Agent-Based Models and Platforms for Parallel Evolutionary Algorithms*

Marek Kisiel-Dorohinicki

Institute of Computer Science
AGH University of Science and Technology, Kraków, Poland
doroh@agh.edu.pl

Abstract. The goal of the paper is to provide an overview of classical and agent-based models of parallel evolutionary algorithms. Agent approach reveals possibilities of unification of various models and thus allows for the development of platforms supporting the implementation of different PEA variants. Design considerations based on *AgWorld* and *Ant.NET* projects conclude the paper.

Keywords: Parallel evolutionary algorithms, evolution in multi-agent systems.

1 Introduction

Today *evolutionary algorithms* (EA) are used for more and more complex problems requiring large populations and long computational time (cf. e.g. [2]). Parallel implementations seem to be a promising answer to this problem, especially that evolutionary processes are highly parallel by nature. What is more, it turns out that some parallel models of evolutionary computation are able to provide even better solutions than comparably sized classical evolutionary algorithms — considering not only the quality of obtained solutions and convergence rate, but first of all the convergence reliability [4].

In the first part of the paper classical models of *parallel evolutionary algorithms* (PEA) are discussed. Then agent approach is proposed as a means to develop a unified model covering different variants of PEA. Two agent-based architectures of distributed evolutionary computation illustrate this idea. The paper ends with a short presentation of *AgWorld* and *Ant.NET* projects that aid realisation of both classical and agent-based models of PEA.

2 Classical Models of Parallel Evolutionary Algorithms

Evolutionary algorithms, as an abstraction of natural evolutionary processes, are apparently easy to parallelize and many models of their parallel implementations have been proposed [1,4]. The standard approach (sometimes called a *global parallelisation*) consists in distributing selected steps of the sequential algorithm among several processing

* This work was partially sponsored by State Committee for Scientific Research (KBN) grant no. 4 T11C 027 22.

M. Bubak et al. (Eds.): ICCS 2004, LNCS 3038, pp. 646–653, 2004.

units. In this case a population is unstructured (*panmictic*) and both selection and mating are global (performed over the whole population).

Decomposition approaches are characterised by non-global selection/mating and introduce some spatial structure of a population. In a *coarse-grained* PEA (also known as *regional* or *multiple-deme* model) a population is divided into several subpopulations (regions, demes). In this model selection and mating are limited to individuals inhabiting one region and a migration operator is used to move (copy) selected individuals from one region to another. In a *fine-grained* PEA (also called a *cellular* model) a population is divided into a large number of small subpopulations with some neighbourhood structure. Here selection and mating are performed in the local neighbourhood (overlapping subpopulations). It is even possible to have only one individual in each subpopulation (this is sometimes called a *massively* parallel evolutionary algorithm).

And finally there are also methods which utilise some combination of the models described above, or even a combination of several instances the same model but with different parameters (*hybrid* PEAs).

2.1 Global Parallelisation

In PEA with a global population selected steps of the sequential algorithm (mostly evaluation of individuals) are implemented in a distributed or multiprocessor environment.

In a *master-slave* model (fig. 1) one processing unit (*master*) is responsible for management of the algorithm and distribution of tasks to other processing units (*slaves*). This often consists in sending selected individuals to slaves to perform some operations (e.g. calculate fitness). In a *sequential selection* model a master processing unit waits for finishing computation in all slave nodes so there is a clear distinction between successive generations. This makes the implementation simple, yet the cost of idle time of slaves is a potential bottleneck of this method. In a *parallel selection* model a master processing unit does not wait for all slaves but when one finishes the work it is immediately allocated a new task. In this case selection may be done in several variants, for example in form of a tournament (so one needs to know only a subset of fitness values).

Conversely, in a system with *shared memory* all processors have access to all individuals. This approach has all advantages of the master-slave one without its handicap, yet requires some kind of control over possibly simultaneous operations on individuals.

2.2 Regional Models

Regional models introduce coarse-grained spatial structure of a population, suitable for implementation in a distributed architecture. Each node has its own (sub)population and runs a separate thread of evolution (fig. 2), thus conceptually subpopulations 'live' on geographically separated regions [9]. A new operator – *migration* – controls the process of exchanging individuals between regions. The model is usually described by a few parameters: a number of regions, a number of individuals in each region, as well as migration topology, rate/interval and a strategy of choosing individuals to migrate.

