Decision Making in Manufacturing and Services
Vol. 11 • 2017 • No. 1–2 • pp. 53–61

Scheduling of Identical Jobs with Bipartite Incompatibility Graphs on Uniform Machines. Computational Experiments

Szymon Duraj*, Paweł Kopeć*,
Marek Kubale*, Tytus Pikies*

Abstract. In this paper, we consider the problem of scheduling unit-length jobs on three or four uniform parallel machines to minimize the schedule length or total completion time. We assume that the jobs are subject to some types of mutual exclusion constraints, modeled by a bipartite graph of a bounded degree. The edges of the graph correspond to the pairs of jobs that cannot be processed on the same machine. Although the problem is generally NP-hard, we show that our problem can be solved to optimality in polynomial time under some restrictions imposed on the number of machines, their speeds, and the structure of the incompatibility graph. The theoretical considerations are accompanied by computer experiments with a certain model of scheduling.

Keywords: batch scheduling, bipartite graph, polynomial algorithm, uniform machines

Mathematics Subject Classification: 90B35

Submitted: May 01, 2017

Revised: September 17, 2017

1. INTRODUCTION

Suppose we must send n chemical substances from Place A to Place B, and we have m containers with a capacity of n products each. We can send them out simultaneously by rail, by road, and/or by water transport, etc. However, in no container can there be two substances that may react with each other (e.g., due to a transport accident). The aim is an assignment of the chemical substances to the containers so that the total time of transportation of all the substances is as minimal as possible.

Our problem can be expressed as the following scheduling problem. Suppose we have n identical jobs $j_1, ..., j_n$, so we assume that they all have unit execution times in symbols UET. They are to be processed on m non-identical machines $M_1, ..., M_m$. These machines run at different speeds $s_1, ..., s_m$, respectively. However, they are

* Gdańsk University of Technology, Faculty of Electronics, Telecommunications and Informatics, Gdańsk, Poland, e-mail: szyduraj@student.pg.gda.pl

DOI: https://doi.org/10.7494/dmms.2017.11.1-2.53

uniform in the sense that, if job j_i is executed on machine M_j, it takes $1/s_j$ time units to be completed. This refers to the situation where the machines are of different generations (old and slow, new and fast, etc.). In addition, we assume that some pairs of jobs cannot be processed on the same machine due to certain safety or spatial reasons. More precisely, we assume in this paper that underlying incompatibility graph G whose vertices are jobs and edges correspond to pairs of jobs being in conflict is bipartite. For example, such a situation holds if we have to transport cyanides together with acids. By definition, a load on any machine M_j forms an independent set (color) in G. Therefore, we will be using the terms job/vertex and color/load/independent set interchangeably in what follows.

So far, this model of scheduling has been studied in only two papers (Furmańczyk and Kubale, 2017a, 2017b). They considered standard uniform machines and assumed $C_{\max} = \max\{C_i : i = 1, 2, ..., n\}$ to be the criterion of optimality where C_i is the completion time of job j_i. Therefore, our present paper is partially based on the results of those articles.

The rest of the paper consists of two parts. The first part is theoretical and contains two sections. Since the problem of scheduling identical jobs on two machines has been solved by Pikies (manuscript in preparation), we review the known results concerning the scheduling of identical jobs on three or four standard uniform machines to minimize schedule length $C_{\max}$ and total completion time $\sum C_i$ in Section 2. In Section 3, we consider the problem of scheduling identical jobs on three or four uniform batch machines to minimize $\sum C_i$. Since the general problem is NP-hard, we focus on polynomial algorithms in both sections that guarantee optimal solutions in some special cases. This means that we do not consider herein special cases that can be approximated to within a fixed factor of optimality.

The second part of this paper is practical and consists of Section 4, in which we report on the computer experiments with an algorithm for the optimal scheduling of identical jobs with bicubic incompatibility graphs on three uniform machines.

