Partially Concurrent Open Shop Scheduling and Graph Colourings

Hagai Ilani · Elad Shufan · Tal Grinshpoun

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 take a step further in the study of PCOSS by investigating the connection between PCOSS and graph colouring problems. This connection 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 focus on specific PCOSS instances, such as uniform PCOSS, and PCOSS with preemption, which correlate to a real-life timetabling project of assigning technicians to a fleet of airplanes.

Keywords Graph colouring · Open shop scheduling · Concurrent machines · Technician timetabling

1 Introduction

A *partially-concurrent open shop scheduling* (PCOSS) problem [6, 7] 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 (j, k) refers to the processing of job $j = 1, 2, \ldots, n$ on machine $k = 1, 2, \ldots, m$. The processing

Hagai Ilani
Department of Industrial Engineering and Management, SCE – Shamoon College of Engineering, Ashdod, Israel
E-mail: hagai@sce.ac.il

Elad Shufan
Physics Department, SCE – Shamoon College of Engineering, Beer-Sheva, Israel
E-mail: elads@sce.ac.il

Tal Grinshpoun
Department of Industrial Engineering and Management, Ariel University, Ariel, Israel
E-mail: talgr@ariel.ac.il

time of operation (j,k) is denoted by p_{jk}. PCOSS is a generalization of open shop scheduling (OSS). Its two extremes are the well-known *standard OSS*, in which no concurrency is allowed, and *concurrent OSS* [18,15,13], 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 are presented by an undirected graph, called a *conflict graph*. A schedule for the PCOSS is equivalent to a problem of acyclically orienting the conflict graph [7].

As a generalization of OSS, the PCOSS model can describe a large variety of real-life scenarios. The timetabling project that inspired the development of the PCOSS involves the assignment of technicians to airplanes in an airplane garage [6,7]. 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 optimization 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 [2] and the concurrent [18] OSS versions. Nevertheless, the PCOSS 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 [7]. 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 the connection between PCOSS conflict graphs and graph colouring problems. The merit of making such a connection is that it 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 [19], and this feature continues to the present day [3,21,16].

We begin by identifying a basic link between the makespan of a PCOSS with unit processing times and the chromatic number of the problem's conflict graph. This link lays the foundation to the connection between graph colouring and more general PCOSS instances that consist of integral (integer-valued) processing times. In the latter case our results concern problems that allow integral preemptions, i.e., preemptions at integral time points.

Next, we restate a known result of representing OSS as a problem of edge colouring [20,5] in the language of conflict graph vertex colouring . This is done in order to learn the PCOSS colouring and in particular the special case where the PCOSS conflict graph is perfect. For this case, the colouring problem and therefore the PCOSS 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 organized as follows. Section 2 introduces the basic relations between PCOSS and graph colourings. The connection to the edge colouring of the bipartite graph, starting from standard OSS and continuing with PCOSS, is studied in Section 3. The case of uniform PCOSS is investigated in Section 4. A discussion (Section 5) concludes the paper.

2 Basic relations between PCOSS and graph colourings

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. The latter 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 to different colours. Let us take a look at the graph whose vertex set is the given set of operations and whose edges are pairs of conflicting operations. By identifying the relevant time slots with a set of colours, the problem of assigning time slots to the operations 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, let us put this relation in the context of PCOSS.

A PCOSS is defined by a set of n jobs and m machines, a processing time matrix $PT = [p_{jk}]$, and a conflict graph, G, which describes whether pairs of operations may be processed concurrently or not. Denote $J = \{1, 2, \ldots, n\}$ and $K = \{1, 2, \ldots, m\}$. The conflict graph has $J \times K$ as its vertex set[1]. Vertices (j, k) and (i, l), which represent the corresponding operations, are adjacent if they may not be processed concurrently.

In Figure 1 we show 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

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

[1] Omitting vertices that represent operations with zero processing times.

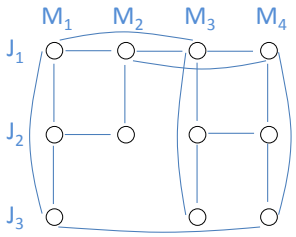

Fig. 1 An example of a PCOSS conflict graph.

The relation between PCOSS and graph colouring is stated in the next theorem, which is a cornerstone for the present research. The proof of the theorem is a straightforward consequence of the construction that was described in the beginning of the section.

Theorem 1 *For a PCOSS with unit processing times and a given conflict graph G, it holds that* $\min\{C_{max}\} = \chi(G)$.

In Figure 2(a) a proper colouring of the example conflict graph is given, using three colours. The corresponding schedule is shown in Figure 2(b) with $C_{max} = 3$.

