Tomasz PRIMKE
Politechnika Śląska

TWO HEURISTIC ALGORITHMS FOR SCHEDULING TASKS IN PROJECTS

Summary. Due to projects deadlines, scheduling tasks in projects is a very important problem. The problem gets more complex, when the available resources can be assigned only to specific tasks. In this paper, two different heuristic algorithms, for solving such a problem, are presented. Both the algorithms are based on the critical path method.

DWA ALGORYTMY HEURYSTYCZNE DO SZEREGOWANIA ZADAŃ W PROJEKTACH

Streszczenie. Ze względu na terminy, szeregowanie zadań w projektach jest bardzo istotnym problemem. Zagadnienie to staje się jeszcze bardziej złożone, gdy dostępne zasoby mogą zostać wykorzystane tylko do wykonania określonych zadań. W artykule opisano dwa różne algorytmy heurystyczne, które można wykorzystać do rozwiązania tego problemu. Algorytmy te zostały oparte na metodzie ścieżki krytycznej.

1. Introduction

Nowadays, many small enterprises work in the IT industry. Their products are various kinds of software applications: desktop, mobile and browser-based. Many applications are developed for specific customers order.

In general, the process of software development is very complex. At the beginning, all the details are discussed with customer. Then, both business and technical analysis are performed. When customer accepts the final offer, the work on software architecure starts. At the next stage, software is implemented and tested, and it should be put into production phase afterwards.

At all those stages, specialists are required. They are analysts, software architects, managers, programmers and testers. Sometimes, even specialists in one field have different skills, which are required to perform different tasks. In the end, each team member should be assigned to proper tasks, according to those skills.

Of course, each client order is a project, and it should be managed like one. The management process requires specification of all the tasks, and then scheduling those tasks. Considering the mentioned facts, this problem is not trivial to solve.

In recent years, in the IT industry, the agile methods are very popular in projects management [4]. They proved to be more successfull, than the waterfall model. Their focus, though, is rather on general methodology, and on detailed procedures, than on scheduling.

In this paper, the problem of scheduling tasks in project is described. Then, two different algorithms are presented. Both the algorithms are based on the well-known critical path method. After some research, described in the section 4., the algorithms are modified in order to achieve better results.

2. The scheduling problem

Any project can be presented as a set of tasks. For each task, a processing time is specified. There are also precedence relations between tasks. A simple example is presented in the fig. 4.

More formally, it can be assumed, that a set of tasks T is given. For each task $i \in T$, a processing time p_i is specified. The precedence relations between tasks can be presented as a set of tuples $P \ni (i, j), i \in T, j \in T, i \neq j$. For each $(i, j) \in P$, the following constraint should be met:

$$c_i \leqslant s_j$$

where:

c_i the completion time of the task i,

s_j the start time of the task j.

Also a set of resources (employees) R is given. The tasks should be processed with the resources (employees). It is assumed, that for each resource $r \in R$, a set of tasks T_r is specified, and any task $i \in T_r$ from this set can be performed on the resource r.

It should be noted, that some tasks can be performed on different resources. In such case, it is assumed, that the task's processing time is the same, regarless of the resource used.

A schedule can be defined as a set of tuples:

$$(i, r, s_i, c_i) \in S$$

For all tuples in the set S, the following constraints should be met:

- once a task is started, on some resource, it will be completed without any interruptions, on the same resource,

$$\forall (i, r, s_i, c_i) \in S, c_i = s_i + p_i$$

- a task should be performed on a proper resource,

$$\forall (i, r, s_i, c_i) \in S, i \in T_r$$

- only one task can be performed, on any resource, at any moment of time,

$$\forall (i, r, s_i, c_i) \in S, \forall (j, r, s_j, c_j), i \neq j, c_i \leqslant s_j \vee c_j \leqslant s_i$$

- a task can be started, provided that all the preceding tasks have been completed,

$$\forall (i, r, s_i, c_i) \in S, \forall j : (j, i) \in P, c_j \leqslant s_i$$

- a task is scheduled only once,

$$\forall(i, r_1, s_i, c_i) \in S, \nexists(j, r_2, s_j, c_j) \in S, i = j \wedge s_i = s_j$$

- all the tasks have been scheduled,

$$\forall i \in T, \exists(i, r, s_i, c_i) \in S$$

- the earliest start moment, for any task, is zero

$$\forall i \in T, s_i \geqslant 0$$

Having a feasible schedule, the problem is to find a solution with a minimal makespan:

$$\min \leftarrow c_{max} = \max_i c_i, i \in T$$

The scheduling problem, described in this section, is similar to the well-known parallel machines scheduling one [3]. The difference is made by the additional constraints, which prevent scheduling tasks on any resource. For each task, the set of available resources is limited. Each resource can be regarded as an employee (or a team of employees), and thus can be characterized by some skills. Each task requires some skills, in order to be performed. The sets of tasks T_r can be derived from those skills. The reason for using the sets T_r instead of skills was to simplify the model. This simplification doesn't affect the model's applicability, yet there are other simplications, which do.

