Jumping Genes-Mutators Can Rise Efficacy of Evolutionary Search

Alexander V. Spirov

The Sechenov Institute of Evolutionary Physiology and Biochemistry, 44 Thorez Ave., St. Petersburg, 194223, Russia

and

Dept. of Applied Mathematics and Statistics, The State University of New York at Stony Brook, Stony Brook NY 11794-3600, USA

Email: spirov@kruppel.ams.sunysb.edu

fone: 631-632-8370

fax: 631-632-8490

Alexander B. Kazansky

The Sechenov Institute of Evolutionary Physiology and Biochemistry, 44 Thorez Ave., St. Petersburg, 194223, Russia

Email: kazansky@iephb.nw.ru

fone/fax: +7 (812) 552 3219

Abstract

Genetic Algorithms (GA) and Genetic Programming were inspired by ideas from evolutionary biology. However modern Evolutionary Computation (EC) only in outline reminds the strategies of biological evolution. The application of other algorithms and biological ideas may substantially improve the performance of this area of computer science. Namely, the selfish (or parasitic) mobile genetic elements - transposons are good candidates for this breakthrough. These genomic parasites live on a substratum of genomes of whole biological communities. Many biologists assume that processes in the world of transposons are the main source of evolution creativity. They thought to act as wise higher-level mutators for their hosts. In this communication we propose a strategy of construction of a new approach exploiting the most essential aspects of co-evolution of the hosts-chromosomes and their genetic parasites. We named this strategy as the Two-level Evolving Worlds. The key feature of the approach is usage of artificial transposons. We apply it to one of known benchmark problems - the John Muir ant's trail test. We found that our enhancement of GA technique by the artificial transposons obviously increase the efficacy of searching of the ant's navigation algorithm. We investigate in details the way of the transposons action as intelligent mutators of host-chromosomes.

1 INTRODUCTION

Many areas of evolutionary computation, especially genetic algorithms (GA), and genetic programming (GP), are inspired by achievements in genetics and evolutionary biology. However modern evolutionary biology has since advanced considerably, revealing that genes are not simply parameter settings, but components of a complex biochemical machine (Cf. Luke et al., 1999; Lee and Antonsson, 2001; Lones and Tyrrell, 2001).

On the other hand, many branches of modern evolutionary computation research are aimed at evolution of mechanisms (neural networks, decision trees, cellular automata, L-systems, finite state automata). For these domains, recent genomic achievements seems more appropriate as an inspirational model then classic set of Darwinian algorithms.

There is a feeling that the field of EC is getting more inspired with the latest achievements in biology, trying to make the evolutionary algorithms more effective. Such techniques as transposition, host-parasite interaction, gene-regulatory networks and some others have yet been applied to EC.

•*Host-parasite methods*: These methods are based on the co-evolution of two different populations, one of them acting as “parasite”, and the other acting as “host”. The parasites usually encode a version the problem domain, and the hosts the solution to the problem (Hillis, 1990; Potter and De Jong, 1994; 1995; De Jong and Potter, 1995; Olsson, 1996; 2001).

•*Transposition operators* (“*bacterial” algorithms):* The basic idea of these approaches is to make intra-chromosome crossovers, that is, crossover of a

chromosome with another part of itself, or else asymmetric crossover, in which a donor chromosome transfers part of its genetic material to an acceptor chromosome (Harvey, 1996, Nawa et al., 1996; Simoes and Costa, 2001). In some cases, these operators seem to be better than classical genetic algorithms for combinatorial optimization problems.

•*Gene-regulatory networks approach*: Luke et alls (1999) use a method similar to genetic regulatory networks to evolve finite state automata that represent a language grammar. It is appropriate also to mention here the Burke et alls (1998) project, as well as "enzyme genetic programming" (Lones and Tyrrell, 2001).

•*Evolution based on the selfish elements*: Corno et alls (1998) implemented the *Selfish Genetic Algorithm* inspired by Dawkins concept of the *selfish gene*. The algorithm evolves a *Virtual Population*, in which alleles compete for appearance in their respective locus in the genotype.

