Journal of Universal Computer Science, vol. 22, no. 6 (2016), 856-873
submitted: 30/11/15, accepted: 28/5/16, appeared: 1/6/16 © J.UCS

PLA Based Strategy for Solving RCPSP by a Team of Agents

Piotr Jędrzejowicz
(Chair of Information Systems
Gdynia Maritime University
Morska 83, 81-225 Gdynia, Poland
pj@am.gdynia.pl)

Ewa Ratajczak-Ropel
(Chair of Information Systems
Gdynia Maritime University
Morska 83, 81-225 Gdynia, Poland
ewra@am.gdynia.pl)

Abstract: In this paper the dynamic interaction strategy based on the Population Learning Algorithm (PLA) for the A-Team solving the Resource-Constrained Project Scheduling Problem (RCPSP) is proposed and experimentally validated. The RCPSP belongs to the NP-hard problem class. To solve this problem a team of asynchronous agents (A-Team) has been implemented using multiagent system. An A-Team is the set of objects including multiple agents and the common memory which through interactions produce solutions of optimization problems. These interactions are usually managed by some static strategy. In this paper the dynamic learning strategy based on PLA is suggested. The proposed strategy supervises interactions between optimization agents and the common memory. To validate the proposed approach computational experiment has been carried out.

Key Words: Resource-Constrained Project Scheduling, RCPSP, Optimization, Agent, A-Team, Population Learning Algorithm, PLA
Category: H.1, H.4

1 Introduction

The Resource Constrained Project Scheduling Problem (RCPSP) has attracted a lot of attention and many exact, heuristic and metaheuristic solution methods have been proposed in the literature in recent years [Hartmann, 06], [Agarwal, 11], [Paraskevopoulos, 12], [Fang, 12]. The current approaches to solve these problems produce either approximate solutions or can only be applied for solving instances of the limited size. Hence, searching for more effective algorithms and solutions to the problems is still a lively field of research. One of the promising directions of such research is to take advantage of the parallel and distributed computation solutions, which are features of the contemporary multiagent systems [Wooldridge, 09].

Modern multiagent system architectures are an important and intensively expanding area of research and development. There is a number of multiple-agent approaches proposed to solve different types of optimization problems. One of them is the concept of an A-Team, originally introduced in [Talukdar, 96]. The idea of the A-Team was used to develop the software environment for solving a variety of computationally hard optimization problems called JABAT [Jędrzejowicz, 06], [Barbucha, 09]. JADE based A-Team (JABAT) system supports the construction of the dedicated A-Team architectures. Agents used in JABAT assure decentralization of computation across multiple hardware platforms. Parallel processing results in more effective use of the available resources and ultimately, a reduction of the computation time.

Population Learning Algorithm (PLA) proposed in [Jędrzejowicz, 99] is a population-based method inspired by analogies to the social education systems in which a diminishing number of individuals enter more and more advanced learning and improvement stages. PLA divides the process of solving the problem into stages, in which the considered optimization problem is solved using a set of independent learning/improvement procedures.

A-Team is a system composed of the set of objects including multiple agents and the common memory which through interactions produce solutions of optimization problems. Several strategies controlling the interactions between agents and memories have been recently proposed and experimentally validated. The influence of such interaction strategy on the A-Team performance was investigated in [Barbucha, 10]. In [Jędrzejowicz, 14], [Jędrzejowicz, 15] the dynamic interaction strategy based on Reinforcement Learning for A-Team solving the RCPSP and MRCPSP has been proposed. The similar topics were also considered for different multi-agent systems, e.g. [Pelta, 06], [Cadenas, 09].

In this paper the PLA based dynamic interaction strategy for the A-Team solving the RCPSP is proposed and experimentally validated. PLA strategy is used to control the parameters and manage the process of searching for solutions to the RCPSP instances by a team of agents. In this approach the parameters depend on the current state of the environment and the stage of learning. Both are being changed dynamically during the computation.

The proposed A-Team produces solutions to the RCPSP instances using four kinds of the optimization agents. They include simple local search, tabu search metaheuristic, crossover search and path relinking procedures.

The paper is constructed as follows: Section 2 contains the RCPSP formulation. Section 3 gives some information on JABAT environment. Section 4 contains the general idea of the PLA construction. Section 5 provides details of the proposed PLA dynamic interaction strategy and its implementation in JABAT. Section 6 describes settings of the computational experiment carried-out with a view to validate the proposed approach and contains a discussion of the computational experiment results. Finally, Section 7 contains conclusions and suggestions for future research.

