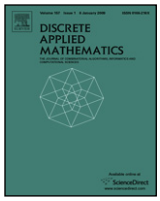
Discrete Applied Mathematics 158 (2010) 1624–1632

Contents lists available at ScienceDirect

Discrete Applied Mathematics

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



Finding monotone paths in edge-ordered graphs

J. Katrenič *, G. Semanišin

Institute of Computer Science, P.J. Šafárik University, Faculty of Science, Jesenná 5, 041 54 Košice, Slovak Republic

ARTICLE INFO

Article history:
Received 21 February 2009
Received in revised form 17 May 2010
Accepted 28 May 2010
Available online 25 June 2010

Keywords:
Ascent
Monotone path
NP-completeness
Complexity
Edge-ordering
k-path

ABSTRACT

An *edge-ordering* of a graph $G = (V, E)$ is a one-to-one function f from E to a subset of the set of positive integers. A path P in G is called an *f-ascent* if f increases along the edge sequence of P. The *height* $h(f)$ of f is the maximum length of an f-ascent in G.

In this paper we deal with computational problems concerning finding ascents in graphs. We prove that for a given edge-ordering f of a graph G the problem of determining the value of $h(f)$ is *NP*-hard. In particular, the problem of deciding whether there is an f-ascent containing all the vertices of G is *NP*-complete. We also study several variants of this problem, discuss randomized and deterministic approaches and provide an algorithm for the finding of ascents of order at least k in graphs of order n in running time $O(4^k n^{O(1)})$.

1. Introduction and motivation

In this paper we consider only finite non-oriented graphs without loops and multiple edges. If there is no danger of confusion we use n and m for the order and the size of a graph, respectively. In general, we use standard graph and computational complexity terminology and notation.

A one-to-one mapping f from E to the set of positive integers is called an *edge-ordering* of a graph $G = (V, E)$. For $e \in E$, we call $f(e)$ the *label* of e. A simple path of G for which f increases along the edge sequence is called an *f-ascent* of G, and a *(k, f)-ascent* if it has length k. The *height* $h(f)$ of f is the maximum length of an *f-ascent*.

Denote the set of all edge-orderings of G by $\mathcal{F}$. A graph theoretical invariant *altitude* of a graph G is defined in the following way:

$$\alpha(G) = \min_{f \in \mathcal{F}} h(f).$$

This invariant was first studied by Chvátal and Komlós in [6]. Some general bounds of $\alpha(G)$ were obtained by Graham and Kleitman in [12]. A survey of some known results on $\alpha(G)$ can be found e.g. in [3,4,8]. Burger et al. [3] presented an algorithm that was used for determining the altitude of the complete graphs of order 7 and 8. On the other hand, $\alpha(K_n)$ is still unknown for $n \geq 9$.

Later on, the concept of the *depression* of a graph was studied in [7,11,18]. An f-ascent is called *maximal* if it is not contained in a longer f-ascent. The depression $\epsilon(G)$ of G is the least integer k such that every edge-ordering of G has a maximal ascent of length at most k.

Behind the theoretical applications, a monotone path may appear in many practical situations like the following: Let the vertices represent business executives and the edges their mutual communication concerning a given commodity. An

* Corresponding author. Tel.: +421 904 393 104; fax: +421 55 6220949.
E-mail addresses: jan.katrenic@upjs.sk (J. Katrenič), gabriel.semanisin@upjs.sk (G. Semanišin).

0166-218X/$ – see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.dam.2010.05.018

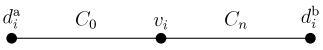


Fig. 1. A structure that is joined to a vertex v_i of the original graph.

owner of the commodity wants to sell it if he or she obtains more money than was paid for the commodity to its previous owner. In such a case the trajectory of the commodity forms a monotone path.

In this paper we deal with several variants of the problem of finding monotone paths under some edge-ordering. In the first section, we show that the problem of determining the value $h(f)$ for a given graph G and an edge-ordering f is *NP*-hard. Furthermore, we prove *NP*-completeness of finding ascents in complete graphs, finding an ascent containing all n vertices, or finding an ascent of length n^ϵ, for an arbitrary fixed positive $\epsilon < 1$. Moreover, we prove that no constant factor polynomial time approximation algorithm is possible for the longest monotone path problem unless $P = NP$.

In the last section, we apply randomized methods used for the k-path problem, defined in the following way: *Given a graph G and an integer k, either construct a simple path of k vertices in G or report that no such path exists.*

The k-path problem has been studied rather intensively. Earlier algorithms [2,17] for the k-path problem have running time bounded by $O(2^k k! n^{O(1)})$. Papadimitriou and Yannakakis [20] studied a restricted version of the problem, the $(\log n)$-path problem, and conjectured that the $(\log n)$-path problem can be solved in polynomial time. This conjecture was confirmed by Alon et al. [1], who presented a new technique of color-coding that provides a randomized algorithm for the k-path problem in running time $O(5.44^k n^{O(1)})$. We show how to improve this technique to reach running time $O(4.31^k n^{O(1)})$.

Recently, a new randomized divide and conquer algorithm for the k-path problem was introduced [13,15], that solves the k-path problem in running time $O(4^k n^{O(1)})$. In the last section, we apply this technique for the k-ascent problem to obtain a randomized algorithm with running time $O(4^k n^{O(1)})$.

