Ann Oper Res
DOI 10.1007/s10479-017-2503-6

ORIGINAL PAPER

Partially concurrent open shop scheduling with integral preemptions

Hagai Ilani[1] · **Elad Shufan**[2] · **Tal Grinshpoun**[3]

Abstract Partially-concurrent open shop scheduling (PCOSS) was recently introduced as a common generalization of the well-known open shop scheduling model and the concurrent open shop scheduling model. PCOSS was shown to be NP-hard even when there is only one machine and all operations have unit processing time. In the present paper we study PCOSS problems with integral processing times that allow preemptions at integral time points. A special and simple subclass of the problems at focus is that of unit processing times, which is considered separately. For these two cases a schedule is related to the colouring of a graph called the conflict graph, which represents the operations that cannot be performed concurrently. This enables us to extract insights and solutions from the well-studied field of graph colouring and apply them to the recently introduced PCOSS model. We then focus on two special cases of the problem—the case where the conflict graph is perfect, and the case of uniform PCOSS, in which all the jobs, including their conflicts, are identical. The development of the PCOSS model was motivated from a real-life timetabling project of assigning technicians to a fleet of airplanes. The latter case of uniform PCOSS correlates to instances in which the fleet of airplanes is homogeneous.

Keywords Open shop scheduling · PCOSS · Integral processing times · Integral preemption · Graph colouring · Technician timetabling

✉ Tal Grinshpoun
talgr@ariel.ac.il

Hagai Ilani
hagai@sce.ac.il

Elad Shufan
elads@sce.ac.il

[1] Department of Industrial Engineering and Management, SCE – Shamoon College of Engineering, Ashdod, Israel

[2] Physics Department, SCE – Shamoon College of Engineering, Ashdod, Israel

[3] Department of Industrial Engineering and Management, Ariel University, Ariel, Israel

Published online: 04 May 2017

1 Introduction

A *partially-concurrent open shop scheduling* (PCOSS) problem (Grinshpoun et al. 2014, 2015) consists of n jobs that should be processed on m machines, where *some* of the operations of a job are allowed to be processed concurrently. An operation O_{jk} refers to the processing of job $j = 1, 2, \ldots, n$ on machine $k = 1, 2, \ldots, m$. The processing time of operation O_{jk} is denoted by p_{jk}. PCOSS is a generalisation of open shop scheduling (OSS). Its two extremes are the well-known *standard OSS* (Dorndorf et al. 2001), in which no concurrency is allowed, and *concurrent OSS* (Wagneur and Sriskandarajah 1993; Ng et al. 2003; Mastrolilli et al. 2010), in which *all* the operations of a given job are allowed to be processed concurrently. The set of operations that cannot be performed concurrently in a given PCOSS is presented by an undirected graph, called a *conflict graph*. In case preemptions are not allowed, a schedule for the PCOSS is equivalent to a problem of acyclically orienting the conflict graph (Grinshpoun et al. 2015).

As a generalisation of OSS, the PCOSS model can describe a large variety of real-life scenarios. The timetabling project that inspired the development of the PCOSS model involves the assignment of technicians to airplanes in an airplane garage (Grinshpoun et al. 2014, 2015). A set of tasks should be performed on a given fleet of planes, and every task should be done by a technician who has the expertise to do this task alone. The optimisation problem is to schedule the tasks in order to minimise a given objective function. This problem is mentioned in the literature with respect to both the standard (Bräsel and Kleinau 1996) and the concurrent (Wagneur and Sriskandarajah 1993) OSS versions. Nevertheless, the PCOSS model is more appropriate for describing this scenario, because in reality some tasks can be performed simultaneously on a plane while other tasks disturb each other, and therefore cannot be performed concurrently.

A common objective function is the *makespan* $C_{max} = \max\{C_j \mid 1 \leq j \leq n\}$, where C_j is the completion time of job j. It was proven that in the general PCOSS case, denoted $O|pconc|C_{max}$, the problem is NP-hard (Grinshpoun et al. 2015). In fact, even the problem with only one job and unitary processing times ($O|pconc, n = 1, p_{jk} = 1|C_{max}$), was shown to already be NP-hard, due to its equivalence to the problem of orienting an undirected graph (the PCOSS conflict graph) in order to minimise the size of the longest directed path.

In the present paper we take a step further in the study of PCOSS by investigating two PCOSS types: (1) *unit-time PCOSS*, i.e., PCOSS with unit processing times ($O|pconc, p_{jk} \in \{0, 1\}|C_{max}$); and (2) *integral-preemption PCOSS*, i.e., PCOSS with integral (integer-valued) processing times that allow preemptions at integral time points ($O|pconc, p_{jk} \in \mathbb{N}, \textit{int-pmtn}|C_{max}$). Unit-time PCOSS constitutes a special and simple subclass of integral-preemption PCOSS. For these types there is a strong connection between the PCOSS solution and the problem of graph colouring. This connection enables extracting insights and solutions from the well-studied field of graph colouring, and applying them to the recently developed PCOSS model. This line of research is inspired by a list of studies that link between various scheduling problems and graph colouring problems. For example, graph colouring was introduced several decades ago with regard to course timetabling (Welsh and Powell 1967), and this feature continues to the present day (Burke et al. 1994; de Werra 1997; Rickman 2014).

We begin by identifying a basic link between the makespan of a unit-time PCOSS and the chromatic number of the problem's conflict graph. This link lays the foundation to the connection between graph colouring and the more general model of integral-preemption PCOSS.