Migration topology describes how individuals migrate from one region to another. This often depends on software architecture and the most common are hypercube, ring,

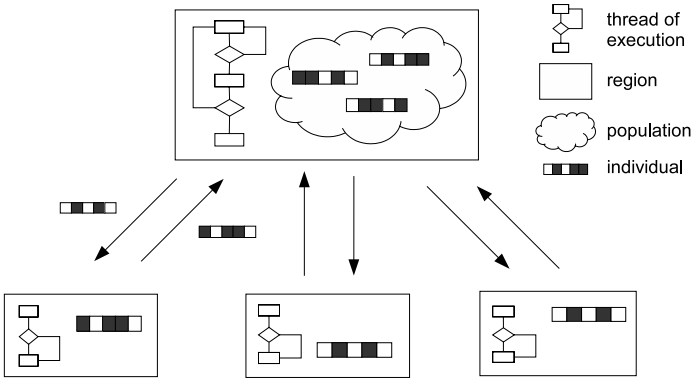

Fig. 1. A master-slave model of a globally parallel PEA

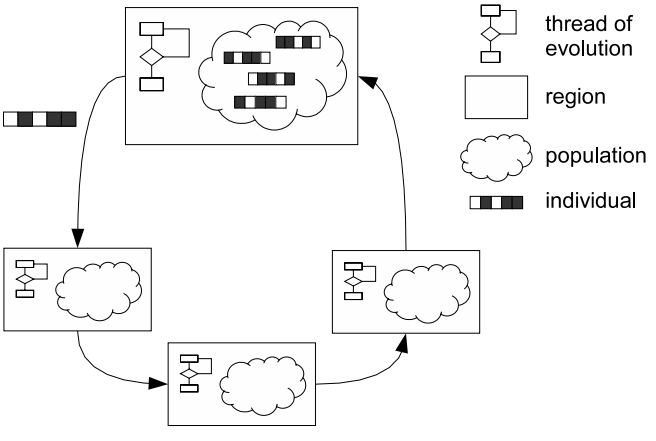

Fig. 2. A regional PEA – arrows indicate possible migration in stepping stone topology

or k-clique. In an *island* model individuals can migrate to any other subpopulation, while in a *stepping stone* model individuals can migrate only to neighbouring region(s). Migration rate and interval denote how many and how often individuals migrate. Of course migration rate should be greater if migration interval is longer. Typically the best individuals are chosen for migration and immigrants replace the worst individuals in a destination region. Other possibility is that immigrants replace the most similar individuals (e.g. using Hamming distance as a measure) or just replace emigrants.

2.3 Cellular Models

In cellular PEA a population has a fine-grained structure with neighbourhood relation defined [10]. In this case local selection/mating is performed over neighbouring individuals — subpopulations are not geographically separated but rather overlap and consist

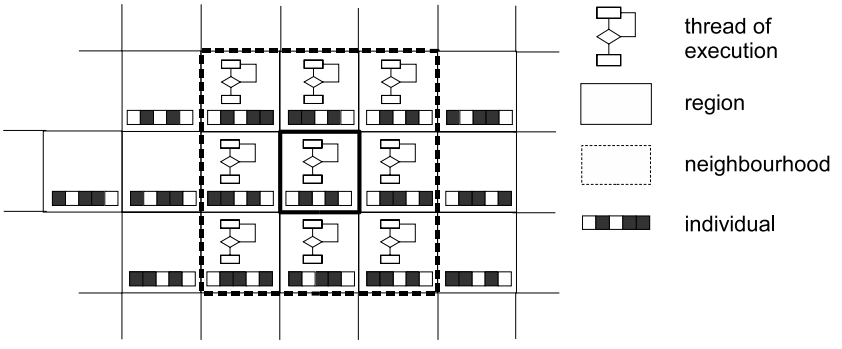

Fig. 3. A cellular PEA with local neighbourhood of central individual marked

of a few individuals (even only one individual). Usually the neighbourhood structure is a grid (each individual has a fixed number of neighbours) with a torus topology. A parameter is needed to describe a neighbourhood radius, which defines a direct neighbourhood of an individual where selection and mating is performed.