2. STANDARD UNIFORM MACHINES

We begin with some basic notions concerning graph theory. Let $G = (V, E)$ be a connected graph. We say that G is *cubic* if it is a 3-regular graph (i.e., $\deg(v) = 3$ for each $v \in V$). A bipartite cubic graph is said to be *bicubic*. A bipartite graph is called *bisubcubic* if, for each v, we have $\deg(v) \leqslant 3$.

Analogously, G is *quartic* if it is 4-regular. A bipartite quartic graph is called *biquartic*, and a bipartite graph with property $\deg(v) \leqslant 4$ is called *bisubquartic*. By degree Δ of G, we mean the maximal degree among all vertices of G. In Figure 1, we show the smallest bicubic graph (a) and smallest biquartic graph (b).

Note that any bicubic (biquartic) graph can be 2-colored with partition sets of equal size; however, this is not true for bisubcubic (bisubquartic) graphs. Since all of these graphs are bipartite, they are easy colorable in linear time while traversing them in a depth-first search (DFS) manner. Generally speaking, any chromatic coloring corresponds to the $C_{\max}$ criterion, and any chromatic sum coloring (i.e., coloring that minimizes the sum of colors used) corresponds to the $\sum C_i$ criterion.

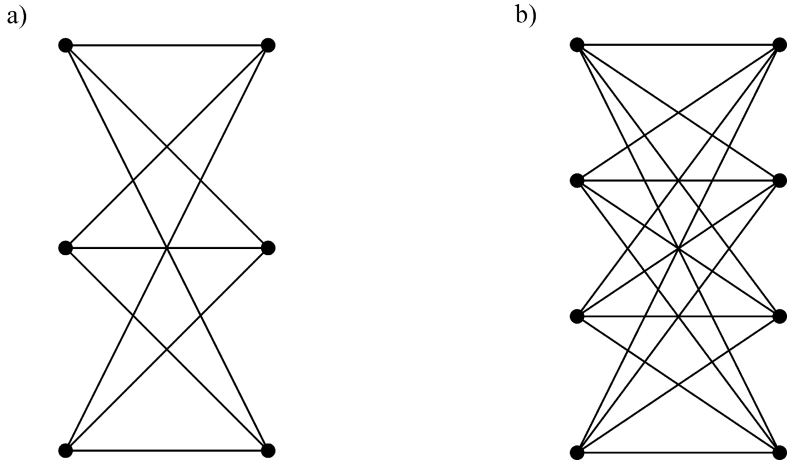

Fig. 1. *The smallest graphs: a) bicubic; b) biquartic*

Let $m = 4$, and let s_j be the speed of machine $M_j, j = 1, ..., 4$. Without a loss of generality, we assume that $s_1 \geqslant ... \geqslant s_4$. Our first theorem gives a negative result:

Theorem 1. (Furmańczyk and Kubale, 2017a) *Problem* $Q4|UET, bipartite|C_{\max}$ *is* NP*-hard even if* $s_1 = s_2 = s_3$.

Furmańczyk and Kubale (2017b) proved that, if incompatibility graph $G \neq K_{3,3}$ is bicubic, then 3-machine problem $Q3|UET, bicubic|C_{\max}$ can be solved to optimality in time $O(n^2)$. Therefore, our first positive result is formulated as follows:

Theorem 2. (Furmańczyk and Kubale, 2017b) *Problem* $Q3|UET, bicubic|C_{\max}$ *can be solved to optimality in time* $O(n^2)$.

We suppose that this result may be extended to bisubcubic graphs and the corresponding algorithm can be implemented to run in linear time. Therefore, we formulate our first hypothesis:

Conjecture 1. Problems $Q3|UET, bisubcubic|C_{\max}$ and $Q3|UET, bisubcubic|\sum C_i$ can be solved in $O(n)$ time.

