DOI: 10.15439/2015F273
ACSIS, Vol. 5

A new benchmark dataset for Multi-Skill Resource-Constrained Project Scheduling Problem

Paweł B. Myszkowski, Marek E. Skowroński, Krzysztof Sikora
Institute of Informatics, Department of Artificial Intelligence
Faculty of Computer Science & Management, Wrocław University of Technology, Poland
Email: {pawel.myszkowski, m.e.skowronski, krzysztof.sikora}@pwr.edu.pl

***Abstract*—In this paper novel project scheduling difficulty estimations are proposed for Multi-Skill Resource-Constrained Project Scheduling Problem (MS–RCPSP). The main goal of introducing the complexity estimations is an attempt of estimation the project complexity before launching the optimization process. What is more, the dataset instance generator is also presented as a tool to create new instances for extending the research area. Furthermore, the dataset proposed in previous works is extended by new instances, described thoroughly and released as a benchmark dataset. The dataset instances are also scheduled using simple heuristic and greedy algorithm in duration- and cost- oriented optimization modes. Finally, a brief summary of investigated methods and potential further research directions is presented.**

***Index Terms*—scheduling, RCPSP, dataset, benchmark, heuristics, indicator**

I. Introduction

Resource-Constrained Project Scheduling Problem stands as one of the most important and the most investigated [9], [10] kind of known types of scheduling problems. It is because of its practical nature and the need to find good ways for resolving it not only for scientific, but also industry purposes. Its goal is to find the best schedule for the project, by assigning scarce resource to defined tasks. The quality of the schedule is mostly defined as its duration, cost or some combinations of those indicators.

As MS–RCPSP is the extension of classical RCPSP, it makes the problem NP–hard [2]. Hence, there is no way to find a method that would be able to find the optimal solution in polynomial, reasonable time. Therefore, one of the main approach to solve RCPSP and its potential extensions is to use soft computing methods, especially metaheuristics [19].

To make the problem definition more practical in industrial point of view, we introduced the skill domain. Tasks require some specified skill to be performed by resources owning some subset of skills defined in the project. Therefore, not every resource is able to perform every task in the project. It makes the problem more constrained but on the other hand – more realistic. RCPSP extended by skills domain is called Multi–Skill RCPSP (MS–RCPSP).

The goal of the paper is to present several indicators for the difficulty of the project to be scheduled. The difficulty could be understood as a measure how much the solution space is constrained – how hard is to build feasible and good enough schedule. The secondary objective is to share the dataset and propose it as a benchmark for other researchers, to build a common platform for evaluating methods solving MS–RCPSP.

The rest of the paper is organised as follows. Section II presents some approaches in solving MS–RCPSP using some of the mentioned metaheuristics. Then section III describes the problem statement. Then Section IV presents complexity estimations we proposed for MS–RCPSP. Section V describes the way how new instances are generated. Furthermore in the Section VI the dataset has been presented and then its instances have been used in experiments in Sec. VII while the last Section VIII describes approaches we have recently investigated and proposes ways for further research.

II. Related work

NP–hard [2], combinatorial nature of MS–RCPSP is one of the reasons of common use of metaheuristic–based approaches in solving the problem. Nevertheless, some constraint programming methods or simpler heuristics are also used to solve this kind of problems [20].

However, there is still lack of papers regarding multi–objective Multi–Skill extension of RCPSP. Some approaches solving MS–RCPSP in project duration domain [1], [16] or project cost domain [12] could be found. On the other hand, there are methods solving classical RCPSP extended by cost domain, but without skills considerations. Such research has been presented in [15], [6], [4], [13] and [24]. Hence, we have decided to combine those two elements: multi–objective optimization and multi–skill domain for project scheduling problem.

Although classical RCPSP is deeply investigated and numerous approaches could be easily compared using PSPLIB instances, it is very hard to find multi–objective MS–RCPSP methods working on datasets that could be regarded as a benchmark. Some papers describe instances artificially generated ([5], [16]), while some others propose methods of PSPLIB dataset adaptation ([1], [3], [7], [12]). We analysed some published benchmark datasets, but they were usually unsuitable for our approach as they do not cover multi-objective nature of the problem, even multi–skill domain has been developed ([23]). The other unsuitable example is a benchmark for Multi–Mode RCPSP (MM-RCPSP) published in [21], but it does not involve skills constraints and make the main focus on multi–mode characteristics. Hence, the need of definition new dataset has arisen.

Some difficulty estimations for any project scheduling problems could be found in [11] or [13]. However those proposed difficulty estimations based mostly on tasks, precedence relations between them and resource properties. There is a lack of difficulty estimations that would be dedicated for MS–RCPSP, involving skills domain.

Among many papers regarding the resource – constrained project scheduling problems and its extensions, we found that we had something in common with the approach presented in [13]. Despite some similarities, there are some crucial elements that make our approach, defined in detail in [14], different. We regard both of them as worth of investigating. Tab. I presents the comparison – similarities and differences in four main areas we decided to point out.

Based on the information in the Tab. I, some common elements between our approach and the one presented in [13] in all of the mentioned areas can be found. Firstly, investigated problems are similar in a way that both of them regard multi–objective optimization in Multi–Skill Resource–Constrained Project Scheduling Problem (MS–RCPSP). Both problems are additionally defined by some resource, precedence and skills constraints. However those constraints differ in details (i.e. distinguishing skill types).

There are also some similarities regarding the dataset published. First of all, both of the datasets are published in the Internet, so anyone has an access to dataset instances and can use them to investigate own optimization methods in MS–RCPSP. What is more, the number of dataset instances is the same. However, the strategy of building the dataset was different. We used information about number of different skill types and number of precedence relations. Not only the most common indicators, like number of tasks or number of resources, what can be found in [13]. Based on those additional indicators we tried to build as most balanced dataset as possible. Therefore, we tried to adjust the number of resources, skills and precedence relations in a way to make our complexity indicators similar for 100 and 200-tasks project instances as well.

III. Problem statement

The goal of the MS–RCPSP is to order given set of tasks and assign resources to them in a way to provide feasible and as good solution as possible. The quality of the solution could be measured in its duration, cost or any other measure defined according to business requirements.

In MS–RCPSP the set of tasks (J) is given, while every task has to be performed during the project execution. Each task is described by its start (S_j) and finish dates (F_j), duration (d_j) and skill required by it to be performed. What is more, tasks can be related between themselves by precedence relations. It means some tasks (successors) cannot start before some other would be finished (predecessors). It makes the solution space for given instance more constrained, as there are fewer possibilities to put the tasks in given period. Predecessors of given task j are defined as P_j while overall number of predecessors in a project is p.

