Discrete Applied Mathematics 234 (2018) 210–217

Contents lists available at ScienceDirect

Discrete Applied Mathematics

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

Scheduling of unit-length jobs with cubic incompatibility graphs on three uniform machines☆

Hanna Furmańczyk[a,*], Marek Kubale[b]

[a] Institute of Informatics, University of Gdańsk, Wita Stwosza 57, 80-952 Gdańsk, Poland

[b] Faculty of Electronics, Telecommunications and Informatics, Gdańsk University of Technology, Narutowicza 11/12, 80-233 Gdańsk, Poland

ARTICLE INFO

Article history:
Received 24 May 2015
Received in revised form 14 November 2015
Accepted 27 January 2016
Available online 22 February 2016

Keywords:
Cubic graph
Equitable coloring
NP-hardness
Polynomial algorithm
Scheduling
Uniform machine

ABSTRACT

In the paper we consider the problem of scheduling n identical jobs on 3 uniform machines with speeds s_1, s_2, and s_3 to minimize the schedule length. We assume that jobs are subjected to some kind of mutual exclusion constraints, modeled by a cubic incompatibility graph. We show that if the graph is 2-chromatic then the problem can be solved in $O(n^2)$ time. If the graph is 3-chromatic, the problem becomes NP-hard even if $s_1 > s_2 = s_3$. However, in this case there exists a 10/7-approximation algorithm running in $O(n^3)$ time. Moreover, this algorithm solves the problem almost surely to optimality if $3s_1/4 \leq s_2 = s_3$.

1. Introduction

Imagine you have to arrange a dinner for, say 30, people and you have at your disposal 3 round tables with different numbers of seats (not greater than 15). You know that each of your guests is in bad relations with exactly 3 other people. Your task is to assign the people to the tables in such a way that no two of them being in bad relations seat at the same table. In the paper we show how to solve this and related problems.

Our problem can be expressed as the following scheduling problem. Suppose we have n identical jobs $j_1, \ldots, j_n$, so we assume that they all have unit execution times, in symbols $p_i = 1$, to be processed on three non-identical machines M_1, M_2, and M_3. These machines run at different speeds s_1, s_2, and s_3, respectively. However, they are *uniform* in the sense that if a job is executed on machine M_i, it takes $1/s_i$ time units to be completed. It refers to the situation where the machines are of different generations, e.g. old and slow, new and fast, etc.

Our scheduling model would be trivial if all the jobs were compatible. Therefore we assume that some pairs of jobs cannot be processed on the same machine due to some technological constraints. More precisely, we assume that each job is in conflict with exactly three other jobs. Thus the underlying *incompatibility graph* G whose vertices are jobs and edges correspond to pairs of jobs being in conflict is cubic. For example, all graphs in our figures are cubic. The number of jobs n must be even, since the sum of degrees of all vertices in G, i.e. $3n$, must be even. A load L_i on machine M_i requires the

☆ This project has been partially supported by Narodowe Centrum Nauki under contract DEC-2011/02/A/ST6/00201.
* Corresponding author.
E-mail addresses: hanna@inf.ug.edu.pl (H. Furmańczyk), kubale@eti.pg.gda.pl (M. Kubale).

http://dx.doi.org/10.1016/j.dam.2016.01.036
0166-218X/

processing time $P(L_i) = |L_i|/s_i$, and all the jobs are ready for processing at the same time. By definition, each load forms an independent set (color) in G. Therefore, in what follows we will be using the terms job/vertex and color/independent set interchangeably. Since all tasks have to be executed, the problem is to find a 3-coloring, i.e. a decomposition of G into 3 independent sets I_1, I_2, and I_3 such that the schedule length $C_{\max} = \max\{P(I_i) : i = 1, 2, 3\}$ is minimized, in symbols $Q3|p_i = 1, G = cubic|C_{\max}$.

In this paper we assume three machines for the following reason. If there is only one machine then there is no solution. If there are two machines, the problem becomes trivial because it is solvable only if G is bipartite and it has only one solution since there is just one decomposition of G into sets I_1 and I_2, each of size $n/2$. If, however, there are three machines and G is 3-chromatic, our problem becomes NP-hard. Again, if G is 4-chromatic (and $m = 3$), there is no solution.

There are several papers devoted to chromatic scheduling in the presence of mutual exclusion constraints. Boudhar in [1, 2] studied the problem of batch scheduling with complements of bipartite and split graphs, respectively. Finke et al. [8] considered the problem with complements of interval graphs. Other models of batch scheduling with incompatibility constraints were studied in [5,6]. Our problem can also be viewed as a particular variant of scheduling with conflicts [7]. In all the papers the authors assumed identical parallel machines. However, to the best of our knowledge little work has been done on scheduling problems with uniform machines involved (cf. Li and Zhang [12]).