The authors of (Furmańczyk and Kubale, 2017a) also considered bisubquartic graphs and $m = 4$ machines. They proved that, if $s_1 \geqslant 12s_2$ and $s_2 = s_3 = s_4$, then problem $Q4|UET, bisubquartic|C_{\max}$ can be solved to optimality in time $\mathrm{O}(n^{1.5})$. The same refers to the total completion time criterion. Therefore, we have the following:

Theorem 3. (Furmańczyk and Kubale, 2017a) *If* $s_1 \geqslant 12s_2$ *and* $s_2 = s_3 = s_4$ *then the problems* $Q4|UET, bisubquartic|C_{\max}$ *and* $Q4|UET, bisubquartic|\sum C_i$ *can be solved in* $O(n^{1.5})$ *time.*

Below, we present a simplified version of the corresponding algorithm.

Algorithm 1 Scheduling of bisubquartic graphs.

Input: Bisubquartic graph G and 4 uniform machines with speeds $s_1 \geqslant s_2 \geqslant s_3 \geqslant s_4$.
Output: Optimal schedule when $s_1 \geqslant 12s_2$ and $s_2 = s_3 = s_4$.

1) Find a maximum independent set I in G.
2) Let G' be a subgraph induced on the set of vertices $V(G) - I$.
3) If G' is isomorphic to $K_{3,3}$ then swap any vertex from G' with its neighbor in G which belongs to I.
4) Find an equitable coloring of G' using the methods presented in Chen and Yen (2012). Let (A, B, C) be the classes of this coloring.
5) Assign $M_1 \leftarrow I, M_2 \leftarrow A, M_3 \leftarrow B, M_4 \leftarrow C$.

3. BATCH UNIFORM MACHINES

A batch is a set of jobs that must be processed jointly. In general, we have two types of batch machines: s-batch and p-batch. In s-batching problems, the length of a load on M_j (and so, the finishing time of all jobs in the batch) is the sum of the processing times of all of the jobs in the batch. In p-batching problems, the length of a load on M_j is the maximum of the completion times of all jobs in the corresponding batch. In this paper, we will assume the p-batch model (*batch* for short). Moreover, we assume total completion time as the criterion of optimality.

Małafiejski *et al.* (2004) proved that the chromatic sum problem is NP-hard on bipartite graphs of degree Δ greater than or equal to 5. On the other hand, Kosowski (2009) showed that the number of colors used in a best chromatic sum coloring of any bipartite graph is bounded by $\lceil \Delta/2 \rceil + 1$. Easy calculations show that, when $s_1 = 2s_2 = 3s_3 = ...$, our scheduling problem reduces to chromatic sum coloring. For this reason, we begin with an NP-completeness theorem in a general case; i.e., where the degree of G is unbounded.

Theorem 4. (Małafiejski *et al.*, 2004; Kosowski, 2009) *Problem* $Qm|batch, UET, bipartite|\sum C_i$ *is* NP*-hard even if* $s_1 = 2s_2 = ... = ms_m$, *where* $m = \lceil \Delta/2 \rceil + 1$.

If the machine speeds are equal, our scheduling problem becomes easy, since any 2-coloring of G resolves the problem. Also, if G is bicubic or biquartic, then we can split its vertices into independent sets I_1 and I_2 of equal size and assign $n/2$ job vertices of I_1 to M_1 and the remaining $n/2$ jobs to M_2. It is easy to see that this would be an optimal solution. Therefore, we have:

Theorem 5. *If* $s_1 \geqslant s_2 \geqslant s_3 \geqslant s_4$ *then problems* $Q3|batch, UET, bicubic|\sum C_i$ *and* $Q4|batch, UET, biquartic|\sum C_i$ *can be solved in* $O(n)$ *time.*

Also, if $s_1 = s_2 \geqslant s_3$ and G is simply bipartite, then any 2-coloring of G resolves our problem. Since such a coloring can be obtained in time $O(n^2)$ at worst, we have:

Theorem 6. *If* $s_1 = s_2 \geqslant s_3$ *then problem* $Q3|batch, UET, bipartite|\sum C_i$ *can be solved in* $O(n^2)$ *time.*

From Małafiejski *et al.* (2004) and Kosowski (2009) we know that the chromatic sum problem is solvable in $O(n^2)$ if G is bisubquartic. Moreover, only three colors suffice to attain optimal coloring. Thus, we have the next polynomial result.