In the described model, for any task, all the resources, that the task can be performed on, are identical: the duration time of the task is independent on the used resource. In fact, all the resources (usually real people) are different, and perform differently while processing the same task. The differencies may be neglectible, as well as quite large.

It should be also noted, that in this paper, only one project is considered, and all the resources are used to complete it. In most real-life cases, many different projects are ongoing, and the resources are dynamically allocated to them, with respect to the changing needs and priorities.

For those reasons, the presented problem should be rather regarded as simplified, and used only for some introductory research purposes.

3. The algorithms

Two algorithms have been suggested to solve the problem defined in the section 2. Both the algorithms are based on the well-known CPM method.

Using the CPM method, it is possible to find all the critical tasks in a project. Since it is assumed, that all the tasks will be completed without any interruptions, the crucial moments for all the tasks are their starting times. The CPM method allows to calculate the earliest possible starting time of all the tasks. For all the critical tasks, any delay of this moment will cause the delay of whole project. For this reason, it is very important to focus on those tasks while scheduling.

All the calculations, though, are performed with the assumption, that is it possible to start any task, as soon as possible. When the number of available resources is limited,

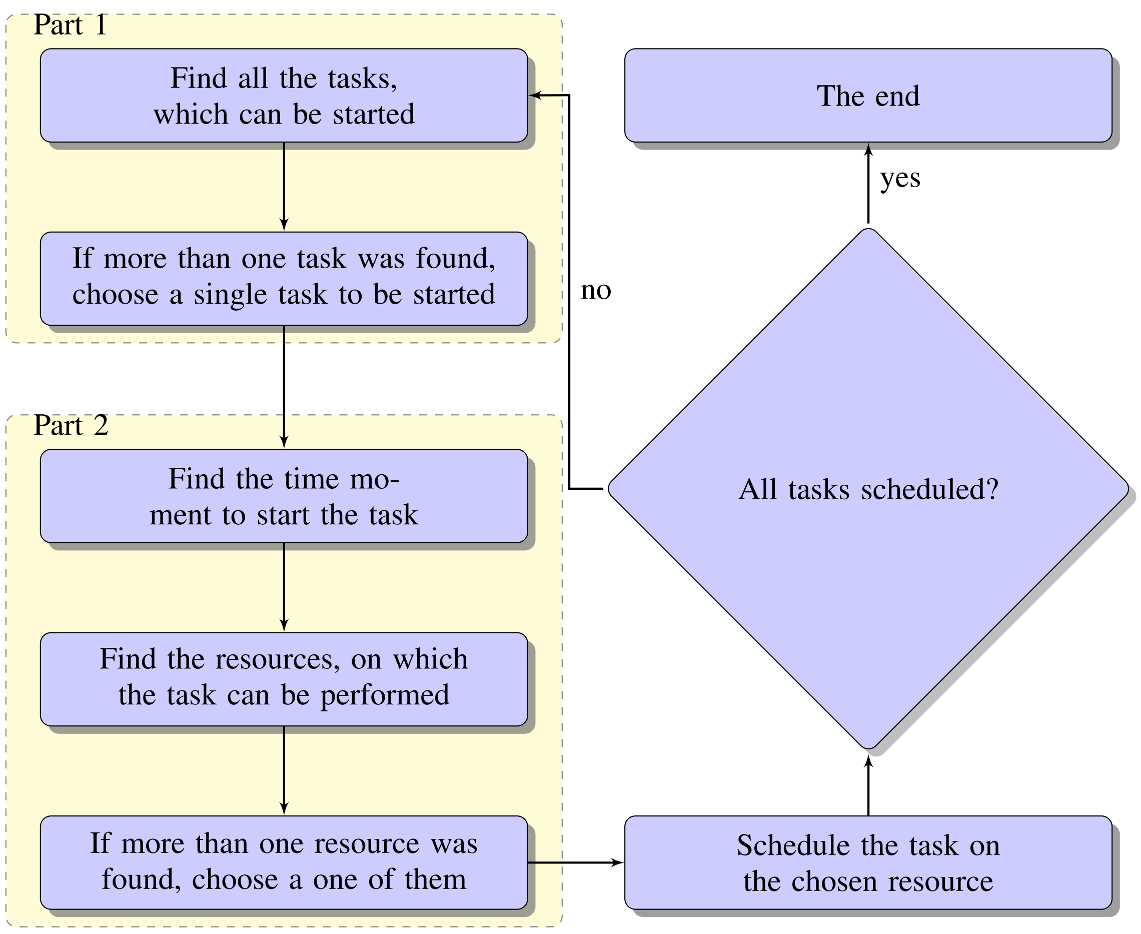
Part 1
Find all the tasks, which can be started
If more than one task was found, choose a single task to be started
Part 2
Find the time moment to start the task
Find the resources, on which the task can be performed
If more than one resource was found, choose a one of them
Schedule the task on the chosen resource
All tasks scheduled?
no
yes
The end