So far, it has not been found in the literature a technique that is general enough to be applied to a wide range of problems, and that, in some cases, is able to yield as good or better results than evolutionary algorithms

This stimulates us to search for prospective mechanisms that simulate the creative, heuristic and self-organizing character of (biological) evolution (Spirov, 1996a; 1996b; Spirov and Samsonova, 1997; Spirov and Kadyrov, 1998; Spirov et al., 1998; Spirov and Kazansky, 1999). The mobile selfish genetic elements (synonymous or related terms are jumping genes, transposons, retroviruses) are good candidates for this breakthrough (Makalowski, 1995). Many biologists speculate that processes in the world of transposons, living on a substratum of genomes of the whole biological communities, are the main source of macroevolution creativity (Doolittle and Sapienza, 1980; Orgel and Crick, 1980; Brosius, 1991).

In this connection, special interest is attracted by well-known examples of both competitive and cooperative strategies in populations of transposons.

In this communication we propose a strategy of construction of a new approach exploiting the most essential aspects of co-evolution of the hosts-chromosomes with their genetic parasites. We named this strategy as the *Two-level Evolving Worlds*. The key feature of the approach is usage of *artificial transposons*. We treat transposons as high-level and intelligent mutators. In the next part we give the definition of the strategy. To demonstrate the efficacy of a new approach we apply it to one of known benchmark problems - the John Muir ant's trail test (Jefferson et al. 1992; Koza, 1992).

1.1 THE TWO-LEVEL EVOLVING WORLD

Parasites and parasite ensembles always accompany biological evolution. Tom Ray simulated this process in his *Tierra* (Ray, 1991).

A special kind of parasites is genomic parasites living in the host genome. Known biological proverb says that "*the viruses in all of us - the viruses that make us*".

In the course of evolutionary time, parasites form "community" of their own. They populate the united genomic space of many hosts. We shall name these parasites as *InfoParasites (IP)*, and the "community of the parasites" as *IP world*.

There are examples of evolvable virtual worlds such as Swarm, Creatures, Network Tierra (Daniels, 1999; Cliff and Grand, 1999; Ray, 2001). In the course of evolution the worlds of that type can split over IP and host co-evolving worlds, i.e. they can become the two-leveled. It is the question of time and such worlds' complexity. In less complex virtual worlds similar splitting could be realized "by hand", as in the case of developing world of computer viruses.

1.1.1 Strategy of Development of The Two-level Worlds

We assume that the simplest realization of the two-layer evolving worlds would be as follows:

the hosts-world is GA-like system (standard GA in the simplest case). The manifold of hosts' chromosomes-strings is the environment for IPs. In the simplest case these GAs don't have any mutation operators of their own;

the InfoParasites are the LISP-like programs, manipulating with the hosts' strings. (For our applications these programs must include the SEARCH function performing the search of patterns in the host strings). IPs live in hosts, they are transmitted vertically (when host reproduces) and horizontally (from one host to another, as infection or computer virus);

genotypes of parasite and host are encoded by the same text, i.e. the same string of symbols is interpreted in two different languages, the host's and the parasite's one;

"bad" (too harmful) parasites are eliminated together with their hosts, "good" parasites minimize their harmfulness (for example, by exploiting unessential parts of host's chromosomes).

1.1.2 Intelligent Mutators

IPs acts as intelligent and sophisticated mutators. They can generate arbitrary procedures of manipulations with

hosts' chromosomes. In general, these operators can be the unitary, binary or plural ones. Each host has got the mutators of its own. In the simplest case IPs are the only source of the host's mutations.

If IP founds hopeful mutation strategy, then both host and parasite will get chance for reproduction, the parasite rides on a new turn of evolution on the transformed host. Virtually we have co-evolution of hosts and their intelligent mutators-parasites.

1.2 THE ARTIFICIAL ANT PROBLEM

The artificial ant problem is the simulation of an ant navigation aimed at passing through the labeled trail placed in a grid world (Jefferson et al. 1992; Koza, 1992). The trail was nicknamed as "The John Muir Trail" in the UCLA experiment (Jefferson et al., 1991). Each labeled cell is numbered sequentially, from the 1st which is settled directly next to the starting cell, through to the last cell. The ant's task is to pass through the labeled cells one by one (the more the better) for the limited time period. The ants are simple finite-state automata or an artificial neural network, which can move along the grid world and test their immediate surroundings. The trail starts off quite easy to follow, and gradually gets more difficult, as the turns become more unpredictable and gaps appear (See Fig.2). Therefore, the successful ant's program must be quite sophisticated. The problem has been repeatedly used as a benchmark problem (For references See Langdon and Poli, 1998).