Theorem 7. (Małafiejski *et al.*, 2004; Kosowski, 2009) *If* $s_1 = 2s_2 = 3s_3$ *then problem* $Q3|batch, UET, bisubquartic|\sum C_i$ *can be solved in* $O(n^2)$ *time.*

We conclude our list of polynomially solvable special cases with a guess that the following holds true:

Conjecture 2. If $s_1 \geqslant s_2 \geqslant s_3$ then problem $Q3|batch, UET, bisubquartic|\sum C_i$ can be solved in $O(n^2)$ time.

At least this is true if $s_1 = s_2 \geqslant s_3$ by Theorem 6 and remains so if $s_1 \geqslant s_2 = s_3$. The latter follows from the following argument. We can find a maximum independent set I_1 in G in $O(n^{1.5})$ time and assign its vertices to M_1. Since G is bipartite, the remaining vertices of $G - I_1$ can be split in any way into two independent subsets and assigned to M_2 and M_3.

The theorems of Sections 2 and 3 are summarized in Table 1, which presents complexity presenting the state-of-art concerning the chromatic model of scheduling for bipartite incompatibility graphs.

Table 1. *The complexity of scheduling identical jobs with bipartite incompatibility graphs*

machines	**graph**	m	**speeds**	**goal**	**complexity**	**references**
standard	*bipartite*	4	$s_1 = s_2 = s_3 >> s_4$	$C_{\max}$	NP-hard	Furmańczyk and Kubale (2017a)
standard	*bicubic*	3	$s_1 \geqslant s_2 \geqslant s_3$	$C_{\max}$	$O(n^2)$	Furmańczyk and Kubale (2017b)
standard	*bisubquartic*	4	$s_1 \geqslant 12s_2$, $s_2 = s_3 = s_4$	$C_{\max}$	$O(n^{1.5})$	Furmańczyk and Kubale (2017a)
standard	*bicubic*	3	$s_1 \geqslant s_2 \geqslant s_3$	$\sum C_i$	$O(n^2)$	Furmańczyk and Kubale (2017b)
standard	*bisubquartic*	4	$s_1 \geqslant 12s_2$, $s_2 = s_3 = s_4$	$\sum C_i$	$O(n^{1.5})$	Furmańczyk and Kubale (2017a)
p-batch	*bipartite*	m	$s_1 = 2s_2 = 3s_3 = ...$	$\sum C_i$	NP-hard	Małafiejski *et al.* 2004
p-batch	*bicubic*	3	$s_1 \geqslant s_2 \geqslant s_3$	$\sum C_i$	$O(n)$	Theorem 5
p-batch	*biquartic*	4	$s_1 \geqslant s_2 \geqslant s_3 \geqslant s_4$	$\sum C_i$	$O(n)$	Theorem 5
p-batch	*bipartite*	3	$s_1 = s_2 \geqslant s_3$	$\sum C_i$	$O(n^2)$	Theorem 6
p-batch	*bisubquartic*	3	$s_1 = 2s_2 = 3s_3$	$\sum C_i$	$O(n^2)$	Małafiejski *et al.* 2004, Kosowski 2009

4. COMPUTER EXPERIMENTS

We have performed computer experiments for scheduling with bicubic incompatibility graphs to minimize schedule length. Notice that, in this case, any optimal solution minimizes the total completion time as well; i.e., we solved both the $Q3|UET, bicubic|C_{\max}$ and $Q3|UET, bicubic|\sum C_i$ problems. We used the polynomial algorithm described below.

