Annales UMCS Informatica AI XII, 1 (2012) 17–29
DOI: 10.2478/v10065-012-0006-2

Annales UMCS
Informatica
Lublin-Polonia
Sectio AI
http://www.annales.umcs.lublin.pl/

Resource Allocation Optimization in Critical Chain Method

Grzegorz Pawiński[1*], Krzysztof Sapiecha[1†]

[1] *Department of Computer Science, Kielce University of Technology al.1000-lecia P.P. 7, Kielce, Poland*

Abstract – The paper presents resource allocation optimization in Critical Chain Project Management (CCPM). The cheapest project schedule is searched with respect to time constraints. The algorithm originally developed for the hardware-software co-design of heterogeneous distributed systems is adapted to work with human resources and CCPM method. The results of the optimization showed significant efficiency of the algorithm in comparison with a greedy algorithm. On average, the optimization gives 14.10% of cost reduction using the same number of resources. The gain varies depending on the number of resources and the time constraints. Advantages and disadvantages of such an approach are also discussed.

1 Introduction

Project management is a continuously growing discipline. One of its vital problems is creating a project schedule. The most popular project scheduling techniques are the Ganntt chart, Critical Path Method (CPM) as well as Project Evaluation and Review Technique (PERT). Their short review can be found in [**1**]. CCPM is their competitor. It is based on the theory of constraint (TOC) applied by Dr. Goldratt to project management. TOC is a philosophy used to develop specific management techniques and focuses on constraints that prevent the project from reaching its goals. In [**2**], the relationships between the Goldratt's ideas and the CPM/PERT approach are described.

Resource allocation, called as well Resource-Constrained Project Scheduling Problem (RCPSP), attempts to reschedule the project tasks efficiently using limited renewable resources such that the maximal completion time of all activities is minimised. RCPSP

*g.pawinski@tu.kielce.pl
†k__sapiecha@tu.kielce.pl

is an NP-complete problem which is computationally very hard. The paper by Tormos and Lova [**3**] presents a brief description of two different approaches to solve this problem: heuristic and optimal. Heuristic approaches are necessary because they are the only methods of solving non-linear, complex problems in acceptable time.

The paper presents a resource constraint-based optimization algorithm for a resource allocation in the CCPM method. In the project management, the cheapest project schedule is searched with respect to time constraints. The algorithm from [**4**] was adapted to work with human resources and CCPM method. No similar attempts have been made before.

The next section contains a short overview of related work. The problem is stated in section 3, and the motivation for the research is given in section 4. Section 5 describes the algorithm of the optimization. Optimization results are given in section 6. The paper ends with conclusions.

2 Related work

PERT and CPM were mostly developed in the late 1950's. Scheduling procedures tend to focus on time rather than on the use of scare resources. Goldratt points out that these methods cause problems such as project late completion and over spending. The procedures concentrate on completion of individual tasks on time with the belief that the project will be on time, as well. The TOC project management attempts to take into consideration typical human behaviour during project planing and control [**5**]. CCPM eliminates contingency reserve embedded in individual tasks and aggregates all of them into a *project buffer*. Project duration can be reduced as a result of decreasing total contingency reserve [**6**]. The *critical chain* is a series of tasks which determines the earliest project finish. Unlike the *critical path*, it takes resource availability into account. Tasks executed by the same resource are scheduled in series. Paths with non-critical tasks are scheduled as late as possible (ALAP) and fed into the critical chain through *feeding buffers*. The feeding buffers contain contingency reserves aggregated from non-critical tasks and prevent the critical chain from delays [**6**].