2 Problem Formulation

A single-mode resource-constrained project scheduling problem (RCPSP) consists of a set of n activities, where each activity has to be processed without interruption to complete the project. The dummy activities 1 and n represent the beginning and the end of the project. The duration of an activity j, $j = 1, \dots, n$ is denoted by d_j where $d_1 = d_n = 0$. There are r renewable resource types. The availability of each resource type k in each time period is r_k units, $k = 1, \dots, r$. Each activity j requires r_{jk} units of resource k during each period of its duration, where $r_{1k} = r_{nk} = 0$, $k = 1, ..., r$. All parameters are non-negative integers. There are precedence relations of the finish-start type with a zero parameter value (i.e. $FS = 0$) defined between the activities. In other words activity i precedes activity j if j cannot start until i has been completed. The structure of a project can be represented by an activity-on-node network $G = (SV, SA)$, where SV is the set of activities and SA is the set of precedence relationships. SS_j (SP_j) is the set of successors (predecessors) of activity j, $j = 1, \dots, n$. It is further assumed that $1 \in SP_j$, $j = 2, \dots, n$, and $n \in SS_j$, $j = 1, \dots, n-1$. The objective is to find a schedule S of activities starting times $[s_1, \dots, s_n]$, where $s_1 = 0$ and resource constraints are satisfied, such that the schedule duration $T(S) = s_n$ is minimized.

The objective is to find a minimal schedule in respect of the makespan that meets the constraints imposed by the precedence relations and the limited resource availabilities.

The above formulated problem as a generalization of the classical job shop scheduling problem belongs to the class of NP-hard optimization problems, see [Błażewicz, 83]. The considered problem class is denoted as $PS|prec|C_{max}$ [Brucker, 99] or $m, 1|cpm|C_{max}$ [Demeulemeester, 02].

3 The JABAT Environment

JABAT is the software environment facilitating the design and implementation of the A-Team architecture for solving various combinatorial optimization problems. The problem-solving paradigm on which the proposed system is based can be best defined as the population-based approach.

JABAT produces solutions to combinatorial optimization problems using a set of optimization agents, each representing an improvement algorithm. Each improvement (optimization) algorithm when supplied with a potential solution to the problem at hand, tries to improve this solution. The initial population of solutions (individuals) is generated or constructed. Individuals forming the initial population are, at the following computation stages, improved by independently acting optimization agents. The main functionality of the proposed environment includes organizing and conducting the process of search for the best solution.

The behavior of the A-Team is controlled by the, so called, interaction strategy defined by the user. An A-Team uses a population of individuals (solutions) and a number of optimization agents. All optimization agents within the A-Team work together to improve individuals from their population in accordance with the interaction strategy.

Important classes in JABAT include TaskManager and PlatformManager which are used to initialize the agents and maintain the system. Objects of these classes also act as agents:

TaskManager - is responsible for initializing the process of solving the problem instance. It creates other agents (e.g. PlatformManager, SolutionManager) and reads all system and data parameters needed.

PlatformManager - organizing the process of migrations between different platforms. It creates copies of agents.

To adapt the proposed architecture for solving the particular problem, the following classes of agents need to be designed and implemented:

SolutionManager - represents and manages the process of solving the problem instance by the A-Team e.g. the set of optimization agents and the population of solutions stored in the common memory.

OptiAgent - represents a single improving algorithm (e.g. local search, simulated annealing, genetic algorithm etc.).

To describe the problem Task class representing the instance of the problem and Solution class representing the solution is used. Classes describing the problem are responsible for reading the data, preprocessing the data and generating random instances of the problem. Additionally, an interaction strategy based on the PLA is used to managing and maintaining a population of current solutions in the common memory, as described in section 4.

JABAT has been designed and implemented using JADE (Java Agent Development), which is a software framework proposed by TILAB [Bellifemine, 03] supporting the implementation of the multiagent systems. More detailed information about the JABAT environment and its implementations can be found in [Jędrzejowicz, 06], [Barbucha, 09], [Barbucha, 11].

4 Population Learning Algorithm (PLA)

Population Learning Algorithm introduced originally in [Jędrzejowicz, 99] is a population-based method inspired by analogies to a phenomenon of social education processes in which a diminishing number of individuals enter more and more advanced learning stages. PLA take advantage of the following features common to organized education systems:

- A huge number of individuals enter the system.
- Individuals learn through organized tuition, interaction, self-study, trials and errors.
- Learning process is inherently parallel (different schools, curricula, teachers, etc.).
- Learning process is divided into stages.
- More advanced stages are entered by a diminishing number of individuals from the initial population.
- At higher stages more advanced learning and improvement techniques are used.
- A final stage is reached by only a fraction of the initial population.

