Coevolution, Memory and Balance

Jan Paredis
MATRIKS
Universiteit Maastricht
P.O. Box 616, NL-6200 AL Maastricht
The Netherlands
e-mail: jan@riks.nl

Abstract

This paper studies the role of two mechanisms - memory and balance - to exploit the arms race resulting from predator-prey interactions when solving a given problem. Memory ensures that individuals are not only well adapted to the current members of the opposite population but also to earlier generations of opponents. A balanced (co)evolution, on the other hand, adapts the speed of evolution (i.e. the reproduction rate) to the performance of a population. It leads to a steady progress in both populations. Indirectly, a balanced (co)evolution avoids a premature loss of genetic diversity. This in turn, diminishes the need for a long memory span. The current paper shows how both mechanisms can be incorporated in Coevolutionary Genetic Algorithms (CGAs). Empirical results support the importance of, and interaction between, both mechanisms.

1 Introduction

In nature, organisms undergo selective pressures from other organisms - belonging to the same or to other species. Predator-prey interactions are a common example of selective pressure between organisms. These interactions are characterized by an *inverse fitness interaction*: success on one side is felt by the other side as failure to which must be responded in order to maintain one's chances of survival. This typically results in an *arms race* in which the complexity of both predator and prey increases stepwise. It is a challenge to integrate this mechanism into an evolutionary algorithm in order to further improve its performance.

Paredis [1999] provides an overview of the use of coevolution for computational purposes starting from the early days of computing. Here, only work directly relevant to the issues of the current paper is discussed. Hillis [1992] coevolved networks for sorting lists of sixteen numbers. Such a sorting network consists of comparisons between two numbers (and the possibility of swapping both numbers). The goal was to find a network, which correctly sorts all possible lists of sixteen numbers, with as few comparisons as possible. Hillis used two populations. The first population consisted of sorting networks. The individuals in the second population were sets of test-lists. These lists contain numbers to be sorted. Both populations were geographically distributed over a grid with each location containing one set of test-lists and one sorting network. At each generation a sorting network was tested on the set of test-lists at the same location. The fitness of a sorting network was defined as the percentage of correctly sorted test-lists. The fitness of the set of test-lists, on the other hand, was equal to the percentage of test-lists incorrectly sorted by the network. This is similar to the inverse fitness interaction between predator and prey.

Hillis has shown that - in comparison with traditional non-coevolutionary single population GAs - better sorting networks can be obtained. Inspired by this work, Paredis [1994a] introduced a Coevolutionary Genetic Algorithm (CGA) which uses a partial but continuous fitness evaluation resulting from encounters between individuals belonging to different species. The first CGA was used to train a Neural Network (NN) on a classification task. Next, an abstract class of problems which can be solved by CGAs was defined: so-called *test-solution problems* [Paredis 1996]. This large class of problems in Artificial Intelligence and Computer Science in general, involves the search for a solution which satisfies a given set of tests.

A CGA uses two populations to solve test-solution problems. One contains the tests which a solution should satisfy. The other consists of potential solutions to the problem. CGAs have been applied to various other test-solution problems, such as: constraint satisfaction [Paredis 1994b], process control [Paredis, 1998], path planning [Paredis and Westra, 1997], and the evolution of cellular automata (CA) for density classification [Paredis, 1997]. The main purpose of the applications described above is to help us understand, explore, and illustrate the operation of CGAs. Recently, researchers have been using CGAs in real-world applications such as object motion estimation from video images [Dixon *et al.*, 1997] and time tabling for an emergency service [Kragelund, 1997].

