Future Generation Computer Systems 37 (2014) 390–400

Contents lists available at ScienceDirect

Future Generation Computer Systems

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

Computing agents for decision support systems

D. Krzywicki, Ł. Faber, A. Byrski *, M. Kisiel-Dorohinicki

AGH University of Science and Technology, Al. Mickiewicza 30, 30-059 Krakow, Poland

HIGHLIGHTS

- Applicability of agent-oriented metaheuristics to decision support systems.
- Functional-programming based prototypes of agent-based computing systems.
- Experiments regarding scalability and performance of the implemented systems.

ARTICLE INFO

Article history:
Received 4 July 2013
Received in revised form
1 January 2014
Accepted 7 February 2014
Available online 17 February 2014

Keywords:
Decision support systems
Multi-agent systems
Scalability
Performance

ABSTRACT

In decision support systems, it is essential to get a candidate solution fast, even if it means resorting to an approximation. This constraint introduces a scalability requirement with regard to the kind of heuristics which can be used in such systems. As execution time is bounded, these algorithms need to give better results and scale up with additional computing resources instead of additional time. In this paper, we show how multi-agent systems can fulfil these requirements. We recall as an example the concept of Evolutionary Multi-Agent Systems, which combines evolutionary and agent computing paradigms. We describe several possible implementations and present experimental results demonstrating how additional resources improve the efficacy of such systems.

1. Introduction

The need to gather and analyse vast amounts of information from numerous sources has grown in importance. Such data is often a basis for simulations and computations that support decision making. It may be needed to run many computing tasks, in order either to test different parameters in a model or to verify a statistical hypothesis. An exhaustive search for optimal solutions to a decision making problem is usually time-consuming and thus not acceptable in real-time conditions. Instead, metaheuristics may quickly provide good-enough options to be further considered in the decision making process [1].

Examples of use cases with real-time constrains, where a quick approximated solution may be better than an outdated optimal one, may include:

- portfolio optimisation—a decision support system can apply different models to the available market data and allow the user to quickly react to arising trends [2];
- crisis management—in crisis situations, such as fire outbreaks, flooding or earthquakes, intensive simulations are required in order to suggest possible evacuation routes or to assign rescue units to tasks. Geographical information usually needs to be considered, yielding optimisation problems similar to transportation-related ones [3];
- production planning—decision support systems can help in scheduling work, rescheduling production plans in the case of hardware failures, implementing just in time strategies or balancing conflicting goals (e.g. high system throughput vs low machinery usage) [4].

Metaheuristics may still require a significant computational power if the acceptable solution is to be found in a reasonable time. For this purpose, large-scale infrastructure is usually used, such as clusters, grids or clouds. To fully benefit from this computational power, it is required to appropriately plan their development and deployment, along with adequate tools and careful testing.

Because of its intrinsic decentralisation [5], the agent approach is well suited to design scalable distributed models and has been applied in various decision support systems. This approach may be summarised as the introduction of artificial intelligence techniques into the system, transforming it from a passive tool into an active collaborator in decision making. A number of such case-oriented systems have been proposed and verified in practice [6,7].

* Corresponding author. Tel.: +48 126339406.
E-mail addresses: daniel.krzywicki@agh.edu.pl (D. Krzywicki), faber@agh.edu.pl (Ł. Faber), olekb@agh.edu.pl (A. Byrski), doroh@agh.edu.pl (M. Kisiel-Dorohinicki).

http://dx.doi.org/10.1016/j.future.2014.02.002
0167-739X/

Well-known general-purpose agent-based development tools (such as JADE [8], RePast [9] or MadKit [10]) may not be the best choice to implement such computational intensive simulations, when throughput and scalability are more important than code migrations or FIPA-compliant communication. Therefore, over the last 10 years, we have been involved in the development of several alternative platforms dedicated to large scale agent-based simulations and computations [11–13].

In this work, we discuss the implementation aspects of using computing agents in large-scale environments, with a focus on performance. We compare different approaches to agent execution and parallelism, based on the metaheuristic called evolutionary multi-agent systems (EMAS), which is a hybrid of agent-oriented and evolutionary-based computing [14]. We introduce the concept of *meeting arenas*, which allows to design more efficient and scalable multi-agent systems. Nevertheless, we show that explicit parallelism, when each agent is mapped onto a thread, can be much less effective than a simple but optimised sequential implementation. Finally, we show that such agent-based metaheuristics can be easily scaled with additional computational resources.

