User-Assisted Management of Agent-Based Evolutionary Computation

Aleksander Byrski and Marek Kisiel-Dorohinicki

AGH University of Science and Technology, Kraków, Poland
{olekb,doroh}@agh.edu.pl

Abstract. In the paper the need and general idea of user-assisted management of computational intelligence systems is discussed. First some methodological issues are presented with particular attention paid to agent-based approaches. These general considerations are supported by the case of evolutionary multi-agent system (EMAS) with immunological selection, applied to multi-modal function optimization subjected to a user-driven tuning procedure. Finally, preliminary experimental results considering the influence of selected parameters on the performance of the system are shown.

1 Introduction

Computational intelligence relies mostly on heuristics based on evolutionary, connectionist, or fuzzy systems paradigms, but also on other techniques, which used by themselves or in combination exhibit behavior that, in some sense, may be considered intelligent. One of the main advantages of such approaches is their ability to produce sub-optimal solutions even in cases, when traditional methods fail. One of the main drawbacks of these approaches is that they need to be adapted to each problem to be solved. Their configuration (a number of parameters in the simplest case) determines their behavior and thus often significantly influences their solving capabilities.

In practice the configuration of a particular technique appropriate for a given problem is usually obtained after a number of experiments performed by a user, which is often an unreliable and time-consuming task. Meanwhile the choice of specific mechanisms and their parameterization is crucial for the accuracy and efficiency of the examined technique. That is why various approaches were proposed to deal with the problem more reliably and possibly without involving a user directly. Yet automatic discovery of an acceptable system configuration by means of some optimization technique requires resolving some methodological issues, e.g. needs the formulation of the search domain and criterion, which may be difficult in many cases.

In the paper the problem of tuning of agent-based computational systems is considered. The abstraction of an agent provides a natural perspective for designing decentralized systems and may facilitate the construction of hybrid soft computing systems [1]. Immunological mechanisms introduced into agent-based evolutionary optimization may serve as an example of such approach [2]. At the same time, these immunological mechanisms may be perceived as a means for automatic tuning of evolutionary processes in the population of agents.

This paper starts with the discussion of the structure of management of computational system, aiming at discovering its acceptable configuration, with a particular emphasis on agent-based models. Next, as an example of such a system, the idea of *immunological evolutionary multi-agent system* is presented. The results of the user-assisted tuning of the system and their discussion conclude the paper.

2 User-Assisted Configuration of a Computational System

Most computational intelligence systems need a vast amount of work before they are able to provide acceptable solutions for a given problem. Hybrid systems, which put together elements of different techniques, by the effect of synergy often provide better solving capabilities and more flexibility [3], but they are obviously more difficult to configure. This is mainly because such systems are usually very hard to model and analyze. In many cases it may be even admitted that the principles of their work are not known, just like for many evolutionary algorithms: there were some approaches to model specific algorithms, yet they were successful only in very simple cases, like for a simple genetic algorithm [4].

When theoretical analysis of a computational model fails, its configuration needs to be decided experimentally. A result depends on the experience and patience of an expert, who is to perform many experiments to check accuracy and efficiency of the particular technique with different settings. Instead of an expert some automatic procedure may be applied to search for an optimal configuration of the computational system. In this case some criterion is indispensable to drive the search, which formulation may be really difficult in many cases. It must take into consideration ambiguity of the system evaluation based on its current state, or even based on gathered data from a single run, because of stochastic character of many soft computing techniques.

The task of automatic optimization of a complex computational system presents at least two more levels of problems for its designer. First, a particular class of optimization algorithms must be chosen that is suitable for that particular application. It is worth to remember that most optimization techniques are dedicated for parametric optimization, so a computational system must be described in terms of a fixed set of decision variables of acceptable types. Also the evaluation of each examined configuration may be a really time consuming job because it may require multiple system runs. Second, the configuration of the optimization algorithm needs to be tuned, so in fact the problem is repeated at the higher level of abstraction. Such approach in the field of optimization techniques is often called a *meta*-algorithm (compare the idea of meta-evolutionary approach [5]).

Indeed, it seems that automatic discovery of a system configuration for computational intelligence techniques is often an important but only auxiliary tool. The role of a human expert cannot be overestimated, because of a variety of available options and characteristics that describe the system behavior. Yet, it would be helpful, if an expert were equipped with an appropriately defined procedure and automatic system for testing, defining parameters, visualizing the results, lastly automatically searching for optimal parameters of the system.

