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

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

3. Connection between searching in posets and the graph ranking problem

Let a dag D with one target r be given. A *spanning tree* of D is any connected subgraph T such that $V(T) = V(D)$, $|E(T)| = |V(T)| - 1$. A *branching* of D is a spanning tree T, which has one target. Note that in that case the target in T must be the vertex r. Let us introduce the Minimum Edge Ranking Branching (MERB) problem, where the goal is to find, for a given dag D, a branching T such that $\chi'_r(G(T))$ is minimum, i.e. $\chi'_r(G(T)) \leq \chi'_r(G(T'))$ for each branching T' of D. The symbol T^* will be used to denote an optimal solution to this problem. Given an edge ranking c of a simple graph G, a color i is *visible* for an edge $e \in E(G)$ (a vertex $v \in V(G)$) in c if there exists a path connecting e (resp. v) to an edge e', $c(e') = i$, and all edges of this path have colors smaller than i. Note that if G is connected then the color $\chi'_r(G)$ appears exactly once in c. Other simple facts concerning edge rankings are: $\chi'_r(H) \leq \chi'_r(G)$, where $H \subseteq G$, and $\chi'_r(G) \geq \Delta(G)$.

The next two lemmas state the correspondence between the MERB problem and the graph searching problem.

Lemma 1. *We have $\chi'_r(G(T^*)) \leq w(D)$.*

Proof. Let A be an optimal search strategy for D. We use an induction on the number of vertices of D to find a branching T such that $\chi'_r(G(T)) \leq w(A)$. If $|V(D)| = 1$ then $\chi'_r(G(D)) = 0 = w(A)$. Assume that the hypothesis is true for each dag on at most n vertices and let D be a dag with $n+1$ vertices. Let $t \in V(D)$. By the definition we have $A(D,t) = (v, A(D_v,t), A(D-V(D_v),t))$, for some $v \in V(D)$. Observe that D_v and $D - V(D_v)$ are dags with targets v and r, respectively. Thus, by the induction hypothesis, we have that there exist branchings T_1 and T_2 of D_v and $D - V(D_v)$, respectively, such that

$$\chi'_r(G(T_1)) \leq w(A(D_v, t)), \tag{2}$$

$$\chi'_r(G(T_2)) \leq w(A(D - V(D_v), t)). \tag{3}$$

We define an edge ranking c for $G(T)$ in such a way that $c|_{V(G(T_i))} = c_i$, $i = 1, 2$ where c_i is an optimal edge ranking of $G(T_i)$, $i = 1, 2$. Finally, the edge $\{v, u\}$, where $(v, u) \in E(T)$, gets the unique color $\max\{\chi'_r(G(T_i)) : i = 1, 2\} + 1$ under c. Note that $T_1 \cup T_2 = T - \{\{u, v\}\}$. So, c is a proper edge ranking of $G(T)$ and by (1)–(3) it uses at most $w(A)$ colors. □

Observe that the vertex u in the proof of Lemma 1 can be chosen arbitrary, i.e. the only restriction is that $u \in N^-(v)$. Note that D has been defined in such a way that it does not contain any transitive arcs and the above fact implies that it has a desired branching.

Lemma 2. *We have $\chi'_r(G(T^*)) \geq w(D)$.*

Proof. Let c be an edge k-ranking of $G(T^*)$. We recursively create a search strategy A and we prove by induction on the number of vertices of a branching that $w(A) \leq k$. The edge $\{u, v\} \in E(G(T^*))$ colored with k by c is unique. Assume w.l.o.g. that $(v, u) \in E(T^*)$. Since T^* is a branching, if we remove the edge $\{u, v\}$ from $G(T^*)$ then we obtain two branchings T^*_v and $T^* - V(T^*_v)$. Observe that $\max\{\chi'_r(G(T^*_v)), \chi'_r(G(T^* - V(T^*_v)))\} = k - 1$. By the induction hypothesis we have that $k - 1 \geq w(D_v)$ and $k - 1 \geq w(D - V(D_v))$, which completes the proof. □

We have proved the following.