In PLA an individual represents a coded solution, or a part of it, of the considered problem. Initially, a number of individuals, known as the initial population, is randomly generated or constructed using some construction heuristics. Once the initial population has been generated, individuals enter the first learning stage. It involves applying some, possibly basic and elementary, improvement schemes. These can be based, for example, on some local search procedures. The improved individuals are then evaluated and better ones pass to subsequent stages. A strategy of selecting better or more promising individuals at each stage must be defined and duly applied. At following stages the whole cycle is repeated. Individuals are subject to improvement and learning, either individually or through information exchange, and the selected ones are again promoted to a higher stage with the remaining ones dropped-out from the process. At the final stage the remaining individuals are reviewed and the best one represents a solution to the problem at hand.

5 A-Team with the Dynamic Interaction Strategy Controlled by PLA

JABAT was successfully used by the authors for solving RCPSP and MRCPSP (see [Jędrzejowicz, 08]) as well as RCPSP/max and MRCPSP/max (see [Jędrzejowicz, 09]), where static interaction strategies have been used.

In [Jędrzejowicz, 14], [Jędrzejowicz, 15] the dynamic interaction strategies based on Reinforcement Learning have been proposed and successfully used. In this approach the dynamic interaction strategy based on PLA is proposed.

To adapt JABAT to solving RCPSP the sets of classes and agents were implemented. The first set includes classes describing the problem. They are responsible for reading and preprocessing of the data and generating random instances of the problem. The set includes:

RCPSPTask inheriting from the Task class and representing the instance of the problem,

RCPSPSolution inheriting from the Solution class and representing the solution of the problem instance,

Activity representing the activity of the problem,

Resource representing the renewable or nonrenewable resource.

The second set includes classes describing the optimization agents. Each of them includes the implementation of an optimization algorithm used to solve the RCPSP. All of them are inheriting from the OptiAgent class. Optimization agents are implementations of specialized algorithms: CA, PRA, LSAm, LSAe, TSAm and TSAe described below. The prefix Opti is assigned to each agent with its embedded algorithm:

OptiCA - implementing the Crossover Algorithm (CA),

OptiPRA - implementing the Path Relinking Algorithm (PRA).

OptiLSAm - implementing the Local Search Algorithm (LSAm),

OptiLSAe - implementing the Local Search Algorithm (LSAe),

OptiTSAm - implementing the Tabu Search Algorithm (TSAm),

OptiTSAe - implementing the Tabu Search Algorithm (TSAe),

The CA is an algorithm based on the idea of the one point crossover operator. For a pair of solutions one point crossover is applied. The additional parameter determines the frequency the operation is performed. The best schedule is remembered and finally returned.

The PRA is an implementation of the path-relinking algorithm. For a pair of solutions a path between them is constructed. The path consists of schedules obtained by carrying out a single move from the preceding schedule. The move is understood as moving one of the activities to a new position in the schedule. For each schedule in the path the value of the respective solution is checked. The best schedule is remembered and finally returned.

The LSAm and LSAe are a local search algorithms which find local optimum either by moving each activity to all possible places in the schedule (in LSAm) or by exchanging pairs of activities (in LSAe). For each combination of activities the value of possible solution is calculated. The best schedule is remembered and finally returned.

The TSAm and TSAe are implementations of the tabu search metaheuristic. TSAm finds local optimum by moving each activity to all possible places in the schedule. TSAe finds local optimum by exchanging each two activities in the schedule (the neighborhood of the schedule). The best move from the neighborhood of the solution, which is not tabu, is chosen and performed. The best schedule is remembered and finally returned.

In our earlier approaches to manage the interaction between agents and common memory the static strategies were used, including:

Basic - where the random solution from the population is read and sent to the optimization agent. Next, the solution sent by the agent replaces a random solution from the population.

Blocking1 - where additionally, all the solutions send to optimization agents are blocked. If the number of non blocked solutions is less than the number of solutions needed by optimization agent the blocked solutions are released.

Blocking2 - where randomly selected worse solution from the population is replaced by the one sent by the optimization agent. Moreover, one new solution is generated randomly every fixed number of iterations, which replaces the worst solution from the population.

The dynamic strategies considered in our previous approaches are based on the Reinforcement Learning (RL) where four reinforcement rules are considered:

RL1 - controlling the replacement of one individual from the population by other randomly generated one.

