Proceedings of the Federated Conference on Computer Science and Information Systems pp. 75–82
DOI: 10.15439/2017F318
ISSN 2300-5963 ACSIS, Vol. 11

Co–Evolutionary Algorithm solving Multi–Skill Resource–Constrained Project Scheduling Problem

Paweł B. Myszkowski, Maciej Laszczyk, Dawid Kalinowski
Collective Intelligence Department
Wrocław University of Science and Technology
Wyb. Wyspiańskiego 27, 50-370 Wrocław, Poland
email: pawel.myszkowski@pwr.edu.pl, maciej.laszczyk@pwr.edu.pl

***Abstract*—This paper presents methods solving MS–RCPSP as a main task–resource–time assignment optimization problem. In the paper there are presented four variants of Evolutionary Algorithm applied to MS–RCPSP problem: concerning prioritization the tasks (or resources), combined task–resources prioritizing approach and co–evolution based approach that effectively solves problem dividing it to two subproblems. All approaches are examined using benchmark MS–RCPSP iMOPSE dataset and results show that the problem decomposition is effective. All experiments are described, statistically verified and summarized. Conclusions and promising areas of future work are presented.**

I. Introduction

MANY practical problems can be effectively solved by (meta)heuristics. Particularly, the problems that are NP–hard, over-constrained, combinatorial and have huge solution landscape. Scheduling belongs to this group of problems, which is applied in the real-world, where the problem exists with rare and/or expensive resources. In this situation, the project manager role is to find such resource usage to realize set of tasks in the most effective way. Mainly, it is a quite casual definition of Project Scheduling Problem (PSP), where effectiveness measure is the project realization time. The PSP is too general and in real-world usage is extended to Resource Project Scheduling Problem, where generally speaking, resources are not only limited but also not every resource can be applied to each task. In practical application, such problem specialization goes further, e.g. in IT industry to realize product/service several various types of resources must cooperate to build high-quality software. Such (human) resources differ in skills and levels (e.g. "Java programmer – advanced", "software architect – basic") and can be employed in various roles that need particular skills. Of course, resources differ also in salaries, which makes the optimization problem focused on time and/or cost. This way Multi–Skill Resource–Constrained Project Scheduling Problem (MS–RCPSP) can be described.

The MS–RCPSP problem is presented in literature (e.g.[3][12][8]) as extension of RCPSP [4]. It is NP–hard [2] and there is no effective algorithm to solve it. Thus, several types of (meta)heuristic methods can be effectively applied. There can be found applications of (MS–)RCPSP in literature based on: heuristics [3][12], tabu search [11], evolutionary algorithms (EA) with specialized operators [10], (hybridized) ant colony optimization [9], teaching–learning-based optimization algorithm [13], differential evolution [14], hybridized differential evolution [6] and many others.

The MS–RCPSP problem is connected to resource–task–time assignment. Many approaches deal with it using reduced solution space by problem modification. E.g. in hybridized differential evolution [6] metaheuristic DE operates on search space that prioritizes resource and task sequencing is solved by greedy–based method. Such approach is effective, but greedy usage may cause that method to get stuck in local optima. This is not the only way of problem decomposition. Some methods employ a natural co–evolution mechanism which can be applied in RCPSP problems too, e.g.[15]. This paper concerns co–evolutionary algorithms that eliminate greedy usage. In proposed method MS–RCPSP problem is decomposed into two subproblems: effective (1) task prioritizing and (2) resource prioritizing to build a final feasible schedule.

The main motivation of this paper is to examine the effectiveness of co–evolution usage. To do that several EA–based approaches are tested and compared. One approach uses genome, which proposes only resources' priorities (EA_R) that are converted by greedy to build schedule. Other reference approach (EA_T) proposes tasks' priorities and analogously greedy–like algorithm converts into schedules. To eliminate the greedy usage, an approach is introduced with connected genome that proposes resources' and tasks' priorities (EA_RT). All provided methods use the same problem solution landscape: representation, selection, genetic operators and fitness function. Finally, using co-evolution EA (Co_RT) two problem are separated (resource– and task– priorities) into two separate populations, and the only connection is kept by selection to build final schedule to get the fitness function value.

The rest of the paper is organized as follows. In section II the MS–RCPSP problem brief statement with constraints and requirements is described. The description of proposed EA–based approaches are given in section III, where details of examined methods are provided, especially details that are connected with adaptation of EA to MS–RCPSP problem. Section IV-B presents experimental procedure, parameters tunning method, used dataset and finally gained results summarized and concluded (see IV-C). Last section V concludes the article and presents potential directions for future work.

II. Formulation of MS–RCPSP

In classical RCPSP problem, there is defined a task–resource–timeslot assignment. Some constraints must be satisfied to get a feasible schedule. Every task is no–preemptive and can be described by duration, start and finish time. Tasks are related by precedence relation, defining which tasks need to be completed before others can be started. A set of discrete time (timestamps) and resource in RCPSP is used. Only one resource can be assigned to a given task. Additionally, the resource cannot be assigned to more than one task in overlapping period – dedicated resources [1] have been used.

The MS–RCPSP introduces some practical extensions, e.g. adds the skills domain and the resource salary (as an hourly wage) paid for performed work, while resources represent human resource and are varied by salary. Each task requires a subset of skills and not every resource can be applied to its realization. In provided MS–RCPSP model every resource possesses a subset of skills from the skill pool (e.g. analyst, architect, developer, tester, etc.) defined in a project. Each resource' skill is given with familiarity level – it means that the resource R is capable of performing the task T only if R disposes skill required by T, at the same or higher level. The sample of capabilities of performing tasks by resources as skill matrix is shown in the Fig. 1.

