Web Accessible A-Team Middleware

Dariusz Barbucha, Ireneusz Czarnowski, Piotr Jędrzejowicz,
Ewa Ratajczak-Ropel, and Izabela Wierzbowska

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

Abstract. The paper proposes a middleware called JABAT (Jade-based A-Team). JABAT allows to design and implement A-Team architectures for solving combinatorial optimization problem. JABAT is intended to become the first step towards next generation A-Teams which are fully Internet accessible, portable, scalable and in conformity with the FIPA standards.

Keywords: JABAT, A-Team, optimization, computationally hard problems, multi-agents systems.

1 Introduction

Last years a number of significant advances have been made in both the design and implementation of autonomous agents. A number of applications of agent technology is growing systematically. One of the successful approaches to agent-based optimization is the concept of an asynchronous team (A-Team), originally introduced by Talukdar [15].

A-Team is a multi agent architecture, which has been proposed in [15] and [16]. It has been shown that the A-Team framework enables users to easily combine disparate problem solving strategies, each in the form of an autonomous agent, and enables these agents to cooperate to evolve diverse and high quality solutions [14]. According to [15] an asynchronous team is a collection of software agents that solve a problem by dynamically evolving a population of solutions. Each agent works to create, modify or remove solutions from the population. The quality of the solutions gradually evolves over time as improved solutions are added and poor solutions are removed. Cooperation between agents emerges as one agent works on solutions produced by another. Each agent encapsulates a particular problem-solving method along with methods to decide when to work, what to work on and how often to work.

The reported implementations of the A-Team concept include two broad classes of systems: dedicated A-Teams and platforms, environments or shells used as tools for constructing specialized A-Team solutions. Dedicated (or specialized) A-Teams are usually not flexible and can be used for solving only particular types of problems. Among example A-Teams of such type one can mention the OPTIMA system for the general component insertion optimization problem

[13] or A-Team with a collaboration protocol based on a conditional measure of agent effectiveness designed for Flow optimization of railroad traffic [4].

Among platforms and environments used to implement A-Team concept some well known include IBM A-Team written in C++ with own configuration language [14] and Bang 3 - a platform for the development of Multi-Agent Systems (MAS) [12]. Some implementations of A-Team were based on universal tools like Matlab [16]. Some other were written using algorithmic languages like, for example the parallel A-Team of [5] written in C and run under PVM operating system.

The above discussed platforms and environments belong to the first generation of A-Team tools. They are either not portable or have limited portability, they also have none or limited scalability. Agents are not in conformity with the FIPA (The Foundation of Intelligent Psychical Agents) standards and there are no interoperability nor Internet accessibility. Migration of agents is either impossible or limited to a single software platform.

To overcome some of the above mentioned deficiencies a middleware called JABAT (Jade-based A-Team) was proposed in [11]. It was intended to become the first step towards next generation A-Teams which are portable, scalable and in conformity with the FIPA standards. JABAT allows to design and implement an A-Team architecture for solving combinatorial optimization problems. In this paper we report on e-JABAT which is an extension of JABAT to become the fully Internet-accessible solution.

The paper is organized as follows: Section 2 gives a short overview of the JABAT features. Section 3 introduces the concept of e-JABAT and describes the required actions, which have to be carried out by the user wishing to use the Web-based JABAT interface to obtain a solution to the problem at hand. Section 4 offers more details on the e-JABAT architecture. Section 5 contains some comments on flexibility of e-JABAT. Finally, Section 6 contains conclusions and suggestions for future research.

2 Main Features of the JABAT Middleware

JABAT is a middleware supporting design and development of the population-based applications intended to solve difficult computational problems. The approach is based on the A-Team paradigm.

Main features of JABAT include:

- The system can solve instances of several different problems in parallel.
- The user, having a list of all algorithms implemented for the given problem may choose how many and which of them should be used.
- The optimization process can be carried out on many computers. The user can easily add or delete a computer from the system. In both cases JABAT will adapt to the changes, commanding the optimizing agents working within the system to migrate.