Corollary 1. *For a given dag D with one target we have $w(D) = \chi'_r(G(T^*))$.* □

Corollary 2. *If D is a tree with one target then $w(D) = \chi'_r(G(D))$.* □

Theorem 1 ([8]). *There exists a linear-time algorithm for finding an optimal edge ranking of a given tree.* □

Corollary 3. *There exists a linear-time algorithm for finding optimal search strategy in a tree-like poset.*

Proof. Let T be the directed tree corresponding to the Hasse diagram of the tree-like poset. We use the algorithm given in [8] to find an optimal edge ranking c of $G(T)$. Let $\{u, v\}$ be an edge of $G(T)$ and let T' be the connected component of $G(T) - \{e \in E(G(T)) : c(e) > c(\{u, v\})\}$ containing $\{u, v\}$. In order to compute the corresponding search strategy efficiently we compute for the edge $\{u, v\}$ of $G(T)$ pointers to the edges having the biggest labels in the connected components of $G(T') - \{\{u, v\}\}$. Let $P(\{u, v\})$ denote their set. Observe that there are at most two such pointers. Moreover, given $P(\{u, v\})$ for each $\{u, v\} \in E(T)$, the proof of Lemma 2 gives a linear-time recursive algorithm.

The algorithm given in [8] computes for each vertex v of a rooted tree $G(T)$ the set $S(v)$ of colors visible for v and assigned to the edges of $G(T_v)$. For each $v \neq r$ define $S'(v) = \{i \in S(v) : i < c(\{u, v\})\}$, where u is the father of v in $G(T)$. Moreover let $S'(r) = S(r)$. Observe that $\sum_{v \in V(T)} |S'(v)| = O(|E(T)|) = O(|V(D)|)$, because for each edge e its color belongs to exactly one set $S'(v)$. We add to P the elements as follows. For each $v \in V(T)$ and each pair of integers $i, j \in S'(v)$ such that $i < j$, $i+1, \ldots, j-1 \notin S'(v)$ add to $P(e_2)$ the edge e_1, where $e_1, e_2 \in E(T_v)$, $c(e_1) = i$, $c(e_2) = j$ and the paths connecting e_1, e_2 to v contain only colors smaller than i and j, respectively. Furthermore if $v \neq r$ then for $i = \max(S'(v))$ add the edge $e \in E(G(T_v))$ colored with i (and connected to v by a path with all colors smaller than i) to $P(\{u, v\})$. □

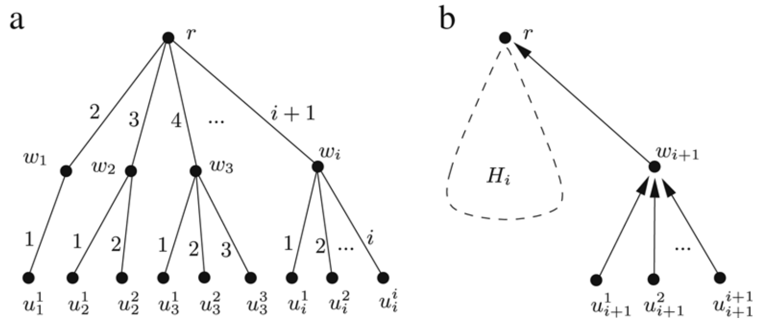

Fig. 1. (a) graph $G(H_i)$ and its optimal edge ranking; (b) graph H_{i+1}.

4. The MERB problem is hard

Let us recall the Minimum Set Cover (MSC) problem. Given a collection $\mathcal{C} = \{S_1, \ldots, S_l\}$ of subsets of $S = \{a_1, \ldots, a_n\}$, and an integer k, does there exist $\mathcal{C}' \subseteq \mathcal{C}$ such that $|\mathcal{C}'| \leq k$ and $\mathcal{C}'$ *covers* S, i.e. for each $a \in S$ there exists $S' \in \mathcal{C}'$ such that $a \in S'$?

We will reduce the above NP-complete problem [5] to the decision version of the MERB problem: given a dag D (with one target), and an integer k', does it exist a branching T of D such that $\chi'_r(G(T)) \leq k'$?

Now let us assume that an instance of the MSC problem is given. We have to create an appropriate digraph D and define an integer k'. First we define a digraph H_i, $i \geq 1$ as follows

$$V(H_i) = \{r\} \cup \{w_1, \ldots, w_i\} \cup \{u_j^s : j = 1, \ldots, i, s = 1, \ldots, j\},$$

$$E(H_i) = \{(w_j, r) : j = 1, \ldots, i\} \cup \{(u_j^s, w_j) : j = 1, \ldots, i, s = 1, \ldots, j\}.$$

Fig. 1(a) depicts the graph $G(H_i)$ and its optimal edge ranking (we will prove the optimality in Lemma 3) while Fig. 1(b) shows the recursive construction of the graphs H_i, $i > 1$.