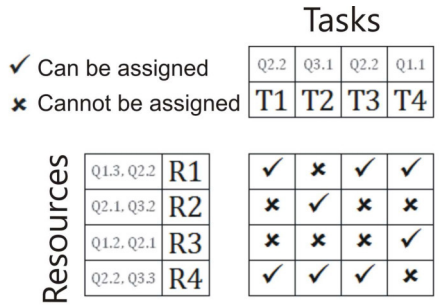

Fig. 1: Example of skill matrix [9]

In the skill matrix presented in Fig. 1, skills required by task to be performed have been written over task definition. Skills owned by resources have been written next to resource definition. For example, resource $R1$ has access to skills $Q1$ and $Q2$ with familiarity level 3 and 2 respectively. It means that $R1$ is capable of performing tasks $T1$, $T3$ and $T4$ as skills required for them are no higher level than the ones owned by $R1$. On presented example, $R1$ cannot be assigned to $T2$ because $R1$ does not posses required skill $Q2$. However, resource $R2$ can be assigned to task $T2$ (has required $Q3$ skill) and analogously resource $R3$ is a proper one for task $T4$. Finally, resource $R4$ can perform tasks $T1$, $T2$ and $T3$. Another situation occurs in case $R3$, if it possesses skill $Q2$ but cannot be assigned to $T1$ and $T3$ because these tasks require $Q2$ at higher familiarity level than this resource disposes.

The goal of RCPSP is to find such task–resource assignments to make the final feasible schedule as short as possible. The combinatorial nature of the RCPSP makes it NP–hard [2] problem. Analogously, the solution of MS–RCPSP is a feasible schedule – the one in which resource units and precedence constraints are preserved. Moreover, skills domain extends the schedule feasibility for MS–RCPSP from the classical RCPSP definition – only resources capable of performing given tasks can be assigned to them.

The MS–RCPSP as an optimization problem can be analyzed as two separate goals: (1) optimization of final schedule duration and (2) optimization of final schedule cost. More formal definition of MS–RCPSP as optimization problem has been presented in II-A – and is based on work [6][9]. The MS–RCPSP has been defined in cooperation with international corporation (Volvo Group IT). Next section describes MS–RCPSP more formally – is based on [8][9][6].

A. MS–RCPSP definition

The feasible Project schedule (PS) consists of $J = 1, ..., n$ tasks and $K = 1, ..., m$ resources. A non pre–emptive duration d_j, start time S_j and finish time F_j is defined for each task. Set of predecessors of given task j are defined as P_j. Each resource is defined by its hourly rate salary s_k and owned skills $Q^k = 1, ..., r$, while a pool of owned skills is a subset of all skills defined in project $Q^k \in Q$. Value l_q denotes the level of given skill, while h_q describes its type and q_j is a skill required by j to be performed. Therefore, by J^k subset of tasks that can be performed by $k-th$ resource is defined. Analogously, K_j is a subset of resources that can perform task j. Duration of a project schedule is denoted as τ. The cost of performing j task by k resource is denoted as $c_j^k = d_j * s_k$, where s_k describes the salary of resource k assigned to j.

For simplicity, we have modified the cost of the task's performance from c_j^k to c_j, because only one resource can be assigned to given task in the duration of the project. Hence, there is no need to distinguish various costs for the same task. Moreover, we have introduced variable that defines whether k is assigned to j in given time t: $U_{j,k}^t \in \{0; 1\}$. If $U_{j,k}^t = 1$, k is assigned to j in t. Analogously, k is not assigned to j in t if $U_{j,k}^t = 0$.

Resource assigned to task j is denoted as k_j. Furthermore, every resource is denoted by its start time T_k - the time when it starts working on the project.

Feasible project schedule (PS) belongs to the set of all feasible and non–feasible solutions (violating precedence-, resource- and skills- constraints) : $PS \in PS_{all}$.

Formally, the problem could be regarded as optimization (minimization) problem and stated as follows:

$$\min f(PS) = \min [f_\tau(PS), f_C(PS)] \tag{1}$$

The Eq. 1 describes the duration $f_\tau(PS)$ and cost optimization $f_C(PS)$ respectively, where 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. It

occurs when all tasks are performed sequentially in the project - one after another. Disregarding how many and how flexible resources are.

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

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

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. Note that c_{max} and c_{min} do not respect skill constraints. It means that c_{min} value could be reached also for non–feasible solution, analogously to c_{max}.

To get feasible schedule some constrains must be provided, as follows:

$$\forall_{k \in K} s_k \geq 0, \forall_{k \in K} Q^k \neq \emptyset \tag{4}$$

$$\forall_{j \in J} F_j \geq 0; \forall_{j \in J} d_j \geq 0 \tag{5}$$

$$\forall_{j \in J, j \neq 1, i \in P_j} F_i \leq F_j - d_j \tag{6}$$

$$\forall_{i \in J^k} \exists_{q \in Q^k} h_q = h_{q_i} \wedge l_q \geq l_{q_i} \tag{7}$$

$$\forall_{k \in K} \forall_{t \in \tau} \sum_{i=1}^{n} U_{i,k}^t \leq 1 \tag{8}$$

$$\forall_{j \in J} \exists_{!t \in \tau, !k \in K} U_{j,k}^t = 1 \tag{9}$$

The first constraint (see Eq. 4) preserves the positive values of resource salaries and ensures that every resource has no-empty set of skills. Eq. 5 states that every task has positive finish date and duration, while Eq. 6 shows the precedence constrains rule. Next two equations: Eq. 7 introduces skill constraints and transforms RCPSP into MS–RCPSP. Constraint given in Eq. 8 describes that resource can be assigned to no more than one task at given time during the project. The last constraint (see Eq. 9) says that each task must be performed in schedule PS by one resource assignment.

