Discrete Applied Mathematics 156 (2008) 2493–2500

Contents lists available at ScienceDirect

Discrete Applied Mathematics

journal homepage: www.elsevier.com/locate/dam

Edge ranking and searching in partial orders

Dariusz Dereniowski

Department of Algorithms and System Modeling, Gdańsk University of Technology, Poland

ARTICLE INFO

Article history:
Received 19 September 2006
Received in revised form 31 July 2007
Accepted 6 March 2008
Available online 25 April 2008

Keywords:
Dag
Edge ranking
Graph searching
Poset
Spanning tree

ABSTRACT

We consider a problem of searching an element in a partially ordered set (poset). The goal is to find a search strategy which minimizes the number of comparisons. Ben-Asher, Farchi and Newman considered a special case where the partial order has the maximum element and the Hasse diagram is a tree (tree-like posets) and they gave an $O(n^4 \log^3 n)$-time algorithm for finding an optimal search strategy for such a partial order. We show that this problem is equivalent to finding edge ranking of a simple tree corresponding to the Hasse diagram, which implies the existence of a linear-time algorithm for this problem.

Then we study a more general problem, namely searching in any partial order with maximum element. We prove that such a generalization is hard, and we give an $O(\frac{\log n}{\log(\log n)})$-approximate polynomial-time algorithm for this problem.

1. Introduction

Assume that a partially ordered set (poset) $(S, \leq_R)$ is given, and let t be an element that we want to find. We can select an element $x \in S$ and ask a question of the form "$t \leq_R x$". If the answer is "yes" then we continue our search in the set $\{a \in S : a \leq_R x\}$ and if the answer is "no" then we have to search in the complimentary part. The goal is to find, for a given poset, a search strategy which asks in the worst case as few questions as possible. This problem has several applications [1]: software testing (finding the bug in a program), finding corrupted nodes in a tree-like data (like file systems or databases), or information retrieval. Another motivation is that this problem is a generalization of the binary search in linearly ordered sets. A related searching model has been considered in [9].

A special case of the above problem was considered in [1], where the authors assumed that the Hasse diagram of the poset is a rooted tree (tree-like poset). There exists an algorithm of running time $O(n^4 \log^3 n)$, where n is the number of elements in the tree-like poset, which finds an optimal search strategy [1]. Authors in [10] gave an exponential-time algorithm for finding search strategies in general posets, and they minimized the average cost of the search. In this paper we assume that a partial order with maximum element is given (the Hasse diagram does not have to be a tree). We prove that finding an optimal search strategy for such a poset is hard and we give a polynomial-time approximation algorithm with sublogarithmic approximation ratio.

The edge ranking problem has the following formulation. Given a simple graph G, a function $c: E(G) \to \{1, \ldots, k\}$ is an *edge k-ranking* of G if each path connecting two edges x, y satisfying $c(x) = c(y)$ contains an edge z such that $c(z) > c(x)$. The smallest number k such that there exists an edge k-ranking is called the *edge ranking number* of G and is denoted by $\chi'_r(G)$. The numbers $1, \ldots, k$ are called colors: the edge ranking problem is a modification of the classical graph coloring problem (in particular if the edges x and y share a common vertex then $c(x) \neq c(y)$ and this property will be used several times in this paper). An edge ranking of G is *optimal* if it uses $\chi'_r(G)$ colors, i.e. $\chi'_r(G) = k$. In the edge ranking problem the goal is to find an optimal edge ranking for a given graph G. The edge ranking problem is hard in general [7], and in the case of multitrees [3]. On the other hand there exists a linear-time algorithm for finding an optimal edge ranking of a simple tree [8]. We describe

E-mail address: deren@eti.pg.gda.pl.

0166-218X/$ – see front matter © 2008 Elsevier B.V. All rights reserved.
doi:10.1016/j.dam.2008.03.007

a connection between the problem of searching for an element in posets and the edge ranking problem. This connection allows us to derive, using several facts concerning edge rankings, the results for the searching problem defined above. This connection in particular implies the existence of a linear-time algorithm for finding optimal search strategy for a tree-like poset, which improves the result given in [1]. The edge ranking problem has applications in the parallel assembly of modular products from their components [3,4] or in the parallel database query processing [2,11].

The paper is organized as follows. Section 2 gives a formal definition of the search problem in partially ordered sets. We also give necessary graph theoretic terminology. In Section 3 we define the problem MERB, where the goal is to find for a directed acyclic graph D with one target its spanning tree with one target (called *branching*), such that the edge ranking number of the underlying simple tree is as small as possible. We show that this problem is equivalent to finding an optimal search strategy in a poset with maximum element. In Section 4 we show that both problems are hard even in some restricted cases. Section 5 gives an approximate algorithm for finding branchings of low degree, and in Section 6 we give an approximate algorithm for finding search strategies.

