Evolutionary consequences of coevolving targets

Ludo Pagie and Paulien Hogeweg
Utrecht University, Department of Bioinformatics
Padualaan 8, 3584 CH, Utrecht
fax 030-2513655
The Netherlands
pagie@encode.biol.ruu.nl

February 26, 1998

Abstract

Most evolutionary optimization models incorporate a fitness evaluation which is based on a predefined static set of test cases or problems. In the natural evolutionary process selection is of course not based on a static fitness evaluation. Organisms do not have to combat every existing disease during their lifespan; organisms of one species may live in different or changing environments; different species coevolve. This leads to the question how information is integrated over many generations.

This study focuses on the effects that different fitness evaluation schemes have on the types of genotypes and phenotypes that evolve. The evolutionary target is a simple numerical function. The genetic representation is in the form of a program (i.e. a functional representation, as in genetic programming). Many different programs can code for the same numerical function. In other words, there is a many-to-one mapping between 'genotypes' (the programs) and 'phenotypes'. We compare fitness evaluation based on a large static set of problems and fitness evaluation based on small coevolving sets of problems. In the latter model very little information is presented to the evolving programs regarding the evolutionary target per evolutionary time step.

In other words the fitness evaluation is very sparse. Nevertheless the model produces correct solutions to the complete evolutionary target in about half of the simulations. The complete evaluation model on the other hand does not find correct solutions to the target in any of the simulations. More important, we find that sparse evaluated programs are better generalizable compared to the complete evaluated programs when they are evaluated on a much more dense set of problems. In addition, the two evaluation schemes lead to programs that differ with respect to mutational stability; sparse evaluated programs are less stable than complete evaluated programs.

Keywords Coevolution, genotype-phenotype mapping, information integration, generalizability, mutational stability, genetic programming.

1 Introduction

Evolutionary optimization processes are based on the biological evolutionary process (Goldberg, 1989; Holland, 1992). Most artificial evolutionary models however include a static fitness evaluation function, which clearly does not exist in the natural evolutionary process. In nature the fitness of an individual depends in many ways on non-static features. Organisms live in different environments and interact and coevolve with other organisms.

In 1991 Hillis presented a spatially embedded coevolutionary optimization model in which sorting algorithms coevolved with sorting problems (Hillis, 1992). The sorting algorithms were evaluated on the basis of local problems instead of on the basis of a globally defined static set of problems. Since the 'evolutionary goals' of the sorting problems were opposed to those of the sorting algorithms but were nevertheless dependent on each other, Hillis called the sorting algorithms *hosts* and the sorting problems *parasites*. He found that the incorporation of the coevolving problems resulted in better (i.e. faster) sorting algorithms than those that evolved with a static set of sorting problems. Hillis attributed the difference in the success of the two evaluation schemes to two properties of the coevolving scheme. First, the coevolution of the parasites prevented the population of hosts from 'getting stuck' on local optima. Second, the search process was more efficient because the coevolving parasites focused on those problems that had not yet been solved correctly.

Several authors have studied optimization models in which the fitness evaluation of individuals depends on other individuals in the same population or in other populations. They have reported that such models yield higher fitness values and involve lower computational costs than traditional evolutionary optimization models or other optimization techniques. Three main forms of coevolutionary models can be distinguished, although many variants are used.

- Host-parasitoid models in which candidate solutions are evaluated on the basis of small subsets of a data set which defines the evolutionary target. The subsets coevolve with the candidate solutions (Hillis, 1992; Paredis, 1994; Paredis, 1995).

- In competitive evolutionary models candidate solutions compete with each other in game-like tournaments. The fitness of the solutions de-

pends on the ratio of wins and losses in these tournaments[1] (Angeline and Pollack, 1993; Rosin and Belew, 1997; Sims, 1994; Juillé and Pollack, 1996).

- In cooperative evolutionary models individuals of several different (coevolving) species are combined before they are evaluated with respect to an evolutionary target [1] (Husbands, 1994; Potter and De Jong, 1994; Vafaie and De Jong, 1996; Potter et al., 1995).