The proposed MS–RCPSP allows to define problem as multiobjective [5] (see Eq. 1): duration– and cost– oriented one. One criteria has to trade off certain other criteria because cheaper schedule is mostly longer in realization. Such problem can be solved as a weighted linear combination, as follows:

Evaluation function is formulated as follows:

$$\min f(PS) = w_\tau f_\tau(PS) + (1 - w_\tau) f_c(PS) \tag{10}$$

where: w_τ – it is weight of duration component and has non–negative values: $w_\tau \in [0; 1]$. Such definition makes possible 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 ($1 - \omega_\tau$) aspect. It means that setting the weight of duration aspect to 1.0 automatically sets the weight of cost to 0.0 and vice versa. Specifically, both weights can 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, $\omega_\tau = 1$) [7], [6], balanced optimization (BO, $\omega_\tau = 0.5$) and cost optimization (CO, $\omega_\tau = 0$) [8]. As CO is rather a trivial task that can be solved by greedy–based approach, BO can be analyzed as cost/duration middle ground. In this paper we focus only on the DO – it means that we minimize only schedule makespan. MS–RCPSP reduced to duration–oriented optimization can be considered as a variation of widely studied parallel machine scheduling problem with minimum makespan objective.

As MS–RCPSP is combinatorial NP–hard problem the estimation of the total solution space (feasible and non–feasible solutions included) size (SS) can be estimated as follows:

$$SS(n, m) = n! * m^n \tag{11}$$

Computing factorial of tasks number provides the number of combinations of ordering tasks within the timeline. It is easy to notice that such estimation allows setting any order, skipping precedence constraints. The second element of Eq. 11 provides the number of resource–to–task assignments, including situation that the same resource is assigned to all tasks and no skill constraints are preserved (non–feasible solution). To show the size of solution space, let's consider the 'simple' project schedule with 100 tasks and 20 resources – it gives $SS(100, 20) = 1.19 * 10^{288}$ all possible solutions.

III. Proposed EA–based approaches

In this section four EA–based approaches to MS–RCPSP have been presented. Approaches differ in genome interpretation and schedule (as phenotype) build method. Methods of initialization, representation, crossover, mutation, fitness function, selection are common for all. In each approach to MS–RCPSP sequential representation (vector) of the genome has been implemented. Such representation is similar to classical TSP (Travelling Salesman Problem), e.g. vector <3,2,1,4> can be represented as the priority for resource or tasks (depends on approach). Such representation allows us to use TSP standard operators: swap as mutation and one–point crossover. Moreover, we use tournament selection and random initialization.

The fitness function is the crucial procedure – e.g. if EA gives an individual priorities for resources, tasks sequence should be proposed by procedure – a schedule builder is used to generate the final schedule. If EA individual gives priorities for tasks and resources the Schedule Generator Scheme (**SGS** – details in III-A) works. In approach EA_T (or EA_R) where resources (tasks) priorities should be proposed, greedy–based algorithm is used selecting first fit element. Mostly, four EA approaches differ in the genome interpretation and schedule build method as follows:

In the **EA_R** the individual consists of priority for each resource. To evaluate genome SGS builds schedule using greedy approach. Another approach, in **EA_T** the individual consists of priority for each task. In evaluation procedure the greedy builds schedule. The approach that links such two methods is **EA_RT** – an individual comprises two parts: priority resource vector and task priority vector. To keep priorities the final schedule is generated by SGS procedure.

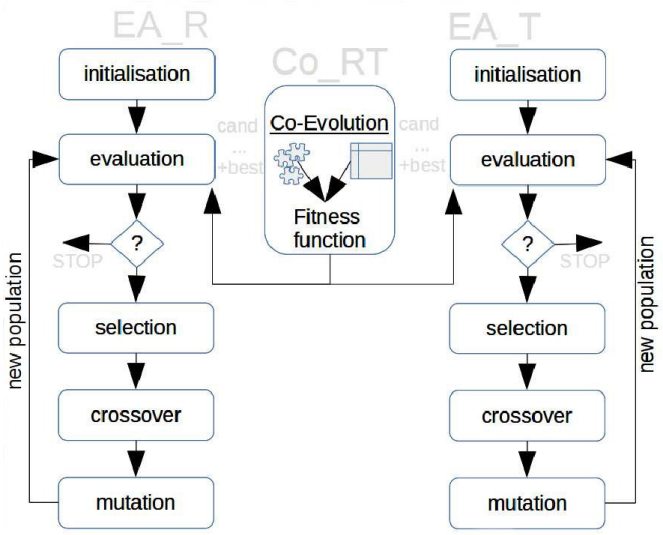

Fig. 2: Evolutionary Algorithms for MS–RCPSP: EA_T, EA_R and CO_RT.