Some CGA applications used a test population whose contents remained fixed over time [Kragelund, 1997; Paredis, 1994a; 1994b]. In others, both populations fully evolved [Dixon *et al.*, 1997; Paredis, 1997; 1998; Paredis

and Westra, 1997]. In the latter case, there is pressure on both populations to improve, i.e. the solutions "try" to correctly satisfy as many individuals as possible in the test population. At the same time the tests "try" to make life hard for the solutions. It is not in their "interest" that the problem be solved. Especially if there is no solution satisfying all possible tests then the tests might keep the solutions under constant attack. Even if there exists a solution satisfying all tests, it might be virtually unreachable given the coevolutionary dynamics. This might be caused by the high degree of epistasis in the linkages between both populations. Due to such linkages a small change to the individuals in one population might require extensive changes in the other population. This was exactly what happened in the CA application [Paredis, 1997]: the tests were able to keep ahead of the solutions by "jumping" back and forth between two regions - in close proximity of each other - in the space of all tests. After each such jump of the test population, extensive changes to the "genes" of the solutions, i.e. CAs, were required in order to satisfy the tests. Paredis [1997] provides a simple diversity preserving scheme which prevents the tests from keeping ahead of the solutions. In fact, it achieves this by randomly inserting and deleting individuals from the test population. It also suggests other mechanisms, such as fitness sharing or the use of different reproduction rates in both populations. The latter approach is investigated here. The goal is to obtain a good *balance* such that there is a steady progress in both populations. This is to avoid the tests from "out-evolving" the solutions and to ensure that there is sufficient variation in the fitness of the members of a population. The latter is necessary to guide (co)evolution.

The use of *fitness sharing* to preserve diversity in a competitive coevolutionary context has been investigated by a number of researchers. The Othello application of Smith and Gray [1994] is probably one of the earliest. Rosin and Belew [1995] combine competitive fitness sharing and shared sampling. The latter ensures that representative opponents are selected for the "competition duels". Juillé and Pollack [1996] proposed another way to ensure population diversity: in a classification application, a solution receives a larger reward when it correctly classifies a test which is classified correctly by none (or only a few) of the other solutions. More recently, Rosin [1997] has identified the causes why coevolutionary progress can come to a halt: the loss of important niches, coevolutionary cycles, or a lack of balance. He observed that, even with fitness sharing, solutions might "forget" how to compete against old extinct types of individuals which might be difficult to rediscover. For this reason he introduces a "Hall of Fame", containing the best individuals of previous generations which are used for testing new individuals. This introduces some kind of *memory* in the system. In order to keep the progress in both population balanced, Rosin [1997] introduces a "phantom parasite" which provides a niche for "interesting" individuals.

The final goal of a CGA solving a test-solution problem is to find a solution which satisfies all possible tests. Any other "victory" of the solution population is only a Pyrrhic victory. A solution population whose members satisfy all members in the test-population, but who do not satisfy tests which are unlikely to make their way in the test-population, is useless. This is another reason why it is important to preserve the genetic diversity of the test population for long enough a period.

The structure of this paper is as follows. The next section describes the use of CGAs for test-solution problems in more detail. Section three describes how to obtain a balanced coevolution within a CGA and how the memory can be increased. The fifth section evaluates the impact of these mechanisms when solving multimodal problems described in section four. Finally, the paper closes with a discussion and conclusions.

2 CGAs For Test-Solution Problems

As a first step in applying CGAs to test-solution problems, both initial populations are filled with randomly generated individuals. Their fitness is calculated by pairing them up with 20 randomly selected members of the other population. In order to calculate the fitness of an initial solution, one counts the number of tests it satisfies. Similarly, the fitness of a test is the number of solutions (out of a randomly chosen set of twenty solutions) which violate the test. As a matter of fact, each individual - test or solution - has a history which stores the fitness feedback - or pay-off - resulting from such an encounter. A solution receives a pay-off of one if it satisfies the test. Otherwise it receives a zero. The opposite is true for tests: they get a pay-off of one if the solution encountered does not satisfy a test. The fitness of an individual is equal to the sum of the pay-offs in its history. It is also good to note that the populations are sorted on fitness: the fitter the individual the higher its rank in the population.

