Proceedings of the 2013 Federated Conference on
Computer Science and Information Systems pp. 159–166

Novel heuristic solutions for Multi–Skill Resource–Constrained Project Scheduling Problem

Paweł B. Myszkowski, Marek E. Skowroński, Łukasz Podlodowski
Institute of Informatics, Department of Artificial Intelligence
Faculty of Computer Science & Management, Wrocław University of Technology, Poland
Email: pawel.myszkowski@pwr.wroc.pl, m.e.skowronski@pwr.woc.pl, 168037@student.pwr.wroc.pl

***Abstract*—In this article some novel scheduling heuristics for Multi–Skill Resource–Constrained Project Scheduling Problem have been proposed and compared to state-of-the-art priority rules, based on task duration, resource salaries and precedence relations. New heuristics stand an aggregation of known methods, but are enhanced by skills domain. The goal of the paper is to investigate, whether evaluated methods can be used as robustness enhancement tools in metaheuristics, mostly evolutionary algorithms. Experiments have been performed using artificially created dataset instances, based on real–world instances. Obtained results prove that such methods stand interesting feature that can be included to more complex methods and increase their robustness.**

I. Introduction

RESOURCE-Constrained Project Scheduling Problem (RCPSP) is one of the most widely described [1], [4], [10], [11], [12] combinatorial problems in the literature. RCPSP describes the problem [4], where the set of predefined tasks and resources are given. The objective of RCPSP is to assign tasks to resources in the way to make the overall project schedule as cheaper and shorter as possible – time & cost optimization. However, there are many constraints that have to be satisfied to make the schedule feasible. Given resource cannot be assigned to more than one task in specified time. Moreover, the precedence relations between tasks have to be obeyed.

The RCPSP can be extended to Multi–Skill RCPSP (MS–RCPSP) [16], where the set of predefined skills pool is given. In MS–RCPSP each task requires some skill in specified level to be performed. Each resource disposes some subset of skills. Hence not every resource can perform every task. The ability of performance specified task has to be checked before the assignment. If specified resource is assigned to given task, even it does not cover required skills, the project schedule become infeasible.

As MS–RCPSP is a combinatorial, NP–hard problem [1], there is no optimal solution that can be found in acceptable, finite time. That is why the soft computing methods are often used, mostly metaheuristics. However, simpler heuristics are often used. Usually they obtain suboptimal solution, but their advantages are different - they are much faster than more complex metaheuristics, like Evolutionary Algorithms (EA) [8], [14] or Ant Colony Optimization [13]. What is more, heuristics provide repeatable results, while the non-deterministic character of metaheuristics can be regarded as one of the most important disadvantage.

Heuristics used to solve the RCPSP and its extensions are called *scheduling priority rules* (SPR) [10]. They are usually related to the main elements that create the problem description - tasks with their precedence relations, resources or skills. In this paper five SPR will be presented and evaluated on proposed dataset: known from the literature, like based on a task duration, a resource salary, and task successors' list size. Some novel approaches are proposed, based on an adjustment between required and available skills.

Proposed SPR could be applied to more complex methods, like metaheuristics, i.e. as a method of generating initial population in EA, to make the search more effective. Furthermore SPR can be combined with the EA–based approaches, constructing hybrid metaheuristics, where those methods could be regarded as a local search methods. The goal of this paper is to investigate whether proposed SPR are effective enough to recommend them to such applications.

The rest of the paper is organised as follows. Section II describes other approaches to solve the (MS)–RCPSP with the SPR paradigm. Section III describes proposed SPR, while Section IV provides experiments made with described approaches, based on proposed dataset. Finally, Section V presents discussion of obtained results, while Section VI provides the conclusions and some ideas of future work.

II. Related work

SPR have been widely investigated in the literature (e.g. [2], [3], [10]). In [2] some standard priority rules has been presented: minimum total slack (MINSLK), minimum late finish time (MINLFT) and minimum processing time (MINPTM). Furthermore some less popular rules have been introduced: maximum number of immediate successors (MAXNIS), maximum remaining work (MAXRWK) and maximum processing time (MAXPTM). Proposed priority rules has been tested on a randomly created dataset. Results have shown that the most sufficient priority rule for proposed dataset is MAXNIS.