Algorithm 2 Scheduling of bicubic graphs.

Input: Bicubic graph $G \neq K_{3,3}$ and three uniform machines with speeds $s_1 \geqslant s_2 \geqslant s_3$.
Output: Optimal schedule.

1) If $s_1 < s_2 + s_3$ then go to Step 5.
2) Find an (I, J)-coloring of graph G where I, J are partition sets.
3) Let $n_2 = \lceil .5ns_2/(s_2 + s_3) \rceil$. Split color class J into 2 subsets: B of size n_2 and C of size $n_3 = n/2 - n_2$ or B of size $n_2 - 1$ and C of size $n_3 = n/2 - n_2 + 1$ depending on which of the two partitions gives a better solution.
4) Assign $M_1 \leftarrow I, M_2 \leftarrow B, M_3 \leftarrow C$ and stop.
5) Calculate the approximate numbers of jobs (n_1, n_2, n_3) to be processed on M_1, M_2, M_3 in an ideal schedule as follows:

$$n_1 = ns_1/s, n_2 = ns_2/s, n_3 = ns_3/s, \text{ where } s = s_1 + s_2 + s_3.$$

6) Verify which of the following types of colorings:

$$[\lfloor n_1 \rfloor, \lceil n_2 \rceil, n - \lfloor n_1 \rfloor - \lceil n_2 \rceil], [\lceil n_1 \rceil, \lfloor n_2 \rfloor, n - \lceil n_1 \rceil - \lfloor n_2 \rfloor]$$

or

$$[\lceil n_1 \rceil, \lceil n_2 \rceil, n - \lceil n_1 \rceil - \lceil n_2 \rceil]$$

guarantees a better solution and call it OPT.
7) Let (A, B, C) be a coloring of G realizing OPT obtained by using a modified CLW method described in Furmańczyk and Kubale (2017b).
8) Assign $M_1 \leftarrow A, M_2 \leftarrow B, M_3 \leftarrow C$.

Furmańczyk and Kubale (2017b) proved the following.

Theorem 8. Algorithm 2 *runs in* $O(n^2)$ *time to produce an optimal schedule.*

As mentioned, this algorithm can be executed in $O(n^2)$. Basically, we first divide the graph into two equal partition sets I and J. This is possible, since any bicubic graph has two independent sets each with a size of $n/2$ (n is even due to the Handshaking Lemma). Then, we subdivide one partition set into two smaller partitions of sizes $|B|$ and $|C| = n/2 - |B|$, where $|B|$ is the desired size of the middle color class. Finally, we gradually move the vertices from the biggest to the smallest color class until we reach their final sizes. Experiments were made for machine speeds $s_1 = s_2 = s_3$ when the number of vertices to move was the biggest. Moving one vertex can be done in a worst-case time of $O(n)$. However, our experiments showed that, in a vast majority of cases, our algorithm ran in overall linear time. This is so because we can move more

than one vertex in one iteration. Hence, in our implementation, the first iteration had usually done most of the task already. The computer used for our computations was an HP Elitebook 2540p with an Intel Core i7-640LM. During the tests, only system processes were launched apart from our testing program. Of course, the testing program was executed as a single thread.

The graphs used for testing were generated using a modified Algorithm 1 from Steger and Wormald (1999). In a nutshell, we first randomly divide the vertices into two sets, each with a size of $n/2$. Each vertex has three places for incoming edges (henceforth called points). Next, we divide these points into two sets that correspond to the sets of vertices. Now, we pull one random point from the first set and one random point from the second and join them together as an edge (if it is legal to connect them). If an edge was added, we remove those points from the corresponding sets. We repeat this step until there are no points left. If a graph that comes out is disconnected or we are left only with points that cannot be connected (because joining them would make up a multiple edge), the algorithm is run again; however, the probability of this seems to tend to zero as n goes to infinity. As a rule, the process of generating a graph can be accomplished in $\mathrm{O}(n)$ time, provided that it does not have to be repeated.