Next, we discuss the special case where the PCOSS conflict graph is *perfect*. We restate a known result of representing OSS as a problem of edge colouring (de Werra 1970; Gonzalez and Sahni 1976) in the language of conflict graph vertex colouring. When the conflict graph is perfect the colouring problem, and therefore the PCOSS problem, become polynomially solvable.

Finally, we relate to the case of a *uniform PCOSS*, i.e., a PCOSS in which all the jobs are identical. Such problems are important because they appear in many real-life scenarios, such as a heterogeneous fleet of airplanes that require standard maintenance.

The remainder of the paper is organised as follows. Section 2 gives some background on the OSS and PCOSS problems. Section 3 introduces the basic relations between PCOSS and graph colourings. Perfect conflict graphs are studied in Sect. 4, followed by the investigation of the uniform PCOSS case in Sect. 5. A discussion (Sect. 6) concludes the paper.

2 Background

This paper focuses on two PCOSS cases (unit-time and interal-preemptions) and their relation to graph colouring. As such, it is desirable to provide proper background on the PCOSS model. We begin with a short background on open shop scheduling, including an example with integral preemptions, and then continue to describe the recently proposed PCOSS model.

2.1 Open shop scheduling

An open shop scheduling problem consists of n jobs that should be processed on m machines. An operation O_{jk} refers to the processing of job j in machine k. The processing time of operation O_{jk} is denoted by p_{jk}. An open shop scheduling problem can be formally defined by a set $J = \{1, 2, \ldots, n\}$ of jobs, a set $K = \{1, 2, \ldots, m\}$ of machines, and a processing time matrix $PT = [p_{jk}]$.

The term "open" relates to the lack of a predefined machine order for each job, as opposed to other, more constrained, shop scheduling models such as *job shop scheduling* and *flow shop scheduling*. Having less a-priori constraints may, on the one hand, lead in some situations to trivialisation of the problem, for instance when relating to some specific objective functions. On the other hand, this openness implies that OSS problems have larger solution spaces than other, more constrained, shop scheduling models.

An example instance of an OSS with 4 jobs and 3 machines is given next. Note that zero processing time effectively means that the corresponding operation is not part of the problem.

$$PT = \begin{pmatrix} 2 & 2 & 2 \\ 3 & 0 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & 3 \end{pmatrix} \tag{1}$$

In standard OSS every job may visit a single machine at a time, and every machine may host a single job at a time. Therefore, the total processing times of any of the jobs, and the total processing times of any of the machines, serve as lower bounds for the makespan of any schedule, even when preemptions are allowed. A classic result in the theory of OSS is that this bound can be achieved in the cases of unit-time OSS and integral-preemption OSS (de Werra 1970; Gonzalez and Sahni 1976). Machine-oriented Gantt charts of possible optimal schedules for the above example, with and without preemptions, are given in Fig. 1. Without

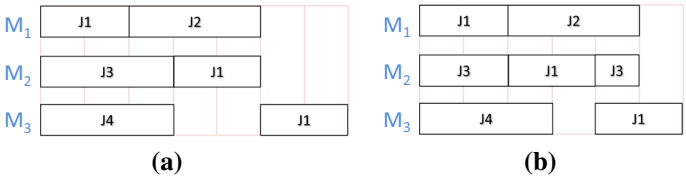

Fig. 1 A machine-oriented Gantt chart of a possible optimal schedule for OSS **a** without preemption, and **b** with preemption

preemption (Fig. 1a) the makespan is $C_{max} = 7$, and when preemption is allowed (Fig. 1b) C_{max} equals the lower bound of 6.

2.2 Partially-concurrent open shop scheduling

As opposed to standard OSS that allows no concurrency at all between operations of the same job, the fully-concurrent OSS model (Wagneur and Sriskandarajah 1993; Ng et al. 2003; Mastrolilli et al. 2010) allows all the operations of a given job to be processed concurrently. Grinshpoun et al. (2015) introduced a generalisation of these two extremes—the PCOSS model, which allows some operations that belong to the same job to be processed concurrently and at the same time enables the restriction of concurrent processing of other pairs of operations belonging to that same job. The exact set of operations that cannot be performed concurrently in a given PCOSS is determined by the conflict graph, which is defined next.

Definition 1 A *conflict graph* is a graph $G = (V, E)$, in which the vertex set $V = J \times K$ represents the operations of the problem and the edge set E represents conflicting operations, i.e., operations that cannot be performed concurrently are adjacent.

A PCOSS problem can then be formally defined by a set $J = \{1, 2, \ldots, n\}$ of jobs, a set $K = \{1, 2, \ldots, m\}$ of machines, a processing time matrix $PT = [p_{jk}]$, and a conflict graph G. In the case of standard (or fully-concurrent) OSS the conflict graph is degenerated, because these models inherently allow none (all) of the operations belonging to the same job to be processed concurrently. Contrary to that, the conflict graph in PCOSS holds important information that differentiates between various PCOSS instances.

For example, consider the processing time matrix of Eq. 1 (4 jobs and 3 machines). In Fig. 2 we show conflict graphs for the standard OSS, a possible PCOSS instance, and the fully-concurrent OSS. Note that operations with zero processing time are omitted.