The pseudo-code below describes the basic cycle the CGA executes after the initial populations are generated. First, 20 encounters are executed in which a test and solution are paired up. The SELECTion of these individuals is biased towards highly ranked individuals, i.e. the fitter individuals are more likely to be SELECTed. In a CGA, the most fit individual is 1.5 times more likely to be selected than the individual with median fitness. Next, the actual ENCOUNTER happens during which it is tested whether the solution satisfies the test it encounters. The result of the encounter is one if it does. Otherwise it is zero. This result is also the pay-off received by the solution. The test receives the "inverse" pay-off. The function TOGGLE in the basic cycle implements this inverse fitness interaction between tests and solutions. It changes a one into a zero and vice versa. Again, each individual stores the pay-off it receives in its history. At the same time, the pay-off that the individual received least recently is removed from the history. This guarantees that the history will always contain the 20 most recently received pay-offs. This UPDATE of the HISTORY of an individual, is followed by an UPDATE of the FITNESS of the individual: it is set equal

to the sum of the pay-offs in its history. Because both populations are kept sorted on fitness, an individual might move up and down in its population as a result of the update of its fitness.

After the execution of the encounters, one offspring is generated in each population. This happens as follows. First, two parents are SELECTED from the population, again with the same bias towards fitter individuals as explained above. Next, (2-point reduced surrogate) CROSSOVER and (adaptive) MUTATion are used in order to generate the child from the bit-string of each of its parents. Then the FITNESS of the CHILD is calculated as the sum of the pay-off received from encounters with 20 SELECTed individuals from the other population. If this fitness is higher than the minimum fitness in the population to which its parents belong then the child is INSERTED at the appropriate rank location in this population. At the same time, the individual with the minimum fitness is removed. In this way, both populations remain sorted on fitness.

```
DO 20 TIMES
   sol:= SELECT(sol-pop)
   test:= SELECT(test-pop)
   res:= ENCOUNTER(sol,test)
   UPDATE-HISTORY-AND-FITNESS(sol,res)
   UPDATE-HISTORY-AND-FITNESS(test,TOGGLE(res))
ENDDO
s-parent1:= SELECT(sol-pop)
s-parent2:= SELECT(sol-pop)
s-child:= MUTATE-CROSSOVER(s-parent1,s-parent2)
f:= FITNESS(sol-child)
INSERT(sol-child,f,sol-pop)
t-parent1:= SELECT(test-pop)
t-parent2:= SELECT(test-pop)
t-child:= MUTATE-CROSSOVER(t-parent1,t-parent2)
f:= FITNESS(test-child)
INSERT(test-child,f,test-pop)
```

A couple of remarks are in place here. First, note that in order to avoid unreliable pay-off from possible mediocre offspring, the encounters with new offspring do only result in pay-off for the child, not for the individuals it is encountering.

Second, the partial but continuous fitness feedback resulting from the encounters is called life-time fitness evaluation (LTFE). Two parameters are associated with it. The first is the size of the histories. The second is the number of encounters per cycle. In all previous CGA applications, both parameters are set - quite arbitrarily - to 20. Except when mentioned explicitly, this will be the case here too.

3 Increasing the Memory and Keeping the Balance

This paper investigates whether an increase of the memory of the individuals prevents the two species (solutions and tests) from performing a cyclical dance of changes without really improving their quality. By increasing the memory, individuals which learn new things at the expense of things learned earlier are penalized. The balancing mechanism proposed here is devised to prevent the tests from outperforming too much the solutions. It also ensures that as the solutions get better, more effort is put into finding tests which are not satisfied by the solutions.

3.1 Memory

The steady-state character of the CGA together with the continuous nature of LTFE provides memory to the individuals. An individual's fitness reflects its performance against members of the opposite population over a period of time. The variant proposed here accumulates the pay-offs received during the entire lifetime of an individual. In this case, the fitness is defined as the number of times a pay-off of one is received divided by the total number of encounters the individual was involved in. This alternative for LTFE is called lifelong fitness evaluation (LLFE). With LLFE, the fitness of an individual reflects its performance against the members of the opposite population it encountered during its entire lifetime. In this way, there is a selective advantage for individuals to also "beat" members of less recent generations of the opposite population. Hence, in comparison with LTFE, LLFE increases the memory of the individuals.

3.2 Balance

The need for a balanced evolution immediately raises two questions: 1) How can the CGA recognise that the evolution is getting unbalanced? 2) Once an imbalance is recognised, how can the balance be restored?