This model is strongly related to massively parallel computing. In such cases usually each processor contains one individual and is responsible for calculating its fitness and executing genetic operators. Neighbourhood topology is then determined by MPC architecture.

3 Agents for Parallel Evolutionary Computation

Designing complex evolutionary computation systems (e.g. consisting of a huge number of individuals or with spatially structured hybrid subpopulations) requires considering several aspects, that strongly affect both ease of building/modifying/tuning and computational properties (mainly efficiency). Recognizing common structure and identifying core functionality of such systems allows for development of universal software tools and libraries (platforms) facilitating their realisation. It seems that agent-based architectures may help a lot in this difficult task.

3.1 PEA Common Organisation

It seems that all (or almost all) classical and hybrid models of parallel evolutionary algorithms may be modelled using the following entities:

Environment – represents the whole system, main logical or physical aggregating unit used to directly manage regions, flocks or even individuals.

Region – plays role of a distribution unit, immobile entity, which may contain different kinds resources and may be used to directly manage flocks or individuals.

Flock – a unit of migration, mobile aggregating entity, which may be used to manage individuals.

Individual – an evolution unit and thus a basic element of the computation.

This structural model with four levels of organisation may be easily illustrated as a hierarchy of elements. Of course not every level of organisation is present in each particular technique or its purpose may be different, e.g.:

- in a *coarse-grained* PEA only **Regions** and **Individuals** are present,
- in a *fine-grained* PEA there is a geographically-structured **Environment**, and **Individuals**.

The hierarchical structure of hybrid PEAs can be organised in several ways revealing different possibilities of parallelisation.

3.2 Agent Systems and Evolutionary Computation

During the last decade the idea of an intelligent autonomous agent – a software entity, which is situated in some environment and autonomously acts on it so as to satisfy its own goals – gains more and more interest, and thus software agents are used in various domains [6]. At first sight evolutionary computation and agent approach seem to have nothing in common – the former is a search and optimisation technique, while the latter is a general concept of modelling decentralised systems. Yet one may notice that evolutionary processes are decentralised by nature and indeed multi-agent systems turn out to be a perfect tool for modelling them [7].

For each classical model of PEA alone the use of agents seems hardly advantageous – as an autonomous active entity, an agent may be identified with a region (coarse-grained PEA) or cell (fine-grained PEA). But considering a platform covering a wide range of different PEA models, agents may allow for a great deal of unification in the modelling scheme, no matter whether they would represent individuals or populations of them [3]. Depending on this decision two different agent-based architectures of evolutionary computation systems may be distinguished: in an evolutionary multi-agent system an agent represents a single individual, in a flock-based one an agent manages a group of individuals inhabiting one region (a flock). Of course in both approaches it is possible to realise classical models of PEA, but their strengths are manifested only for more complicated cases, e.g. for hybrid soft computing systems (cf. [8]).

3.3 Evolutionary Multi-agent Systems

In EMAS phenomena of *death* and *reproduction*, crucial for existing of inheritance and selection – the main components of evolution process, are modelled as agent actions:

- action of death results in the elimination of the agent from the system,
- action of reproduction is simply the production of a new agent from its parent(s).

Inheritance is accomplished by an appropriate definition of reproduction, like in classical evolutionary algorithms. Core properties of the agent (genotype) are inherited from its parent(s) – with the use of mutation and recombination. Besides, the agent may possess some knowledge acquired during its life, which is not inherited. Both the inherited and acquired information determines the behaviour of the agent in the system (phenotype).