2 METHODS AND APPROACH

While the ant test was implemented at least in two different C++ libraries (Zongker and Punch, 1995), we gave preference to the Peter Brennan's version (Brennan, 1994). This "ANT program" was designed in such a way that to isolate, as far as possible, the components of the genetic algorithm from the trail-following experiment and the ant representation. Brennan's ants are finite state automata.

2.1 TECHNIQUE OF MOBILE GENETIC ELEMENTS - TRANSPOSONS

Mobile Genetic Elements (MGEs) - transposons are akin to computer viruses. They are the autonomous programs, which are transmissible horizontally (viz., from one site to another one on the same or another chromosome) or vertically (from the ancestor to the descendants in the reproduction process). These autonomous parasitic programs cooperate with the host genetic programs, thus realizing process of self-replication - the only aim, which can be associated with that activity. We developed some new operators which are the computer program procedures, performing processes of replication, mutation and invasion of MGEs into specific sites on chromosomes, as well as interactions of MGE with the chromosome (interrelations of parasite - host type).

It is appropriate here to make some notes, concerning the terminology. MGE technique comprises the procedures for initialization of mobile genetic elements and procedures for operating with these elements. Hereinafter in this section mobile elements will be referred to as "viruses", whereas the procedures, operating with them will be termed as "MGE operators". There are only two types of operators. The one-place operator is an analogue of point mutation and the two-place (binary) operator realizing the procedure of transmission of virus from one chromosome (host) to another chromosome (another host).

2.1.1 Viruses

Let us recall that the ant binary string - chromosome is coding a state transition table of finite state automation. Altogether there are 32 finite states of automation, ranging from STATE#0 up to STATE#31. All operators start reading and interpreting the table beginning from the STATE#0. For example, STATE #0 determines one of the four actions or instructions (FWD - "forward", RGT - "to the right", LFT - "to the left" or NOP - "do-nothing") and the number of the next state, depending on binary input value (0 or 1). This finite state automation can be represented as a state transition diagram and interpreted as a decision tree but, as far as references to already passed by states are permissible, that tree can have loops.

Henceforward we will refer to these state number sequences, which ant can pass through moving along the branches of the tree and according sequences of instructions (routines), which it will perform, as "patterns". In other words, pattern is concrete sequence of states, which an ant can come through and sequence of instructions, which an ant can perform, when it passes from state to state. Concrete example of patterns are given on the Fig. 1. Hereinafter, the abbreviations of instructions in the pattern will be referred to as elements of pattern.

We use this concrete definition of our *virus* (mobile genetic element - transposon). Virus is the pattern, having the following properties:

the pattern should include elements which number lie in the range between minimum and maximum values;

the pattern should not contain NOP elements and internal circles;

the pattern should be finished up with a reference to the initial state. The transitions cycle will be executed until only white squares remain ahead of the ant.

2.1.2 MGE - operators

MGE - operators scan the predetermined quota of chromosomes in population. Successively decoding chromosome record, this operator is seeking for procedure sequences, which are identified as virus. But, MGE operator perceives procedures and state transitions only with the proviso that there is no labeled square ahead of the ant, i.e. under condition *input=0* (See fig. 1).

State	Input=0
0	**LFT/#17**
17	**FWD/#13**
13	**FWD/#21**
21	**LFT/#9**
9	**LFT/#0**

Figure 1. Here is an example of a virus. The virus is a closed five-element cycle of states transitions (0, 17, 13, 21, 9, and again, 0). There are 32 states at all. Each state determines two alternate actions, depending on input signal. The input signal is what an ant sees before him. If the cell before him is black then the input is 1, in opposite case the input is 0. Each of alternative actions includes one of four possible movements (FWD, RGT, LFT or NOP) and transition to the next state.

Two-place MGE operator provides the transmission of the virus from an ant to another one, thus realizing the reproduction procedure of this virus in gene pool of the host (ant) population. This procedure performs the following operations.