The rest of this paper is split into two parts depending on the chromaticity of cubic graphs. In Section 2 we consider 2-chromatic graphs. In particular, we give an $O(n^2)$-time algorithm for optimal scheduling of such graphs. Section 3 is devoted to 3-chromatic graphs. In particular, we give an NP-hardness proof and an approximation algorithm with good performance guarantee. Our algorithm runs in $O(n^3)$ time to produce a solution of value less than 10/7 times optimal, provided that $s_1 > s_2 = s_3$. Moreover, this algorithm solves the problem almost surely to optimality if $3s_1/4 \leq s_2 = s_3$. Finally, we discuss possible extensions of our model to arbitrary job lengths, to disconnected graphs, and to more than three machines.

2. Scheduling of 2-chromatic graphs

We begin with introducing some basic notions concerning graph coloring. A graph $G = (V, E)$ is said to be *equitably k-colorable* if and only if its vertex set can be partitioned into independent sets $V_1, \ldots, V_k \subset V$ such that $\|V_i| - |V_j\| \leq 1$ for all $i, j = 1, \ldots, k$. The smallest k for which G admits such a coloring is called the *equitable chromatic number* of G and denoted $\chi_=(G)$. Graph G has a *semi-equitable k-coloring*, if there exists a partition of its vertices into independent sets $V_1, \ldots, V_k \subset V$ such that one of these subsets, say V_i, is of size $\notin \{\lfloor n/k \rfloor, \lceil n/k \rceil\}$, and the remaining subgraph $G - V_i$ is equitably $(k-1)$-colorable. In the following we will say that graph G has $(V_1, \ldots, V_k)$-coloring to express explicitly a partition of V into k independent sets. If, however, only cardinalities of color classes are important, we will use the notation $[|V_1|, \ldots, |V_k|]$.

Let us recall some basic facts concerning colorability of cubic graphs. It is well known from Brooks theorem [3] that for any cubic graph $G \neq K_4$ we have $\chi(G) \leq 3$, where $\chi(G)$ is the classical chromatic number of G and K_4 is the complete graph on four vertices. On the other hand, Chen et al. [4] proved that every 3-chromatic cubic graph can be equitably colored without introducing a new color. Moreover, since a connected cubic graph G with $\chi(G) = 2$ is a bipartite graph with partition sets of equal size, we have the equivalence of the classical and equitable chromatic numbers for 2-chromatic cubic graphs. Since the only cubic graph for which the chromatic number is equal to 4 is K_4, we have

$$2 \leq \chi_=(G) = \chi(G) \leq 4 \tag{1}$$

for any cubic graph. Moreover, from (1) it follows that for any cubic graph $G \neq K_4$, we have

$$n/3 \leq \alpha(G) \leq n/2 \tag{2}$$

where $\alpha(G)$ is the independence number of G. Note that the upper bound is tight only if G is bipartite.

Let $\mathcal{Q}_k$ denote the class of connected k-chromatic cubic graphs and let $\mathcal{Q}_k(n) \subset \mathcal{Q}_k$ stand for the subclass of cubic graphs on n vertices, $k = 2, 3, 4$. Clearly, $\mathcal{Q}_4 = \{K_4\}$. In what follows we will call the graphs belonging to $\mathcal{Q}_2$ *bicubic*, and the graphs belonging to $\mathcal{Q}_3$- *tricubic*.

As mentioned, if G is bicubic then any 2-coloring of it is equitable and there may be no equitable 3-coloring (cf. $K_{3,3}$ shown in Fig. 1). On the other hand, all graphs in $\mathcal{Q}_2(n)$ have a semi-equitable 3-coloring of type $[n/2, \lceil n/4 \rceil, \lfloor n/4 \rfloor]$. Moreover, they are easy colorable in linear time while traversing them in a depth-first search (DFS) manner.

Let s_i be the speed of machine M_i for $i = 1, 2, 3$, and let $s = s_1 + s_2 + s_3$. Without loss of generality we assume that $s_1 \geq s_2 \geq s_3$. If there are just 6 jobs to schedule then the incompatibility graph $G = K_{3,3}$ and there is only one decomposition of it into 3 independent sets shown in Fig. 1(a), as well as there is only one decomposition of G into 2 independent sets shown in Fig. 1(b), of course up to isomorphism. The length of minimal schedule is $\min\{\max\{3/s_1, 2/s_2, 1/s_3\}, 3/s_2\}$. Therefore, we assume that our graphs have at least 8 vertices.