- The system is fed in the batch mode - consecutive problems may be stored and solved later, when the system assesses that there is enough resources to undertake new searches.

The use case diagram depicting the functionality of JABAT is shown in Fig. 1.

Fig. 1. Use case diagram of the functionality of JABAT

JABAT produces solutions to combinatorial optimization problems using a set of optimising agents, each representing an improvement algorithm. The process of solving of the single task (i.e. the problem instance) consists of several steps. At first the initial population of solutions is generated. Individuals forming the initial population are, at the following computation stages, improved by independently acting agents, thus increasing chances for reaching the global optimum. Finally, when the stopping criterion is met, the best solution in the population is taken as the result.

The way the above steps are carried out is defined by the “strategy”. There may be different strategies defined in the system, each of them specifying:

- how the initial population of solutions is created (in most cases the solutions are drawn at random),
- how to choose solutions which are forwarded to the optimizing agents for improvement,
- how to merge the improved solutions returned by the optimizing agents with the whole population (for example they may be added, or may replace random or worst solutions),
- when to stop searching for better solutions (for example after a given time, or after no better solution has been found within a given time).

To validate the system a number of experiments has been conducted. Experiments have involved a variety of combinatorial optimization problems. The results reported in [10], [2], [6], [7], [8] have proved ability and effectiveness of JABAT with regard to solving computationally hard problems.

3 e-JABAT

In this paper we propose an extension of JABAT making it accessible through a web interface, in which most of the original functionality is available for users from all over the world. The system with its interface is further on referred to as e-JABAT. A working copy of e-JABAT may be found at the address <http://jabat.wpit.am.gdynia.pl>.

3.1 Solving Tasks

To solve a task within the system, a user has to register at the e-JABAT website. Registered users obtain access to the part of the website in which tasks can be uploaded for solving. The uploaded tasks are sequenced and solved in the order of uploading. They may be solved in parallel (even the tasks of different problems). After a task has been solved the user can download the results saved in a text file in the user's space.

Thus, the user can:

- upload a file containing a task to be solved,
- observe the status of all tasks he has uploaded (waiting, being solved, solved),
- observe the logfiles in which some additional information or error messages may be found,
- download the files with solutions,
- delete his tasks.

The user willing to upload a task to the system must also:

- Choose from the available list the problem that the task belongs to. At present four different problems have been implemented in e-JABAT and are available for the users of the web interface. These problems are:
 - resource constrained project scheduling problem with single and multiple node (RCPSP, MRCPSP),
 - clustering problem (CP),
 - euclidean planar traveling salesman problem (TSP) and
 - vehicle routing problem (VRP).

For each of these problems the format of the file containing the task to be solved is specified and published in the website.

- Choose which optimizing agents should be involved in the process of searching for the solution. Each of the optimizing agents within the system represents different optimization algorithm. For the problems implemented in the system there are available agents executing the following algorithms:
 - the local search algorithm, algorithm based on the simple evolutionary crossover operator and the tabu search algorithm for the RCPSP,
 - the Lin-Kernighan algorithm and the evolutionary algorithm for the TSP,
 - the 2-optimum algorithm operating on a single route, the λ interchange local optimisation method, the evolutionary algorithm and the local search algorithms for the VRP,

- the random local search, the hill-climbing local search and the tabu search algorithms for the CP.

For each of these algorithms a short description is available at the JABAT website.

- Choose a strategy from the list of available strategies and optionally define a set of options for this strategy, for example the size of the initial population or the length of time after which the search for better solutions stops.

For each optimizing agent the user may define the minimum and the maximum number of running copies of the agent. The system will initially use the minimal specified number and then the number will increase if there is enough computational resources available.

Fig. 2 shows the task upload screen, where the user’s choices are shown.

