EQUITABLE COLORING AND SCHEDULING ON IDENTICAL MACHINES

Sarah NOURI

RECITS laboratory,
Faculty of Mathematics,
USTHB University,
BP 32 El-Alia, Bab Ezzouar,
Algiers, Algeria.
Centre de Recherche
sur l'Information Scientifique et
Technique (CERIST)
05, Rue des 3 frères aissou,
Ben Aknoun Algiers Province,
Algeria

Mourad BOUDHAR

RECITS laboratory,
Faculty of Mathematics,
USTHB University,
BP 32 El-Alia, Bab Ezzouar,
Algiers, Algeria.

ABSTRACT

In this work we are interested in the complexity of the equitable coloring of the bipartite graph in which we prove that the problem of 2-equitable coloring of union of complete bipartite graphs and the 3-equitable coloring of a connected bipartite graph are $\mathscr{NP}$-complete.

It also discusses the scheduling of conflicting jobs (that cannot be executed on the same machine) on identical machines while evenly distributing the load between them. The paper presents some complexities of subproblems and describes and evaluates a branch and bound algorithm, a Mixed Integer Programming formulations (that can solve some instances with 100 jobs) and six proposed heuristics where their good performance is shown in computational experiments. Such problem with identical processing times can be viewed as r-equitable k-coloring.

1. INTRODUCTION

A proper and complete vertex coloring of a graph is a way of coloring all its vertices such that no two adjacent vertices are of the same color. A proper coloring of the vertices of a graph is equitable if, for any two different classes of colors, the difference between the sizes of these classes is at most one. The notion of equitable coloring is topical in recent years because coloring has become a major and active area of graph theory, and several applications of this problem have emerged and motivated mathematicians. This notion appeared in October 1973 thanks to Meyer[4] who was motivated by an article of Tucker[5]. Tucker presented in his article a coloring problem of a graph, where the vertices represent the garbage collection routes, with two vertices (routes) are adjacent if and only if these two routes cannot be run on the same day. This transport problem consists in finding a partition of the routes so that they are visited all in 6 days. Meyer thought it would be more interesting if the number of routes run each day were approximately the same along the week (so finding an equitable 6-coloring). For this garbage collection routes problem, the routes may not be identical in length or garbage quantity. We propose to assign weights to the vertices and to partition the graph so that the difference between the sum of the independent sets' weights is as small as possible. That's lead us to consider a scheduling problem

of uninterruptible jobs on identical machines, where some conflict jobs cannot be executed on the same machine (these constraints are modeled by a conflict graph). In this graph, the vertices represent jobs and two vertices are adjacent if and only if corresponding jobs are in conflict. We want to schedule jobs while minimizing the large difference between the scheduling lengths of the machines ε_{max}.

Many real-life problems can be seen as an application of our problem :
— Distributing process having different lengths evenly across a computer network so that no single device is overwhelmed.
— The attribution of courses ; that may not have the same duration ; to slots so as to avoid simultaneously organizing pairs of incompatible courses and to evenly distribute the courses between available time intervals.
— Scheduling television commercials during commercial breaks of the same length[1].
In [2] authors generilazed the equitable coloring as follow :

Definition 1.1 *An r-equitable k-coloring of the vertices of G is a k-coloring, such that, for any two different classes of colors, the difference between the sizes of these classes is at most r.*

2. EQUITABLE COLORING

We have shown that equitable coloring of the bipartite graph is NP-hard when the number of colors is equal to 2 or 3.

Theorem 2.1 *The problem of* 2*-equitable coloring remains $\mathcal{NP}$-complete for union of complete bipartite graphs.*

Proposition 2.2 3*-equitable coloring of a connected bipartite graph is $\mathcal{NP}$-complete.*

3. SCHEDULING ON IDENTICAL MACHINES WITH CONFLICT GRAPHS

Let $G = (J,E)$ be a conflict graph where J is a set of n jobs $(J_j)_{j=1,\cdots,n}$ and each job J_j has a processing time p_j. We schedule J on m identical machines $(M_i)_{i=1,\cdots,m}$, such that no jobs adjacent in G can be processed on the same machine. In the indicated problem, a job can be processed by any machine and the aim is to schedule jobs while minimizing the large difference between the scheduling lengths of the machines ε_{max}. This problem denoted $Pm|G|\varepsilon_{max}$ has a solution if $m \geqslant \chi(G)$, with $\chi(G)$ the chromatic number of G.

3.1. Complexity of the problem

Proposition 3.1 *The following problems :*
— $P2||\varepsilon_{max}$,
— $P2|G|\varepsilon_{max}$ *with G a unconnected bipartite graph,*
— $P2|G|\varepsilon_{max}$ *with all vertices of G are of degree* 1,
— $P2|G, p_j = 1|\varepsilon_{max}$ *with G a union of complete bipartite graphs and* $\varepsilon_{max} \leqslant 1$,
— $P3|G, p_j = 1|\varepsilon_{max}$ *with G a connected bipartite graph and* $\varepsilon_{max} \leqslant 1$,
— $Pm|p_j = 1, G|\varepsilon_{max}$.
are $\mathcal{NP}$-Hard.