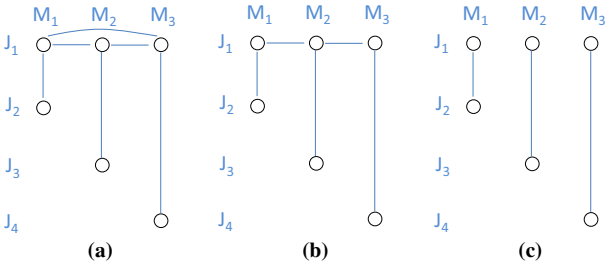

Fig. 2 Examples of conflict graphs for **a** a standard OSS, **b** a PCOSS instance, and **c** a fully-concurrent OSS

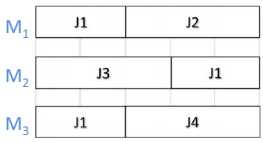

Fig. 3 A machine-oriented Gantt chart of a possible optimal schedule for the PCOSS instance of Fig. 2b

In the PCOSS model, the total processing time of all operations belonging to the same job no longer serves as a lower bound for the makespan. Here, for example, even when preemptions are not allowed, an optimal schedule has a makespan of $C_{max} = 5$, as shown by the Gantt chart in Fig. 3.

3 Basic relations between PCOSS and graph colouring

We start with establishing a well-known relationship between scheduling and graph colouring.

Suppose a set of operations should be executed in a set of non-overlapping time slots so that any conflicting operations will not be assigned to the same time slot. Such a problem can be modelled as a graph colouring problem. Given a graph $G = (V, E)$, where V is the vertex set and E the edge set, a *proper* colouring of G is an assignment of colours to the vertices so that adjacent vertices are assigned with different colours.

Let us consider the conflict graph of a unit-time PCOSS. A schedule in such a setting consists of homogeneous time slots. By matching the relevant time slots with a set of colours, the problem of assigning the operations to time slots becomes the problem of properly colouring the mentioned graph. The classical graph colouring problem is to find a proper colouring that makes use of as few colours as possible. The minimum number of colours needed to properly colour a given graph is called the *chromatic number* of the graph and denoted $\chi(G)$. Now, we proceed to establish the above relation between scheduling and graph colouring in the context of PCOSS. The next theorem constitutes the foundation of the present research. Its proof is a straightforward consequence of the construction that was described in the beginning of the section.

Theorem 1 *The minimum makespan of a unit-time PCOSS equals the chromatic number of its conflict graph G, i.e.,* $\min\{C_{max}\} = \chi(G)$.

Figure 4 depicts an example of a conflict graph for a unit-time PCOSS, for which p_{jk} equals either 0 or 1. The example processing time matrix is given by

$$PT = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \end{pmatrix} \tag{2}$$

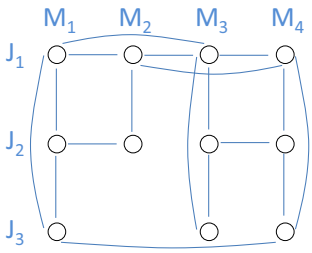

Fig. 4 An example of a unit-time PCOSS conflict graph

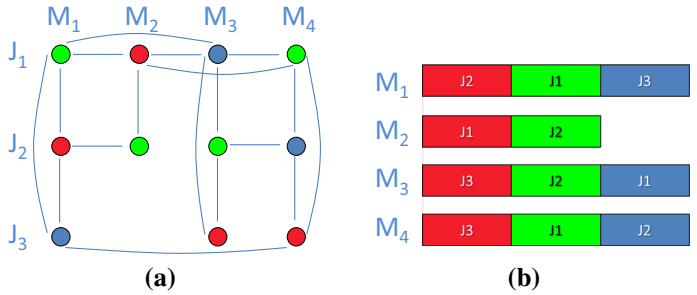

Fig. 5 A possible schedule for the unit-time PCOSS of Fig. 4 given by **a** colouring of the conflict graph with *three colours*, and **b** the corresponding machine-oriented Gantt chart (*red* → *green* → *blue*). (Color figure online)

Figure 5a presents a proper colouring of the example conflict graph using three colours. The corresponding schedule with $C_{max} = 3$ is illustrated in Fig. 5b.

The following definitions are required in order to deal with PCOSS with integral processing times.

Definition 2 A *weighted conflict graph* is a weighted version of the conflict graph, in which the weights of the vertices are the processing times of the corresponding operations.

Definition 3 Given a graph $G = (V, E)$ and integral positive weights on its vertices, w, a *w-proper* colouring is an assignment of $w(v)$ distinct colours to each vertex $v \in V$ so that adjacent vertices have no assigned colours in common.

Definition 4 The minimum number of colours needed for a w-proper colouring is called the *w-chromatic number* of G and will be denoted $\chi_w(G)$.

The problem of w-proper colouring a graph is known in the graph colouring literature as the set colouring problem or the multi-colouring problem. The w-chromatic number is also known as the weighted chromatic number (Caramia and Dell'Olmo 2001).

Now, consider the model of PCOSS with arbitrary integral processing times. The aforementioned relation with graph colouring still holds for this more general model, but only under the condition that integral preemptions are allowed. The next theorem states this precisely.

Theorem 2 *The minimum makespan of an integral-preemption PCOSS equals the w-chromatic number of its weighted conflict graph G, i.e.,* $\min\{C_{max}\} = \chi_w(G)$.

