Tabu Search approach for Multi–Skill Resource–Constrained Project Scheduling Problem

Marek E. Skowroński, Paweł B. Myszkowski, Marcin Adamski, Paweł Kwiatek
Institute of Informatics, Department of Artificial Intelligence
Faculty of Computer Science & Management, Wrocław University of Technology, Poland
Email: {pawel.myszkowski, m.e.skowronski}@pwr.woc.pl, {183750,180040}@student.pwr.wroc.pl

Abstract—**In this article two approaches of Tabu Search in Multi–Skill Resource–Constrained Project Scheduling Problem (MS–RCPSP) have been proposed, based on different neighbourhood generation methods. The first approach assumes swapping resources assigned to pair of tasks, while the second one proposes assigning any resource that could perform given task. Both approaches need to respect the skill constraints. The objective of this paper is to research the usability and robustness of proposed approaches in solving MS–RCPSP. Experiments have been performed using artificially created dataset instances, based on real–world instances, got from Volvo IT and verified by experienced project manager. Presented results show that Tabu Search (TS) based methods are efficient approaches that could be developed in the further work.**

I. Introduction

RESOURCE-Constrained Project Scheduling Problem (RCPSP) is a classical problem (e.g. [3], [11]). Its objective is to assign scarce resources to given tasks to make the project schedule as short / cheap as possible. Because of its combinatorial nature, it is known as NP–hard [1]. It means that it is not possible to find optimal solution in finite, polynomial time. It also suggests using some soft computing methods, which do not always provide optimal solutions, but usually sub–optimal acceptable solutions in reasonable processing time. There are several metaheuristics [12] used for solving RCPSP and its extensions – Evolutionary Algorithms (EA) [4], [6], [7], [9], [15], [22], [26], Simulated Annealing (SA) [2], [5], [14], Ant Colony Optimization (ACO) [16] or Tabu Search (TS) [17], [18], [20], [21], [24]. The last of mentioned methods would be investigated in this paper. We decided to develop TS–based methods because of its relative simplicity in comparison to other metaheuristics. On the other hand, it could be comparably effective to other methods.

RCPSP could be extended by the skills domain to Multi–Skill Resource–Constrained Project Scheduling Problem (MS–RCPSP) [8], [19]. Each task requires given skill in specified familiarity level, while each resource disposes some skills pool. It causes that not every resource can perform every task and the schedule is more difficult to be built.

(MS–)RCPSP is very practical problem. Project managers in significant companies still often need to schedule their projects manually, what is extremely time consuming. Because of human fail-ability, manually scheduling could also cause obtaining infeasible schedule solutions, where not every constraints are satisfied. Providing computer–aided methods could save a lot of time and ensures that every designed constraints would be met. What is also important, automated AI–based methods mentioned earlier could give final solution in minutes, while experienced project manager needs a few hours to prepare a schedule for the same–sized project (with the same number of resources and tasks).

Proposed (MS–)RCPSP definition has been designed in strict cooperation with Volvo IT Department in Wroclaw. Provided requirements, assumptions and constraints have been considered, approved and then included in problem description, what enlarges the practical value of this paper.

The rest of the paper is organised as follows. Section II describes other approaches to solve the (MS–)RCPSP using metaheuristics, especially TS. Section III presents the MS–RCPSP problem statement, while Section IV describes the approaches proposed in this paper. Section V provides conducted experiments of proposed methods in a given dataset. Finally, section VI presents the conclusions of obtained results and suggests some ideas of future work.

II. Related work

To make a structured overview of related work, we divided it into groups of methods. EA–based [4], [6], [7], [9], [15], [22], [26] methods mostly use task–vector representation of an individual. The order in a vector describes the order of tasks' performance in a project. Schedule can be generated in a serial [7], [22], [25], [26] or parallel [12] generation scheme. Because of proposed individuals representation, mostly semi–blind crossover and mutation operators could be applied, like swap [22], [23], [26] or insert [7], [25] mutation and one–point [7], [9], [25], two–point [7], position–based [26] or uniform [15] crossover. Some papers presents more dedicated operators, like peak crossover in [22].