Fig. 1. A general scheme of user-assisted management of a computational system

In Fig. 1 a general structure of management of a computational system is presented. In the simplest case it may be realized as a set of scripts run by some engine, based on the user recommendations. Several characteristics of the system may be monitored and possibly stored into a database. Then gathered data may be visualized and analyzed by the user, or processed by some optimization algorithm to obtain some estimation of the system quality. Both the user and analyzer module may use similar goal function in order to look for the best configuration of the system.

3 Management of an Agent-Based Computational System

Agent-based computational systems open possibilities of introducing on-line tuning strategies. These may be realized as specialized agents, which may perform analysis of the system behavior while it is still running. Of course because of assumed agents' autonomy they cannot directly control computing agents, but rather apply some indirect strategy based e.g. on resource management, which in turn may influence the behavior of computing agents.

For multi-agent systems building effective and efficient monitoring mechanisms is not an easy task. This is mainly because of the assumed autonomy of agents, but for computational systems also because of the number and variety of agents that produce huge amount of data, which quickly become out-of-date. Also problems of distribution and heterogeneity of agents are of vast importance. The proposed solution assumes local on-line processing of only required (subscribed) information via *monitoring services*, available both to the agents and external software tools via dedicated interaction protocols [6].

In a computational MAS acquisition of required information may be realized mostly by the core infrastructure, since it "knows" a lot about the agents' states. Thus in this case a monitoring subsystem should be tightly integrated with the agent platform. Fig. 2 shows a typical structure of a computational MAS together with a monitoring services

Fig. 2. Monitoring infrastructure for computational MAS

provider, which is a local authority responsible for management of all monitoring resources in a particular location (computing node). Since some directory of monitoring resources is indispensable to support processing and delegation of monitoring services, the monitoring services provider also delivers appropriate interfaces for agents of the system and external clients to facilitate identification of agents, their properties, and actual state.

The prototype implementation of the monitoring subsystem for a computational MAS was realized for *AgE* platform¹ — a software framework facilitating agent-based implementations of distributed (mostly evolutionary) computation systems. Monitoring services rely on the instrumentation of base classes equipped with the mechanism of properties and notifications, which may be used by monitoring services provider, according to *Observer* design pattern [7].

4 Evolutionary and Immunological Multi-Agent Systems

The idea of agent-based evolutionary optimization most generally consists in the incorporation of evolutionary processes into a multi-agent system at a population level. In its fine-grained model (EMAS – *evolutionary multi-agent systems*) it means that besides interaction mechanisms typical for agent-based systems (such as communication) agents are able to *reproduce* (generate new agents) and may *die* (be eliminated from the system). Inheritance is accomplished by an appropriate definition of reproduction (with mutation and recombination), which is similar to classical evolutionary algorithms. Selection mechanisms correspond to their natural prototype and are based on the existence of non-renewable resource called *life energy*, which is gained and lost when agents perform actions [8].

This shortly described approach proved working in a number of applications, yet it still reveals new features, particularly when supported by specific mechanisms, like immunological approach proposed as a more effective alternative to the classical energetic selection (iEMAS – *immunological EMAS*). In order to speed up the process of

¹ <http://age.iisg.agh.edu.pl>

selection, based on the assumption that "bad" phenotypes come from the "bad" genotypes, a new group of agents (acting as lymphocytes) may be introduced [2]. They are responsible for recognizing (assuming some predefined affinity function) and penalizing (by decreasing agent's energy or even removing an agent) agents with genotypes similar to the pattern possessed.

More thorough description of these ideas may be found e.g. in the referenced papers [8] [2]. Important here is that lymphocytes may be considered as autonomous agents performing on-line analysis of the system behavior (by searching for "bad" agents) and influencing computational agents (by penalizing them) in order to speed up the process of selection (see 3). Thus original concept of EMAS is not modified, instead newly introduced agents may be perceived as a means for automatic adaptation of computation.

It should be also stressed that the variety of parameters describe the system behavior in both EMAS and iEMAS – among the most important one may distinguish:

- number of demes (islands),
- initial number of agents on a deme,
- initial agents' energy,
- death energy level,
- duration of negative selection,
- agent rendezvous period,
- agent-agent evaluation rate,
- lymphocyte-agent similarity function coefficient,
- lymphocyte-agent penalization rate,
- lymphocyte prize for "bad" agent discovery.

