Mathematical Programming 78 (1997) 169–177

Dynamic trees as search trees via Euler tours, applied to the network simplex algorithm

Robert E. Tarjan [1]

Department of Computer Science, Princeton University, Princeton, NJ 08544 and NEC Research Institute, 4 Independence Way, Princeton, NJ 08540, USA

Received 15 September 1995; revised manuscript received 17 May 1996

Abstract

The *dynamic tree* is an abstract data type that allows the maintenance of a collection of trees subject to joining by adding edges (*linking*) and splitting by deleting edges (*cutting*), while at the same time allowing reporting of certain combinations of vertex or edge values. For many applications of dynamic trees, values must be combined along paths. For other applications, values must be combined over entire trees. For the latter situation, an idea used originally in parallel graph algorithms, to represent trees by Euler tours, leads to a simple implementation with a time of $O(\log n)$ per tree operation, where n is the number of tree vertices. We apply this representation to the implementation of two versions of the network simplex algorithm, resulting in a time of $O(\log n)$ per pivot, where n is the number of vertices in the problem network. © 1997 The Mathematical Programming Society, Inc. Published by Elsevier Science B.V.

1. Introduction

Consider a collection of unrooted trees, each initially a single vertex and no edges, on which two structural update operations are allowed:

link$(\{v, w\})$: Combine the trees containing vertices v and w by adding the edge $\{v, w\}$. This operation does nothing if v and w are already in the same tree.

cut$(\{v, w\})$: Break the tree containing edge $\{v, w\}$ in two by deleting the edge $\{v, w\}$. This operation does nothing if $\{v, w\}$ is not an existing tree edge.

[1] Research at Princeton University partially supported by the National Science Foundation, Grant No. CCR-8920505, and the Office of Naval Research, Contract No. N0014-91-J-1463. Work during a visit to M.I.T. partially supported by ARPA Contract No. 14-95-1-1246. Email: ret@cs.princeton.edu.

0025-5610/97/$17.00
Published by Elsevier Science B.V.
PII S0025-5610(97)00015-4

Suppose further that each tree vertex v has an associated real value, denoted by $val(v)$, and that the following operations on values are allowed:

find-val(v): return $val(v)$.

find-min-val(v): return a vertex of minimum value in the tree containing vertex v.

change-val(v, x): set $val(v)$ equal to x.

add-val(v, x): add x to $val(w)$ for each vertex w in the tree containing v.

Section 2 of this paper presents a simple implementation of these six tree operations with a running time of $O(\log n)$ per operation, where n is the number of vertices in the tree or trees involved in the operation. The idea used is to linearize each tree by constructing an *Euler tour* that traverses each edge once in each direction and includes one stop at each vertex, and to represent such a tour by a search tree. Linking and cutting of trees translate into simple combinations of catenation and splitting operations on tours, and on the corresponding search trees. The $O(\log n)$ time bound per tree operation is amortized if splay trees [22] are used in the representation and worst-case if balanced search trees are used. Section 3 describes the use of the Euler tour structure in the implementation of two versions of the network simplex algorithm.

The Euler Tour representation of trees was originally used in fast parallel graph algorithms [26]. Later, Miltersen et al. [18] adapted it to represent trees subject to linking and cutting but without vertex values. Henzinger and King [16] independently adapted it to represent trees each of whose vertices has an associated list and an associated value which is the size of the associated list; values are combined using sum instead of minimum. Both of these representations use balanced search trees.

Additional related work deals with a class of dynamic trees in which vertex or edge values are combined along paths, rather than over an entire tree. Dynamic trees of this kind arise in various network flow algorithms [10–13,21,23,25] and in other settings [4,6]. Two different representations of such trees have been proposed. The first, by Sleator and Tarjan [21,22,25] decomposes each tree into vertex-disjoint paths and represents these paths by search trees, either biased search trees [3] or splay trees [22]. With the former representation the time per tree operation can be made $O(\log n)$ in the worst case; with the latter representation the time per tree operation is $O(\log n)$ amortized over a worst-case sequence of operations. Frederickson [7,8] proposed a rather different representation, called the *topology tree*, which is related to the rake and compress operations used in parallel tree processing [17]. This representation, too, has an $O(\log n)$ worst-case time bound per tree operation.

Both the Sleator–Tarjan representation and the Frederickson representation can be applied to the problem we consider here, achieving the same $O(\log n)$ time bound (see e.g. [10]). But we regard these solutions are inferior, for two reasons. First, both data structures must be extended to handle tree vertices of arbitrary degree. Second, both structures, even without the unbounded degree extension, are noticably more complicated than the Euler tour structure, which ultimately is just a straightforward application of search trees, and is likely to be easier to implement and more efficient in practice.