By an *ideal schedule* we mean a schedule in which:

(i) machine M_1 performs as many jobs as possible and M_2 and M_3 finish at the same time, if $s_1 \geq s_2 + s_3$, or
(ii) machines M_1, M_2, and M_3 all finish at the same time, if $s_1 < s_2 + s_3$.

An example of ideal schedule is shown in Fig. 2(a).

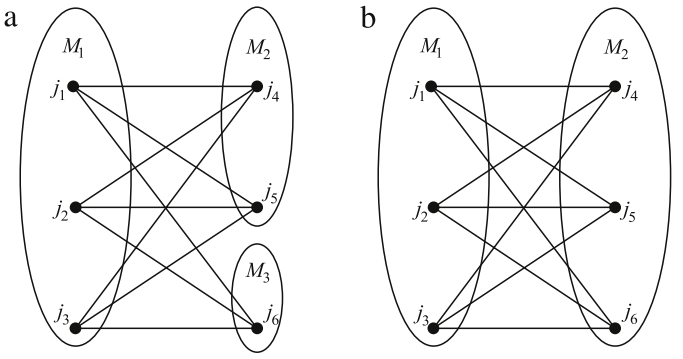

Fig. 1. Two decompositions of $K_{3,3}$: (a) into 3 independent sets, (b) into 2 independent sets.

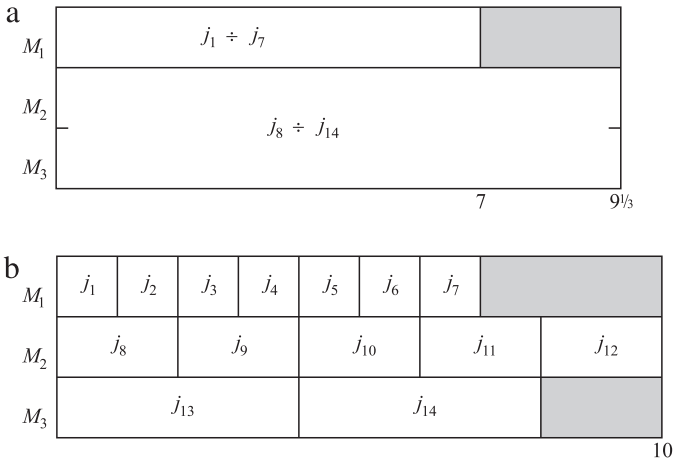

Fig. 2. Gantt chart of a schedule for $n = 14$ and $s_1 = 2s_2 = 4s_3$: (a) ideal; (b) optimal.

Lemma 1. *If $s_1 \geq s_2 \geq s_3$ then the length of ideal schedule is determined by the length of schedule on M_2, that is*

$$C(M_2) = \begin{cases} \frac{1}{2}n/(s_2 + s_3) & \text{if } s_1 \geq s_2 + s_3, \\ n/s & \text{if } s_1 < s_2 + s_3. \end{cases}$$

Proof. (i) If $s_1 \geq s_2 + s_3$ then, by definition, as many as possible jobs should be allocated to M_1. The maximal number of jobs on the first machine is $n_1 = n/2$. The remaining $n/2$ jobs should be assigned to M_2 and M_3. Suppose we merge M_2 with M_3 into one machine M_{23}. Its speed is $(s_2 + s_3)/2$. Hence the time needed to process $n/2$ jobs on M_{23} is $\frac{1}{2}n \cdot 2/(s_2 + s_3)$. Now, if we split M_{23} into M_2 and M_3, we get $C(M_2) = C(M_3) = \frac{1}{2}n/(s_2 + s_3)$.

(ii) If $s_1 < s_2 + s_3$ then the total load of n jobs must be balanced evenly among all the machines. Hence the time needed to process all the jobs is $\frac{1}{3}n \cdot 3/(s_1 + s_2 + s_3) = n/s$. □

Obviously, the numbers of jobs on machines must be integer. Therefore, we must check which of the three variants of a schedule, i.e. with round-up and/or round-down on M_1 and M_2, guarantees a better solution. This leads to the following `Algorithm 1` for optimal scheduling of bicubic graphs.

A crucial point of `Algorithm 1` is Step 7 where we use a modified procedure due to Chen, Lih, and Wu [4], which we call a `CLW` procedure. In the nutshell, we first check if there is a pair of vertices one from the largest and the other from the smallest class whose colors can be simply swapped. If there is no such pair, we have to consider such a bipartite subgraph that swapping the vertices between its partition sets (possibly with another subset being involved in the swapping) results in decreasing the *width* of coloring, i.e. the difference between the cardinality of the largest and smallest independent set. `CLW` procedure was used by its authors to prove that every tricubic graph can be equitably colored without introducing a new color. The proof relies on successive decreasing the width of coloring one by one until a coloring is equitable. At first, they choose a vertex that must be recolored. Then the three authors consider numerous cases. In most of them the above-mentioned swappings between all three color classes are required. For details, we refer the reader to [4].

