ARTICLE IN PRESS

Discrete Applied Mathematics ▮ (▮▮▮▮) ▮▮▮–▮▮▮

Contents lists available at ScienceDirect

Discrete Applied Mathematics

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

Graph multi-coloring for a job scheduling application

Simon Thevenin [a], Nicolas Zufferey [a,c,*], Jean-Yves Potvin [b,c]

[a] *Geneva School of Economics and Management, GSEM - University of Geneva, Switzerland*

[b] *Département d'informatique et de recherche opérationnelle - Université de Montréal, Québec, Canada*

[c] *Centre Interuniversitaire de Recherche sur les Réseaux d'Entreprise, la Logistique et le Transport (CIRRELT), Québec, Canada*

ARTICLE INFO

Article history:
Received 30 March 2015
Received in revised form 18 May 2016
Accepted 23 May 2016
Available online xxxx

Keywords:
Metaheuristics
Job scheduling
Graph coloring

ABSTRACT

In this paper, we introduce a graph multi-coloring problem where each vertex must be assigned a given number of different colors, represented as integers, and no two adjacent vertices can share a common color. In the variant considered, the number of available colors is such that not all vertices can be colored. Furthermore, there is a bound on the number of vertices which can be assigned the same color. A gain is associated with each vertex and the first objective is to maximize the total gain over all colored vertices. Secondary objectives consider the sequence of colors assigned to each vertex. More precisely, the range and the number of interruptions must be minimized, where the range corresponds to the difference between the largest and smallest colors assigned to a vertex. This variant of the graph multi-coloring problem is of interest because it can model practical job scheduling applications. An integer linear programming formulation is first proposed to address small-size instances. A construction heuristic, as well as local search methods, are then reported to tackle larger instances. The local search methods are based on several neighborhood structures, each one focusing on a specific property of the problem. Different ways to combine these neighborhood structures are also investigated.

1. Introduction

The graph coloring problem (GCP) consists of assigning a single color (integer) to each vertex of an undirected graph, such that no two adjacent vertices share the same color, with the objective of minimizing the number of colors. The graph multi-coloring problem is an extension of the GCP, where each vertex must be assigned a preset number of different colors and no two adjacent vertices can share a common color. In this case, a vertex coloring corresponds to a sequence of colors (from the smallest to the largest). The objective is again to minimize the number of used colors.

Our goal is to solve a variant of the multi-coloring problem that is suitable for job scheduling problems with the two following main characteristics: (1) there is no feasible solution where all jobs can be performed, thus leading to a job acceptance (or job rejection) problem, and (2) each job requires the simultaneous use of two or more resources. Some enthusiasm has recently been observed in the research community for problems with characteristic (1), also called *order acceptance and scheduling problems*, because the demand often exceeds the production capacity. On the other hand, characteristic (2) refers to multiprocessor scheduling problems where two or more resources are required simultaneously to process a job. Although these problems originate from the scheduling of processors in computer systems, they are also found in other production environments. Examples include chemical plants [37], equipment manufacturers for power supply

* Correspondence to: Uni-Mail, Bd du Pont-d'Arve 40, 1211 Geneva 4, Switzerland.
E-mail addresses: simon.thevenin@unige.ch (S. Thevenin), n.zufferey@unige.ch (N. Zufferey), potvin@iro.umontreal.ca (J.-Y. Potvin).

http://dx.doi.org/10.1016/j.dam.2016.05.023
0166-218X/

Please cite this article in press as: S. Thevenin, et al., Graph multi-coloring for a job scheduling application, Discrete Applied Mathematics (2016), http://dx.doi.org/10.1016/j.dam.2016.05.023

factories [48] and production of key components of aircrafts [33]. As far as we know, these two characteristics have never been studied jointly, although they can model real-world problems, like the beauty care manufacturing application reported later.

We consider two types of resource: the *shared* resources (e.g., parallel machines) are involved in the processing of all jobs and limit the number of jobs that can be processed concurrently, and the *critical* resources are unique. Two jobs requiring the same critical resource, in addition to shared resources, are said to be *incompatible*. Critical resources might come in a few exemplars in practical applications, but a common approach is to simplify the problem by aggregating them and assigning them at once to a subset of jobs in a first step, so that they can be considered unique thereafter (see [7,12,13]).