The JABAT Environment Project

About

[Project](#)

[JABAT](#)

[Authors](#)

Problems & agents

[RCPSP](#)

[VRP](#)

[TSP](#)

Strategies

[Basic](#)

[Blocking1](#)

Usage

[Configuration](#)

[New task](#)

[Your tasks](#)

Help

[Help & source](#)

[Contact](#)

Your task

Problem

RCPSP

	name	select	min	max
Agents	RCPSP.OptILSA	☐		
	RCPSP.OptTSA	☒	1	5
	RCPSP.OptCH	☐		
	RCPSP.OptMCSbH	☐		

Strategy

Basic

	name	select	value	description
Options	populationSize	☒	30	number of individuals in the initial population, eg. "23"
	stopMinutes	☒	5	number of minutes, eg. "1.5"

Data file

run

Copyright 2007, Department of Information Systems, Gdynia Maritime University

Fig. 2. Task upload screen

The report file created for each user’s task includes the best solution obtained so far (that of the maximum or minimum value of fitness), average value of fitness among solutions from current population, the actual time of running and the time in which the best solution was last reached. The file is created after the initial population has been generated and then the next set of data is appended to the file every time the best solution in the population changes. The final results are added to the content of the file when the stopping criterion has been met. The report on the process of searching for the best solution may be later analysed by the user. It can be easily read into a spreadsheet and converted into a summary report with the use of the pivot table.

3.2 Adding/Deleting Resources

JABAT makes it possible for optimisation agents to migrate or clone to other computers. By the use of mobile agents the system offers decentralization of

computations resulting in a more effective use of available resources and reduction of the computation time. Each registered user may launch a JADE container on his current host and attach it to the copy of JABAT running on `jabat.wpit.am.gdynia.pl`.

4 e-JABAT Architecture

The system consists of two parts: JABAT engine, responsible for the actual solving of computational task and web interface, in which a user can upload the tasks and their parameters and download the results (Fig. 3).

Fig. 3. e-JABAT architecture

4.1 Web Interface

Users obtain access to the JABAT engine through the web interface that has been created with the use of Java Server Faces and Facelets technologies. The interface

allows the user to specify the task and to provide some additional information concerning details on how the task solving process should be carried. The task uploaded by the user is saved in the directory from which it can be later read by the JABAT engine. The information given by the user during the process of uploading the task are written in an XML file stored in the area called input directory from which JABAT can read it.

Each XML batch file stored in the input directory contains a single set of data provided by the user: the problem name, one or more instance data, list of optimising agents which should be run in order to solve the task, name of the selected strategy and additional options for this strategy. The XML input files stored in the initial directory are read by the JABAT engine and solved in the order of appearance. The results are stored in the output directory, from which they can be downloaded by the users to which they belong.

4.2 The JABAT Engine

The JABAT engine is built using JADE (Java Agent Development Framework), a software framework proposed by TILAB [17] for the development and run-time execution of peer-to-peer applications. JADE is based on the agents paradigm in compliance with the FIPA [9] specifications and provides a comprehensive set of system services and agents necessary to distributed peer-to peer applications in the fixed or mobile environment. It includes both the libraries required to develop application agents and the run-time environment that provides the basic services and must be running on the device before agents can be activated [3].

JADE platforms have containers to hold agents, not necessarily on the same computer. In JABAT containers placed in different platforms are used to run agents responsible for searching for optimal solutions using pre-defined solution improvement algorithms.

Within the JABAT engine the following types of agents are used:

- *OptiAgents* - representing the solution improvement algorithms,
- *SolutionManagers* - managing the populations of solutions,
- *TaskManagers* - responsible for initialising the process of solving an instance of a problem (it for example creates and deletes agents that are designated to the task)
- *SolutionMonitors* - recording the results,
- *PlatformManager* - organising the process of migration between different containers and
- *ErrorMonitor* - monitoring unexpected behavior of the system.

Agents Responsible for Solving a Task. The most important for the process of solving a task are *OptiAgents* and *SolutionManagers*. They work in parallel and communicate with each other exchanging solutions that are either to be improved when they are sent to *OptiAgents*, or stored back into the common memory when sent to *SolutionManager*.

Each *OptiAgent* is a single improvement algorithm. An *OptiAgent* can communicate with all *SolutionManagers* working with instances of the same problem.