In [3] most of above mentioned priority rules have been examined, like MINSLK, MINLFT, MINPTM and MAXPTM. The work–related priority rules has been changed to total work: minimum total work (MINTWRK) and maximum total work (MAXTWRK). The MINPTM and MAXPTM has been

978-1-4673-4471-5/$25.00 © 2013, IEEE

examined, however they have been called in different ways—shortest operation first and longest operation first, but their meaning is the same to above mentioned. Authors presented some novel heuristics, based on tasks criticality or load balancing factor, which became more suitable for solving RCPSP than standard ones, as it was presented in the results section.

An approach presented in [5] proposes priority rule based on resource availability. After each task is scheduled, the resource availability is computed and it influences on further resource–to–task assignments. An opposite measure for resource availability - moving resource strength (MRS) is introduced. The bigger the MRS value is, the more busy the resource is and the probability of choosing it for further tasks is smaller. MRS is computed locally – for given time window in a project. The MRS is included for most of typical priority rules (LFT, SLK, MTS).

Some researchers also proposed combined approaches, where priority rules are included in EA [11], [17]. In those approaches priority rules are used to generate the initial solution from which the whole initial population could be generated. Other approach [6] proposes using priority rule as a general method of creating the whole schedule from the resource–to–task assignments. Resources are assigned to given tasks and then the schedule is built by adding following tasks in given order, preserving resource– and precedence–constraints. In that approach SPR based on successors definition has been investigated.

The standard notation for scheduling has been presented in [4]. It introduces the formal language for describing the scheduling problem using SPR in comparison to metaheuristics and other more complex methods. It also describes other known priority rules and methods to solve various RCPSPs with different objective functions.

Most of state-of-the-art scheduling priority rules has been tested on the PSPLIB [9] benchmark dataset. The experiments and results have been widely described in [10], [11]. An update of performed evaluation has been performed in [12] and some novel methods have been also proposed there. Moreover, priority rule heuristic methods have been compared to metaheuristics, i. e. genetic algorithms (GA) [7]. Even if GA produces comparative results, priority rule heuristics are indispensable to create initial solution.

Solutions based on priority rules for multi–mode RCPSP (MM–RCPSP) are presented in [15]. MM–RCPSP assumes that task can be performed in several modes while every mode could cause various task's performance cost or its duration. For that problem typical priority rules have been proposed: LFT, SLK, NIS, LPM and SPM. Obtained results could lead to conclusion that LFT and LST priority rules are the most suitable.

III. Scheduling priority rules

A priority rule contains information to construct a list of tasks that ranks all project tasks in a certain order to determine the priorities in which the tasks are assigned to the project schedule. Such a list is constructed in order to assign priorities to tasks based on the following project information:

- task description: information about time or cost estimates of the tasks also determines the task priorities.
- precedence constraint information: information obtained from simple critical path scheduling tools determines the task priorities.
- resource skills information: information about the project resources and skills pools they cover, determines the task priorities.

The constructed list of tasks is then used and tasks are removed one by one from the list and are put in the schedule in the heuristic scheduling process.

Constraints that have to be preserved for each priority rule:

- Resource can be assigned to specified task only if it owns the required skill in required or higher familiarity level.
- If task has predecessors, it cannot be started before the last of predecessors would be finished.
- If a *conflict* occurs during the scheduling the specified task t, it has to be removed by shifting the start time of t just after the finish time of conflict-related task. The *conflict* is defined as a situation, when given resource is assigned to more than one tasks, which are performed in overlapping periods of time.

A. Simple priority rules

We proposed three simple priority rules, where tasks are ordered to be performed by satisfying simple conditions, presented below.

1) Task duration–based: In task duration–based priority rule (TD) tasks are ordered by their duration (ascending or descending). Then resources are assigned to tasks in predefined order. If there is more than one resource that can be assigned to specified task, the cheapest one (with the smallest standard rate) is selected. If there is more than one resource with the smallest standard rate, the first one from the pool is taken to be assigned. The TD bases on MINPTM and MAXPTM state-of-the-art priority rules.

