Resource allocation problem in project management

Irina Burkova[1], *Boris* Titarenko[2,*] *Amir* Hasnaoui[3] and *Roman* Titarenko[3]

[1]Institute of Control Sciences of Russian Academy of Sciences, Russia
[2]Moscow State University of Civil Engineering, Yaroslavskoe shosse, 26, Moscow, 129337, Russia
[3]La Rochelle Business School, France

Abstract. Resource allocation problems in project management are notoriously complex. Therefore the development of efficient algorithms for solving various specific cases is a real problem. This paper shows a specific case of the problem, where a program has a particular structure. The resource allocation problem in such a program is reduced to classical Johnson's problem or job-shop scheduling problem. Effective solution methods, by way of reducing to maximum flow problems, are suggested for some types of resources. For other cases, heuristic rules are developed, with a description of the situations in which these rules allow good enough solutions to be obtained.

1. Introduction

Job shop scheduling (or job-shop problem) is an optimization problem in computer science and operations research. There are n activities $J_1, J_2, ..., J_n$ of varying processing times, which need to be scheduled on m machines trying to minimize the makespan, which is the total length of the schedule. The problem is presented usually as an online problem (dynamic scheduling). The decision of scheduling an activity can only be made online, when the activity is presented to the algorithm.

This problem is one of the best known combinatorial optimization problems, and was the first problem for which competitive analysis was presented by Graham [1].

There are a large number of papers dealing with the different approaches on this topic. Pioneering papers were published by Johnson [2]; Demeulemeester and Herroelen [3]; and Garey [4].

Depending on the scope of application all these papers can be divided into the following groups:

1. Papers dealing with optimal solutions (see, e.g., [5]).
2. Those where heuristic scheduling methods are developed (see, e.g., [6–9].
3. Describing algorithms for the two-stage scheduling problem (see, e.g., [10, 11).
4. Describing algorithms for the three-stage scheduling problem (see, e.g., [12-14].
5. Based on Johnson's algorithm (see, e.g., [2, 15])

.

* Corresponding author: boristitarenko@mail.ru

Some papers under this topic are devoted to the application of research operations methods in project management (see, e.g., [16–18]). A significant proportion of the models and methods of project management are the tasks of building schedules for project implementation, mainly related to the allocation of limited resources. Resource allocation problems on networks refer to complex multi-extremal problems.

There are a small number of models with exact solution methods. In general case, approximate and heuristic algorithms are applied. The complexity of the problems increases even more when we consider transfer time of resources between jobs – the simplest task of determining the sequence of work execution by one team becomes a complex travelling salesman task when transfer time period of a team from job to job is considered.

The paper proceeds as follows. Section 2 introduces a general framework for problem solving and all relevant definitions. In Sections 3–5, for three resource subclasses (Subclasses 1-3), it develops efficient algorithms based on the method of network planning, a crucial maximum flow problem. In Section 6, it shows that the problems in subclasses 4-7 have no efficient exact methods of solution and their solutions are heuristic algorithms, algorithms of local optimization or genetic algorithms. Resource allocation in heuristic algorithms is based on heuristic priority rules of activities. Section 7 concludes the paper.

2. Problem definition

Let us consider a program that consists of *n* projects; each of them is a coherent chain of three activities. The amount of work W_{ij} and dependence $f_{ij}\left(u_{ij}\right)$ of execution activity speed on the amount of resources u_{ij} it performs are set for each activity. Activities may be performed using resources of different types, of which there is a limited number. The problem lies in the allocation of resources in a program minimizing its duration.

General case problems are the so-called *NP*-hard problems with no effective methods of exact solution [16]. Assume that the dependence $f_{ij}\left(u_{ij}\right)$ is

$$f_{ij}\left(u_{ij}\right)=\begin{cases} u_{ij}, & u_{ij} \le a_{ij} \\ a_{ij}, & u_{ij} \ge a_{ij} \end{cases},$$

where a_{ij} – maximum number of resources that can be used to perform work (*i, j*).