Figure 2 visualizes the dataset. Figure 3 presents the average execution times, where the average is a mean of 20 n-vertex graphs for each even n in a range of [1000, 10000]. Figure 4 presents the generation times with respect to n for all graphs used in the experiments. The less-visible trend line probably represents the cases when the generating process had to be repeated, because either the final graph was disconnected or joining the last two points that remained would make up a multiple edge.

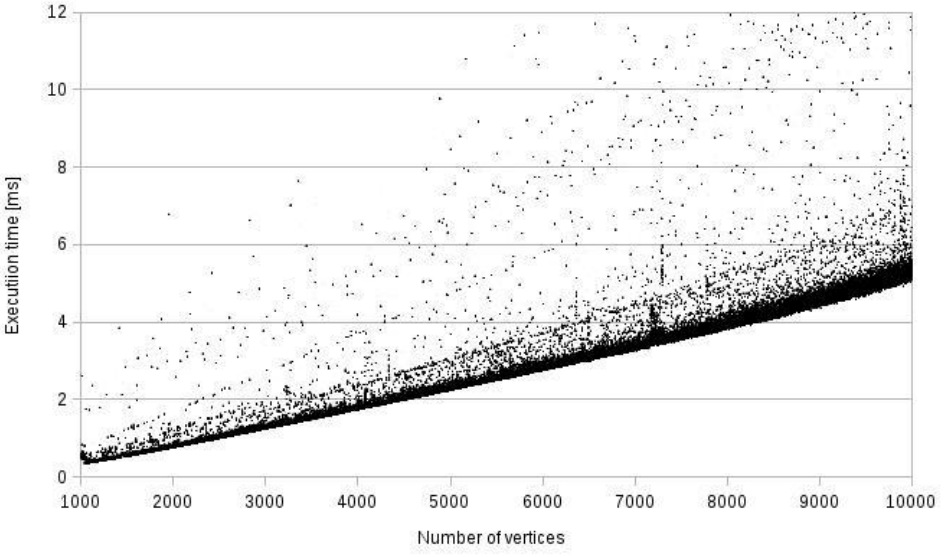

Fig. 2. *Running times of Algorithm 2*

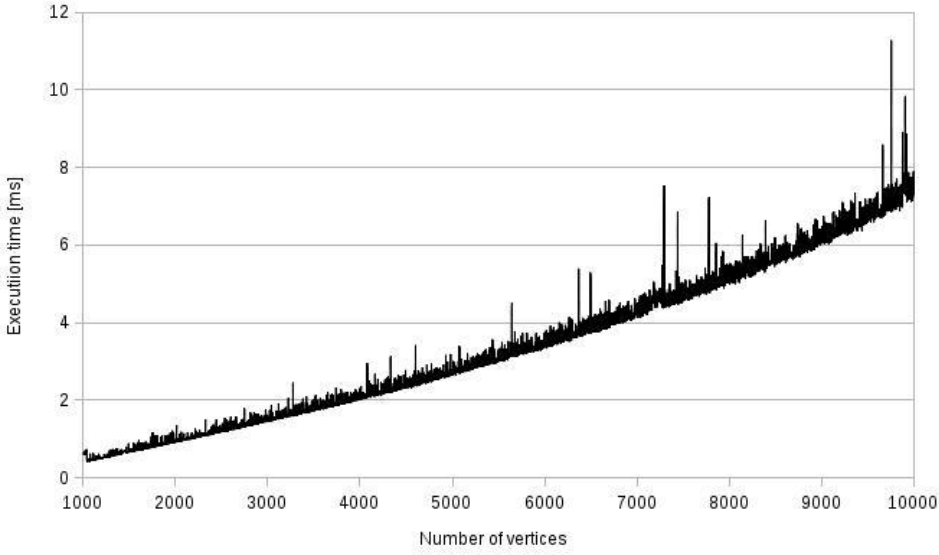

Fig. 3. *Average running times of Algorithm 2. Adjacent points are connected by a straight-line segment*