Furthermore, the set of resources (K) is also given. Every resource k is described by its salary (s_k) and skills covered (Q^k). Therefore, a subset of tasks than can be performed by k resource could be obtained and is denoted as J^k.

Skill required by task to be performed determines which resource can be assigned to it. Every skill type could appear in a project in various familiarity levels, denoted as an integer value from 0 to 4. Resources with skill type required by given task but on the lower than required level cannot be assigned to such task. The number of all skills (including different familiarity levels) is denoted as q, while the number of skill types in the project is denoted as $\bar{q}$

What is more, any resource cannot be assigned to more than one task in the overlapping period. If such a situation occurs, **conflict** is detected and has to be resolved, to get valid, feasible solution. It is made by shifting some of conflicting tasks in time–line in a way to make it start just after another conflicted task would be finished. The decision which of conflicted tasks should be taken to be shifted is made by checking which has been previously added to the project definition, because we do not distinguish various levels of task priority. Each task is equally important to be scheduled in the project.

Any project schedule (PS) has to be conflicts–free and has to satisfy the precedence constraints between tasks. If it satisfied those both kind of constraints, we would call it as a **feasible** schedule. Only feasible ones can be regarded as finite solutions. What is more, every infeasible solution could be made feasible. However, making schedule feasible could make its duration longer.

A. Calendar restrictions

Due to use Microsoft Project as a base for our dataset, we needed to obtain some calendar restrictions that are strictly related to used software.

First of all, standard calendar in Microsoft Project is designed to handle projects in real–life, where classical five–days week of work is used. It was also a requirement asked by the VolvoIT enterprise that we cooperate in the research field of project scheduling. However, weekends are taken into account. If resource is assigned to task that cannot be finished before weekend, it will be finished after the weekend. The task duration is bigger but number of man–hours (or man–days) required for given task does not change. Therefore various project duration measures can be obtained. One can be made based on the overall duration of project – between its start and finish dates, including weekends where tasks are not performed but influence on other tasks' start dates and the project finish. Other approach could be to ignore any festivals and weekends and regard seven days week of work. Until now we prefer the first approach as it is more practical.

Furthermore, the localization issue has to be taken into account when considering calendar restrictions. Depending on the localization settings, some changes in the calendar could appear, regarding some national or cultural–related festivals. In example, there would be other free days in China than in Poland, where different festivals are taken place.

TABLE I
SIMILARITIES AND DIFFERENCES BETWEEN IMOPSE [14] APPROACH AND THE APPROACH PRESENTED IN [13]

Area	Similarities	Differences
Problem definition	Multi-skill Multi-objective Resource-constrained Precedence relations Skills Minimal one resource required by given task to be performed Repair operator → enlarging project duration	Resource load - 'dedication' measure No skill levels Task can be assigned to more than one resource No conflict, different rule of repair operator Approach more academic than practical
Dataset	Same number of instances: 36 Published in the Internet	Instances distinguished only by number of tasks and number of resources Constant vs. varied number of skill types in a dataset instance
Generator	Published in the Internet as a benchmark Developed in the JAVA programming language	No graphical user interface Output format not connected with MS Project
Methodology	Time vs. cost tradeoff Focus on multiobjective, Pareto–based optimization	Complexity estimators vs. hypervolume, attainment

Linking above constraints with potential dynamic date of project start – the date, when the first task is assigned to resource in the timeline – there is a risk that the same project with the same task–to–resource assignments (schedule) can be finished in various dates, depending on the day of start. Project instances start at various dates, except the ones with D* suffix that have been prepared strictly for given enterprise and were required to be start all at the same day. It has been set to 12th April of 2012.

To avoid those calendar restrictions .def format has been introduced. It is described in detail in Subsec. VI-A.

B. Evaluation function

The goal of MS–RCPSP is to find the best (as quick or / and as cheap as possible) final project schedule. Hence, we could present it as bi–objective optimization problem. Because of totally different domains of duration and cost, we cannot simply aggregate those two objectives. Therefore, the normalization process is performed, to get the value scope between 0 and 1. It allows us to aggregate those two objectives and combine them into one evaluation function.

We have also preserved the possibility to choose which objective is more important in given optimization process. It is made by setting weights both for the duration (ω_τ) and cost aspect. The sum of both weights sum to 1 and the scope of values is from 0 to 1. It means that setting the weight of duration aspect to one automatically sets the weight of cost to 0 and vice versa. Naturally that weight can be set by float value. Specifically, both weights could be set to 0.5. In that case, both objectives would be equally important in the optimization process. We proposed three baseline weight configurations: duration optimization (DO, $\omega_\tau = 1$), balanced optimization (BO, $\omega_\tau = 0.5$) and cost optimization (CO, $\omega_\tau = 0$) [14].

An important remark is that those objectives are in opposition to each other. It means that setting weights to make the optimization process more cost–oriented could cause getting cheaper project schedule, but with the risk that final schedule would be longer. Analogously, shorter project schedule could be obtained with spending more money on it.

Evaluation function is formulated as follows:

$$\min f(PS) = \omega_\tau f_\tau(PS) + (1 - \omega_\tau) f_c(PS) \tag{1}$$

where: w_τ – weight of duration component, $f_\tau(PS)$ – duration evaluation component, $f_c(PS)$ – cost evaluation component. Both components are non–negative values, while $w_\tau \in [0; 1]$.

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

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

Where: τ_{max} – maximal (pessimistic) possible duration of the schedule PS, computed as the sum of all tasks' duration [14]. It occurs when all tasks are performed serially in project: one–by–one. No matter, how many and how flexible resources are.

The cost component $f_c(PS)$ is defined as follows:

$$f_c(PS) = \frac{\sum_{i=1}^{J} c_j - c_{min}}{c_{max} - c_{min}} \tag{3}$$

where: 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 [14]. Note: c_{max} and c_{min} do not involve skill constraints. It means that c_{min} value could be reached also 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 (SS), as:

$$SS(n, m) = n! * m^n \tag{4}$$

Where n is a number of tasks and m is a number of resources [14]. However, that estimation also takes into account non–feasible solutions, because skill–constraints are not satisfied. To give an example, let's assume $n = 10$ and $m = 5$ – 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 the 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, above example is a simplification and in real world problems we meet a higher number tasks (about $n = 100$) and resources ($m = 20$) – it gives $SS(100, 20) = 1.19 * 10^{288}$ of all solutions.