Let N_j, j=1, 2, 3 denote the number of *j*-type resources (j = 1,2,3). Show that there is an optimal solution in which all types of activities are carried out by maximum number of resources (possibly with breaks) in a time $\tau_{ij}=\dfrac{W_{ij}}{a_{ij}}$.

If the condition $u_{ij} < N_j$ is met at some activity *(i, j)* during the time interval τ then it is $u_{kj} < N_j$ at some other activity *(k, j)* of the same type during the same time interval (or several types of activity). During τ time $W_{ij}=u_{ij}\tau$ of the work amount (*i, j*) and $W_{kj}=u_{kj}\tau$ of the work amount (*k, j*) are carried out.

Let us define another scheduling plan. First, the resources N_j are sent to activity *(i, j)*.

In addition the work amount W_{ij} is done in a time

$$\tau_I = \frac{U_{IJ}\tau}{N_I} < \tau .$$

Then the resources are sent to activity *(k, j)* where the amount of work W_{kj} is done in a time

$$\tau_j = \frac{W_{kj}}{N_i} = \frac{u_{kj}\tau}{N_j} < \tau ;$$

and the activity is completed when $\tau_1 + \tau_2 = \tau$.

So in the new plan the first activity is completed earlier and the second one is completed at the same time in comparison to the initial plan. Every time repeating this operation when $u_{ij} < N_j$, we come to a plan, when $u_{ij} = N_j$ for all types of activities if the activity is carried out.

In this case, we have a traditional "Flow shop 2 machines problem" known for its complexity.

Let us consider several subclasses:

1. Resources of the first type are limited, i.e.

$$\sum_{i=1}^{n} a_{i1} > N_1 . \quad (1)$$

Resources of the second and third types are sufficient and the corresponding activities are carried out in the minimum time τ_{i1}.

2. Resources of the second type are limited, i.e.

$$\sum_{i=1}^{n} a_{i2} > N_2 . \quad (2)$$

The activities of the first and third types are carried out in the minimum time τ_{i2}.

3. Resources of the third type are limited, i.e.

$$\sum_{i=1}^{n} a_{i3} > N_3 . \quad (3)$$

The activities of the first and second types are carried out in the minimum time.

4. Resources of the first and second types are limited, i.e. there is the condition (1) and (2). The activities of the third type are carried out in the minimum time τ_{i3}.

5. Resources of the first and third types are limited, i.e. there is the condition (1) and (3). The activities of the second type are carried out in the minimum time τ_{i2}.

6. Resources of the second and third types are limited, i.e. there is the condition (2) and (3). The activities of the first type are carried out in the minimum time τ_{i1}.

7. Resources of all types are limited, i.e. there is the condition (1), (2) and (3).

Consider the solution methods for these subclasses of problem.

Subclass 1

Once the resources of the second and third types are sufficient, the duration of activities of the second and third types are respectively equal to τ_{i2} and τ_{i3}, $i = \overline{1,n}$.

Denote

$$q_1 = \tau_{i2} + \tau_{i3}. \tag{4}$$

Sort the activities in the descending order q_1, i.e. $q_1 \geq q_2 \geq \ldots \geq q_n$.

To solve the problem let us define a bipartite network out of $2(n+1)$ nodes (Fig. 1).

Nodes of the first network layer (excluding the input node) correspond to projects but nodes of the second layer correspond to time interval. Let *T* be the time of completion of the program. Set the bandwidth of arcs (*0*, *i*) is equal to $c_{0i} = W_i$, and the bandwidth of the arcs (*j*, *z*) is equal to

$$\begin{aligned} c_{1z} &= (T - q_1) N_1 = \Delta_1 N_1; \\ c_{jz} &= (q_{j-1} - q_j) N_1 = \Delta_j N_1. \end{aligned} \tag{5}$$

Set the bandwidth of the arcs (*i*, *j*) is equal to $c_{ij} = a_{i1}\Delta_j$, $i, j = \overline{1,n}$, $i \geq j$.

Define a flow of the maximum value in the resulting network.