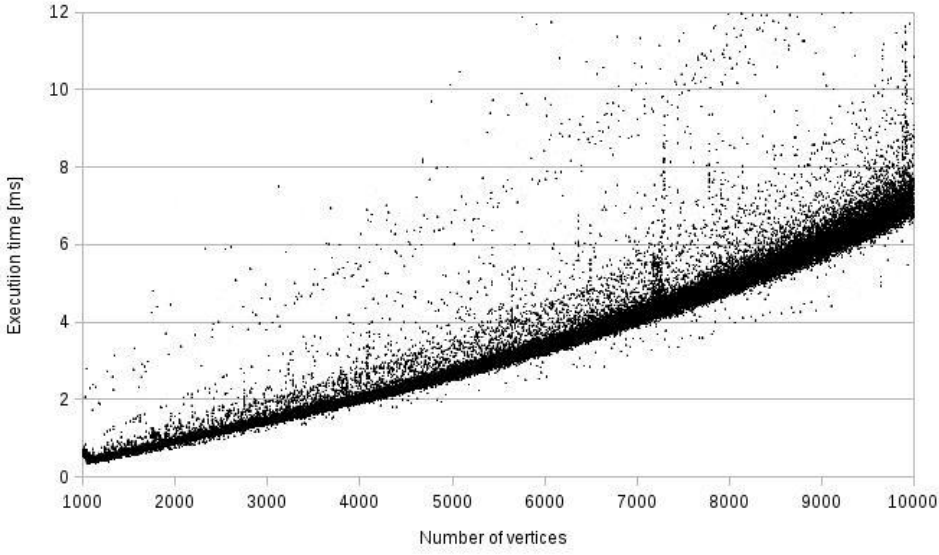

Fig. 4. *Times of generating random bicubic graphs*

5. FINAL REMARKS

In this article, we collected algorithmic results devoted to the problem of the scheduling of identical jobs on a few uniform machines subject to mutual exclusion constraints that follow from machine safety or spatial reasons. Since the general problem is NP-hard, we focused on polynomially solvable special cases involving a restricted number of machines, simplified structures of incompatibility graphs, machine speed limits, etc.

In the second part of the paper, we implemented a $O(n^2)$ algorithm for scheduling subject to bicubic incompatibility graphs in order to minimize schedule length. It appears that this algorithm is extremely fast. For example, huge graphs with several thousands of vertices are scheduled to optimality within several milliseconds. Frequently, a generation of such random graphs takes longer than their coloring.

REFERENCES

Chen, B.-L., Yen, C.-H., 2012. Equitable Δ-coloring of graphs, *Discrete Mathematics*. **312**(9), pp. 1512–1517.

Furmańczyk, H., Kubale, M., 2017a. Scheduling of unit-length jobs with bipartite incompatibility graphs on four uniform machines. *Bulletin of the Polish Academy of Sciences: Technical Sciences*, **65**(1), pp. 29–34.

Furmańczyk, H., Kubale, M., 2017b. Scheduling of unit-length jobs with cubic incompatibility graphs on three uniform machines. *Discrete Applied Mathematics* (in Press).

Kosowski, A., 2009. A note on the strength and minimum color sum of bipartite graphs. *Discrete Applied Mathematics*, **157**(11), pp. 2552–2554.

Małafiejski, M., Giaro, K., Janczewski, R., Kubale, M., 2004. Sum coloring of bipartite graphs with bounded degree. *Algorithmica*, **40**(4), pp. 235–244.

Steger, A., Wormald, N.C., 1999. Generating random regular graphs quickly. *Combinatorics, Probability and Computing*, **8**(4), pp. 377–396.