RL2 - controlling the method of selecting an individual for replacement.

RL3 - controlling the method of selecting a group of individuals from the population (individuals are grouped according to certain features).

RL4 - controlling the method of selecting an individual from the particular group for particular OptiAgent.

The best dynamic interaction strategies proposed in our previous approaches [Jędrzejowicz, 15] were based on the Blocking2 strategy and include:

RL123 - in which, RL1 and RL2 and RL3 are used.

RL1234 - in which, RL1 and RL2 and RL3 and RL4 are used.

The strategy proposed in this paper is based on the Blocking2 strategy where the PLA idea together with the reinforcement learning features have been combined. The basic features of the PLA based dynamic interaction strategy are as follows:

- All the individuals in the initial population of solutions are generated randomly and stored in the common memory.
- The individuals for improvement are selected from the common memory randomly and blocked which means that once selected individual (or individuals) cannot be selected again until the OptiAgent to which they have been sent returns the solution or the learning stage is finished.
- The returning individual, which represent the feasible solution, replaces its original version before the attempted improvement. It means that the solutions are blocked for the particular OptiAgent and the returning solution replaces the blocked one or the worst from the blocked one. If none of the solutions is worse, the random one is replaced. All solutions blocked for the considered OptiAgent are released and returned to the common memory.
- The new feasible solution is generated with fixed probability P_{mg} and replaces another one. The methods of generating and replacing solutions in the population are described below.
- For each level of learning the environment state is remembered and used in the learning scheme. This state includes: the best individual and the population average diversity, weights and probabilities. The state is calculated every fixed number of iterations $nITns$. To reduce the computation time, average diversity of the population is evaluated by comparison with the best solution only. Diversity of two solutions for the RCPSP problem is evaluated as the sum of differences between activities starting times in a project.
- Computations are divided into learning stages. In each stage different parameter settings, different set of optimization agents and different stopping criteria are used.

To describe the proposed PLA-based strategy the following notation will be used:

P - population of individuals;

avgdiv(P) - current average diversity of the population P;

$nITwi$ - number of iterations without an improvement of the goal function carried out by LSA and TSA which is used to stop their computation;

$nITns$ - number of iterations after which a new environment state is calculated;

$cTime$ - computation time (minutes);

nLS - number of learning stages in the PLA.

Additionally, two probability measures have been used:

p_{mg} - probability of selecting the method mg for selection of a new individual;

p_{mr} - probability of selecting the method mr for replacing an individual in the population;

There are four possible methods of generating a new individual:

mgr - randomly;

$mgrc$ - using one point crossover operator for two randomly chosen individuals;

mgb - random changes of the best individual in the population;

$mgbc$ - using crossover operator for two randomly chosen individuals from the five best individuals from the population.

The weight w_{mg} for each method is calculated, where $mg \in Mg$, $Mg = \{mgr, mgrc, m_{gb}, m_{gbc}\}$. The $wmgr$ and $wmgrc$ are increased where the population average diversity decreases and they are decreased in the opposite case. The w_{mgb} and w_{mgbc} are decreased where the population average diversity increases and they are increased in the opposite case. The probability of selecting the method mg is calculated as

$$p_{mg} = \frac{w_{mg}}{\sum_{mg \in Mg} w_{mg}} .$$

There are three methods of replacing an individual from the population by a new one:

mrr - new solution replaces the random one in the population;

mrw - new solution replaces the random worse one in the population;

mrt - new solution replaces the worst solution in the population.

Experiments show that replacing the worse and worst solution is beneficial to intensify exploitation while replacing the random one intensifies exploration. The weight w_{mr} for each method is calculated, where $mr \in Mr$, $Mr = \{mrr, mrw, mrt\}$. The w_{mrr} is increased where the population average diversity decreases and it is decreased in the opposite case. The w_{mrw} and w_{mrt}

is decreased where the population average diversity decreases and they are increased in the opposite case. The probability of selecting the method mr is calculated as

$$p_{mr} = \frac{w_{mr}}{\sum_{mr \in Mr} w_{mr}} .$$

The current environment state parameters are updated after any significant change: generating a new solution, receiving the solution from OptiAgent and replacing solution in the population. The update includes:

- set w_{mgr}, w_{mgrc}, w_{mgb}, w_{mgbc};
- set w_{mrr}, w_{mrw}, w_{mrt};
- remember the best solution;
- calculate the avgdiv(P) .

```
PLA_strategy
{
    generate the initial population $P$
    calculate environment state
    for($i$=0; $i < nLS$; $i = i + 1$)
    {
        while(none of the stopping criteria is met)
        {
            use OptiAgents to improve solutions
            generate a new solution $S_{new}$ with $p_{mg}$
            replace the individual in $P$ by $S_{new}$ with $p_{mr}$
            calculate environment state
        }
    }
}
```

Figure 1: General schema of the PLA based strategy

The general schema of the proposed PLA strategy for the A-Team is presented in Figure 1. It is worth noticing that computations performed inside while loop are carried out independently and possibly in parallel, within the agent environment used.