Lemma 3. *We have $\chi'_r(G(H_i)) = i + 1$, $i \geq 1$. If c is an optimal edge ranking of $G(H_i)$ then the colors $\{2, \ldots, i + 1\}$ are visible for r.*

Proof. We prove by induction on $i \in \mathbb{N}$. The lemma clearly holds for $i = 1$. Consider the subgraph H_i and the edge ranking c of $G(H_i)$ and assume that the claim holds for H_{i-1}. We have that $\deg_{G(H_i)}(w_i) = i + 1$. Since the edges adjacent to the same vertex should have pairwise different colors we obtain that $\chi'_r(G(H_i)) \geq i + 1$. Fig. 1(a) depicts an edge $(i + 1)$-ranking of $G(H_i)$ which gives that $\chi'_r(G(H_i)) = i + 1$. Since $\deg_{G(H_i)}(w_{i-1}) = i$, the biggest color $i \leq p \leq i + 1$ assigned to an edge e adjacent to w_{i-1} is visible for r. The colors $c(\{r, w_i\}), \ldots, i + 1$ and adjacent to w_i are visible for r. Assume for a contradiction that $c(\{r, w_i\}) < i + 1$. Clearly, $c(\{r, w_i\}) \neq p$, because e and $\{r, w_i\}$ are adjacent or connected with a path containing the edge $\{r, w_{i-1}\}$, where by assumption $c(\{r, w_{i-1}\}) < p$. So, $c(\{r, w_i\}) < p$. Note that e and each of the edges $\{w_i, u_i^j\}$, $j \in \{1, \ldots, i\}$, are connected by a path with edges having colors smaller than p. So, no edge $\{w_i, u_i^j\}$ for $j = 1, \ldots, i$ gets the color p – a contradiction with the fact that c is an edge $(i + 1)$-ranking. By the induction hypothesis, $\chi'_r(G(H_{i-1})) = i$ and the set of colors visible for r in $c|_{E(G(H_{i-1}))}$ is $\{2, \ldots, i\}$. Let $c(\{r, w_i\}) = i + 1$ and $\{c(\{w_i, u_i^j\}) : j = 1, \ldots, i\} = \{1, \ldots, i\}$, which completes the proof. □

In order to distinguish the vertices of different subgraphs H_i we will write in the following $r(H_i)$, $w_j(H_i)$, $u_j^s(H_i)$ instead of r, u_j, w_j^s, $j = 1, \ldots, i$, $s = 1, \ldots, j$. Let us define $N = \max\{n, l\}$. The target in a dag D' is denoted by r. Let $N^+_{D'}(r) = \{v_1, \ldots, v_{2N+2}\} \subseteq V(D')$. Since r is the target it holds $N^-_{D'}(r) = \emptyset$. Now we create D'_{v_i} for each $i = 1, \ldots, 2N + 2$. If $i = 1, 2$ then $D'_{v_i} = (\{v_i\}, \emptyset)$. For $i = 3, \ldots, N + 1$ we have $D'_{v_i} = H_{i-2}$. For $i = N + 2, \ldots, 2N + 1$ we define the subgraph D'_{v_i} as follows

$$D'_{v_i} = (\{v_i, w_i^1, \ldots, w_i^{i-1}\}, \{(w_i^j, v_i) : j = 1, \ldots, i - 1\}).$$

If $i = 2N + 2$ then

$$V(D'_{v_i}) = \{v_i\} \cup \{w_i^1, \ldots, w_i^{i-k-1}\} \cup \{u_i^1, \ldots, u_i^{i-2}\},$$

$$E(D'_{v_i}) = \{(w_i^j, v_i) : j = 1, \ldots, i - k - 1\} \cup \{(u_i^j, w_i^{i-k-1}) : j = 1, \ldots, i - 2\}.$$

Fig. 2 depicts the corresponding simple graph $G(D')$, and its optimal edge ranking (the optimality follows from the fact that $\Delta(G(D')) = 2N + 2$).

In our reduction there are vertices corresponding to the elements of the set S denoted by $v[a_1], \ldots, v[a_n]$. Similarly, for each $S_i \in \mathcal{C}$ we introduce a vertex $v[S_i]$, $i = 1, \ldots, l$. Now we define the digraph D corresponding to the instance of the MSC problem:

$$V(D) = V(D') \cup \{v[S_1], \ldots, v[S_l]\} \cup \{v[a_1], \ldots, v[a_n]\},$$

$$E(D) = E(D') \cup \{(v[S_i], v_{i+1}) : i = 1, \ldots, l\} \cup \{(v[S_i], v_{2N+2}) : i = 1, \ldots, l\} \cup \{(v[a_j], v[S_i]) : a_j \in S_i, S_i \in \mathcal{C}\}.$$