The answer to the first question is relatively simple: if both populations evolve in a balanced manner then they will receive roughly the same amount of pay-off during the encounters. Conversely, the evolution is getting more imbalanced as the pay-off received by both populations is more unequal.

Now, the second question can be addressed. What should be done when one population gets considerably more pay-off than the other? Somehow the weaker population has to be given a helping hand. Its efforts to counter the increasing opposition should be increased. This is done by allowing different reproduction rates in both populations. In the standard CGA, two offspring are generated per cycle, one in each population. Now, only one offspring will be generated per cycle. The percentage of successes in the last twenty encounters is used to stochastically determine which population evolves. If none of the 20 encounters is successful then the tests may well be too good, i.e. difficult, for the solutions currently in the solution population. Hence, the probability that the solution-population evolves (i.e. two parents are SELECTED from it, and if their offspring is fit enough, it is INSERTed in the solution population) is equal to one. As the percentage of successes increases this probability linearly decreases (see figure 1). It becomes zero when all encounters are successful. In this case, the probability that the test population reproduces is equal to one.

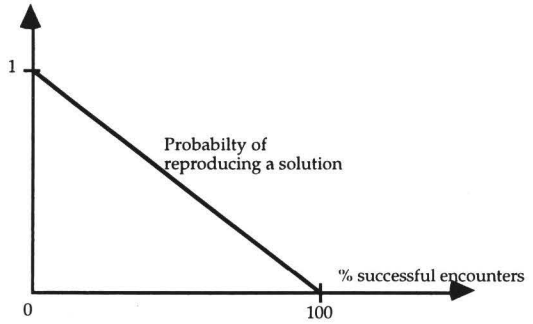

Figure 1. In order to determine which population reproduces, a real number is uniformly drawn from the interval [0,1]. If this number is smaller than the probability of reproducing a solution then the solution population evolves else the test population evolves.

4 Testing the CGA on the P-Model

De Jong *et al.*, [1997] introduce a simple multimodal problem generator which allows to perform controlled experiments testing the mechanisms described above. As a first step, a set of P random bit-strings of length N is generated. These represent the location of P peaks in a N-dimensional boolean hypercube. The strength[1] of an individual is equal to the proportion of bits the string has in common with the nearest peak. When P is equal to one, a single peaked strength-landscape is obtained. For larger P, the modality, as well as the degree of epistasis, increases.

The P-model allows us to test the mechanisms introduced before on problems with different degrees of difficulty. An additional advantage of the P-model - in comparison with, for example, the NK-model [Kaufman, 1989] - is that the optimal value of strength is known: it is one. Furthermore, for different P, the variance in strength is not significantly reduced.

In our experiments, the P-model is used as follows. Each population, test and solution, has its own set of optima which might be different in location as well as number, i.e. they can have a different value of P. During an encounter, the individual with the highest strength gets a pay-off of one, the other receives zero pay-off. These pay-offs are used to calculate the fitness as described in section 2. As was the case for test-solution problems, the pay-offs are restricted to two values, zero and one, instead of just being equal to the respective strengths. This to facilitate the generalization of the conclusions drawn from the experiments described here to test-solution problems in general. Note that when the individuals involved in an encounter have the same strength, the solution gets the pay-off of one and the test gets zero pay-off. This to ensure that there exists a solution which beats all tests. Ideally, the CGA should find this individual which has a strength of one.

[1] De Jong et al use the term fitness. We use the term strength in order to avoid confusion with the definition of fitness in CGAs.

5 Empirical Results

Just as was the case in earlier CGA applications, both populations contain 100 individuals. Also the same genetic operators are used: reduced surrogate two-point crossover and adaptive mutation. Furthermore, N is set equal to 50. In the experiment described here, the degree of epistasis of the test strength landscape is higher than that of the solution landscape. Due to space restrictions two other experiments, in which the solution strength landscape has the highest P, or where both populations have the same P value are not reported here. These two additional experiments confirm the conclusions drawn from the experiment discussed here.