2. Preliminaries

A *directed path* is a graph P_n such that

$$P_n = (\{v_1, \ldots, v_n\}, \{(v_1, v_2), (v_2, v_3), \ldots, (v_{n-1}, v_n)\}).$$

Let $D = (V(D), E(D))$ be a directed graph. We say that D' is a *subgraph* of D, $D' \subseteq D$, if $V(D') \subseteq V(D)$ and $E(D') \subseteq E(D)$. Given a set $S \subseteq V(D)$, the subgraph *induced by* S is defined as $D[S] = (S, \{(u, v) \in E(D) : u, v \in S\})$. A digraph D is *acyclic* if it does not contain as a subgraph a directed path P_n such that $v_1 = v_n$. A vertex $v \in V(D)$ is *reachable* from a vertex $u \in V(D)$ if there is a directed path from u to v in D. Define $D_v = D[\{u \in V(D) : v \text{ is reachable from } u\}]$. For a set of vertices (respectively edges) S of D we define $D - S = D[V(D) \setminus S]$ ($D - S = (V(D), E(D) \setminus S)$, respectively).

For a vertex $v \in V(D)$ let

$$N_D^+(v) = \{u \in V(D) : (u, v) \in E(D)\},$$
$$N_D^-(v) = \{u \in V(D) : (v, u) \in E(D)\}$$

and $N_D(v) = N_D^+(v) \cup N_D^-(v)$. The *outdegree*, *indegree* and *degree* of v are defined as

$$\deg_D^-(v) = |N_D^-(v)|, \deg_D^+(v) = |N_D^+(v)|, \deg_D(v) = |N_D(v)|,$$

respectively. We say that arc (u, v) is *outgoing* from u and *incoming* to v. The *indegree* and *degree* of D are defined as $\Delta^+(D) = \max\{\deg_D^+(v) : v \in V(D)\}$ and $\Delta(D) = \max\{\deg_D(v) : v \in V(D)\}$, respectively. We say that a vertex v of a directed graph D is a *target* if $\deg_D^-(v) = 0$. In this paper we only consider directed acyclic graphs (dags) with one target.

Given a digraph D, we define a simple graph

$$G(D) = (V(D), \{\{u, v\} : (u, v) \in E(D) \text{ or } (v, u) \in E(D)\}).$$

For a simple graph G and $v \in V(G)$ we analogously define $N_G(v) = \{u \in V(G) : \{u, v\} \in E(G)\}$, the *degree* of v, $\deg_G(v) = |N_G(v)|$, and the *degree* of G, $\Delta(G) = \max\{\deg_G(v) : v \in V(G)\}$. A simple graph H is a *subgraph* of G, $H \subseteq G$, if $V(H) \subseteq V(G)$ and $E(H) \subseteq E(G)$. A simple graph G is *connected* if there exists a path between each pair of vertices, i.e. for each $u, v \in V(G)$ there exists a sequence of vertices $v_0 = u, v_1, \ldots, v_j = v$ such that $\{v_i, v_{i+1}\} \in E(G)$ for $i = 0, \ldots, j - 1$. A directed graph D is *connected* if $G(D)$ is connected. Observe that a directed acyclic (*dag*) graph with one target is connected.

Instead of considering a poset $(S, \leq_R)$, we consider its representation, namely a dag D such that $V(D) = S$ and there is an arc (u, v) in $E(D)$ if and only if $u \leq_R v$ and there is an edge connecting u and v in the corresponding Hasse diagram. This, in particular, implies that D is a dag with one target r. Now we define the search problem in terms of a directed graph. Let t be a vertex of D that we want to find. If $|V(D)| = 1$ then a *search strategy* $A(D, t)$ outputs the only element $t \in V(D)$. If $|V(D)| > 1$ then $A(D, t) = (v, A(D_v, t), A(D - V(D_v), t))$, where $v \in V(D) \setminus \{r\}$ is an element which is being compared to the desired element t. The second element in the triplet is a search strategy executed in the case of an "yes" answer (i.e. $t \in V(D_v)$, or equivalently $t \leq_R v$), and the last element in the triplet is a search strategy used if the answer is "no" (i.e. $t \notin V(D_v)$). The cost $w(A, t)$ of finding an element t is the number of questions asked (the number of comparisons made) during the search of t by A. Then, the cost of A is

$$w(A) = \max_{t \in V(D)} w(A, t).$$

Finally, for a given D we define

$$w(D) = \min_{A \in \mathcal{A}_D} w(A),$$

where $\mathcal{A}_D$ is the set of all search strategies for the dag D. In this paper we consider the graph searching problem where the goal is to find, for a given dag D, an optimal (i.e. of cost $w(D)$) search strategy. The following equation follows directly from the definition

$$w(D) = 1 + \min_{v \in V(D)} \max\{w(D_v), w(D - D_v)\}. \tag{1}$$