2. *NP*-completeness of the ascent finding problem

In this section we deal with the following problem:

Problem 1 (k-*ascent Problem*)**.** Consider a graph G, a positive integer k and an edge-ordering f of G. Is there any (k, f)-ascent in G?

It is easy to see that this problem belongs to the class *NP*, because there exists a non-deterministic algorithm that can choose k edges in a non-deterministic way and then polynomially check whether they form an ascent of G.

In what follows we shall prove that the k-path problem (a well known *NP*-hard problem) can be polynomially reduced to Problem 1. More precisely, we shall show that for a given graph G and number k one can construct a new graph G' with an edge-ordering f and a number k' such that G contains a simple path of k vertices if and only if G' has a (k', f)-ascent.

In order to simplify our considerations we use the following notation that was already introduced in [3,5,10,12,21,22].

Definition 1. Let $\mathbf{P} = (E_1, \ldots, E_t)$ be an ordered partition of the edge set E of G and let f be any edge-ordering of G satisfying

$$e_1 \in E_i \quad \text{and} \quad e_2 \in E_j, \quad \text{where } i < j, \text{ implies } f(e_1) < f(e_2).$$

Such an edge-ordering is called **P**-*consistent*.

For $i = 1, \ldots, t$ we denote by f_i the restriction of the mapping f to E_i and by G_i the subgraph of G induced by the edge set E_i. It is obvious that in such a case f_i is an edge-ordering of G_i. Moreover, if $\mathbf{P} = (E_1, \ldots, E_t)$ is an ordered partition of E, then in the edge sequence X of an f-ascent of G the edges from E_i precede the edges from E_j, for each pair (i, j) with $i < j$. Hence an ascent X can be written as $X = X_1, \ldots, X_t$, where X_i (possibly empty) is an f_i-ascent of G_i.

Now we describe the construction that will be used in the proof of the main result of this section. Denote by n the number of vertices in a given graph G. Let $V(G) = \{v_1, \ldots, v_n\}$. Let us put $k' = 2k$. We shall construct a **P**-consistent edge-ordering

$$\mathbf{P} = (C_0, C_1^a, C_1^b, C_2^a, C_2^b, \ldots, C_{n-1}^a, C_{n-1}^b, C_n).$$

We begin by describing how to construct the new graph G' and the edge-ordering f from G. Firstly, we join two new vertices d_i^a and d_i^b to each vertex v_i of G, for each $i \in \{1, \ldots, n\}$ (see Fig. 1). This means that we add $2n$ new vertices of degree 1.

Now we have to assign suitable labels to the new edges. We shall use the labels from some sets $C_0, C_1^a, C_1^b, C_2^a, C_2^b, \ldots, C_{n-1}^a, C_{n-1}^b, C_n$. One can easily see that we can make them sufficiently large in order to obtain the labeling described as follows. For the new edges $v_i d_i^a$ and $v_i d_i^b$ we choose labels such that $v_i d_i^a \in C_0$ and $v_i d_i^b \in C_n$, respectively.

Secondly, we remove the original edges $v_i v_j$, for $i < j$, and add $2n-2$ new vertices $h_{i,j,l}$, $h_{j,i,l}$, for $l = 1, \ldots, n-1$, together with $4n-4$ new edges of the form $v_i h_{i,j,l}$, $v_i h_{j,i,l}$, $v_j h_{i,j,l}$, $v_j h_{j,i,l}$, for $l = 1, \ldots, n-1$ (see Fig. 2). The new edges receive labels according to the following scheme:

$$v_i h_{i,j,l} \in C_l^a \quad v_i h_{j,i,l} \in C_l^b$$
$$v_j h_{i,j,l} \in C_l^b \quad v_j h_{j,i,l} \in C_l^a.$$

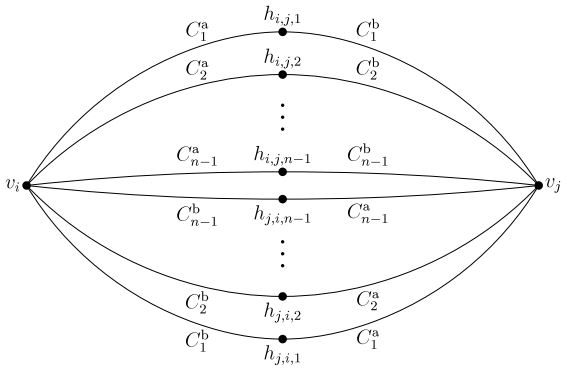


Fig. 2. A structure that is inserted instead of the original edge v_iv_j.

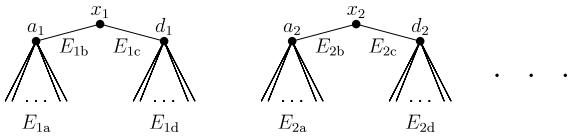


Fig. 3. Structures added in the construction.

The new graph G' contains $n + 2n + m(2n-2) = 3n + 2nm - 2m$ vertices $\{v_1, \ldots, v_n, d_1^{\mathrm{a}}, \ldots, d_n^{\mathrm{a}}, d_1^{\mathrm{b}}, \ldots, d_n^{\mathrm{b}}\} \cup \{h_{i,j,l} : v_iv_j \in E(G), l \in \{1, \ldots, n-1\}\}$ and the edge set of G' has $m(4n-4) + 2n$ edges.