Fig. 1. Scheduling algorithm

this assumption may not be true. The problem, defined in the section 2, is even more complex, since for any task, only specific resources can be used. This constraint limits the possible scheduling solutions.

For the reasons mentioned above, the scheduling algorithm, presented in the fig. 1, was designed.

The presented algorithm can be divided into two, separate parts. In the first part (steps 1 – 2), a task is chosen. The second part (steps 3 – 5) is used to choose the start moment for the task, and to choose the resource. Although the CPM method cannot be used for the second part, it can be used for the first one. In the following subsections, two different algorithms are presented. Both the algorithms are based on the CPM method. The algorithm for choosing the start moment, and the resource to perform the chosen task on, is described in the subsection 3.3.

3.1. The alpha algorithm

In the first part of the algorithm, described in the section 3, the problem is only which task to choose, when more than one task can be started. This problem can be solved with the algorithm, presented in the fig. 2. The idea is to choose a critical, possibly the longest task.

For the project presented in the fig. 4, at the beginning only the first task can be scheduled. Once the task is completed, there are two different tasks available: the task

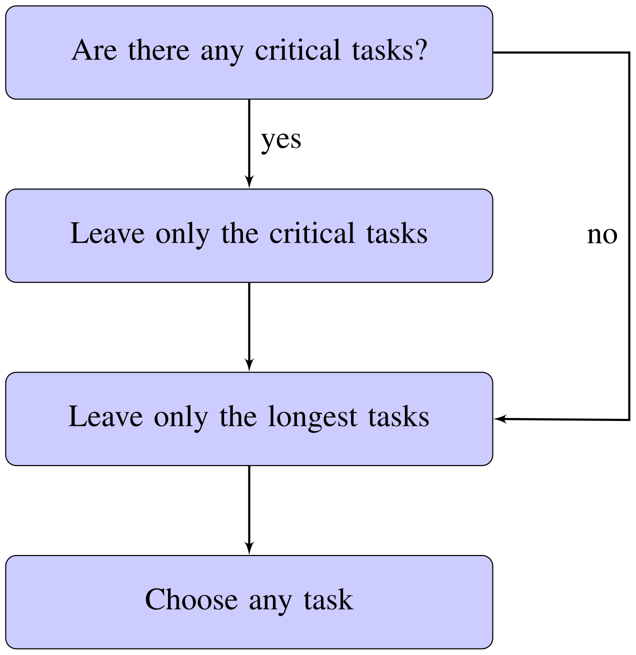
Are there any critical tasks?
yes
Leave only the critical tasks
no
Leave only the longest tasks
Choose any task

Fig. 2. Alpha algorithm

number 2 and the task number 9. As can be observed in the fig. 4, only the task number 2 is critical. Hence, this task should be chosen to be scheduled.

3.2. The beta algorithm

The problem of choosing tasks to schedule can be also solved by assigning priorities to all the tasks. Once each task is assigned a (unique) priority, it is possible to choose the task with the highest priority. This is the way the beta algorithm works.

In this algorithm, the CPM method is used to calculate the priorities for all the tasks [2], as presented in the fig. 3.