Such approach (EA_RT) makes that genome "doubled" in size (consists of priorities for tasks and resources), the solution landscape is enlarged and fitness function values may differ significantly for "similar" genomes. Thus, the **Co_RT** approach links "good" sequence of tasks priorities to resources using natural co-evolution mechanism: in one population evolution processes resources' priorities, the second one consists of priorities for tasks. Schematically Co_RT method is presented on Fig. 2. There, two populations are linked by fitness function: to build final schedule, priorities for tasks and resources are needed to run SGS procedure. For every individual, several complementary individuals are selected (it is Co_RT parameter) from the other population to build schedule – the best-gained value of the fitness function is given. Additionally, to keep Co_RT more stable for every individual schedule is built using complementary component (resource/tasks sequences) from the best last generation solution.

A. Schedule Generator Scheme

To evaluate an individual in EA approaches a procedure that converts genotype to schedule Schedule Generator Scheme (SGS) is needed. Such procedure must deal with three types of individuals, that includes (1) only task priority (EA_T) (2) only resources priority (EA_R) and (3) both tasks priority and resource priority (in EA_RT and CO_RT). In Pseudocode 1 such procedure has been presented schematically. As an argument, it takes priorities (task and/or resources) and returns final schedule which can be then evaluated. If SGS has not been provided T_pri (task priorities sequence), it gets default sequence that is copied from instance definition. In the case of empty resource priorities (R_pri) EA algorithm generates randomly default sequence that is used in the evolution process.

Listing 1: Pseudocode of Schedule Generator Scheme (SGS).

```
Schedule SGS_procedure ( T_pri, R_ pri )
T_seq := Tasks
R_seq := Resources

while T_seq!=null
  if (T_pri!=null)
  Task := max(T_seq.CanBeDone(), Priority)
  else  Task := T_seq.CanBeDone().first()

  if (R_pri!=null)
  R := max( R_seq.getCapable(), Priority)
  else  R: = R_seq.firstCapable()

  TimeStamp := R.end()
  Schedule.assign( Task, R, Timestamp )
  R.end := task.start + task.duration
  T_seq = T_seq / Task
end // while
return schedule.
```

The SGS procedure keeps the task sequence from T_pri if it is possible. However, to satisfy the tasks precedence constraints, in each case the $CanBeDone()$ method is executed. The same situation occurs in resource selection procedure – it is selected resource that is capable of given task realization and has the highest priority. If SGS doesn't have prioritized resources/tasks it works like greedy–like algorithm - takes first–fit element.

IV. Experiments and results

This section describes experiments that have been done to empirically verify several research questions:

Q1. Does the Greedy algorithm guided by metaheuristic has a tendency to stuck in local optima?
Q2. Is the priority–based sequence vector representation effective?
Q3. Which priority–based approach using greedy–based SGS (EA_T or EA_R), is more effective?
Q4. How effective is combined approach EA_RT that eliminates greedy but enlarges the solution landscape?
Q5. How effective, in comparison to above methods, is co–evolutionary approach that eliminates Greedy usage and keeps standard size of solution landscape?

Above research questions should be answered empirically using the experimental procedure.

A. Experiments' procedure

In experiments iMOPSE benchmark dataset [8] is used – a part of iMOPSE project [1]. Dataset published on the Internet consists of 36 MS–RCPSP iMOPSE instances that differ in number of tasks, resources, skills and relations. The iMOPSE dataset is a part of iMOPSE library supported by instance

[1]iMOPSE project homepage: http://imopse.ii.pwr.wroc.pl/ . The best schedules generated by EA_R, EA_T, EA_RT and Co_RT have been published there.

generator, validator, visualization tools and provided use–cases with published java codes (more in [5]).

For empirical comparison of methods results, each method has been tunned (see configurations in Tab. I). However, for each method, the number of births has been reduced to 20,000. It is worth noticing that the Co_RT method uses multiple candidate resource–task assignment ($cand\_assign$ value usually equals to 3) to get better fitness functions. Such method can be treated as some local search procedure that causes Baldwin Effect. However, it doesn't cause strict new solution generation and has no influence on genes. That's why the number of births in Co_RT is reduced to 20,000 but taking into account number of fitness function calculation such value is bigger than the limit.

Each experiment has been repeated 30 times, and results are averaged, and standard deviation value is given. Comparison of gained results has been statistically examined using Wilcoxon signed–rank test.

B. Experiments results

Each method is ran 30 times to generate solutions for given problem instances. Data in Tab. II presents results of several proposed methods: EA_R, EA_T, EA_RT and Co_RT. As reference method hybrid Differential Evolution and Greedy (DEGR) [6] is given in two configurations: using DEGR(pop_size=200, $generations$=500) and DEGR(pop_size=200, $generations$=10,000). The first configuration satisfies the condition of 20,000 births, but in publication [6] the second is given as the best found.

Data presented in Tab. II shows that the best examined method is Co_RT because sum of all generated (average) schedules equals to 11,639 (sum of std_dev=43.35). However, the second place took method EA_T where all averaged schedules least 11,681 (sum of std_dev=45.52). It means that the Co_RT gives solutions 0.35% better than EA_T – this slight improvement is statistically significant: the Wilcoxon signed–rank test proves it ($W_{0.05}$=443 $>$ W_c=208). It is worth to mention that Co_RT outperforms other methods giving in four cases the best-found solutions for instances: 200_40_45_9, 200_40_133_15, 200_10_135_9_D6 and 200_40_90_9. For these instances, we investigated the evolution process. For instance 200_40_133_15 (see Fig.3) the evolution process searches effectively for EA_T and CO_RT, but gets stuck very fast for EA_T. The similar situation is in instances 200_10_135_9_D6 (see Fig.4), where CO_RT outperforms other approaches giving solution very fast, EA_T and EA_RT need more time to reach a similar solution. The case of 200_40_45_9 (see Fig.5) instance shows that CO_RT works the most effectively and other methods cannot compete. Quite similar situation occurs on Fig.6 (instance 200_40_90_9), where CO_RT outperforms other methods, but EA_T gives near solutions.

