SQL ALGORITHM FOR SOLVING MARKOV MODELS BY GRAPH METHOD

A simple graph algorithm for finding stabilized probabilities of the finite Markov models implemented in SQL is presented. The algorithm generates systematically all oriented spanning trees of a transition graph. The method is demonstrated on the computation of probabilities in the MMPP/M/1/K queue.

Keywords: Markov models, queue, graph algorithm, SQL algorithm

1. Introduction

The original graph study of steady-state distribution for stable Markov process with finite state sets is presented by Markl [1]. Stabilized probabilities \( \pi \) of the stable Markov process associated with a weighted digraph (called transition graph) \( G = (S, E, c) \) are given by formulas

\[
\pi_i = \frac{B_i}{\sum_{j \in S} B_j} \quad i \in S, \tag{1}
\]

where \( B_i, i \in S \) is the sum of weights of all the spanning trees of \( G \) that have their roots in vertex \( i \).

For processes with many states and many possible transitions between them it is not easy to find all spanning trees. We apply a simple algorithm that systematically generates all spanning trees with given roots in SQL.

2. Basic definitions

We start with giving the some definitions from graph theory that will be used in this paper. A (simple) digraph \( G = (V, E) \) consists of a finite set \( V \) of vertices and set \( E \) of edges – ordered pairs of distinct vertices; that is, each edge \((u, v)\) is directed from tail \( u \) to head \( v \). A (directed) \( u \to v \) path from vertex \( u \) to vertex \( v \) is such a sequence \( u = v_1, (v_1, v_2), v_2, ..., (v_{k-1}, v_k), v_k = v \) that \( v_i \in V \) for \( i = 0, ..., k \) and \( v_i \neq v_j \) for \( 0 < i < j < k \), and that \( (v_{i-1}, v_i) \in E \) for \( i = 1, ..., k \). If \( u = v \) then \( u \to v \) path is called a cycle. A digraph in which each pair of vertices lies on a common cycle is called strongly connected. A digraph in which for each pair \( \{u, v\} \subseteq V \) there exists \( u \to v \) path or \( v \to u \) path is called connected. The component of a digraph is maximal connected subgraph of a digraph.

A digraph that has no cycle is called a directed acyclic graph (DAG). A rooted tree is a DAG in which one vertex, the root, is distinguished and in which all edges are implicitly directed to the root. (Note that in the standard definition [7] of the rooted tree the orientation of the edges is away from the root.) The weight of the (edge) weighted rooted tree \( T = (V, E_T, w) \) we mean number

\[
w(T) = \prod_{h \in E_T} w(h) \tag{2}
\]

A spanning rooted tree \( (T, r) \) with root \( r \), shortly spanning \( r \)-tree, of a strongly connected digraph \( G \) is a rooted tree \( T \) that is a subgraph of \( G \) and that contains every vertex of \( G \). The weight of the a spanning \( r \)-tree \( (T, r) \) of a (edge) weighted digraph \( G = (V, E, w) \) is weight \( w(T) \) of the rooted tree \( T = (V, E_T) \) given by (2).

3. SQL algorithm

The strongly connected digraph \( G = (V, E) \) is given. We may assume without loss of generality that \( V = \{1, 2, ..., n\} \) and root \( r = 1 \). The cases where \( r \neq 1 \) can be transformed to the case \( r = 1 \) by renumbering of vertices \( V \). Let the set \( E \) be represented by a table \textit{Edge} with two columns:

- \textit{Edge.u} is head of a directed edge \((u, v)\)
- \textit{Edge.v} is tail of a directed edge \((u, v)\)

and a table \textit{Solution} with two columns of \textit{array}\{1, ..., n\}:

- \textit{Solution.tree} is the subtree – subgraph of spanning 1-tree
- \textit{Solution.comp} is the components of tree:\textit{Solution}

In Figure 1 we have the example of the spanning 1-tree \( T = (V, H_1) \) where \( V = \{1, 2, 3, 4, 5\} \) and \( H_1 = \{(2, 4), (3, 5), (4, 1), (5, 4)\} \) are represented by array \textit{tree} = \{0, 4, 5, 1 4\}. In Figure 2

\[\text{Fig. 1 The spanning 1-tree as array [0, 4, 5, 1 4]}\]
we have the example of the subtree of the spanning 1-tree which is represented by array \( \text{tree} = [0, 4, 0, 1, 4] \) and has two components \( T_1 = \{1, 2, 4, 5\}, \{(2, 4), (4, 1), (5, 4)\} \) and \( T_2 = \{3\} \) which are represented by array \( \text{comp} = [1, 1, 3, 1, 1] \).