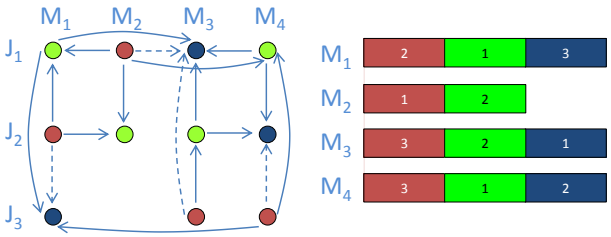

Fig. 2 (a) The conflict graph is coloured with three colours: red → green → blue. The acyclic orientation defines a schedule. (b) The corresponding Gantt chart.

For establishing a more general relationship between PCOSS and graph colouring, the following definition is needed.

Definition 1 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. The minimum number of colours needed for a w-proper colouring is called the w-chromatic number[2] of G and will be denoted $\chi_w(G)$. This colouring problem is known in the graph colouring literature as the set colouring problem or the multi-colouring problem.

[2] The "w-chromatic number" is also known as the weighted chromatic number [4].

Consider PCOSS with arbitrary integral processing times. For this general model the relationship with graph colouring is still relevant but integral preemptions must be allowed. The next theorem states this precisely.

Theorem 2 *The minimum makespan of a PCOSS, with allowed integral preemptions, equals the w-chromatic number of its conflict graph, where the vertices' weights are the processing times of the corresponding operations.*

Proof Consider a PCOSS with integral processing times. Let us decompose each operation (j,k) to p_{jk} small operations, each with a single unit of processing time. The decomposed PCOSS is a PCOSS with unit processing times, with an additional modification that each machine processes not only one operation per job, but p_{jk} operations. The modified conflict graph will be the original graph after substituting for each vertex, v, a clique of size $p_{jk}(v)$, where $p_{jk}(v)$ is the processing time of the operation presented by v (see Figure 3). The construction which led to Theorem 1 does not influence the theorem's assertion. Because colouring the modified graph is equivalent to w-colouring the original conflict graph with $w \equiv p$, we get that the makespan of the decomposed PCOSS equals $\chi_w(G)$. Finally, the theorem follows the observation that a schedule of the decomposed PCOSS is a schedule of the original PCOSS with preemptions at integral times. ■

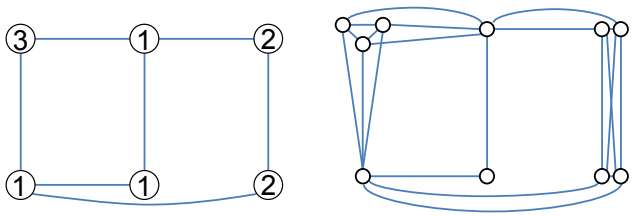

Fig. 3 (a) An example of a weighted graph and (b) its corresponding modified graph.

How do Theorems 1 and 2 help *efficiently* solve the problems of unit PCOSS and PCOSS with integral preemptions?

In general, it is known that the classical graph colouring problem is NP-hard. However, for some PCOSS instances the conflict graph has a special structure and consequently the colouring problem is tractable and so is the PCOSS. A special case for which the unit PCOSS and the PCOSS with integral preemptions are polynomially solvable is the standard OSS.

3 From standard OSS to PCOSS

Following the discussion in the previous section, it is natural to study the standard OSS in order to generalize some of the results to the more flexible

model of PCOSS. Polynomial algorithms for OSS with unit processing times and with arbitrary integral processing times and integral preemptions were developed in [20] and [5]. The algorithms are based on a natural representation of OSS as matching problems in a bipartite graph. More precisely, given an OSS instance, create a bipartite graph, $BG = (J \cup K, O)$, with the job set J, and the machine set K, as the two parts of its vertex set. The edge set O will be the set of operations with positive processing times; i.e., pairs (j, k) of job j and machine k for which the processing time of the operation (j, k) is strictly positive. For the case of unit processing times, the next theorem gives a min-max relation for the minimum makespan [20,5].

Theorem 3 *The minimum makespan for OSS with unit processing times equals* $\max\{\max\{\sum_{j=1}^{n} p_{jk} | k \in K\}, \max\{\sum_{k=1}^{m} p_{jk} | j \in J\}\}$.

The proof of the theorem is based on two features. First, the observation that 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. Second, according to König's colouring theorem [10,12] the edge set of any bipartite graph can be covered by Δ matchings, where Δ stands for the maximum degree of a vertex in a given graph.