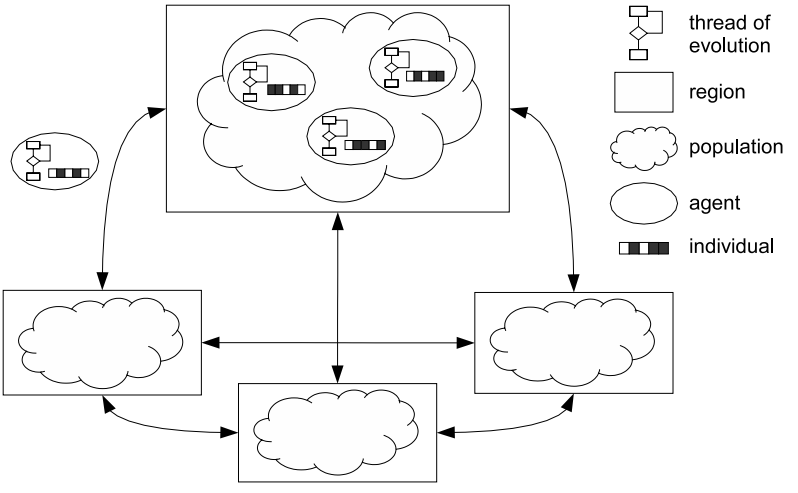

Fig. 4. Evolutionary multi-agent system

Selection is the most important and most difficult element of the model of evolution employed in EMAS [7]. This is due to assumed lack of global knowledge (which makes it impossible to evaluate all individuals at the same time) and autonomy of agents (which causes that reproduction is achieved asynchronously). The proposed principle of selection is based on the existence of non-renewable resource called *life energy*. The energy is gained and lost when the agent executes actions in the environment. Increase in energy is a reward for 'good' behaviour of the agent, decrease – a penalty for 'bad' behaviour (of course which behaviour is considered 'good' or 'bad' depends on the particular problem to be solved). At the same time the level of energy determines actions the agent is able to execute. In particular low energy level should increase possibility of death and high energy level should increase possibility of reproduction.

3.4 Flock-Based Multi-agent Model of Evolutionary Computation

A flock-based multi-agent system (FMAS) extends an island model of PEA providing additional organisational level. Subpopulations on the islands are divided into flocks, where independent processes of evolution (e.g. some classical sequential EA) are managed by agents. In such an architecture it is possible to distinguish two levels of migration, just like in *dual individual* distributed genetic algorithm [5]:

- exchange of individuals between flocks on one island,
- migration of flocks between islands.

Also merging of flocks containing similar individuals or dividing of flocks with large diversity allows for dynamic changes of population structure to possibly well reflect the problem to be solved (the shape of fitness function, e.g. location of local extrema). (Self-)management of agents-flocks may be realised with the use of non-renewable resources, just like selection in EMAS.

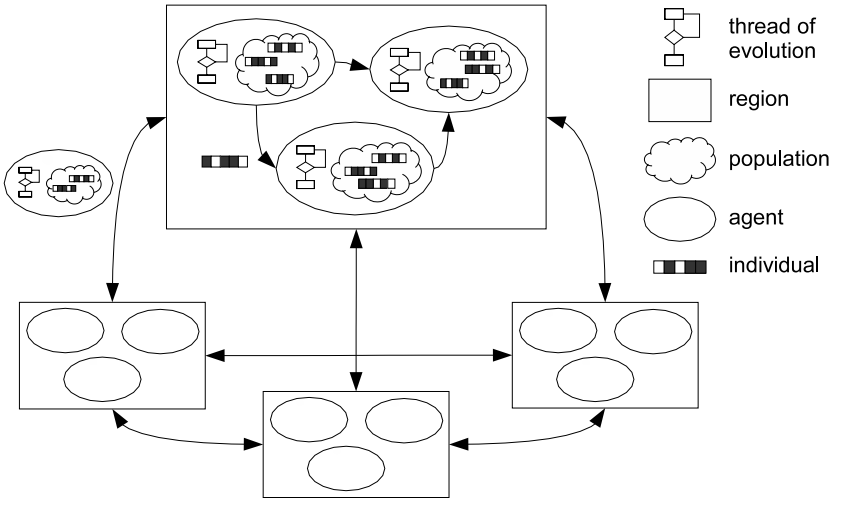

Fig. 5. Flock-based approach to multi-agent evolutionary computation

3.5 Agent-Based Platforms Supporting PEA Implementation

Based on the shortly presented models two software frameworks (platforms) were designed so as to aid realisation of various PEA applications in different domains [3]. Both platforms have similar logical organisation, yet differ in the technology used. The *AgWorld* project is based on parallel computation paradigm (PVM – Parallel Virtual Machine), whereas the *Ant.NET* project utilises software components (.NET) technology. Of course each platform (and technology) has its advantages and shortcomings.