IV. Complexity estimations

As a result of cooperation with VolvoIT Department in Wroclaw [18], [20], [19], we defined following elements [14]:

- requirements and constraints dedicated to the industry,
- project scheduling difficulty indicators.

Project difficulty indicators have been verified and approved by experienced project manager in the enterprise.

The main goal of investigating such estimations was to compare how the project elements (tasks, resources, precedence relations, skills) characteristics could influence on the optimization process based on the quality of obtained result (project schedule duration or performance cost) or optimization processing time.

Proposed difficulty estimations are described below. All estimations are normalised before being taken to compute the overall complexity measure.

A. affiliation (λ)

States, how much tasks are related between them. The bigger value means the tasks are more related. The project complexity is bigger, because the scheduling flexibility is restricted (more tasks are related to others, so they cannot be scheduled flexibly). It is computed as follows:

$$\lambda = \frac{p}{n} \tag{5}$$

Where p – number of precedence relations, n – number of tasks.

B. load (ν)

Reflects, how much resources are loaded by tasks. The bigger value means, the more tasks are assigned to one resource (the project complexity is bigger, because the solution space is bigger). It is computed as follows:

$$\nu = \frac{n}{m} \tag{6}$$

Where m – number of resources.

C. time difference (Φ_T)

Describes how tasks are varied by their duration. The bigger value means tasks are more varied. That makes scheduling more difficult, because tasks' order influences on overall duration time. It is computed as follows:

$$\Phi_T = \frac{\sigma_d}{d_{max} - d_{min}} \tag{7}$$

Where: σ_d – standard deviation of tasks' duration in schedule, d_{max} – maximal task duration in schedule, d_{min} – minimal task duration in schedule.

D. cost difference (Φ_C)

Indicates how tasks are varied by their performance cost. The interpretation is similar to the time difference (Φ_T). It is computed as follows:

$$\Phi_C = \frac{\sigma_C}{c_{max} - c_{min}} \tag{8}$$

Where: σ_C – standard deviation of tasks' cost in schedule, c_{max} – maximal task cost in schedule, c_{min} – minimal task cost in schedule.

E. variety (μ)

Reflects how resources are varied by their skills. The bigger value means the project is more difficult to be scheduled because tasks are more dedicated to resources (no other can be assigned to the specified task). It is computed as follows:

$$\mu = \frac{q}{m} \tag{9}$$

Where: q – number of different skills existing in the project. Important: each level of the same skill name is regarded as a new skill.

F. universality (β)

States the average number of resource skills. The bigger value means it is easier to schedule a project because resources are more universal. It is computed as follows:

$$\beta = \frac{\sum_{i=1}^{m} Q^i}{m} \tag{10}$$

Where: Q^i – number of skills owned by i resource.

G. adjustment (π)

Shows how many resources available to be assigned in the project are capable of performing tasks that are needed to be performed. Ergo – how many resources can deal with each task. The bigger value means it is more difficult to schedule a project because resources are strictly adjusted to the tasks by their skills covered, and skills needed. It is computed as follows:

$$\pi = \frac{\sum_{i=0}^{Q} \Delta(q_i) * \sigma(\Delta(q))}{max(\Delta(q_1), \Delta(q_2), ..., \Delta(q_q)) * q} \tag{11}$$

Where:

$$\Delta(q_i) = \frac{|Q^i - TQ(i)|}{min(RQ(i), TQ(i))} \tag{12}$$

Where: $RQ(i)$ – number of resources covering skill i (normalized by number of all resources (m) in the project). $TQ(i)$ – number of tasks, that require skill i to be performed (normalized by number of all tasks (n) in the project).

H. Flexibility (θ)

The flexibility θ of the instance PS has been estimated as the sum of a number of potential assignments of tasks to a given resource divided by number of resources (n). It can be stated as follows:

$$\theta = \frac{\sum_{k=1}^{n} \bar{J}^k}{m} \tag{13}$$

Where $\bar{J}^k$ is the number of tasks that can be performed by resource k, while m is the number of resources in a project.

Having discussed the usage of those estimations' legitimacy, each measure has been subjectively weighted and confirmed by an experienced project manager (to determine their priority in overall project's difficulty measure). Having those weights set up, the project's (*PF*) difficulty measure function could be defined as follows:

$$diff(PF) = 8\lambda(PF) + 9\nu(PF) + 3\Phi_\tau(PF) + \\ +3\Phi_C(PF) + 6\mu(PF) - 4\beta(PF) + 7\pi(PF) + 5\theta(PF) \tag{14}$$

The bigger the value $diff(PF)$ is, the more difficult to schedule the project is. Universality measure has been taken with a negative value. It is because the bigger the universality value is, the project is easier to schedule as resources are more skill–flexible and can be assigned to more different tasks, relaxing more skill constraints.

Depending on project manager preferences, weights assigned to given estimations could be changed, what would influence on the overall $diff(PF)$ measure.

V. Instances generator

The main goal of implementing the dataset instance generator is to provide other researchers the possibility to investigate their methods not only on proposed dataset instances, but also on some other that would be created individually by given researcher. Dataset instance generator has been prepared for MS–RCPSP but it can be easily adjusted to handle classical RCPSP instances like PSPLIB [8]. It has been implemented in JAVA programming language, using MPXJ[1] library for processing project files from MS Project. It can create project definition not only in .mpp (XML) format, but also the simpler (.def) one. The more detailed description of .def format is available in Subsec. VI-A. Instances have been created based on the real–life project instances got from international entreprise (Volvo IT).

Instances generator is an element of resources developed in our **Intelligent Multi–Objective Project Scheduling Environment**[2] platform. Besides instances generator, the platform contains solution validator (see Subsec. VI-B), instances we generated and used to verify our approaches and the best found solutions for those instances in three above–mentioned optimization modes: DO, BO, CO. Every solution is saved in ready–to–use in MS Project .xml format, containing all tasks, resources, skills, precedence relations and obtained schedule.

The general process of generation new instances could be split into main steps:

1) Read and validate parameter values provided by the end user
2) Define resources,
3) Define skills and assign them to resources,
4) Define tasks and precedence relations,
5) Assign resources if necessary,
6) Save project.