6 Computational Experiment

6.1 Settings

To evaluate the effectiveness of the proposed approach the computational experiment has been carried out using benchmark instances of RCPSP from PSPLIB[1] - test sets: sm30 (single mode, 30 activities), sm60, sm90, sm120. Each of the first three sets includes 480 problem instances while set sm120 includes 600 instances. The experiment involved computation with a fixed number of optimization agents, fixed population size, and the stopping criteria indicated by the environment state. Some of the parameters depend on the computation stage.

In the experiment the following global parameters have been used:

$|P| = 30$

$nITns = 5$

$nLS = 1$, 2 or 3

To enable comparisons with other algorithms known from the literature, the number of schedules generated during computation is calculated. In the presented experiments the number of schedules denoted as $nSGS$ is limited to 5000.

To validate the approach three PLA based strategies are checked: PLA1S with one learning stage, PLA2S with two stages and PLA3S with three stages. In each stage a different set of parameters and OptiAgents are used. These sets of settings are shown in the Tab. 1-3.

Stage 1	
initial weights	$w_{mgr} = 25$, $w_{mgrc} = 25$, $w_{mgb} = 25$, $w_{mgbc} = 25$ $w_{mrr} = 34$, $w_{mrw} = 33$, $w_{mrt} = 33$
optimization agents	OptCA, OptPRA, OptLSAm(10), OptLSAe(10), OptTSAm(20), OptTSAe(20)
stopping criteria	avgdiv(P)¿0.05 and $nSGS$¡5000

Table 1: PLA1S

To calculate weights an effective approach based on reinforcement learning proposed in [Jędrzejowicz, 14] have been used. In case of the positive reinforcement the additive adaptation for the weights is used: $w = w + 1$, and in case of

[1] See PSPLIB at http://www.om-db.wi.tum.de/psplib/

Stage 1	
initial weights	$w_{mgr} = 50$, $w_{mgrc} = 50$, $w_{mgb} = 0$, $w_{mgbc} = 0$
	$w_{mrr} = 100$, $w_{mrw} = 0$, $w_{mrt} = 0$
optimization agents	OptCA, OptPRA, OptLSAm(10), OptLSAm(10)
stopping criteria	avgdiv(P)¿0.1 or $nSGS$¡2500
Stage 2	
initial weights	$w_{mgr} = 25$, $w_{mgrc} = 25$, $w_{mgb} = 25$, $w_{mgbc} = 25$
	$w_{mrr} = 34$, $w_{mrw} = 33$, $w_{mrt} = 33$
optimization agents	OptLSAe(20), OptTSAm(20), OptTSAe(20)
stopping criteria	avgdiv(P)¿0.05 and $nSGS$¡2500

Table 2: PLA2S

Stage 1	
initial weights	$w_{mgr} = 50$, $w_{mgrc} = 50$, $w_{mgb} = 0$, $w_{mgbc} = 0$
	$w_{mrr} = 100$, $w_{mrw} = 0$, $w_{mrt} = 0$
optimization agents	OptCA, OptPRA
stopping criteria	avgdiv(P)¿0.1 and $nSGS$¡2000
Stage 2	
initial weights	$w_{mgr} = 25$, $w_{mgrc} = 25$, $w_{mgb} = 25$, $w_{mgbc} = 25$
	$w_{mrr} = 34$, $w_{mrw} = 33$, $w_{mrt} = 33$
optimization agents	OptLSAm(10), OptLSAe(10)
stopping criteria	avgdiv(P)¡0.5 and $nSGS$¡1500
Stage 3	
initial weights	$w_{mgr} = 0$, $w_{mgrc} = 0$, $w_{mgb} = 50$, $w_{mgbc} = 50$
	$w_{mrr} = 0$, $w_{mrw} = 20$, $w_{mrt} = 80$
optimization agents	OptTSAm(20), OptTSAe(20)
stopping criteria	avgdiv(P)¿0.05 and $nSGS$¡1500

Table 3: PLA3S

the negative reinforcement the additive $w = w - 1$ or root adaptation $w = \sqrt{w}$ is used. If the utility value falls below 1, it is reset to 1; if the utility value exceeds a certain max_w, it is reset to max_w if the utility value is assigned a non-integer value, it is rounded down.