For the project presented in the fig. 4, the highest priority should be assigned to the first task. Then, after the first task is removed, there are two separate branches: one starting with the second task, and the second one starting with the ninth one. In the upper branch, the longest (and also critical) path is highlighted in the fig. 4. In the lower branch, the longest path consists of the tasks 9th, 11th, 12th, 13th and 14th. The length of this path is 102 units. On the other way, the total length of the tasks 14th, 8th, 6th and 4th is 104 units. So, the second priority should be assigned to the 2nd task, then the third priority to the 3rd task, the fourth priority to the 4th task, and the fifth priority to the 9th task.

3.3. Managing resources

Once the task is known, it should be scheduled on some resource. For this purpose, the following algorithm is used:

1. Find the earliest, possible time moment to start the chosen task.
2. If there are many resources, on which the task can be performed, choose the least loaded one.

In the first step, the start moment of the chosen task is prioritized. The start moment is determined by two different factors: (a) the completion times of all the preceding

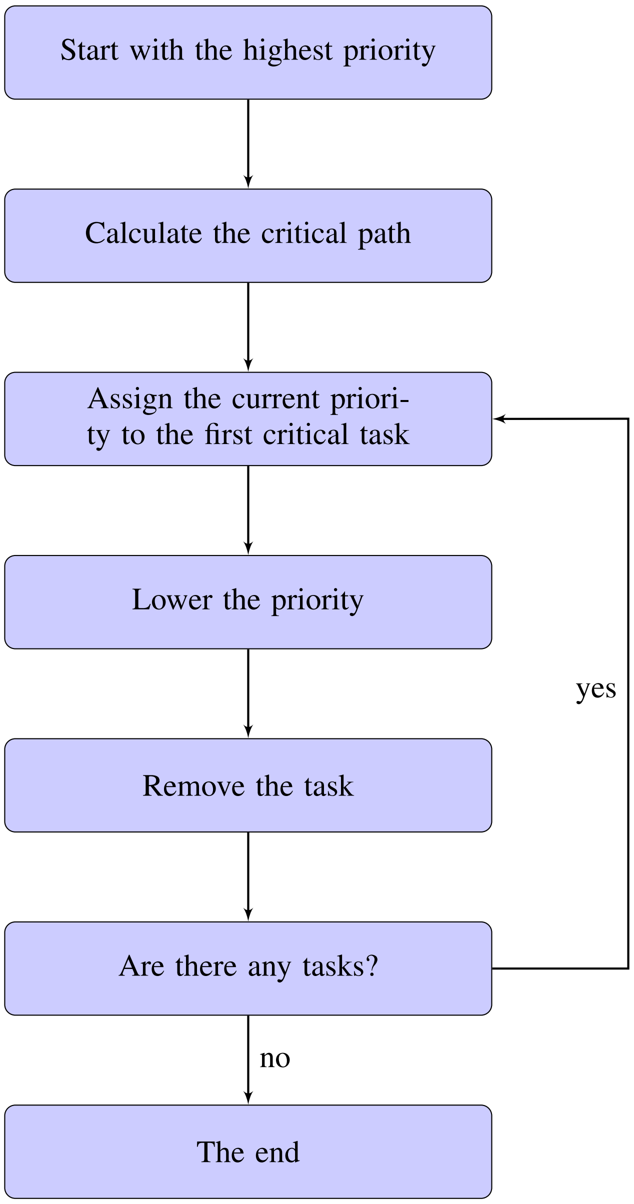
Start with the highest priority
Calculate the critical path
Assign the current priority to the first critical task
Lower the priority
Remove the task
Are there any tasks?
yes
no
The end

Fig. 3. Beta algorithm

tasks, and (b) the availability of resources, on which the task can be performed.

Once the start moment has been chosen, it is still possible, that more than one resource can be used to perform the task. In such situation, the loads of those resources should be calculated. Only the least loaded resources should be considered. If there are such resources, any one of them can be used.

4. Research

Using the two algorithms, described in the section 3, the examples presented in the figures 4 and 5 were solved. In both those cases, it was assumed, that only two resources are available. The table 1 presents possible assignments of those resources to tasks; two different sets were examined. In the first set, called dedicated, some tasks can be performed only on one of the resources (all such tasks have been stressed in the table).

In the second set, called universal, all the tasks can be performed on any resource.

Table 1

Resources for tasks

	Resource 1 (T_1)	Resource 2 (T_2)
Dedicated	**1**, **2**, **3**, 4, **5**, 6, **8**, 10, 13	4, 6, **7**, **9**, 10, **11**, **12**, 13, **14**
Universal	1 – 14	1 – 14

Table 2

Base results

Example	Resources set	Alpha	Beta
Separate branches	Dedicated	199	213
	Universal	163	159
Mixed branches	Dedicated	182	180
	Universal	160	160

The table 2 presents the values of makespan obtained for both the examples, analysed in this paper, for the dedicated and universal set of resources. It can be noted, that for the dedicated set of resources, the alpha algorithm performs better than the beta one. For the separate branches example, the difference is about 7%, while for the mixed branches example, the results are similar.