Theorem 1. *The minimum T, at which the flow of the maximum value saturates the input arcs, defines the optimal solution of the problem.*

Proof. Let *T* be set. Denote with x_{is} the amount of the first activity of the *i*-type project that is carried out in the *s*-interval: the first interval is the interval $(0, T - q_1)$, the second one is $(T - q_1, T - q_2)$, etc.

To ensure that all activities have been completed it is necessary and sufficient to fulfill the conditions $\sum_{s=1}^{i} x_{is} > W_i$, $i = \overline{1,n}$ under limitations:

$$\sum_{i=s}^{n} x_{is} \leq c_{sz}, \quad s = \overline{1,n};$$

$$0 \leq x_{is} \leq c_{is}, \quad i, s = \overline{1,n}.$$

It follows that $\{x_{is}\}$ is the flow and so the minimum *T* at which the flow saturates the input of the arcs corresponds to the minimum time of completion of the program. □

A lower bound for the minimum duration of the program can be obtained from the conditions

$$V_1 = \sum_i W_{i1} \leq \sum_s c_{sz} = (T - q_n) N_1.$$

It allows the following result to be reached:

$$T \geq \frac{V_1}{N_1} + q_n. \tag{6}$$

Example 1. Let us consider a program consisting of the four projects for which details are given in Tab. 1.

Table 1. Summary information on four projects.

i	1	2	3	4	V_i
W_{i1}	12	10	18	20	60
W_{i2}	18	9	20	12	59
W_{i3}	24	15	8	18	65
a_{i1}	4	2	3	5	
a_{i2}	3	3	4	6	
a_{i3}	6	3	4	6	
τ_{i1}	3	5	6	4	
τ_{i2}	6	3	5	4	
τ_{i3}	4	5	2	3	

From formula (4) we obtain $q_1 = 10, q_2 = 8, q_3 = 7, q_4 = 5$.

Let $N_1 = 6$. From the condition (6) we obtain the initial score of the completion time of the program $T_1 = \frac{60}{6} + 5 = 15$.

Fig. 2 shows the network (the numbers in parentheses are equal to the bandwidth).

The maximum flow is equal to $58 < V_1 = 60$. It follows that T_1 is necessary to increase. Notice that the increasing of *T* increases only the bandwidth of arcs c_{i1} $(i = \overline{1,n})$ and c_{1z}. The minimum increase of *T* is

$$\delta = \frac{60 - 58}{N_1} = \frac{1}{3}.$$

It allows to increase the flow on the arcs (0, 1), (1, 1) and (1, *z*) by 2 units. The resulting flow saturates the input arcs. Therefore we get the optimal solution with the program duration $T_{\min} = 15\frac{1}{3}$.

Subclass 2

So, as the resources of the second type in this subclass are limited, the durations of the first and the third types are equal to the minimum τ_{i1} and τ_{i3}. Notice that the durations of τ_{i1} define the early start dates of the second type activities that are equal to $t_i^{\rho} = \tau_{i1}$ and the durations of τ_{i3} for a given *T* define the late finish dates of the second type activities that are equal to $t_i^n = T - \tau_{i3}$.

Just as in the subclass 1, let us create a bipartite network where nodes of the first layer correspond to projects and nodes of the second layer correspond to time intervals matching up t_i^{ρ} and t_i^{n} arranged in ascending order. The method of creating nodes of the second layer is shown in the Example 2.

Example 2. Take the data from the Example 1 (Tab. 1) and let $N_2 = 6$. Firstly, get a lower bound for the program duration. To do this, notice that the early start time dated of the second type are equal to $\tau_{i1} = 3$. After completion of all activities of the second type it is necessary at least $\tau_{i3} = 2$ units of time to finish the program. Finally, the performance of all activities of the second type requires at least

$T_2 = \frac{V_2}{N_2} = \frac{59}{6} = 9{,}833 \approx 10$ (accurate to whole numbers).