In the further parts of this section, following steps would be described in detail. The pseudocode of the generator has been presented in Alg. 1

Algorithm 1 Generator pseudocode

```
1: $pool \leftarrow \emptyset$
2: #generate_resources
3: for $r \in K$ do
4:     #generate_resource
5:     $set\_standard\_salary(minSt, maxSt)$
6:     #set_skills($r_i$)
7:     $q \leftarrow setNumSkills(min, max)$
8:     for $j = 0; j < q$ do
9:         $set\_skill\_type\_from\_range(minST, maxST, q_j)$
10:        $set\_skill\_level\_from\_range(minSL, maxSL, q_j)$
11:        if $skill\_not\_exists(q_j, pool)$ then
12:            $pool \leftarrow pool.add(q_j)$
13:        $r \leftarrow addSkill(q_j)$
14: #generate_tasks
15: for $t \in J$ do
16:     $t \leftarrow set\_duration(minDuration, maxDuration)$
17:     $t \leftarrow set\_skill(pool)$
18: #generate_relations
19: for $i \in P$ do
20:     $relSource \leftarrow rand(T)$
21:     $relDest \leftarrow rand(relSource, T, min, max)$
22:     $relSource \leftarrow addPredecessor(relSource)$
23: if $make\_assignments$ then
24:     #assign_resources [initial schedule builder]
25:     for $n \in J$ do
26:         $R \leftarrow capable\_resources(n)$
27:         $r' \leftarrow rand(R)$
28:         $assign(n, r')$
29: $save\_result$
```

A. Resources

The number of resources that would be generated is provided as a parameter for the proposed tool. For every generated resource its standard rate salary is set as a random between the minimal and maximal value (see Alg. 1, line: 5) set by the end–user in the configuration of the generator.

B. Skills

Analogously to resource definition, the number of different skill types is set by the end–user during the configuration. We

[1]http://mpxj.sourceforge.net
[2]http://imopse.ii.pwr.edu.pl

declared 4 levels of the skill familiarity for given resource. However, the end–user is also obliged to define how many types of skills could be covered with given resource. It is desired that number of skill types owned by resource would be no greater than the number of skill types existing in the project. Number of skill types is set randomly from the minimal and maximal value (set by end–user). However, during recent dataset instances generation, we decided to make the number of skill types for every resource as a constant – minimal (min) and maximal (max) number of skill types have been set as the same value – line 7.

During skill generation process for given resource, a skill type is selected from given range of types (line: 9) while skill level is also selected from given scope (line: 10). We decided to make four levels of skills as it covered the requirements presented by project manager from the enterprise. If selected skill is not available in skills pool, it is both assigned to the resource and added to the skills pool (line 13). It provides that skills required by any tasks to be performed would be selected from the pool of skills that are owned by at least one resource (line 17).

C. Tasks

Having resources, and skills covered by them defined, tasks could be obtained. The number of tasks is set by the end–user. What it more user also sets the duration scope of the task (line: 16). Those are the bounds within the task duration would be randomly set (line: 16). For the sake of generation project instances for our research, we made an assumption that task duration would be the number between 8 and 40 hours. It reflects to the range between 1 and 5 days of any task's duration. The skill required by any task is selected from the pool of available skills in given project instance (line: 17).

D. Precedence relations

One of the last steps during generation process is to define the precedence relations. End–user defines the number of those relations. S/he is also responsible for defining the general scope of relations. It means, the bigger relations scope set, the bigger distance between tasks is allowed in building the precedence relations diagram (line: 21). In other words, setting small relation scope could cause that resulted schedule would contain precedence relations between tasks that have been defined one by one or with slight distance (like task first and third). However, if the relation scope would be set to a bigger value, there could be relations in the final schedule between some tasks defined in the beginning and the end of the generation process (e.g. precedence relation between the first and the last task defined).

The bigger the distance between source and destination task is, the more complex the project instance critical path is. As a consequence, duration–based optimization would potentially be more difficult for such project instance.

E. Assign resource

Finally, the initial schedule could be built by assigning resources to given tasks, preserving precedence and skill constraints (lines: 26–28). Produced schedule would always be feasible. The way how resources are assigned to tasks is set randomly. Hence, if there is more than one resource that can be assigned to given task, then generator could assign this task in different ways in different executions of generation process. Schedule is generated using the Serial Generation Scheme [9], what provides that generated schedule would be always feasible.

F. Save project to file

The last step in the process of generation an instance is to save (line: 29) the resulted project. If user sets the output file type to *xml* (*mpp*), then generator produces the result in the format that could be easily loaded in Microsoft Project tool. If user selects *def* output format or does not select any, then the result would be saved in more compact format that could be read by any text editor. If *assign resources* option has been ticked, then tasks can have resources assigned. However it regards only generating output only in xml (mpp) format. The name of produced file relates to the name proposed by the end–user in given text field in the configuration screen.

VI. Dataset summary

Due to evaluate not only the project schedule duration, but also the cost of the schedule including skills domain, we cannot use the standard PSPLIB benchmark dataset [8] in our research; 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 36 project instances, which have been artificially created, in a base of real–world instances, got from the Volvo IT Department in Wroclaw.

The dataset summary has been presented in the Table II. There are two groups of created project instances: one contains 100 tasks and the second – 200 tasks as typical ones performed in given international enterprise. Within each group, project instances are varied by number of available resources and the precedence relationship complexity. Number of resources for instances from both groups were chosen in a way to preserve constant average resource load and average task relations ratio for given instances. The skill variety has been set up to 9 or 15 different skill types for each project instance while any resource can dispose of exactly six different skill types. Because of the different resources and relations number, the scheduling complexity for each project is varied.

This dataset stands as an extension of dataset presented in [18], [19], [20], and that is the reason some instances are named with suffix *Dx*. This suffix refers to dataset instances that have been previously created and presented in those papers. Because of the extension the dataset, the need of introducing more clear namesystem has arisen. Suffix has been added to refer previously created files, keeping the naming convention applied after dataset extension.

A. Project definition format (.def)

Because of changing the research's approach to be more generic, we decided to focus more on the dataset instances

TABLE II
COMPLEXITY INDICATORS AND DIFFICULTY MEASURE FOR iMOPSE DATASET INSTANCES. PROJECT INSTANCES REGARDED AS THE MOST DIFFICULT TO BE SCHEDULED ARE WRITTEN **BOLD**, WHILE THOSE ONES, WHO ARE INDICATED AS THE EASIEST TO SCHEDULE ARE WRITTEN *ITALIC*.