The most interesting results, though, were obtained for the universal set of resources. For this set, it should be possible to obtain the optimal solution, since there are no constraints on resources usage. The results for the separate branches example are slightly different, and the beta algorithm is better.

When those results were obtained, the separate branches algorithm was solved manually, assuming the universal resources set. The optimal solution was obtained (it is presented in the table 3), and the makespan was 147. It was obvious, that both the algorithms were not performing very well, since even for the universal set of resources, the obtained results were worse.

As it can be noted, in the solution presented in the table 3, all the critical tasks are scheduled on the first resource, while the second resource is used to perform all the other tasks. There are some crucial moments: the 5th task should be completed, before the 6th task is started, and the 7th task should be completed before the 8th task is started. It is also important to complete the 13th task, before the last, 14th task is started. Scheduling in such a way allows to obtain a solution without any unnecessary idle times.

This analysis leads to the conclusion, that the second part of the algorithm presented in the section 3 should be designed in a special way. For the case of two resources, one resource should be assigned to critical tasks (whenever it is possible), while the other one should be assigned to all the other tasks. Using this approach, the algorithms alpha and beta were redesigned, and then applied to solve the same problems. The obtained results are presented in the table 4 (and the details of the best solutions are presented in the tables 5 and 6).

Table 3

Optimal solution for the separate branches problem

Task	Resource 1		Task	Resource 2	
	Start (s_i)	End (c_i)		Start (s_i)	End (c_i)
1	0	10			
2	10	30	9	10	50
3	30	43	11	50	60
4	43	93	5	60	68
6	93	103	7	68	72
8	103	133	10	72	76
			12	76	98
			13	98	114
14	133	147			

Table 4

Improved results

Example	Resources set	Alpha	Beta	Alpha-critical	Beta-critical
Separate branches	Dedicated	199	213	155	155
	Universal	163	159	147	147
Mixed branches	Dedicated	182	180	208	180
	Universal	160	160	160	160

As it can be noted, for the problem of separate branches, the results are better. Both the improved algorithms gave the same results, and were able to find the optimal solution for the universal set of resources. Yet for the problem of mixed branches, the performance of the alpha-critical algorithm is clearly worse. For the beta-critical algorithm, the results are the same, as for the beta algorithm.

Considering all the four examined algorithms, the beta-critical one seems to be the best.

5. Summary

The literature on projects management is focused on many aspects of this field [1]. Although scheduling is one of them, it is usually reduced to the network methods, like CPM. No detailed scheduling algorithms are discussed. On the other hand, there are many scheduling methods [3], yet they are usually related to projects management.

The algorithms, described in this paper, are simple and easy to implement. They could be used as an extension to project management software, like Trac or Redmine.

The most difficult part in the scheduling problem, described in this paper, is availability of resources. Each resource can be assigned to specific tasks only. In the section 4, a change of the algorithm used to assign resources to tasks was presented. In most examined cases, this change caused the achieved results to be better.

It should be noted, though, that the results presented in this paper are rather in-

Table 5
The best solution for the mixed branches problem with dedicated resources
($c_{max} = 180$)

Task	Resource 1		Task	Resource 2	
	Start (s_i)	End (c_i)		Start (s_i)	End (c_i)
1	0	10			
4	10	60			
2	60	80			
3	80	93			
10	100	104	9	60	100
8	136	166	11	100	110
			12	110	132
			7	132	136
			6	136	146
			13	146	162
			14	166	180

Table 6
The best solution for the mixed branches problem with universal resources
($c_{max} = 160$)

Task	Resource 1		Task	Resource 2	
	Start (s_i)	End (c_i)		Start (s_i)	End (c_i)
1	0	10			
4	10	60	2	10	30
9	60	100	3	30	43
10	100	104	6	43	53
5	104	112	11	60	70
7	112	116	12	70	92
8	116	146	13	116	132
14	146	160			

troduction for future research. On the one hand, more complex examples should be considered, with much more tasks. On the other hand, in many real-life cases, the same resources (employees) are engaged in more than one project. The algorithms described in this paper can be used as a base for handling such more complex situations.