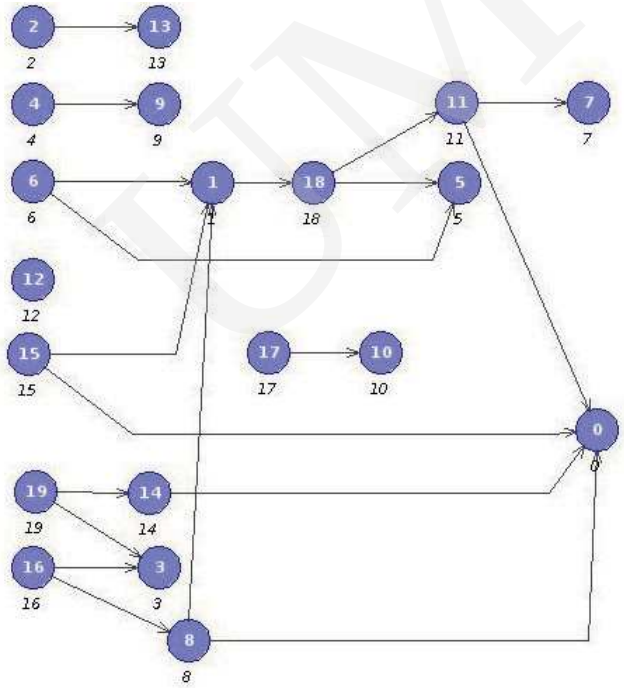
3 Motivation

Deterministic baseline schedules do not have any protection against uncertainty. Resource allocation with the CCPM method seems to overcome the problems that its predecessors simply neglected. Herroelen and Leus state that the vast majority of commercial software do not provide optimal algorithms for resource-constrained scheduling [**7**]. Instead, they use simple priority rules such as the latest start time (LST), the latest finish time (LFT) for generating the baseline schedule, which may be far off the optimum. Moreover, different sub-optimal procedures for solving the RCPS problem may give different feasible schedules with different critical sequences [**7**]. Goldratt argues that it does not matter which critical sequence is chosen and indicates

that the impact of the scheduling method used is seldom larger than the uncertainty of the project [**5**]. Nevertheless, creating a good baseline schedule does matter and it seems that implementation of resource scheduling optimization algorithm would be desired to enhance the process and to reduce the project total cost. Experimental results showed high efficiency of the algorithm, in the similar area, for co-synthesis of distributed embeded systems [**4**]. Such an application would be a support for project managers in project planing, project implementation, and project control.

4 Problem statement

Let us assume that a project consists of a set of tasks which are precedence related by the finish–start relationships with zero time lags. The task graph $G = (V, E)$ represents a model of the project plan, where the node v_i corresponds to one task. Each edge e_{ij} is a precedence relationship which means that the node v_i has to be finished before the node v_j can be started. An example of the task graph is shown in Figure 1.



Task	$t_{0,5}$	$t_{0,9}$
0	3d	8d
1	2d	5d
2	2d	5d
3	2d	5d
4	2d	5d
5	2d	5d
6	2d	6d
7	2d	5d
8	2d	5d
9	2d	6d
10	2d	6d
11	2d	5d
12	2d	6d
13	3d	8d
14	2d	5d
15	2d	5d
16	2d	5d
17	2d	6d
18	2d	6d
19	2d	6d

Fig. 1. An example of the task graph.

With each task the following parameters are associated:

- aggressive task completion estimate ($T_{0.5}(i)$) - 50 % of confidence that task i will be finished on time.

Table 1. Resource library.

Resource	Cost of usage	Cost rate
1	21	0.96
2	20	1.05
3	21	0.96
4	21	0.99
5	21	1.02
6	20	0.96
7	21	0.98

- safe task completion estimate ($T_{0.9}(i)$) - 90 % of confidence that task i will be finished on time.

A task has non-zero duration and requires a renewable human resource (type R) for its execution. Each human resource is universal but unique, and therefore it is a unit. Resource usage is related to some cost, described by the following parameters:

- unit cost $C_u(j)$, it is independent of the number of allocated tasks.
- cost of execution of a given task per day $C_e(i, j)$.

Due to the limited resource capacity, every resource has to perform different tasks sequentially. As a result, we can define human resources with different rates and different deployment costs. Illustrative values of resource parameters, gathered in the resource library, are presented in Table 1.