Using DEGR method in configuration DEGR(pop_size=200, $generations$=500) is not competitive to Co_RT, therefore we selected as reference DEGR(pop_size=200, $generations$=10,000) configuration that gives better results. It can be noticed that in seven instances DEGR gives better solution than Co_RT and in 9 other cases solutions are similar. But summarized duration of (average) schedules last 11,971 (sum of std_dev=183.15) which means that CO_RT gives 2.78% of improvement and Wilcoxon signed–rank test results verified positively such difference ($W_{0.05}$=477 $>$ W_c=208). Moreover, the std_dev values show that Co_RT is more stable method than DEGR – DEGR std_dev=183.15 versus CO_RT std_dev=43.35.

Results presented in Tab. II show that the worst results are given by EA_R, where genome proposes priorities for resources and tasks are selected by greedy–like method. The chromosome extension by task priorities (EA_RT) makes that method return shorter schedules by 8.9% than EA_R. However, the standard deviation values are higher – EA_R std_dev=36.98 versus EA_RT std_dev=84.76.

All the best-found schedules generated by EA_R, EA_T, EA_RT and Co_RT have been published on iMOPSE project homepage.

C. Summary

Results of experiments presented in Tab. II showed that the best-examined method is Co_RT giving several of the best-found solutions. But the results of other methods are very valuable because they help to answer research questions asked at the beginning of this section.

The first question ($Q1$) cannot be answered easily because results of two approaches that use greedy (EA_T and EA_R) compared to EA_RT results shows that task priorities are more important and greedy gives effective solutions that can compete with Co_RT results. However, EA_R is less effective than EA_T, which answers another research ($Q3$) question.

Another question ($Q2$) concerns how effective is the priority–based sequence vector representation. All proposed approaches that use it are compared to DEGR vector with float values representation. Presented results show that such sequence representation is easy in implementation and can compete with more complex used in DEGR.

The answer to question ($Q4$) only apparently is simple, because EA_T gives better solutions than EA_RT. However, provided limit of births (20,000) reduces EA_RT "space" for evolution process. A quite large standard deviation (std_dev=84.76) value confirms this fact. For EA_T such value equals to std_dev=45.52.

The last question is the most important aspect of the paper. Is co–evolutionary (Co_RT) approach to MS–RCPSP effective? This question ($Q5$) is answered positively, and several arguments are presented in this section. Co_RT not only outperforms other tested methods but also gives the best-known solutions for four instances.

V. Conclusions and future work

This paper concerns if Co–evolutionary algorithms are effective for solving combinatorial NP–hard problems, MS–RCPSP. Gained results showed that problem decomposition to resource and task assignment using co-evolutionary mechanism is a powerful idea. As reference,

TABLE I: Methods' configurations

	pop_size	*generations*	P_m	P_x	*selection*	*cand_assign*
EA_R, EA_T	660	300	0.02	0.8	tournament 10%	-
EA_RT	660	300	0.005	0.2	tournament 10%	-
Co_RT	500	200	0.02	0.8	tournament 10%	3

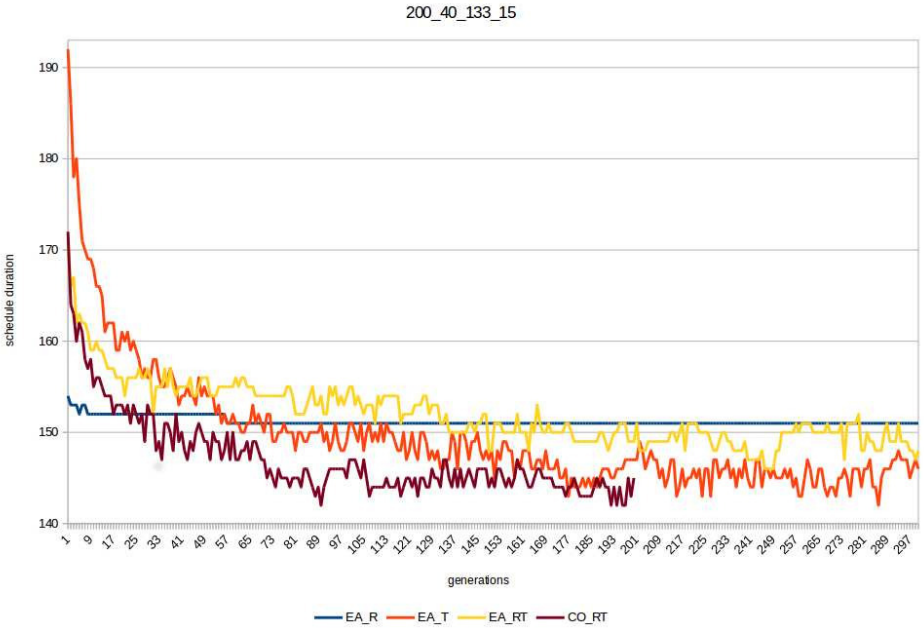

Fig. 3: Example of evolution process for MS–RCPSP: EA_T, EA_R, EA_RT and CO_RT.