SA–based approaches [2], [5], [14] are also often investigated. In [2] classical precedence–feasible activity list representation with the serial generation scheme is used. The neighbourhood is created by insertion method of regarded task within its last predecessor and the first successor. Cooling scheme assumes annealing of multiple starting solutions. The other approach of generating a neighbourhood is presented in [5], where the new solution in neighbourhood is created randomly, preserving precedence– and resource constraints. In [5] some hybrid of TS and SA has been proposed, where the memory of moves has been added to traditional SA approach.

The last of mentioned approaches, presented in [14] assumes investigating set of SA–based methods with multiple start, where the initial solution is prepared in various ways, e. g. randomly or using scheduling priority rules.

In TS–based methods [17], [18], [20], [21], [24] several elements can be variously implemented. The neighbourhood can be created by swapping or inserting activities [20] in the task–order list. In [20] the tabu list size is dependent on the number of critical tasks to be scheduled in a project. What is more, several strategies of diversification have been proposed there, while not better solutions have been recently found and strategies of intensification, where a solution with very good quality has been found. In [21] a starting solution is obtained using the minimal slack heuristic rule (MINSLK). The same approach assumes regarding move as an exchange of two activities positions in the activity list and the *difference move* measure is introduced, computed as the difference between solutions which are regarded by given move. An aspiration criterion is also proposed in [21] – if the move is on a tabu list, but produced solution (S) is better than the best (B) found so far, S replaces B. The termination criterion is set as a number of iterations which could not find better solution. Similar starting solution, using priority rule heuristics, is proposed in [17], [18], where classical neighbourhood generation methods have been also proposed: swap and insertion. In an approach presented in [24] classical TS method has been extended by local search procedures, to find better solutions.

Other (MS–)RCPSP heuristic solutions have been included in [12], [13], where local search methods and other population based methods have been also presented, e. g. ACO–based approach, that was also investigated in [16].

III. Problem statement

In MS–RCPSP we assume that project consists of several main elements: tasks, precedence relations, resources and skills. **Tasks** are described by their start and finish dates, duration and skill required to be performed. Tasks are often related by **precedence relations** – some tasks cannot be started before their potential predecessors would not be finished. Analogously, task's successor cannot be started before it's finish date. **Resources** are described by their salary and **skills** pools they own. As it was mentioned earlier, not every resource can perform each task, when given resource does not own skill required by given task. Only one resource can be assigned to given task.

A. Conflicts, solution's feasibility

Furthermore given resource cannot be assigned to more than one task in an overlapping period of time – if such a situation occurs, we defined it as a **conflict** and it has to be resolved. Conflict solving is made by shifting one of conflicted tasks just after the other one in the timeline. Because conflict resolving could disturb the precedence relations constraints, they should be preserved after each conflict resolution. Without resolving conflicts and preserving critical path constraints, produced solutions would be infeasible and could not be regarded as final, correct schedules. Solutions where resource is assigned to task when it does not own required skill in specified level is also regarded as infeasible.

B. Evaluation function

The (MS–)RCPSP objective is to schedule the project as quick or / and cheap as possible. It could be presented as **multi–objective optimization problem**: project schedule's **duration minimization** and project's performance **cost minimization**. Those objectives are generally in opposition. Reducing a project duration could cause enlarging a project's cost and vice versa. That is why, the *happy medium* is often sought – how to reduce the value of first objective, to get larger but still acceptable value of the second objective. In project scheduling problem domain it is often called *time & cost trade–off problem.*

A single project schedule (PS) solution is represented as a resource–to–task assignments vector $A = [a_i^j] : i = 1, 2, ..., t$, $j = 1, 2, ..., r$, where a_i^j represents the assignment of j-resource to i-task, t-number of tasks and r - number of resources.

To evaluate a solution, we proposed weighted, evaluation function:

$$\min f(PS) = w_\tau f_\tau(PS) + (1 - w_\tau) f_c(PS) \qquad (1)$$

where: w_τ - weight of duration component, $f_\tau(PS)$ - duration evaluation component, $f_c(PS)$ - cost evaluation component.