The cost of task completion is evaluated using the following rule:

$$C_c(i, j) = T_{0.9}(i) \cdot C_e(i, j) \tag{1}$$

where $C_c(i, j)$ is the cost of task i completion, executed by resource j.

Project duration (makespan) is determined by the completion time of all tasks. The main goal is to allocate resources in order to minimise the project total cost and finish it in a certain time. The project total cost may be specified using the following equation:

$$C = C_p \cdot T_p + \sum_{j=1}^{n} \left(C_u(j) + \sum_{i=1}^{m} C_c(i, j) \right) \tag{2}$$

where C_p is the project execution cost per day, T_p is the project duration, n is the number of resources, m is the number of tasks.

5 Optimization

The critical chain method consists of five steps, which are the following: creating project plan, defining resource library, allocating resources and optimization, identifying critical and non-critical chains, and project tracking. Project starts with building a task graph. At this stage, default task estimates are assigned.

In the second step a set of available resources is defined. Project activities and resources are linked together, to form a resource library, in such a way that all tasks

are assigned with every resource. The cost of task completion by a particular resource is automatically calculated, as described in the previous section. According to his own experience and the knowledge of the workforce, a project manager may change individual time estimates or even prohibit a resource allocation to the task. It should be emphsized that as far as human resources are concerned, some of them may be irreplaceable. Afterwards, the generated data are passed on input to the resource allocation and optimization steps.

The algorithm is a metaheuristic one, and originally was applied to hardware-software co-design of heterogenous distributed systems [**8**]. The adaptation mostly consists in taking into account specific features of human resources participating in a project schedule, contrary to specific features of hardware-software resources implementing functionalities of a computer system. It starts with the initial point and searches for the cheapest solution satisfying given time constraints. In each pass of the iterative process, current project schedule is being modified in order to get closer to the optimum. Searching direction is determined by the metric of a gain, presented in section 5.3. The algorithm constitutes the input of the project plan, resource library, project time constraint (deadline) and project execution cost per day (C_p). Next sections describe the main components, which are the initial solution, the metric of the gain and schedule refinement.

Identification of critical and non-critical chains is done in the fourth step. Before the identification, in the obtained project schedule, time estimates are reduced from the aggressive to safe ones. Contingency reserves are removed from critical tasks and aggregated in the project buffer. A size of the project buffer is smaller than the sum of individual reserves and it is evaluated as follows:

$$PB = \sqrt{B_1 + B_2 + \cdots + B_m} \tag{3}$$

where B_i is the contingency reserve from task i, m is the number of tasks in the critical chain. Feeding buffers are calculated for non-critical tasks in the same way. They are inserted at the end of non-critical chains to prevent from passing any delay to the critical chain. After buffers are inserted, the original critical chain remains unchanged even if one of the feeding chains is longer. Some tasks may be shifted right, when resource conflicts occur but without changing their precedence relationships. The investigation of buffer sizing techniques can be found in [**9**].

Finally, in the project tracking step, the project execution can be managed. The project manager may observe progress in work and make necessary changes to the existing project plan, such as adjusting tasks start time or duration. Furthermore, a baseline can be created for comparison of the realistic and planned schedules.

5.1 Initial solution

Good priority rule based methods are needed to determine the initial schedule. In this research, three heuristic procedures generating the fastest, the cheapest and random schedule, were investigated. An output from each of the procedures is a suboptimal

solution which the algorithm tries to enhance, according to the metric of the gain. Such a project schedule is suitable for the algorithm as the starting point, because it increases its flexibility. At the beginning, the procedures identify eligible activities, i.e. the tasks without predecessors, trying to find a resource for each task. A resource giving the smallest increase of the project duration or total cost, in the first and second procedures, respectively is allocated to the task. Afterwards, the task is replaced in a list of eligible activities by its all successors ready to be executed. The iterative process is repeated until the list is not empty. In the random schedule, searching for the best resource is unnecessary. Any resource can be allocated to any task, from the list, with the same probability.