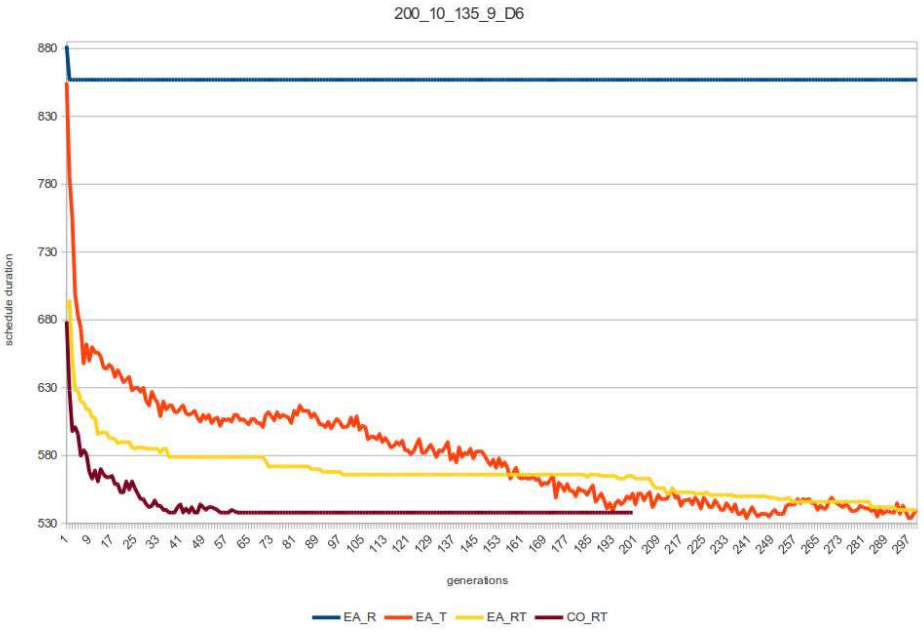

Fig. 4: Example of evolution process for MS–RCPSP: EA_T, EA_R, EA_RT and CO_RT.

results of evolutionary algorithms using the same representation have been compared. Moreover, the reference method DEGR [6] results also confirm the dominance of Co_RT.

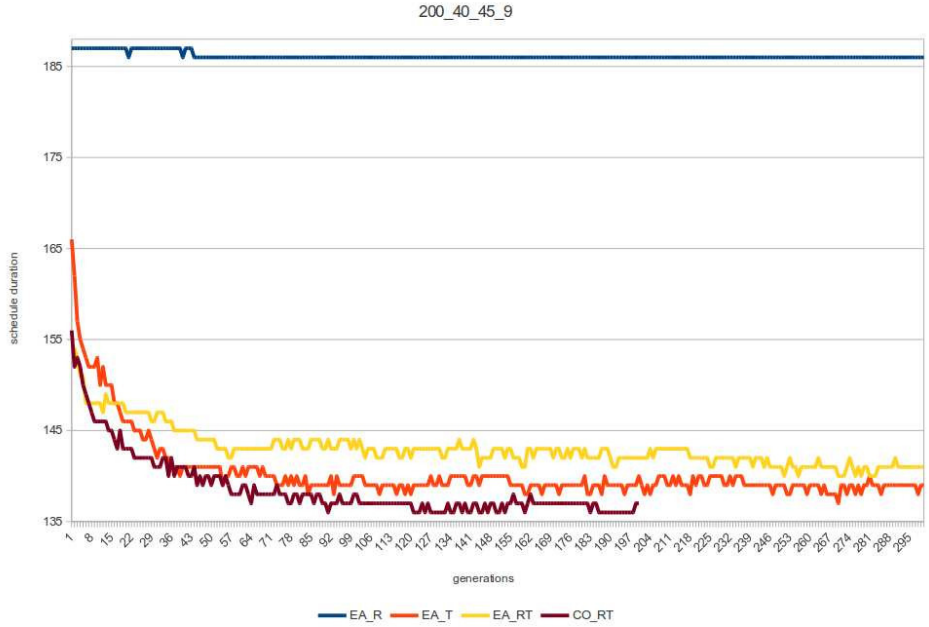

Fig. 5: Example of evolution process for MS–RCPSP: EA_T, EA_R, EA_RT and CO_RT.