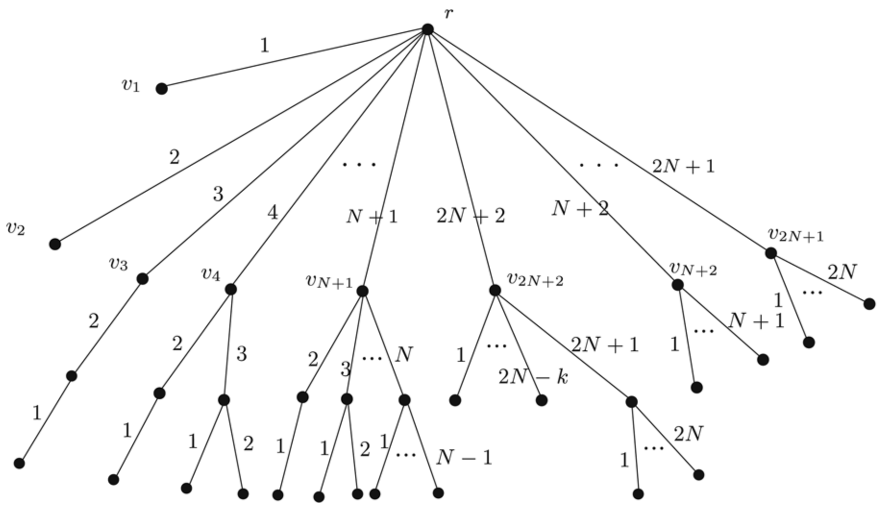

Fig. 2. The simple graph $G(D')$ and its edge $(2N+2)$-ranking.

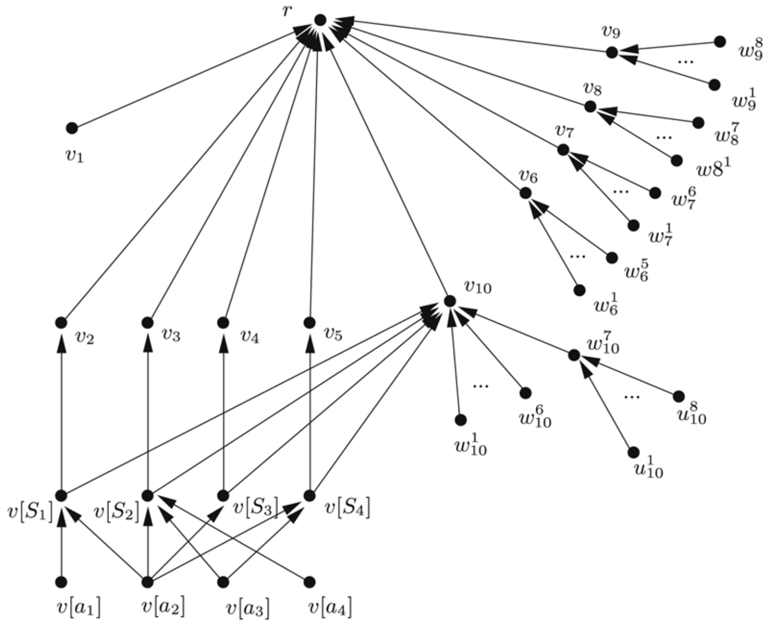

Fig. 3. An example of a complete digraph D.

Finally let $k' = 2N + 2$.

Let us first create an example of a complete dag D. Assume that the following instance of the MSC problem is given:

$$S = \{a_1, a_2, a_3, a_4\},$$
$$S_1 = \{a_1, a_2\}, \qquad S_2 = \{a_2, a_3, a_4\}, \qquad S_3 = \{a_2\}, \qquad S_4 = \{a_2, a_3\}.$$

Let $k = 2$. In this case $N = 4$. The corresponding digraph D is given in Fig. 3, where for simplicity the dag D does not contain subgraphs H_1, H_2, H_3.

From the definition of D it follows that if T is any branching of D then

$$E(D') \subseteq E(T). \tag{4}$$

Lemma 4. *Let T be a branching of D. We have that $\chi'_r(G(T)) \le 2N + 2$ if and only if for each $i = 1, \ldots, 2N + 2$ it holds $\chi'_r(G(T_{v_i})) \le i - 1$.*

Proof. ($\Leftarrow$) Each edge $\{r, v_i\} \in E(G(T))$ can be labeled with color i, which together with an optimal edge ranking of $G(T_{v_i})$, $i = 1, \ldots, 2N + 2$, gives an edge $(2N + 2)$-ranking of $G(T)$.