5.2 Gain

The gain defines the quality of improvement of the schedule, and is basically the same as in [**4**]. The main goal is to reduce cost of the system and predict total impact of these changes on optimization. Taking into account only changes of the project cost ΔC, may lead the algorithm to be trapped into a local minimum. Total impact of modifications on every new solution is measured as an increase of slack time $\Delta\Omega$. Slack time is computed as follows:

$$\Omega = \sum_{i=1}^{m} (L_i - E_i) \tag{4}$$

where L_i is the latest task i start time, E_i is the earliest task i start time, m is the number of tasks [**9**]. Usually, the greater slack time, the more possibilities of resource allocation. If for any of the tasks L_i is less than E_i, the current solution does not satisfy the time constraints and is rejected.

Finally, the gain ΔE obtained from modification of the current solution, taking into account changes of the project total cost and slack time, is evaluated using the following formula [**4**]:

$$\Delta E = \left\{ \begin{array}{ll} \frac{-\Delta C}{\Delta \Omega} & , for \Delta\Omega < 0 \\ -\Delta C & , for \Delta\Omega = 0 \\ -\Delta C \cdot \Delta\Omega & , for \Delta\Omega > 0 \end{array} \right\} \tag{5}$$

5.3 Schedule refinement

At the very beginning of the algorithm, an initial project schedule is generated. Each pass of the iterative process is an attempt to improve the current solution in the two-stage procedure. In the first stage a new resource is inserted. All tasks from other resources are being considered and those giving the best gain are moved to the resource. Resources without allocated tasks are not taken into further consideration. An output of the first stage is the project schedule with the greatest gain.

Unlike the first stage, in the second one an attempt to decrease the number of resources is done in order to get rid of the more expensive ones. A resource without allocated tasks may be removed, only. Hence, to remove a resource the algorithm tries

to move all tasks allocated to the resource onto other resources. A new resource for the task, giving the best gain, is searched only among those with already assigned tasks. If a new project schedule has a positive gain, it becomes the best one. The new solution giving worse gain than the current one will never be chosen. The iterative process is repeated until no improvement can be found.

At the very end, project tasks may be shifted right using the ALAP algorithm to the latest feasible position into their forward free slack. It should be noticed that all the tasks are scheduled without violating their logical relationships or resource constraints. Moreover, the obtained project schedule has to satisfy given time constraints. Project schedules not meeting a deadline are rejected.

6 Optimization results

6.1 An example

A small example project was used to demonstrate particular stages and the merits of the optimization algorithm. Series of tests were performed in order to examine how time constraints influence project costs. A project plan containing 20 tasks with randomly generated data, was taken. Figure 1 shows an example of a task graph. Each task may have:

- at most 4 precedence relationships with the probability 0.35 of being inserted.
- aggressive and safe time estimates in the range of 2-3 and 4-8, respectively.

Resource library consists of 7 resources. Resource parameters were also randomly generated. The unit cost and cost of execution may vary up to 5% from the default values, which are 20 and 1.0 respectively. An example of the resource library is shown in Table 1. A benefit from having a different number of available resources was also tested.

The initial schedule was generated by greedy algorithms presented in the previous section. Tests were made for the project execution cost per day $C_p = 1$. Various project time constraints were examined. The bigger the constraint, the more flexibility in the resource allocation. In the first experiment the cheapest initial schedule was taken and optimized with the algorithm. The results of optimization are shown in Figure 2. Project duration and project cost were obtained with respect to time constraints. The figures in each row represent the results obtained using 4, 5 and 7 available resources, respectively, from the top. Figure 3 shows the results of the second experiment where the fastest initial schedule was taken. The experiments were repeated using 5, 6 and 7 available resources, for the project execution cost per day $C_p = 3$.