Actually, their procedure works for every 3-coloring of any bicubic graph, except for $K_{3,3}$. More precisely, in Step 7 of `Algorithm 1`, where we want to receive an (A, B, C)-coloring (named as OPT) with cardinalities of color classes

Algorithm 1 Scheduling of bicubic graphs

Input: Graph $G \in \mathcal{Q}_2(n)$, $G \neq K_{3,3}$ and machine speeds s_1, s_2, s_3 such that $s_1 \geq s_2 \geq s_3$.
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. Split color J into 2 subsets: B of size $n_2 = \lceil .5n/(s_2 + s_3) \rceil$ and C of size $n_3 = n/2 - n_2$.
4. Assign $M_1 \leftarrow I, M_2 \leftarrow B, M_3 \leftarrow C$ and stop.
5. Calculate approximate numbers of jobs n_1, n_2, and n_3 to be processed on M_1, M_2, and 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:
$$[\lfloor n_1 \rfloor, \lceil n_2 \rceil, n - \lfloor n_1 \rfloor - \lceil n_2 \rceil], \ [\lceil n_1 \rceil, \lfloor n_2 \rfloor, n - \lceil n_1 \rceil - \lfloor n_2 \rfloor] \text{ or } [\lceil n_1 \rceil, \lceil n_2 \rceil, n - \lceil n_1 \rceil - \lceil n_2 \rceil]$$
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 Procedure 1.
8. Assign $M_1 \leftarrow A, M_2 \leftarrow B, M_3 \leftarrow C$.

$|A| \geq |B| \geq |C|$, we have to start with 2-coloring of bicubic graph G, i.e. (I, J)-coloring, where I, J are its partition sets. Next we split the color class J into two: B of cardinality $|B|$ and C' of size $n/2 - |B|$. Hence, we initially have (A', B, C')-coloring with $|A'| = n/2$, $|C'| = n/2 - |B|$, where the largest class is clearly A', while the smallest class is C'. If this coloring with the width of $|B|$ is not the desirable (A, B, C)-coloring with the width of $|A| - |C|$, then we use CLW for decreasing the width from $|B|$ to $|A - C|$. Let us notice that such a width decreasing step is applied only to the first and the third color class, without changing the cardinality of the second class which is still equal to $|B|$. The whole modified CLW procedure is given below as Procedure 1. The complexity of modified CLW is the same as the complexity of the original CLW procedure for making any 3-coloring of tricubic graph equitable, namely $O(n^2)$. This is so because the part of the algorithm responsible for decreasing the width of coloring by one may be done in linear time. Since this step must be repeated at most $n/6$ times, the complexity of modified CLW procedure follows. This complexity dominates the running time of Algorithm 1.

Procedure 1 Modified CLW algorithm

Input: Graph $G \in \mathcal{Q}_2(n)$, $G \neq K_{3,3}$ and integers $a \geq b \geq c$ such that $a + b + c = n$.
Output: (A, B, C)-coloring of G such that $|A| = a$, $|B| = b$, $|C| = c$.

1. Find an (I, J)-coloring of graph G.
2. Split J into 2 subsets: B of size b and C of size $n/2 - b$.
3. While $|C| < c$ do
decrease the width of coloring by one using the CLW method [4].

The above considerations lead us to the following

Theorem 1. Algorithm 1 *runs in* $O(n^2)$ *time to produce an optimal schedule.* □

3. Scheduling of 3-chromatic graphs

First of all notice that if $s_1 = s_2 = s_3$ then the scheduling problem becomes trivial since any equitable coloring of G solves the problem to optimality. Therefore we assume that only two possible speeds are allowed for machines to run, more precisely that $s_1 > s_2 = s_3$.

In the following we take advantage of the following

Lemma 2 (*Furmańczyk, Kubale [10]*). *Let* $G \in \mathcal{Q}_3(n)$ *and let* $k = n/10$, *where* $10|n$. *The problem of deciding whether* G *has a semi-equitable coloring of type* $[4k, 3k, 3k]$ *is* NP*-complete.* □

Now we are ready to prove

Theorem 2. *The* $Q3|p_i = 1, G \in \mathcal{Q}_3(n)|C_{\max}$ *problem is* NP*-hard even if* $s_1 > s_2 = s_3$.