First, a pair of ants is chosen at random. Then, the chromosome of any of them is scanned in search of the virus. If the virus is found, it is replicated in the partner chromosome, irrespectively of initial record character in that chromosome. The chromosome scanning starts from the zero line (state#0) and goes on as far as the first virus is met. If no virus is met, scanning finishes up only when the chromosome record ends. So, scanning ceases irrespectively of the remaining chromosome un-scanned part content.

One-place MGE operator is a sort of point mutation, realized under particular conditions. This is what we call an *intelligent mutator*. In detail, the operator acts in such a way. If it finds a pattern in the predetermined length range, and the action NOP completes this pattern, then this instruction is substituted for the one of the three other actions (FWD, RGT or LFT). Specifically, this NOP is substituted for the action from the fifth element of the pattern, counted in order. But, if the found pattern is completed by the reference to the one of the elements inside pattern (internal cycle), then we have the following. The action of this element is substituted for the action of the fifth element, counted backward from the end, the reference being substituted for found at random reference to the element outside of the pattern.

3 RESULTS

The test trail, used in this work is illustrated in Fig. 2. It can be seen that up to the 64th element our trail coincide with the Los Altos one, but the next part of the trail includes chaotically scattered elements of high complexity. Being trained on much simpler preceding trail part, the ant is not prepared to surmount the subsequent, complicated sector (biologists would say that the ant is not pre-adapted to new conditions it faced with in this sector). More specifically, problems arise at attempts to get over gaps between the 64th and the 65th, or the 67th and the 68th cells.

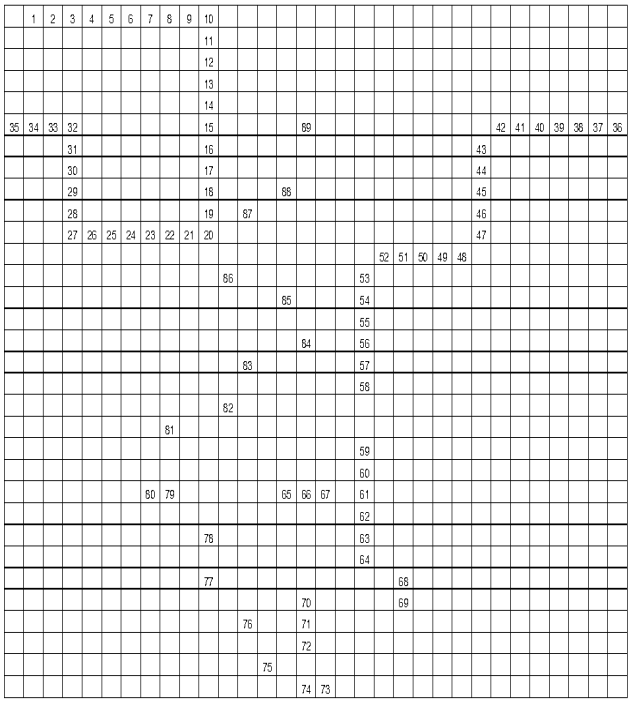

Figure 2. Ant trail used in our computer experiments. The trail itself is a series of squares on a 32x32 white toroidal grid. Each cell is numbered sequentially, from the 1st to the 89th. The first two gaps of the higher complexity are between 64th and 65th and 67th and 68th.

3.1 MGES REALLY ACCELERATES THE EVOLUTIONARY SEARCH

The preliminary computer experiments showed that the accelerating effect of MGE is especially noticeable for small populations, when the probability of the effective navigation algorithm finding by applying standard crossover and mutation operators is low.

On this basis, the following experiments were carried out on populations of 100 ants. The choice of such a small

population is also explained by our aim to carry out a comprehensive analysis of MGE dynamics. Such an analysis is not feasible for large populations of ants because of great number of viruses.

With the aim of demonstrating of the MGE technique efficiency we performed 100 independent runs of the program, 5000 generations each. The results of test and control runs (population with MGE and without MGE correspondingly) were compared in several series with the different values of standard mutation parameters. Everywhere in this section we will accept that the effective navigation algorithm should overcome the level of maximum score in 64 for 330 time steps.

The results of program runs with the MGE operator and without it are illustrated in Fig. 3. It can be seen, that MGE technique obviously increases the probability of finding of effective navigation algorithm for small populations and for a little number of generations.