We shall see that each vertex $h_{i,j,l}$ has a strong influence on the existence of ascents running through v_i and v_j and containing edges from $C_l^{\mathrm{a}} \cup C_l^{\mathrm{b}}$.

Now we verify the correctness of our construction.

Theorem 1. *The graph G contains a simple path of k vertices if and only if G' has a $(2k, f)$-ascent.*

Proof. $\Rightarrow$ Let $W = (v_{i_1}, v_{i_2}, \ldots, v_{i_k})$ be a simple path in G. Then it is easy to verify that

$$\lambda = (d_{i_1}^{\mathrm{a}}, v_{i_1}, h_{i_1,i_2,1}, v_{i_2}, h_{i_2,i_3,2}, v_{i_3}, \ldots, v_{i_{k-1}}, h_{i_{k-1},i_k,k-1}, v_{i_k}, d_{i_k}^{\mathrm{b}})$$

is a $(2k, f)$-ascent in G'.

$\Leftarrow$ Observe first that the graph G' is a bipartite graph such that the first partition consists of the original vertices of the type "v" and the second partition consists of the new vertices of the types "d" and "h".

Since all vertices of the type "d" are of degree 1, we know that all internal vertices of any path in the graph G' are either of type "v" or of type "h" and moreover they alternate within the vertex sequence of the path.

Consider now any $(2k, f)$-ascent λ in the graph G'. Clearly, λ contains at least k vertices of type "v". Denote those vertices along λ as $v_{i_1}, v_{i_2}, \ldots, v_{i_k}$. Now, λ can be written in the form $\lambda = \{\ldots, v_{i_1}, \text{"}h\text{"}, v_{i_2}, \text{"}h\text{"}, v_{i_3}, \ldots, \text{"}h\text{"}, v_{i_{k-1}}, \text{"}h\text{"}, v_{i_k}, \ldots\}$. Then one can easily see that vertices $v_{i_1}, v_{i_2}, \ldots, v_{i_k}$ form a simple path of k vertices in the original graph G. □

Furthermore, besides the decision form of the problem, we shall see that our construction of G' from the original graph G has the following property.

Theorem 2. *There is a polynomial time algorithm, which takes a path of k vertices in G and constructs an $(2k, f)$-ascent in G' and vice versa.*

3. The problem of finding complete ascents

Now we shall deal with a related problem:

Problem 2 (*Complete Ascent Finding Problem*)**.** Given a graph G and an edge-ordering f of G, determine whether there is an f-ascent containing all vertices of G.

Clearly, this problem is a special case of Problem 1; hence it belongs to the class *NP*. In order to prove its *NP*-completeness, we provide a reduction from the original Problem 1 to this one. From a given triple (G, f, k) we produce a graph G' and an edge-ordering f' such that G contains a (k, f)-ascent if and only if G' contains a $(|V(G')|, f')$-ascent. Graph G' will be constructed from the graph G by using the structures presented in Fig. 3.

The resulting graph G' contains G as a subgraph. The labels of the edges of this subgraph remain the same as in the original edge-ordering f of G.

We shall construct a **P**-consistent edge-ordering f':

$$\mathbf{P} = (E_G, E_{1a}, E_{1b}, E_{1c}, E_{1d}, E_{2a}, E_{2b}, E_{2c}, E_{2d}, \ldots, E_{(n-k-1)d})$$

where E_G contains exactly the edges from the original graph G. The other sets will contain new edges of G' that will be described below.

Besides the original vertices of G, G' will contain $3(n-k-1)$ new vertices $a_1, a_2, \ldots, a_{n-k-1}$, $d_1, d_2, \ldots, d_{n-k-1}$ and $x_1, x_2, \ldots, x_{n-k-1}$.

Moreover, for each $i \in \{1, \ldots, n-k-1\}$ and for each vertex u of $V(G)$ we add new edges of the following form:

- a_iu—these edges will belong to the set E_{ia} of **P**;
- d_iu—these edges will belong to the set E_{id} of **P**;
- a_ix_i—these edges will belong to the set E_{ib} of **P**;
- d_ix_i—these edges will belong to the set E_{ic} of **P**.

Now we can prove the following result:

Theorem 3. *A graph G contains a (k, f)-ascent if and only if there is an f'-ascent in the graph G' containing all its vertices.*

Proof. $\Rightarrow$ Let λ be a (k, f)-ascent in the given graph G. Let us label the vertices of G that do not belong to λ by $v_{i_1}, v_{i_2}, \ldots, v_{i_{(n-k-1)}}$. Then

$$\lambda' = (\lambda, a_1, x_1, d_1, v_{i_1}, \ldots, a_j, x_j, d_j, v_{i_j}, \ldots, a_{(n-k-1)}, x_{(n-k-1)}, d_{(n-k-1)}, v_{i_{(n-k-1)}})$$

is an f'-ascent containing all the vertices of G'.

$\Leftarrow$ Assume now that λ' is an f'-ascent in the graph G' containing all its vertices. Since E_G is the first set in **P**, the ascent λ' can be written in the form $\lambda' = \lambda_1, \lambda_2$, where λ_1 contains only the edges from the original graph G and λ_2 contains only the added edges. Moreover note that λ_1 is an f-ascent in the original graph G.