The percentage difference between the results obtained by the optimization algorithm and the greedy methods for $C_p = 3$ is presented in Tables 2-3. In the first two experiments, a large increase of the project duration is caused by relatively short tasks duration and time constraints set above 56 days. It is too large for this project but it allows showing a specific behaviour of the algorithm. The increase drops due to the

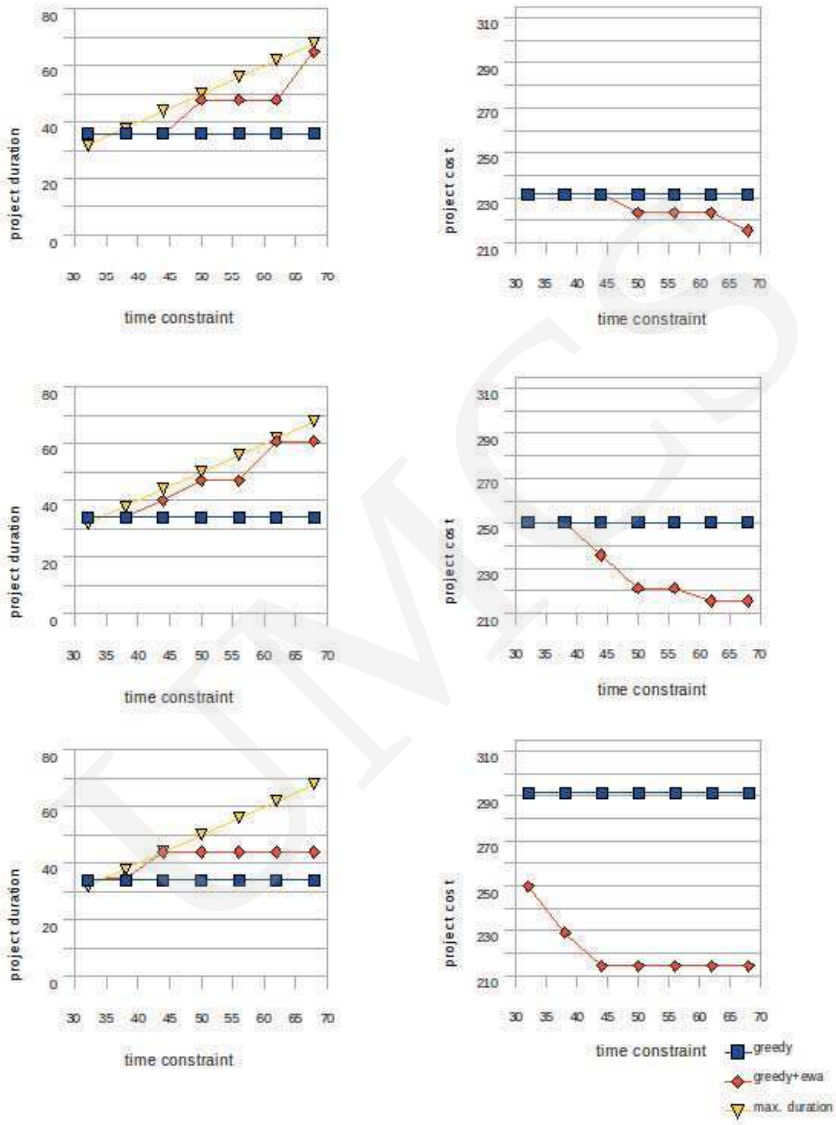


Fig. 2. Optimization results of the resource allocation for the fastest initial schedule using 4, 5 and 7 available resources, respectively, from the top.

increasing number of available resources. More resources allow finding shorter, and therefore cheaper project schedules.

In the last two experiments, every day of the project execution significantly increases the project total cost, which the algorithm minimizes. As a result, project schedules are shorter. Moreover, the obtained results have lower time increase and are much cheaper. In some cases, the project cost was reduced by 17.48% with no time increase.