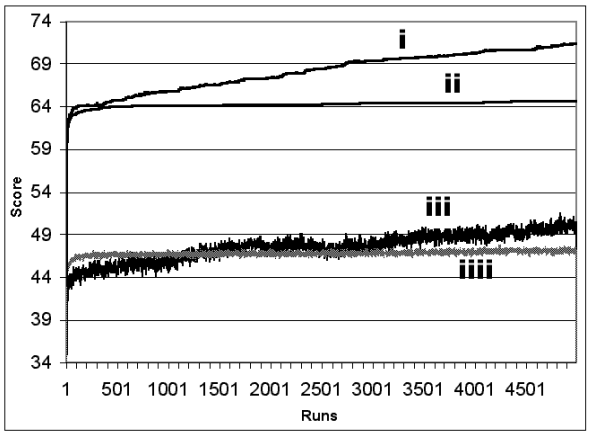

Figure 3. Numerical experiments, demonstrating statistically certain increasing of the GA efficiency due to the effect of MGE operators. A comparison of the mean and the best-of-generation score dynamics (MGE operator being activated) with the control (MGE operator is disabled). The score values are averaged over 100 runs in both cases. The size of population = 100; the number of generations = 5000; the pattern size varies from 5 to 11; crossover rate (P/bit)/generation = 0.0001; mutation rate (P/bit)/generation = 0.04; **i** are the best-of-generation scores and **iii** are the mean scores for the runs with MGE operators; **ii** are the best-of-generation scores and **iiii** are the mean scores for the control runs (without MGE operators).

As it is evident from the graphs on Fig. 3, the mean and the best-of-generation score scores in experiment and in control are growing, to a first approximation, linear in time. But the increment of growth in experiment with MGE is substantially higher, than in control.

It may be suggested that MGE operators raise ant variability mainly in nonspecific manner thus supplementing mutation effect of standard operators. But, this suggestion is not substantiated by the detailed analysis of mutation process. We carried out control runs with different values of standard mutations: the high level of standard mutation does not raise the effectiveness of the navigation algorithm search, moreover, it decreases this effectiveness.

3.2 HORIZONTAL TRANSMISSION OF MGES IS NECESSARY FOR THEIR EFFECTIVE ACTION

As far MGEs are transmitted vertically (from ancestors to descendants), MGE of the host, that have superiority in reproduction success is rapidly spreading in the population and gives new forms. But this process *per se* is insufficient for the effective acceleration of ant learning. Two-place MGE operator, performing horizontal distribution of MGE from one ant to another is a necessary for rising of ant training ability. In Fig. 4 we illustrate the results of comparing of the test, presented in Fig.3, with the similar test, in which frequency of applying of two-place MGE operator was reduced by the factor of 10 and accounted 5%. This parameter determines the proportion of population, which is subjected to the action the two-place MGE operator in a generation. In previous experiments, this quota accounted 50%.

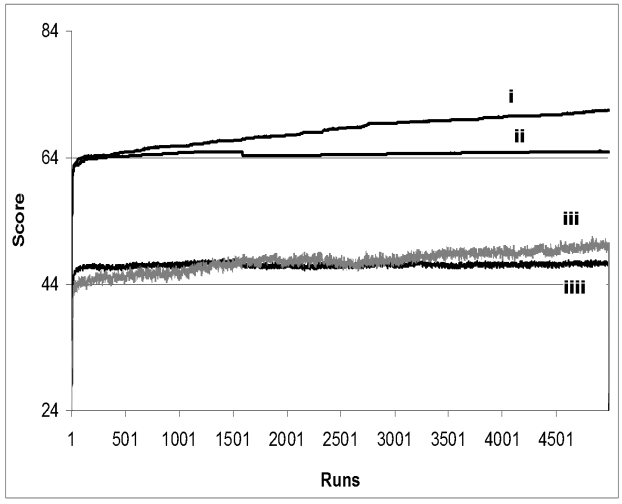

Figure 4. The influence of decreasing of frequency of applying of two-place MGE operator on the ant learning abilities. **i** are the best-of-generation scores and **iii** are the mean scores for the runs with high frequency of the two-place MGE operator action (50%); **ii** are the best-of-generation scores and **iiii** are the mean scores for the runs with low frequency of the two-place MGE operator action (5%). The other parameters are the same as in the previous experiments (see caption to Fig. 3).