Components' weights are applied to tune up the importance of time and/or cost factor in the given project optimization. The duration–aided optimization means setting time weight close to value 1 that automatically reduce the cost weight near zero. Analogously weights in the cost–aided optimization are tuned.

The time component $f_\tau(PS)$ is calculated as follows:

$$f_\tau(PS) = \frac{d_{PS}}{\tau_{max}} \qquad (2)$$

where: d_{PS} - duration of schedule PS, τ_{max} - maximal possible duration of schedule PS, computed as the sum of all tasks' duration. The cost component $f_c(PS)$ is defined as follows:

$$f_c(PS) = \frac{\sum_{i=1}^{t} c_i^j}{c_{max} - c_{min}} \qquad (3)$$

where: c_i^j - standard cost of performing task i by resource j, c_{min} - minimal schedule cost – a total cost of all tasks assigned to the cheapest resource, c_{max} - maximal schedule cost – a total cost of all tasks assigned to the most expensive resource. c_{max} and c_{min} do not involve skill constraints. It means that c_{min} value could be reached only for non–feasible solution. Analogously to c_{max}.

C. Solution space size

Given number of tasks and number of resources, we can estimate the solution space size, as:

$$SS(t, r) = t! * r^t \qquad (4)$$

However, that estimation takes also into account non–feasible solutions, because skill–constraints are not satisfied. To give an example, let's assume $t = 10$ and $r = 5$ – without any precedence relations we get $SS(10,5) = 3.54 * 10^{13}$ combinations. It is worth mentioning, that each task can be placed only once in schedule, but resources could be assigned more often. An extreme situation occurs if the same one resource would be assigned to perform each task.

Large solution space size makes impossible checking each of the combinations manually. However, space includes also non–feasible solutions that do not satisfy defined conditions. Moreover, given example is a simplification and in real world problems we meet a higher number tasks (about $t = 100$) and resources ($r = 20$) – it gives $SS(100,20) = 1,19 * 10^{288}$ of all solutions. As solution space is constrained, relatively large and the MS–RCPSP problem is NP–hard it proves the legitimacy of metaheuristics usage.

IV. Proposed approaches

The overall TS approach is to avoid entrainment in cycles by forbidding moves which take the solution, in the next iteration, to point in the solution space previously visited (hence tabu). TS proceeds according to the supposition that there is no point in accepting a new (poor) solution unless it is to avoid a path already investigated. This insures new regions of a problems solution space will be investigated in with the goal of avoiding local minima and ultimately finding the desired solution.

To perform, TS needs some parameters to be set. The neighbourhood size defines the size of neighbourhood, that is created based on currently best solution and from which the better solution is sought. Number of iterations stands how many times the neighbourhood would be generated and sought for better solution (stopping criterion). Tabu list size tells, how many recent swaps should be recorded in the list of forbidden ones. For simplicity in this paper, we decided not to introduce any aspiration criteria for TS in our investigations.

A. Initial solution generation

Initial solution is straightforwardly loaded from a file that contains the project data. The project schedule had been previously prepared using EA with classical crossover (one–point) and mutation (swap–based) operators that satisfies all constraints (precedence, skills and conflicts resolving). Therefore, it could be regarded as a feasible solution. The feasibility of initial solutions has been confirmed and approved by experienced project manager from Volvo IT.

B. Neighbourhood generation

To generate a neighbourhood, a new solution generation method has to be provided. We designed new solution as generated in two general steps: setting assignments to tasks and then build the schedule, respecting precedence constraints and resolve conflicts. We proposed and compared two methods of setting assignments to tasks. The first one bases on the swapping resources within the pair of tasks, while the second approach assumes assigning any resource that is capable of performing given task. Above mentioned rules provide the feasibility of generated solution.

Swap–based neighbourhood (SBN): In this approach potential solution is created in the following way. For given assignment another is sought that enables swapping resources between those assignments. Swapping is possible only if both resources are capable of being assigned to tasks related to chosen assignments – skill constraints are preserved. If no other assignment, which can be used for swapping with given one, is found, new assignment is selected and the procedure of searching *swapping mate* for it is repeated.