Other studies, however, have shown that in some cases coevolution does not lead to better results (Thompson, 1996).

Here we present results of a study in which we compare static fitness evaluation and sparse, coevolving fitness evaluation of candidate solutions, the latter being similar to the model studied by Hillis. Our model is based on a simple evolutionary optimization process. We specify an external optimization problem or *evolutionary target* which is defined with respect to a so-called 'complete' set of test cases or *problems*. The fitness evaluation of the statically evaluated solutions is based on this 'complete' set of problems, whereas the partial fitness evaluation of the coevolving solutions, or *hosts*, is based on coevolving subsets of the 'complete' set. The evolutionary process is placed in a 2-D space which leads automatically to a tournament-like selection process as in the competitive evolutionary models mentioned above. Since we use a static evolutionary target we can easily compare static fitness evaluation and coevolving sparse fitness evaluation in terms of, for instance, optimization time and correctness of solutions. Another important advantage of using a static evolutionary target instead of a more open-ended target is that we can easily study how information regarding the target is integrated over evolutionary time.

The 'complete' set of problems does not change during evolution. The parasites on the other hand can mutate, changing some of the problems. It is important to note that the problems contained in the parasites are elements of the 'complete' set of problems. Thus in both fitness evaluation regimes the solutions can attain maximum fitness by solving all problems of the complete set. However, the (partially evaluated) hosts can also attain maximal fitness by solving only those problems of the complete set on which they are actually

[1]Both competitive and cooperative evaluation schemes are used in models in which the evolutionary target is predefined and thus static, and in models in which the evolutionary target is defined solely with respect to the behavior of the opponent or the cooperator.

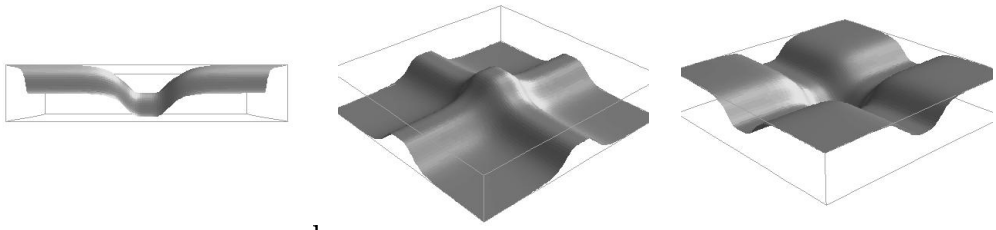
evaluated. Of course the parasites that specified those problems will have minimum fitness and will thus quickly be outcompeted or mutated.

We code the solutions which are to approximate the evolutionary target in functional form (i.e. as a program) as in genetic programming. Such coding leads to a multitude of implementations of any one function. The resulting many-to-one mapping between genotypes, i.e. programs, and phenotypes, i.e. programs evaluated on a certain set of problems, influences the evolutionary process considerably (c.f. Schuster, 1989; Huynen et al., 1993; Altenberg, 1994; Hightower et al., 1995). The reverse is also the case: different evolutionary processes may consistently lead to different types of genotypes (Huynen and Hogeweg, 1994). In our model the two fitness evaluation schemes lead to programs which markedly differ at the genotypic level in terms of mutational stability, as well as at the phenotypic level in terms of generalizability.

2 The Model

We studied the two different fitness evaluation schemes in the context of several evolutionary targets. In most of these studies the same trend is visible. First we will describe a model that incorporates one particular evolutionary target. In section 3 we will discuss results obtained from this one model. After that we will briefly describe results of models with higher dimensional evolutionary targets.

The evolutionary target that we used in the following model is a simple 2-D numerical function: $\frac{1}{(1+X^{-4})} + \frac{1}{(1+Y^{-4})}$, see fig. 1. The problems on which the fitness evaluation of the solutions is based are simply X, Y values. The problems of the complete set are regularly distributed over the problem domain; 26×26 problems in the domain $X = -5.0, 5.0$ and $Y = -5.0, 5.0$ with an interval of 0.4. The problems of the coevolving sets of problems are elements of the complete set. Obviously the target function does not represent any biological function. However, using this artificial numerical function we can easily study the two different fitness evaluation schemes. In addition, the generalizability of evaluated solutions can easily be studied by changing the set of problems for which they are evaluated. It is also clear that the 'complete' and the coevolving sets of problems (the latter contains only nine problems, see below) 'cover' the domain of the target function in different ways. Whereas the 'complete' set of problems covers the domain adequately,



a. b. c.