Proof. In the proof we will use a reduction of the coloring problem from Lemma 2 to our scheduling problem.

So suppose that we have a tricubic graph G on $n = 10k$ vertices and we want to know whether there exists a 3-coloring of G of type $[4k, 3k, 3k]$. Given such an instance we construct the following instance for a scheduling decision problem: machine speeds for M_1, M_2, and M_3 are $s_1 = 4/3, s_2 = s_3 = 1$ and the limit on schedule length is $3k$. The question is whether there is a schedule of length at most $3k$. The membership of this problem in class NP is obvious.

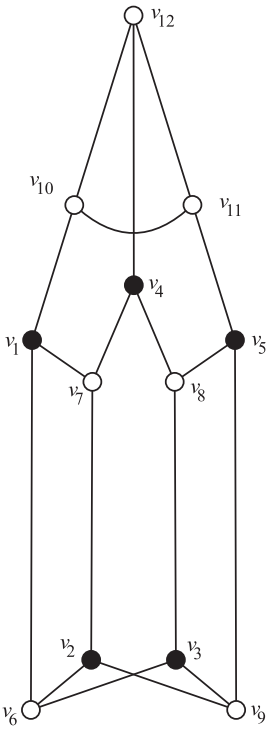

Fig. 3. Graph G for which the `Greedy` procedure (with ties broken by choosing the vertex with smallest index) finds an independent set I (vertices in black) such that $G - I$ contains K_3.

If there is a schedule of length $\leq 3k$ then it is of length exactly $3k$ since it cannot be shorter. Such a schedule implies the existence of a semi-equitable coloring of G of type $[4k, 3k, 3k]$.

If G has a coloring of type $[4k, 3k, 3k]$ then our scheduling problem has clearly a solution of length $3k$.

The NP-hardness of $Q3|p_i = 1, G \in \mathcal{Q}_3(n)|C_{\max}$ follows from the fact that its decision version is NP-complete. □

Since our scheduling problem is NP-hard, we have to propose an approximation algorithm for it. The general idea behind our heuristics is to find a big bipartizing independent set in G and allocate its job vertices to the fastest machine M_1. Then the remaining vertices are split evenly among M_2 and M_3. If, however, we fail to find such an independent set then we look for a semi-equitable or equitable coloring of G depending on the relative speeds of M_1 and M_2.

Procedure `Greedy` repeatedly chooses a vertex v of minimum degree, adds it to its current independent set and then deletes v and all its neighbors. Its complexity is linear. The following `Procedure 2` gives a more formal description of it.

Note that `Greedy` does not guarantee that $G - I$ is bipartite. It may happen that there remain some odd cycles in the subgraph, even if a big independent set is found. An example of such situation is given in Fig. 3. Nevertheless, the authors proved in [11] that given a graph $G \in \mathcal{Q}_3(n)$ with $\alpha(G) \geq 0.4n$, there exists an independent set I of size k in G such that $G - I$ is bipartite for $\lfloor (n - \alpha(G))/2 \rfloor \leq k \leq \alpha(G)$.

Now we have to prove that if for an independent set I the inequality $|I| \geq 0.4n$ holds and graph $G - I$ is bipartite then $G - I$ is equitably 2-colorable. Indeed, assume that $|I| = 0.4n$. Notice that $0.6n$ vertices of $G - I$ induce binary trees (some of them may be trivial) and/or graphs whose 2-core is equibipartite (even cycle possibly with chords). Note that deleting an independent set I of cardinality $0.4n$ from a cubic graph G means also that we remove $1.2n$ edges from the set of all $1.5n$ edges of G. The resulting graph $G - I$ has $0.6n$ vertices and $0.3n$ edges. Let d_i, $0 \leq i \leq 3$, be the number of vertices in $G - I$ of degree i. Certainly, $d_0 + \cdots + d_3 = 0.6n$. Since the number of edges is half of the number of vertices, the number of isolated vertices, d_0, is equal to $d_2 + 2d_3$. If $d_0 = 0$, then $G - I$ is a perfect matching and its equitable coloring is obvious. Suppose that $d_0 > 0$. Let S denote the set of isolated vertices in $G - I$. Let us consider subgraph $G - I - S$. Each vertex of degree 3 causes the difference between cardinalities of color classes which is ≤ 2, similarly each vertex of degree 2 causes the difference at most 1. The difference between the cardinalities of color classes in any coloring fulfilling these conditions does not exceed $d_2 + 2d_3$ in $G - I - S$. Thus, the appropriate assignment of colors to isolated vertices in S makes the whole graph $G - I$ equitably 2-colored. Therefore, an equitable coloring of $G - I$ required in Step 3 of `Algorithm 2` can be obtained as follows. First we color non-isolated vertices greedily by using for example a `DFS` method. In the second phase we color isolated vertices with this color that has been used fewer times in the first phase. This can be accomplished in $O(n)$ time.