Random–based neighbourhood (RBN): A new potential solution, that can be visited by the TS procedure, is created by changing the assignment of given task in following way. List of resources that can perform given task (dispose skill required by task) is obtained and then any different resource than currently assigned is chosen.

Fig.1 presents schematically ways of generating new solutions that can be included into new neighbourhood. OK signs in this figure presents which resources can perform given tasks. In RBN, we consider only one task, for which new resource is sought. In SBN, we need to find two tasks and resources that are capable to swap assignments between them. The *QX.Y* notation describes skills owned by resources or required by particular task to be performed. In provided example task *T1* requires skill *Q2* at proficiency level *2* to be performed. Moreover, resource *R1* owns skill *Q1* at proficiency level equal *3* and skill *Q2* at proficiency level equal *2*. Thus R1 is able to perform *T1*. *R1* is also able to perform task *T4* because *R1* owns skill (*Q1*) required by *T4* (level 1) with higher proficiency level (*3*). Analogously R1 is also capable of performing *T3*.

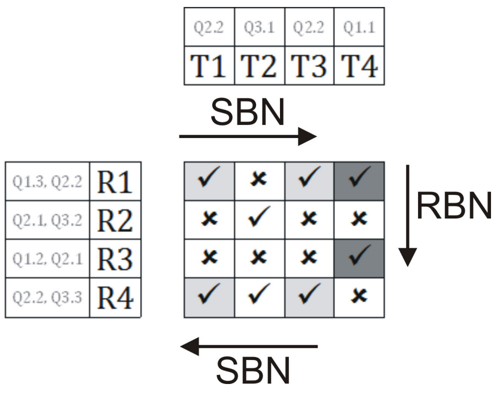

Fig. 1. Illustration of neighbourhood generation methods

Example results of neighbourhood generation methods have been presented in the Fig.2. For SBN only resources for tasks $T1$ and $T3$ could be swapped. Hence, as a result, swapping of mentioned pair of resources assigned to indicated tasks has been performed. For RBN other resource could be assigned to every task. In this example, we decided to change the assignment of $T4$. This example explains the main difference – SBN involves two tasks, while RBN changes assignment of only one task.

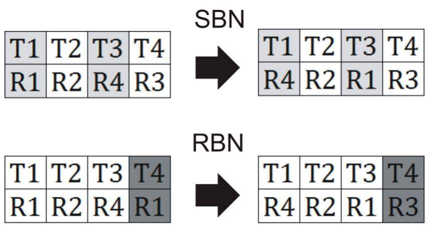

Fig. 2. Example of neighbourhood generation methods

After setting an assignments' vector, the schedule is built in a few steps. First, all tasks have their start time dates set to the project's start date. Then the conflict fixing mutually with the critical path preserving methods are launched, until no conflicts would be found and the schedule could be regarded as a feasible. Neighbourhood generation methods are designed not to allow generating infeasible solutions.

C. Move description

A move describes how the solution B has been created in the base of solution A. In our approach, the move stores the information about task and resource related to assignments that have been changed in neighbourhood generation procedure. In SBN the information about pair of tasks and pair of resources within which the swap has been performed.

Let's see some example: task A has been assigned to resource 1 - *A(1)* and task B has been assigned to resource *B(2)*. After the swap the situation is as follows: *A(2)* and *B(1)*. The information stored in move is as follows: *A(1),B(2)→A(2),B(1)*.

In RBN less information is stored in particular move. It is because this neighbourhood creation method involves only one task. Hence, using previous example, the information stored in a move looks like: *A(1)→A(2)*.

V. Experiments and results

The goal of conducted experiments was to compare two different approaches of creating neighbourhood in TS and investigate whether proposed TS approaches could be used in solving MS–RCPSP in effective way. To evaluate solution – the resulted project schedule – its duration time ([days]) and performance cost ([c.u][1]) were investigated.

A. Dataset

Due to evaluate not only the project schedule duration, but also the cost of the schedule, we cannot use the standard PSPLIB benchmark dataset [10], that does not contain any information about the task performance cost. What is more, PSPLIB dataset instances do not reflect the MS–RCPSP. Hence, we prepared the dataset, containing six project instances, that have been artificially created[2], in a base of real–world instances, got from the Volvo IT Department in Wroclaw.

