DINA: Dynamic INtelligent Agents

Kovács, Dániel László
Budapest University of Technology and Economics
Department of Measurement and Information Systems
Budapest, H-1117, Magyar tudósok körútja 2., Hungary
dkovacs@mit.bme.hu

Abstract ***– Dynamic Intelligent Agents are agents capable of intelligent behavior in any chosen dynamic environment. A new agent architecture is proposed, revising the classical concepts of system-environment relation, thus agents capable of "optimal" exploratory behavior and heterogeneous cooperation can be created. At first an overview is presented, followed by the description of a possible implementation tested on a set of micro-world test cases. Finally conclusions are drawn concerning the effectiveness of the concept.***

Keywords ***– Artificial Intelligence, Genetic Algorithms, Intelligent Systems, Planning, Problem-solving***

I. INTRODUCTION

An *agent* is an autonomous, adaptive entity situated in an environment. *Intelligent* means that it consistently tries to expand its view (to gain knowledge) of its surroundings and to understand the relations to and within its environment. *Dynamic* environments have their own tendencies, ongoing processes, i.e. activities independent of the agent (other agents for example).

The paper presents an idea of how to relate the mentioned concepts in a model that is general enough, yet still applicable. For this reason the hitherto used agent-environment interaction schemes are reinterpreted. They are expanded in a way by revising the classical decision making mechanism and mental representation of the environment.

Agents are important topic because of the growing need for intelligent applications both in scientific and daily activities [1]. The need for intelligent tools is essential in fields that are hardly manageable or even inaccessible for humans (i.e. space or sea exploration, managing complex systems). Assisting human activity with more effective machineries is also a well-known issue. A harmless example could be the recent intelligent vacuum cleaner, the Internet search engines, or the like.

The goal was to develop a general theory [2] that makes it possible for agents to achieve real *intelligent behavior*. The paper first introduces the general concepts of Dynamic Intelligent Agents (DINA), proceeding with the description of a possible (currently Prolog-based) implementation. Then its functionality is illustrated on a set of test cases with final conclusions concerning the effectiveness of the approach.

II. AN OVERVIEW OF DINA

In our approach the concept that connects intelligent agents with dynamic environments is *evolution*. Agents evolve in the environment, making decisions, acting, and gaining or loosing accordingly to a particular concept of fitness. They elaborate their optimal decisions by investigating alternatives via the internal evolution of their own models - alter egos - in possible worlds extrapolated from the actual observable state of the environment. This means, that not only environments can evolve agents – agents themselves can also evolve their so-called "thoughts" analogously to concepts of memetics [3]. The assumption, that evolution is both responsible for the dynamics of environments and for the intelligence of agents is plausible, when being compared to the empirical notion of human evolution. Besides, self-similarity [4] is also a part of nature and potentially fruitful to provide analogies. Evolution within evolution is (we presume) the main concept holding the key for real *intelligent behavior,* i.e. a*rtificial intelligence*.

Talking about evolution not in a biological, but rather in a formal, technical way implies the mention of *genetic algorithms* [5], stochastic search methods proven to be globally convergent [6]. Consequently, by using algorithms based on genetic algorithms, i.e. *genetic programming* [7] or *gene expression programming* [8], it is possible to construct architectures that realize the above mentioned evolution in evolution. Due to the proven convergence, globally "optimal" evolution can be achieved enabling the system to realize e.g. "optimal" exploration planning [9].

III. THE FRAMEWORK

First the intended meaning of some of the most important terms is clarified:

- **Evolution** is a process, which evolves elements of the same type within an environment using natural means of selection.
- An **environment** is a problem-space representing a task, where the solution of the problem(s) means the solution of the task.
- An **agent** is an adaptive, problem solving entity, which is a part of an evolution and attempts to solve a task.

Now the general concept of DINA can be grasped as:

Given an environment, i.e. a problem-space, where a population of agents is being developed via an unsupervised evolution, every agent evolves its best action via an inner, supervised evolution.

