Neurocomputing 146 (2014) 301–307

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

Reinforcement Learning strategies for A-Team solving the Resource-Constrained Project Scheduling Problem

P. Jędrzejowicz, E. Ratajczak-Ropel *

Department of Information Systems, Gdynia Maritime University, Morska 83, 81-225 Gdynia, Poland

ARTICLE INFO

Article history:
Received 31 October 2013
Received in revised form
18 April 2014
Accepted 20 May 2014
Available online 26 June 2014

Keywords:
Project Scheduling
Resource-Constrained Project Scheduling
RCPSP
Multiagent System
A-Team
Reinforcement Learning

ABSTRACT

In this paper strategies for the A-Team with Reinforcement Learning (RL) for solving the Resource Constrained Project Scheduling Problem (RCPSP) are 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 the JABAT 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 a static strategy. In this paper the dynamic learning strategies are suggested. The proposed strategies based on reinforcement learning supervising interactions between optimization agents and the common memory. To validate the approach and compare strategies computational experiment has been carried out.

© 2014 Elsevier B.V. All rights reserved.

1. Introduction

The Resource Constrained Project Scheduling Problem (RCPSP) has attracted a lot of attention and many exact and heuristic algorithms have been proposed for solving it [1,12,22,15]. The current approaches to solve this problem 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 RCPSP 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 multiagent systems [27].

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 by Talukdar et al. [24]. The idea of the A-Team was used to develop the software environment for solving a variety of computationally hard optimization problems called JABAT [2,16]. 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.

Reinforcement Learning (RL) [6,23,17] belongs to the category of unsupervised machine learning algorithms. It can be described as learning which action to take in a given situation (state) to achieve one or more goal(s). The learning process takes place through interaction with an environment. At each discrete time step the current state is observed and some action(s) from the set of actions available at that state is taken. A numerical reward signal is returned to inform about the quality of actions. The optimal policy in order to maximize the reward signal is discovered. Actions may affect not only the immediate reward but also the next situation and, through that, all subsequent rewards.

RL is usually used in solving combinatorial optimization problems at three levels [26]:

The direct level - RL is directly applied to the problem.
The metaheuristic level - RL is used as a component of the respective metaheuristic.
The hyperheuristic level - RL is used as a component of the respective hyperheuristic.

The other field where RL is commonly used is the Multi-Agent Reinforcement Learning (MARL) where multiple reinforcement learning agents act together in the common environment [10,25]. In this paper RL is used to support strategy of searching for the optimal solution by an A-Team.

* Corresponding author.
E-mail addresses: pj@am.gdynia.pl (P. Jędrzejowicz), ewra@am.gdynia.pl (E. Ratajczak-Ropel).

http://dx.doi.org/10.1016/j.neucom.2014.05.070
0925-2312/© 2014 Elsevier B.V. All rights reserved.

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 strategy on the A-Team performance was investigated by Barbucha et al. [3]. In [5] the reinforcement learning based strategy for the synchronous team of agents has been considered. The similar topics were also considered by other authors for different multi-agent systems, e.g. [11,20].

In this paper the RL based on utility values to learn interaction strategy for the A-Team solving the RCPSP is proposed and experimentally validated. The new concept is using the RL to control the strategy parameters instead of the parameters required by the respective metaheuristics. It is expected that introducing the proposed RL will result in obtaining high quality solutions in an efficient manner.

Optimization agents used to produce solutions to the RCPSP instances in the considered A-Team represent four metaheuristic algorithms: local search, tabu search, crossover and path relinking. The proposed approach is based on the earlier JABAT adaptation for the RCPSP described in [13]. The most important difference is introduction of the RL based strategy. Details of the implementation are described in Section 4.

The paper is constructed as follows: Section 2 contains the RCPSP formulation. Section 3 gives some information on JABAT environment. Section 4 provides details of the proposed RL based JABAT implementation. Section 5 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 6 contains conclusions and suggestions for future research.

2. Problem formulation

A single-mode resource-constrained project scheduling problem consists of the 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, $j=1,\ldots,n$ 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, $k=1,\ldots,r$. Each activity j requires r_{jk} units of resource k during each period of its duration, where $r_{1k}=r_{nk}=0$, $k=1,\ldots,r$. 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, $j=1,\ldots,n$. It is further assumed that $1\in SP_j, j=2,\ldots,n$, and $n\in SS_j, j=1,\ldots,n-1$. The objective is to find a schedule S of activities starting times $[s_1,\ldots,s_n]$, where $s_1=0$ and resource constraints are satisfied, such that the schedule duration $T(S)=s_n$ is minimized.

The above formulated problem as a generalization of the classical job shop scheduling problem belongs to the class of NP-hard optimization problems [8]. The considered problem class is denoted as PS|$prec$|C_{max} [9].

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.

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 its population in accordance with the interaction strategy.

Important classes in JABAT include TaskManager and Platform Manager 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 of a 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 is used to define a set of rules applicable to managing and maintaining a population of current solutions in the common memory described in Section 4.

JABAT has been designed and implemented using JADE (Java Agent Development Framework), which is a software framework proposed by TILAB [7] supporting the implementation of the multiagent systems. More detailed information about the JABAT environment and its implementations can be found in [2,4,16].

4. A-team with interaction strategy controlled by RL

JABAT was successfully used by the authors for solving the RCPSP, MRCPSP, RCPSP/max and MRCPSP/max problems [13,14,4].

Download English Version:

https://daneshyari.com/en/article/406590

Download Persian Version:

https://daneshyari.com/article/406590

Daneshyari.com