In the current paper, the performance of 8 CGA variants is compared. The name of each variant consists of a triplet. The first part indicates whether LTFE or LLFE is used to calculate the fitness of the solutions. The second part represents the type of fitness calculation used for the test population. The third part is either labelled "TRAD" or "BAL". The former indicates that the traditional "unbalanced" algorithm - as described in section 2 - is used, the latter uses the balanced algorithm. L-T-BAL, for example, refers to the **BAL**anced variant which uses lifelong fitness evaluation (L**L**FE) for the solution population, and lifetime fitness evaluation (L**T**FE) for the test population.

In order to allow for a fair comparison, all variants generate 5000 offspring. This means that the variants with a balanced evolution execute 5000 cycles, whereas the "unbalanced" variants - which produce two offspring per cycle - execute 2500 cycles. In order to ensure that all variants use the same amount of computing time and fitness feedback, the balanced variants perform 10 encounters per cycle instead of 20.

The experiment repeats the following steps 50 times. First, a solution strength landscape (P= 250) and a test strength landscape (P= 500) are generated. Next, each variant is run on these landscapes. Finally, the strength of the highest ranking solution individual of each variant is compared. Table 1 summarizes the results obtained. The first column of this table shows how often a given variant produces a highest ranking solution with a strength greater (or equal) than the strength of the highest ranking solution individuals of the other variants. The second column indicates how often it was the second best, etc. Table 1 shows, for example, that for 38 out of 50 runs the highest ranking solution individual of T-T-BAL has the highest strength. Note that the total of all numbers in the first column of the table is above 50. This is because the highest ranking solution of several variants might have the same strength.

A couple of observations can be made from the table. First of all, the upper left quadrant of the table (four leftmost columns of the four top rows) and the bottom right quadrant contain small numbers. The large numbers are located mainly in the two other quadrants. Or, in other words, the balanced CGAs consistently outperform the traditional CGAs in which each population evolves at the same rate. The balanced variant without LLFE, T-T-BAL, performs

best. It is not only 38 times the best of the eight variants. In 46 (38 + 8) out of 50 runs it finds the best or second best strength. Hence, this indicates that LLFE does not bring any improvement, on the contrary.

	1st	2nd	3rd	4rth	5th	6th	7th	8th
T-T-TRAD	5	2	0	3	9	8	12	11
L-T-TRAD	1	0	2	2	9	10	13	13
T-L-TRAD	2	1	2	3	17	14	8	3
L-L-TRAD	0	1	0	1	11	10	16	11
T-T-BAL	38	8	2	2	0	0	0	0
L-T-BAL	29	9	5	5	2	0	0	0
T-L-BAL	22	5	9	12	1	1	0	0
L-L-BAL	22	8	10	9	0	1	0	0

Table 1: Ranking of the strength of the highest ranking solution after 50 runs (see text)

The same picture emerges from table 2 which compares the variants on the basis of the highest solution strength encountered during the entire run. Again, the balanced variants outperform the unbalanced ones considerably. Once more, T-T-BAL is the winner.

	1st	2nd	3rd	4rth	5th	6th	7th	8th
T-T-TRAD	7	4	4	3	10	3	12	7
L-T-TRAD	2	1	3	3	9	8	6	18
T-L-TRAD	6	2	4	7	7	12	5	7
L-L-TRAD	2	1	0	4	13	10	10	10
T-T-BAL	28	7	10	2	3	0	0	0
L-T-BAL	24	7	9	4	4	0	1	1
T-L-BAL	22	10	6	5	6	0	1	0
L-L-BAL	12	10	13	9	3	1	2	0

Table 2: Ranking of the highest solution strength generated during 50 runs (see text)

A third observation is that, at the end of the run of each balanced variant, the solution population always contains at least one individual with the largest strength encountered during the entire run. For, T-T-TRAD, L-T-TRAD, T-L-TRAD, and L-L-TRAD, on the other hand, this is only true for 37, 35, 33, and 40 out of 50 runs, respectively. Moreover, at the end of a run with a balanced CGA, this individual with the highest strength can often be found at the top of the solution population. For T-T-BAL, L-T-BAL, T-L-BAL, and L-L-BAL this is the case in 27, 27, 19 and 30 out of 50 runs, respectively. This is considerably lower for T-T-TRAD (6), L-T-TRAD (12), T-L-TRAD (12), and L-L-TRAD (8). This increased ability of the balanced variants to keep the best individual at the top of the population, explains why the difference between the balanced and unbalanced variants is most outspoken in Table 1.