To define a suitable architecture the general concepts must be embedded in the classical framework of agent-environment model (Fig. 1):

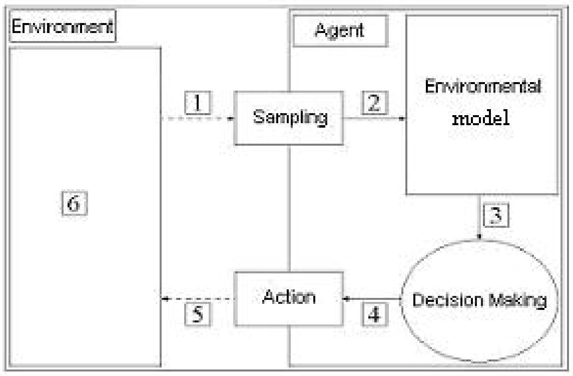

Fig. 1. General framework of DINA

An agent (an intelligent system) is embedded in an environment, where – being a part of an unsupervised evolution (competing for resources to gain advantage over others) – it thus attempts to solve a task represented by its current environment. For this purpose it samples the environment (1) creating its inner representation (2), which is then used by a decision-making mechanism (3) to produce an action (4) executed toward the environment (5), modifying its present state (6). An agent continues this loop, until it stops by itself (e.g. runs out of resources, etc), or is externally interrupted.

A. Inner and outer representation

We gain a more detailed view by looking at the environment as if it would be an *"outer representation",* its image an *"inner representation"* and the decision-making mechanism a *"program evolution"* based on genetic algorithms. Without knowing the generality, we can assume that inner and outer representations are collections of facts represented by logic statements.

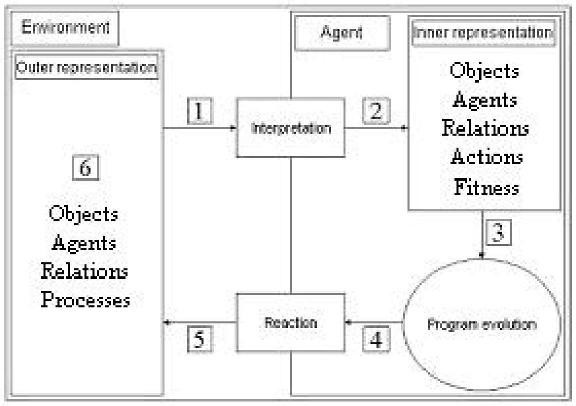

Fig. 2. Detailed framework of DINA

The inner representation of the outer representation is its *interpretation* (Fig. 2). It is a *model* of the outer representation (the environment) *"thought to be possible".* Outer representation consists of objects, other agents, independent environmental processes and relations between them. The first step is the interpretation of the outer representation. An agent senses the environment (1) and then constructs its inner representation (2), which models facts sensed in the outer representation. Also a *fitness function* is created, which is used to evaluate conditional action-chains, i.e. conditional plans or programs built of models of possible elementary actions. A population of such programs is evolved by using the fitness function (3). The best program evolved by the *program evolution* (4) is executed as a reaction toward the environment (5), modifying its present state (6).

B. Local and global reality

Inner and outer representations are rarely equal in content and expressive power. Usually only a "surface" of the latter is "projected" into the former depending upon the specific realization of the agent's sensors. Therefore agents can only use a part of the outer representation to construct its inner variant: the part, which can be sensed by them. This leads us to a more accurate model:

- The complete environment, i.e. the outer representation is the **global reality** of an agent.
- An agent's perception of the global reality is its **local reality**.