2) Resource salary–based: In resource salary–based priority rule (RS) resources are sorted by their standard rate salary and then are assigned to tasks in an order the tasks were added to the project description. If there is more than one resource with the smallest standard rate, the first one from the pool is taken to be assigned. The tasks' order is taken directly from the dataset instance.

3) Successors' list size–based: In successors' list size–based priority rule (SLS) tasks are sorted by the size of successors' list. Then resources are assigned to tasks in defined order. The SLS bases on the state-of-the-art MAXNIS priority rule. If there is more than one resource that can perform given task, the cheapest one is taken (the same like in TD). If there is still more than one resource, that can be chosen, the first one from the pool is taken.

B. Complex heuristics

Above mentioned methods base on one or maximum two sorting criteria that define the tasks order. However, those methods do not utilize sufficiently the information about the resource load balancing or skills covered by resource and skills required by tasks. Thus we propose novel SPR approaches, based on mentioned aspects.

1) Skill adjustment–based: In skill adjustment–based heuristic (SA) skills are ordered by the adjustment measure (π), compared as a difference between number of tasks requiring specified skill and number of resources owning it and then normalized to [0;1] values. For each skill, the list of tasks that require it in specified level is obtained. If the list size is bigger than one, tasks are sorted by their duration time and the first one from the ordered list is taken to be assigned by the first resource in ordered resource list by standard rate salary. If there is more than one resource than can perform specified task in the same cost, the first one from the list is used.

This priority rule has been extended by one decision to made - whether the adjustment measure should be computed only once where the following skill is taken into consideration or should be computed after each task would be assigned. In extreme, if each task had different required skill, this decision would be useless. However, such situation occurs very rarely. Hence, it was useful to check the importance of that decision in the context of obtained results. This method has been presented as a pseudo–code in the Algorithm 1.

Because the sorting order for π, task duration and resource salary can be ascending or descending and the dynamic adjustment decision had to be included, we proposed four two-state parameters: P_π regards the sorting order of π, P_d concerns task duration sorting order, P_s declares the sorting order for resource salary and the last one - P_a treats whether the dynamic adjustment feature is active or disabled. Each of those parameter can be set to A (ascending for sorting order and active for dynamic adjustment feature) or D (descending sorting order or disabled dynamic adjustment feature).

2) Resource properties – based: In resource properties – based heuristic (RP) resource that should be assigned to specified task is selected by using a complex method, where many conditions are checked. First, only valid resources are obtained – who dispose required skill in specified level. Then the subset of obtained resources is created that contains only resources, who have specified skill in the highest / lowest acceptable level. If there is more than one resource that satisfies above condition, resources are compared by their free time during the project lifetime. The free time is computed for each resource as a difference between the project duration and sum of hours that reflects all tasks assigned to specified resource. If there are still more than one resource that can be assigned, the cheaper / more expensive one is chosen, by comparing their standard rate. If there are more than one resources with the smallest / biggest resource standard rate, the overtime rate is compared. The last comparison stage regards the number of skill types that are owned by resource. Finally, if there is still more than one possibility of choosing resource, the first one from the list (regarding the order from creating the project instance) is taken. The whole RP procedure is shown as a pseudo-code in the Algorithm 2.

Algorithm 1 SA(P_π, P_d, P_s, P_a) priority rule

Require: Defined tasks (T), resources (R), relations and skills
Ensure: Feasible schedule (set of task–to–resource assignments (A)

$Q \leftarrow sortByAdjustmentMeasure(P_\pi)$
for $q \in Q$ **do**
  $T_q \leftarrow tasksWithExpectedSkill(q)$
  $T_q \leftarrow sortByDuration(P_d)$
  for t to T **do**
    for r to R **do**
      if $resourceCanDoTask(r, t)$ **then**
        $R' \leftarrow add(r)$
    $R' \leftarrow sortByStandardRateSalary(P_s)$
    $R' \leftarrow getCheapestResources()$
    $r' \leftarrow getFirst(R')$
    $a_i \leftarrow t(r')$
    $A \leftarrow add(a_i)$
  if $dynamicAdjustment(P_a)$ **then**
    $\pi \leftarrow adjustment()$
    $Q \leftarrow sortByAdjustmentMeasure(\pi)$

In analogy to SA, some parameters have to be included, to determine the sorting order of investigated elements. Hence P_{sl} concerns ordering skills by their level, P_t regards the order of resources by their free time in the project, P_{sr} sorts the resource in the order of their standard rate, P_{or} - the same like previous but regards overtime rate and P_{st} defines the sorting order for number of skills owned by resource.

