Available online at www.sciencedirect.com

ScienceDirect

Discrete Applied Mathematics 156 (2008) 1918–1928

DISCRETE APPLIED MATHEMATICS

www.elsevier.com/locate/dam

Vertex coloring acyclic digraphs and their corresponding hypergraphs☆

Geir Agnarsson[a], Ágúst S. Egilsson[b], Magnús M. Halldórsson[c]

[a]Department of Mathematical Sciences, George Mason University, MS 3F2, 4400 University Drive, Fairfax, VA 22030, USA
[b]Science Institute, University of Iceland, IS-107 Reykjavik, Iceland
[c]Department of Computer Science, University of Iceland, IS-107 Reykjavik, Iceland

Received 21 November 2004; received in revised form 11 July 2007; accepted 26 July 2007
Available online 22 October 2007

Abstract

We consider vertex coloring of an acyclic digraph $\vec{G}$ in such a way that two vertices which have a common ancestor in $\vec{G}$ receive distinct colors. Such colorings arise in a natural way when bounding space for various genetic data for efficient analysis. We discuss the corresponding *down-chromatic number* and derive an upper bound as a function of $D(\vec{G})$, the maximum number of descendants of a given vertex, and the degeneracy of the corresponding hypergraph. Finally, we determine an asymptotically tight upper bound of the down-chromatic number in terms of the number of vertices of $\vec{G}$ and $D(\vec{G})$.
© 2007 Elsevier B.V. All rights reserved.

MSC: 05C15; 05C20; 05C65; 05C85; 05D90; 06A06

Keywords: Genetic databases; Digraph; Ancestor; Down-set; Vertex coloring; Hypergraph; Block design

1. Introduction

The purpose of this article is to discuss a special kind of vertex coloring for acyclic digraphs, where vertices with a common ancestor must receive distinct colors. We discuss some properties of such colorings, similarity and differences with strong hypergraph colorings, and derive an upper bound which, in addition, yields an efficient coloring procedure.

Digraphs representing various biological phenomena and knowledge are ubiquitous in the life sciences and in drug discovery research, e.g. the gene ontology digraph maintained by the Gene Ontology Consortium [7]. An overview of several projects relating to indexing of semistructured data (i.e. acyclic digraphs) can be found in [1]. In these biological digraphs it is important to be able to access the ancestors of nodes in a fast and efficient manner.

Consider the problem of finding a representation of an acyclic digraph in a database to allow for fast access to the set of ancestors of a given node. The ancestors of a node are its in-neighbors in the transitive closure of the digraph. If the digraph is sparse and shallow, the transitive closure is also sparse. Thus, an adjacency matrix representation would be neither efficient nor fast. On the other hand, a matrix has the advantage of corresponding nicely to the relational representation of modern databases. In a database relation that corresponds to an adjacency matrix, the non-empty elements in each column correspond to the in-neighbors of the node indexing that column. This vertex set can then be

☆ Some of the results presented here appeared in a preliminary form in [3].
E-mail addresses: geir@math.gmu.edu (G. Agnarsson), egilsson@hi.is (A.S. Egilsson), mmh@hi.is (M.M. Halldórsson).

0166-218X/$ - see front matter © 2007 Elsevier B.V. All rights reserved.
doi:10.1016/j.dam.2007.07.022

Table 1
An example for $n = 6$

	g_1	g_2	g_3	g_4	g_5	g_6
g_1	1	0	0	1	1	0
g_2	0	1	0	1	0	1
g_3	0	0	1	0	1	1
g_4	0	0	0	1	0	0
g_5	0	0	0	0	1	0
g_6	0	0	0	0	0	1

$\rightarrow$

	1	2	3
g_1	g_1	g_4	g_5
g_2	g_6	g_4	g_2
g_3	g_6	g_3	g_5
g_4	–	g_4	–
g_5	–	–	g_5
g_6	g_6	–	–

combined by joins with other tables that are also indexed by vertices, giving an effective language of querying based on graphical properties. Thus, it would be preferable to find a representation that both has a matrix structure, yet consists of relatively small rows.

One compact matrix representation would be to store the adjacency lists in compacted array form, where a list with k elements is stored in the first k array elements. In this case, however, there is no easy way of accessing all the edges entering a given vertex. While this could be alleviated by storing the inverted adjacency matrix, note that in the context of database access, rows are conceptually different from columns. Instead, we seek a compacted representation where all in-edges of a given node are stored in the same column. We say that a many-to-one mapping of vertices to columns that preserves adjacency lists, has the *AC-property*. By recording the mapping of nodes to their respective column storing their in-neighbors, one obtains the same desirable properties of adjacency matrices in the context of a relational database. If the graph is sparse, the possibilities of storage reduction are significant. The actual improvement is related to the number of colors needed in a certain coloring of the digraph, which we now briefly discuss.

A proper *down-coloring* of a digraph is a vertex coloring where vertices with a common ancestor receive different colors. The *down-chromatic number* of a digraph is the minimum number of colors in a down-coloring of the digraph. In a compacted matrix representation of the transitive closure of a digraph, we assign multiple vertices to the same column, but in such a way that their in-adjacency lists must be disjoint. Two vertices have disjoint sets of in-neighbors in the transitive closure if, and only if, they have no common ancestor. Therefore, a down-coloring of a digraph corresponds to a valid compacted representation of its transitive closure, and the down-chromatic number is the minimum number of columns needed in such a representation.

Example. Consider the digraph $\vec{G}$, on $n = 6$ vertices representing genes, where a directed edge from one vertex to a second one indicates that the first gene is an ancestor of the second gene.

$$V(\vec{G}) = \{g_1, g_2, g_3, g_4, g_5, g_6\},$$
$$E(\vec{G}) = \{(g_1, g_4), (g_1, g_5), (g_2, g_4), (g_2, g_6), (g_3, g_5), (g_3, g_6)\}.$$

In the adjacency matrix representation of this 6 node digraph, we assign a column to each vertex g_i. As we see in the left diagram of Table 1, most of the entries of this 6×6 matrix are empty. By reducing the number of columns in such a way that the AC-property still holds, we obtain a smaller and more compact 6×3 matrix representation as seen on the right diagram of Table 1. There, the ith row still contains the descendants of g_i and the ancestors of g_i are those g_j's whose rows g_i appears in. Note that (i) each g_i appears in exactly one column and (ii) two genes appear in the same column only if their sets of ancestors are disjoint. Here we can view the column numbers 1, 2, and 3 as distinct colors assigned to each vertex. Note further that in this example the transitive closure of $\vec{G}$ is simply $\vec{G}$ itself. An explicit example of how such a coloring can speed up queries in the gene ontology digraph can be found in the appendix of [6].

Hence, the following two questions regarding such colorings, one computational and the other theoretical, are quite natural: (1) For a given digraph (of no particular structure!) how can we assign reasonably few colors to the vertices/columns efficiently, and (2) in general, how large can the discrepancy theoretically be between the actual minimum number of colors needed and the obvious lower bound of needed colors?