A-Team for Solving the Resource Availability Cost Problem

Piotr Jędrzejowicz and Ewa Ratajczak-Ropel

Department of Information Systems, Gdynia Maritime University
Morska 83, 81-225 Gdynia, Poland
{pj,ewra}@am.gdynia.pl

Abstract. In this paper the agent system based on A-Team and E-JABAT architecture for solving the resource availability cost problem (RACP) is proposed and experimentally tested. RACP known also as RIP (resource investment problem) belongs to the NP-hard problem class. To solve this problem an A-Team consisting of an asynchronous agents implemented using E-JABAT middleware have been proposed. Three kinds of optimization agent have been used. Computational experiment involves evaluation of the proposed approach.

Keywords: project scheduling, resource availability cost problem, RACP, resource investment problem, RIP, optimization, A-Team, agent, agent system.

1 Introduction

The paper proposes an agent based approach to solving instances of the resource availability cost problem (RACP) known also as resource investment problem (RIP). The considered problem have attracted less attention then other project scheduling problems, for example resource-constrained project scheduling problem (RCPSP). However, it is of great practical significance. It is used to model, for example, the bridge construction, staff management problems or negotiations the price of a project [17],[1]. In this problem the total costs of using a given amount of resource for the project is minimized. A solution of this problem consists of a set of activity starting times and a set of resource capacities, while respecting a project deadline. The problem is NP-hard.

RACP problem was introduced by Mohring (1984) [17] as the resource investment problem (RIP). He proposed an exact algorithm based on the known procedure for the RCPSP problem to solve it. Demeulemeester (1995) [9] proposed the next exact algorithm based on a branch-and-bound procedure for the RCPSP developed by himself and Herroelen (1992) [7], [8]. Rodrigues and Yamashita (2010) [22] modified the algorithm of Demeulemeester by reducing the search space using new bounds for branching scheme.

A few heuristic and metaheuristic algorithms are proposed to solve the RACP problem in the literature. Drexl and Kimms (2001) [10] develop two lower bounds for this problem using Lagrangean relaxation and column generation techniques,

N.-T. Nguyen et al. (Eds.): ICCCI 2012, Part II, LNAI 7654, pp. 443–452, 2012.

respectively. Both procedures are capable of yielding feasible solutions as well, so they also proposed two optimization guided heuristics. Yamashita et al. (2006) [26] proposed a multi-start heuristic based on the scatter search methodology using dynamic updating of the reference set, frequency-based memory within the diversification generator, and a combination method based on path relinking. Shadrokh and Kianfar (2007) [23] develop a genetic algorithm for the RACP in which the tardiness is permitted with penalty. Ranjbar et al. (2008) [21] developed two algorithms: a path relinking procedure and a genetic algorithm, in which a schedule is created with a precedence feasible priority list given to the schedule generation scheme. Van Peteghem and Vanhoucke (2011) [25] proposed an artificial immune system algorithm inspired by the vertebrate immune system and using new fitness function, the probability function for the composition of capacity lists, and the K-means diversity evaluation function for the preservation of diversity. Additionally the modification of the RACP problem has been proposed in several papers.

Approaches mentioned above to solve the RACP problem produce either approximate solutions or can be only applied for solving instances of the limited size. Hence, searching for more effective algorithms and solutions to the RACP/RIP problem 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 the common feature of the contemporary multiple-agent systems.

The multiple-agent systems are an important and intensively expanding area of research and development. There exists a number of multiple-agent approaches proposed to solve different types of optimization problems. One of them is the concept of an asynchronous team (A-Team), originally introduced by [24]. The idea of A-Team was used to develop the JADE-based environment for solving a variety of computationally hard optimization problems called E-JABAT ([12],[2]). E-JABAT is a middleware supporting the construction of the dedicated A-Team architectures based on the population-based approach. The mobile agents used in E-JABAT allow for decentralization of computations and use of multiple hardware platforms in parallel, resulting eventually in more effective use of the available resources and reduction of the computation time.

In this paper the E-JABAT-based A-Team architecture for solving the RACP problem instances is proposed and experimentally validated. A-Team includes optimization agents which represent heuristic algorithms. The behavior of the A-Team is defined by the, so called, working strategy. In the proposed approach the architecture for the RACP problem implemented using the E-JABAT environment and three optimization algorithms based on local search, path relinking and Lagrangean relaxation, has been proposed.

The paper is constructed as follows: Section 2 of the paper contains the RACP problem formulation. Section 3 gives some information on E-JABAT environment. Section 4 provides details of the proposed A-Teams architecture designed for solving the RACP instances. Section 5 describes settings of the computational experiment carried-out with a view to validate the proposed approach.