We start the paper with a discussion on the applicability of the agent-oriented paradigm and metaheuristics in decision support systems (Section 2), along with an EMAS example. In Section 3, we introduce the most common approaches to parallelism in agent-oriented computing and follow with a review of popular agent platforms in Section 4. We describe in Section 5 how to implement an evolutionary multi-agent system using two different approaches—a synchronous and asynchronous one. Finally, we conclude the paper by comparing the performance and scalability of both approaches in Section 6.

2. Agent-based metaheuristics in decision support

Decision Support Systems (DSS) are information systems that support different business or organisational activities involving decision-making. They are especially useful in situations where quickly changing, hard to specify in advance conditions are encountered. Referring to Power's taxonomy for DSSs [15] this paper focuses on Model-driven DSSs, which help the users in the analysis of the current situation by allowing to manipulate statistical, simulational or optimisational models.

2.1. Metaheuristics for DSSs

The models used in DSSs are usually very complex and computationally hard, because the underlying problems are very difficult as well. In such cases, one often turns to solutions based on so-called heuristic methods, which provide "good-enough" solutions without caring whether they may be proved to be correct or optimal [1]. These methods trade-off precision, quality and accuracy in favour of smaller execution time and computational effort. They are necessary to deal with difficult problems, and are referred to as methods of the last resort [16].

A general definition of a heuristic algorithm, which does not specify details such as a particular problem, search space or operators, is called a *metaheuristic*. For example, a simple algorithm such as greedy search may be defined without going into more details as "an iterative, local improvements of a solution based on random sampling" [17].

A simple but adequate classification of metaheuristics (cf. [18]) distinguishes two groups of techniques. Single-solution metaheuristics work on a single solution to a problem, seeking to improve it. The examples are greedy search, tabu search or simulated annealing. Population-based metaheuristics explicitly work with a population of solutions and put them together in order to generate new solutions. The examples are evolutionary algorithms, immunological algorithms, particle swarm optimisation, ant colony optimisation, memetic algorithms and other similar techniques. They are usually inspired by nature and imitate different phenomena observed in e.g., biology, sociology, culture or physics [19].

2.2. Agent approach

The key concept in multi-agent systems (MAS) consist in intelligent interactions, such as coordination, cooperation, or negotiation. Therefore, multi-agent systems are ideal in representing problems which can be solved using multiple methods by numerous entities with various perspectives. One of the most important features in a multi-agent system is the autonomy of the agents, as they can fulfil the tasks assigned to them according to their own strategy and the situation observed in their environment. In consequence, agents are adaptable and proactive [5].

Combining the agent-oriented approach with population-based metaheuristics seems natural but has yet been the topic of little work. The entities processed in the course of the computation can often be considered autonomous and treated as agents in a common environment. The operations involving many such entities can be defined as interactions between these agents.

This change of modelling perspective allows to perceive parts of the system on a higher abstraction level and build hybrid systems which combine techniques from different metaheuristics. New problems often arise, such as the lack of global knowledge or the need for proper synchronisation of the agents' actions. However, interesting and useful effects also often result from the cooperation of different mechanisms in one system [20].

In this paper, we focus on an example of such a hybrid approach, in which agents are subject to an evolutionary process. Such a combination yields interesting new features when compared to classical evolutionary algorithms, such as a decentralised an emergent selective pressure.

2.3. Evolutionary multi-agent systems

Agents in an evolutionary multi-agent system (EMAS) represent solutions to a given optimisation problem.

Inheritance is achieved through reproduction, with the possible use of variation operators such as mutation and recombination, like in classical evolutionary algorithms. Yet agents are to be autonomous in their decisions and no global knowledge is available to them. Therefore, in contrast to classical evolutionary algorithms, selection needs to be decentralised and involve peer-to-peer interactions instead of being system-wide.

In order to do that, a solution based on the acquisition and exchange of non-renewable resources has been proposed in [21]. The quality of the solution represented by the agent is expressed by the amount of resources the agent owns. In general, these resources should pass from worse agents to better ones. This might be realised through encounters between agents, which cause better ones to end up with more resources and make them more likely to reproduce. Worse agents lose resources which increases the probability of their death. Because of such indirect dynamics of reproduction and death, agents' lifespans overlap and so do the generations. Moreover, the size of the population is dynamic and can be changed by varying the amount of available resources. A detailed study of computing with EMAS, in particular the influence of its different parameters on the computing efficiency may be found in [22].

Agents are grouped within *environments* which define the information and resources an agent has access to. Agents can interact