Algorithm 2 RP($P_{sl}, P_t, P_{sr}, P_{or}, P_{st}$) priority rule

Require: Defined tasks (T), resources (R), relations and skills
Ensure: Feasible schedule (set of task–to–resource assignments (A)

for t to T **do**
  $skill \leftarrow getSkillName(t)$
  for r to R **do**
    if $resourceCanDoTask(r, t)$ **then**
      $R' \leftarrow add(r)$
  $R_{HSL} \leftarrow resourcesOrderedBySkillLevel(skill, P_{sl})$
  if $size(R_{HSL}) > 1$ **then**
    $R_{MFT} \leftarrow resourcesOrderedByFreeTime(P_t)$
    if $size(R_{MFT}) > 1$ **then**
      $R_{MSR} \leftarrow resourcesOrderedByStandardRate(P_{sr})$
      if $size(R_{MSR}) > 1$ **then**
        $R_{MOR} \leftarrow resourcesOrderedByOvertimeRate(P_{or})$
        if $size(R_{MOR}) > 1$ **then**
          $R_{MST} \leftarrow resourcesOrderedBySkillTypes(P_{st})$
          if $size(R_{MST}) > 1$ **then**
            $a_i \leftarrow getFirst(R_{MST})$
          else
            $a_i \leftarrow R_{MST}$
            ...
  $A \leftarrow add(a_i)$

Similarly to SA priority rule, the set of values for parameters in RP priority rule is the same: A – ascending, D – descending.

IV. Experiments and results

The goal of conducted experiments was to investigate whether proposed SPR stand a robust way of solving MS–RCPSP and thus – whether they can be used in combination with EA. To evaluate solution – the resulted project schedule – its duration time ([days]) and performance cost ([c.u][1]) were investigated.

A. Dataset

Because not only the project schedule duration, but also the cost of the schedule should be evaluated, we cannot use the standard PSPLIB benchmark dataset [9], that does not contain any information about the task performance cost. What is more, PSPLIB dataset instances do not reflect the multi–skill model. We propose the dataset containing six project instances that has been artificially created[2], in a base of real–world instances, got from the Volvo IT Department in Wroclaw. The dataset instances have been verified by experienced project manager from mentioned enterprise.

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. Because of the different resources and relations number, the scheduling complexity for each project was varied.

B. Set-up

For SA priority rule 4 parameters has been investigated – $2^4 = 16$ experiments with different parameters' configurations have been performed (see Tab. III). In RP summary table (see Tab. IV) records containing the same values for each project instance with different parameters' configuration have been filtered out. The number of experiments is equal to the number of parameters' configuration: $2^5 = 32$.

C. Experiments

Experiments have been divided into the following parts. Evaluation of project duration and performance cost (see Table II) for first three priority rules. Furthermore, an evaluation of the same project properties (duration and cost) has been performed for SA (Tab. III) and RP heuristics (Tab. IV).

[1]Currency units

[2]http://www.ii.pwr.wroc.pl/˜myszkowski/scheduling

All experiments have been performed on a personal computer equipped with Intel Core 2 Duo P8700 (2.53 GHz at each core), 4 GB memory RAM and Windows 7 Professional with Service Pack 1. For such configuration, SPR processing times were negligible small (1-3 [s]), thus they were not presented in details in this paper. The experimental environment was provided by the Eclipse IDE and Java programming language. We used some specific libraries[3] to process MS Project files.

We decided to highlight the best obtained results in specified tables (see Tab. II, III, IV). However, as described problem is multi-objective, we highlighted the best duration– and cost oriented optimization results for given project instance. If there were more than one best result for specified objective, that with smaller value of second objective was highlighted as the best one.

D. Results obtained for TD, SLS, RS

The experimental results for TD, SLS and RS has been presented in Table II. Taking into account the consideration of obtained results for evaluation of project duration for TD, SLS and RS, we can see that that the SLS became *a winner* in 5/6 times. Both ascending and descending mode for SLS priority rule got the best results four times. RS priority rule has never obtained the best result, what can be explained by the fact that this rule does not *care about* the duration of the project. It is focused only on the cost aspect.