($\Rightarrow$) Let c be an edge $(2N + 2)$-ranking of $G(T)$. Note that each color appears exactly once for the edges incident to r, because $\deg_{G(T)}(r) = 2N+2$ and the colors for these edges have to be pairwise different. Moreover, for $e \in E(G(T_{v_i}))$ satisfying $c(e) = \max(c(E(G(T_{v_i}))))$, $i \in \{2, \ldots, 2N + 2\}$ we have $c(e) < c(\{r, v_i\})$, because otherwise the colors of $\{r, v_i\}$, e are visible from r and thus forbidden for all edges $\{r, v_j\}$, $j = 1, \ldots, 2N + 2$, $j \neq i$, which implies that c uses at least $2N + 3$ colors which by assumption is not possible. Furthermore, for $i \geq 3$ we have that $\chi'_r(G(T_{v_i})) \geq \Delta(G(T_{v_i})) = i - 1$. So, the edge ranking c restricted to the edges incident to r is $c(\{r, v_i\}) = i$ for $i = 3, \ldots, 2N + 2$. Note that the edges $\{r, v_1\}$, $\{r, v_2\}$ must be labeled with 1 and 2. Since $E(T_{v_1}) = \emptyset$ we may w.l.o.g. assume that $c(\{r, v_1\}) = 1$ and $c(\{r, v_2\}) = 2$. □

Lemma 5. *If $\mathcal{C}'$ is a solution to the MSC problem then there exists a branching T of D such that $\chi'_r(G(T)) \leq k' = 2N + 2$.*

Proof. By Eq. (4) we only have to define the arcs of T outgoing from $v[S_i]$, $i = 1, \ldots, l$, and $v[a_i]$, $i = 1, \ldots, n$. Let $(v[S_i], v_{2N+2}) \in E(T)$ if $S_i \in \mathcal{C}'$ and $(v[S_i], v_{i+1}) \in E(T)$ if $S_i \notin \mathcal{C}'$. Since $\mathcal{C}'$ covers S, for each $a_i \in S$ there exists $S_j \in \mathcal{C}'$, such that $a_i \in S_j$. So, let $(v[a_i], v[S_j]) \in E(T)$. Clearly, T is a branching of D and by Lemma 4 we only have to prove that $\chi'_r(G(T_{v_i})) \leq i - 1$ for each $i = 1, \ldots, 2N + 2$. We have $E(T_{v_1}) = \emptyset$ and $E(T_{v_2}) \subseteq \{(v[S_1], v_2)\}$. If $i \in \{3, \ldots, N + 1\}$ then $E(T_{v_i}) \setminus E(H_i) \subseteq \{(v[S_{i-1}], v_i)\}$ and by Lemma 3 color 1 is available for the edge $\{v[S_{i-1}], v_i\} \in E(G(T))$. If $i \in \{N+2, \ldots, 2N+1\}$ then the claim clearly holds, because $|E(T_{v_i})| = i - 1$. Finally, let $i = 2N + 2$. Label the edges corresponding to the arcs incoming to w^{2N-k+1}_{2N+2} with $1, \ldots, 2N$. Let $\{v_{2N+2}, w^{2N-k+1}_{2N+2}\}$ be labeled with $2N + 1$. The edges corresponding to the remaining arcs of D' incoming to v_{2N+2} get colors $1, \ldots, 2N - k$. Thus, there are k remaining colors, namely $2N - k + 1, \ldots, 2N$, which can be assigned to the edges $\{v_{2N+2}, v[S_j]\}$, where $S_j \in \mathcal{C}'$. Since $2N - k + 1 > N$, the edges corresponding to the arcs incoming to $v[S_j]$ can get colors $1, \ldots, \deg^+_T(v[S_j])$ for each S_j such that $(v[S_j], v_{2N+2}) \in E(T)$.