Proof Consider an integral-preemption PCOSS and its weighted conflict graph G. Let us decompose each operation O_{jk} into p_{jk} small operations, each with a single unit of processing time. The decomposed PCOSS is a modified type of unit-time PCOSS, in which each machine processes p_{jk} operations per job. The corresponding modified conflict graph $\tilde{G}$ is an unweighted version of G in which each weighted vertex $v_{jk} \in G$ that represents operation O_{jk} is substituted by a clique of p_{jk} unweighted vertices in $\tilde{G}$ (see Fig. 6). We follow the same construction of Theorem 1. Due to the equivalence between the weights and the processing times in G ($w \equiv p$), colouring of the modified unweighted graph $\tilde{G}$ is equivalent to w-colouring of the original weighted graph G. Consequently, the makespan of the decomposed PCOSS equals $\chi_w(G)$. Due to the possibility of integral preemptions, the schedule of the decomposed PCOSS is equivalent to that of the original PCOSS, which concludes the proof. □

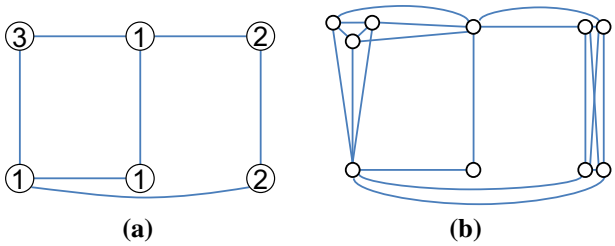

Fig. 6 **a** An example of a weighted conflict graph G, and **b** its corresponding modified (unweighted) conflict graph $\tilde{G}$

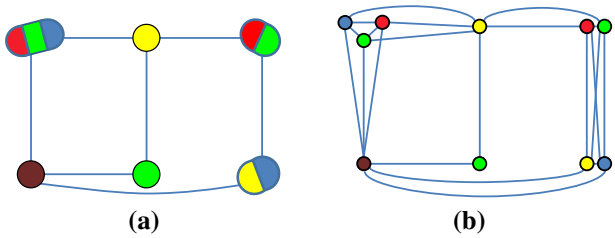

Fig. 7 **a** A possible w-proper colouring of G, and **b** the corresponding colouring of $\tilde{G}$. (Color figure online)

An example of a weighted conflict graph for a PCOSS problem with 2 jobs and 3 machines is given in Fig. 6a, along with the corresponding modified graph $\tilde{G}$ (Fig. 6b). Possible w-proper and proper colourings for these graphs are presented, respectively, in Fig. 7.

A question remains—how do Theorems 1 and 2 help *efficiently* solve the problems of unit-time PCOSS and integral-preemption PCOSS?

In general, it is known that the classical graph colouring problem is NP-hard. However, for graphs with some special structure, the colouring problem is known to be tractable. A family of such graphs is that of perfect graphs. In the following section we relate to the special case in which the conflict graph of a unit-time PCOSS and an integral-preemption PCOSS is perfect, which in turn makes the scheduling problem polynomially solvable.

4 Perfect conflict graphs

Given a graph G, the *clique number* $\omega(G)$ represents the size of the maximal clique in G. In general, because any clique of size k needs k distinct colours to be properly coloured, we have $\chi(G) \geq \omega(G)$. This section deals with graphs for which

$$\chi(G) = \omega(G). \tag{3}$$

The question of determining the conditions under which a given graph satisfies Eq. 3 is an old one that has kept Graph Theory researchers busy for many years. The most noticeable family of graphs that satisfies Eq. 3 is the family of perfect graphs. A graph is called perfect if Eq. 3 holds not only for the graph itself but also for any induced subgraph of it. Grötschel et al. (1984) proved that any perfect graph G can be coloured with $\omega(G)$ colours in polynomial time. This, together with Theorem 1, leads to:

Corollary 1 *If the conflict graph G of a unit-time PCOSS is perfect, then a schedule that minimises the makespan can be found in polynomial time, with $C_{max} = \omega(G)$.*

Moving to integral-preemption PCOSS, we recall the construction of the modified conflict graph in the proof of Theorem 2. There, each weighted vertex was substituted by a clique of size equal to the weight of the vertex. Lovász (1972) proved that substituting a clique for any vertex of a perfect graph[1] results in a graph that is also perfect. We therefore get:

Corollary 2 *If the weighted conflict graph G of an integral-preemption PCOSS is perfect, then a schedule that minimises the makespan can be found in polynomial time, with C_{max} given by the maximum weight of a clique in G.*

The most natural case of a PCOSS with a perfect conflict graph is that of standard OSS. Polynomial algorithms for unit-time OSS ($O|p_{jk} \in \{0,1\}|C_{max}$) and for integral-preemptions OSS ($O|p_{jk} \in \mathbb{N}, \textit{int-pmtn}|C_{max}$) were developed by de Werra (1970) and Gonzalez and Sahni (1976). These are based on a natural representation of an OSS as a matching problem in a bipartite graph, with the job set J and the machine set M as the two parts of its vertex set. Here we rephrase these results in the language of conflict graphs.

Theorem 3 *The minimum makespan of a unit-time OSS equals the clique number of its conflict graph G, i.e., $\min\{C_{max}\} = \omega(G)$, and can be found in polynomial time.*

Theorem 4 *The minimum makespan of an integral-preemption OSS equals the maximum weight of a clique in its weighted conflict graph G, and can be found in polynomial time.*

Proof In order to prove the above theorems it is enough to show that the conflict graph of an OSS instance is perfect. Let us consider a bipartite graph $BG = (J \cup K, O)$, where the edge set O is the set of operations with positive processing times, i.e., pairs (j,k) of job $j \in J$ and machine $k \in K$ for which $p_{jk} > 0$. We proceed to show that the conflict graph G is the line graph of BG:

1. Operations are represented in BG by edges and in G by vertices.
2. The edges in G are the pairs of operations that are in conflict, i.e., they either share a specific job or a specific machine. In BG these are pairs of edges that share a common vertex at one of their ends.