Now we need to prove that λ_2 does not contain more than $n-k$ vertices from the original graph G (in such a case λ_1 must contain at least $k+1$ vertices of G). The ascent λ_2 must contain all the new vertices, and in particular all the vertices from the set $\{x_1, x_2, \ldots, x_{(n-k-1)}\}$. According to the edge-ordering f', it is not difficult to see that the order of these vertices must be the same in λ_2, i.e.

$$\lambda_2 = \ldots, x_1, \ldots, x_2, \ldots, x_{(n-k-1)}, \ldots.$$

Since λ_2 must contain all new vertices from the graphs presented in the Fig. 3, no vertex x_i can be the last vertex of λ_2. Hence each vertex x_i is an internal vertex and the vertices a_i and d_i must directly precede and succeed the vertex x_i in the sequence of vertices of λ_2 respectively. Therefore

$$\lambda_2 = \ldots, a_1, x_1, d_1, \ldots, a_2, x_2, d_2, \ldots, a_{(n-k-1)}, x_{(n-k-1)}, d_{(n-k-1)}, \ldots.$$

As λ_2 does not contain any edge of type v_iv_j, λ_2 contains at most $n-k$ vertices from the original graph. This completes the proof. □

Problem 2, which concerns the existence of ascents containing all vertices, is a special case of Problem 1, where $k = n-1$. Similarly, according to the *NP*-completeness of Problem 2 it is easy to formulate new *NP*-complete problems, for any fixed positive integer k:

Problem 3. Given a graph G with n vertices and an edge-ordering f of G, determine whether there is an f-ascent containing $n-k$ vertices.

Problem 4. Given a graph G with n vertices and an edge-ordering f of G, determine whether there is an f ascent containing $\sqrt[k]{n}$ vertices.

In both cases, a proof of the *NP*-completeness lies in the reduction of Problem 2 to Problem 3 or Problem 4, respectively, by adding an appropriate number of new isolated vertices to the original graph. Consequently, for an arbitrary fixed positive $\epsilon < 1$ the following problem is *NP*-complete, as well:

Problem 5. Given a graph G of n vertices and an edge-ordering f of G, determine whether there is an f-ascent containing $n^{1-\epsilon}$ vertices.

Some negative results for the related k-path problem can be applied to the k-ascent problem, as well. Karger et al. [14] proved that, for any $\epsilon < 1$, the problem of finding a path of length $n - n^{\epsilon}$ in an n-vertex Hamiltonian graph is *NP*-hard. This implies that no constant factor approximation algorithm is possible for the longest path problem unless $P = NP$.

In order to prove a similar result for the k-ascent problem, one can easily use the reduction described in the proofs of Theorems 1 and 2 and the result of [14]:

Theorem 4. *For any $\epsilon < 1$, the problem of finding an ascent of length $n - n^{\epsilon}$ for a given edge-ordering f and an n-vertex graph containing an $(n-1, f)$-ascent is NP-hard.*

This also directly implies that there exists no constant factor approximation algorithm for the longest ascent problem unless $P = NP$.

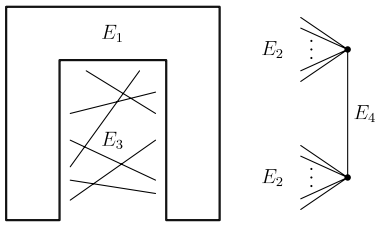


Fig. 4. A reduction of Problem 2 to Problem 6.

4. Finding ascents in complete graphs

In this section we shall consider the following problem that provides a modification of the previous ones. In contrast with the case for the previous problems we shall restrict our attention to a particular class of graphs.

Problem 6 (*Problem of Finding Ascents with a Prescribed Edge in Complete Graphs*)**.** Given an edge-ordering f of a complete graph G and one edge e, determine whether there is an f-ascent containing the edge e and all vertices of G.

Fig. 4 shows the principle of a reduction of Problem 2 to this problem.

In the reduction we shall construct an (E_1, E_2, E_3, E_4)-consistent edge-ordering. The new graph is constructed from the original one by adding two new vertices and all possible edges in such a way that the resulting graph is complete. The labels of the original edges remain unchanged and these edges belong to the set E_1. The new edges that are added between original vertices belong to the set E_3. The edge that connects two new vertices belongs to E_4. Finally, all the remaining edges (that were added between original vertices and new vertices) belong to E_2. The edge belonging to E_4 will be chosen to be the edge e from the problem formulation.

Let us denote by λ an ascent containing all the vertices and the edge e in E_4. Clearly, the edge $e \in E_4$ must appear as the last edge in λ. Thus only one edge of E_2 can be used in λ, and it must precede the last edge. Since we have used a consistent edge-ordering, λ can contain no edge of E_3. Summarizing these facts we have that λ can contain only edges of E_1 belonging to the original graph followed by one edge of E_2 and the edge from E_4.

Theorem 5. *The problem of finding ascents with a prescribed edge in complete graphs belongs to the class NP-complete problems.*

5. An estimation of the complexity of a brute-force like algorithm

Let us denote by $\Delta(f)$ the number of all f-ascents in the graph G. It is easy to construct an algorithm for solving Problem 2 or Problem 3 with the time complexity $O(n\Delta(f))$. Now we shall establish an upper bound for the value of $\Delta(f)$. We shall assume that f is a function from E to the set $\{1, 2, \ldots, m\}$ only. Denote the set of all such edge-orderings of G by $\mathcal{F}_G$. Clearly, the number of such edge-orderings is $m!$. One can rather easily see that all other edge-orderings of G can be transformed to this form.