An agent perceives global reality through its local reality. There may be several global realities producing the same perception, i.e. the same local reality. Consequently agents construct alternative global realities, here called **fantasies** depending upon experience and perception. Fantasies are global realities *"thought to be possible"* (similarly to the possible worlds in models of various modal logics [10]). Meanwhile the agent initializes a population of programs built of models of possible elementary actions. Fantasies are then used to evaluate the goodness of these programs. At first the agent places every individual of the program population into every fantasy. It executes them calculating their average utility on all the fantasies depending on their effect, their operational activity or any other of their attributes. This value is then used as the fitness value of the given program. A program being executed in a fantasy is called an **agent's double** because it is also a representation of an agent itself. The best agent-double evolved is chosen to be executed by the agent. This concept is shown in Fig. 3.

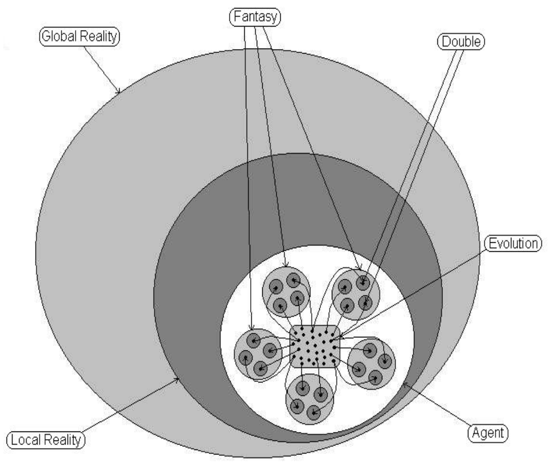

Fig. 3. Agent-environment relation in detail

Every agent has a (learnt) model – depending on its *"experience"* – of how its sensors and effectors work, i.e. of how global reality affects its local reality and of how actions affect global reality. This experience is a basis of the *"self-simulation"*, i.e. of generating fantasies and running agent-doubles. Experience is collected through time by continually sensing a local reality. Both experience and the models (deduced from it) are part of the inner representation, which is a kind of knowledge base of the agent.

C. Linear and non-linear memory

Agents can have different representations of their experience. They can construct a *linear memory* collecting their previously executed actions in a list of <*When, Where, What*> statements, where *When* means the time the action was accomplished, *Where* means the local reality sensed at the moment and *What* is the description of the action itself. Moreover it is possible for agents to also have a (very different) *nonlinear memory*, which is a collection of <*Where, What*> statements with *Where* now meaning an agent's position in its inner representation where *What* (sensed local reality) was felt. While linear memory is a list of executed actions, non-linear memory is an implicit map of the global reality. Using nonlinear memory and models describing sensors and effectors, agents can construct fantasies enabling them to proceed with evolving their actions. For understanding this process we must first clarify the meaning of models for knowledge representation.

D. Models for representation of knowledge

By *models* here we mean collections of (not necessarily grounded) logical statements and logical constraints describing relations between the attributes (variables) of the statements. An example is shown in Fig. 4, where two sets of such statements *(p* and *q)* are related to each other by using logical constraints on the values of their variables.

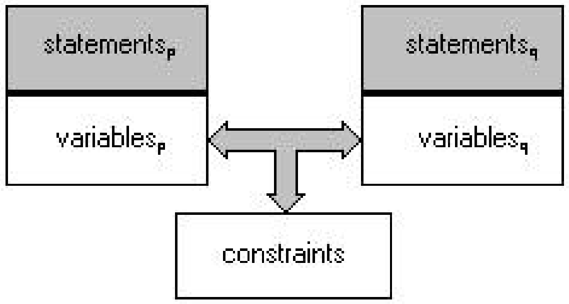

Fig. 4. An example of a relational model

For example, given a fully grounded statement describing the current state of the local reality and a partially grounded statement describing the current state of the inner representation, an agent can possibly instantiate (or just narrow the domain of) some variables in the inner representation, thus concluding some facts about the current state of the (assumed) global reality.

By the use of such models describing the connection between actions and their effects on the state of the global reality, the relations between (and inside) the state of local and global reality, agents are able to generate fantasies and to properly represent themselves in them via agent-doubles.