TABLE I
MS–RCPSP dataset description

Property	**D1**	**D2**	**D3**	**D4**	**D5**	**D6**
Tasks	100	100	100	200	200	200
Resources	20	10	5	40	20	10
Skills	9	9	9	9	9	9
Relations	20	26	22	133	148	129

The dataset summary has been presented in the Table I. There are two groups of created project instances: one contains 100 tasks and the second – 200 tasks. Within the group, project instances are varied by number of available resources and the precedence relationship complexity. It led to create three different project instances both with 100 and 200 tasks. The skill variety has been set up to constant 9 different skill types for each project instance, while any resource can dispose no more than six different skill types. Because of the different resources and relations number, the scheduling complexity for each project is varied.

B. Experiments' set–up

The experiments have been divided into investigating the influence of TS parameters' configurations for project duration and performance cost in three various components' weights in evaluation function: duration optimization (DO: $w_t = 1$), balanced optimization (BO: $w_t = 0.5$) and cost optimization (CO: $w_t = 0$). Each experiment for given parameter configuration has been repeated ten times.

To present the average results in detail, a standard deviation measure should be introduced and applied to each average value. However, to keep the results' presentation clear and simple, we decided not to present it. It is because we have obtained the standard deviation measure values and they generally were not much significant. Only 2 of conducted experiments provided the standard deviation value bigger than 10% of the average.

The processing time of both approaches was varied within the range from about 40 seconds (D1 project instance with neighbourhood size = 10) to about 900 seconds (D6 project instance with neighbourhood size = 45)[3].

C. Experiments' performance

During the experiments, following parameters have been examined: tabu list size and neighbourhood size. Each parameter configuration was ran in both neighbourhood creation methods. Number of iterations has not been investigated. During experiments we noticed that after, in average, 200 iterations, no better solutions have been produced, regardless the remaining parameters' configuration and used dataset instance. Based on that observation, we set up the iterations size as a constant value equal to 250 iterations. Neighbourhood size and tabu list size parameters have been chosen experimentally.

For DO and CO, the best result has been indicated as the one with the smallest value of duration and cost respectively.

[1]Currency units
[2]http://www.ii.pwr.wroc.pl/~myszkowski/scheduling
[3]Intel Core 2 Duo P8700 (2.53 GHz at each core) and 4 GB memory RAM.

If there was more than one result with the smallest value of time or cost, the one with the smallest value of the second optimization aspect has been chosen as the best. For BO the best result has been obtained based on the smallest value of evaluation function.

D. Results' discussion

Analysing results we can notice that optimization direction (more duration– or more cost–oriented) is related with the neighbourhood creation method. Looking at the Tab.**??** the best results for DO have been found in SBN method part in 5 from 6 dataset instances. On the other hand, the best CO results have been all obtained using RBN. An interesting observation regards the results for BO optimization mode. Project instances consisting 100 tasks have been optimized the best using SBN method, while the best optimization results for project instances consisting 200 tasks have been found in the part of table regarding the RBN.

Analysing the influence of examined TS parameters to obtained results, some following observations could be indicated. Taking into account the influence of the neighbourhood size to the DO results, we can suggest using smaller neighbourhood size values. It is because that the best DO results have been obtained for the $N = 10$ (4 best solutions – *winners*). $N = 20$ and $N = 45$ provided best results only once per each N value parameters' configurations. On the other hand, all best solutions for CO have been found for $N = 45$. It could lead to some general conclusion that enlarging the search space could be beneficial for cost optimization but does not have to help obtaining better solutions, where project duration is the core aspect of optimization. Above mentioned observation could be derived to BO mode but with a respect to a conclusion made in previous paragraph – for BO best solutions of project instances consisting 200 tasks have been obtained for RBN with the $N = 45$. BO results for project instances consisting 100 tasks cannot allow us to make any general conclusions.