As RS priority rule became the worst approach for scheduling focused on the duration aspect, it turned out to be the best way when the cost–oriented scheduling mode is required. When the ascending mode for RS priority rule was set, the best results for each project instance was obtained (6/6). However, changing the order of this priority rule from ascending to descending make it the less effective approach. It suggests that this is one of the most performance cost–sensitive SPR.

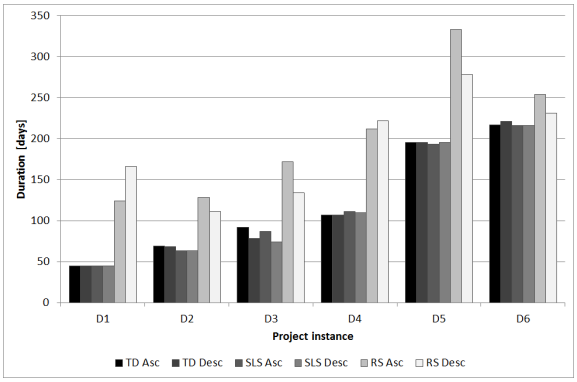

Fig. 1. Project schedule duration for TD, SLS and RS priority rules

Project duration and performance cost results for TD, SLS and RS priority rules have been also illustrated in the Fig. 1 and Fig. 2 respectively. Indicated figures gives as a clue how changing the number of resources in project could influence on the potential of scheduling optimization. The more tasks

[3]Microsoft Project Exchange - http://mpxj.sourceforge.net

TABLE II
PROJECT DURATION AND PERFORMANCE COST OBTAINED FOR TD, RS, SLS PRIORITY RULES

Method		Dataset instance											
		D1		D2		D3		D4		D5		D6	
		days	cost	days	cost	days	cost	days	cost	days	cost	days	cost
TD	Asc	45	52806	69	43262	92	40677	107	103102	195	93285	217	105686
	Desc	45	52607	68	43892	78	40862	**107**	**101319**	195	93535	221	105205
SLS	Asc	**45**	**47530**	63	43329	87	40346	111	73157	**193**	**59627**	**216**	**75033**
	Desc	45	48257	**63**	**43221**	**74**	**40286**	110	73749	195	59339	216	75141
RS	Asc	**124**	**30104**	**128**	**26323**	**172**	**30164**	**212**	**46133**	**333**	**51496**	**254**	**71986**
	Desc	166	83312	111	60384	134	52957	222	148407	278	142072	231	131272
	Avg	78	52436	84	43402	106	40882	145	90978	232	83226	226	94054

are statistically assigned to resource, the longer the project schedule could be.

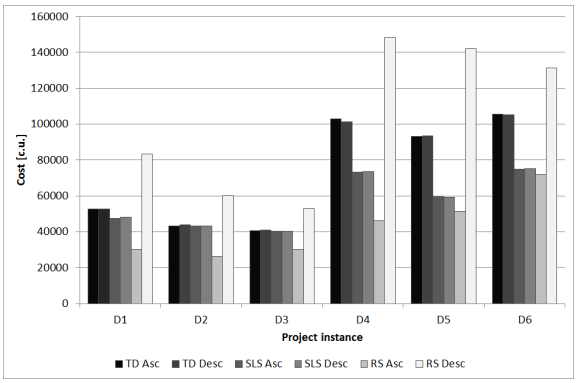

Fig. 2. Project schedule cost for TD, SLS and RS priority rules

E. Results obtained for SA

However duration results obtained for the dataset using SA heuristic are much diversified (see Tab. III), an interesting observation has been made. For 4 from 6 project instances, the best solution was found where the second parameter value, which was reflected to task duration sorting order, had been set to D. In other words - sorting tasks by their descending duration provides the best duration results. What is also interesting, for two remaining project instances the best duration results were obtained when task duration sorting order had been set to ascending, but the skill adjustment order had been set to descending.