The obvious lowering of ant learning abilities with the decreasing of frequency of the two-placed operator application is seen from the diagram. Disabling of the

operator lowers the efficacy further and makes it almost equal to the control (case without MGE).

4 DISCUSSION

The problem of programming an artificial ant to follow the Santa Fe trail has been repeatedly used as a benchmark problem in GP (For references See Langdon and Poli, 1998). Recently Langdon and Poli have shown that performance of several techniques is not much better than the best performance obtainable using uniform random search (Langdon and Poli, 1998). According to these authors, the search space is large and forms a Karst landscape containing many false peaks and many plateaus riven with deep valleys. The problem fitness landscape is difficult for hill climbers and the problem is also difficult for Genetic Algorithms as it contains multiple levels of deception.

There are many techniques capable of finding solutions to the ant problem (GA, GP, simulated annealing, hill clmbing) and although these have different performance the best typically only do marginally better than the best performance that could be obtained with random search (Langdon and Poli, 1998). That is why the ant problem may be indicative of real optimization problem spaces.

4.1 DOMINANT MGE ARE THE COMPONENTS OF THE EFFECTIVE NAVIGATION ALGORITHMS

The results of careful analysis of organization of several tens of dominant viruses, taken from those ant populations, which coped with the navigation task, can be summarized as follows.

1) By the definition, the virus program begins and ends with the zero state, i.e., it is a loop, executed over and over until the ant will meet the labeled cell.

2) Four-fold execution of the virus-program produces in most cases the closed ant trajectories, i.e., the ant will return to the starting position. As a rule, the closed contour is located in domains the size of 4×4 or 5×5 cells.

3) As a rule, the virus-program is beginning to work not from the zero state but from the Nth state, which is specific to every virus, not beginning with the initial, zero state. This transition into the Nth state takes place as soon as the ant (host of the virus) runs against the unlabeled cell.

4) Start the virus-program from the Nth state provides the execution of the simplest navigation algorithm, necessary for overcoming the simplest gaps, arranged in the first half of the trail ("looking around", then one step ahead, "looking around" again and so forth). This algorithm provides the successful passage of trail up to the 64^{th} cell inclusive.

5) The majority of program-viruses guarantee overcoming of the element of high complexity between the 64^{th} and the 65^{th} cells.

6) Some viruses are not suitable for the navigation programs. In that case the chromosome elements, arranged in virus-free domain take control over navigation.

The detailed analysis of the organization of dominant MGE forms in populations, which are succeeded in finding of the effective navigation programs, showed, that the MGE themselves become the components of these programs. Namely, the case in point is about the part of navigation program that is used for effective "snuffing around" in situation, when ant faces with a wide gap.

4.2 WISE MUTATORS HAVE A SEARCH SPACE CONFINING EFFECT

The Muir's Trail search space has rugged geometry due to specific and discrete character of the problem. That is why, the gradient methods are not effective here. Moreover, this ant navigation problem is classified as a GA hard problem, especially if trail is not designed specially for ant population training. The efficiency of MGE in the role of intelligent mutators can be measured by their search space domain confining ability. Therefore, the selection criteria inserted into MGE operators had to increase the probability of the effective navigation algorithm finding on the element of high complexity.

A comparison of mutation frequencies in experiment and control with the according learning rates confirms multiple reduction of evaluation numbers, needed for reaching of the same required learning in experiments with MGE. Mutation frequencies for basic experiments (Fig.3) in control accounts: crossover rate + mutation rate = 0.0001+0.04 P/bit/generation; MGE1 and MGE2 operators add in average 0.0027 and 0.0075 P/bit/generation accordingly. In other words, MGE in average adds to value 0.041 about 0.012 P/bit/generation. This addition brings to multiple acceleration of ant population learning! Hence, according to fig. 3, up to the end of the experiment (4622 time-step) the control set gives max score 6.47, whereas in the test set this value is attained already on the 451 time-step, i.e. 10 times sooner.

