A Linear-Time Algorithm for Finding an Ambitus

B. Mishra and R. E. Tarjan

Abstract. We devise a linear-time algorithm for finding an ambitus in an undirected graph. An ambitus is a cycle in a graph containing two distinguished vertices such that certain different groups of bridges (called $B^+$, $B^-$, and $B^{+\cdot-}$-bridges) satisfy the property that a bridge in one group does not interlace with any bridge in the other groups. Thus, an ambitus allows the graph to be cut into pieces, where, in each piece, certain graph properties may be investigated independently and recursively, and then the pieces can be pasted together to yield information about these graph properties in the original graph. In order to achieve a good time-complexity for such an algorithm employing the divide-and-conquer paradigm, it is necessary to find an ambitus quickly. We also show that, using ambitus, linear-time algorithms can be devised for abiding-path-finding and nonseparating-induced-cycle-finding problems.

Key Words. Abiding-path, All-bidirectional-edges problem, Ambitus, Bridge, Nonseparating induced cycle.

1. Introduction. The concept of an ambitus was first introduced in [5] and [6], in the process of devising an efficient divide-and-conquer algorithm for the all-bidirectional-edges problem. An ambitus is a cycle in a graph containing two distinguished vertices such that certain different groups of bridges (called $B^+$, $B^-$, and $B^{+\cdot-}$-bridges) satisfy the property that a bridge in one group avoids (i.e., does not interlace with) every bridge in the other groups. (See Section 2 for a more formal definition of an ambitus.) Thus, an ambitus allows the graph to be cut into pieces, where, in each piece, certain graph properties may be investigated independently and recursively, and then the pieces can be pasted together to yield information about these graph properties in the original graph. In order to achieve a good time complexity for such an algorithm employing the divide-and-conquer paradigm, it is necessary to find an ambitus quickly, i.e., in $O(|E| + |V|)$ time. Such an algorithm first appears in [6] and yields a time complexity of $O(|E|\cdot|V|)$ for the algorithm to find all bidirectional edges of an undirected graph.

In many respects, this algorithm is a generalization of the planarity-testing algorithm, due to Hopcroft and Tarjan [3]. Like their planarity-testing algorithm, the ambitus-finding algorithm needs to decompose the graph into a set of internally

1 The research of B. Mishra was supported in part by National Science Foundation Grants DMS-8703458 and CCR-9002819. R. E. Tarjan's research at Princeton University was partially supported by DIMACS, a National Science Foundation Science and Technology Center, Grant No. NSF-STC88-09648, and by National Science Foundation Grant CCR-8929505.

2 Department of Computer Science, Courant Institute of Mathematical Sciences, New York University, 719 Broadway, New York, NY 10003, USA.

3 Department of Computer Science, Princeton University, Princeton, NJ 08544, USA, and NEC Research Institute, Princeton, NJ 08540, USA.

Received August 29, 1989; revised February 12, 1990. Communicated by Greg N. Frederickson.
vertex disjoint subpaths, although the decomposition needed for our purpose has to be somewhat different. Unlike their planarity-testing algorithm, the ambitus-finding algorithm cannot traverse the subpaths in an order, known a priori; rather the order, in which the subpaths are traversed, has to be determined dynamically. For this purpose, we have developed a novel data structure that maintains a set of (integral) intervals and supports a fast FIND-AND-UPDATE operation that detects the interval corresponding to a subpath to be visited next and updates the set of intervals, appropriately. This data structure has a good amortized time complexity, and may be of independent interest.

Subsequently, many other researchers have developed new algorithms either for ambitus finding in special classes of graphs or for certain other closely related concepts in general graphs. Two such related concepts are abiding paths and nonseparating induced cycles. In Section 8 we present linear-time algorithms for abiding paths in nonseparable graphs and nonseparating induced cycles in 3-connected graphs. The abiding paths have to be found in a “divide” step for an algorithm, due to Ohtsuki [7], for the two-vertex-disjoint-paths problem. Although Ohtsuki first claimed to have a linear-time algorithm for the abiding-path-finding problem, without the relevant implementation details, it is unclear if the algorithm he sketched can achieve the claimed time complexity. Nonseparating induced cycles need to be found repeatedly (at most \(|V|\) times) in an algorithm, due to Cheriyan and Maheswari [1], that finds three independent spanning trees rooted at a distinguished vertex, in a 3-connected graph.

In [4] Krishnan et al. have developed a simple linear-time algorithm to find an ambitus in a planar graph. Although their algorithm is simple and elegant, the techniques employed do not generalize to other classes of graphs. In [10] Sundar has developed an \(O(|E| + |V| \log |V|)\)-time algorithm to find an abiding path in a nonseparable graph. This algorithm can be used for the ambitus-finding problem, if the graph is suitably modified. However, such an algorithm has a linear-time behavior, only if the graph is dense. In [1] Cheriyan and Maheswari have developed a linear-time algorithm to find a nonseparating induced cycle in a 3-connected graph. With suitable modifications to certain basic steps in the algorithm, a different linear-time algorithm for the ambitus-finding problem can be devised.

The paper is organized as follows: In Section 2 we define some graph theoretic terminology and introduce other key concepts required in the paper. In Section 3 we demonstrate that an ambitus always exists in a nonseparable graph, and an ambitus can be found by a naïve algorithm of complexity \(O(|E| \cdot |V|)\). In Section 4 we provide a sketch of the main algorithm, and in the subsequent two sections we provide the implementation details, in order to guarantee a linear-time behavior. In the last section we give two simple applications of the ambitus-finding algorithm.

2. Preliminaries. In this section we define some graph theoretic terminology and introduce other key concepts required in the paper. Most important among these are the terms: bridge, residual path, cross-cut, and ambitus. The definitions are similar to those used in the context of Tutte’s theorem on Hamiltonian circuits in