Section 6 contains a discussion of the computational experiment results. Finally, Section 7 contains conclusions and suggestions for future research.

2 Problem Formulation

Single resource availability cost problem consists of a set of $n+2$ activities, where each activity has to be processed without interruption to complete the project. The dummy activities 0 and $n+1$ represent the beginning and the end of the project. The duration of an activity j, $j = 0, \ldots, n+1$ is denoted by d_j where $d_0 = d_{n+1} = 0$. There are r renewable resource types. The availability of each resource type k in each time period is unlimited but using each unit of each resource type costs. There are r cost values, one for each resource c_k, $k = 1, \ldots, r$. Each activity j requires r_{jk} units of resource k during each period of its duration, where $r_{1k} = r_{nk} = 0$, $k = 1, ..., r$.

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, $j = 1, \ldots, n$. It is further assumed that $0 \in SP_j$, $j = 1, \ldots, n+1$, and $n+1 \in SS_j$, $j = 0, \ldots, n$.

There is also a time limit impose for the project execution as deadline D. All parameters, except costs are non-negative integers.

The objective is to find a schedule S of activities starting times $[s_1, \ldots, s_n]$, where $s_1 = 0$ and $s_{n+1} \leq D$ and resource requirements $[r_1, \ldots, r_k]$, such that the total resource cost is minimized.

Formally, the RACP problem can be described as follows:

$$\min \sum_{k=1}^{r} c_k r_k \tag{1}$$

s.t.

$$s_i + d_j \leq s_j \quad \forall (i,j) \in SA \tag{2}$$

$$\sum_{i \in A_t} r_{ik} \leq r_k \quad \forall k = 1, \ldots r, \; t = 1, \ldots, D \tag{3}$$

where A_t denotes the set of activities processed in time t,

$$s_{n+1} \leq D \tag{4}$$

$$s_0 = 0 \tag{5}$$

$$r_k \geq 0 \quad \forall k = 1, \ldots r \tag{6}$$

The above formulated problem as a generalization of the classical job shop scheduling problem belongs to the class of NP-hard optimization problems [4],[17].

RACP can be denoted as $PS_m, \infty | prec | \sum C_k max \; r_k(S,t)$ [5] or $m, 1 | cpm, \delta_n | rac$, ($rac$ means resource availability costs) [11].

3 The E-JABAT Environment

E-JABAT is a middleware allowing to design and implement A-Team architectures for solving various combinatorial optimization problems, such as the resource-constrained project scheduling problem (RCPSP), the traveling salesman problem (TSP), the clustering problem (CP), the vehicle routing problem (VRP). It has been implemented using JADE framework. The problem-solving paradigm on which the proposed system is based can be best defined as the population-based approach.

E-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. An initial population of solutions (individuals) is generated or constructed. Individuals forming an initial population are, at the following computation stages, improved by independently acting agents. Main functionality of the proposed environment includes organizing and conducting the process of search for the best solution.

To perform the above described cycle two main classes of agents are used. The first class called OptiAgent is a basic class for all optimization agents. The second class called SolutionManager is used to create agents or classes of agents responsible for maintenance and updating individuals in the common memory. All agents act in parallel. Each OptiAgent represents a single improvement algorithm (for example: local search, simulated annealing, tabu search, genetic algorithm etc.).

Other important classes in E-JABAT include: Task representing an instance or a set of instances of the problem and Solution representing the solution. To initialize the agents and maintain the system the TaskManager and PlatformManager classes are used. Objects of the above classes also act as agents.

E-JABAT environment has been designed and implemented using JADE (Java Agent Development Framework), which is a software framework supporting the implementation of multi-agent systems. More detailed information about E-JABAT environment and its implementations can be found in [12] and [2].

4 E-JABAT for Solving the RACP Problem

E-JABAT environment was successfully used by the authors for solving the RCPSP, MRCPSP and RCPSP/max problems ([13],[14],[3]). In the proposed approach the new data representation has been proposed dedicated for the RACP problem. Additionally some modification in order to improve the system efficiency has been implemented.

Classes describing the problem are responsible for reading and preprocessing the data and generating random instances of the problem. The discussed set includes the following classes:

– RACPTask inheriting from the Task class and representing the instance of the problem,

- RACPSolution inheriting from the Solution class and representing the solution of the problem instance,
- Activity representing the activity of the problem,
- Resource representing the renewable resource,
- TimeUnit representing the time unit in which the activities are processed.