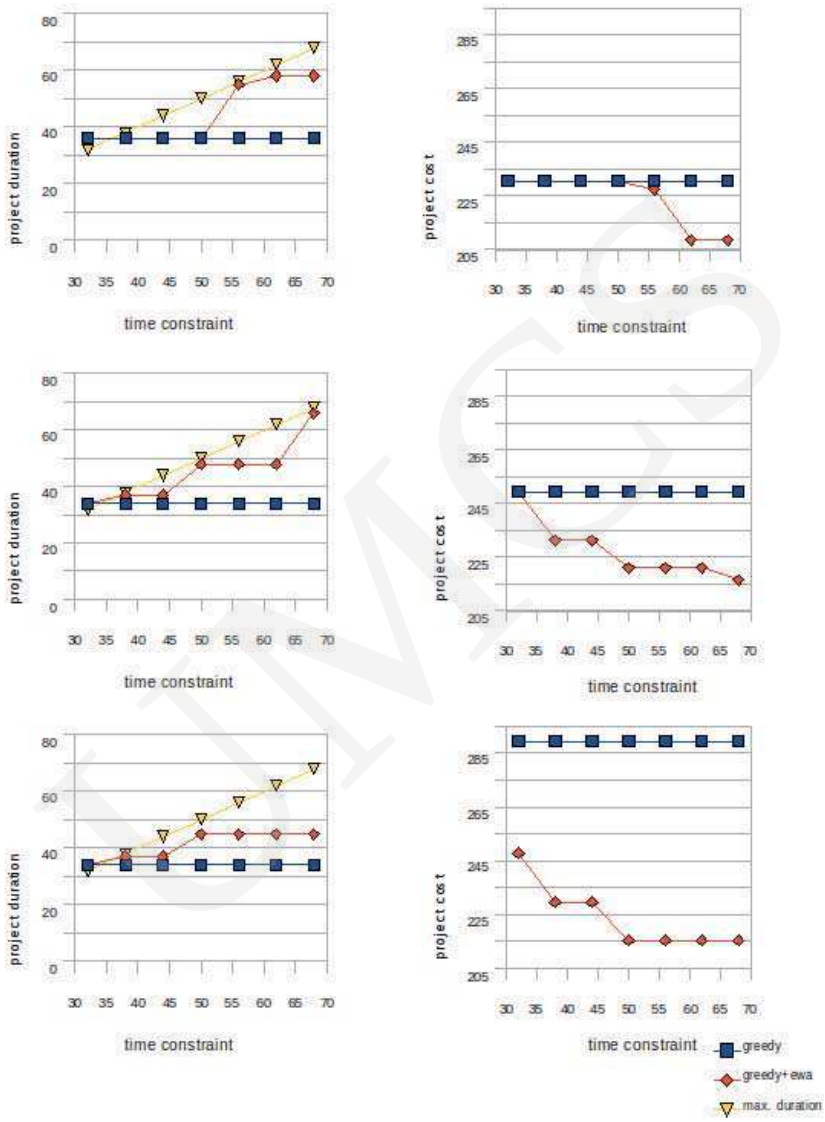


Fig. 3. Optimization results of the resource allocation for the cheapest initial schedule using 4, 5 and 7 available resources, respectively, from the top.

6.2 Statistics

Finally, the efficiency of the algorithm was estimated using 100 randomly generated project plans containing 30 tasks. The results have been obtained for the project execution cost per day $C_p = 3$ using the same resource library. Resource parameters were randomly generated, as described before. It should be noticed that every project is unique and it should be assigned with individual system parameters. For testing

Table 2. Difference between the optimization algorithm and the fastest initial schedule for $C_p = 3$ (%) .

No. Resources	Time constraint	32	38	44	50	56
5	cost	0	0	-0.79	-1.05	-1.05
	time	0	0	17.65	38.24	38.24
6	cost	-12.06	-12.06	-12.06	-12.06	-12.06
	time	0	0	0	0	0
7	cost	-12.35	-17.48	-17.48	-17.48	-17.48
	time	-2.83	0	0	0	0

Table 3. Difference between the optimization algorithm and the cheapest initial schedule for $C_p = 3$ (%) .