Lemma 1. *Let $\lambda = \{v_{i_1}, v_{i_2}, \ldots, v_{i_k}\}$ be a simple path of k vertices in a graph G. Then there are $\frac{m!}{k!}$ edge-orderings such that λ is an ascent in G.*

The following result provides an upper bound for the total number of ascents among all $m!$ edge-orderings.

Lemma 2. *Let G be a graph with n vertices and m edges. Then*

$$\sum_{f \in \mathcal{F}_G} \Delta(f) \leq m! n 2^n.$$

Proof. Observe first that the number of paths of length k in G is at most

$$\binom{n}{k+1} (k+1)!.$$

According to the previous lemma, there are at most

$$\binom{n}{k+1} (k+1)! \frac{m!}{k!} = \binom{n}{k+1} (k+1) m!$$

ascents of length k among all edge-orderings of G. Using this expression we get that in the graph G there are at most

$$\sum_{k=0}^{n-1} \binom{n}{k+1} (k+1) m! = m! \sum_{k=1}^{n} \binom{n}{k} k \leq m! n 2^n$$

different ascents among all edge-orderings of $\mathcal{F}_G$. □

According to Lemma 2, in the average case there are at most $n2^n$ ascents in a graph G of order n. So the average time (if the edge-ordering is chosen arbitrarily), the time complexity for an algorithm that searches for all ascents in G, is $O(2^n n^2)$, while the space complexity is polynomial. This improves the space complexity of a straightforward dynamic programming approach based on the consideration of each subset of vertices, which works in time and space $O(2^n n^2)$.

6. A color-coding scheme for the ascent problem

In the rest of this paper, we study algorithms for finding a k-ascent in a graph of order n, for relatively small values of k. As we have already proved, the problem is *NP*-hard if we are looking for an ascent consisting of n vertices, or if we are looking for an ascent with $\sqrt[k]{n}$ vertices. In what follows, we introduce and apply some techniques designed for the k-path problem.

A well known randomized algorithm for the k-path problem (see e.g. [1,16]) is based on the fact that the problem can be solved effectively (in linear time) if the input graph is *directed acyclic graph* (DAG). A polynomial method for solving the k-path problem in a DAG uses topological sorting and linear dynamic programming [9]. The probabilistic algorithm based on this idea consists of two steps. Firstly, we randomly sort vertices by choosing a random permutation $\pi : V \to \{1, \ldots, |V|\}$ and directing an edge $(u, v) \in E$ from u to v iff $\pi(u) < \pi(v)$. Secondly, we find a longest path in the DAG created. The probability of success is equal to the probability that vertices of a path of length k will stay in the same order after the random sorting, which is $1/k!$. Finally, after $k!$ repetitions of this algorithm, we will obtain a randomized Monte Carlo algorithm for the k-path problem in running time $O(k!n^{O(1)})$, which is polynomial for $k = \log n/\log\log n$. It is easy to see how to exploit the approach described above for the k-ascent problem as well.

Alon et al. [1] presented a randomized and deterministic algorithm in running time $O(2^{O(k)}n^{O(1)})$ that proved a conjecture that a $\log n$-path problem has a polynomial solution [20]. The technique of *color-coding* introduced in [1] solves the k-path problem as follows. We say that a path in a graph G is *properly colored* under a coloring of the vertices in G if no two vertices on the path are colored with the same color. Now suppose that there is a path λ of k vertices. To find a k-path, we firstly color the vertices of the graph G using k colors so that λ is properly colored. Then we can use a (deterministic) dynamic programming algorithm, which finds a longest properly colored path (in running time $O(2^k n^{O(1)})$).

The critical step of this algorithm is how one constructs a coloring for the graph G so that the path λ is properly colored. Alon et al. [1] proposed two approaches to this problem. The first is a randomized algorithm of running time $O(e^k)$. The idea is to produce a random k-coloring, i.e. to choose a random color for each vertex of G. The probability p that a given path of k vertices will remain properly colored is

$$p = \frac{k}{k} \cdot \frac{k-1}{k} \cdot \frac{k-2}{k} \cdot \cdots \cdot \frac{1}{k} = \frac{k!}{k^k} \approx \left(\frac{1}{e}\right)^k.$$

After repeating this step e^k times, the probability of success is greater than $1/2$, which leads to a probabilistic algorithm in running time $O(5.44^k n^{O(1)})$. The second deterministic approach for the k-path problem described in [1] runs in time $O((2c)^k n^{O(1)})$, where the constant c is over 4000.

Recently, this color-coding scheme has been derandomized via the following theorem [13]:

Theorem 6. *For any integers n and k, where $n \geq k$, an (n, k)-family of perfect hashing functions of size $O(6.4^k \log^2 n)$ can be constructed in time $O(6.4^k n \log^2 n)$.*

In other words, [13] provides a method for creating a set of k-colorings of size $O(6.4^k \log^2 n)$ such that for each subset of k vertices there is at least one coloring where the k vertices are colored properly. This plus the dynamic programming algorithm for finding a longest properly colored path gives a deterministic algorithm for the k-path problem in running time $O(6.4^k 2^k n^{O(1)}) = O(12.8^k n^{O(1)})$.