E. Agent-doubles and Fantasies

Agent-doubles, when executed in a fantasy, are provided only with a *"local fantasy"*, as if the fantasy would be their global reality. They are also provided with all the experience gathered by the agent, i.e. they have both the agent's linear and non-linear memory enabling them to properly represent the agent. That way agents *"simulate the world and* *themselves within"* via agent-doubles being embedded and executed in fantasies.

Fantasies are generated the following way. From a non-linear memory agents can deduce facts about the assumed global reality (inner representation) by using logical constraints, i.e. by instantiating (or narrowing the domain of) some variables in the statements describing the inner representation. **Fantasies** are all of the possible instantiations (considering the constraints) of this inner representation.

When creating fantasies agents do not consider those parts of the global reality they haven't yet felt. This way they avoid the problem of omniscience. Only those facts of the global reality appear in their fantasies, which had effect on their local reality. Fig. 5 presents the relation of these concepts:

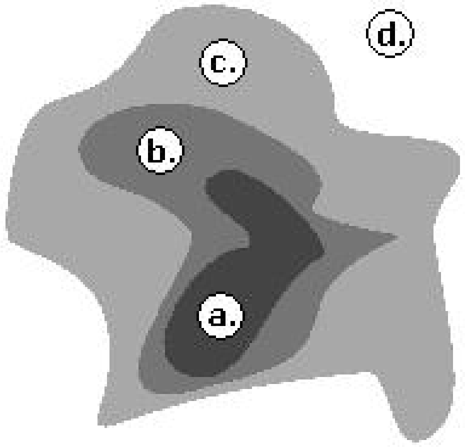

Fig. 5. Global reality and its inner representation

Facts about the global reality that are known by the agent (a.) appear in the inner representation as grounded logical statements, and thus every fantasy contains them. These are statements which variables had been instantiated by the constraints of the models. Facts that were sensed, but are still uncertain (b.) appear as partially grounded statements, and so fantasies are the possible instantiations of these statements. The knowable facts that weren't yet sensed by the agent, but can possibly be sensed as a result of an action sequence (c.), do not appear in the inner representation or in any fantasy (at the moment of sensing). Facts that cannot be sensed (d.) will never appear in the inner representation or in any fantasy.

F. Gene Expression Programming

The only theoretical aspect of DINA still not covered is the way agents and their doubles are coded and evolved. Gene Expression Programming (GEP) [8] (an extension to Genetic Programming (GP) [7]) is used to realize both the evolution of agents and their doubles. It is used because it is efficient and fits well the philosophy of the approach.

Agent-doubles are represented with multi-genic GEP chromosomes, where every gene is a coded decision tree evaluated with a TOP-DOWN method.

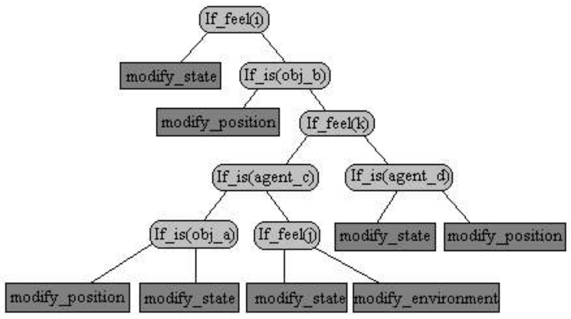

Fig. 6. Example of a decoded gene

Fig. 6 shows a decoded GEP gene, actually a decision tree with conditions as its inner vertices and actions as its leaves. When evaluating such a decision tree with a TOP-DOWN method only one leaf is activated depending on the conditions present, and thus every gene codes a complex conditional-action (a program). Vertices are elementary actions, which are of three different types: perceptive, cognitive and effective actions. Effective actions modify the state of the environment (global reality). Perceptive actions are decisions based upon the current local reality. Cognitive actions are decisions based upon experience.