A bipartite graph for an instance of $O|p_{jk} \in \{0,1\}|C_{max}$ is shown in Figure 4(a), together with a cover of the graph edges by $\Delta = 3$ matching. The corresponding OSS conflict graph, with $C_{max} = 3$, in shown in Figure 4(b).

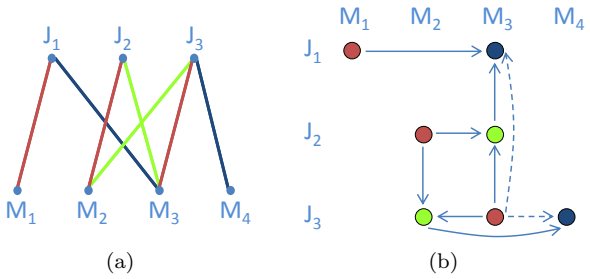

Fig. 4 An OSS example with 3 jobs and 4 machines: (a) the bipartite coloured graph, and (b) the corresponding coloured conflict graph.

The same bipartite construction along with the Birkhoff-von Neumann theorem [1,14] gives a similar relation for the minimum makespan in the model of OSS with integral processing times and integral preemptions.

Theorem 4 *The minimum makespan for OSS with arbitrary integral processing times and integral preemptions equals* $\max\{\max\{\sum_{j=1}^{n} p_{jk} | k \in K\}, \max\{\sum_{k=1}^{m} p_{jk} | j \in J\}\}$.

The model of matchings in a bipartite graph is no longer valid for the more general PCOSS problem. The concurrency enables processing several

operations that relate to one specific job at the same time. In the bipartite graph model for the unit processing times case, this means that the edges in the graph, which represent a set of operations that are processed concurrently, need not be a matching any more. Moreover, the degree of a vertex that represents job j is no longer a lower bound for the total processing time of job j. Therefore, in order to generalize Theorems 3 and 4 we restate them in the context of (conflict) graph colouring, as was presented in the previous section.

Let us take a look at the conflict graph of an OSS instance. The vertices are the operations $(j,k); j \in J, k \in K$, which are represented in the bipartite graph model $BG = (J \cup K, O)$ as edges. The edges of the conflict graph are the pairs of operations that are in conflict; i.e., they either share a specific job or a specific machine. In the bipartite graph these are pairs of edges that have a common vertex as one of their ends. Therefore, we have shown that the conflict graph of an OSS problem is the line-graph of its bipartite graph representation, BG. A matching in BG corresponds to a stable set of vertices in the conflict graph; i.e., a set of vertices that can be properly coloured with a single colour. Hence, a minimum cover of BG by matchings corresponds to a minimal proper colouring of the conflict graph. In addition, for any vertex $v \in J \cup K$, the set of edges that are incident to v corresponds to a clique in the conflict graph. Therefore, the maximum degree of a vertex in BG equals the size of a maximum clique in the conflict graph G, denoted by $\omega(G)$. By the assertion of Theorem 1, it follows that Theorems 3 and 4 take the following form in the conflict graph model:

Theorem 5 *The minimum makespan for OSS with unit processing times equals $\omega(G)$.*

Theorem 6 *The minimum makespan for OSS with arbitrary integral processing times and integral preemptions equals the maximum weight of a clique in its conflict graph with the processing times as its weights.*

In general, because any clique of size k needs k distinct colours to be properly coloured, we have $\chi(G) \geq \omega(G)$. The assertions of Theorems 5 and 6 are the same as saying that the conflict graph of an OSS satisfies the equation $\chi(G) = \omega(G)$. So, when trying to generalize Theorems 5 and 6 to instances of PCOSS, one should ask whether a conflict graph of a PCOSS instance satisfies the equation

$$\chi(G) = \omega(G) \tag{2}$$

or, alternatively, under what conditions does the equation hold?

Naturally, our interest is not only in the answer but also in its algorithmic aspects.

The question of determining the conditions under which a given graph satisfies Equation 2 is an old one that has kept Graph Theory researchers busy for many years. The simple example for which the equation does not hold is an odd cycle of length five or more. As we have just seen, the line graph of a bipartite graph does satisfy Equation 2. The most noticeable family of graphs

that satisfies Equation 2 is the family of perfect graphs. A graph H is called perfect if Equation 2 holds not only for H itself but for any induced subgraph of H. It has been proven by Grötschel et al. [8] that any perfect graph G can be polynomially coloured with $\omega(G)$ colours. Therefore, we can conclude with the following generalization of Theorem 5:

Theorem 7 *If the conflict graph of a PCOSS is perfect, then a schedule that minimises the makespan for an instance with unit processing times can be found in polynomial time.*