Line graphs of bipartite graphs are known to be perfect, i.e., G is perfect. Consequently, the proofs of Theorems 3 and 4 follow directly from Corollaries 1 and 2, respectively. □

As already noted, Theorems 3 and 4 restate known results from OSS research, but they do so through the language of conflict graphs that has not been considered in the past. For both unit-time and integral-preemption OSS the minimum makespan equals $\max\{\max\{\sum_{j=1}^{n} p_{jk} | k \in K\}, \max\{\sum_{k=1}^{m} p_{jk} | j \in J\}\}$. This result can be obtained directly from the bipartite graph representation of an OSS. For any job (machine) the total processing time of all the operations that are related to that job (machine) equals the degree of the corresponding vertex in the bipartite graph. The result for unit-time OSS follows directly from König's colouring theorem (König 1916; Lovász and Plummer 2009), which states that the edge set of any bipartite graph can be covered by Δ matchings, where Δ stands for the maximum degree of a vertex in the graph. A similar bipartite construction along with the Birkhoff-von Neumann theorem (Birkhoff 1946; Neumann 1953) gives the same result for the minimum makespan in integral-preemption OSS.

[1] Recall that substituting a clique C for a vertex $v \in G$, means deleting v and joining every vertex of C to those vertices of G that have been adjacent with v.

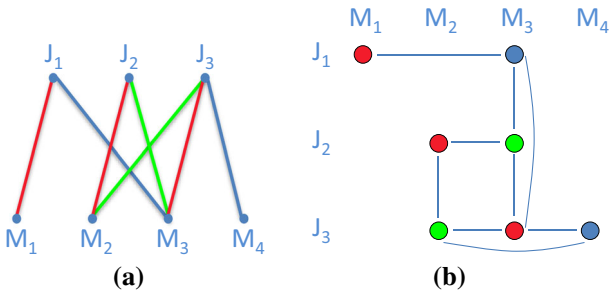

Fig. 8 An example of a unit-time OSS with 3 jobs and 4 machines: **a** the bipartite coloured graph, and **b** the corresponding coloured conflict graph. (Color figure online)

To exemplify the above results, Fig. 8a presents a bipartite graph of a unit-time OSS instance. A cover of the graph edges by $\Delta = 3$ matchings is shown. The corresponding conflict graph G, with $\omega(G) = 3$, is depicted in Fig. 8b.

In the case of the OSS model it is a matter of convenience whether to use the bipartite graph representation or the conflict graph representation. However, in the case of the more general PCOSS model, the bipartite graph representation is no longer valid, because the concurrency enables processing several operations of some job at the same time. Moreover, the degree of a vertex that represents job j is no longer a lower bound for the total processing time of job j.

We next show some general PCOSS instances (i.e., not standard OSS) with perfect conflict graphs. The discussed graphs have a special structure such that different jobs have the same conflicts. This means that the conflict graph G of a PCOSS with n jobs is obtained by a Cartesian product of two graphs[2]—the conflict graph of one job G_1, and the complete graph K_n. Perfectness of a Cartesian product of graphs is discussed by Ravindra and Parthasarathy (1977). The Cartesian product is denoted $G_1 \square K_n$. It is perfect in the following cases:

1. Every block in G_1 is complete. The number of jobs is arbitrary.
2. G_1 is bipartite and $n = 2$ (two jobs).
3. G_1 does not contain as induced subgraph an odd circle C_k or an odd circle with an extra arc $C_k + e$ and $n = 2$ (two jobs).

The first case is especially interesting because it can express real-life scenarios. For instance, consider a job to be an airplane that requires maintenance. The airplane is divided into several areas. Each area corresponds to a block in G_1, and the fact that the block is complete suggests that operations belonging to the same area of the airplane interfere each other. Contrary to that, operations belonging to different areas can be performed concurrently. Finally, some of the operations are located between two areas, and thus interfere both. Figure 9(a) illustrates such a scenario, and the corresponding PCOSS conflict graph that results from two such jobs is presented in Fig. 9b.

The perfectness of these graphs guarantees that the corresponding instances are polynomially solvable for both unit-time and integral-preemption PCOSS. In the next section we focus on conflict graphs of the form $G_1 \square K_n$, which are not necessarily perfect.

[2] A Cartesian product between graphs $G_1 = (V_1, O_1)$ and $G_2 = (V_2, O_2)$ is a graph $G = (V, O)$, with $V_1 \times V_2$ as its vertex set. Any two vertices $(v_1, v_2), (u_1, u_2) \in V$ are adjacent in G if either $v_1 = u_1$ and $v_2 \sim u_2$ or $v_2 = u_2$ and $v_1 \sim u_1$.

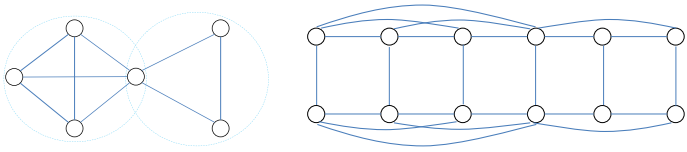

Fig. 9 **a** An example of G_1 that is composed of complete blocks (*dashed areas*), and **b** the corresponding two-job PCOSS conflict graph $G_1 \square K_2$, which is perfect

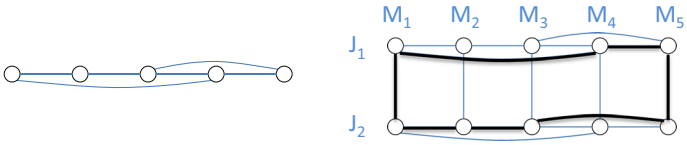

Fig. 10 **a** An example of G_1 that is a perfect graph, and **b** the corresponding two-job PCOSS conflict graph $G_1 \square K_2$, which is not perfect due to the existence of the enhanced 7-cycle

5 Uniform PCOSS

Definition 5 A PCOSS problem is called *uniform* if all the jobs are identical, i.e., they have the same conflict graphs and processing times.