We have not found any interesting relationship between the tabu list size and the optimization's robustness, regardless the configuration of remaining parameters, chosen optimization mode or even project instance. We have found 4 best obtained solutions in DO or BO for the $TL = 10$, but that value has not been confirmed in remaining results as potentially good in optimization. It is very difficult to make any general conclusions and assumptions regarding the influence of tabu list size into the potential of optimization.

VI. Conclusions and further work

The best obtained solutions for each dataset instance in every optimization mode have been presented in Tab.II. For each solution a project duration and performance cost has been presented with the description of configuration, for which the solution has been found. The notation for that description is as follows: *neighbourhood creation method (neighbourhood size, tabu list size)*. E.g., SBN(10,15) means that the solution has been obtained for swap–based neighbourhood solution with neighbourhood size equal to 10 and tabu list size set to 15. That summary table briefly summarizes which neighbourhood creation mode is preferred for a given optimization mode.

Obtained results could lead to a conclusion that neighbourhood creation strategy has significant influence on the optimization ability. Using SBN, better solutions in duration optimization were obtained. SBN provided the best duration optimization results in 5 of 6 project instances. On the other hand RBN generally is more effective in cost optimization. Hence, different approaches could be used for different optimization modes. Despite that, end user would have to be always aware of opposite–character of project duration and performance cost objectives, which cause enlarging one aspect where the optimization is focused on the second one.

TABLE II
SUMMARY TABLE – BEST OBTAINED RESULTS IN INVESTIGATED OPTIMIZATION MODES

ID	DO		BO		CO	
	days	cost	days	cost	days	cost
D1	SBN(10,15)		SBN(45,10)		RBN(45,10)	
	32	40656	37	38939	129	30750
D2	RBN(10,10)		SBN(45,7)		RBN(45,10)	
	33	43542	49	34240	179	26444
D3	SBN(10,7)		SBN(20,7)		RBN(45,7)	
	51	40054	61	36100	133	31645
D4	SBN(45,10)		RBN(20,15)		RBN(45,7)	
	92	88720	125	50438	254	46371
D5	SBN(20,15)		RBN(45,10)		RBN(45,10)	
	179	80448	184	54181	481	52425
D6	SBN(10,15)		RBN(45,15)		RBN(45,15)	
	199	97978	222	75996	330	73126

To lift end–user of setting weights in evaluation function, both TS approaches could be merged in Pareto–based related mechanism. Mixing solutions from both approaches and choosing only the best, non–dominated ones could give the end user more flexibility of choosing the most appropriate schedule of proposed solutions pool. Furthermore two interesting future work directions could be indicated: investigating and applying aspiration criteria for both proposed approaches and creating an initial solution in more directed ways. To cope with the second mentioned idea, scheduling priority rules could be applied, while the first of proposed research directions could enhance the proposed method's robustness.

It would be also useful to compare obtained results with simple hill–climbing method – like TS without storing information about moves. It would provide information about the real usability of applying TS for MS–RCPSP.

References

[1] Blazewicz J., Lenstra J.K., Rinnooy Kan A.H.G.; Scheduling subject to resource constraints: Classification and complexity, Discrete Applied Mathematics (5), pp. 11-24, 1983.

[2] Bouleimen K., Lecocq H.; A new efficient simulated annealing algorithm for the resource-constrained project scheduling problem and its multiple mode version, European Journal of Operational Research (149), pp. 268-281, 2003.

[3] Brucker P., Drexl A., Mohring R., Neumann K., Pesch E.; Resource–constrained project scheduling: Notation, classification, models, and methods, European Journal of Operational Research (112), pp. 3–41, 1998.