Theorem 3.2 $P3|E_{n+1}|\varepsilon_{max}$ *where* E_{n+1} *is the star graph of* $n+1$ *jobs is $\mathcal{NP}$-Hard.*

Theorem 3.3 *Let* $P2|UChains|\varepsilon_{max}$ *be a problem of scheduling UChains (a set of chains) on two identical machines with the objective of minimizing* ε_{max}*, while the processing time of each job of a chain k is* 1 *or* p_k*. This problem is $\mathcal{NP}$-Hard.*

Proposition 3.4 *$Pm|Chain, p_j = 1|\varepsilon_{max}$ is polynomialy solvable in $O(1)$.*

Proposition 3.5 *$Pm|UChains, pj = 1|\varepsilon_{max}$ with UChains a union of chains, is polynomialy solvable in O(1).*

Proposition 3.6 *$Pm|Cycle, p_j = 1|\varepsilon_{max}$ with $m \geq 3$ is polynomialy solvable in $O(1)$.*

Proposition 3.7 *$Pm|Wheel, p_j = 1 \quad , j \neq center|\varepsilon_{max}$ is polynomialy solvable in $O(1)$.*

3.2. Mixed Integer Programming

A Mixed Integer Programming (*MIP*) formulation of the minimum ε_{max} problem follows : For every job j in J and machine i in M, we introduce a binary variable x_{ij} equal to 1 if job j is assigned to machine i and 0 otherwise. Thus, the formulation is :

$$\begin{cases} \min \varepsilon_{max} & & \\ \sum_{i=1}^{m} x_{ij} & = 1, & \forall j \in J \\ x_{ij} + x_{it} & \leqslant 1, & \forall (j,t) \in E, \forall i \in M \\ \sum_{j=1}^{n} p_j x_{ij} - \sum_{j=1}^{n} p_j x_{lj} & \leqslant \varepsilon_{max}, & \forall i \neq l \in M \\ x_{ij} & \in \{0,1\}, & \forall i \in M, \forall j \in J \\ \varepsilon_{max} & \geq 0 & \end{cases}$$

The first set of equalities ensures that every job is scheduled on exactly one machine. The second one force conflicting jobs to be scheduled on different machines, and the third one guarantees the load balancing constraint. It is enough to know just the optimal assignment because the ε_{max} will be the same for any permutation of the jobs assigned to each machine. This formulation has $n + m(m + 2|E| - 1)$ constraints and $mn + 1$ variables.

We also proposed an improvement to the model (noted *PL2*) that allowed us to solve instances five times larger than those solved by the above model. The linear formulations were solved with *CPLEX* 20.1.

4. RESOLUTION METHODS

A *Branch and Bound* (*B&B*) algorithm is one of the main methods for solving $\mathscr{NP}$-hard discrete optimization problems. In [3], the authors present a Branch and Bound algorithm based on the well-known heuristic *DSatur* from which our branching strategy is inspired. In fact, it extends a partial schedule to a complete schedule, for this :

— a partial schedule is obtained by assigning jobs of a maximal clique Q of G to different machines.

— An upper bound is obtained by one of the heuristics described below.

We extend this partial schedule to a complete one by scheduling one by one the rest of unscheduled jobs (We denote by *PS* the set of partial solutions). The algorithm explores branches of this tree via a recursive call to a function called *NODE* and save the partial solutions in a work pool. In order to reduce the processing speed, a parallelism of the *B&B* is adopted. 15 threads (T) were used and each thread explores a branch of the tree (a partial solution) by an algorithm based on depth first search.

We propose also two approximate methods of resolution (*WN* and *WCSG*), by bringing some modifications to the two well-known heuristics of equitable coloring *NAIVE* and *CreateSubgraph*. The general idea behind these heuristics is for the first one, recoloring a vertice colored with the least frequently used color with the most used one, and for the second approache, swaping colors

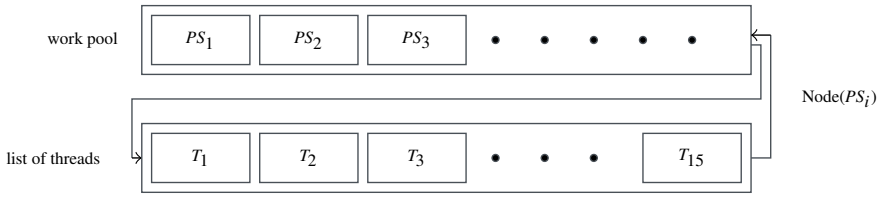

FIGURE 1 – The parallelism principle.

in entire subgraphs induced by vertices colored with the most and the least frequently used colors. We modified how to choose the vertices to be permuted, we permute two jobs if it decreases the large difference between the scheduling lengths of the machines. Four enhancements of the jobs-to-swap search of the *WCSG* heuristic has been established.