On the other hand, the performance of the second operation of the *i*-type project with the maximum intensity of a_{i2} requires τ_{i2} time units that give the minimum duration of the *i*-type project $T_2 = \tau_{i1} + \tau_{i2} + \tau_{i3}$.

Thus, we obtain a lower bound of the program duration:

$$T \geq \max\left[\min_i \tau_{i1} + \min_i \tau_{i3} + \frac{V_2}{N_2};\quad \max_i \left(\tau_{i1} + \tau_{i2} + \tau_{i3} \right) \right] \tag{7}$$

For the example under consideration, we have:

$T \geq \max\left[3 + 2 + {}^{59}\!/_{6};\ \ \max(13;13;13;9) \right] = 14{,}833 \approx 15$ (accurate to whole numbers).

Take $T_1 = 15$. Tab. 2 shows the early start dates and the late finish dates of the second type activities if $T_1 = 15$.

Table 2. Early start dates and late finish dates of the second type activities.

i	1	2	3	4
t_i^{es}	3	5	6	4
t_i^{lf}	11	10	13	12

Arrange these points in the ascending order $t_1^s < t_4^s < t_2^s < t_3^s < t_1^f < t_4^f < t_3^f$ and define the duration of Δ_s intervals between them (Tab. 3.)

Table 3. Interval durations**.**

S	1	2	3	4	5	6	7
Δ_s	1	1	1	4	1	1	1

As we can see from the table, the first interval is $\left(t_1^s; t_4^s\right)$ of the duration 1, the second one is $\left(t_4^s; t_2^s\right)$ of the duration 1, the third one is $\left(t_2^s; t_3^s\right)$ of the duration 1, the fourth one is $\left(t_2^f; t_3^s\right)$ of the duration 4 and etc.

Now it's possible to create a bipartite network. The first layer of nodes, as before, corresponds to projects and the second layer corresponds to intervals. An arc (*j*, *s*) connects the node *j* to the node *s* if the activity can be performed in *s* interval .

The bandwidth of the arcs $\left(0, i\right)$ is equal to the amount of work W_{i2} of the relevant activities of the second type, the bandwidth of the arcs $\left(s, z\right)$ is $\Delta_s N_2$ and the bandwidth of the arcs $\left(i, s\right)$ is $c_{is} = a_{i2}\Delta_s$.It is easy to notice that the flow of the maximum value in the network of Fig. 3 does not saturate the input arcs. Indeed, in the first interval 3 units of the resources are not completely used. The remaining 60 – 3 = 57 units are insufficient to carry out 59 units of the work amount. Increasing *T* by one unit leads to an increase of the fourth interval duration that is $\Delta_4 = 5$ per one unit and thereafter to an increase of the bandwidth of the arcs $\left(i, 4\right)$ by a_{i2} and of the arc $\left(4, z\right)$ by 6. It is not difficult to show that in this case the maximum flow saturates the input arcs.

3. Subclass 3

In this subclass the resources of the third type are limited. Accordingly, the activity durations of the first and second types are equal to τ_{i1} and τ_{i2}. Denote $p_i = \tau_{i1} + \tau_{i2}$. Notice that $t_i^s = p_i$ defines the early start dates for the activities of the third type. Let the activities be arranged in p_i ascending order, i.e. $p_1 \le p_2 \le ... \le p_n$.

Just as in the previous cases, let us create a bipartite network where nodes of the first layer correspond to projects and nodes of the second layer correspond to time intervals:

$$\Delta_s = p_{s+1} - p_s, \ \mathrm{s} = 1, \mathrm{n}-1;$$

$$\Delta_n = T - p_n;$$

where *T* is the time of the program completion.

Respectively, define the bandwidth of the arcs, as is in the previous cases:

$$c_{0i} = W_{i3}, \ c_{sz} = N_3\Delta_s;$$

$$c_{is} = a_{i3}\Delta_s.$$

The minimum *T* at which the maximum flow saturates the input arcs defines the optimal solution of the problem. The proof is similar to Theorem 1. Get a lower bound for the program duration that is more accurate than in the previous cases. To this end define the minimal *r* interval number such that $\sum_1^r a_{i3} > N_3$.