The same approach can be adopted for a k-ascent problem as well. One can produce a random k-coloring for a given graph G and follow the same idea as in [1]. The only difference is that one has to implement a dynamic programming algorithm to find a longest properly colored ascent instead of a simple path, but it is still possible in running time $O(2^k n^{O(1)})$.

It is worthy of note that the randomized concept of color-coding can be improved by using more that k colors in the first stage of the algorithm. We propose the following algorithm in order to find a k-path (or k-ascent) in a given graph G:

1. Create a random $\lceil 1.3k \rceil$-coloring, i.e. assign a random color from the set $\{1, 2, \ldots, \lceil 1.3k \rceil\}$ to each vertex of the given graph G.
2. Find the longest properly colored path in G.

The probability p that a given path of k vertices will remain properly colored is

$$p = \frac{\lceil 1.3k \rceil}{k} \cdot \frac{\lceil 1.3k \rceil - 1}{k} \cdots \frac{\lceil 0.3k \rceil}{k} = \frac{\lceil 1.3k \rceil!}{k^k \cdot \lceil 0.3k \rceil!} \approx \left(\frac{1}{e}\left(\frac{1.3}{0.3}\right)^{0.3}\right)^k \approx 0.571^k.$$

Therefore, after repeating this algorithm 1.751^k times, the probability of success will be over $1/2$. A dynamic programming algorithm for finding the longest properly colored path in a $1.3k$-coloring works in running time $O(2^{1.3k} n^{O(1)})$. This gives us a new probabilistic algorithm in running time $O(1.751^k \cdot 2^{1.3k} \cdot n^{O(1)}) = O(4.312^k n^{O(1)})$, which directly improves on the original algorithm in running time $O(5.44^k n^{O(1)})$ mentioned in previous papers.

7. Randomized divide and conquer for k-ascent

Recently, the complexity upper bounds for the k-path problem were improved using a randomized divide and conquer algorithm [13,15] in running time $O(4^k k^{3.42} m)$ that is better than the previous best algorithms.

In what follows, we show how to apply this technique to the k-ascent problem. Let us say that an ascent λ_1 is *smaller* than the ascent λ_2 if the label of the last edge in λ_1 is smaller than the label of the last edge in λ_2. An ascent in G is a (u, k)-*ascent* if it contains exactly k edges and if the last vertex of the ascent is u. An ascent in G is called a (u, k, x)-*ascent* if it forms (u, k)-ascent and the label of the last edge is at most x,

Let p be a path ending at the vertex u and (u, v) be an edge in G. We denote by $p + (u, v)$ the path p elongated by the edge (u, v).

We find a $(k-1)$-ascent in the graph G and a fixed edge-ordering by calling the function **find-ascents** with parameters $(\emptyset, G, k)$. Theorem 7 claims that the probability of success (the algorithm finds at least one $(k-1)$-ascent under the assumption that one exists) is at least 0.5.

Function find-ascents(P', G', k)

Input: G' a subgraph of G, P' a set of k'-ascents in G that contains no vertex in G', an integer $k \geq 1$;
Output: a set P of ascents, each is a concatenation of a k'-ascent in P' and a $(k-1)$-ascent in G';

1. $P := \emptyset$;
2. **if** $k = 1$ **then**
   if $P' = \emptyset$ **then**
   return all 0-ascents in G';
   else
   for each (u, k')-ascent p in P' and each vertex v in G' **do**
   if (u, v) is an edge in G such that $p + (u, v)$ forms a $(v, k'+1)$-ascent **then**
   if P does not contains a smaller $(v, k'+1)$-ascent **then**
   erase from P all $(v, k'+1)$-ascents ;
   add $p + (u, v)$ to P ;
   return P;
3. **loop** $2.51 \cdot 2^k$ times **do**
3.1. randomly partition the vertices of G' into two parts V_L and V_R;
   denote by G_L and G_R the subgraphs induced by V_L and V_R, respectively;
3.2. $P_L :=$ **find-ascents**(P', G_L, $\lceil k/2 \rceil$);
   if $P_L \neq \emptyset$ **then**
3.3. $P_R :=$ **find-ascents**(P_L, G_R, $k - \lceil k/2 \rceil$);
   for each $(u, k' + k - 1)$-ascent p in P_R **do**
   if P does not contains a smaller $(u, k' + k - 1)$-ascent than p **then**
   erase from P all $(u, k' + k - 1)$-ascents ;
   add p to P' ;
4. **return** P ;

Theorem 7. *Let k be a positive integer and $G = (V, E)$ be a graph with n vertices and m edges. Let u be an arbitrary vertex of G. Then if G contains a $(u, k-1)$-ascent, with probability larger than $1 - 1/e > 0.632$, the set P returned by the algorithm* **find-ascents**$(\emptyset, G, k)$ *contains a $(u, k-1)$-ascent. The algorithm* **find-ascents**$(\emptyset, G, k)$ *runs in time $O(4^k k^{3.42} m)$ and in a polynomial space.*

Proof. To prove the first part, we verify the following claims using mathematical induction on k:

1. If $P = \emptyset$ and G' has a $(u, k-1, x)$-ascent, then with probability larger than $1 - 1/e$, the set P returned by the algorithm **find-ascents**(P', G, k) contains a $(u, k-1, x)$-ascent.
2. If $P' \neq \emptyset$ and G' has a $(u, k-1, x)$-ascent whose other end vertex can be connected to an end vertex of an ascent in P', then with probability larger than $1 - 1/e$, the set P returned by the algorithm **find-ascents** (P', G', k) contains a $(u, k' + k - 1, x)$-ascent.