4.1. Computational experiments

The proposed algorithms have been tested experimentally on a large number of randomly generated instances. We generate two sets ($A_1 \cup A_2$) of conflict graphs G of different densities, with a number of jobs $n_1 \in \{10,12,15,18,20,25\}$ for A_1 and $n_2 \in \{25,100,1000\}$ for A_2 (9 graphs for each n_1 and 12 graphs for each n_2). Processing times were generated from a uniform distribution in a given range I_j (5 ranges considered $I_1 = [1,10[$, $I_2 = [1,50[$, $I_3 = [1,100[$, $I_4 = [50,100[$ and $I_5 = [20,70[$). For each pair (G, I_j) we generate 30 instances and we run the algorithms to solve them on m_j machines. For A_1 the number of machines is between $\chi(G)$ and $\chi(G)+3$, and for A_2 between $\chi(G)$ and $\chi(G)+5$ (We note by m_j the number of machines $\chi(G)+j-1$). The number of instances is 3960 for low density, 4020 for medium density and 2280 for high density.

The proposed algorithms were coded using $C++11$ and compiled with $g++$ 9.3.0, the computational experiments were carried out using an Intel (R) Xeon (R) Silver 4110 CPU @ 2.10 GHz with 16 logical cores, 16G of RAM and the OS used is Ubuntu 20.04 LTS Focal Fossa server.

In this section we present some results obtained by our methods.

Since the proposed heuristics are improvement heuristics, an initial solution (O_0) must be provided as input. O_0 with $\chi(G)$ machines is an optimal solution to the problem of coloring of G with $\chi(G)$ colors. For the initial solutions with $\chi(G)+k$ machines, they are obtained by reassigning a random number of jobs from the same machine to a new machine until a schedule with $\chi(G)+k$ machines used is obtained.

Table 1 represents the average of the percentage of the number of instances where the heuristics improved the initial solution compared to the density. The table do not show the impact of processing time intervals. In fact, we have noticed that the standard deviation between the percentages of the different processing time intervals for the same heuristic is at most 0.024.

In Table 2, we report the number of resolved instances of A_2 of different densities with $n = 100$ in less than one hour. For each density we test the improvement of the *PL*2 and *B&B* on a graph with the five processing time ranges. In total, 150 instances for each density. The missing instances are either due to the insufficient memory space (noted *IMS*) or the resolution time which exceeded one hour (noted T). Each cell in the following table contains two values : the first is the number of instances resolved by *PL*2 and the second the number of instances resolved by *B&B*.

From the experimental results, we notice that :

Density	Low	Medium	High
WN	66.84	46.44	28.28
WCSG	49.60	43.74	27.89
WCSG1A	64.42	46.18	28.26
WCSG2A	60.23	33.58	12.60
WCSG3A	69.34	40.48	19.52
WCSG4A	65.98	39.28	13.59

TABLE 1 – The average of the percentage of the improvement of O_0

$\frac{PL2}{B\&B}$	m_1	m_2	m_3	m_4	m_5
Low	$\frac{150}{150}$	$\frac{150}{-(IMS)}$	$\frac{-(IMS)}{-(IMS)}$	$\frac{-(IMS)}{-(IMS)}$	$\frac{-(IMS)}{-(IMS)}$
Medium	$\frac{150}{150}$	$\frac{150}{150}$	$\frac{149_{(1IMS)}}{-(IMS)}$	$\frac{149_{(1IMS)}}{-(IMS)}$	$\frac{72_{(39IMS,\,43T)}}{-(IMS)}$
High	$\frac{150}{150}$	$\frac{150}{150}$	$\frac{150}{150}$	$\frac{150}{-(IMS)}$	$\frac{150}{-(IMS)}$

TABLE 2 – by Densities

— ALL heuristics provided good solutions within a maximum arithmetic mean of 9% from the optimal solution.
— *WN* yields good results compared to *WCSG* and its improvements.
— The second improvement of *WCSG* is weaker compared to *WCSG* and the latter's improvements.
— the third and the fourth improvements of *WCSG* are closely competitive.
— For small instances, there is a high chance of obtaining an optimal solution very quickly via *B&B* than *MIP*.
— For large instances and due to the problem of insufficient memory space, the improvement of *MIP* solved more instances than *B&B* in less than an hour.

5. REFERENCES

[1] Gaur,D.R., Krishnamurti, R., and Kohli.R. (2009). *Confict resolution in the scheduling of television commercials*. Operations Research, 57(5) :1098-1105.

[2] Hertz, A., and Ries, B. (2014).*A note on r-equitable k-colorings of trees*. Yugoslav Journal of Operations Research, 24(2),293 - 298.

[3] Méndez-Díaz, I., Nasini, G., and Severín, D. *A DSATUR-based algorithm for the Equitable Coloring Problem*. Computers and Operations Research, 57, 41-50(2015).

[4] Meyer, W. (1973). *Equitable coloring*. The American Mathematical Monthly, 80(8), 920-922.

[5] Tucker, A. (1973). *Perfect Graphs and an Application to Optimizing Municipal Services*. SIAM Review, 15(3), 585–90.