Define the amount of work carried out during the first $\left(r-1\right)$ intervals. This amount is equal to:

$$V_3(r)=\sum_{i=1}^{r-1}\min\left[W_{i3};\ \ a_{i3}\sum_{i=s}^{r-1}\Delta_s\right]=a_{13}\sum_{1}^{p-1}\Delta_s+a_{23}\sum_{2}^{p-1}\Delta_s+\ldots+a_{p-1,1}\Delta_{p-1}.$$

A lower bound is equal to $T\geq\sum_{1}^{r-1}\Delta_s+\frac{V_3-V_3(r)}{N_3}+p_1$.

Example 3. Take the data from the Example 1 (Tab. 1). Set $N_3=8$, $r=2$, $V_3(2)=12$. The score is $T\geq\Delta_1+\frac{65-12}{8}+6=14\frac{5}{8}$.

Notice that the score obtained by analogy with the previous cases is equal to

$$c_1+\frac{V_3}{N_3}=6+\frac{65}{8}=14\frac{1}{8}<14\frac{5}{8}.$$

Bipartite graph for the case $T = 15$ is shown in Fig. 1.

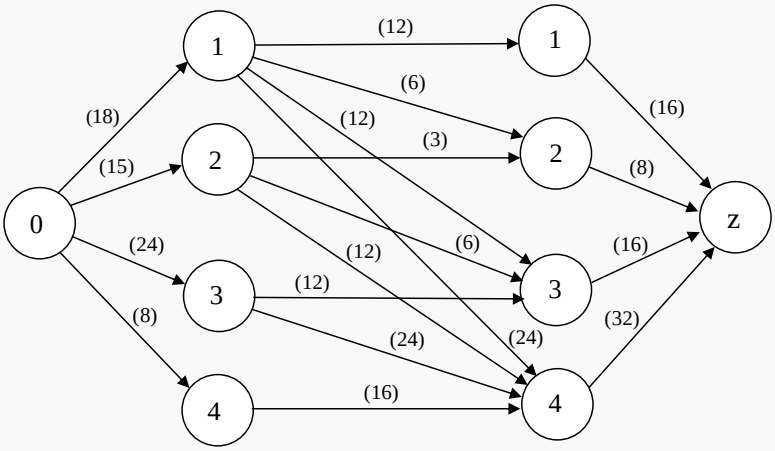

Fig. 1. Bipartite graph for Example 3.

In this case, the maximum flow saturates the input arcs and that's why the solution obtained is optimal (accurate to whole number *T*).

For this subclass let us consider also nonlinear concave dependence of $f_{i3}(u_{i3})$. Let assume that $f_{i3}(u_{i3})=u_{i3}^{\alpha};\ \ i=\overline{1,n}$ where $\alpha<1$. For the case $p_i=0,\ i=\overline{1,n}$ (i.e. there are no activities of the first and second types) it was proved by Barkalov et al [17] that the minimum program duration is

$$T_{\min}=\frac{W_{eq}}{N_3^{\alpha}};\tag{8}$$

where

$$W_{eq}=\left[\sum W_{i3}^{1/\alpha}\right]^{\alpha};\tag{9}$$

and is the equivalent to the program amount. Let us give a generalization of this result to the subclass 3.

Consider the following algorithm for the resource allocation.

The first step: $t_1 = p_1$. All the resources are sent to activity (1, 3).

The k-step: $t_k = p_k$. Denote $\delta(k-1)$ that is the equivalent to the uncompleted work amount of W_{i3}, $i = \overline{1, k-1}$.

Define the equivalent work amount $W_{eq}(k) = \left(\delta^{1/\alpha}(k-1) + W_{k3}^{1/\alpha}\right)^{\alpha}$ and the minimum duration

$$\Delta(k) = \frac{W_{eq}(k)}{N_3^{\alpha}}.$$

If $\Delta(k) > p_{k+1}$ define the uncompleted part of the equivalent work amount

$\delta(k) = W_{eq}(k) - N_3^{\alpha}(p_{k+1} - p_k)$.