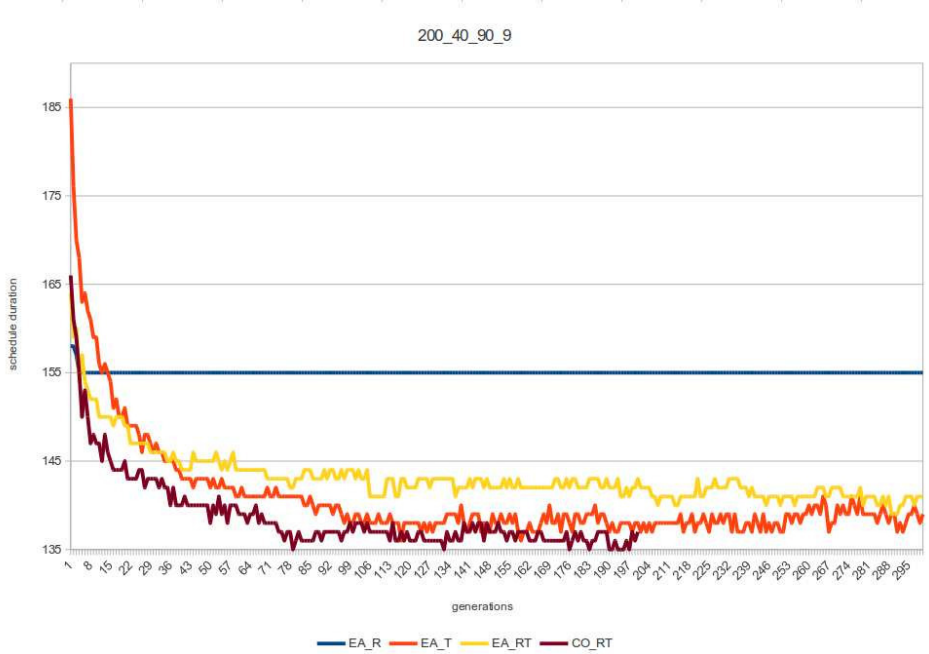

Fig. 6: Example of evolution process for MS–RCPSP: EA_T, EA_R, EA_RT and CO_RT.

In the paper, several research questions have been answered, but there are still many open issues. Proposed approaches use simple representation that is not specialized to MS–RCPSP – e.g. how effective can be the approach that uses specialized crossover/mutation operator? In co-evolution, we can see the large potential, and future work should be connected with it: more experiments with parametrization of Co_RT (without 20,000 births limit), various selection method and scalability. Moreover, the very promising direction is effective clone reduction method, a flexible size of population and adaptation mechanism. And the last but not least, as MS–RCPSP problem is bi-objective, we want to extend proposed co-evolutionary approach to multicriteria optimization.

TABLE II: Comparison results of evolutionary approaches: EA_R, EA_T, EA_RT, Co_RT and reference DEGR [6]

instance	EA_R		EA_T		EA_RT		Co_RT		DEGR gen=500		DEGR gen=10,000	
	avg	std_dev	avg	std_dev	avg	std_dev	avg	std_dev	avg	std_dev	avg	std_dev
100_5_20_9_D3	437.00	0.00	**387.13**	0.34	388.37	1.82	**387.07**	0.25	398.70	5.77	393.20	0.92
100_5_22_15	515.00	0.00	**484.57**	0.50	**484.63**	0.48	**484.63**	0.48	486.80	2.10	**484.50**	0.53
100_5_46_15	616.00	0.00	**529.80**	2.14	531.97	3.05	**529.7**	1.71	534.10	3.57	**529.00**	0.00
100_5_48_9	538.00	0.00	491.17	0.90	491.60	2.12	491.00	0.68	492.80	2.15	**490.10**	0.32
100_5_64_15	550.00	0.00	**482.80**	1.62	484.27	2.46	**482.50**	1.48	487.60	2.84	483.00	0.82
100_10_26_15	264.43	1.67	**235.07**	0.77	236.13	1.80	**235.07**	0.96	244.80	5.37	**235.00**	1.05
100_10_27_9_D2	265.00	2.00	211.07	1.69	216.73	2.42	**209.87**	1.52	238.80	4.32	220.30	2.50
100_10_47_9	272.57	0.88	**254.43**	1.05	260.73	2.72	**254.90**	0.91	259.30	2.11	256.40	0.70
100_10_48_15	261.53	0.67	246.57	1.31	251.63	2.75	247.00	1.46	251.70	3.20	**245.00**	0.67
100_10_64_9	274.33	1.64	**244.97**	1.45	255.37	4.20	246.00	2.03	256.30	2.50	245.80	1.32
100_5_64_9	532.00	0.00	476.63	1.02	477.93	3.16	477.03	1.52	477.40	0.97	**474.90**	0.32
100_20_46_15	197.00	0.00	**161.00**	0.00	**161.00**	0.00	**161.00**	0.00	165.30	3.43	164.00	0.00
100_20_47_9	149.30	1.07	**126.63**	1.20	133.07	1.79	**126.30**	0.94	141.50	4.01	127.50	3.31
200_40_45_9	185.63	0.80	137.53	0.76	140.43	0.96	**136.20***	1.05	177.90	5.45	182.50	17.83
200_40_133_15	150.93	0.93	145.07	1.57	148.13	2.32	**141.77***	1.20	166.90	7.28	151.40	8.26
100_10_65_15	263.60	1.23	247.77	1.52	256.50	4.99	248.50	1.95	252.90	2.64	**245.30**	1.16
100_20_22_15	149.90	1.42	**128.13**	0.67	131.03	1.47	**128.43**	0.99	141.40	4.20	130.70	0.67
100_20_23_9_D1	199.00	1.86	**172.00**	0.00	**172.00**	0.00	**172.00**	0.00	**172.00**	0.00	**172.00**	0.00
100_20_65_9	142.43	1.61	**125.97**	1.14	135.37	2.74	**125.77**	1.02	145.30	2.21	129.10	2.73
100_20_65_15	233.93	1.44	**205.00**	0.00	**205.00**	0.00	**205.00**	0.00	240.00	0.00	240.00	0.00
200_10_50_9	494.10	0.65	**486.57**	0.56	488.37	1.38	**486.80**	0.79	497.30	2.83	487.80	1.62
200_10_50_15	506.37	0.84	**487.13**	0.62	489.53	1.38	**487.23**	1.20	492.70	3.09	487.90	0.74
200_10_84_9	535.00	1.10	**509.60**	1.28	514.20	2.41	**509.70**	1.27	520.60	2.55	**509.30**	2.11
200_10_85_15	498.53	1.48	481.13	2.32	484.90	3.08	479.47	2.81	484.30	3.43	**478.00**	1.56
200_10_128_15	488.00	0.00	472.50	3.03	479.90	5.87	471.40	2.82	466.10	2.23	**463.10***	0.88
200_10_135_9_D6	860.53	7.43	553.00	10.37	549.70	17.66	**535.57***	6.11	829.20	40.51	694.80	67.90
200_20_54_15	274.03	1.14	**261.90**	1.40	265.20	1.38	**261.50**	1.28	276.80	4.80	**261.00**	1.89
200_20_55_9	264.73	0.81	248.30	0.64	250.73	1.29	**247.87**	0.85	270.40	3.17	257.80	10.37
200_20_97_9	268.23	0.92	246.90	1.42	251.07	2.34	**245.93**	1.48	267.80	8.99	247.60	8.93
200_20_97_15	**336.00**	0.00	**336.00**	0.00	**336.00**	0.00	**336.00**	0.00	**336.00**	0.00	**336.00**	0.00
200_20_145_15	264.83	1.19	248.30	1.68	253.80	3.25	247.37	2.07	256.10	4.18	**238.50**	0.71
200_20_150_9_D5	**900.00**	0.00	**900.00**	0.00	**900.00**	0.00	**900.00**	0.00	926.80	24.12	906.90	11.82
200_40_45_15	217.07	1.67	**159.00**	0.00	**159.00**	0.00	**159.00**	0.00	164.00	0.00	164.00	0.00
200_40_90_9	153.90	1.16	138.30	0.97	142.37	1.96	**135.00***	1.39	173.40	8.14	181.30	22.07
200_40_91_15	157.93	1.39	136.20	1.60	139.70	1.51	**133.77**	1.12	160.60	6.42	144.80	9.44
200_40_130_9_D4	**513.00**	0.00	**513.00**	0.00	**513.00**	0.00	**513.00**	0.00	**513.00**	0.00	**513.00**	0.00
sum	12929	36.98	**11671**	**45.52**	11779	84.76	**11639**	**43.35**	12366	178.58	11971	183.15
avg	359.16	1.03	**324.20**	**1.26**	327.20	2.35	**323.31**	**1.20**	343.52	4.96	332.54	5.09