Dataset instance	Features				Indicators								Difficulty
	m	n	p	$\bar{q}$	λ	ν	Φ_d	Φ_c	μ	β	π	θ	
100_10_26_15	100	10	26	15	0.100	0.053	0.316	0.123	1.000	0.117	0.200	0.278	0.243
100_10_27_9_D2	100	10	27	9	0.100	0.062	0.316	0.412	1.000	0.087	0.102	0.417	0.267
100_10_47_9	100	10	47	9	0.100	0.095	0.314	0.182	1.000	0.087	0.094	0.437	0.259
100_10_48_15	100	10	48	15	0.100	0.097	0.313	0.125	1.000	0.113	0.263	0.292	0.263
100_10_64_9	100	10	64	9	0.100	0.129	0.321	0.129	1.000	0.083	0.176	0.453	**0.277**
100_10_65_15	100	10	65	15	0.100	0.131	0.321	0.137	1.000	0.120	0.179	0.281	0.256
100_20_22_15	100	20	22	15	0.050	0.044	0.317	0.119	1.000	0.075	0.116	0.263	0.221
100_20_23_9_D1	100	20	23	9	0.050	0.052	0.317	0.356	1.000	0.045	0.135	0.451	0.265
100_20_46_15	100	20	46	15	0.050	0.093	0.321	0.125	1.000	0.072	0.102	0.264	0.229
100_20_47_9	100	20	47	9	0.050	0.095	0.314	0.122	1.000	0.043	0.081	0.397	0.243
100_20_65_15	100	20	65	15	0.050	0.131	0.314	0.117	1.000	0.073	0.091	0.248	0.232
100_20_65_9	100	20	65	9	0.050	0.131	0.318	0.114	1.000	0.045	0.068	0.426	0.251
100_5_20_9_D3	100	5	20	9	0.200	0.043	0.315	0.503	1.000	0.133	0.147	0.480	**0.296**
100_5_22_15	100	5	22	15	0.200	0.044	0.317	0.207	1.000	0.140	0.250	0.325	**0.275**
100_5_46_15	100	5	46	15	0.200	0.093	0.320	0.243	1.000	0.153	0.345	0.281	**0.296**
100_5_48_9	100	5	48	9	0.200	0.097	0.315	0.294	1.000	0.140	0.213	0.376	**0.291**
100_5_64_15	100	5	64	15	0.200	0.129	0.322	0.197	1.000	0.147	0.176	0.294	**0.276**
100_5_64_9	100	5	64	9	0.200	0.129	0.315	0.149	1.000	0.133	0.176	0.391	**0.285**
200_10_128_15	200	10	128	15	0.100	0.064	0.314	0.115	1.000	0.117	0.136	0.130	0.218
200_10_135_9_D6	200	10	135	9	0.100	0.084	0.318	0.428	1.000	0.087	0.139	0.200	0.254
200_10_50_15	200	10	50	15	0.100	0.025	0.317	0.111	1.000	0.110	0.130	0.147	0.212
200_10_50_9	200	10	50	9	0.100	0.025	0.318	0.130	1.000	0.087	0.127	0.212	0.222
200_10_84_9	200	10	84	9	0.100	0.042	0.313	0.124	1.000	0.083	0.200	0.231	0.238
200_10_85_15	200	10	85	15	0.100	0.043	0.315	0.121	1.000	0.107	0.176	0.143	0.223
200_20_145_15	200	20	145	15	0.050	0.073	0.313	0.124	1.000	0.072	0.096	0.133	*0.209*
200_20_150_9_D5	200	20	150	9	0.050	0.093	0.315	0.386	1.000	0.043	0.055	0.228	0.238
200_20_54_15	200	20	54	15	0.050	0.027	0.314	0.119	1.000	0.070	0.102	0.124	*0.200*
200_20_55_9	200	20	55	9	0.050	0.028	0.315	0.115	1.000	0.045	0.226	0.230	0.233
200_20_97_15	200	20	97	15	0.050	0.049	0.315	0.118	1.000	0.073	0.116	0.123	*0.206*
200_20_97_9	200	20	97	9	0.050	0.049	0.314	0.116	1.000	0.045	0.078	0.206	0.212
200_40_130_9_D4	200	40	130	9	0.025	0.088	0.316	0.342	1.000	0.023	0.165	0.183	0.243
200_40_133_15	200	40	133	15	0.025	0.067	0.314	0.131	1.000	0.038	0.067	0.118	*0.201*
200_40_45_15	200	40	45	15	0.025	0.023	0.314	0.115	1.000	0.038	0.046	0.135	*0.190*
200_40_45_9	200	40	45	9	0.025	0.023	0.316	0.113	1.000	0.023	0.122	0.216	0.212
200_40_90_9	200	40	90	9	0.025	0.045	0.314	0.117	1.000	0.023	0.116	0.221	0.216
200_40_91_15	200	40	91	15	0.025	0.046	0.317	0.112	1.000	0.037	0.075	0.131	*0.198*

stored in *.def* format that is easier to maintain and use by researchers. Hence we adopted instances created for MS Project to more generic form.

It led to remove the summary tasks that are specific for MS Project *.mpp* format. Summary tasks are used to group atomic tasks into more complex (i.e. task called 'development' could be split to some atomic tasks: database structures creation, development of business logic and development of user interface). However, MS Project allows to use summary tasks as predecessors for others. Therefore we multiplied precedence relations by copying them from predecessor's summary task to all of tasks included by this summary one. As a result new *Dx* instances have been created. Furthermore, some tasks, represented as summary ones, have been removed from the project. To be consistent with previous works, we keep names of those files the same. Those files are provided with additional description explaining the difference in number of tasks and precedence relations between file name and file content.

Adjusted dataset instances with *Dx* suffix have smaller number of tasks. Number of precedence relations is significantly bigger in all *Dx* instances. Roughly describing, it is more than twice precedence relations as in former instances, while number of tasks has been decreased in all instances in about 20% (about 20 tasks for instances with 100 tasks and 40 for instances with 200 tasks).

We have also presented in Tab. II the values of proposed complexity estimations. Finally, the overall complexity measure, as an aggregation value of complexity estimations components has been presented. Based on the overall complexity value, the most complex projects in scheduling point of view has been highlighted by bold. The overall complexity measure has been computed according to the Eq. 14. Changing weights of complexity estimations components would affect the final complexity value for each dataset instance. Hence, the complexity of each project could be different depending on priorities set by project manager.

In our approach all universality estimations values are equal to 1. It is because we made an assumption that every resource

TABLE III
COMPARISON OF RESULTS OBTAINED FOR GREEDY ALGORITHM, SIMPLE HEURISTICS, ACO [14] AND HANTCO [14] FOR VARIOUS OPTIMIZATION MODES FOR CALENDAR–CONSTRAINED DATASET INSTANCES (*.mpp*).