The n-step: Define the minimum program duration $T_{\min} = \frac{W_{eq}(n)}{N_3}$.

Theorem 2. *The described algorithm gives the optimal solution of the problem.*

Proof. Let us prove the theorem by induction. The theorem, obviously, is true for $n = 1, 2$. Suppose that it is true for *n*-project number. Consider $(n+1)$ project. According to the assumption for *n*-projects, the theorem is true. Notice that the program duration is minimal, if $\delta(n)$ takes the minimum value. Let us prove that the described algorithm gives the minimum value of $\delta(n)$. It is quite obvious because by the time of p_n the maximum part of the equivalent work amount of $W_{eq}(n)$ has been done. □

Example 4. Let $n = 4$, $\alpha = 1/2$. Data values of p_i and W_{i3} are given in Tab 4. Let $N_3 = 4$.

Table 4. Data values of p_i and W_{i3}.

i	1	2	3	4
W_i	10	8	8	6
p_i	2	4	6	7

The first step: $t_1 = 2$. The activity has been started (1.3). By the time of $p_2 = 4$ the amount of $2\sqrt{N_3} = 4$ units has been completed. It remains to perform $\delta(1) = 10 - 4 = 6$.

The second step: $t_2 = 4$.

$$W_{eq}(2)=\sqrt{36+64}=10.$$

By the time of $p_3=6$ the equivalent amount of $2\sqrt{N_3}=4$ units has been completed. It remains to perform $\delta(2)=10-4=6$.

The third step: $t_3=6$.

$$W_{eq}(3)=\sqrt{36+64}=10.$$

By the time of p_4 the equivalent amount of $\sqrt{N_3}=2$ units has been completed. It remains to perform $\delta(3)=10-2=8$.

The fourth step: $t_4=7$.

$$W_{eq}(4)=\sqrt{64+36}=10.$$

The minimum program duration is $T_{\min}=\dfrac{W_{eq}(4)}{\sqrt{N_3}}+7=12$.

4. Subclasses 4, 5, 6 and 7

The problems of these subclasses have no efficient exact methods of solution and, typically, their solutions are heuristic algorithms, algorithms of local optimization or genetic algorithms. Resource allocation in heuristic algorithms is based on heuristic priority rules of activities. Basically, there are three priority rules [16].

Rule 1 (according to the level of critical activities). First and foremost, the activities with the maximum degree of criticality are carried out (with minimum late start dates).

Rule 2. First and foremost, the minimum duration activities of τ_i are carried out.

Rule 3. First and foremost, the activities with minimal late finish dates are carried out.

Late start and finish dates of activities are defined by the duration of all activities equal to the minimum of τ_{ij} and to the project duration $T=\max_i+\tau_{i1}+\tau_{i2}+\tau_{i3}$.

Late start date of the activity *(i, j)* is equal to $t_{ij}^s=T-\sum_{k=j}^{3}\tau_{ij}$ and its late finish date is

$$t_{ij}^f=T-\sum_{k=j+1}^{3}\tau_{ij}.$$

For the activities of the first type:

$$t_{i1}^s=T-\tau_{i1}-\tau_{i2}-\tau_{i3};$$
$$t_{i1}^f=T-\tau_{i2}-\tau_{i3}.$$

For the activities of the second type:

$$t_{i2}^s=T-\tau_{i2}-\tau_{i3};$$
$$t_{i2}^f=T-\tau_{i3}.$$

For the activities of the third type:

$$t_{i3}^s=T-\tau_{i3};$$

$$t_{i3}^{f} = T .$$

The choice of rules is defined by a specific problem.

So if the amount of the third type V_3 exceeds the amounts of activities V_1 and V_2 of the first and second types, it is important to start the performance of activities of the third type. In this case, apply the modified rule 2 (first and foremost, the projects with a minimum total duration of activities of the first and second types are carried out).