The parameters values have been chosen experimentally based on earlier experiments for the RCPSP in JABAT and the preliminary experiments for the PLA strategy conducted using data set sm60.

The experiment has been carried out using nodes of the cluster Holk of the Tricity Academic Computer Network built of 256 Intel Itanium 2 Dual Core 1.4 GHz with 12 MB L3 cache processors and with Mellanox InfiniBand interconnections with 10Gb/s bandwidth. During the computation one node per four optimization agents was used.

6.2 Results

Experiment results are summed up in Tab. 4-7.

Strategy	MRE	MCT [s]	MTCT [s]
Blocking2	0.028%	6.43	72.62
RL123	0.018%	2.18	34.96
PLA1S	0.021%	2.31	38.24
PLA2S	0.017%	2.34	36.86
PLA3S	0.016%	2.45	37.53

Table 4: Results for benchmark test set sm30 (RE from the optimal solution)

Strategy	MRE from the best known solution	MRE from the CPLB	MCT [s]	MTCT [s]
Blocking2	0.64%	11.44%	32.70	75.56
RL123	0.51%	11.16%	12.50	62.30
PLA1S	0.51%	11.17%	12.55	62.32
PLA2S	0.47%	11.09%	12.35	61.52
PLA3S	0.48%	11.10%	13.12	62.45

Table 5: Results for benchmark test set sm60 (MRE from the best known solution and CPLB)

During the experiment the following characteristics of the computational results have been calculated and recorded: Mean Relative Error (MRE) calculated as the deviation from the optimal solution for sm30 set or from the best known results and the Critical Path Lower Bound (CPLB) for sm60, sm90 and sm120 sets, Mean Computation Time (MCT) needed to find the best solution and Mean

Strategy	MRE from the best known solution	MRE from the CPLB	MCT [s]	MTCT [s]
Blocking2	1.19%	11.38%	36.05	74.51
RL123	0.98%	10.91%	24.23	77.29
PLA1S	0.99%	10.95%	25.11	77.41
PLA2S	0.95%	10.45%	23.43	75.13
PLA3S	0.94%	10.41%	23.39	75.36

Table 6: Results for benchmark test set sm90 (MRE from the best known solution and CPLB)

Strategy	MRE from the best known solution	MRE from the CPLB	MCT [s]	MTCT [s]
Blocking2	3.17%	34.43%	78.38	137.10
RL123	2.91%	33.27%	88.11	202.41
PLA1S	2.87%	33.02%	90.27	200.56
PLA2S	2.51%	32.19%	89.47	197.29
PLA3S	2.49%	32.20%	89.51	199.32

Table 7: Results for benchmark test set sm120 (MRE from the best known solution and CPLB)

Total Computation Time (MTCT) needed to stop all optimization agents and the whole system. The number of schedules generated by SGS heuristics is limited to 5000 for all optimization agents used during search for the solution for each problem instance. Each instance has been solved five times and the results have been averaged over these solutions.

In each case all solutions were feasible. The results generated by the proposed approach are good and very promising. The mean relative errors for PLA3S and RL1234 below 1% in case of 30, 60 and 90 activities and below 2.5% in case of 120 activities have been obtained. The maximum relative error is below 6% and 8% respectively. It should be noted that in case of the PLA1S with a single learning stage the results are very similar to these obtained by the RL123 where similar reinforcement learning rules are used. In Figure 2 it can be seen that introducing more learning stages improves the results but the differences between PLA2S and PLA3S are very small.

Simultaneously, it can be observed that introducing more complex interaction strategy for the A-Team insignificantly influences computation time. Hence