5 CONCLUSIONS

- The enhancement of GA by jumping genes-mutators substantially increases the efficacy of GA performance in known benchmark test – ant problem.
- The jumping genes-mutators (artificial transposons) act as intelligent mutators, that "elaborate" code blocks with high evolvability value.

Acknowledgments

This work is supported by INTAS grant No 97-3095.

References

Altenberg L. (1994) The evolution of evolvability in genetic programming. In: K. E. Kinnear, ed. *Advances in Genetic Programming*. MIT Press, Cambridge, pp. 47-74.

Brennan P. (1994) ANT: Simulated Evolution on a PC, manuscript.

Brosius J. (1991) Retroposons - Seeds of evolution. *Science* 251, 753.

Burke D.S., De Jong K.A., Grefenstette J.J., Ramsey C.L. and Wu A. S. (1998) Putting more genetics into genetic algorithms, *Evolutionary Computation*, 6:4, 387-410.

Cliff D. and Grand S. (1999) The *Creatures* Global Digital Ecosystem. *Artificial Life* 5(1): 77-93.

Corno F., Reorda M. S. and Squillero G. (1998) The selfish gene algorithm: a new evolutionary optimization strategy. In: *Proceedings of the 1998 ACM symposium on Applied Computing*, February 27 - March 1, 1998, Atlanta, GA, USA, pp. 349-355.

Daniels M. (1999) Integrating Simulation Technologies with Swarm, *Agent Simulation: Applications, Models and Tools*, October 1999, Argonne National Laboratory, University of Chicago.

Doolittle W. F. and Sapienza C. (1980) Selfish genes, the phenotype paradigm and genome evolution. *Nature* 284: 601-603.

Harries K. and Smith P. (1997) Exploring alternative operators and search strategies in genetic programming. In John R. Koza, Kalyanmoy Deb, Marco Dorigo, David B. Fogel, Max Garzon, Hitoshi Iba, and Rick L. Riolo, eds, *Genetic Programming 1997: Proceedings of the Second Annual Conference*, Stanford University, CA, USA, 13-16 July 1997. Morgan Kaufmann, pp. 147-155.

Harvey I. (1996) The microbial genetic algorithm, unpublished work, available at ftp://ftp.cogs.susx.ac.uk/pub/users/inmanh/Microbe.ps.gz

Hillis W.D. (1990) Co-evolving parasites improve simulated evolution as an optimization procedure, *Physica D*, 42:228-234.

Jefferson D., Collins R., Cooper C., Dyer M., Flowers M., Korf R., Taylor C. and Wang A. (1991) Evolution as a Theme in Artificial Life: The Genesys/Tracker System. In: *Artificial Life II, SFI Studies in the Sciences of Complexity, vol. X*, edited by C.G. Langton, C. Taylor, J.D. Farmer, and S. Rasmussen. Addison-Wesley, pp.417-434.

De Jong K.A. and Potter M.A. (1995) Evolving complex structures via cooperative coevolution, In: *Forth Annual Conference on Evolutionary Computation*, San Diego, CA, 1-3 March 1995.

Koza J.R. (1992) *Genetic Programming: on the Programming of Computers by Means of Natural Selection*. MIT Press, Cambridge, Mass.

Langdon W. B. and Poli R. (1998) Why ants are hard. In John R. Koza, Wolfgang Banzhaf, Kumar Chellapilla, Kalyanmoy Deb, Marco Dorigo, David B. Fogel, Max H. Garzon, David E. Goldberg, Hitoshi Iba, and Rick Riolo, editors, *Genetic Programming 1998: Proceedings of the Third Annual Conference*, University of Wisconsin, Madison, Wisconsin, USA, 22-25 July 1998. Morgan Kaufmann, pp.193-201.

Lee C-Y, and Antonsson E.K., Adaptive Evolvability via Non-Coding Segment Induced Linkage, Proceedings of the Genetic and Evolutionary Computation Conference, San Francisco, CA, 2001.

Lones M.A. and Tyrrell A.M., Biomimetic Representation in Genetic Programming, In: *Proceedings of the Workshop on Computation in Gene Expression at the Genetic and Evolutionary Computation Conference 2001 (GECCO2001)*, San Francisco, California, USA, July 2001, pp. 199-204.