If the activities of the second type have the greatest amount of V_2 then the combined rule applies. First rule 2 applies, then after the activities of the second type have been completed apply rule 3. If the activities of the first type have the greatest amount of V_1 then the modified rule 3 applies (first and foremost, the activities of the first type with the maximum total duration of activities of the second and third types are carried out).

5. Conclusion

This paper shows the resource allocation problems in the program. For a number of resource types (Subclasses 1-3), effective methods for their solution have been developed, which are implemented by way of reducing these problems to maximum flow problems. Heuristic rules of solution are suggested for some difficult cases (Subclasses 4-7).

There were considered situations in which these rules allow good enough solutions to be obtained. It is interesting to consider further the resource allocation problems in the program using the criterion of maximizing profits and justification of the effectiveness of the heuristic rules for different classes of problems.

Other extensions of the reviewed problem are also of great interest. So, different projects of the program can be tied up with dependencies showed in a network diagram. They can be like "start-finish", "start-start", "finish-start" and "finish-finish" or nonbinding dependencies (i.e. may not be executed, but their violation leads either to the increase of the project duration or to the increase of its cost).

A program can have several goals. For instance, a region development program has economic, social, environmental and other goals; therefore, the problem becomes multi-criteria. There are so-called multi-purpose projects, which implementation contributes to the different objectives. The possible presence of the so-called interdependent projects should be also noted, joint implementation of which gives an additional (synergistic) effect.

Further development of the work is also associated with risk management. Responses to the risks associated generally with the activities for risks reduction or with exclusion of high-risk projects from the program. In practice, however, qualitative risk assessment (low, medium, high) are generally used. It should be noted that today the theory of risk management programs based on their qualitative assessments is rather poorly developed.

References

1. R. Graham, Bell System Technical Journal **45**, 9, 1563–1581 (1966)
2. S.M. Johnson, Naval Research Logistics **1**, 61–68 (1954)
3. E.L. Demeulemeester, W. Herroelen, Kluwer Academic Publisher, 1–710 (1996)
4. M.R. Garey, Mathematics of Operations Research Vol. 1, No. 2, 117–129 (1976)
5. F.F. Boctor, International Journal in Production Research **34**, 2331–2351 (1996)
6. C. Andrés, S. Hatami, International Journal of Production Research **44**, 4713–4735 (2011)
7. F. Croce, A. Grosso, F. Salassa, Annals of Operations Research **213**, 67–78 (2014)

8. Y. Sun, C.Y. Zhang, L. Gao, X.J. Wang, International Journal of Advanced Manufacturing Technology **55**, 723–739 (2011)
9. M. Abouei Ardakan, A. Hakimian, T. Rezvan, International Journal of Computer Integrated Manufacturing Vol. 27, No. 6, 519–528 (2014)
10. C. Lee, T. Cheng, B. Lin, European Journal of Operational Research **114**, 420–429 (1993)
11. M. Liu, Y. Xu, C. Chu, F. Zheng, Theoretical Computer Science **410**, 2099–2109 (2011)
12. V. Chankong, Y.Y. Haimes, Proceedings of the 5th International Conference on Industrial Engineering and Industrial Management, Cartagena (2011)
13. C. Koulamas, G.J. Kyparisis, Computers & Operations Research **28**, 689–704 (2001)
14. A. Maleki-Darounkolaei, M. Modiri, R. Tavakkoli-Moghaddam, I. Seyyedi, Management Science **39**, 616–625 (2012)
15. H. Allaoui, A. Artiba, International Journal of Production Economics **121**, 81–87 (2009)
16. P.S. Barkalov, I.V. Burkova, A.V. Glagolev, V.N. Kolpachev, Scientific edition of Institute of Control Sciences, 1–65 [in Russian] (2002)
17. P.S. Barkalov, V.N. Burkov, Scientific edition of Institute of Control Sciences, 1–56 [in Russian] (2001)
18. V.N. Burkov, I.V. Burkova, Automation and remote control Vol. 73, No. 7, 1242–1255 (2012)