Faster Algorithms for the Nonemptiness of Streett Automata and for Communication Protocol Pruning

Monika Rauch Henzinger\textsuperscript{1} * and Jan Arne Telle\textsuperscript{2} **

\textsuperscript{1} Systems Research Center, Digital Equipment Corporation, Palo Alto, CA
\textsuperscript{2} Department of Informatics, University of Bergen, Norway

Abstract. This paper shows how a general technique, called \emph{lock-step search}, used in dynamic graph algorithms, can be used to improve the running time of two problems arising in program verification and communication protocol design.

(1) We consider the nonemptiness problem for Streett automata: We are given a directed graph $G = (V, E)$ with $n = |V|$ and $m = |E|$, and a collection of pairs of subsets of vertices, called Streett pairs, $(L_i, U_i), i = 1..k$. The question is whether $G$ has a cycle (not necessarily simple) which, for each $1 \leq i \leq k$, if it contains a vertex from $L_i$ then it also contains a vertex of $U_i$. Let $b = \sum_{i=1}^{k} |L_i| + |U_i|$. The previously best algorithm takes time $O((m+b)\min\{n,k\})$. We present an algorithm that takes time $O(m \min\{\sqrt{m\log n, k}, n\} + b \min\{\log n, k\})$.

(2) In communication protocol pruning we are given a directed graph $G = (V, E)$ with $l$ special vertices. The problem is to efficiently maintain the strongly-connected components of the special vertices on a restricted set of edge deletions. Let $m_i$ be the number of edges in the strongly connected component of the $i$th special vertex. The previously best algorithm repeatedly recomputes the strongly-connected components which leads to a running time of $O(l^2 \sum_i m_i^2)$. We present an algorithm with time $O(\sqrt{l} \sum_i m_i^{1.5})$.

1 Introduction

Maintaining the strongly-connected components of a digraph $G = (V, E)$ efficiently under vertex or edge deletions is an unsolved problem. No data structure is known that is faster than recomputation from scratch. This is unfortunate since such a data structure would speed up various algorithms. In this paper we describe two such algorithms and show how a technique, called \emph{lock-step search}, used in dynamic graph algorithms, can improve their running time.

* monika@src.dec.edu. This research was done while at the Department of Computer Science, Cornell University, Ithaca, NY and was supported by an NSF CAREER Award, Grant No. CCR-9501712.

** telle@ii.uib.no. This research was done while visiting the Department of Computer Science, Cornell University, Ithaca, NY and was supported by a fellowship from the Norwegian Research Council.
Nonemptiness for Streett automata

A Streett automaton with Streett pairs \( (L_i, U_i) \) is an automaton on infinite words where a run is accepting if for all pairs \( i \), if the run visits infinitely many times some state in \( L_i \) then it also visits infinitely many times some state in \( U_i \). The first problem that we consider is called the nonemptiness problem for Streett automata: Given a directed graph \( G = (V, E) \) and a collection of pairs of subsets of vertices, called Streett pairs, \( (L_i, U_i), i = 1..k \), determine if \( G \) has a cycle (not necessarily simple) which, for each \( 1 \leq i \leq k \), if it contains a vertex from \( L_i \) then it also contains a vertex of \( U_i \).\(^3\)

Nonemptiness checking of Streett automata is used in computer-aided verification. Consider, for example, the problem of checking if a strongly fair finite-state system \( A \) satisfies a specification \( \phi \) in linear temporal logic ("model checking"). The system \( A \) can be modeled as a Streett automaton. The negated specification \( \neg \phi \) can be translated into an equivalent Büchi automaton, and therefore Streett automaton, \( B_{\neg \phi} \). Then model checking reduces to checking the nonemptiness of the product Streett automaton \( A \times B_{\neg \phi} \) [7].

Let \( |V| = n, |E| = m \) and \( b = \sum_{i=1..k} |L_i| + |U_i| \). The previously best algorithms for this problem take time \( O((m + b) \min\{n, k\}) \) [1, 3]. We present an \( O(m \min\{\sqrt{m \log n}, k, n\} + b \min\{\log n, k\}) \) algorithm for the problem. The improved running time is achieved through (1) lock-step search and (2) an efficient data structure for representing the Streett pairs \( (L_i, U_i) \). In model checking, frequently \( G \) has bounded out-degree. In this case \( m = O(n) \) and our algorithm has running time \( O(n \min\{\sqrt{n \log n}, k\} + b \min\{\log n, k\}) \).

Protocol Pruning

A communication system defines interactions between different components using exact rules, called protocols. Since protocol standards have become very complex, various approaches try to simplify protocols. A new technique by Lee, Neotrvali, and Sabnani [4] models a protocol as a collection of communicating finite state machines, and prunes the protocol without constructing the composite machine. The finite state machines are represented as a directed graph with \( l \) special vertices (start states), one per machine. Interactions between machines are modeled as dependencies among edges. Their algorithm repeatedly "prunes off" (i.e. deletes) edges of the graph and recomputes the strongly-connected components of the special vertices. Which edges are deleted in the next iteration depends on dependencies between the edges left in the current strongly-connected components of the special vertices. If the strongly-connected components have not changed between two iterations the algorithm terminates with these strongly-connected components representing the pruned protocol machine.

Let \( m_i \) be the number of edges in the strongly-connected component of the \( i \)th special vertex. Recomputing the strongly-connected components from scratch

\(^3\) Note that the nonemptiness problem for Streett automata usually includes also a designated root vertex (the start state) and requires that the cycle asked for be reachable from the root. For ease of presentation, we assume that a simple linear preprocessing step has computed the input graph \( G \) consisting only of vertices reachable from the root, since the other vertices will not affect the solution.