No. Resources	Time constraint	32	38	44	50	56
5	cost	0	-3.78	-3.78	-3.78	-3.78
	time	0	8.82	8.82	8.82	8.82
6	cost	-5.99	-10.45	-10.45	-10.45	-10.45
	time	0	5.88	5.88	5.88	5.88
7	cost	-12.33	-15.73	-15.73	-15.73	-15.73
	time	-2.86	5.71	5.71	5.71	5.71

purposes, the parameters were set to be the same for all project plans. Each of the time constraints (40, 60 and 80 days) was tested using the resource library containing from 7 to 11 available resources. The results are given in Tables 4-5.

The greedy algorithm allocates tasks to all resources. However, some resources have higher unit costs than those of execution of assigned tasks. This increases costs of the project when a greater number of the resources is used. Increasing time constraints gives similar results because the resource library does not change. Unlike the greedy algorithm, the optimization slightly decreases the project cost while the number of resources is growing. As concerns the resource library, resources giving the best gain are chosen, only. Hence, the optimization allows decreasing the project cost if the time constraints are larger. The project schedules may be longer and therefore cheaper results can be found.

The cheapest average cost obtained by the greedy algorithm was 489.9 with the average duration 53,54 days using 7 resources while for the optimization algorithm it was 444.99 with the average duration 58.57 days using 9 resources. The result is cheaper by 9.17% and longer by 9.39%. The shortest average project duration was 51.82 days with the average cost 536.44 and 51.88 days with the average cost 460.8 for the greedy and optimization algorithms, respectively. It gives 0.12% of the time increase and 14.10% of the cost reduction using the same number of resources.

Table 4. Results of the arithmetic mean of 100 optimization tests for the fastest initial schedule

No. resources	Time constraint	40		60		80	
		cost	time	cost	time	cost	time
7	greedy	498,11	55,79	490,97	53,34	491,9	53,72
	opt	458,71	55,26	448,58	55,41	447,67	56,62
	diff (%)	-7,67	-0,82	-8,40	5,18	-8,77	6,16
8	greedy	512,98	54,39	509,01	53,07	516,41	55,45
	opt	460,5	54,33	456,34	54,85	455,77	59,85
	diff (%)	-9,95	0,20	-10,11	4,26	-11,50	8,85
9	greedy	524,88	52,46	523,97	52,81	525,96	54,00
	opt	457,78	52,62	445,25	55,02	444,99	58,57
	diff (%)	-12,59	0,87	-14,86	5,76	-15,21	10,30
10	greedy	537,08	51,83	541,80	53,12	536,44	51,82
	opt	460,80	51,88	454,24	56,01	449,69	57,32
	diff (%)	-14,03	0,32	-16,07	7,17	-15,98	13,26
11	greedy	553,62	52,17	553,92	53,64	556,19	53,07
	opt	457,16	52,65	449,79	58,13	448,22	55,82
	diff (%)	-17,21	1,84	-18,65	11,09	-19,31	6,94

greedy - the results obtained by the greedy algorithm, opt - the results obtained by the optimization algorithm, diff - the difference between the obtained results.

Table 5. Results of the arithmetic mean of 100 optimization tests for the cheapest initial schedule

No. resources	Time constraint	40		60		80	
		cost	time	cost	time	cost	time
7	greedy	496,99	55,91	489,90	53,54	490,44	53,8
	opt	465,11	55,65	453,96	56,11	450,65	57,21
	diff (%)	-6,24	-0,36	-7,19	6,19	-7,92	7,28
8	greedy	509,36	54,44	505,79	53,23	513,38	55,58
	opt	471,76	54,40	460,39	57,03	456,09	61,24
	diff (%)	-7,08	0,16	-8,79	8,99	-10,93	11,40
9	greedy	522,50	52,47	521,75	52,85	523,80	54,06
	opt	466,61	52,51	451,06	57,45	450,87	61,12
	diff (%)	-10,48	0,41	-13,44	11,48	-13,76	15,59
10	greedy	535,11	51,93	539,72	53,16	535,27	51,92
	opt	482,79	52,21	462,49	57,29	457,83	58,84
	diff (%)	-9,62	0,97	-14,26	10,14	-14,35	16,94
11	greedy	552,24	52,20	550,03	53,62	555,75	53,07
	opt	471,70	52,58	455,86	57,32	454,88	57,93
	diff (%)	-14,33	1,43	-17,03	9,19	-18,08	12,00