By evolving chromosomes consisting of genes built of such elementary actions, conditional action-chains (i.e. programs or plans) are evolved, where decisions are realized by perceptive and cognitive actions. This way chromosomes code conditional plans, i.e. agent-doubles. Agents are coded with multiple chromosomes, where every chromosome is responsible for a different aspect of the agent's architecture and behavior. Some of them can code production-rules that realize the interpretation-mechanism; others can be responsible for parametrizing the inner evolution, etc.

By evolving agents and because of a globally convergent computational model of evolution [5], intelligent systems capable of designing "optimal" plans for dynamic environments can be evolved. From the point of view of planning this means, that theoretically the hardest problems, i.e. exploration problems can be "optimally" solved. Exploration problems [9] are problems afflicted by inaccessible environments, lack of knowledge and possibly faulty sensors and effectors. The agent, i.e. the exploration planner doesn't know at first the effects of his actions, but explores them by operating in the environment. The evolution of such agents produces an "optimal" exploration planner, which is able to appropriately learn the model of its environment and thus the effects of its actions. Because of a globally optimal inner evolution, the agent is able to produce contingency plans for "optimally" solving the problem posed by the environment.

IV. IMPLEMENTATION

The DINA is currently Prolog-based. Sicstus Prolog 3.10.0 [11] was used for implementing the concrete architecture with the extension of the "Constraint Logic Programming on Finite Domains" (CLP(FD)) library [12] for creating relational models, "Definite Clause Grammars" (DCG) [13] for realizing production-rules (responsible for agents' structure and behavior coded by their GEP chromosomes), and the "Tool Command Language and Tool Kit" (TCL/TK) library for creating a Graphical User Interface (GUI).

The particular implementation of the general architecture is not complete, and is problem specific to facilitate the testing. The interpretation mechanism of agents works with a complete model of the environment, i.e. the models representing inner and outer representations are equivalent, and so an agent mustn't yet explore the effects of its actions, etc. Consequently, only contingency problems can thus be solved generally in an "optimal" way. The implementation is problem (or domain) specific in a sense, that it was implemented only for some test-worlds.

V. TESTING

The DINA was tested in several micro-worlds. The so called Wumpus-world [9] and Table world [14] were used to estimate the goodness of the approach. Though seemingly simple, even these grid worlds hold a very special challenge for intelligent agents.

A. Wumpus world

In Wumpus-world agents must confine themselves to a relatively simple local reality. They have only five binary feelings, nonetheless they must survive in a global reality that is much more complex and even dangerous. This difference between the complexity of the global and local reality makes it hard for them to succeed, i.e. to eliminate all Wumpuses, to find all the gold and to escape via the entry point. Tests have shown, that DINA was comparable to human performance almost in every test. Fig. 7 shows a typical example of a 5^x3 Wumpus world, where the agent (lower left corner) can only feel perceptions depending on the state of the neighboring four squares.

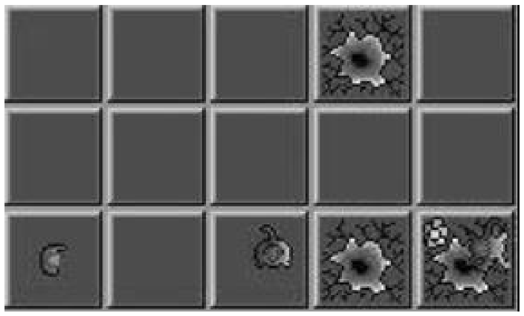
Fig. 7. An example of Wumpus world

Smell is felt, if a Wumpus (middle of lower row) is close by. Similarly wind is felt, if an abyss is in the neighborhood. Both are deadly for the agent to encounter. Agents have a limited number of shots to eliminate Wumpuses, which scream if being shot. A scream can be heard in the whole Wumpus world. If an agent finds gold by stepping onto it, then it can see it shine. Also an agent can feel a push, if being crashed into a wall.