Finally, let us define a population with a highest fitness smaller or equal to 0.1 as an *extinct* population. In none of the 50 runs, a balanced CGA variant produced an extinct (test or solution) population. This in contrast with the unbalanced variants. As a matter of fact, extinctions only occurred in populations which used LTFE. Out of 50 runs, the solution population of T-T-TRAD and T-L-TRAD became extinct 10 and 3 times respectively. Extinction of test populations occurred 20 and 14 times. The former when T-T-TRAD was used. The latter when L-T-TRAD was used. Clearly, such extinctions must be avoided because uniform low fitness values do not provide enough information to guide coevolution.

6 Conclusions and Discussion

As described in the introduction, other researchers have used fitness sharing to preserve genetic diversity in a coevolutionary environment. The balanced CGA deals with the direct cause of the loss of diversity: selection. When selection is too severe, the population rapidly converges to a (sub)optimum. In a CGA, selection occurs through competition between new offspring and individuals already in the population. The balancing mechanism proposed here makes the reproduction rate (and hence the amount of selection) dependent on the success of the population. In this way, the algorithm avoids that the better population "out-evolves" the weaker one. In fact, once an imbalance occurs, the weaker population is allowed to catch up: more search effort is spent on it. This at the expense of the stronger population whose progress - and convergence - is slowed down.

The experiments show that a balanced coevolution results in an increased performance. Two reasons can be identified for this success. A first one was already given in the previous paragraph: balanced coevolution maintains genetic diversity. In this way, individuals are confronted with a diverse opposition. Secondly, the balancing mechanism, together with the inverse fitness interaction between both populations, ensures that the fitness variance between members of a population is maximal. This because when each population receives about the same amount of pay-off then each population receives roughly as many zero pay-offs as ones. This results in fitness variance which, in its turn, guides (co)evolution. In addition to this, the balanced variants maintain the best solutions found, in (the top of) the population.

In the experiments above an increase in memory did not result in increased performance. There is not yet a satisfying explanation for this. Possibly the steady-state nature of a CGA and the partial and continuous fitness feedback of LTFE provide more than enough memory. Note that although LTFE does not take into account pay-offs received more than 20 encounters ago, the very survival of an old individual indicates that it has been able to withstand numerous opponents. Alternatively, the type of problems used here might preclude the need for memory. In fact, as indi-

viduals climb the strength-landscape, the set of opponents which beat them monotonically decreases. Hence, there is no risk of "forgetting" to beat opponents which were beaten by previous, inferior, individuals. The experiments do, however, suggest a possible relationship between memory and balance: in the absence of a balancing mechanism only populations using LLFE could avoid "extinctions". Hence, balance does seem to decrease the need for memory.

It is good to look whether nature also "uses" a balancing mechanism. When a predator species becomes too successful then more prey is killed. In the short term, the population size of the prey decreases while that of the predators increases. This puts a natural brake on the growth of the predator species because food (i.e. prey) becomes more and more scarce[2]. Similarly, too effective prey will, on one hand, lead to a rapid increase of the prey population size, which, in its turn, provides a larger food resource for the predators. On the other hand, the predator population size decreases, i.e. the selective pressure to improve increases. Hence, in nature, fluctuations in population size keep (co)evolution in balance. This in contrast with the fixed population sizes in CGAs which do not allow for real extinctions of test or solution species. Hence, an alternative balancing mechanism is introduced here which adapts the selective pressures through a change of reproduction rates.

Recently, Olsson (1998) introduced a balancing mechanism which lets a population evolve until it contains an individual (a "champion") beating all the individuals in the other population. Next, the other population evolves until it finds an individual beating all members of the first population. This process is then repeated but at each stage all the previous champions as well as the current members of the opposite population should be beaten. Such a mechanism could be introduced in a balanced CGA by marking champions. These marked individuals should always remain in the population.