The second set includes classes describing the optimization agents. Each of them includes the implementation of an optimization heuristic used to solve the RCPSP problem. All of them are inheriting from OptiAgent class. In the proposed dedicated A-Team this set includes the following classes:

- OptiLRA denoting the Lagrangean Relaxation Algorithm (LRA),
- OptiLSA denoting the Local Search Algorithm (LSA),
- OptiPRA denoting Path Relinking Algorithm (PRA),

The LRA is an implementation of the heuristic based on the Lagrangean relaxation method proposed by Drexl and Kimms in [10]. The relaxed problem of minimizing the total weighted completion times of the activities subject to precedence constraints is solved after convertion to minimum cut problem [18]. The implementation of the push relabel maximum flow algorithm described in [6] was used. The solution obtained represents a feasible suboptimal solution of the RACP problem.

Additionally, the above mentioned optimization agent and its algorithm based on the Lagrangean relaxation method is used to compute and update lower and upper bound for the processing instance. The bounds values are stored in RACPTask and used to stop computation in case when the lower bound or upper bound is reached by an agent.

The LSA is a local search algorithm which finds the shortest schedule for the considered problem with fixed resource availabilities by making a move. The move is understood as moving one of the activity to a new position in the schedule. All possible places in the schedule are checked in one iteration. For each combination of activities the value of possible solution is calculated. The best schedule is remembered and finally returned. The resource availabilities are calculated as follows:

- for feasible initial solution - the resource availabilities are decreased by x_k coefficient but not less then to the resource availability lower bound:

$$r_k = max(r_k - x_k(r_k - r_k^{LB}), r_k^{LB}), \text{ for } k = 1, \ldots, r, \tag{7}$$

- for infeasible initial solution - the resource availabilities are increased by y_k coefficient but not more then to the resource availability upper bounds:

$$r_k = min(r_k + y_k(r_k^{UB} - r_k), r_k^{UB}), \text{ for } k = 1, \ldots, r. \tag{8}$$

The resource availability lower r_k^{LB} and upper bound r_k^{UB} are calculated initially and updated during computation by the LRA algorithm. The coefficients x_k, for k=1,...,r. are set initially to 10% and updated during computation as well.

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 in the case of LSA 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 using minimal for these two solutions resource availabilities. The best schedule is remembered and finally returned.

An individual is represented as a schedule of activities S. The final solution is obtained from the schedule for fixed resource availabilities by Serial Generation Scheme (serial SGS) procedure [16].

All optimization agents (OptiAgents) co-operate together using their A-Team common memory managed by the SolutionManager. The working strategy of SolutionManager has been defined as follows:

- All individuals in the initial population of solutions are generated randomly, improved by the LRA algorithm and stored in the common memory.
- 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 all other individuals have been tried.
- Returning individual replaces the first found worse individual. If a worse individual cannot be found within a certain number of reviews (where review is understood as a search for the worse individual after an improved solution is returned) then the worst individual in the common memory is replaced by a randomly generated one.
- The computation time is defined by the no improvement time gap set by the user. If in this time gap no improvement of the current best solution has occurred, the A-Team stops computations.

5 Computational Experiment Settings

To evaluate the effectiveness of the proposed approach and compare the results the computational experiment has been carried out using benchmark instances generated by Yamashita at al. [26] for their computational experiment. The instances of RCPSP for 30, 60, 90 and 120 activities and 4 resource types are taken from the PSPLIB [19], and instances for RCPSP for 6 and 8 resource types has been generated by ProGen [15] using the following settings:

- Resource factor (RF): 0.25, 0.5, 0.75 and 1.0,
- Network complexity (NC): 1.5, 1.8 and 2.1.

Next, the instances has been adopted to RACP problem using Drexl and Kimms methodology [10] by removing the resource availability requirements, adding the costs drawn from a uniform distribution $U[1, 10]$ and adding the deadlines calculated using deadline factor $DF = 1.2$ ($D = DF \max_{i=0}^{n+1} s_i^{CP}$, where s_i^{CP} denotes the earliest starting times taken from the critical path).

The test set includes 144 problem instances. The experiment involved computation with the fixed number of optimization agents, fixed population size, and the limited time indicated by the no improvement time gap.

The proposed A-Team includes 3 optimization agents representing the LRA, LSA and PRA algorithms described in Section 4 - one of each type. The population has included 10 individuals, and the no improvement time gap has been set to 3 minutes. The values of the parameters are chosen on the basis of the previous experiments [12], [13], [14].

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 three optimization agents was used.