Praca finansowana ze środków przewidzianych na BK-213/RAu1/2016 temat 9.

BIBLIOGRAPHY

1. Kerzner H.: Project Management. A Systems Approach to Planning, Scheduling and Controlling. John Wiley & Sons, Inc., New Jersey, 2009.

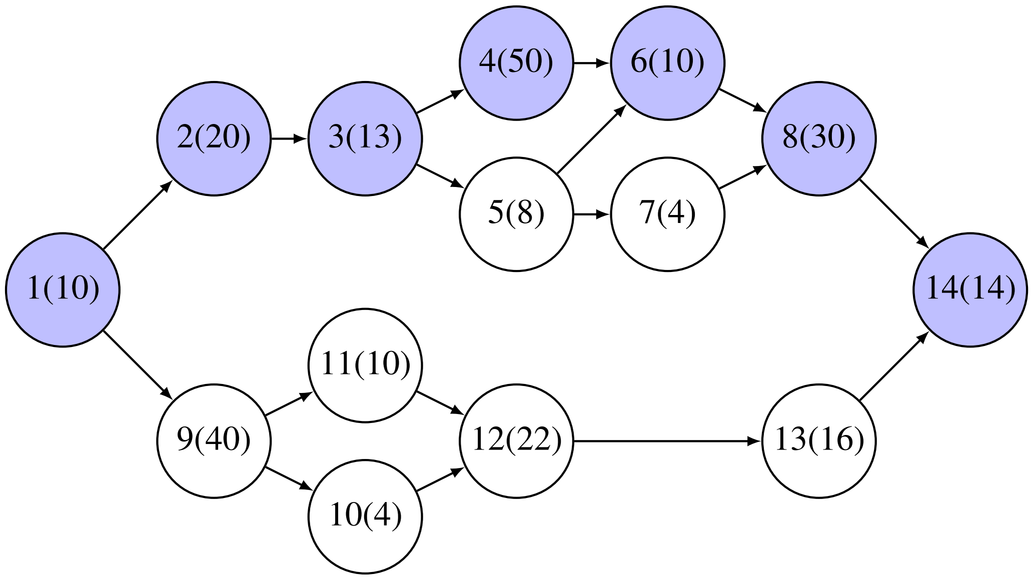

Fig. 4. A project with two separate branches

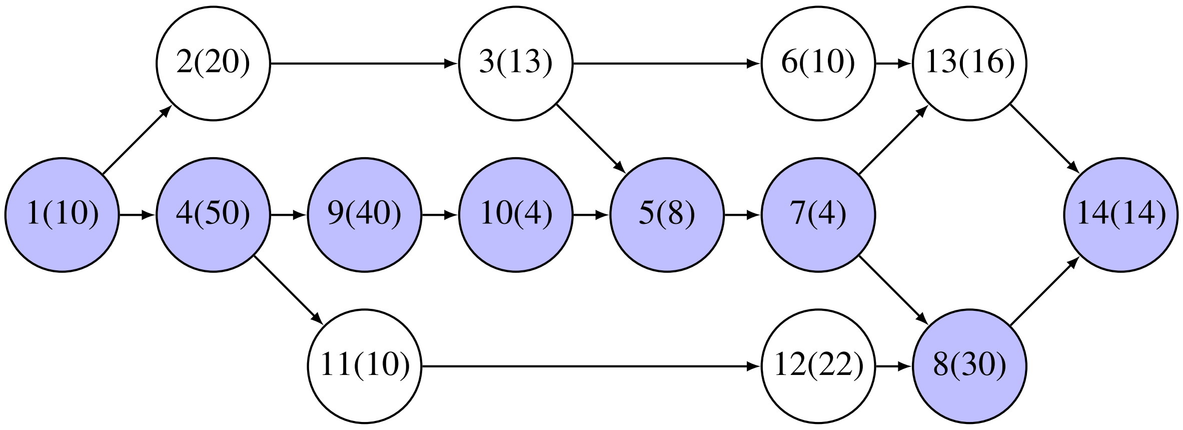

Fig. 5. A project with mixed branches

2. Lippman D.: Math in Society. David Lippman, 2013, http://www.opentextbookstore.com/mathinsociety/
3. Pinedo M.L.: Scheduling: Theory, Algorithms, and Systems. Springer, London 2016.
4. Rubin S.K.: Essential Scrum. A Practical Guide to the Most Popular Agile Processes. Addison-Wesley, 2012.