Moreover, according to a lemma of Lovász [11], substituting a clique for any vertex of a perfect graph[3] results in a graph that is also perfect. We therefore get the analogue generalization of Theorem 6:

Theorem 8 *If the conflict graph of a PCOSS is perfect, then a schedule that minimises the makespan for an instance with arbitrary integral processing times and integral preemptions can be found in polynomial time.*

In addition to perfect graphs, there could be other situations where the conflict graph satisfies Equation 2 and enables a polynomial solution to the PCOSS problem. An interesting question is how the structure of the conflict graph of any single job influences the structure of the whole conflict graph. Particularly, if the conflict graph of each job satisfies Equation 2 does it hold for the entire conflict graph? In the next section we regard these questions.

4 Uniform PCOSS

A *uniform* PCOSS has a conflict graph of a special structure – all the jobs have the same conflict graph. It is interesting to understand how the characteristics of the conflict graph of one job influence the uniform case with several jobs. In particular, if one knows how to schedule (colour) one job, how simple is it to schedule a problem of several jobs with identical conflicts? The uniform problem is also interesting from practical reasons. Many scheduling problems with concurrency constraints indeed have a similar structure for all the jobs. An example is the problem that inspired the development of the PCOSS model – assigning technicians (machines) to airplanes (jobs). A uniform conflict graph is obtained in case the airplanes must undergo exactly the same treatment and the technicians constraints are airplane-independent. In what follows we give a formal definition of the uniform scenario. Colouring issues are first discussed for the case of unit processing times and then for general processing times with preemption allowed.

Definition 2 A PCOSS problem is called *uniform* if the conflict graphs, as well as the processing times, are the same for all the n Jobs.

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

The conflict graph of a uniform PCOSS is a Cartesian product of the conflict graph of one job, denoted G_1, and a complete graph K_n. The Cartesian product is denoted $G_1 \square K_n$. An example for G_1 and the corresponding conflict graph of two jobs $G_1 \square K_2$ is given in Figure 5.

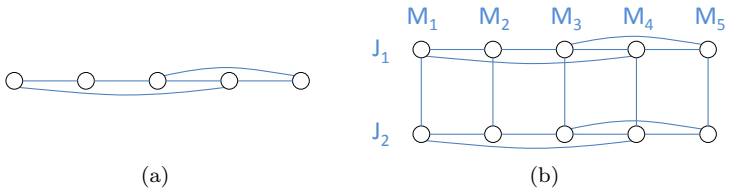

Fig. 5 (a) An example of G_1, a conflict graph for one job. (b) The two job uniform conflict graph $G_1 \square K_2$.

Theorem 9 *The minimum makespan of a uniform PCOSS with unit processing times equals the maximum between: (a) The chromatic number of* G_1*, and (b) The number of machines.*

Proof The proof follows directly from the known result [17,9] about the chromatic number of a Cartesian product of any given graphs G and H: $\chi(G \square H) = \max\{\chi(G), \chi(H)\}$. In particular, $\chi(G_1 \square K_n) = \max\{\chi(G_1), n\}$. ■

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 job 1 using the first $\chi(G_1)$ colours.
3. Colour the next jobs by cyclically permuting the colours.

Corollary 1 *If the job conflict graph* G_1 *is efficiently colourable, then the conflict graph of the uniform PCOSS is also efficiently colourable.*

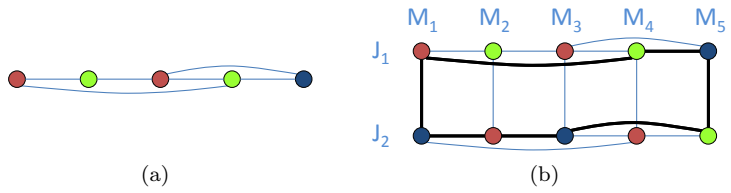

Fig. 6 (a) The one job conflict graph G_1 is coloured with $\chi(G_1) = 3$ colours. Note that in this example G_1 is perfect. (b) Colouring $G_1 \square K_2$ by cyclically permuting the colours. The enhanced 7-cycle indicates that $G_1 \square K_2$ is not perfect.

The example of Figure 5 is coloured in Figure 6 following the above algorithm. This example also demonstrates that even if G_1 is perfect, the graph $G_1 \square K_n$ need not be perfect. Still it can be coloured efficiently.

The above procedure is not naturally extended to the integral preemptions uniform PCOSS. This is because the modified conflict graph described in Theorem 2 is not generally given by a simple Cartesian product. More precisely, in general $\tilde{G} \neq \tilde{G_1} \square K_n$. Here, $\tilde{G_1}$ denotes the modified graph of one job and $\tilde{G}$ the modified conflict graph. Figure 7 demonstrates this inequality for a uniform weighted graph with two jobs and two machines.