[4] Chen Z., Chyu C.; An Evolutionary Algorithm with Multi–Local Search for the Resource-Constrained Project Scheduling Problem, Intelligent Information Management (2), pp. 220–226, 2010.
[5] Das P. P., Acharyya S.; Simulated Annealing Variants for Solving Resource Constrained Project Scheduling Problem: A Comparative Study, Proceedings of 14th International Conference on Computer and Information Technology, pp. 469–474, 2011.
[6] Hartmann S.; A competitive genetic algorithm for resource–constrained project scheduling, Naval Research Logistics (45), pp. 733–750, 1998.
[7] Hindi K. S., Yang H., Fleszar K.; An Evolutionary Algorithm for Resource–Constrained Project Scheduling, IEEE Transactions on evolutionary computation (6), pp. 512–518, 2002.
[8] Kadrou Y., Najid N.M.; A new heuristic to solve RCPSP with multiple execution modes and Multi-Skilled Labor , IMACS Multiconference on Computational Engineering in Systems Applications (CESA), pp. 1302–1309, 2006,
[9] Kim J.L.; Permutation–based elitist genetic algorithm using serial scheme for large–sized resource–constrained project scheduling, Proceedings of the 2007 Winter Conference Simulation Conference, pp. 2112–2118, 2007.
[10] Kolisch R., Sprecher A., PSPLIB - A project scheduling problem library, European Journal of Operational Research (96), pp. 205–216, 1996.
[11] Kolisch R., Serial and parallel resource-constrained project scheduling methods revisited: Theory and computation, European Journal of Operational Research (90), pp. 320–333, 1996.
[12] Kolisch R., Hartmann S., Experimental evaluation of state-of-the-art heuristics for the resource-constrained project scheduling problem, European Journal of Operational Research (127), pp. 394–407, 2000.
[13] Kolisch R., Hartmann S., Experimental investigation of heuristics for resource-constrained project scheduling: An update, European Journal of Operational Research (174), pp. 23-37, 2006.
[14] Liu S., Tukel O.I., Rom W.; Flexible Scheduling Approach for Resource-Constrained Project Scheduling Problems, Proceedings of the 7th World Congress on Intelligent Control and Automation, pp. 3522–3526, 2008.
[15] Mendes J.J.M., Gonvalces J.F., Resende M.G.C.; A random key based genetic algorithm for the resource constrained project scheduling problem, Computers & Operations Research (36), pp. 92-109, 2009.
[16] Merkle D., Mittendorf M., Schmeck H.; Ant Colony Optimization for Resource–Constrained Project Scheduling, IEEE Transactions on Evolutionary Computation (6/4), pp. 333–346, 2002.
[17] Pan H.I., Hsaio P.W., Chen K.Y.; A study of project scheduling optimization using Tabu Search algorithm, Engineering Applications of Artificial Intelligence (21), pp. 1101-1112, 2008.
[18] Pan N.H., Lee M.L., Chen K.Y.; Improved Tabu Search Algorithm Application in RCPSP, Proceedings of the International MultiConference of Engineers and Computer Scientists (Vol I), 2009.
[19] Santos M., Tereso A. P.; On the multi-mode, multi-skill resource constrained project scheduling problem - computational results, Soft Computing in Industrial Applications, Advances in Intelligent and Soft Computing (96), pp. 239–248, 2011.
[20] Thomas P. R., Salhi S.; A Tabu Search Approach for the Resource Constrained Project Scheduling Problem, Journal of Heuristics (4), pp. 123-139, 1998.
[21] Tsai Y.W., Gemmill D. D.; Using tabu search to schedule activities of stochastic resource–constrained projects, European Journal of Operational Research (111), pp. 129–141, 1998.
[22] Valls V., Ballestin F., Quintanilla S.; A hybrid genetic algorithm for the resource–constrained project scheduling problem, European Journal of Operational Research (185), pp. 495-508, 2008.
[23] Valls V., Ballestin F., Quinanilla S.; An Evolutionary Approach to the Resource-Constrained Project Scheduling Problem, MIC2001 - 4th Metaheuristics International Conference, pp. 217–220, 2001.
[24] Verhoeven M. G. A.; Tabu search for resource-constrained scheduling, European Journal of Operational Research (106), pp. 266–276, 1998.
[25] Wang H., Lin D., Li M.Q.; A competitive genetic algorithm for Resource–Constrained Project Scheduling Problem, Proceedings of the Fourth International Conference on Machine Learning and Cybernetics, pp. 2945–2949, 2005.
[26] Zhang H., Xu H., Peng W., A Genetic Algorithm for Solving RCPSP, 2008 International Symposium on Computer Science and Computational Technology, pp. 246–249, 2008.