An agent sends out the message about its readiness to work. Such message contains information about the number of solutions from the common memory required to execute the improvement procedure. In response the *SolutionManager* sends the details of the task and appropriate number of solutions. The respective *OptiAgent* processes them and sends back the improved solution or solutions. The process iterates, until some stopping criterion is met.

Each *SolutionManager* is responsible for a single population of solutions created to solve a single task. Its actions include generation of the initial pool of solutions, sending solutions to the *OptiAgents*, merging improved solutions with the population store in the common memory and deciding when the whole solution finding process should be stopped. All these activities are managed in accordance with the strategy that has been chosen for the particular task. This strategy is a part of the *SolutionManager* provided as one of the set of the agent parameters.

Apart from the above *SolutionManager* is also responsible for sending periodic reports on the state of computations to the *SolutionMonitor* monitoring the respective task. The *SolutionMonitor*, in turn, prepares and saves information on the results thus obtained in the report file available to the user.

Agents Responsible for Creating/Deleting/Relocating Other Agents.

There are two important agents categories managing the process of solving tasks: *TaskManagers* and *PlatformManager*. Their role is to create, delete, relocate or copy agents responsible for the actual problem-solving process.

There is only one *TaskManager* in the system, responsible for reading input data and creating or deleting all agents designated to the particular task. *TaskManager* may initialise the process of solving next task before the previous has stopped if there is any such task waiting and if the global system setting allows for that.

The *PlatformManager* manages optimization agents and system platforms. It can move optimization agents among containers and create (or delete) their copies to improve computations efficiency. The *PlatformManager* work is based on the following simple rules:

- the number of *OptiAgents* cannot exceed the maximum number and can not be smaller than the minimum number of *OptiAgents*, as specified by the user,
- if JABAT has been activated on a single platform (computer), then all *OptiAgents* would be also placed on this platform,
- if JABAT has been activated on multiple platforms, with main container placed on one computer and the remote joined containers placed on other computers, then *OptiAgents* are moved from the main container to outside containers to distribute the workload evenly.

5 Flexibility of e-JABAT

The JABAT engine has been designed in such a way, that it can be easily extended to solving new problems or solving them with new algorithms. The main

idea is to reduce the amount of work of the programmer who wants to solve new problems or wishes to introduce new ways of representing tasks or solutions, new optimising algorithms or finally new replacement strategies. e-JABAT makes it possible to focus only on defining these new elements, while the processes of communication and population management procedures will still work. More detailed information about extending the functionality of JABAT can be found in [1].

6 Conclusions

The goal of the research presented in this paper was to propose a middleware environment allowing Internet accessibility and supporting development of A-Team systems. The solution - e-JABAT - has achieved this goal. Some of the advantages of e-JABAT have been inherited from JADE. The most important advantage seem to be e-JABAT ability to simplify the development of distributed A-Teams composed of autonomous entities that need to communicate and collaborate in order to achieve the working of the entire system. A software framework that hides all complexity of the distributed architecture plus a set of predefined objects are available to users, who can focus on the logic of the A-Team application and effectiveness of optimization algorithms rather than on middleware issues, such as discovering and contacting the entities of the system. It is believed that the proposed approach has resulted in achieving Internet accessible, scalable, flexible, efficient, robust, adaptive and stable A-Team architectures. Hence, e-JABAT can be considered as a step towards next generation A-Team solutions.

During the test and verification stages JADE-A-Team has been used to implement several A-Team architectures dealing with well known combinatorial optimization problems. Functionality, ease of use and scalability of the approach have been confirmed.

Further research will concentrate on extending scalability and efficiency features of e-JABAT. One of such features under current development is an extension of the system functionality to use the middleware through the Internet solely on a computer or computers directly controlled or owned by the user. Another development under way is the construction of the intelligent help which would provide guidance to users less advanced in using Java technologies.