Dataset instance	DO								CO					
	Heuristic		Greedy		ACO		HAntCO		Heuristic		ACO		HAntCO	
	Days	Cost	Days	Cost	Days	Cost	Days	Cost	Days	Cost	Days	Cost	Days	Cost
100_10_26_15	37	126361	38	119336	32	124687	**31**	126216	85	**70326**	85	**70326**	85	**70326**
100_10_27_9_D2	38	44309	38	43438	34	44999	**33**	42199	129	**26323**	129	**26323**	129	**26323**
100_10_47_9	41	142759	40	135161	36	143100	**34**	140865	145	**90992**	145	**90992**	145	**90992**
100_10_48_15	36	135534	44	120664	33	133062	**33**	133495	85	**87187**	85	**87187**	85	**87187**
100_10_64_9	39	113124	43	117993	35	110643	**33**	113774	121	**62102**	121	**62102**	121	**62102**
100_10_65_15	40	152955	43	140782	35	150294	**32**	149185	98	**106296**	98	**106296**	98	**106296**
100_20_22_15	25	117493	24	112135	20	120949	**19**	123642	86	**55240**	87	55240	87	55240
100_20_23_9_D1	32	53154	32	50279	32	52119	**23**	53358	119	30104	121	30107	117	**30104**
100_20_46_15	28	138270	29	133739	25	138565	**24**	138568	75	**68899**	75	**68899**	75	**68899**
100_20_47_9	21	129160	28	140626	21	124817	**18**	134312	131	**55197**	131	**55197**	131	**55197**
100_20_65_15	32	110503	34	118569	**27**	109831	27	108991	69	**57085**	69	**57085**	69	**57085**
100_20_65_9	25	127149	24	124291	23	130934	**21**	126659	114	**59736**	114	**59736**	114	**59736**
100_5_20_9_D3	57	40539	55	40958	**50**	41029	53	40811	167	**30164**	167	**30164**	167	**30164**
100_5_22_15	63	119266	77	128354	**60**	119434	60	119158	86	**109111**	86	**109111**	86	**109111**
100_5_46_15	75	202238	80	202607	**67**	204110	67	204730	125	**184409**	125	**184409**	125	**184409**
100_5_48_9	72	193383	78	196893	**62**	191712	62	191888	130	**175225**	130	**175225**	130	**175225**
100_5_64_15	71	141407	66	141882	62	144972	**61**	143956	141	**109091**	141	**109091**	141	**109091**
100_5_64_9	71	102439	67	107014	61	102777	**61**	101297	173	**72848**	173	**72848**	173	**72848**
200_10_128_15	71	180812	78	198378	62	178264	**60**	178375	159	**134425**	143	136551	143	136551
200_10_135_9_D6	216	105593	216	93426	216	99375	**186**	103561	256	**71986**	274	72036	270	71986
200_10_50_15	66	189660	75	183673	63	191856	**62**	190956	167	**84308**	167	**84308**	167	**84308**
200_10_50_9	66	251158	70	250732	65	250075	**64**	250850	318	**105198**	318	105232	318	**105198**
200_10_84_9	70	224121	66	222976	69	226666	**66**	222655	338	**117543**	316	117754	318	**117543**
200_10_85_15	65	304277	68	301357	**61**	306949	62	302064	215	**195820**	215	195820	215	**195820**
200_20_145_15	36	275983	46	277097	36	278199	**35**	272504	158	**143497**	152	143688	158	**143497**
200_20_150_9_D5	183	92821	183	95667	186	91461	**177**	92567	337	**51496**	296	51678	345	51496
200_20_54_15	37	295786	41	290656	39	299993	**34**	298822	125	**161412**	131	161614	125	**161412**
200_20_55_9	37	230150	37	232766	38	231094	**36**	223879	332	**70057**	250	72176	332	**70057**
200_20_97_15	49	290399	69	346527	42	280951	**42**	277860	171	**156951**	169	157202	171	**156951**
200_20_97_9	**35**	273378	43	282379	37	275819	35	278797	169	**98480**	150	99901	169	**98480**
200_40_130_9_D4	112	101879	112	90907	112	94488	**108**	104965	214	**46133**	205	48419	216	46275
200_40_133_15	24	276456	**23**	279170	27	281933	24	279073	155	97345	131	99329	144	**97345**
200_40_45_15	31	260738	32	269623	25	248717	**23**	256687	213	**87955**	161	91010	213	**87955**
200_40_45_9	**22**	270758	23	276416	26	273632	25	270428	334	**77236**	179	94142	315	82192
200_40_90_9	24	290028	**20**	294909	26	287694	24	298340	285	**80732**	142	96312	247	84038
200_40_91_15	**19**	249909	35	250843	25	257927	23	241492	184	**86476**	132	88616	184	**86476**

has the same number of different skills and this is also the maximal number of potential skills covered by resource, used in normalization. If we decided to make the number of skills covered by resource various, depending to given resource, then the universality estimation values would not be always equal to 1.0.

B. *iMOPSE Solution Validator*

We released also an additional tool to validate generated solutions in case of preserving all constraints defined in MS–RCPSP. Such validator is available on the iMOPSE project website[3]. Validator checks whether all tasks have any resource assigned (assignments validation), final schedule is conflict–free (conflicts validation), any task having predecessors is set to be started after all its predecessors would be finished (precedence relations validation) and whether any task has resource assigned that is capable of performing it (skill validation).

Validator shows not only the validation results but also the quality the validated solution – its duration measured in hours and cost measured in some currency units. If some validation rules are broken, they are shown to end–user.

Validator is compatible with .def project definition format. For further information how to use the validator, please refer to documents related with the tool – User's Manual or Case study – available on iMOPSE Platform.

[3]http://imopse.ii.pwr.edu.pl

VII. EXPERIMENTS AND RESULTS

The main goal of conducted experiments was to link and compare both (*.mpp* [14] and *.def* based) approaches, considering the impact of calendar restrictions.

We decided to use simple duration– and cost– oriented heuristic [20], greedy algorithm and compare them with ACO and HAntCO approaches described in [14]. Furthermore, greedy algorithm and simple heuristics have been used to schedule .def dataset instances.

However, proposed heuristic and greedy approaches for cost optimization turned out to become the same method. Therefore, presented results are divided into main two parts regarding optimization modes: duration optimization ($\omega_\tau = 1$) and cost optimization ($\omega_\tau = 0$). Each of those main part is also divided for three parts in cost optimization (heuristic, ACO, HAntCO) and four parts in duration optimization (heuristic, greedy, ACO, HAntCO).