Using *AgWorld* platform efficient parallel computation systems can be created, where groups of agents are placed in different tasks that can be run on individual computing nodes. Large systems, especially in a coarse-grained structure, can benefit much from using PVM this way.

Ant.NET proves useful for implementing applications requiring the use of sophisticated structures and dependencies between agents. The proposed architecture assumes that different aspects of the system (e.g. different selection or reproduction mechanisms) are implemented as software components that can be easily switched to change the behaviour of the whole system, or even a part of the system. Thus *Ant.NET* should become a good choice for implementing rather simulation systems, while efficient parallel computation requires more effort on .NET platform.

4 Concluding Remarks

Parallelisation of evolutionary algorithms is a conceptually easy task, since evolution is a parallel process by nature. Thus one may find lots of different parallel implementations of evolutionary computation paradigm, but still a unified model of parallel evolution and a universal platform allowing for comparison of all models is lacking. In the paper agent

approach was proposed as a possible solution to this problem – providing a unified model of evolution that became a base for development of two software platforms: *AgWorld* and *Ant.NET*.

Agent-based evolutionary computation also provides a more complex model of evolution and thus closer to its natural prototype. It should enable the following:

- local selection allows for intensive exploration of the search space, which is similar to classical parallel evolutionary algorithms,
- activity of an agent (individual phenotype in EMAS or subpopulation behaviour in FMAS) depends on its interaction with the environment,
- self-adaptation of the population size is possible when appropriate selection mechanisms are used.

However it still remains an open question whether agent-based models are more powerful then the these of classical (parallel) evolutionary algorithms considering their computational properties. This would be a subject of further research.

References

1. P. Adamidis. Parallel evolutionary algorithms: a review. In *Proc. of the 4th Hellenic-European Conference on Computer Mathematics and its Applications*, 1998.
2. T. Bäck, D. B. Fogel, and Z. Michalewicz, editors. *Handbook of Evolutionary Computation*. Institute of Physics Publishing and Oxford University Press, 1997.
3. A. Byrski, L. Siwik, and M. Kisiel-Dorohinicki. Designing population-structured evolutionary computation systems. In T. Burczyński, W. Cholewa, and W. Moczulski, editors, *Methods of Artificial Intelligence (AI-METH 2003)*. Silesian Univ. of Technology, Gliwice, Poland, 2003.
4. E. Cantú-Paz. A summary of research on parallel genetic algorithms. *IlliGAL Report No. 95007. University of Illinois*, 1995.
5. T. Hiroyasu, M. Miki, M. Hamasaki, and Y. Tabimura. A new model of distributed genetic algorithm for cluster systems: Dual individual DGA. In *Proc. of the Third International Symposium on High Performance Computing*, 2000.
6. N. R. Jennings, K. Sycara, and M. Wooldridge. A roadmap of agent research and development. *Journal of Autonomous Agents and Multi-Agent Systems*, 1(1):7–38, 1998.
7. M. Kisiel-Dorohinicki. Agent-oriented model of simulated evolution. In W. I. Grosky and F. Plasil, editors, *SofSem 2002: Theory and Practice of Informatics*, Lecture Notes in Computer Science. Springer-Verlag, 2002.
8. M. Kisiel-Dorohinicki, G. Dobrowolski, and E. Nawarecki. Agent populations as computational intelligence. In L. Rutkowski and J. Kacprzyk, editors, *Neural Networks and Soft Computing*, Advances in Soft Computing. Physica-Verlag, 2003.
9. W. Martin, J. Lienig, and J. Cohoon. *Island (migration) models: evolutionary algorithms based on punctuated equilibria*, chapter C6.3. In Bäck et al. [2], 1997.
10. C. Pettey. *Diffusion (cellular) models*, chapter C6.4. In Bäck et al. [2], 1997.