\[ \text{Fig. 2 The subtree of the 1-tree with two components} \]

We can now describe the algorithm which generates all spanning 1-trees:

\[ \text{procedure } \text{SQL1Trees} \ (\text{Edge}) \]

\[ \text{for } i = 1 \text{ to } n \text{ do } \ (* \text{ Initialization } *) \]

\[ \text{Solution}.\text{tree}[i] = 0, \text{Solution}.\text{comp}[i] = i \]

\[ \text{for } k = 1 \text{ to } n-1 \text{ do } \]

\[ \text{SELECT DISTINCT } \ (* \text{ Subgraphs of spanning } 1\text{-trees } * ) \]

\[ \text{SetValue} (\text{Solution}.\text{tree}, \text{Edge}.u, \text{Edge}.v) \text{ AS tree} \]

\[ \text{SetValue} (\text{Solution}.\text{comp}, \text{Edge}.u, 1) \text{ AS comp} \]

\[ \text{FROM} \ \text{Solution, Edge} \]

\[ \text{WHERE} \]

\[ \text{Solution}.\text{comp}[\text{Edge}.u] \neq 1 \ \text{AND} \ \text{Solution}.\text{comp}[\text{Edge}.v] = 1 \]

\[ \text{GROUP BY tree, comp} \]

\[ \text{TO FILE} \ \text{Solution} \]

\[ \text{Edges of the transition graph } G \]  

\begin{tabular}{|l|l|l|l|l|l|l|l|l|l|}
\hline
u & 1 & 2 & 3 & 3 & 3 & 4 & 4 & 5 & 5 & 6 & 6 \\
\hline
v & 2 & 3 & 1 & 1 & 4 & 5 & 2 & 3 & 6 & 3 & 6 & 4 & 5 \\
\hline
\end{tabular}

Before we start the main SQL-algorithm, we calculate the values of null subtree \((V, \emptyset)\) with \( n \) components. One new edge \((u, v)\) appends in the \( k \)-step, if it is possible else subtree is deleted. So the subtrees \((V, H_k)\) where \(|H_k| = k\) are generated step by step. After the last step we have all spanning 1-trees.

A new subtree and its components are stored in an array \text{tree} and \text{comp} which are updated by the function:

\[ \text{function } x = \text{SetValue}(x, i, j) \]

\[ x[i] = j \]

Note that after the last step are \text{comp} = [1, 1, ..., 1] because the spanning 1-trees are connected digraphs.

4. Example of \( \text{MMPP}_2/M/1/K \) queue

We consider a queueing system studied by Peško [2] with two-state Markov modulated Poisson process \( \text{MMPP}_2 \), a single exponential distributed server and a finite waiting room. The problem of switch design and admission control in a high speed network is modeled in [6] as \( \text{MMPP}_2/(GI)/1/K \) queue. The transition graph for the \( \text{MMPP}_2/M/1/K \) queue is a weighted digraph

\[ \text{Fig. 3 Transition graph } G \text{ of } \text{MMPP}_2/M/1/3 \text{ queue} \]

\( G = (S, E, c) \). In Figure 3 we have the small example of the \( \text{MMPP}_2/M/1/3 \) queue with maximal 2 customers in waiting room where \( \alpha \) and \( \beta \) are ON and OFF rate of source, \( \lambda \) is the arrival rate from ON source and \( \mu \) the service rate.

We have a table \text{Edge} in the Table 1. The costs of the edges of the transition graph are omitted. Note that the table \text{Edge} is shown horizontally instead of a usual vertical layout. The initialization table \text{Solution} is in the first row of the Table 2. After \( k \)-th step \((k = 1, 2, 3, 4)\) we have a new \text{Solution} in Table 2 with added column of step marked \( k \). All spanning 1-trees are generated in the last \text{Solution} table 3 and described in figure 4. And so \( B_1 \) - the sum of weights of all the spanning 1-trees is

\[ B_1 = \alpha \mu^4 + 2 \alpha^2 \mu^3 + \alpha^3 \mu^2 + 2 \alpha \beta \mu^3 + 2 \alpha \beta \mu^2 + \alpha \beta \mu^2 \cdot \alpha \lambda \mu^2 \cdot \beta \]