Although experiments with different degrees of epistasis support the conclusions given here, other CGA applications are needed to confirm their generality. Other balancing mechanisms, for example based on the (difference in) fitness of the individuals instead of on the number of successes during the encounters, might further improve performance.

References

De Jong, K., Potter, M. A., Spears, W., (1997), Using Problem Generators to Explore the Effects of Epistasis, Proceedings of the Seventh International Conference on Genetic Algorithms (ICGA 97), Baeck, T., (ed.), Morgan Kaufmann Publishers.

Dixon, E. L., Pantsios Markhauser, C., Rao, K. R., (1997) *Object Motion Estimation Technique for Video Images based on a Genetic Algorithm*, IEEE Transactions on Consumer Electronics.

Hillis, W. D., (1992), *Co-evolving Parasites Improve Simulated Evolution as an Optimization Procedure*, in Artificial Life II, Langton, C.G.; Taylor, C.; Farmer, J.D., and Rasmussen, S., (eds), Addison-Wesley, California.

Juille, H., Pollack, J., B., (1996), *Co-evolving Intertwined Spirals*, Proceedings of the Fifth Annual Conference on Evolutionary Programming, MIT Press.

Kauffman, S. A., (1989), Adaptation on Rugged Fitness Landscapes, Lectures in the Sciences of Complexity, Stein, E. (ed), Addison-Wesley.

Kragelund, (1997), *Solving a Timetabling Problem using Hybrid Genetic Algorithms*, Software - Practice & Experience, vol. 27 (10).

Olsson, B., (1998), *A Host-Parasite Genetic Algorithm for Asymmetric Tasks*, Proceedings of the European Conference on Machine Learning

Paredis, J., (1994a), *Steps towards Coevolutionary Classification Neural Networks,* Proceedings Artificial Life IV, R. Brooks, P. Maes (eds), MIT Press.

Paredis, J., (1994b), *Coevolutionary Constraint Satisfaction*, Proceedings PPSN-III, Lecture Notes in Computer Science, vol. 866, Davidor, Y., Schwefel, H-P., Manner, R. (eds.), Springer Verlag.

Paredis, J., (1996), *Coevolutionary Computation,* Artificial Life Journal, Vol. 2, nr. 4, Langton, C. (ed.), MIT Press.

Paredis, J., (1997), *Coevolving Cellular Automata: Be aware of the Red Queen!,* Proceedings of the Seventh International Conference on Genetic Algorithms (ICGA 97), Baeck, T., (ed.), Morgan Kaufmann Publishers.

Paredis, J., Westra, R., (1997), *Coevolutionary Computation for Path Planning*, Proceedings 5th European Congress on Intelligent Techniques and Soft Computing (EUFIT 97), H.-J. Zimmermann (ed.), Verlag Mainz.

Paredis, J., (1998), *Coevolutionary Process Control,* Proceedings of the International Conference on Artificial Neural Networks and Genetic Algorithms (ICANNGA97), G. D. Smith (ed.), Springer, Vienna 1998.

Paredis, J., (1999), *Coevolutionary Algorithms*, The Handbook of Evolutionary Computation, 1st supplement, Bäck, T., Fogel, D., Michalewicz, Z. (eds.), Oxford University Press (in press).

Rosin, C., D., Belew, R., K., (1995), *Methods for Competitive Co-evolution: Finding Opponents Worth Beating*, Proceedings of the Sixth International Conference on Genetic Algorithms (ICGA 95), Eshelman, L. (ed.), Morgan Kaufmann Publishers.

Rosin, C., D., (1997), Coevolutionary Search Among Adversaries, PhD Thesis, Univ. of California, San Diego

Smith, R. E., Gray, B., (1994), *Coadaptive Genetic Algorithms, an Example in Othello Strategies*, Proceedings of the Florida Artificial Intelligence Symposium 1994.

[2] Analoguous to the CGA, a two-population model with one prey population and one predator population is assumed. When a population predates on multiple species then a decline of one of its prey populations might result in more intensive predation on the other population(s).