us see a so called 1-bridge added 2-b-SPG, G as shown in Fig.3, which is constructed by adding a single bridge edge (bridge for short) (u, v) between the two paths of 2-b-SPG. We can calculate LSN of the graph by doing the following operation.

⟨Operation⟩

Repeat the following (i) and (ii) until all the transformed graphs become SPGs (see Fig.3):

(i) Choose two proper vertices, x and y, respectively from the two paths and add bridges, (x, y) and (y, x), individually to get two new graphs;

(ii) Delete redundant edges from the new graphs. □

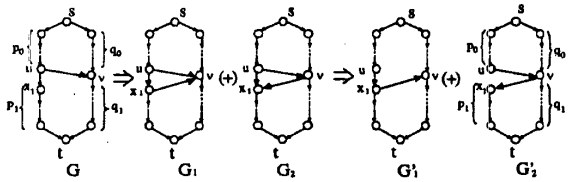
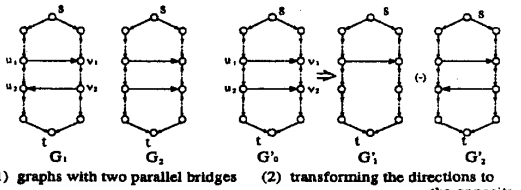


Fig.3 : A 1-bridge added 2-b-SPG and its transformed graphs.

The result of G of Fig.3 is as follows:

$$LSN(G) = \sum_{i=0}^{p_1} (C(p_0 + 1 + i, q_0) \times C(p_1 - i, q_1)).$$

For a 2-bridge added 2-b-SPG that is constructed by adding two crossed edges between the two paths, the problem can result in the one of 1-bridge added 2-b-SPG by applying Theorem 1 and 2. For 2-bridge added 2-b-SPGs with parallel bridges as shown in Fig.4 (1), the problem is not so easy. We can restrict the 2-bridge added 2-b-SPGs to ones whose two bridges are with opposite directions, otherwise we need only to do the transformation like Fig.4 (2). For such a graph G, we can calculate LSN by doing the operation.



(1) graphs with two parallel bridges (2) transforming the directions to the opposite

Fig.4 : Extended 2-b-SPGs with two parallel bridges

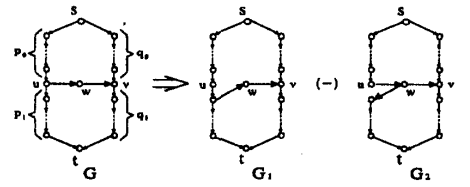


Fig.5 : A bridge path added 2-b-SPG and its transformed graphs.

Based on the same idea, we can calculate LSN for n-bridge added 2-b-SPG, whose bridges are whatever crossed or parallel. These results are easily applicable to such a class of graphs that are composed of n-bridge added 2-b-SPGs in series or in parallel as stated in Theorem 2.

Further we can calculate LSN of graphs extended from SPGs by adding a bridge-path (with two edges) between specific paths as shown in Fig.5. Such graphs can be eventually transformed into 1-bridge added 2-b-SPGs by doing the operation. Then we have

$$LSN(G) = \sum_{i=0}^{p_1} \sum_{j=0}^{p_1 - i} (C(p_0 + 2 + i + j, q_0) \times C(p_1 - i - j, q_1)).$$

Similarly bridge-path (with n edges) added 2-b-SPGs can be transformed into 1-bridge added 2-b-SPGs. So finally we can obtain LSN for such extended SPGs.

5 . Conclusions

We have proposed in this paper fundamental theorems on calculating LSN for st-DAGs, and given methods to obtain LSN for SPGs as well as a class of extended SPGs that are composed of n-bridge or bridge-path added 2-b-SPGs in series or in parallel. Our final target is to find out a method to calculate LSN for a general DAG.

References

[1] D. Knuth: the Art of Computer Programming, 1, Addison-Wesley, Reading, MA, 1968.

[2] W. Bein, J. Kamburowski and M. Stallmann: "Optimal reduction of two-terminal directed acyclic graphs", SIAM J. Comput., 21, pp. 1112-1129, 1992.

[3] J. Valdes, R.E. Tarjan and E.L. Lawler: "The recognition of series parallel digraphs", SIAM J. Comput.,11, pp. 298-313, 1982.