6 Computational Experiment Results

During the experiment the following characteristics of the computational results have been calculated and recorded: mean and maximal relative error (Mean RE) calculated as the deviation from the best solution obtained by Yamashita at al. [26] for three heuristics: scatter search with dynamic update (SSD) and two multi-start heuristic (FMS and RMS), the number of best results obtained, mean computation time required to find the best solution (Mean CT) and mean total computation time (Mean total CT). Each instance has been solved five times and the results have been averaged over these solutions.

Table 1. Performance of the proposed A-Team in terms of the mean relative error and number of the best results obtained

	#Activities				Mean	#Best results
	30	60	90	120		
A-Team for RACP	0.51%	1.23%	1.30%	1.52%	1.14%	96

Table 2. Performance of the proposed A-Team in terms of the mean computation time in seconds

	#Activities				Mean
	30	60	90	120	
A-Team for RACP	89.78s	123.42s	269.30s	315.92s	199.61s

Table 3. Performance of the proposed A-Team in terms of the mean total computation time in seconds

	#Activities				Mean
	30	60	90	120	
A-Team for RACP	141.04s	345.67s	470.53s	537.05s	373.57s

Table 4. Literature reported results [26]. Mean RE from the best known solution obtained by Yamashita at al. [26] and number of the best solutions for three heuristics: SSD, FMS and RMS.

	#Activities				Mean	#Best results
	30	60	90	120		
SSD	0.17%	0.00%	0.00%	0.00%	0.04%	137
FMS	0.42%	0.97%	1.33%	1.51%	1.06%	33
RMS	0.72%	1.77%	1.92%	2.26%	1.67%	31

Table 5. Literature reported results [26]. Mean computation time and mean total computation time in seconds for three heuristics: SSD, FMS and RMS.

	Mean CT	Mean total CT
SSD	1609.55s	3262.01s
FMS	945.09s	3135.13s
RMS	133.92s	3117.85s

Performance of the proposed A-Team is presented in Tables 1, 2 and 3. These results are compared with the results reported in the literature [26] shown in Tables 4 and 5.

The experiment results show that the proposed E-JABAT based A-Team for RACP implementation is effective and the results are comparable with the literature reported results. In each case the 100% of feasible solutions has been obtained. The times obtained in the experiment are quite good, however in the case of the agent based approaches it is difficult to directly compare computation times. The results obtained be a single agent may or may not influence the results obtained by the other agents. Additionally the computation time includes the time used by agents to prepare, send and receive messages.

7 Conclusions

Experiment results show that the proposed implementation based on the dedicated A-Team architecture is an effective and competitive tool for solving instances of the RACP problem. Presented results are comparable with solutions known from the literature. It can be also noted that they have been obtained in a comparable time. Time comparisons in this case might be misleading since the proposed A-Teams have been run using different numbers and kinds of processors. In case of the agent-based environments the significant part of the time is used for agent communication which has an influence on both - computation time and quality of the results.

The presented implementation and experiment is a first approach to construct A-Team for RACP problem. The experiment should be extended to examine the

A-Team behavior for different no improvement time gaps, different numbers of optimization agents and different population sizes. The other optimization algorithms and ideas to improve this implementation should be considered and tested.

Future research will concentrate on implementing more sophisticated procedures and optimization agents, as well as on searching for the best configuration of the heterogenous agents used during computations.

Acknowledgments. The authors are grateful to Proffesor Denise S. Yamashita and Proffesor Sávio B. Rodrigues from Federal University of São Carlos for making available the benchmark datasets and solutions for RACP problem.

The research has been supported by the Ministry of Science and Higher Education grant no. N N519 576438 for years 2010–2013. Calculations have been performed in the Academic Computer Centre TASK in Gdansk.