The cost–oriented scheduling (see Tab. III) with the use of SA priority rule leads to the conclusion, that cost–related parameter (resource standard rate) is the most important. Moreover, results show that this is the only one parameter that influences to the method's robustness. If cost sorting order is set to ascending, provided results as the best, while changing the parameter to the descending value makes often results even the worst from all of investigated methods. What is also interesting, for all of performed experiments related to evaluating the cost for SA priority rule, only two resulting values were obtained.

What is also worth mentioning, duration optimization results for SA priority rule were significantly worse in comparison to those obtained for duration optimization in TD, SLS, RS and RP priority rules. While the average duration for sample project instance (D1) is equal to 78 for TD, SLS, RS (see Tab.II) or 70 (see Tab.IV) for RP priority rules, the average duration value for the same project instance in SA (see Tab.III) is about two times bigger - 149 days! On the other hand, cost aspect for D1 has been changed from average 52436 [c.u.] (see Tab.II to average 56708 [c.u.] (see Tab.III).

F. Results obtained for RP

As it is presented in the Table IV, ascending skill level criterion generally leads to get better results. Most often the best results has been placed into the parameters' configuration pool consisting skill level criterion ascending and resource availability criterion descending. It could lead to the conclusion that those two parameters influence the heuristic optimization potential the most. However, for the D6 project schedule instance, the best configuration regards ascending skill level criterion and ascending resource availability. On the other hand, for two from six project instances the best duration optimization results were obtained also for descending skill level criterion, while the resource availability criterion still remains descending.

Looking at the cost optimization results (see Tab. IV) the first impression can be made that those results are in opposition to project duration results. It proves the previous assumption that project cost and duration are in opposition. It means: reducing the performance cost leads to enlarging the project duration. The best cost optimization results were obtained when the second parameter – resource availability – has been set to ascending. An interesting fact is that steering remaining parameters does not influence on the final result. However it involves only the parameter configuration, when the skill level criterion is set descending.

Nevertheless the average values for cost optimization in SA is generally smaller than those relevant values for SA, the better optimization results were obtained for SA. The differences in average values come from smaller standard deviation of performance cost values in parameter configurations of RP. Cost obtained for RP are relatively small, but the smallest values were obtained in SA. On the other hand, invalid parameter configuration in SA could lead to increase the performance cost drastically.

TABLE III
PROJECT DURATION AND PERFORMANCE COST FOR SA PRIORITY RULE

Parameter configuration	Dataset instance											
	D1		D2		D3		D4		D5		D6	
	days	cost	days	cost	days	cost	days	cost	days	cost	days	cost
AAAA	**122**	**30104**	169	26322	189	30163	**213**	**46132**	356	51495	294	71986
AAAD	128	30104	135	26322	179	30163	237	46132	347	51495	311	71986
AADA	166	83311	126	60383	151	52957	258	148407	294	142071	258	131272
AADD	177	83311	119	60383	144	52957	234	148407	313	142071	281	131272
ADAA	**121**	**30104**	144	26322	**171**	**30163**	225	46132	349	51495	294	71986
ADAD	135	30104	140	26322	178	30163	237	46132	**329**	**51495**	292	71986
ADDA	162	83311	120	60383	138	52957	248	148407	**275**	**142071**	244	131272
ADDD	176	83311	127	60383	**135**	**52957**	245	148407	335	142071	273	131272
DAAA	127	30104	144	26322	174	30163	236	46132	342	51495	288	71986
DAAD	127	30104	144	26322	176	30163	**212**	**46132**	387	51495	292	71986
DADA	176	83311	111	60383	142	52957	241	148407	322	142071	**239**	**131272**
DADD	178	83311	111	60383	142	52957	242	148407	324	142071	247	131272
DDAA	**122**	**30104**	**131**	**26322**	177	30163	229	46132	349	51495	**259**	**71986**
DDAD	**122**	**30104**	**131**	**26322**	195	30163	229	46132	356	51495	264	71986
DDDA	166	83311	**105**	**60383**	**135**	**52957**	248	148407	309	142071	259	131272
DDDD	176	83311	**105**	**60383**	**135**	**52957**	224	148407	305	142071	259	131272
Average	149	56708	129	43353	160	41560	235	97270	331	96783	272	101629