However, the most time consuming is Step 2, where the `FKR` procedure is invoked. This procedure is too complicated to be described here. The general idea is as follows: given $G \in \mathcal{Q}_3(n)$ and an independent set I of size at least $0.4n$ such that

Algorithm 2 Scheduling of tricubic graphs

Input: Graph $G \in \mathcal{Q}_3(n)$ and machine speeds s_1, s_2, s_3 such that $s_1 > s_2 = s_3$.
Output: Suboptimal schedule.

1. Apply procedure `Greedy` (described in `Procedure 2`) to find an independent set I of G. If $|I| < 0.4n$ then go to Step 5.
2. If $G - I$ is not bipartite then apply procedure `FKR` (cf. [11]) to get an independent set A, $|A| = |I|$, which bipartizes G and put $I = A$.
3. Find an equitable 2-coloring (B, C) of $G - I$.
4. If $s_1 \geq 2s_2$ then assign $M_1 \leftarrow I, M_2 \leftarrow B, M_3 \leftarrow C$ and stop else go to Step 6.
5. Find any 3-coloring of G (cf. [13]) and apply procedure `CLW` in order to obtain an equitable coloring (A, B, C) of G. Go to Step 9.
6. Calculate approximate numbers of jobs n_1, n_2, and n_3 to be processed on M_1, M_2, and M_3 in an ideal schedule, as follows:
$$n_1 = ns_1/s, n_2 = n_3 = ns_2/s, \text{ where } s = s_1 + s_2 + s_3.$$
7. Verify which of the following types of colorings guarantees a better legal solution:
$$[\lfloor n_1 \rfloor, \lceil n_2 \rceil, n - \lfloor n_1 \rfloor - \lceil n_2 \rceil] \text{ or } [\lceil n_1 \rceil, \lfloor n_2 \rfloor, n - \lceil n_1 \rceil - \lfloor n_2 \rfloor]$$
If this is the first type then let $n^* = \lfloor n_1 \rfloor$ else $n^* = \lceil n_1 \rceil$.
8. If $n^* < |I|$ then starting from a semi-equitable coloring (A, B, C) find a semi- equitable/equitable coloring (A', B', C'), where $|A'| = n^*$, using the modified `CLW` procedure. Treat the new 3-coloring as (A, B, C).
9. Assign $M_1 \leftarrow A, M_2 \leftarrow B, M_3 \leftarrow C$.

Procedure 2 `Greedy`

Input: Graph $G \in \mathcal{Q}_3(n)$.
Output: Independent set I of G.

1. Set $I = \emptyset$.
2. While $V(G) \neq \emptyset$ do
set $G = G - N[v]$ and $I = I \cup \{v\}$, where v is a minimum degree vertex in G and $N[v]$ is its closed neighborhood.

$G - I$ is 3-chromatic, we transform it step by step into an independent set I' such that $|I'| = |I|$ and $G - I'$ is 2-chromatic (see [11] for details). Since one step of swapping two vertices between I and $V - I$ requires $O(n^2)$ time, the complexity of `FKR` is $O(n^3)$.

The above considerations lead us to the following

Theorem 3. `Algorithm 2` *runs in* $O(n^3)$ *time.* □

Note that if there are just 6 jobs to schedule then the incompatibility graph $G = TP$, where TP is the triangular prism shown in Fig. 4. There is only one decomposition of TP into 3 independent sets and the length of minimal schedule is $2/s_2$. Similarly, if $n = 8$ then each of tricubic graphs has a 3-coloring of type [3, 2, 2] only, and the length of minimal schedule is $\max\{3/s_1, 2/s_2\}$. If $n = 10$ then all we can get is a 3-coloring of type [4, 4, 2] or a 3-coloring of type [4, 3, 3]. The latter leads to an optimal solution of value $\max\{4/s_1, 3/s_2\}$. If, however, $n = 12$ then three types of colorings are possible: a non-equitable with [5, 5, 2], a semi-equitable with [5, 4, 3] and an equitable of type [4, 4, 4]. It is easy to see that only the last one guarantees optimal solution whose length is $4/s_2$. The reader may check by inspection that all the above mentioned tricubic graphs on 12 or fewer vertices are scheduled to optimality by `Algorithm 2`. Having this in mind we shall prove two facts concerning the performance guarantees for this algorithm.

Theorem 4. `Algorithm 2` *returns a solution of value less than* $\frac{10}{7}C^*_{\max}$.