The subclass of uniform problems is very interesting for practical reasons. Many real-life PCOSS problems have a similar structure for all their jobs. An example is the problem that inspired the development of the PCOSS model—assigning technicians (machines) to airplanes (jobs). In this example, a uniform conflict graph is obtained when all the airplanes must undergo exactly the same treatment and the concurrency constraints of the technicians are airplane-independent. Specifically, this is exactly the case when a homogeneous fleet of airplanes undergoes periodic maintenance.

In the previous section we already discussed the construction of a uniform PCOSS conflict graph by the Cartesian product $G_1 \square K_n$, without explicitly using the terminology of uniform PCOSS. The context there was to show that under some circumstances the resulting conflict graph can be perfect (see Fig. 9). Next, we demonstrate that this is not always the case, even when the conflict graph of each job G_1 is perfect. Figure 10a presents a conflict graph of one job G_1, which is perfect. Nonetheless, the two-job uniform conflict graph that results from the Cartesian product $G_1 \square K_2$ is not perfect (Fig. 10b). This can be witnessed by the existence of a 7-cycle (enhanced in the illustration).

Although not evident from the specific example in Fig. 10, in general the characteristics of the conflict graph of one job (G_1) may potentially influence the uniform case with several such identical jobs. Such a comprehension gives rise to interesting questions regarding uniform PCOSS problems. In particular, if one knows how to schedule (colour) one job, how simple is it to schedule a problem of several jobs? Such colouring issues are first discussed for uniform unit-time PCOSS, followed by the more general case of uniform integral-preemption PCOSS.

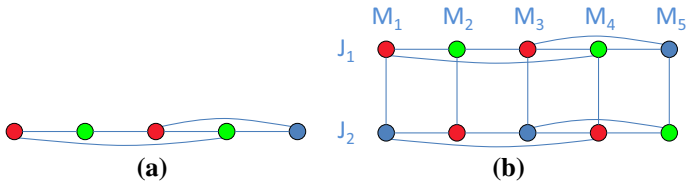

Fig. 11 **a** An example colouring of the graph G_1 from Fig. 10a with $\chi(G_1) = 3$ colours, and **b** the corresponding colouring of $G_1 \square K_2$ produced by cyclically permuting the colours. (Color figure online)

5.1 Uniform unit-time PCOSS

Theorem 5 *The minimum makespan of a uniform unit-time PCOSS equals the maximum between the chromatic number of G_1 and the number of machines, i.e.,* $\min\{C_{max}\} = \max\{\chi(G_1), n\}$.

Proof There is a known result that the chromatic number of a Cartesian product of any two given graphs G and H equals $\max\{\chi(G), \chi(H)\}$ (Sabidussi 1957; Klavar 1996). In particular, $\chi(G_1 \square K_n) = \max\{\chi(G_1), n\}$. This, together with Theorem 1 concludes the proof. □

It is worthwhile to give a short constructive algorithm for colouring $G_1 \square K_n$ using $\max\{\chi(G_1), n\}$ colours:

1. Choose some order of the $\max\{\chi(G_1), n\}$ colours.
2. Colour G_1 of the first job using the first $\chi(G_1)$ colours.
3. Colour the next jobs by cyclically permuting the colours.

Corollary 3 *If the job conflict graph G_1 is efficiently colourable, then the conflict graph of the uniform (unit-time) PCOSS is also efficiently colourable.*

The example of Fig. 10 is coloured in Fig. 11 following the above algorithm. This example demonstrates that although the graph $G_1 \square K_n$ is not perfect, it can still be coloured efficiently.

5.2 Uniform integral-preemption PCOSS

In Sect. 4 we proved that when the conflict graph is perfect, whether for unit-time (standard) OSS or unit-time PCOSS, the problem of finding minimum makespan is polynomially solvable (Theorem 3 and Corollary 1). These results were then naturally extended to the integral-preemption case (Theorem 4 and Corollary 2). It is therefore interesting to ask whether the results from the previous subsection regarding uniform unit-time PCOSS (Theorem 5 and Corollary 3) can be similarly extended to the analogue case of uniform integral-preemption PCOSS. More precisely, given a weighted conflict graph G of a uniform integral-preemption PCOSS, with G_1 being the conflict graph of a single job, we ask the following questions:

1. Is it true that $\chi_w(G) = \max\{\chi_w(G_1), p \cdot n\}$, where p is the maximum processing time (weight) of any operation?
2. Is it true that whenever G_1 is efficiently w-colourable then so is G?

Regarding the first question, an example with 2 jobs and 6 machines, depicted in Fig. 12, gives a negative answer!