B. Table world

In Table world, though the difference between global and local reality is smaller, agents have to cooperate to accomplish the task. Tests have shown, that even a heterogeneous cooperation could be achieved, i.e. a cooperation emerging from different agent-behaviors, where agents were able to work together on a task by executing different plans. Fig. 8 shows a typical example of a Table world, where A_i are the agents that have to bring table T to its destination D.

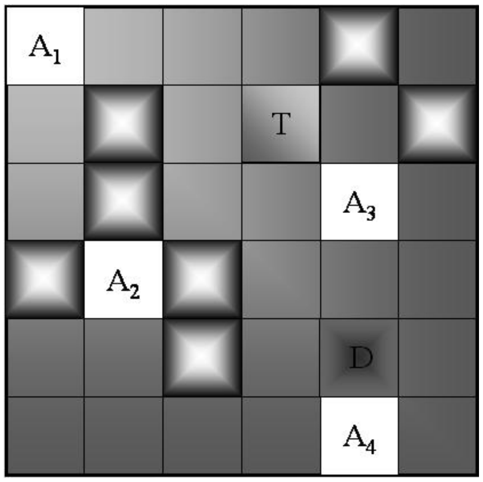

Fig. 8. An example of Table world

The table is too heavy for a single agent, so agents have to cooperate (by homing around the table and bringing it down to its destination) to accomplish the task.

VI. TEST RESULTS

Despite the fact that large computational resources were needed for relevant testing (checking all parameter combinations) the effective AND-parallelism of the approach enabled to run different tests, agents, fantasies or even agent-doubles simultaneously. This way the time for a relevant testing was reduced drastically, i.e. from months to days.

In Table world agents were competing (or cooperating) with each other. Agent-doubles were coded with two-genic chromosomes, which – representing only two plan steps – were "lengthened" by executing them cyclically, i.e. they actually coded cyclic plans. Every agent had four-four (same) agent-doubles in its fantasies, where every gene of the chromosome (thus responsible for the joint behavior of the agent-doubles) was evaluated with a different fitness function. One gene was responsible for homing, the other for herding. A composition of these fitness functions was used to calculate the overall fitness value of the chromosome (depending on the collective activity of the four agent-doubles). Thus, by evolving agent-doubles, homogeneous cooperation was achieved, i.e. agent-doubles cooperated by using the same plan. Agents evolving their actions thus separately achieved heterogeneous cooperation.

In Wumpus world human adversaries were asked to compare their skills against the approach. Several tests were made. Human participants played separately, once on every test. After a human player finished, an agent was tested with the same initial conditions. The number of steps made by the human opponent limited the agent's steps. Fig. 9 shows a comparison of "thinking" times in a typical test situation.

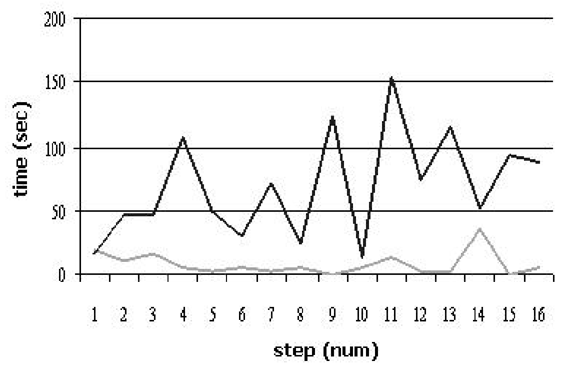

Fig. 9. "Thinking"-times of a typical Agent vs. Human test

The dark line shows the "thinking" times of DINA, the light line shows the thinking-times of a human opponent. There seems to be no correlation between them because, in contrary to humans, agents weren't yet "remembering" the solutions evolved in previous steps.

Fig. 10 shows a summary of the average performance of DINA compared to the performance of human players in Wumpus world. DINA resulted in less deaths, it collected a similar amount of gold, eliminated more Wumpuses, but somehow didn't manage to escape as well, as human players. This can be traced to a fault in the current implementation.