More formally, we assume a planning horizon divided into a number of discrete time units. We have a set of n jobs, along with an incompatibility graph $G = (V, E)$, where V is the set of jobs, and E is a set of edges between incompatible jobs. These jobs cannot be performed during a common time unit, and no more than l jobs can be scheduled concurrently because the shared resources are available only in l exemplars. Preemption is allowed, that is, each job can be stopped at integer time points and resumed later. All jobs must terminate before a preset global deadline D that stands for the end of the planning horizon. This deadline is such that only a subset of jobs can be scheduled. Each job i has an integer processing time of p_i time units, and a gain g_i is earned when it is entirely performed. There is no gain for a partially performed job, so it is better in this case not to perform it at all. The goal is to optimize the three following objectives in lexicographic order:

- f_1: maximize the total gain of the scheduled jobs;
- f_2: minimize the total number of job interruptions;
- f_3: minimize the total flow time (the flow time of a job i is the total time spent by i in the production facility).

It should be noted that a close problem was considered in [45], but the very sensitive job acceptance issue was ignored. The above lexicographic approach was chosen due to a natural hierarchy among the three objectives in many practical applications. More precisely: (1) minimizing the total gain of the scheduled jobs is the most important objective because it is related to the total income, (2) considering job interruptions in production scheduling is relevant when each job is made of multiple batches of the same or different products (see below). However, too many interruptions should be avoided as it leads to delays and unproductive work, (3) through minimization of the total flow time, the amount of work in progress (which generates inventory costs) is reduced. Clearly, these three objectives interact together. For instance, it was shown in the job scheduling literature that allowing preemption is a way to reduce the makespan (e.g., [36]). This was also observed in [45] for a problem related to (P). Although the makespan is not one of our objectives, it implies in our context that more jobs can be scheduled within a given deadline, thus increasing f_1.

An example of this type of problem is encountered at an international consumer goods company whose headquarters are in Geneva. It concerns the manufacturing of beauty care lotions, which is characterized by large quantities sold and a large variety of products. The production system uses continuous flow production and is very flexible (as it can be quickly reconfigured). To prepare a given lotion type, different resources are required simultaneously: tanks of different raw materials, a mixer, employees to perform and supervise the production process, and chemists to control the quality and safety during the production process. Due to their technical characteristics (or specific skills), the critical resources are tanks of raw products and chemists. That is, two products requiring a common critical resource cannot be processed concurrently. A time unit (typically several hours) corresponds to the time required to fill a tank of finished product (it is similar for all product types). The production system can only be reconfigured after the filling of each tank, and the changeover time is negligible. In other words, it can be assumed that preemption is only allowed at integer time points. Each job refers to the quantity (in tank units) of a product type to be delivered at a given distribution center. Each job is associated with a processing time (related to quantity) and a gain equal to the gross margin. The problem is to schedule the production for the next week, considering incompatibilities due to critical resources, and the limited amount of shared resources. Job acceptance is also an issue, as it might not be possible to schedule all of them. The objective is then to maximize the total gain associated with the scheduled jobs.

The variant of the graph multi-coloring problem that is studied in this paper to address this type of scheduling problems can be stated as follows. We have an undirected graph $G = (V, E)$, where V is the vertex set and E is the edge set. Each vertex i stands for a job and there is an edge between two vertices if the corresponding jobs are incompatible. Each color represents a particular time unit of the scheduling horizon. Color c is then assigned to vertex i if one time unit of the corresponding job is processed during that time unit c. Each vertex i must be assigned p_i colors as defined by the processing time of the job. The total number of available colors is D and this number is such that it is not possible to fully color all vertices. In addition, each color c cannot be assigned to more than l vertices (due to the limited number of shared resources). Each vertex is associated with a gain g_i and the resulting problem (P) consists of coloring a subset of vertices in the graph, such that no two adjacent vertices share the same color, while optimizing the following objectives in lexicographic order:

- f_1: maximize the total gain over all fully colored vertices;
- f_2: minimize the sum of the number of interruptions in the sequence of colors assigned to each vertex, over all fully colored vertices;
- f_3: minimize the sum of the range of colors assigned to each vertex, over all fully colored vertices, where the range is defined as the difference between the largest color and the smallest color.

Please cite this article in press as: S. Thevenin, et al., Graph multi-coloring for a job scheduling application, Discrete Applied Mathematics (2016), http://dx.doi.org/10.1016/j.dam.2016.05.023