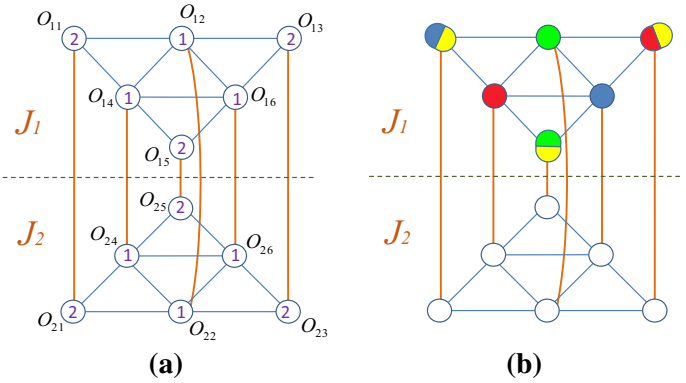

Fig. 12 **a** A weighed conflict graph of a PCOSS with 2 jobs and 6 machines. **b** A w-proper colouring of G_1 using $\chi_w(G_1) = 4$ colours. In order to w-colour G_2 a fifth colour is required. (Color figure online)

Consider the conflict graph in Fig. 12a. Both $\chi_w(G_1)$ and $p \cdot n$ equal 4. It is easy to see that the maximum processing time is $p = 2$, and therefore $p \cdot n = 4$. Next, we explain why $\chi_w(G_1) = 4$. First, G_1 contains a w-clique of size 4, for instance $\{O_{11}, O_{12}, O_{14}\}$, hence $\chi_w(G_1) \geq 4$. Next, the upper part of Fig. 12b illustrates a 4-colouring of G_1, thus $\chi_w(G_1) = 4$. Hence, $\max\{\chi_w(G_1), p \cdot n\} = 4$.

Conversely, the w-chromatic number of the weighted graph G is greater than 4 as we proceed to explain. First, we claim that the colouring of G_1 shown in Fig. 12b is unique up to permuting the colours. Indeed, the operations (vertices) $\{O_{12}, O_{14}, O_{16}\}$, form a 3-clique and therefore must be coloured by 3 different colours, say green ,red and blue, respectively. The operation O_{11}, which is adjacent to operations O_{12} and O_{14} has to be coloured by two colours other than green and red. Therefore, O_{11} must to be coloured by blue and a fourth colour, say yellow. By similar arguments, O_{13} must be coloured by red and yellow, and O_{15} must be coloured by green and yellow.

Since the colouring of G_1 is unique, in order to determine that $\chi_w(G) > 4$ it suffices to show that the colouring of G_1 depicted in Fig. 12b cannot be extended to a 4-colouring of the whole graph G. Because the colouring of G_1 is unique, then so is the colouring of G_2, which is the conflict graph of job 2 in the example. Now, suppose that an extension of the colouring of G_1 to a 4-colouring of G exists. Following our colouring of G_1, in any 4-colouring of G_2 the operations $\{O_{21}, O_{23}, O_{25}\}$ must be coloured by some common colour (as is the case with $\{O_{11}, O_{13}, O_{15}\}$, which are all coloured yellow). However, they cannot be coloured yellow since they are adjacent to $\{O_{11}, O_{13}, O_{15}\}$. Moreover, they also cannot be coloured with blue, red or green, since these colours appear in O_{11}, O_{13}, O_{15}, respectively. Consequently, in any extension of the colouring of G_1, the operations $\{O_{21}, O_{23}, O_{25}\}$ must share some fifth colour.

The latter example is one of a family of counter examples. In general, consider a weighted graph G_1 with $2n$ vertices $v_1, v_2, \ldots, v_n, u_1, u_2, \ldots, u_n$. The set $\{v_1, v_2, \ldots, v_n\}$ forms a clique of n vertices, each with one unit weight. The set $\{u_1, u_2, \ldots, u_n\}$ forms an independent set of n vertices, each with a weight of two units. In addition, for each j, $1 \leq j \leq n$, u_j is connected by an edge to all the v_i-s, except for v_j. In other words, G_1 is the complement of the graph shown in Fig. 13.

The reason for the difference between the unit-time case and the integral-preemption case is described next. In the unit-time case the whole conflict graph G is produced from the

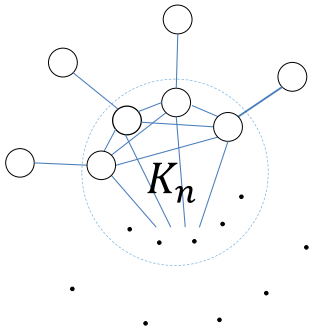

Fig. 13 A graph whose complement forms a conflict graph of G_1 with the property $\chi_w(G) > \max\{\chi_w(G_1), p \cdot n\}$

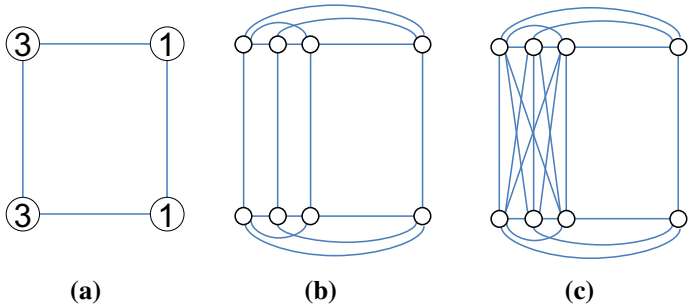

Fig. 14 **a** A weighted uniform conflict graph G. **b** The graph $\tilde{G_1}\square K_2$ is different from **c** the modified conflict graph $\tilde{G}$

conflict graph of a single job G_1 by a simple Cartesian product ($G_1\square K_n$). Contrary to that, the modified conflict graph described in the proof of Theorem 2 and illustrated in Fig. 6 is not generally obtained by a simple Cartesian product. This is demonstrated in Fig. 14. A weighted conflict graph of a uniform integral-preemption PCOSS with 2 jobs and 2 machines is depicted in Fig. 14a. The Cartesian product of the single job modified conflict graph with K_2 is shown in Fig. 14b, while the whole modified graph $\tilde{G}$ is presented in Fig. 14c. These graphs are clearly different.