References

1. Artigues, C., Demassey, S., Néron, E.: Resource-Constrained Project Scheduling. Models, Algorithms, Extensions and Applications. ISTE Ltd. and John Wiley & Sons, Inc. (2008)
2. Barbucha, D., Czarnowski, I., Jędrzejowicz, P., Ratajczak-Ropel, E., Wierzbowska, I.: e-JABAT - An Implementation of the Web-Based A-Team. In: Nguyen, N.T., Jain, L.C. (eds.) Intel. Agents in the Evol. of Web & Appl. SCI, vol. 167, pp. 57–86. Springer, Heidelberg (2009)
3. Barbucha, D., Czarnowski, I., Jędrzejowicz, P., Ratajczak-Ropel, E., Wierzbowska, I.: Parallel Cooperating A-Teams. In: Jędrzejowicz, P., Nguyen, N.T., Hoang, K. (eds.) ICCCI 2011, Part II. LNCS (LNAI), vol. 6923, pp. 322–331. Springer, Heidelberg (2011)
4. Błażewicz, J., Lenstra, J., Rinnooy, A.: Scheduling subject to resource constraints: Classification and complexity. Discrete Applied Mathematics 5, 11–24 (1983)
5. 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, 3–41 (1999)
6. Cherkassky, B.V., Goldberg, A.V.: On Implementing Push-Relabel Method for the Maximum Flow Problem. In: Balas, E., Clausen, J. (eds.) IPCO 1995. LNCS, vol. 920, pp. 157–171. Springer, Heidelberg (1995)
7. Demeulemeester, E.L.: Optimal Algorithms for Various Classes of Multiple Resource-Constrained Project Scheduling Problems, Ph.D. thesis, Department of Applied Economics, Katholieke Universiteit Leuven, Belgium (1992)
8. Demeulemeester, E.L., Herroelen, W.S.: A Branch-and-Bound Procedure for the Multiple Resource-Constrained Project Scheduling Problem. Management Science 38, 1803–1818 (1992)
9. Demeulemeester, E.L.: Minimizing resource availability costs in time-limited project networks. Management Science 41, 1590–1598 (1995)
10. Drexl, A., Kimms, A.: Optimization guided lower and upper bounds for the resource investment problem. Journal of the Operational Research Society 52, 340–351 (2001)

11. Herroelen, W., De Reyck, B., Demeulemeester, E.L.: A classification scheme for project scheduling. In: Węglarz, J. (ed.) Handbook of Recent Advances in Project Scheduling, pp. 1–26. Kluwer, Dordrecht (1999)
12. Jędrzejowicz, P., Wierzbowska, I.: JADE-Based A-Team Environment. In: Alexandrov, V.N., van Albada, G.D., Sloot, P.M.A., Dongarra, J. (eds.) ICCS 2006, Part III. LNCS, vol. 3993, pp. 719–726. Springer, Heidelberg (2006)
13. Jędrzejowicz, P., Ratajczak-Ropel, E.: New Generation A-Team for Solving the Resource Constrained Project Scheduling. In: Proc. the Eleventh International Workshop on Project Management and Scheduling, Istanbul, pp. 156–159 (2008)
14. Jedrzejowicz, P., Ratajczak-Ropel, E.: Solving the RCPSP/max Problem by the Team of Agents. In: Håkansson, A., Nguyen, N.T., Hartung, R.L., Howlett, R.J., Jain, L.C. (eds.) KES-AMSTA 2009. LNCS (LNAI), vol. 5559, pp. 734–743. Springer, Heidelberg (2009)
15. Kolisch, R., Sprecher, A., Drexl, A.: Characterization and generation of a general class of resource-constrained project scheduling problems. Management Science 41, 1693–1703 (1995)
16. Kolisch, R.: Serial and parallel Resource-Constrained Project Scheduling Methods Revisited: Theory and Computation. European Journal of Operational Research 43, 23–40 (1996)
17. Möhring, R.: Minimizing Costs of Resource Requirements in Project Networks Subject to a Fixed Completion Time. Operations Research 32, 89–120 (1984)
18. Möhring, R.H., Schulz, A.S., Stork, F., Uetz, M.: Solving project scheduling problems by minimum cut computations. Management Science 49, 330–350 (2003)
19. PSPLIB, `http://129.187.106.231/psplib`
20. Radermacher, F.J.: Scheduling of Project Networks. Annals of Operations Research 4, 227–252 (1985)
21. Ranjbar, M., Kianfar, F., Shadrokh, S.: Solving the resource availability cost problem in project scheduling by path relinking and genetic algorithm. Appl. Math. Comput. 196, 879–888 (2008)
22. Rodrigues, S., Yamashita, D.: An exact algorithm for minimizing resource availability costs in project scheduling. European Journal of Operational Research 206, 562–568 (2010)
23. Shadrokh, S., Kianfar, F.: A genetic algorithm for resource investment project scheduling problem, tardiness permitted with penalty. European Journal of Operational Research 181, 86–101 (2007)
24. Talukdar, S., Baerentzen, L., Gove, A., de Souza, P.: Asynchronous Teams: Cooperation Schemes for Autonomous, Computer-Based Agents. Technical Report EDRC 18-59-96. Carnegie Mellon University, Pittsburgh (1996)
25. Van Peteghem, V., Vanhoucke, M.: An artificial immune system algorithm for the resource availability cost problem. Flexible Services and Manufacturing Journal, 1936-6582, 1–23 (2011)
26. Yamashita, D., Armentano, V., Laguna, M.: Scatter search for project scheduling with resource availability cost. European Journal of Operational Research 169, 623–637 (2006)