Now let us assume that a solution to the MERB problem is given, i.e. T is a branching of D, such that $\chi'_r(G(T)) \leq k' = 2N+2$. By Lemma 4 we have that

$$\chi'_r(G(T_{v_i})) \leq i - 1, \quad i = 1, \ldots, 2N + 2. \tag{5}$$

This, and Lemma 3 imply that

$$(v[S_i], v_{i+1}) \in E(T) \Rightarrow \deg^+_T(v[S_i]) = 0, \tag{6}$$

where $i \in \{1, \ldots, l\}$. This means that if $(v[a_i], v[S_j]) \in E(T)$ then $(v[S_j], v_{2N+2}) \in E(T)$. So, we define

$$S_i \in \mathcal{C}' \quad \text{iff } (v[S_i], v_{2N+2}) \in E(T), \quad i \in \{1, \ldots, l\}.$$

By (6) we have that $\mathcal{C}'$ covers S. If $|\{(v[S_i], v_{2N+2}) : S_i \in \mathcal{C}'\}| = k'' > k$ then by the definition of D we have that

$$\deg^+_T(v_{2N+2}) = 2N + 1 - k + k'' > 2N + 1$$

which implies that $\chi'_r(G(T_{v_{2N+2}})) \geq \Delta(G(T_{v_{2N+2}})) > 2N + 1$ — a contradiction with Lemma 4. We have obtained the following.

Lemma 6. *If there exists a branching T of D, such that $\chi'_r(G(T)) \leq 2N + 2$ then there exists a solution to the MSC problem.* □

Theorem 2. *The MERB ranking problem is NP-hard for dags D with one target r, such that each directed path is of length at most* 3.

Proof. The problem is clearly in NP and the number of vertices of D is at most $(2N + 2)^3$, where $N = \max\{n, l\}$. It is easy to see that the length of each directed path in D is at most 3. Lemmas 5 and 6 complete the proof. □

Corollary 1 and Theorem 2 give the following.

Theorem 3. *The problem of searching in a poset with maximum element is NP-hard, even if the height of the Hasse diagram is at most* 3. □

5. Minimum degree branchings of *D*

Let a dag D with one target be given. For brevity denote $n = |V(D)|$. By $\widetilde{T}^*$ denote a minimum degree branching of D. Let $\widetilde{\Delta}^* = \Delta(\widetilde{T}^*)$. We are interested in finding branchings T of low degree, because we will derive in the following an upper bound for the edge ranking number of $G(T)$ as a function of $\Delta(T)$. An approximate algorithm DMDST for this problem has been described in [6]. This algorithm takes any directed graph D as an input, but the running time is not polynomial in the size of D. Since in our case D is acyclic and has one target, we will be able to improve the running time. The outline of DMDST (as it has been described in [6]) is as follows. Initially we have any branching T of D. For each arc $(v, u) \in E(T)$ such that $\deg^+_T(u) > \Delta^+(T) - \lceil \log_{1+\epsilon} n \rceil$ run a procedure Improvement $(D, T, (v, u))$, where $\epsilon > 0$ is any constant. If the branching T changed during the execution of the procedure then Improvement returns true. Otherwise the return value is false. The algorithm repeats the above step until the procedure Improvement returns false for each arc for which it is called.

Let $p(n)$ and I denote the running time of Improvement and the number of improvement steps, respectively. The running time of the DMDST algorithm is $O(p(n) \cdot I)$.

The key part of this algorithm is the procedure Improvement(D, T, (v, u)) which can be described as follows: if there exists an arc $(v, x) \in E(D)$ such that $\deg_T^+(x) < \deg_T^+(u) - 1$ then remove (v, u) from T, add (v, x) to T and return true (such a vertex v is called an *active* vertex); if no such arc (v, x) exists then return false. We have the following lemma.

Lemma 7. *The running time of Improvement is $O(\Delta(G(D)))$.*

Proof. We may assume that for each vertex $z \in V(D)$ the value of $\deg_T^+(z)$ is given. So, the algorithm consists of a loop over the elements $x \in N_D^-(v)$ and for each such element we check the appropriate condition in constant time. We will describe in the proof of Lemma 8 a data structure for storing the sets $N_D^-(v)$, $v \in V(D)$, but these details are not important here. We remove (v, u) from T and add (v, x) to T in $O(\deg_T^+(x) + \deg_T^+(u)) = O(\Delta(G(D)))$ time, assuming that neighborhood list is the data structure for T. In addition we increment $\deg_T^+(x)$ and decrement $\deg_T^+(u)$. $\square$

Now we will bound the number of improvement steps. Define a function $f_T : V(T) \to \mathbb{N}$ in such a way that $f_T(v) = (\deg_T^+(v))^2$ and for a branching T define $F(T) = \sum_{v \in V(T)} f_T(v)$. Assume that T and T' are trees before and after the improvement step, respectively. We have

$$\begin{aligned} F(T) - F(T') &= f_T(u) + f_T(x) - f_{T'}(u) - f_{T'}(x) & (7)\\ &= (\deg_T^+(x) + a)^2 + (\deg_T^+(x))^2 - (\deg_T^+(x) + a - 1)^2 - (\deg_T^+(x) + 1)^2 & (8)\\ &= 2a - 2, \end{aligned}$$