The question of whether the scheduling (minimal makespan) of a uniform integral-preemption PCOSS is polynomially solvable, whenever the scheduling for one of its jobs is polynomially solvable, remains open.

6 Discussion

We have shown that the well-known relation between timetabling and graph colouring is also of importance with respect to the recently introduced model of partially-concurrent open shop scheduling. We focused on two types of problems—unit-time PCOSS and integral-preemption PCOSS. The difficulty of minimising the makespan for these problems correlates to the difficulty of properly colouring the corresponding conflict graph with minimum number of colours. Two main results have been concluded. The first is that whenever the conflict graph is perfect, the two mentioned problems are polynomially solvable. The second result states that minimising the makespan for a uniform unit-time PCOSS is polynomially solvable whenever it is polynomially solvable for any of its single jobs.

The polynomial algorithms for perfect graph colouring (Grötschel et al. 1984) rely on the ellipsoid method for Linear Programming and therefore their realistic efficiency is not clear. It is a question for further research to study PCOSS with conflict graphs that belong to special classes of perfect graphs. It is likely that for classes that arise in contexts of timetabling and scheduling problems, such as comparability graphs and interval graphs, practical polynomial algorithms do exist.

Our second result is a consequence of a known result about the chromatic number of the Cartesian product of two graphs (for n jobs $\chi(G) = \max\{\chi(G_1), n\}$). For the uniform integral-preemption case we have shown that an analogue relation between $\chi_w(G)$ and $\chi_w(G_1)$ is no longer valid. It is unclear whether for the uniform integral-preemption PCOSS there is a correspondence between the difficulty of minimising the makespan for any single job and the difficulty of minimising the makespan for the entire scheduling problem, as there is in the case of unit processing times. Another natural direction for further research is the question of general preemption PCOSS (not necessarily integral preemption), which relates to the relaxed fractional colouring problem.

References

Birkhoff, G. (1946). Three observations on linear algebra. *Univ Nac Tucumán Revista A*, *5*, 147–151.

Bräsel, H., & Kleinau, M. (1996). New steps in the amazing world of sequences and schedules. *Mathematical Methods of Operations Research*, *43*(2), 195–214.

Burke, E. K., Elliman, D. G., & Weare, R. F. (1994). A university timetabling system based on graph colouring and constraint manipulation. *Journal of Research on Computing in Education*, *27*(1), 1–18.

Caramia, M., & Dell'Olmo, P. (2001). Solving the minimum-weighted coloring problem. *Networks*, *38*(2), 88–101.

de Werra, D. (1997). Restricted coloring models for timetabling. *Discrete Mathematics*, *165*, 161–170.

de Werra, D. (1970). On some combinatorial problems arising in scheduling. *CORS Journal*, *8*(ROSE–ARTICLE–1970–001), 165–175.

Dorndorf, U., Pesch, E., & Phan-Huy, T. (2001). Solving the open shop scheduling problem. *Journal of Scheduling*, *4*(3), 157–174.

Gonzalez, T., & Sahni, S. (1976). Open shop scheduling to minimize finish time. *Journal of the ACM (JACM)*, *23*(4), 665–679.

Grinshpoun, T., Ilani, H., & Shufan, E. (2014). Partially-concurrent open shop scheduling. In *Proceedings of the 10th international conference of the practice and theory of automated timetabling (PATAT)* (pp 188–201).

Grinshpoun, T., Ilani, H., & Shufan, E. (2015). The representation of partially-concurrent open shop problems. *Annals of Operations Research*. doi:10.1007/s10479-015-1934-1.

Grötschel, M., Lovász, L., & Schrijver, A. (1984). Polynomial algorithms for perfect graphs. *North-Holland Mathematics Studies*, *88*, 325–356.

Klavar, S. (1996). Coloring graph productsa survey. *Discrete Mathematics*, *155*(1), 135–145.

König, D. (1916). Graphok és alkalmazásuk a determinánsok és a halmazok elméletére. *Mathematikai és Természettudományi Ertesito*, *34*, 104–119.

Lovász, L. (1972). Normal hypergraphs and the perfect graph conjecture. *Discrete Mathematics*, *2*(3), 253–267.

Lovász, L., & Plummer, M. D. (2009). *Matching theory* (Vol. 367). Providence: American Mathematical Society.

Mastrolilli, M., Queyranne, M., Schulz, A. S., Svensson, O., & Uhan, N. A. (2010). Minimizing the sum of weighted completion times in a concurrent open shop. *Operations Research Letters*, *38*(5), 390–395.

Ng, C., Cheng, T. C. E., & Yuan, J. (2003). Concurrent open shop scheduling to minimize the weighted number of tardy jobs. *Journal of Scheduling*, *6*(4), 405–412.

Ravindra, G., & Parthasarathy, K. (1977). Perfect product graphs. *Discrete Mathematics*, *20*, 177–186.

Rickman, J. P. (2014). *The design of a course-timetabling system using graph-coloring and artificial intelligence*. Honors Program Theses, paper 15, Rollins College.

Sabidussi, G. (1957). Graphs with given group and given graph-theoretical properties. *Canadian Journal of Mathematics*, *9*(515), C525.

von Neumann, J. (1953). A certain zero-sum two-person game equivalent to the optimal assignment problem. *Contributions to the Theory of Games*, *2*, 5–12.

Wagneur, E., & Sriskandarajah, C. (1993). Openshops with jobs overlap. *European Journal of Operational Research*, *71*(3), 366–378.

Welsh, D. J., & Powell, M. B. (1967). An upper bound for the chromatic number of a graph and its application to timetabling problems. *The Computer Journal*, *10*(1), 85–86.