Algorithm	Authors	MRE	MCT [s]	Computer
		Set sm30		
Decompos. & local opt	Palpant et al.	0.00	10.26	2.3 GHz
Filter and fan	Ranjbar	0.00	–	–
Event list-based EA	Paraskevopoulos et al.	0.00	0.19	1.33 GHz
VNS–activity list	Fleszar, Hindi	0.01	5.9	1.0 GHz
this approach		*0.02*	*2.458*	*1.4 GHz*
Local search–critical	Valls et al.	0.06	1.61	400 MHz
		Set sm60		
Filter and fan	Ranjbar	10.50	–	–
Event list-based EA	Paraskevopoulos et al.	10.54	16.31	1.33 GHz
Decompos. & local opt	Palpant et al.	10.81	38.8	2.3 GHz
Population–based	Valls et al.	10.89	3.7	400 MHz
this approach		*11.10*	*13.12*	*1.4 GHz*
Local search–critical	Valls et al.	11.45	2.8	400 MHz
		Set sm90		
Filter and fan	Ranjbar	10.11	–	–
Decomposition based GA	Debels, Vanhoucke	10.35	–	–
this approach		*10.41*	*23.39*	*1.4 GHz*
GA–hybrid, FBI	Valls at al.	10.46	–	–
		Set sm120		
Event list-based EA	Paraskevopoulos et al.	30.78	123.45	1.33 GHz
Filter and fan	Ranjbar	31.42	–	–
Population-based	Valls et al.	31.58	59.4	400 MHz
this approach		*32.20*	*89.51*	*1.4 GHz*
Decompos. & local opt.	Palpant et al.	32.41	207.9	2.3 GHz
Local search–critical	Valls et al.	34.53	17.0	400 MHz

Table 8: Literature reported results [Hartmann, 06], [Ranjbar, 08], [Agarwal, 11], [Paraskevopoulos, 12]

additional and other learning stages or methods could be considered. Additionally, the shorter computation times are observed in comparison to the Blocking2 strategy. It is the result of changeable stopping criteria, which in case of the proposed PLA-based strategy are based on the environment state parameters.

The presented results are comparable with the results reported in the literature. In Table 8 results obtained by the heuristic algorithms compared in [Agarwal, 11], [Hartmann, 06], [Ranjbar, 08] are presented. However in case of

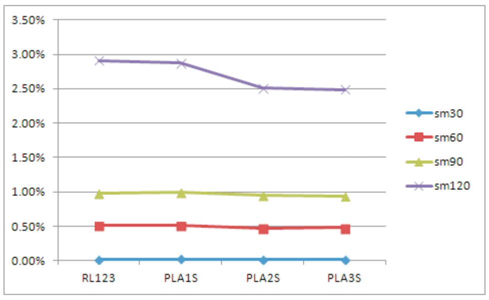

Figure 2: The graphical representation of the mean relative errors from the best or the best known solutions presented in Tables 4–7.

the agent based approach it is difficult to directly compare computation times as well as numbers of schedules. In the proposed agent-based approach computation times as well as number of schedules differ between nodes and optimization agent algorithms working in parallel. Results obtained by a single agent may or may not influence those obtained by other agents. Additionally, computation times include times used by agents to prepare, send and receive messages.

The experiment results show that the proposed implementation is effective and using Population Learning Algorithm to control the dynamic interaction strategy in the A-Teams architecture is beneficial.

7 Conclusions and Future Work

The computational experiment results show that the proposed dedicated A-Team architecture supported by the Population Learning Algorithm to control the interaction strategy is an effective and competitive tool for solving instances of the RCPSP. Presented results are comparable with solutions known from the literature and in some cases outperform them. It can be also noted that they have been obtained in a comparable computation time and number of schedules.

The presented experiment could be extended to examine different and additional parameters of the environment state and solutions as well as iteration numbers, probabilities and weights. On the other hand the additional or other learning stages should be examined. The kind and number of optimization agents (OptiAgents) used in each stage should be interesting to investigate. Additionally, an effective method for tuning optimization agents parameters including a number of iterations needed should be developed.

References

[Agarwal, 11] Agarwal, A., Colak, S., Erenguc, S.: A Neurogenetic Approach for the Resource–Constrained Project Scheduling Problem, Computers & Operations Research 38, (2011) 44–50.

[Barbucha, 09] Barbucha, D., Czarnowski, I., Jędrzejowicz, P., Ratajczak-Ropel, E., Wierzbowska, I.: E-JABAT - An Implementation of the Web-Based A-Team, in: N.T. Nguyen, L.C. Jain (Eds.), Intelligent Agents in the Evolution of Web and Applications, Springer, Heilderberg (2009), 57–86.

[Barbucha, 10] Barbucha, D., Czarnowski, I., Jędrzejowicz, P., Ratajczak-Ropel, E., Wierzbowska, I.: Influence of the Working Strategy on A-Team Performance, Smart Information and Knowledge Management, in E. Szczerbicki, N.T. Nguyen (Eds.), Studies in Computational Intelligence 260 (2010), 83–102.

[Barbucha, 11] Barbucha, D., Czarnowski, I., Jędrzejowicz, P., Ratajczak-Ropel, E., Wierzbowska, I.: Parallel Cooperating A-Teams, in: P. Jędrzejowicz et al. (Eds.); Computational Collective Intelligence. Technologies and Applications. Lecture Notes in Artificial Intelligence 6923 (2011), 322–331.