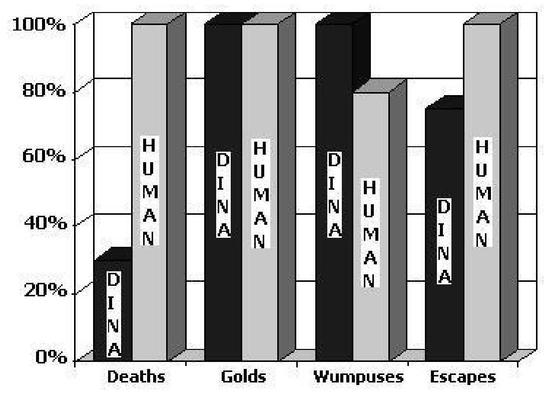

Fig. 10. Average results of Agent vs. Human tests on Wumpus world

VII. CONCLUSIONS

A new approach to design a competitive AI system was introduced. The main idea, i.e. two-level evolution was applied to planning. One evolutional level was responsible for "optimal" contingency planning, while the other (meta-level) was responsible for evolving such planners (extended with the ability to explore their environment), thus producing an "optimal" exploration planner. Although there is no formal description (or verification) of the agent-architecture yet, simulations show convergence and improvement in the utility of the produced solutions.

Future investigations are oriented at making the description of the approach more formal, mainly the interpretation mechanism and the languages describing the models of the environment and agents. A full, problem independent implementation is planned, with real-world tests, where DINA could be responsible for the actions of a robot situated in the physical environment.

It can be concluded, that a general, robust and scalable heuristic method was found for solving the difficult task of "optimal" exploration planning, i.e. the task of intelligent exploratory behavior in any chosen environment. Moreover, test results have shown that it is possible to achieve heterogeneous cooperation as an emergent behavior with a performance comparable to human. These benefits make DINA a promising alternative for solving complex tasks concerning future AI research.

REFERENCES

[1] IBM, Journal on Pervasive Computing, vol. 38, no. 4, 1999 URL: www.research.ibm.com/journal/sj38-4.html
[2] D. L. Kovács, "Evolution of Intelligent Agents: a new approach to automatic planning" in Proc. of the IFAC Workshop on Control Applications of Optimization (CAO'2003), Visegrád, Hungary, 2003.
[3] R. Dawkins, The selfish gene. Oxford University Press. 1989.
[4] M. F. Barnsley, Fractals everywhere. Academic Press. 1988.
[5] D. E. Goldberg, Genetic Algorithms in Search, Optimization, and Machine Learning. Addison-Wesley. 1989.
[6] J. H. Holland, Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence. MIT Press. 1975.
[7] J. R. Koza, Genetic Programming: On the Programming of Computers by Means of Natural Selection. MIT Press. 1992.
[8] C. Ferreira, "Gene Expression Programming: A New Adaptive Algorithm for Solving Problems", Complex Systems, vol. 13, no. 2, pp. 87-129, 2001.
[9] S. J. Russell and P. Norvig, Artificial Intelligence: A Modern Approach. Prentice Hall. 1995.
[10] P. Blackburn, M. de Rijke and Y. Venema, Modal Logic, Cambridge University Press, 2001.
[11] P. Szeredi and T. Benkő, Introduction to logical programming. Budapest University of Technology and Economics Press. 2000.
[12] P. Van Hentenryck, Constraint satisfaction in logic programming. MIT Press. 1989.
[13] A. Colmerauer, Les Grammaires de Metamorphos, Technical Report, Groupe d'Intelligence Artificielle, Marseille-Luminy, Nov. 1975.
[14] Zhang Byouk-Tak and Cho Dong-Yeon, "Co-evolutionary Fitness Switching: Learning Complex Collective Behaviours Using Genetic Programming", Advances in Genetic Programming, vol. 3, pp. 425-445, 1999.