TABLE IV
PROJECT DURATION AND PERFORMANCE COST FOR RP PRIORITY RULE

Parameter configuration	Dataset instance											
	D1		D2		D3		D4		D5		D6	
	days	cost	days	cost	days	cost	days	cost	days	cost	days	cost
AAAAA	85	37657	100	37843	118	42512	200	51905	223	58549	**216**	**97984**
AAADA	**85**	**35951**	**100**	**36957**	118	42512	201	50074	223	58447	**216**	**97984**
ADAAA	**45**	**52693**	**64**	**43094**	**108**	**39260**	110	100583	**193**	**94386**	217	104757
ADADA	**45**	**52693**	**63**	**42948**	**108**	**39260**	**108**	**101366**	199	93582	217	104757
DAAAA	100	38776	140	41511	**174**	**36299**	**285**	**49261**	**247**	**55735**	**244**	**83948**
DDAAA	50	54341	66	42576	108	42100	112	104275	195	88679	216	102737
DDADA	50	54341	66	42576	108	42100	112	103216	195	88679	216	102737
Average	70	45653	93	41126	127	40042	177	76242	215	74224	223	97357

V. RESULTS' DISCUSSION

The best obtained results for the duration and cost optimization has been compiled into the Table V. The table shows that SLS priority rule became the most effective in duration optimization for the most analysed project instances. Only for D4 instance the other method turned out to be more robust (TD). While first of proposed complex heuristic (SA) became not sufficient for duration optimization, the RP heuristic resulted well, being mostly equally efficient as SLS.

TABLE V
THE BEST OBTAINED DURATION AND COST INDICATORS.

ID	DO			CO		
	Method	D	C	Method	D	C
D1	SLS-A	45	47530	SA-ADAA	121	30140
D2	SLS-D	63	43221	SA-DDAA	131	26322
D3	SLS-D	74	40286	SA-DAAA	174	30163
D4	TD-D	107	101319	SA-DAAD	212	46132
D5	SLS-A	193	59627	SA-ADAD	329	51495
D6	SLS-A	216	75033	RS-A	254	51986

Beside the duration optimization, the project performance cost optimization has been also examined for each project instance. The cost optimization results are also presented in the Table V. Obtained results are quite interesting: SA methods became the best cost–optimization–oriented. An interesting fact is that for each best SA parameter configuration, the third criterion that reflects resource standard rate, was always set to ascending (A). Results given for RS were comparably good as those obtained by SA (see Tab. II), thus both SA and RS method became the best cost–optimization method for each project instance.

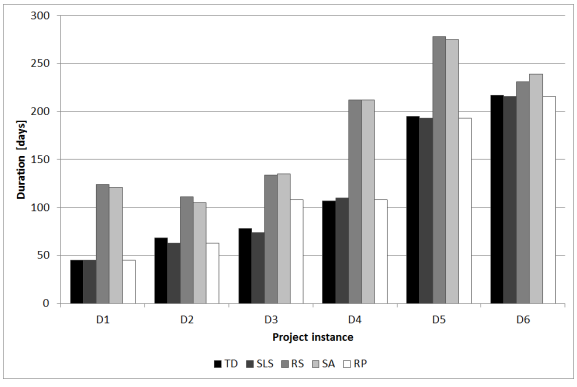

Fig. 3. Summary duration optimization results of examined methods.

The summary of duration and cost optimization for the best parameter configuration has been presented in Fig. 3 (duration optimization) and in Fig. 4 (cost optimization). In these figures

the issue of time/cost trade-off is clearly presented – reducing the project cost makes the performance cost larger and vice versa.

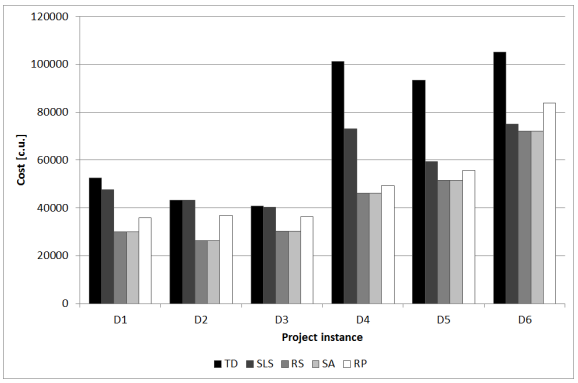

Fig. 4. Summary cost optimization results of examined methods.