The equality (7) follows from the fact that only the degrees of u and x differ in T and T', and (8) has been obtained by substituting $\deg_T^+(u) = \deg_T^+(x) + a$ and using the fact that $\deg_{T'}^+(x) = \deg_T^+(x) + 1$. By the choice of u and x we have that $a \geq 2$. So, after each improvement step the value of F decreases at least by 2. Initially,

$$F(T) = \sum_{v \in V(T)} (\deg_T^+(v))^2 \leq \left(\sum_{v \in V(T)} \deg_T^+(v) \right)^2 = O(n^2).$$

Thus, the number of improvement steps is bounded by $O(n^2)$.

Lemma 8. *The running time of the minimum degree spanning tree algorithm for a dag D with one target is $O(\Delta(D)n^2)$.*

Proof. Assume that for each vertex $v \in V(D)$ the set $N_D^-(v)$ is stored in such a way that a list $l_v(i)$, $i \in \{1, \dots, n-1\}$, contains all the vertices $u \in N_D^-(v)$ with $\deg_T^+(u) = i$. The algorithm uses a set A_i of all active vertices at the beginning of the ith improvement step. Observe that in this way we can find an active vertex in constant time. Since the running time of the procedure Improvement is $O(\Delta(D))$ and there are $O(n^2)$ calls of this procedure, it is sufficient to show that given A_{i-1} we can compute A_i and correct the lists $l_y(i)$ for each $y \in V(D)$ in $O(\Delta(D))$ time when an arc (v, u) has been removed from $E(T)$ and (v, x) has been added to $E(T)$. Let $y \in V(D)$. In the following $\deg_T^+(z)$, $z \in V(D)$, refers to the indegree of z after the ith improvement step. If $y \in N_D^+(x)$ then we move the vertex x from $l_y(\deg_T^+(x) - 1)$ to $l_y(\deg_T^+(x))$. If $y \in N_D^+(u)$ then the vertex u should be moved from $l_y(\deg_T^+(u) + 1)$ to $l_y(\deg_T^+(u))$. There are $O(\Delta(D))$ vertices y considered above and each modification of l_y can be done in $\Theta(1)$ time. Observe that if $l_y(i), l_y(j) \neq \emptyset$ and $l_y(i') = \emptyset$ for each $i' < i$, $l_y(j') = \emptyset$ for each $j' > j$ then y is active if and only if $j - i > 1$. This means that we can check in constant time if a vertex in $N_D^+(x) \cup N_D^+(u)$ should belong to A_i. If $y \notin N_D^+(x) \cup N_D^+(u)$ then the values of $\deg_T^+$ for the vertices in $N_D^-(y)$ do not change after the improvement step, which implies that $y \in A_i$ if and only if $y \in A_{i-1}$. $\square$

Theorem 4 ([6]). *If T is a branching computed by the minimum degree spanning tree algorithm then*

$$\Delta(G(T)) \leq (1 + \epsilon)\widetilde{\Delta}^* + \lceil \log_{1+\epsilon} n \rceil$$

for any $\epsilon > 0$. $\square$

In the following we will use the above result with $\epsilon = 1$, i.e. $\Delta(G(T)) = O(\widetilde{\Delta}^* + \log n)$.

6. An approximate algorithm for finding search strategy

Assume that a dag with one target D is given. Recall that we used the symbols T^* and $\widetilde{T}^*$ to denote a minimum edge ranking branching and a minimum degree branching of D, respectively. Define $\Delta^* = \Delta(T^*)$. Let A^* be an optimal search strategy for D. By Corollary 1 we have that $w(D) = w(A^*) = \chi_r'(G(T^*))$.

We have already described an algorithm for finding low degree branchings of D. Now let us give an approximate algorithm for finding a search strategy for D.

Step 1. Use the algorithm DMDST (described in Section 5) to find a branching T of D;
Step 2. Find an optimal edge ranking c of $G(T)$;

Step 3. Use c to create a search strategy A for D;

Observe that a search strategy A can be created in the way described in the proof of Lemma 2. This, in particular, means that $w(A) = \chi'_r(G(T))$. For brevity we write in the following Δ instead of $\Delta(G(T))$.

We will use the following upper bound for the edge ranking number of a tree.