greedy - the results obtained by the greedy algorithm, opt -the results obtained by the optimization algorithm, diff - the difference between the obtained results.

7 Conclusions

In the paper a new optimization algorithm for RCPSP in the CCPM method is presented. The algorithm optimizes a cost of the project schedule taking into consideration time requirements. The results of optimization show that the algorithm is

able to outperform the initial heuristic project schedules. In fact, when the number of available resources increases, the new technique increases its effectiveness, reducing the project total cost even more with respect to the scheduling methods compared. Even if only 4 resources are available, the algorithm finds a cheaper project schedule. The more costly resources are eliminated or replaced by the cheaper ones. Generally, they are slower, therefore the project duration increases and tends to take up the whole available time.

In the first two experiments, the project duration was not the main goal of the optimization because the project execution cost per day $C_p = 1$. If the value is set to 0, the algorithm will optimize only the cost of the resource allocation. The greater the value the greater influence of the project duration on the project total cost. The last two experiments show that the project schedule may be shortened by using a greater number of resources. Inserting extra resources costs less than executing the project longer by due to resources.

The arithmetic mean of 100 optimization tests was evaluated for the project execution cost per day $C_p = 3$. Statistics show that the gain coming from the optimization varies depending on the number of resources used and the time constraints. Unlike the optimization, the efficiency of the greedy algorithm falls due to a greater number of resources used, and therefore the gain is greater. If the time constraints are higher, the gain is greater as a result of the optimization. Usually, the greater the gain the longer the project duration.

The main strength of the algorithm is a possibility of adjustment to a specific problem. Unlike the greedy algorithm, it does not allocate all resources but as many as needed. Usage of a resource is costly, mainly due to the unit cost. Thus, inserting a new one has to be beneficial. Only resources giving the best gain are selected and therefore the project total cost is reduced. Apart from the project total cost, the presented algorithm takes into account changes of the slack time and therefore has the capacity of getting out of the local minimum. Future work will concentrate on the impact of other factors on the optimization.

References

[1] Wei C., Liu P., Tsai Y., Resource-constrained project management using enhanced theory of constraint, International Journal of Project Management 20 (2002): 561.

[2] Rand G.K., Critical chain: the Theory of Constraints applied to project management, International Journal of Project Management 18 (3) (2000): 173.

[3] Tormos P., Lova A., Competitive Heuristic Solution Technique for Resource-Constrained Project Scheduling, Annals of Operations Research 102 (2001): 65.

[4] Deniziak S., Cost-efficient synthesis of multiprocessor heterogeneous systems, Control and Cybernetics 33 (2004): 341.

[5] Goldratt E.M., Critical Chain, The North River Press Publishing Corporation, Great Barrington (1997).

[6] Steyn H., An investigation into the fundamentals of critical chain project management, International Journal of Project Management 19 (2000): 363.

[7] Herroelen W., Leus R., On the merits and pitfalls of critical chain scheduling, Journal of Operations Management 19 (2001): 559.

[8] Deniziak S., Sapiecha K., Kosynteza rozproszonych systemów heterogenicznych ("Hardware-software co-design of heterogenous distributed systems"), III Krajowa Konferencja: Metody I systemy komputerowe w badaniach naukowych i projektowaniu inżynierskim, Kraków (2001): 437 (in Polish).

[9] Tukel O. I., Rom W. O., Eksioglu S. D., An investigation of buffer sizing techniques in critical chain schedulin, European Journal of Operational Research 172 (2006): 401.