The sum of weights of all the spanning \( r \)-trees \( B_r \) for \( r = 2, ..., 6 \) can be computed by renumbering set of vertices \( S \).
Solution after $k^{th}$ step

\begin{tabular}{|c|c|c|}
\hline
\text{tree} & \text{comp} & \text{k} \\
\hline
[0, 0, 0, 0, 0, 0] & [1, 2, 3, 4, 5, 6] & - \\
[0, 1, 0, 0, 0, 0] & [1, 1, 3, 4, 5, 6] & 1 \\
[0, 0, 1, 0, 0, 0] & [1, 2, 1, 4, 5, 6] & 1 \\
[0, 1, 1, 0, 0, 0] & [1, 1, 1, 4, 3, 6] & 2 \\
[0, 1, 0, 1, 0, 0] & [1, 1, 1, 3, 5, 6] & 2 \\
[0, 1, 0, 0, 1, 0] & [1, 2, 1, 1, 5, 6] & 2 \\
[0, 0, 1, 1, 0, 0] & [1, 2, 1, 4, 1, 6] & 2 \\
[0, 1, 1, 1, 0, 0] & [1, 1, 1, 1, 5, 6] & 3 \\
[0, 1, 1, 0, 1, 0] & [1, 1, 1, 3, 5] & 3 \\
[0, 1, 0, 1, 1, 0] & [1, 2, 1, 1, 5] & 3 \\
[0, 1, 0, 0, 1, 1] & [1, 2, 1, 1, 4] & 3 \\
[0, 0, 1, 1, 1, 0] & [1, 1, 1, 1, 4] & 4 \\
[0, 1, 1, 1, 1, 0] & [1, 1, 1, 1, 3] & 4 \\
[0, 1, 1, 0, 1, 1] & [1, 1, 1, 1, 2] & 4 \\
[0, 1, 0, 1, 1, 1] & [1, 1, 1, 1, 1] & 4 \\
[0, 1, 0, 1, 0, 1] & [1, 1, 1, 1, 1] & 4 \\
\hline
\end{tabular}

Solution after last 5$^{th}$ step

\begin{tabular}{|c|c|c|}
\hline
\text{tree} & \text{comp} & \text{k} \\
\hline
[0, 1, 1, 2, 3, 4] & [1, 1, 1, 1, 1, 1] & - \\
[0, 0, 1, 2, 3, 4] & [1, 1, 1, 1, 1, 1] & 1 \\
[0, 0, 1, 1, 2, 3, 4] & [1, 1, 1, 1, 1, 1] & 1 \\
[0, 0, 1, 1, 1, 2, 3, 4] & [1, 1, 1, 1, 1, 1] & 1 \\
[0, 0, 1, 1, 1, 1, 2, 3, 4] & [1, 1, 1, 1, 1, 1] & 1 \\
\hline
\end{tabular}

The research of the author is supported by the Slovak Scientific Grant Agency under grant No. 1/0490/03.

References:

[1] MARKL, J.: \textit{A Graph Method for Markov Models Solving}, Acta Mathematica et Informatica Universitatis Ostraviensis, Vol. 1, 1993, pp. 75-82

[2] PEŠKO, Š.: \textit{Erlang ON/OFF Moduled Queueing Systems}, Proceedings of the 20$^{th}$ International Conference, Mathematical Methods in Economics 2002, Ostrava, ISBN 80-248-0153-1, 2002, pp. 209-213

[3] ROSS, S. M.: \textit{Stochastic Processes}, John Wiley & Sons, Inc., 1983, ISBN 0-471-09942-2

[4] KAUKIČ, M.: \textit{Open Source Software in Mathematical Education}, 1. International conference, Aplimat, Bratislava 2002, pp. 233-238

[5] REBO, J., BARTL, O.: \textit{Condition of Stability for Tandem Queues with Blocking and Exponential and Erlangian Service Time Distribution}, Studies of the Faculty of Management Science and Informatics, Vol. 8, Žilina, 1999, pp. 67-74

[6] JEAN-MARIA, A., LIU, Z., NAIR, P., TOWSELEY, D.: \textit{Computational Aspects of the Workload Distribution in the MMPP/GI/1 Queue}, IEEE Journal on Selected Areas in Communications, Vol.16, No.5, 1998, pp.640-652

[7] PALUCH, S.: \textit{Graph Theory (Teória grafov)}, Žilinská univerzita, EDIS, 2001, ISBN 80-7100-874-5

[8] MATIAŠKO, K.: \textit{Database Systems (Databázové systémy)}, Žilinská univerzita, EDIS, 2002, ISBN 80-7100-968-7.