Luke S., Hamahashi S. and Kitano H. (1999) "Genetic" programming, GECCO-99: *Proceedings of the Genetic and Evolutionary Computation Conference*, Banzhaf, W. et al, eds. San Fransisco: Morgan Kaufmann.

Makalowski W. (1995) SINEs as a Genomic Scrap Yard. Chap. 5 in *The Impact of Short Interspersed Elements (SINEs) on the Host Genome*, edited by Richard J. Maraia. Austin: R.G. Landes Company.

Nawa N. E., Furuhashi T., Hashiyama T. and Uchikawa Y. (1999) A study on the discovery of relevant fuzzy rules using pseudo-bacterial genetic algorithms, *IEEE Transactions on Industrial Electronics*, 7, (5), 608-616, October 1999.

Orgel L. E. and Crick F. H. C. (1980) Selfish DNA: The ultimate parasite. *Nature* 284: 604-607.

Olsson B. (1996) Optimization using a host-parasite model with variable-size distributed populations. In: *Proceedings of the 1996 IEEE 3rd International Conference on Evolutionary Computation*, IEEE Press, pp. 295-299.

Olsson B. (2001) Co-evolutionary search in asymmetric spaces, In Wang, P.P., ed., *Proceedings of The Fifth Joint Conference on Information Sciences*, Association for Intelligent Machinery, pp. 1040-1043.

Potter M.A. and De Jong K.A. (1994) A cooperative co-evolutionary approach to function optimization, In: *Third Parallel Problem Solving from Nature*, Jerusalem, Israel, pp 249-257.

Potter M.A. and De Jong K.A. (1995) Evolving neural networks with collaborative species, In: *Proc. of the 1995 Summer Computer Simulation Conference*, Ottawa, Ontario, Canada, 24-26 July 1995, pp. 340-345.

Ray T. S. (1991) An approach to the synthesis of life. In: Langton, C., C. Taylor, J. D. Farmer, & S. Rasmussen [eds], *Artificial Life II*, Santa Fe Institute Studies in the Sciences of Complexity, vol. XI, Redwood City, CA: Addison-Wesley, pp. 371-408.

Ray T. S. (2001) Overview of Tierra at ATR. In: *Technical Information, No.15, Technologies for Software Evolutionary Systems.* ATR-HIP. Kyoto, Japan.

Simoes A. and Costa E. (2001) An evolutionary approach to the Zero/One knapsack problem: testing ideas from biology; In: *Procs. 5th Int. Conference on Artificial Neural Networks and Genetic Algorithms (ICANNGA 2001)*, Prague, Czech Republic, 22-25 April 2001.

Spirov A.V. (1996a) Self-Assemblage of gene Networks in Evolution via Recruiting of New Netters. *Lecture Notes in Computer Sciences.* 1141: 91-100.

Spirov A.V. (1996b) Self-organisation of gene networks in evolution via recruiting of new netters. In: *Proceedings of the First International Conference on Evolutionary Computations and Its Applications*, Moscow, Russia, pp. 399-405.

Spirov A.V. and Samsonova M.G. (1997) Strategy of Co-evolution of Transposons and Host Genome: Application to Evolutionary Computations. In: *Proceedings of the Third Nordic Workshop on Genetic Algorithms and their Applications (3NWGA)*, 20 - 22 August 1997, Helsinki, Finland, Ed. Jarmo T. Alander, Finnish Artificial Intelligence Society, pp. 71-82.

Spirov A.V., Kadyrov A.S. (1998) Transposon Element Technique Applied to GA-based John Muir's Trail Test, In: *High-Performance Computing and Networ*king, pp. 925-928.

Spirov A.V., Kazansky A.B. and Kadyrov A.S. (1998) Utilizing of "Parasitic" Mobile Genetic Elements in Genetic Algorithms. In: *International Conference on Soft Computing and Measurments*, St.Petersburg, pp. 266-269.

Spirov A.V. and Kazansky A.B. (1999) Evolutionary Biology and Evolutionary Computations: Parasitic Mobile Genetic Elements in Artifical Evolution, In: *2nd Int. Conf. on Soft Computing and Measurments*, St.Petersburg.

Zongker, D. and Punch, B. (1995) lil-gp 1.0, http://isl.cps.msu.edu/GA/software/lil-gp