Table III presents the obtained results for both optimization modes using both proposed methods (simple heuristic and greedy algorithm). It also contains results obtained by ACO and HAntCO approaches described in detail in [14]. This table presents optimization results for dataset instances with calendar restrictions (*.mpp*).

Greedy algorithm is a method that works iteratively. In every step of greedy scheduling, one task is added to the schedule. The decision, which task to which resource should be assigned in given algorithm step is made based on the current partial schedule. In other words: in a given step, currently best task–to–resource assignment option is chosen and the next step is performed until all tasks would be scheduled. Classical greedy algorithm assumes possibility to analyse not only current state of the partial schedule, but also to investigate several further steps. In that approach combinations of several tasks are analysed and the best one, containing given number of tasks–to–resource assignments is selected and added to the partial schedule. However, in our approach we discuss only current schedule state, omitting the analysis of several assignments. Therefore, number of steps of proposed greedy algorithms would be always equal to the number of tasks in a project.

For duration oriented optimization mode, greedy algorithm analyses which task should be added to make the partial schedule the shortest. For cost-oriented optimization mode, the criteria of selecting tasks bases on the cost of the assignment given task to given resource. For every task, various resources are analysed to be assigned, and the cheaper one is chosen.

In cost-oriented optimization mode, both greedy algorithm and simple heuristic (Resource Salary based [20]) works according to the same schema, described above. However, for the duration–oriented optimization, heuristic and the greedy algorithm differs in details. In the greedy algorithm task is assigned to given resource and then added to the partial schedule then conflicts are fixed and finally the project duration is computed. In simple heuristic (Successors List Size based [20]) firstly the resource that would be the earliest free (not assigned to any task) is selected. Then task is assigned to this found task while its start time is set just after then end of the last of tasks previously assigned to given resource. It allows to build feasible schedule without the necessity of fixing conflicts as the method is the resource conflict–free.

Taking into account results gathered in the Tab. III we can conclude that for duration optimization method, the HAntCO outclassed other methods, provided the best results in 28 of 36 cases (78%). ACO turned out to be the best method for 6 cases (16%), while greedy gave best results in 3 of 36 cases (8%) and heuristic was the most suitable in 2 of 36 cases (6%).

For cost optimization method, simple heuristic gives the best result for almost all of dataset instances (34/36, 94%). However, for remaining two instances heuristic also provided solution with the smallest cost, but the schedule duration was bigger than for other method (HAntCO). For most of the instances (32/36, 89%) HAnt-CO provided the same, best result than obtained from heuristic. ACO–approach provided the same, best results in 18/36 (50%) cases. The most interesting fact for cost optimization is that ACO provided best results mostly for dataset instances containing 100 tasks - 17/18 cases (94%) and only once for dataset instances containing 200 tasks (6%).

In the Tab. IV we compiled the summary of obtained best results for classical optimization methods - heuristics and greedy algorithm for instances not regarding calendar restrictions (*.def*). It can be also found in the iMOPSE website. As we are oriented to use *.def* format in further research, obtained project schedules are measured by hours rather than days as it has been so far, in *.mpp* format. Obtained results stand as a benchmark for further research when using .def format. On the other hand, Tab. III is still regarded as a benchmark for methods working on *.mpp* format.

TABLE IV
SUMMARY OF BEST OBTAINED RESULTS FOR DATASET INSTANCES NOT REGARDING CALENDAR CONSTRAINTS (*.def*).

Dataset instance	Heuristic CO		Heuristic DO		Greedy CO	
	Hours	Cost	Hours	Cost	Hours	Cost
100_10_26_15	728	71616	316	125073	370	130315
100_10_27_9_D2	1184	26771	334	44319	646	42984
100_10_47_9	1224	92771	310	144840	549	162642
100_10_48_15	766	88794	325	138845	344	139761
100_10_64_9	1028	63279	324	117759	533	124897
100_10_65_15	831	108239	285	152669	426	173754
100_20_22_15	756	56151	162	121561	353	98621
100_20_23_9_D1	1219	30643	247	52436	617	63210
100_20_46_15	639	70061	231	142962	394	140994
100_20_47_9	1114	56190	179	130612	390	119462
100_20_65_15	582	58134	298	111130	310	125081
100_20_65_9	964	60954	174	127260	408	147952
100_5_20_9_D3	1408	30728	523	40976	625	38725
100_5_22_15	723	111189	537	120039	630	121369
100_5_46_15	1054	187623	658	207810	693	212261
100_5_48_9	1092	178346	580	196221	779	191888
100_5_64_15	1195	111388	574	146661	640	149635
100_5_64_9	1506	74199	567	109518	597	101062
200_10_128_15	1217	139149	537	179335	780	213091
200_10_135_9_D6	2581	73207	1079	105604	1426	105196
200_10_50_15	1414	86008	549	190555	763	190981
200_10_50_9	2681	106986	536	251903	817	239238
200_10_84_9	2702	119500	589	231457	999	232937
200_10_85_15	1813	199585	545	314599	706	346573
200_20_145_15	1331	146303	293	280623	480	280774
200_20_150_9_D5	3024	52542	1232	94355	1930	116179
200_20_54_15	1054	164142	306	299677	488	322627
200_20_55_9	2809	71262	280	233960	999	276513
200_20_97_15	1491	159680	347	294938	680	324041
200_20_97_9	1515	100421	304	279894	816	301723
200_40_130_9_D4	2038	47050	586	104261	1710	121485
200_40_133_15	1282	99266	183	285299	512	270201
200_40_45_15	1807	89642	267	266970	616	269754
200_40_45_9	2781	79979	198	273818	821	218708
200_40_90_9	2405	82177	173	292873	963	300258
200_40_91_15	1560	88233	179	250005	519	278582

Results obtained in the Tab. IV show that SLS [20] heuristic provides better results in DO mode in all of 36 dataset instances. It clearly shows that SLS heuristic is definitely better optimization method than greedy algorithm in this problem.

However the project definitions are compatible to each other between *.def* and *.mpp* formats, there are some small differences in cost result in CO, using the same method. It is because of the adjustment made when transferring *.mpp* to *.def* format. For the sake of simplicity, task's duration in .def has been rounded up to the integer values. It lead to the differences, because cost of performing project is a sum of each task's performance cost. While task's performance cost is computed as a multiplication of task duration and assigned resource's salary. As a result of rounding up, cost of each task has increased slightly, even though the same resource is assigned to it. Therefore the overall cost is slightly bigger for solutions obtained for *.def* files.

