Theoretical Computer Science 596 (2015) 79–91

Contents lists available at ScienceDirect

Theoretical Computer Science

www.elsevier.com/locate/tcs

Online multi-coloring with advice ☆

Marie G. Christ, Lene M. Favrholdt, Kim S. Larsen *

Department of Mathematics and Computer Science, University of Southern Denmark, Campusvej 55, DK-5230 Odense M, Denmark

ARTICLE INFO

Article history:
Received 23 December 2014
Received in revised form 18 May 2015
Accepted 21 June 2015
Available online 29 June 2015
Communicated by V.Th. Paschos

Keywords:
Advice complexity
Graph multi-coloring
Frequency allocation
Online algorithms
Competitive analysis

ABSTRACT

We consider the problem of online graph multi-coloring with advice. Multi-coloring is often used to model frequency allocation in cellular networks. We give several nearly tight upper and lower bounds for the most standard topologies of cellular networks, paths and hexagonal graphs. For the path, negative results trivially carry over to bipartite graphs, and our positive results are also valid for bipartite graphs. For hexagonal graphs, negative results trivially carry over to 3-colorable graphs, and most of our positive results do as well. The advice given represents information that is likely to be available, studying for instance the data from earlier similar periods of time.

1. Introduction

We consider the problem of graph *multi-coloring*, where each node may receive multiple requests. Whenever a node is requested, a color must be assigned to the node, and this color must be different from any color previously assigned to that node or to any of its neighbors. The goal is to use as few colors as possible. In the *online* version, the requests arrive one by one, and each request must be colored without any information about possible future requests. The underlying graph is known to the online algorithm in advance.

The multi-coloring problem is motivated by *frequency allocation* in cellular networks. These networks are formed by a number of base transceiver stations, each of which covers what is referred to as a cell. Due to possible interference, neighboring cells cannot use the same frequencies. In this paper, we use classic terminology and refer to these cells as nodes in a graph where nodes are connected by an edge if they correspond to neighboring cells in the network. Frequencies can then be modeled as colors. Multiple requests for frequencies can occur in one cell and overall bandwidth is a critical resource.

Two basic models dominate in the discussion of cellular networks, the highway and the city model. The former is modeled by linear cellular networks, corresponding to paths, and the latter by hexagonal graphs. We consider the problem of multi-coloring such graphs.

For practical applications, the assumption that absolutely nothing is known about the future is often unrealistic and hence, many problems have been studied in various semi-online settings. The notion of *advice* offers a quantitative and problem independent way of relaxing the online constraint. In this model, the online algorithm is provided with partial

☆ Supported in part by the Danish Council for Independent Research (FI-2146-09-0073 and DFF-1323-00247) and the Villum Foundation (VKR023219). An extended abstract appeared in the Twelfth Workshop on Approximation and Online Algorithms (WAOA), Lecture Notes in Computer Science, vol. 8952, Springer, 2014.

* Corresponding author.
E-mail addresses: christm@imada.sdu.dk (M.G. Christ), lenem@imada.sdu.dk (L.M. Favrholdt), kslarsen@imada.sdu.dk (K.S. Larsen).

http://dx.doi.org/10.1016/j.tcs.2015.06.044

knowledge about the future by an oracle writing bits of advice to an advice tape. The *advice complexity* of the algorithm is the maximum total number of advice bits read from the tape, as a function of the length of the input sequence.

The advice supplied to the algorithms studied in this paper is essentially (an approximation of) the maximum number of requests given to any clique in the graph. For the application of frequency allocation, it does not seem unrealistic that this information could be derived from previous data.

2. Preliminaries

2.1. The problem

In this section we define the problem of online multi-coloring formally. A graph, $G = (V, E)$, is given from the beginning, and the input sequence consists of requests to nodes in G. Each time a node, v, is requested, the algorithm must assign a color to v. The color must be different from all colors previously given to v and its neighbors, and it must be chosen without any knowledge about possible future requests. The colors are positive integers, and the goal is to minimize the largest color used.

Multi-coloring has been studied in various settings. Some papers study a variation of the problem where requests may be *canceled*, freeing their colors for future requests. In this paper, we only very briefly consider cancellations. Another variation is to allow color changes (reassignment of frequencies). This is called *recoloring*. An algorithm is *d-recoloring* if, in the process of treating a request, it may recolor up to a distance d away from the node of the request.

2.2. Notation

Throughout, we let n denote the number of requests in a given input sequence. We let log denote $\log_2$, the logarithm with base 2.

If A is a multi-coloring algorithm, we let $\mathrm{A}(I)$ denote the number of colors used by A on the input sequence I. When I is clear from the context, we simply write A instead of $\mathrm{A}(I)$.

For any input sequence, I, we let $\textsc{Opt}(I)$ (or simply $\textsc{Opt}$) denote the number of colors used by an optimal offline algorithm when given the requests in I.

2.3. Competitive ratio

The quality of an online algorithm is often given in terms of the competitive ratio, defined by Sleator and Tarjan [35] and named by Karlin et al. [27]. An online multi-coloring algorithm is *c-competitive* if there exists a constant α such that for all input sequences I, $\mathrm{A}(I) \leq c\,\textsc{Opt}(I) + \alpha$. The (asymptotic) *competitive ratio* of A is the infimum over all such c. Results that can be established using $\alpha = 0$ are referred to as *strict* (or absolute).

We use the term *strictly* 1*-competitive* to denote that an algorithm is as good as an optimal offline algorithm, and *optimal* to mean that no better online algorithm exists under the given conditions.

2.4. Advice complexity

We use the advice model by Hromkovič, Královič, and Královič [24], where the online algorithm has access to an infinite advice tape, written by an offline oracle with infinite computation power. In other words, the online algorithm can ask for the answer to any question and read the answer from the tape. Competitiveness is defined and measured as usual, and the advice complexity is simply the number of bits read from the tape. More precisely, for each given sequence length, n, the *advice complexity* of an algorithm is the maximum number of advice bits read by the algorithm, over all sequences of length at most n.

As the advice tape is infinite, we need to specify how many bits of advice the algorithm should read and if this knowledge is not implicitly available, it has to be given explicitly in the advice string. For instance, if we want $\textsc{Opt}$ as advice, then we cannot merely read $\lceil \log(\textsc{Opt} + 1) \rceil$ bits, since this would require knowing something about the value of $\textsc{Opt}$.

One can use a *self-delimiting encoding* as introduced by Elias [21]. We use the variant by Boyar et al. [11], defined as follows: The value of a non-negative integer X is encoded by a bit sequence, partitioned into three consecutive parts. The last part is X written in binary. The middle part gives the number of bits in the last part, written in binary. The first part gives the number of bits in the middle part, written in unary and terminated with a zero. These three parts require $\lceil \log(\lceil \log(X+1) \rceil + 1) \rceil + 1$, $\lceil \log(\lceil \log(X+1) \rceil + 1) \rceil$, and $\lceil \log(X+1) \rceil$ bits, respectively, adding a low order term to the number of bits of information required by an algorithm.

We define $enc(x)$ to be the minimum number of bits necessary to encode a number x, and note that the encoding above is a (good) upper bound on $enc(x)$.