Lemma 9 ([2]). *For each branching T with $n > 2$ vertices we have $\chi'_r(G(T)) \leq \Delta \log_\Delta n$.* $\square$

We have $\chi'_r(G(T)) \geq \Delta$. Since the edge e with the biggest color in each edge ranking of $G(T)$ is unique and $G(T) - \{e\}$ has exactly two connected components we can prove (using a simple induction on the number of vertices of the tree $G(T)$) that $\chi'_r(G(T)) \geq \log_2 n$. So,

$$\chi'_r(G(T)) \geq \frac{1}{2}(\Delta + \log_2 n). \tag{9}$$

Clearly, $\widetilde{\Delta}^* \leq \Delta^*$. We have

$$\begin{aligned}
\frac{w(A)}{w(A^*)} = \frac{\chi'_r(G(T))}{\chi'_r(G(T^*))} &\leq 2\frac{\Delta \log_\Delta n}{\Delta^* + \log_2 n} && (10)\\
&= O\left(\frac{(\widetilde{\Delta}^* + \log_2 n)\log_{\widetilde{\Delta}^*+\log_2 n} n}{\Delta^* + \log_2 n}\right) && (11)\\
&= O\left(\frac{(\Delta^* + \log_2 n)\log_{\Delta^*+\log_2 n} n}{\Delta^* + \log_2 n}\right) && (12)\\
&= O\left(\frac{\log n}{\log(\Delta^* + \log n)}\right) = O\left(\frac{\log n}{\log(\log n)}\right).
\end{aligned}$$

The inequality (10) follows from (9) applied to T^* and Lemma 9 applied to T. Theorem 4 implies (11) while the equality (12) follows from $\widetilde{\Delta}^* \leq \Delta^*$. In order to obtain (11) and (12) we have also used the formula $a \log_a x \geq b \log_b x$ for $a \geq b \geq 1$, $x \geq 1$.

The most time consuming step of the above algorithm is Step 1. So, the above analysis and Lemma 8 give the following.

Theorem 5. *There exists an $O(\log n / \log(\log n))$-approximate algorithm for finding search strategy in a poset with maximum element. The running time of the algorithm is $O(\Delta(D)n^2)$.* $\square$

7. Conclusions

The algorithm for finding search strategy given in this paper computes a low degree branching of a dag D and then an optimal edge ranking of the corresponding simple tree. The edge ranking gives a search strategy. While the second step can be solved optimally, we use an approximate algorithm for the first step. However, from the analysis of the algorithm we can conclude that a better algorithm for finding minimum degree branching does not allow us to improve in general the final approximation ratio, i.e. the ratio would be asymptotically identical if we could efficiently compute a minimum degree branching in the cases when $\widetilde{\Delta}^* = \Omega(\log n)$. An interesting problem is to derive an algorithm with a better approximation ratio for the searching problem.

References

[1] Y. Ben-Asher, E. Farchi, I. Newman, Optimal search in trees, SIAM J. Comput. 28 (1999) 2090–2102.
[2] D. Dereniowski, M. Kubale, Efficient parallel query processing by graph ranking, Fundam. Inform. 69 (2006) 273–285.
[3] D. Dereniowski, Edge ranking of weighted trees, Discrete Appl. Math. 154 (2006) 1198–1209.
[4] A.V. Iyer, H.D. Ratliff, G. Vijayan, Parallel assembly of modular products — an analysis, Tech. Report 88-06, Georgia Institute of Technology, 1988.
[5] R.M. Karp, Reducibility among combinatorial problems, in: R.E. Miller, J.W. Thathcher (Eds.), Complexity of Computer Computations, Plenum Press, New York, 1972, pp. 85–103.
[6] P.N. Klein, R. Krishnan, B. Raghavachari, R. Ravi, Approximation algorithms for finding low-degree subgraphs, Networks 44 (2004) 203–215.
[7] T.W. Lam, F.L. Yue, Edge ranking of graphs is hard, Discrete Appl. Math. 85 (1998) 71–86.
[8] T.W. Lam, F.L. Yue, Optimal edge ranking of trees in linear time, in: Proc. of the 9th Annual ACM-SIAM Symposium on Discrete Algorithms, 1998, pp. 436–445.
[9] N. Linial, M. Saks, Searching ordered structures, J. Algorithms 6 (1985) 86–103.
[10] M.J. Lipman, J. Abrahams, Minimum average cost testing for partially ordered components, IEEE Trans. Inform. Theory 41 (1995) 287–291.
[11] K. Makino, Y. Uno, T. Ibaraki, On minimum edge ranking spanning trees, J. Algorithms 38 (2001) 411–437.