Discrete Applied Mathematics 205 (2016) 109–125

Contents lists available at ScienceDirect

Discrete Applied Mathematics

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

List rankings and on-line list rankings of graphs

Daniel C. McDonald *

Department of Mathematics, University of Illinois, Urbana, IL, USA

ARTICLE INFO

Article history:
Received 15 January 2014
Received in revised form 10 December 2015
Accepted 16 December 2015
Available online 13 January 2016

Keywords:
List ranking
On-line list ranking
Vertex ranking
Ranking number
Tree-depth

ABSTRACT

A *k-ranking* of a graph G is a labeling of its vertices from $\{1, \ldots, k\}$ such that any path on at least two vertices whose endpoints have the same label contains a larger label. The least k for which G has a k-ranking is the *ranking number* of G, also known as *tree-depth*. The *list ranking number* of G is the least k such that if each vertex of G is assigned a set of k potential labels, then G can be ranked by labeling each vertex with a label from its assigned list. Rankings model a certain parallel processing problem in manufacturing, while the list ranking version adds scheduling constraints. We compute the list ranking number of paths, cycles, and trees with many more leaves than internal vertices. Some of these results follow from stronger theorems we prove about on-line versions of list ranking, where each vertex starts with an empty list having some fixed capacity, and potential labels are presented one by one, at which time they are added to the lists of certain vertices; the decision of which of these vertices are actually to be ranked with that label must be made immediately.

1. Introduction

We consider a special type of proper vertex coloring using positive integers, called "ranking". As with proper colorings, there exist variations on the original ranking problem. In this paper we consider the list ranking problem, first posed by Jamison in 2003 [10]. For a graph G, we denote its vertex set by $V(G)$ and its edge set by $E(G)$. For $S \subseteq V(G)$ we let $G - S$ denote the subgraph of G obtained by deleting all vertices in S as well as any edges incident to at least one vertex in S, and we call $G - (V(G) - S)$ the subgraph of G *induced* by S. We also set $\mathbb{N} = \{0, 1, 2, \ldots\}$.

Definition 1.1. Let G be a finite simple graph, and let $f : V(G) \to \mathbb{N}$. An *f-ranking* α of G labels each $v \in V(G)$ with an element of $\{1, \ldots, f(v)\}$ in such a way that if $u \neq v$ but $\alpha(u) = \alpha(v)$, then every u, v-path contains a vertex w satisfying $\alpha(w) > \alpha(u)$. For a positive integer k, a *k-ranking* of G is an f-ranking for the constant function $f = k$. The *ranking number* of G, denoted here by $\rho(G)$ (though in the literature often as $\chi_r(G)$), is the minimum k such that G has a k-ranking.

See Fig. 1 for an example of a ranking. Note that α is a ranking of G if and only if every path contains a unique vertex with largest label, or, equivalently, for $j \geq 1$ each component of $G - \{v : \alpha(v) > j\}$ contains a unique vertex with largest label. Since each edge in G is a path with no internal vertices, adjacent vertices in G receive distinct colors in any vertex ranking of G, so every vertex ranking of G is also a proper coloring of G. Hence $\rho(G) \geq \chi(G)$, where $\chi(G)$ denotes the *chromatic number* of G, the least number of colors needed to properly color G.

Vertex rankings of graphs were introduced in [7], and results through 2003 are surveyed in [10]. Their study was motivated by applications to VLSI layout, cellular networks, Cholesky factorization, parallel processing, and computational geometry. Vertex rankings are sometimes called ordered colorings, and the ranking number of a graph is trivially equal to its

* Tel.: +1 6185938759.
E-mail address: dmcdona4@illinois.edu.

http://dx.doi.org/10.1016/j.dam.2015.12.014
0166-218X/

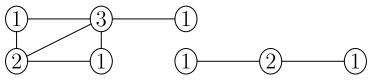

Fig. 1. A 3-ranking of a graph G.

"tree-depth", a term introduced by Nešetřil and Ossona de Mendez in 2006 [12] in developing their theory of graph classes having bounded expansion, which has been the topic of much further study [13–16].

In general, rankings are used to design efficient divide-and-conquer strategies for minimizing the time needed to perform interrelated tasks in parallel [8]. The most basic example concerns a complex product being assembled in stages from its individual parts, where each stage of construction consists of individual parts being attached to previously assembled components in such a way that no component ever has more than one new part. Here, the complex product is represented by the graph G whose vertices are the individual parts and whose edges are the connections between those parts; assuming all parts require the same amount of time to be installed, the fewest number of stages needed to complete construction is $\rho(G)$, achieved by finding some $\rho(G)$-ranking α of G and installing each part v in stage $\alpha(v)$. Similarly, rankings can be used to optimize the disassembly of a product into parts, where each stage of deconstruction consists of individual parts being detached from remaining components in such a way that no component loses multiple parts at the same time; G can be disassembled in $\rho(G)$ stages by removing each part v in stage $\rho(G) - \alpha(v) + 1$.