References

1. Barbucha, D., Czarnowski, I., Jędrzejowicz, P., Ratajczak, E., Wierzbowska, I.: JADE-Based A-Team as a Tool for Implementing Population-Based Algorithms. In: Chen, Y., Abraham, A. (eds.) *Intelligent Systems Design and Applications*, IDSA, Jinan Shandong China, pp. 144–149. IEEE, Los Alamos (2006)
2. Barbucha, D., Jędrzejowicz, P.: An experimental investigation of the synergetic effect of multiple agents solving instances of the vehicle routing problem. In: Grzech, A. (ed.) *Proceedings of the 16th International Conference on Systems Science*, Wroclaw, vol. II, pp. 370–377 (2007)

3. Bellifemine, F., Caire, G., Poggi, A., Rimassa, G.: JADE. A White Paper, Exp 3(3), 6–20 (2003)
4. Blum, J., Eskandarian, A.: Enhancing intelligent agent collaboration for flow optimization of railroad traffic. *Transportation Research Part A* 36, 919–930 (2002)
5. Correa, R., Gomes, F.C., Oliveira, C., Pardalos, P.M.: A parallel implementation of an asynchronous team to the point-to-point connection problem. *Parallel Computing* 29, 447–466 (2003)
6. Czarnowski, I., Jędrzejowicz, P.: Implementation and Performance Evaluation of the Agent-Based Algorithm for ANN Training. In: Nguyen, N.T., Grzech, A., Howlett, R.J., Jain, L.C. (eds.) *KES-AMSTA 2007*. LNCS (LNAI), vol. 4496, pp. 131–140. Springer, Heidelberg (2007)
7. Czarnowski, I., Jędrzejowicz, P.: An agent-based approach to the multiple-objective selection of reference vectors. In: Perner, P. (ed.) *MLDM 2007*. LNCS (LNAI), vol. 4571, pp. 117–130. Springer, Heidelberg (2007)
8. Czarnowski, I., Jędrzejowicz, P.: An Agent-Based Algorithm for Data Reduction. In: Bramer, M., Coenen, F., Petridis, M. (eds.) *Research and Development in Intelligent Systems XXIV and Applications and Innovations in Intelligent Systems XV*, Proceedings of AI 2007, the Twenty-seventh SGAI International Conference on Artificial Intelligence, Springer, London (2007)
9. The Foundation for Intelligent Physical Agents, <http://www.fipa.org/>
10. Jędrzejowicz, P., Ratajczak-Ropel, E.: Agent-Based Approach to Solving the Resource Constrained Project Scheduling Problem. In: Beliczynski, B., Dzielski, A., Iwanowski, M., Ribeiro, B. (eds.) *ICANNGA 2007*. LNCS, vol. 4431, pp. 480–487. Springer, Heidelberg (2007)
11. 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*. LNCS, vol. 3993, pp. 719–726. Springer, Heidelberg (2006)
12. Neruda, R., Krusina, P., Kudova, P., Rydvan, P., Beuster, G.: Bang 3: A Computational Multi-Agent System. In: Proceedings of the IEEE/WIC/ACM International Conference on Intelligent Agent Technology, IAT 2004 (2004)
13. Rabak, C.S., Sichman, J.S.: Using A-Teams to optimize automatic insertion of electronic components. *Advanced Engineering Informatics* 17, 95–106 (2003)
14. Rachlin, J., Goodwin, R., Murthy, S., Akkiraju, R., Wu, F., Kumaran, S., Das, R.: A-Teams: An Agent Architecture for Optimization and Decision-Support. In: Rao, A.S., Singh, M.P., Müller, J.P. (eds.) *ATAL 1998*. LNCS (LNAI), vol. 1555, pp. 261–276. Springer, Heidelberg (1999)
15. Talukdar, S.N., de Souza, P., Murthy, S.: Organizations for Computer-Based Agents. *Engineering Intelligent Systems* 1(2) (1993)
16. Talukdar, S., Baerentzen, L., Gove, A., de Souza, P.: Asynchronous teams: cooperation schemes for autonomous agents. *Journal of Heuristics* 4, 295–321 (1998)
17. Jade - Java Agent Development Framework, <http://jade.tilab.com/>