REFERENCES

[1] Bianco L., Dell Olmo P., Speranza M.G.; Heuristics for multimode scheduling problems with dedicated resources, Eu. J. of Oper. Research (107), pp. 260–271, 1998.

[2] Błazewicz J., Lenstra J.K., Rinnooy Kan A. H. G.; Scheduling subject to resource constraints: Classification and complexity, Discr. App. Math. (5), pp. 11–24, 1983.

[3] Hartmann S., Briskorn D.; A survey of variants and extensions of the resource–constrained project scheduling problem, Eur. J. of Oper. Res. (207), pp. 1–14, 2010.

[4] Hartmann S., Kolisch R.; Experimental investigation of heuristics for resource–constrained project scheduling: An update, Eur. J. of Oper. Res. (174), pp. 23–37, 2006.

[5] Myszkowski P.B., Laszczyk M., Nikulin I. and Skowroński M.E. "iMOPSE: a library for bicriteria optimization in Multi–Skill Resource–Constrained Project Scheduling Problem", *in review process*, Soft Computing Journal.

[6] Myszkowski P.B., Olech L.P., Laszczyk M. and Skowroński M.E. "Hybrid Differential Evolution and Greedy (DEGR) for Solving Multi–Skill Resource–Constrained Project Scheduling Problem", *in review process*, Applied Soft Computing Journal.

[7] Myszkowski P.B., Siemieński J.J., "GRASP applied to Multi-Skill Resource-Constrained Project Scheduling Problem", Computational Collective Intelligence, Volume 9875 of the series Lecture Notes in Computer Science pp.402-411, 2016.

[8] Myszkowski P.B., Skowroński M., Sikora K., "A new benchmark dataset for Multi-Skill Resource-Constrained Project Scheduling Problem", Proc. of FedCSIS Conference (2015), M. Ganzha, L. Maciaszek, M. Paprzycki (eds). ACSIS, Vol. 5, pages 129–138 (2015)

[9] Myszkowski P.B., Skowroński M., Olech L., Oślizło K. "Hybrid Ant Colony Optimization in solving Multi–Skill Resource–Constrained Project Scheduling Problem", Soft Computing Journal, 2015, Volume 19, Issue 12, pp.3599–3619.

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

[11] Skowroński M., 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 FedCSIS Conference (2013), pp. 153-158, 2013.

[12] Skowroński M., Myszkowski P.B., Podlodowski L., "Novel heuristic solutions for Multi–Skill Resource–Constrained Project Scheduling Problem", Annals of Computer Science and Information Systems Volume 1, Proc. of FedCSIS Conference (2013), pp. 159-166, 2013.

[13] Zheng, H., Wang, L. and Zheng, X., Teaching–learning-based optimization algorithm for multi–skill resource constrained project scheduling problem, Soft Comput (2017) 21: 1537.

[14] Yan R., Li. W., Jiang P., Zhou Y., Wu G.; A Modified Differential Evolution Algorithm for Resource Constrained Multi-project Scheduling Problem, Journal of Computers, Vol. 9, No. 8, pp. 1922–1927, 2014.

[15] H. y. Zheng, L. Wang and S. y. Wang, "A co-evolutionary teaching-learning-based optimization algorithm for stochastic RCPSP," 2014 IEEE Congress on Evolutionary Computation (CEC), Beijing, 2014, pp. 587-594.