Proof. Let $\text{Alg}_2(G)$ be the length of a schedule produced by `Algorithm 2` when applied to incompatibility graph G, and let $C^*_{\max}(G)$ be the length of an optimal schedule.

If $s_1 \geq 2s_2$ then it is natural to load as many jobs as possible on the fastest machine M_1 (cf. Lemma 1 and Fig. 2). By inequality (2) the maximal possible number of jobs on M_1 is less than $n/2$, because G is tricubic. Therefore, the schedule length on M_1 is less than $\frac{1}{2}n/s_1 \leq \frac{1}{4}n/s_2$. In the ideal solution the remaining jobs must be split evenly between M_2 and M_3, because $s_2 = s_3$ (cf. Fig. 2(a)). This means that optimal schedule cannot be shorter than $\lceil (n+1)/4 \rceil / s_2$ on M_2. Hence $C^*_{\max}(G) \geq \lceil (n+1)/4 \rceil / s_2$. On the other hand, in the worst case `Algorithm 2` returns a schedule corresponding to an equitable coloring of G (Step 5), (cf. [13]) which means that $\text{Alg}_2(G) \leq \lfloor (n+1)/3 \rfloor / s_2$. Therefore

$$\frac{\text{Alg}_2(G)}{C^*_{\max}(G)} \leq \frac{\lfloor (n+1)/3 \rfloor / s_2}{C^*_{\max}(G)} \leq \frac{\lfloor (n+1)/3 \rfloor / s_2}{\lceil (n+1)/4 \rceil / s_2} < \frac{(n+1)/3}{(n+1)/4} = \frac{4}{3}.$$

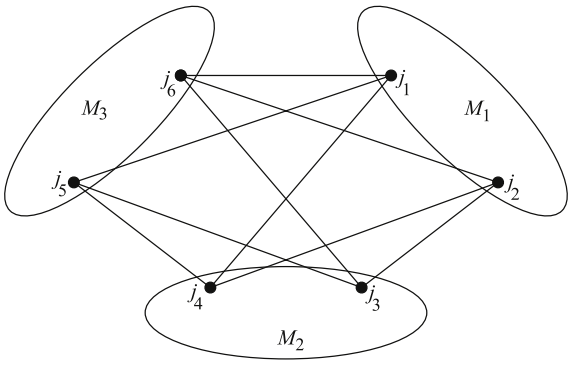

Fig. 4. The prism TP and its decompositions into 3 independent sets.

If $s_1 < 2s_2$ then the faster machine M_1 performs the bigger difference between the worst and best case is. In the worst case $s_1 \cong 2s_2$. By Lemma 1 the length of ideal schedule is n/s. Therefore, $C^*_{\max}(G) \geq \frac{n}{s}$. As previously, at worst our algorithm produces a schedule based on equitable coloring of G whose length is at most $\lfloor (n+1)/3 \rfloor / s_2$. Hence $\mathrm{Alg}_2(G) \leq \lfloor (n+1)/3 \rfloor / s_2$ and

$$\frac{\mathrm{Alg}_2(G)}{C^*_{\max}(G)} \leq \frac{\lfloor (n+1)/3 \rfloor / s_2}{C^*_{\max}(G)} \leq \frac{\lfloor (n+1)/3 \rfloor / s_2}{n/s} = \frac{s \lfloor (n+1)/3 \rfloor}{s_2 n}$$
$$< \frac{4s_2 \lfloor (n+1)/3 \rfloor}{s_2 n} \leq \frac{4 \cdot 5}{14} = \frac{10}{7},$$

since `Algorithm 2` performs optimally for $n \leq 12$ and $\lfloor (n+1)/3 \rfloor / n$ has maximum as $n = 14$. □

Theorem 5. *If* $3s_1/4 \leq s_2 = s_3$ *then* `Algorithm 2` *almost always returns an optimal solution.*

Proof. Frieze and Suen [9] showed that procedure `Greedy` finds an independent set of size $|I| \geq 0.432n - \epsilon n$ in almost all cubic graphs on n vertices, where ϵ is any constant greater than 0. In practice this means that a random graph from $\mathcal{Q}_3(n)$ is very likely to have an independent set of size $|I| \geq 0.4n$ and the probability of this fact tends to 1 as n tends to infinity. Notice that if it is really the case then calculated in Step 7 the optimal number of jobs to be allocated to M_1 fulfills $n^* < |I|$. Therefore `Algorithm 2` at first finds in Steps 2 and 3 a semi-equitable coloring of type $[|I|, \lceil (n - |I|)/2 \rceil, \lfloor (n - |I|)/2 \rfloor]$ and then transforms it into a semi-equitable coloring of type $[n^*, \lceil (n - n^*)/2 \rceil, \lfloor (n - n^*)/2 \rfloor]$ in Step 8. This completes the proof. □