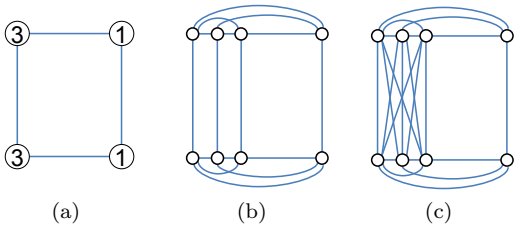

Fig. 7 (a) A weighted uniform conflict graph. (b) The graph $\tilde{G_1} \square K_2$ is different from (c) the modified conflict graph.

5 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 PCOSS. We have regarded PCOSS with unit processing times and PCOSS with arbitrary integral processing times and integral preemptions. 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 PCOSS with unit processing times is polynomially solvable whenever it is polynomially solvable for any of its single jobs.

Our first result relies on the ellipsoid method for Linear Programming and therefore its 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. It is unclear whether for the general uniform PCOSS with integral preemptions 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 for 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

1. Birkhoff, G.: Three observations on linear algebra. Univ. Nac. Tucumán. Revista A **5**, 147–151 (1946)
2. Bräsel, H., Kleinau, M.: New steps in the amazing world of sequences and schedules. Mathematical methods of operations research **43**(2), 195–214 (1996)
3. Burke, E.K., Elliman, D.G., Weare, R.F.: A university timetabling system based on graph colouring and constraint manipulation. Journal of Research on Computing in Education **27**(1), 1–18 (1994)
4. Caramia, M., Dell'Olmo, P.: Solving the minimum-weighted coloring problem. Networks **38**(2), 88–101 (2001)
5. Gonzalez, T., Sahni, S.: Open shop scheduling to minimize finish time. Journal of the ACM (JACM) **23**(4), 665–679 (1976)
6. Grinshpoun, T., Ilani, H., Shufan, E.: Partially-concurrent open shop scheduling. In: Proceedings of the 10th International Conference of the Practice and Theory of Automated Timetabling (PATAT), pp. 188–201 (2014)
7. Grinshpoun, T., Ilani, H., Shufan, E.: The representation of partially-concurrent open shop problems. Annals of Operations Research (2015). DOI 10.1007/s10479-015-1934-1
8. Grötschel, M., Lovász, L., Schrijver, A.: Polynomial algorithms for perfect graphs. North-Holland mathematics studies **88**, 325–356 (1984)
9. Klavar, S.: Coloring graph productsa survey. Discrete Mathematics **155**(1), 135–145 (1996)
10. König, D.: Graphok és alkalmazásuk a determinánsok és a halmazok elméletére. Mathematikai és Természettudományi Ertesito **34**, 104–119 (1916)
11. Lovász, L.: Normal hypergraphs and the perfect graph conjecture. Discrete Mathematics **2**(3), 253–267 (1972)
12. Lovász, L., Plummer, M.D.: Matching theory, vol. 367. American Mathematical Soc. (2009)
13. Mastrolilli, M., Queyranne, M., Schulz, A.S., Svensson, O., Uhan, N.A.: Minimizing the sum of weighted completion times in a concurrent open shop. Operations Research Letters **38**(5), 390–395 (2010)
14. von Neumann, J.: A certain zero-sum two-person game equivalent to the optimal assignment problem. Contributions to the Theory of Games **2**, 5–12 (1953)
15. Ng, C., Cheng, T.C.E., Yuan, J.: Concurrent open shop scheduling to minimize the weighted number of tardy jobs. Journal of Scheduling **6**(4), 405–412 (2003)
16. Rickman, J.P.: The design of a course-timetabling system using graph-coloring and artificial intelligence. Honors Program Theses, paper 15, Rollins College (2014)
17. Sabidussi, G.: Graphs with given group and given graph-theoretical properties. Canad. J. Math **9**(515), C525 (1957)
18. Wagneur, E., Sriskandarajah, C.: Openshops with jobs overlap. European Journal of Operational Research **71**(3), 366–378 (1993)
19. Welsh, D.J., Powell, M.B.: An upper bound for the chromatic number of a graph and its application to timetabling problems. The Computer Journal **10**(1), 85–86 (1967)
20. de Werra, D.: On some combinatorial problems arising in scheduling. CORS Journal **8**(ROSE-ARTICLE-1970-001), 165–175 (1970)
21. de Werra, D.: Restricted coloring models for timetabling. Discrete Mathematics **165**, 161–170 (1997)