[Bellifemine, 03] Bellifemine, F., Caire, G., Poggi, A., Rimassa, G.: JADE. A White Paper, Exp 3, 3 (2003), 6–20.

[Błażewicz, 83] Błażewicz, J., Lenstra, J., Rinnooy, A.: Scheduling subject to resource constraints: Classification and complexity, Discrete Applied Mathematics 5 (1983), 11–24.

[Brucker, 99] Brucker, P., Drexl, A., Möhring, R., Neumann, K., Pesch, E.: Resource-Constrained Project Scheduling: Notation, Classification, Models, and Methods, European Journal of Operational Research 112 (1999), 3–41.

[Cadenas, 09] Cadenas, J.M., Garrido, M.C., Muñoz, E.: Using machine learning in a cooperative hybrid parallel strategy of metaheuristics, Information Sciences 179, 19 (2009), 3255–3267.

[Demeulemeester, 02] Demeulemeester, E., Herroelen, W.: Project scheduling: A research handbook, Kluwer Academic Publishers (2002).

[Fang, 12] Fang, C., Wang, L.: An effective shuffled frog-leaping algorithm for resource-constrained project scheduling problem, Computers & Operations Research 39 (2012), 890–901.

[Hartmann, 06] Hartmann, S., Kölisch, R.: Experimental Investigation of Heuristics for Resource-Constrained Project Scheduling: An Update, European Journal of Operational Research 174 (2006), 23–37.

[Jędrzejowicz, 99] Jędrzejowicz, P., Social learning algorithm as a tool for solving some difficult scheduling problems, Found. Comput. Decis. Sci. 24, 2 (1999), 51–66.

[Jędrzejowicz, 08] Jędrzejowicz, P., Ratajczak-Ropel, E.: New Generation A-Team for Solving the Resource Constrained Project Scheduling, Proc. of the Eleventh International Workshop on Project Management and Scheduling, Istanbul (2008), 156–159.

[Jędrzejowicz, 09] Jędrzejowicz P., Ratajczak-Ropel E.: Solving the RCPSP/max Problem by the Team of Agents, in: A. Hakansson et al.(eds.) Agent and Multi-Agent Systems: Technologies an Applications; Lecture Notes in Artificial Intelligence 5559 (2009), 734–743.

[Jędrzejowicz, 14] Jędrzejowicz, P., Ratajczak-Ropel, E.: Reinforcement Learning Strategies for A-Team Solving the Resource-Constrained Project Scheduling Problem, Neurocomputing 146 (2014), 301–307.

[Jędrzejowicz, 15] Jędrzejowicz, P., Ratajczak-Ropel, E.: Reinforcement Learning Strategy for Solving the MRCPSP by a Team of Agents, Intelligent Decision Technologies, Proceedings of the 7th KES International Conference on Intelligent Decision Technologies (KES-IDT 2015), Red.: Rui Neves-Silva, Lakhmi C. Jain, and Robert J. Howlett, Springer International Publishing, Switzerland (2015), 537–548.

[Jędrzejowicz, 06] Jędrzejowicz, P., Wierzbowska, I.: JADE-Based A-Team Environment, Computational Science - ICCS, Lecture Notes in Computer Sciences 3993 (2006), 719–726.

[Paraskevopoulos, 12] Paraskevopoulos, D.C., Tarantilis, C.D., Ioannou G.: Solving project scheduling problems with resource constraints via an event list-based evolutionary algorithm, Expert Systems with Applications 39, (2012) 3983–3994.

[Pelta, 06] Pelta, D., Cruz, C., Sancho-Royo, A., Verdegay, J.L.: Using memory and fuzzy rules in a cooperative multi-thread strategy for optimization, Information Sciences 176, 13 (2006), 1849–1868.

[Ranjbar, 08] Ranjbar, M.: Solving the resource constrained project scheduling problem using filter-and-fan approach, Applied Mathematics and Computation 201 (2008), 313–318.

[Talukdar, 96] Talukdar, S., Baerentzen, L., Gove, A., De Souza, P.: Asynchronous Teams: Co-operation Schemes for Autonomous, Computer-Based Agents, Technical Report EDRC 18-59-96, Carnegie Mellon University, Pittsburgh (1996).

[Wooldridge, 09] Wooldridge, M.: An Introduction to MultiAgent Systems - Second Edition, John Wiley & Sons (2009).