4. Final remarks

Can our results be generalized without changing the complexity status of the scheduling problem? The answer is ... sometimes. Let us consider bicubic graphs for example. If arbitrary job lengths are allowed then the problem $Q\,3|G \in \mathcal{Q}_2|C_{\max}$ becomes NP-hard even if $s_1 = 2s_2 = 2s_3$. In fact, let I_1, I_2 be a decomposition of G and suppose that the processing time $P(I_1) = 2P(I_2)$. Then all the jobs of I_1 should be assigned to M_1, which results in a schedule of length $C(M_1) = P(I_1)$ on machine M_1. This schedule length equals $C^*_{\max}$ if and only if there is partition of the remaining jobs. Thus a solution to our scheduling problem solves an NP-complete PARTITION problem.

On the other hand, if all n jobs are identical but G is disconnected bicubic and $K_{3,3}$-free then `Algorithm 1` can be modified to obtain an optimal schedule in $O(n^2)$ time. First, we treat $G = G_1 \cup G_2 \cup \cdots \cup G_k$ as a connected graph and calculate the color sizes, say n_1, n_2, and n, where $n_1 + n_2 + n_3 = n$, that guarantee an optimal solution for G. Next, for each $i = 1, \ldots, k$ we split G_i into independent sets A_i, B_i and C_i, so that $\sum_{j=1}^{i} |B_j| / \sum_{j=1}^{i} |G_j|$ is as close to n_2/n as possible, where $|G_j|$ is the order of subgraph G_j. The same should hold for sets A_i and C_i with n_1/n and n_3/n, respectively. Similarly, we can extend `Algorithm 2` to deal with disconnected tricubic graphs in $O(n^3)$ time.

Finally note that if there are more than three machines then our problem remains NP-hard. To show this suppose that $m = 4$ and that machine M_4 is extremely slow, i.e. $s_1 > s_2 = s_3 \gg s_4 \cong 0$ (say $s_4 \ll 4s_2/n$). Then none of jobs should be assigned to M_4. Hence, scheduling on the four machines is NP-hard by the same argument that was used in order to prove Theorem 2.

Acknowledgments

The authors thank the anonymous reviewers for their careful reading of our manuscript and their many insightful comments and suggestions.

References

[1] M. Boudhar, Scheduling a batch processing machine with bipartite compatibility graphs, Math. Methods Oper. Res. 57 (2003) 513–527.
[2] M. Boudhar, Scheduling on a batch processing machine with split compatibility graphs, J. Math. Model. Algorithms 4 (2005) 391–407.
[3] R.L. Brooks, On colouring the nodes of a network, Proc. Cambridge Philos. Soc. Math. Phys. Sci. 37 (1941) 194–197.
[4] B.-L. Chen, K.W. Lih, P.L. Wu, Equitable coloring and the maximum degree, Eur. J. Combinatorics 15 (1994) 443–447.
[5] M. Demange, D. de Werra, J. Monnot, V.Th. Paschos, Time slot scheduling of compatible jobs, J. Sched. 10 (2007) 111–127.
[6] D. de Werra, M. Demange, J. Monnot, V.Th. Paschos, A hypocoloring model for batch scheduling, Disc. Appl. Math. 146 (2005) 3–26.
[7] G. Even, M.M. Haldórson, L. Kaplan, D. Ron, Scheduling with conflicts: online and offline algorithms, J. Sched. 12 (2009) 199–224.
[8] G. Finke, V. Jost, M. Queyranne, A. Sebő, Batch processing with interval graph compatibilities between tasks, Disc. Appl. Math. 156 (2008) 556–568.
[9] A. Frieze, S. Suen, On the independence number of random cubic graphs, Random Graphs Struct. 5 (1994) 649–664.
[10] H. Furmańczyk, M. Kubale, Equitable coloring of corona products of cubic graphs is harder than the ordinary coloring, Ars Math. Contemp. 10 (2016) 333–347.
[11] H. Furmańczyk, M. Kubale, S.P. Radziszowski, On bipartization of cubic graphs by removal of an independent set, Disc. Appl. Math. (2015) http://dx.doi.org/10.1016/j.dam.2015.10.036.
[12] S.-S. Li, Y.-Z. Zhang, Serial batch scheduling on uniform parallel machines to minimize total completion time, Inf. Process. Lett. 114 (2014) 692–695.
[13] S. Skulrattanakulchai, Δ-list vertex coloring in linear time, LNCS 2368 (2002) 240–248.