The claims are obviously true for $k = 1$. Let $k > 1$. First consider the case when $P' = \emptyset$. Suppose that

$$[u_1, u_2, \ldots, u_{k_1}, u_{k_1+1}, \ldots, u_k]$$

is a $(u_k, k-1)$-ascent in G', where $k_1 = \lceil k/2 \rceil$. Then with probability $1/2^k$, in step 3.1 of the algorithm the vertices $u_1, u_2, \ldots, u_{k_1}$ are added into V_L, and vertices $u_{k_1+1}, \ldots, u_k$ into V_R. If this is the case, the graph G_L contains the $(u_{k_1}, k_1 - 1)$-ascent $[u_1, \ldots, u_{k_1}]$, and the graph G_R contains the $(u_k, k - k_1 - 1)$-ascent $[u_{k_1+1}, \ldots, u_k]$. By the inductive hypothesis, with probability larger than $1 - 1/e$, P_L obtained in step 3.2 includes a $(u_{k_1}, k_1 - 1, y)$-ascent, $y \leq f(u_{k_1-1}, u_{k_1})$. Such an ascent can be connected to the ascent $[u_{k_1+1}, \ldots, u_k]$. By the second inductive hypothesis, with probability larger than $1 - 1/e$, P_R

obtained in step 3.3 contains an ascent of length $(k_1 - 1) + 1 + (k - k_1 - 1) = k - 1$ that ends with u_k and its last edge has label at most $f(u_{k-1}, u_k)$. Therefore in each loop of step 3, the probability that a $(u_k, k-1, x)$-ascent is added to the set P is larger than

$$\frac{(1-1/e)^2}{2^k} > \frac{0.6322}{2^k} > \frac{1}{2.51 \cdot 2^k}.$$

Since step 4 of the algorithm loops $2.51 \cdot 2^k$ times, the overall probability that the algorithm returns a set of paths that contains a $(u_k, k-1, x)$-ascent (when $P' = \emptyset$) is

$$1 - \left(1 - \frac{1}{2.51 \cdot 2^k}\right)^{2.51 \cdot 2^k} > 1 - \frac{1}{e}.$$

In the case when $P' = \emptyset$, we follow the same argument as before except that we require that the $[u_1, \ldots, u_{k_1}, u_{k_1+1}, \ldots, u_k]$ ascent in G' may be connected to a k'-ascent in P'. So P_L contains a $(u_{k_1}, k' + k_1 - 1)$-ascent λ that is a concatenation of a k'-ascent in P' and a $(k_1 - 1)$-ascent in G_L, and P_R contains a $(u_k, k' + k - 1)$-ascent that is a concatenation of a $(k - k_1 - 1)$-ascent in G_R.

In order to analyze the time complexity, let us denote by $T(k)$ the running time of the algorithm **find-ascents**$(\emptyset, G, k)$. Without loss of generality, we can assume that $m \geq n$. From the algorithm, we get the following recurrence relation (assuming that the recursive function find-ascents is implemented in time $O(nm^2)$ without the time of recursive calls):

$$T(k) = 2.51 \cdot 2^k [cnm^2 + T(\lceil k/2 \rceil) + T(k - \lceil k/2 \rceil)],$$

where $c > 0$ is a constant. We claim that for all $k > 0$,

$$T(k) \leq c \cdot (10.7)^{\lceil \log k \rceil} 2^{2k} nm^2, \tag{1}$$

and we prove it by induction on k. Obviously $T(1) \leq cnm^2$ if c is sufficiently large, and the inequality (1) holds for $k = 1$. Let $k > 1$. Then $k - \lceil k/2 \rceil \leq \lceil k/2 \rceil$ and $2\lceil k/2 \rceil \leq k + 1$ and we have

$$\begin{aligned} T(k) &= 2.51 \cdot 2^k \left(cnm^2 + T(\lceil k/2 \rceil) + T(k - \lceil k/2 \rceil)\right) \\ &\leq 2.51 \cdot 2^k \left(cnm^2 + 2cnm^2 \cdot (10.7)^{\lceil \log \lceil k/2 \rceil \rceil} 2^{2\lceil k/2 \rceil}\right) \\ &\leq 2.51 \cdot 2^k \left(cnm^2 + \frac{2cnm^2}{10.7} \cdot (10.7)^{\lceil \log \lceil k/2 \rceil \rceil + 1} 2^{k+1}\right). \end{aligned}$$

Since $\lceil \log k \rceil = \lceil \log(\lceil \frac{k}{2} \rceil) \rceil + 1$ for $k > 1$, we can express the previous bound in the following form:

$$\begin{aligned} T(k) &\leq cnm^2 \cdot (10.7)^{\lceil \log k \rceil} 2^{2k} \cdot 2.51 \left(\frac{1}{10.7^{\lceil \log k \rceil} 2^k} + \frac{4}{10.7}\right) \\ &\leq cnm^2 \cdot (10.7)^{\lceil \log k \rceil} 2^{2^k} \cdot 2.51 \left(\frac{1}{10.7 \cdot 4} + \frac{4}{10.7}\right) \\ &< cnm^2 \cdot (10.7)^{\lceil \log k \rceil} 4^k. \end{aligned}$$

Therefore the overall running time of the algorithm **find-ascents**(P, G, k) is $O((10.7)^{\lceil \log k \rceil} 2^{2k} nm^2) = O(4^k k^{3.42} nm^2)$. □

A derandomization of this algorithm can be done by an approach used in [13]. In order to present it we need the following definition.