A ranking of a graph G can also be viewed as a (successful) search strategy [5], wherein stationary searchers and an agile fugitive occupy vertices of G. The searchers place themselves one by one onto the vertices of G, where they remain permanently, until one searcher is placed on the vertex currently occupied by the fugitive, with the placement of each new searcher operating under the following procedure. First, with the previously placed searchers permanently occupying some set $S \subset V(G)$, the location of the fugitive is revealed as some vertex $v \in V(G) - S$. The new searcher then announces some vertex $u \in V(G) - S$ she will occupy (for the rest of the search), but before the new searcher can occupy u, the fugitive is allowed to move any distance along any path in $G-S$ (so the new searcher is placed only with the knowledge that the fugitive must stay in the same component of $G - S$). Given a k-ranking of G, k searchers can always catch the fugitive by placing a new searcher on the highest ranked vertex in the component of $G-S$ containing the fugitive; conversely, k searchers suffice to guarantee capture of the fugitive only if G is k-rankable.

The vertex ranking problem has spawned multiple variations, including edge ranking [3,9], on-line ranking [2,6,11], and list ranking, introduced in [10] and studied here. The list ranking problem is to vertex ranking as the list coloring problem is to ordinary vertex coloring.

Definition 1.2. A function L that assigns each vertex of G a finite set of positive integers is an *f-list assignment* for G if $|L(v)| = f(v)$ for each $v \in V(G)$. If $|L(v)| = k$ for all v, then L is *k-uniform*. An *L-ranking* of G is a ranking α such that $\alpha(v) \in L(v)$ for each v. Say that G is *f-list rankable* if G has an L-ranking whenever L is an f-list assignment for G, and say that G is *k-list rankable* if G is *f-list rankable* for $f = k$. Let the *list ranking number* of G, denoted $\rho_\ell(G)$, be the least k such that G is k-list rankable.

Note that G is f-rankable if G is f-list rankable, since an f-ranking is just an L-ranking if L is the f-list assignment defined by $L(v) = \{1, \ldots, f(v)\}$ for each $v \in V(G)$. Furthermore, $\rho_\ell(G) \geq \chi_\ell(G)$, where $\chi_\ell(G)$ denotes the *list chromatic number* of G, the smallest k such that G can always be properly colored from a list assignment L giving each vertex a set of k potential colors. In terms of our application of assembling a product from parts in stages, obtaining an L-ranking corresponds to finding a feasible schedule for assembling the product when predetermined scheduling constraints limit each individual part v to being attached during a stage listed in $L(v)$. In terms of our searching problem, obtaining an L-ranking corresponds to constructing a strategy for the searchers guaranteed to lead to the capture of the fugitive when predetermined scheduling constraints allow a vertex v to accept a new searcher at time t only if $t = m-j+1$ for some $j \in L(v)$, where m is the maximum label appearing in any list assigned by L.

The complexity of determining whether a given list assignment admits a ranking of a graph was considered by Dereniowski in 2008 [4], who produced a polynomial time reduction from the set cover decision problem (given a set S, j of its subsets, and a positive integer t, is S the union of t of these subsets?) to the list ranking decision problem for certain graphs. This reduction showed the list ranking decision problem to be NP-complete for several classes of trees and their line graphs, including full binary trees. It was also shown that, given a list assignment L for a path, the problem of finding an L-ranking minimizing the maximum label used, or determining that no L-ranking exists, is polynomially solvable.

In Section 2 we introduce three on-line versions of list ranking, which relate to list ranking similarly to the way on-line list coloring (also known as paintability) relates to list coloring [17]. We shall define these on-line versions of list ranking as games between adversaries Taxer and Ranker to be played on a predetermined graph G. At the beginning of the game each vertex is assigned a size for its list of potential labels, but no actual labels. Taxer in effect fills out these lists in real time by iterating through the possible labels one by one, stipulating which vertices have the given label in their lists (the order in which Taxer iterates through the labels depends on the version of the game, and once a label comes up it cannot be revisited). Ranker must decide immediately which of those vertices just selected by Taxer are to receive the given label, extending a partial ranking of G (Taxer is allowed to use knowledge of the partial ranking already created by Ranker when