Let's analyse the differences between the best duration and cost optimization. While the best project duration for D1 is 45 days, such a schedule would cost 47530 [c.u.], the cheapest project schedule could save about 17000 [c.u.] (37%), but would enlarge the project for 169%. For D2 project instance those indicators are 39% of reducing cost and enlarging the project duration by 108%. For D3 project instance 135% of project duration enlargement could cause saving 25% of budget. An interesting example is D4, where reducing the budget by the half, enlarges the duration about two times. D5 project instance turned out to be the least prone for cost optimization, where only 14% of budget could be saved, with the respect of enlarging the project duration by 70%. The last project instance – D6 – has the smallest potential in duration optimization - only 18% of duration was able to be reduced. Consequently, the potential of performance cost optimization for this instance is also reduced.

VI. Conclusions and Further Work

Before making some more general assumptions and conclusions, an important issue has to be reminded. The best optimization results, both time– or cost– oriented influence on the opposite project property. In other words, best time optimization could cause generate the schedule with unacceptable performance cost. Analogously, reducing the cost could be the reason of enormous enlargement of project duration. As it was presented in tables with detailed results.

The main drawback that can be stated for proposed SPR is that they are much less flexible in obtaining the final solution than other methods, like metaheuristics, especially those population–based. E.g. for EA the set of possible solutions is investigated during the EA *runtime*. Thus they can be compared and the *best compromise* can be found - the solution that is cheaper enough, but with acceptable cost. For SPR we do not have the possibility to find such a medium solution.

On the other hand the computing time of above mentioned methods is negligibly small, using such methods in EA could enlarge its processing times. The decision would have to be made, whether potential increase of robustness and flexibility is worth of enlarging processing time.

Obtained results provide also a conclusion that there is no need to make the priority rules too much complicated. TD, SLS and RS are operating in one or maximum two sorted lists (tasks and / or resources), while in the SA one more sorted list has to be made, based on the skills pool. It enlarges the computing complexity, but does not provides better results.

Furthermore, if cost–oriented scheduling is taken into account, the best results have been obtained using the SA heuristic. However, the results are the same like obtained for RS priority rule. It could lead to conclusion, that also cost–oriented optimization could be made with the usage of simpler priority rules.

Performed experiments showed that using SPR give possibility to get sub-optimal solution of proposed problem. Hence, in a further work proposed methods would be applied to EA as a local search (mutation operator) or as an initial population generator method.

As the fitness function in EA could be weighted by their duration and cost component, specified heuristic could be chosen depending to the weight settings of evaluation function in EA. Directed selection of local search or initial population creation method could enhance the final optimization results.

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] Boctor F. F.; Heuristics for scheduling projects with resource restrictions and several resource–duration modes, International Journal of Production Research (31/11), pp. 2547–2558, 1993.
[3] Browning T. R., Yassine A. A.; Resource–constrained multi–project scheduling: Priority rule performance revisited, International Journal of Production and Economics (126), pp. 212–228, 2010.
[4] 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.
[5] Buddhakulsomsiri J., Kim D., S.; Priority rule–based heuristic for multi-mode resource–constrained project scheduling problems with resource vacations and activity splitting, European Journal of Operational Research (178), pp. 374–390, 2007.
[6] 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.
[7] Hartmann S.; A competitive genetic algorithm for resource–constrained project scheduling, Naval Research Logistics (45), pp. 733–750, 1998.
[8] 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.
[9] Kolisch R., Sprecher A., PSPLIB - A project scheduling problem library, European Journal of Operational Research (96), pp. 205–216, 1996.
[10] Kolisch R.; Efficient priority rules for the resource–constrained project scheduling problem, Journal of Operations Management (14), pp. 179–192, 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] 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.

[14] Mendes J. J. M., Goncalves 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.

[15] Lova A., Tormos P., Barber F., Multi-Mode Resource Constrained Project Scheduling: Scheduling Schemes, Priority Rules and Mode Selection Rules, Inteligencia Artificial, Revista Iberoamericana de Inteligencia Artificial, (10), pp. 69–86, 2006.

[16] 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.

[17] 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.