VIII. Conclusions and further work

In this paper some novel difficulty indicators for instances of Multi–Skill Resource–Constrained Project Scheduling Problem have been presented. Furthermore the extended dataset has been presented and suggested as a benchmark for this problem, as no other benchmark dataset can be found that satisfies proposed constraints. Furthermore, those instances have been scheduled using greedy algorithm, to provide an initial platform for comparing results obtained by various researchers.

Proposed complexity estimations stand some first step in project scheduling data analysis. Guessing the project complexity could be helpful in parameters' tuning for various optimization methods. As more complex / difficult to schedule project is, the optimization process would potentially last longer for the same parameter configuration than for other instances. Hence, the decision maker could decide to change the parameters, e.g. by decrease number of method iterations. We managed to make those observations sure in our EA–based approach, where building schedule for the project with suffix D2 generally lasts longer than for the project with suffix D1.

The goal of presenting the dataset instance generator is to allow and encourage other researchers to focus on the problem and possible solutions and methods we propose. We still believe there is a lot to investigate and research. What is more, the dataset instance format we propose is very common in many industries, as the MS Project is a common standard.

We are also on the point of investigating approaches concentrated to different multi–objectiveness handling methods. Most of them we analyse are based on Pareto–front (like NSGA-II [22], [17] or other methods). One of the goals is to find a way how to provide a set of non–dominated results to the project manager, to delegate the matter of making decision which of those proposed solutions is the best, according to the specificity of the company it regards. E.g. in some industries the aim is to finish the project as soon as possible while in some others the most important is to perform it in the cheapest way. Still we would like to give the choice from a pool of some solutions.

References

[1] Al–Anzi F.S., Al–Zamel K., Allahverdi A.; Weighted Multi–Skill Resources Project Scheduling, J. of Software Engineering & Applications (3), pp. 1125–1130, 2010.

[2] 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.

[3] Drezet L.E., Billaut J.C.; A project scheduling problem with labour constraints and time–dependent activities requirements, Int. J. of Production Economics (112), pp. 217-225, 2008.

[4] Gonzalez F., Ramies Rios D., Multi–objective Optimization of the Resource Constrained Project Scheduling Problem (RCPSP) A heuristic approach based on the mathematical model, The Int. J. of Computer Science & Applications (TIJCSA) (2/2), pp. 1–13, 2013.

[5] Hegazy T., Shabeeb A.K., Elbeltagi E., Cheema T.; Algorithm for scheduling with multiskilled constrained resources, J. of Construction Engineering and Management (11-12/2000), pp. 414–421, 2000.

[6] Jaberi M., Jaberi M.; A Multi–objective Resource–Constrained Project–Scheduling Problem Using Mean Field Annealing Neural Networks, J. of Mathematics and Computer science (9), pp. 228–239, 2014.

[7] 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,

[8] Kolisch R., Sprecher A., PSPLIB - A project scheduling problem library, European Journal of Operational Research (96), pp. 205–216, 1996.

[9] 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.

[10] 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.

[11] Latva-Koivisto A., M., Finding a complexity measure for business process models, Research Report, Mat-2.108, Individual Research Projects in Applied Mathematics, 2001.

[12] Li H., Womer K.; Scheduling projects with multi-skilled personnel by a hybrid MILP/CP benders decomposition algorithm, J. of Scheduling, (12), pp. 281-298, 2009.

[13] Luna F., Gonzalez-Alvarez, D. L., Chicano F., Vega-Rodriguez M. A.; The software project scheduling problem: A scalability analysis of multi-objective metaheuristics, Applied Soft Computing Vol. 15, pp.136–148, 2013.

[14] Myszkowski P. B., Skowroński M.E., Olech Ł., Oślizło K.; Hybrid Ant Colony Optimization in solving Multi-Skill Resource-Constrained Project Scheduling Problem, Soft Computing, DOI 10.1007/s00500-014-1455-x, 2014.

[15] Phruksaphanrat B.; Multi–Objective Multi–Mode Resource–Constrained Project Scheduling Problem by Preemptive Fuzzy Goal Programming, World Academy of Science, Engineering and Technology, Int. J. of Mechanical, Industrial Science and Engineering (8/3), pp. 99–103, 2014

[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] Sarker B.R, Yu J., Mungan D., Rahman M.A.A., Parveen S.; Pareto–optimal solution of a scheduling problem on a single machine with periodic maintenance and non pre–emptive jobs, Proceedings of the International Conference on Mechanical Engineering, pp. 1–5, 2007.

[18] Skowroński M. E., Myszkowski P. B., Specialized genetic operators for Multi-Skill Resource-Constrained Project Scheduling Problem, 19th Inter. Conference on Soft Computing Mendel 2013, pp. 57-62, 2013.

[19] Skowroński M. E., Myszkowski P. B., Kwiatek P., Adamski M., Tabu Search approach for Multi-Skill Resource-Constrained Project Scheduling Problem, Annals of Computer Science and Information Systems Volume 1, Proc. of the 2013 Federated Conference on Computer Science and Information Systems, pp. 153-158, 2013.

[20] Skowroński M. E., Myszkowski P. B., Podlodowski Ł., Novel heuristic solutions for Multi-Skill Resource-Constrained Project Scheduling Problem, Annals of Computer Science and Information Systems Volume 1, Proc. of the 2013 Federated Conference on Computer Science and Information Systems, pp. 159-166, 2013.

[21] Van Peteghem, Vanhoucke M., An experimental investigation of metaheuristics for the multi–mode resource–constrained project scheduling problem on new dataset instances, European Journal of Operational Research (235), pp.62-72, 2014.

[22] Vanucci C.S, Bicalho R., Carrano E.G., Takahashi R.H.C.; A Modified NSGA-II for the Multiobjective Multi–mode Resource-Constrained Project Scheduling Problem, WCCI 2012 IEEE World Congress on Computational Intelligence June, pp. 10–15, 2012.

[23] Wang Y., Chen D., Liu S., Zeng Q.; An Instance Generator for Project Scheduling Problems with Multi-Skilled Personnel Constraints, 2012 24th Chinese Control and Decision Conference (CCDC), pp. 3430–3435, 2012.

[24] Yannibelli V., Amandi A.; Hybridizing a multi–objective simulated annealing algorithm with a multi–objective evolutionary algorithm to solve a multi–objective project scheduling problem, Expert Systems with Applications (40), pp. 2421-2434, 2013.