In fact over 30 parameters need to be established for each particular run of the system, and many of them may prove important for the quality of the solutions obtained.

5 Experimental Results

The most important characteristics of the system under consideration is as follows:

- There are three fully connected demes, in every deme there are 20 individuals in the initial population.
- Agents contain real-valued representation of ten dimensional search space.
- Two variation operators are used: discrete crossover and normal mutation with small probability of macro-mutation.
- Lymphocyte contains a mutated pattern of the late agent and use the similarity function based on computing differences for corresponding genes to discover "bad" agents.
- Lymphocytes are rewarded for finding "bad" agents, and removed after a longer time of inactivity.

Below, the results of both introducing immunological mechanisms into EMAS and a user-driven tuning of EMAS and iEMAS parameters are presented. The tuning was

Fig. 3. Best fitness and population size (depending on step of systems' work) in EMAS and iEMAS for different initial number of agents (10–90)

realised using a set of scripts parameterized by the user. Every experiment was repeated 10 times. As the test optimization problem the Rastrigin function in 10 dimensions was chosen.

One of the main goals of the research was to check, how does the computation efficiency and accuracy in EMAS differ from iEMAS depending on different parameters. The efficiency was described by several important characteristics such as population size and the number of fitness function calls, and the accuracy was expressed in terms of the best fitness value. The tuning was based on arbitrarily chosen parameters: initial population size, rendezvous rate (EMAS and iEMAS), and negative selection period (only iEMAS).

First of all, parameters common to EMAS and iEMAS were explored. The results gathered for different values of initial population size are shown in Fig. 3. It is easy to see, that this parameter does not affect in a significant way the work of EMAS – the value of both best fitness and population size are very similar for all the values of the tested parameter. This is very important for stating, that there is no need to introduce hundreds of agents into a subpopulation in order to get reliable results.

Fig. 4. Best fitness and number of fitness function calls (depending on step of systems' work) in EMAS and iEMAS for different rendezvous period (1-7)

However, for iEMAS the situation is quite different. It may be clearly observed, that for high values of the parameter, iEMAS reaches better suboptimal solution of the problem. Characteristics of population size for EMAS and iEMAS differ, because the population in iEMAS is affected by the actions of lymphocytes, so it is important to optimize their parameters, otherwise they may remove promising solutions from the population.

The second parameter that was chosen for presentation is the length of rendezvous period (agents meet and evaluate themselves in intervals of such length). Best fitness and the number of fitness function calls for EMAS and iEMAS were examined (see Fig. 5). Comparing the graphs shown in Fig. 4(a) and Fig. 4(b) one can see, that introduction of lymphocytes leads to decreasing the efficiency of finding suboptimal solutions of the problem. Yet, the graphs shown in Fig. 4(c) and Fig. 4(d) prove, that though decreasing the accuracy, efficiency increases: there are fewer agents in iEMAS than in EMAS, therefore there are also fewer fitness function calls, the system seems to be more suitable for solving problems with complex fitness functions.

Frequent evaluations lead to increasing efficiency (better suboptimal solutions were found) in both systems (see Fig. 4(a), Fig. 4(b)), but the number of fitness function

Fig. 5. Best fitness, population size and lymphocyte matchings (depending on step of system's work) for different lengths of negative selection (5-35)

calls in this case was high (see Fig. 4(c), Fig. 4(d)), so there is a clear tradeoff between efficiency and accuracy. Usually such deliberation lead to assign medium values to the coefficients affecting these characteristics, yet a better answer could be given based e.g. on multi-criteria tuning strategy [9].

After stating that iEMAS may help in reaching better efficiency, and their parameters are crucial, search for optimal parameters of iEMAS was undertaken. One of the vital parameters of iEMAS is the length of negative selection. The results were shown in Fig. 5. Best fitness characteristic is significantly affected by changing the value of the observed parameter (see Fig. 5(a)) obtaining better values for longer periods. It is, because lymphocytes that are longer verified, do not remove good agents from the population. It may be verified by looking on Fig. 5(c) and Fig. 5(d) where count of mature and immature lymphocytes matchings was shown. Longer trained lymphocytes are often removed after meeting high energy agent, so the count of immature matchings is high. However mature matchings are almost independent on the length of negative selection, so the optimal value of this parameter should be medium, because lymphocytes should affect the population of agents, but they should not cause extinction.