Definition 2. An (n, k)-universal set T is a set of n-bit binary strings such that for every subset π of k elements in $\{1, 2, \ldots, n\}$, the collection $\{\pi(s) | s \in T\}$ contains all $2^k k$-bit binary strings. The *size* of the (n, k)-universal set T is the number of strings in T.

Proposition 1 (*[19]*)**.** *There is a deterministic algorithm of running time* $O(2^k k^{O(\log k)} n \log n)$ *that constructs an* (n, k)*-universal set of size* $2^k k^{O(\log k)} \log n$.

Proposition 1 can be used to achieve a deterministic algorithm. Step 3 of our algorithm can be replaced in such a way that instead of choosing partitions V_L and V_R randomly $2.51 \cdot 2^k$ times, we use the (n, k) universal set of size $2^k k^{O(\log k)} \log n$. Using induction on k, we can prove that this deterministic algorithm correctly returns a $(k-1)$-ascent if such an ascent exists for the input graph and edge-ordering. The running time of this deterministic algorithm is bounded by $O(4^{k+o(k)} n^{O(1)})$.

Acknowledgement

The research of the first author was supported in part by Slovak VEGA grant 1/0035/09. The research of the second author was supported in part by Slovak VEGA grant 1/0035/09, APVV-0007-07 and by the Agency of the Slovak Ministry of Education for the Structural Funds of the EU, under project ITMS:26220120007.

References

[1] N. Alon, R. Yuster, U. Zwick, Color-coding, J. ACM 42 (1995) 844–856. http://doi.acm.org/10.1145/210332.210337.
[2] H. Bodlaender, On linear time minor tests with depth-first search, J. Algorithms 14 (1993) 1–23.
[3] A.P. Burger, E.J. Cockayne, C.M. Mynhardt, Altitude of small complete and complete bipartite graphs, Australas. J. Combin. 31 (2005) 167–177.
[4] A.P. Burger, C.M. Mynhardt, T.C. Clark, B. Falvai, N.D.R. Henderson, Altitude of regular graphs with girth at least five, Discrete Math. 294 (2005) 241–257.
[5] A.R. Calderbank, F.R.K. Chung, D.G. Sturtevant, Increasing sequences with nonzero block sums and increasing paths in edge-ordered graphs, Discrete Math. 50 (1984) 15–28.
[6] V. Chvátal, J. Komlós, Some combinatorial theorems on monotonicity, Canad. Math. Bull. 14 (1971) 151–157.
[7] E.J. Cockayne, G. Geldenhuys, P.J.P. Grobler, C.M. Mynhardt, J.H. Van Vuuren, The depression of a graph, Util. Math. 69 (2006) 143–160.
[8] E.J. Cockayne, C.M. Mynhardt, Altitude of $K_{3,n}$, J. Combin. Math. Combin. Comput. 52 (2005) 143–157.
[9] T.H. Cormen, C.E. Leiserson, R.L. Rivest, Introduction to Algorithms, The MIT Press, 1990.
[10] T. Dzido, H. Furmańczyk, Altitude of wheels and wheel-like graphs, Cent. Eur. J. Math. 8 (2010) 319–326.
[11] I. Gaber-Rosenblum, Y. Roditty, The depression of a graph and the diameter of its line graph, Discrete Math. 309 (2009) 1774–1778.
[12] R.L. Graham, D.J. Kleitman, Increasing paths in edge ordered graphs, Period. Math. Hungar. 3 (1973) 141–148.
[13] J. Chen, S. Lu, S.-H. Sze, F. Zhang, Improved algorithms for path, matching, and packing problems, in: SODA'07: Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms, 2007, pp. 298–307.
[14] D. Karger, R. Motwani, G.D.S. Ramkumar, On approximating the longest path in a graph, Algorithmica 18 (1997) 82–98.
[15] J. Kneis, D. Molle, S. Richter, P. Rossmanith, Divide-and-color, Lecture Notes in Comput. Sci. 4271 (2006) 58–67.
[16] Y. Kortsarts, J. Rufinus, How (and why) to introduce Monte Carlo randomized algorithms into a basic algorithms course? J. Comput. Sci. Coll. 21 (2005) 195–203.
[17] B. Monien, How to find long paths efficiently, Ann. Discrete Math. 25 (1985) 239–254.
[18] C.M. Mynhardt, Trees with depression three, Discrete Math. 308 (2008) 855–864.
[19] M. Naor, L.J. Schulman, A. Srinivasan, Splitters and near-optimal derandomization, in: IEEE Symposium on Foundations of Computer Science, FOCS 1995, pp. 182–191.
[20] C. Papadimitriou, M. Yannakakis, On limited nondeterminism and the complexity of the V–C dimension, in: Structure in Complexity Theory Conference, 1993, Proceedings of the Eighth Annual Volume, 1993, pp. 12–18.
[21] Y. Roditty, B. Shoham, R. Yuster, Monotone paths in edge-ordered sparse graphs, Discrete Math. 226 (2001) 411–417.
[22] R. Yuster, Large monotone paths in graphs with bounded degree, Graphs Combin. 17 (3) (2001) 579–587.