6 Conclusion

In the paper problems of configuring computational intelligence systems were discussed, with particular attention paid to agent-based systems. Agent paradigm may facilitate the construction of hybrid systems, which was illustrated by the ideas of evolutionary multi-agent systems with immunological selection applied. These general considerations were supported by an experimental study on tuning of selected parameters of EMAS and iEMAS.

The process of system parameters tuning in the presented agent-based systems is performed on two levels:

- Inner, where certain subset of system's parameters is adjusted in the runtime. In iEMAS, specialized group of agents (lymphocytes) is introduced into the original population of agents. They influence the computation, affecting the structure of the population.
- Outer, where complete set of system's parameters is adjusted, according to the user's preferences and passed to the system before running. It may be done manually by the user, or automatically by some *meta*-approach, such as evolutionary algorithm.

In the course of the user-driven tuning of EMAS and iEMAS, several conclusions were drawn, which may help in future development of such systems:

- Change of starting population size did not affect the accuracy of EMAS, but it was important for iEMAS, which modifies the structure of the population. Better results were found in iEMAS for larger populations, but this parameter must be adjusted carefully, because too many agents in population will lead to high computation cost. There is apparently a tradeoff between accuracy and efficiency in iEMAS, and the lymphocytes clearly affect these aspects.
- In iEMAS because of introduction of lymphocytes, there was much fewer evaluations of fitness function, than in EMAS, so this approach seems good for the optimization problems with costly fitness evaluation. Observed parameters (rendezvous period) is closely bound to the cost of the computation in these systems.
- Negative selection length affected greatly the probability of mature or immature matchings among lymphocytes and agents. This parameter is crucial for immunological selection, and assigning improper value may lead to degeneration of the computation (too high value of this parameter de facto transforms iEMAS into EMAS, because lymphocytes do not have the possibility of modifying the agent population).

These conclusions were followed by more observations and experiments that could not be included in the paper, because of editorial limit. However they will constitute a starting point for further research, which should lead to proposing new strategies for on-line tuning of computational intelligence systems.

References

1. Kisiel-Dorohinicki, M., Dobrowolski, G., Nawarecki, E.: Agent populations as computational intelligence. In: Rutkowski, L., Kacprzyk, J. (eds.) *Neural Networks and Soft Computing*. Advances in Soft Computing, pp. 608–613. Physica-Verlag (2003)

2. Byrski, A., Kisiel-Dorohinicki, M.: Agent-based evolutionary and immunological optimization. In: Shi, Y., van Albada, G.D., Dongarra, J., Sloot, P.M.A. (eds.) ICCS 2007. LNCS, vol. 4488, pp. 928–935. Springer, Heidelberg (2007)
3. Bonissone, P.: Soft computing: the convergence of emerging reasoning technologies. *Soft Computing* 1(1), 6–18 (1997)
4. Vose, M.D.: *The Simple Genetic Algorithm: Foundations and Theory*. MIT Press, Cambridge (1998)
5. Freisleben, B.: Metaevolutionary approaches. In: Bäck, T., Fogel, D.B., Michalewicz, Z. (eds.) *Handbook of Evolutionary Computation*, IOP Publishing and Oxford University Press (1997)
6. Kisiel-Dorohinicki, M.: Monitoring in multi-agent systems: Two perspectives. In: Dunin-Keplicz, B., Jankowski, A., Skowron, A., Szczuka, M. (eds.) *Monitoring, Security, and Rescue Techniques in Multi-Agent Systems*, pp. 563–570. Springer (2005)
7. Gamma, E., Helm, R., Johnson, R., Vlissides, J.: *Design patterns: elements of reusable object-oriented software*. Addison-Wesley Professional (1995)
8. Kisiel-Dorohinicki, M.: Agent-oriented model of simulated evolution. In: Grosky, W.I., Plášil, F. (eds.) *SOFSEM 2002*. LNCS, vol. 2540, Springer, Heidelberg (2002)
9. Deb, K.: *Multi-Objective Optimization using Evolutionary Algorithms*. John Wiley & Sons (2001)