Figure 1: A plot of the target function, with a view from a) the side, b) the bottom and c) from the top. The X,Y domain is [-5,5][-5,5], the maximum of the function approaches 2.0, the minimum (X=0, Y=0) is 0.

the coevolving sets clearly do not.

We embedded the populations of solutions and parasites in space. We used a 2-D toroidal square lattice, with one solution and one parasite per grid cell. The size of the lattice is 50×50 cells, giving population sizes of 2500. Competition for growth is local in space. In each 3×3 neighborhood the solutions are ranked based on their fitness value in ascending order. The i^{th} ranked solution is selected with a probability of $(\frac{1}{2})^i$. The selected solution will grow into the central cell of the nine cells under consideration. The same growth procedure affects the parasites in the coevolutionary case, except that there the ordering is reversed; parasites are ranked in descending order.

The fitness of a solution is defined as the mean of the absolute differences between the target function and the solution over all problems on the basis of which it is evaluated. A solution is considered completely 'correct' if, for all 676 problems in the 'complete' problem set used in the static evaluation scheme, the absolute difference between solution and target function is less than 0.01 (this is a so-called hit).

In the static evaluation scheme the fitness of solutions is based on all 676 problems of the complete set of problems, whereas in the coevolving evaluation scheme the fitness of solutions is based on the 9 problems of the parasites in the surrounding 3×3 neighborhood. The fitness of a coevolving parasite is defined as the absolute difference between the target function and the solution (in the same grid cell as the parasite) evaluated on the basis of that parasite. Since the parasites are confronted with only one host they are more likely to be affected by random fluctuations, such as mutations, in that

host. The fitness evaluation of hosts, on the other hand, is based on nine parasites, so changes in one parasite have a less drastic effect on the fitness evaluation of the host. We found that this asymmetric fitness evaluation gives better results with respect to optimization time, than the symmetric fitness evaluation.

The parallelization of the evolutionary process resulting from the spatial embedding leads to an increase in the genetic diversity of each population and thereby possibly to enhanced performance of the optimization process (Collins and Jefferson, 1991). Combined with the localized interactions within and between populations the spatial embedding can lead to specialization of the coevolving populations with respect to each other (Husbands, 1994) and thus to the natural incorporation of features such as niching and sharing (Rosin and Belew, 1997; Mahfoud, 1995). Clearly, sharing also occurs automatically in the statically evaluated model as a result of the spatial embedding.

The genetic representation of the solutions is based on genetic programming. That is, the function that we use set is composed of: $\{+, -, *, \%\}$. The division operator % is said to be protected in the sense that division by zero gives 1.0. In genetic programming the division operator is normally implemented in this way to ensure that the programs maintain syntactic closure under the genetic operators (Koza, 1992). The terminal set is composed of: $\{X, Y, \Re\}$, where $\Re$ is the ephemeral random constant (Koza, 1992). Note that one does not have to use constants in order to create a correct program. The constant 1.0 in the target function can easily be obtained by dividing one variable by itself. In fact, not all functions in the function set are needed to create a correct program; the target function can be implemented with only division, plus either addition or subtraction. The use of superfluous function and terminal sets increases the number of possible implementations of a correct solution, and thus the freedom of the evolutionary process to 'choose' a program. We used crossover and point mutations as genetic operators, with probabilities of 40 and 20 per cent respectively.

The genotypes of the parasites, which in this model specify only one X,Y-problem, are simply the values of the variables. Mutation of a parasite means that one of the variable values is changed into a neighboring value, i.e. plus or minus 0.4. Ten per cent of all parasites are mutated every time step. The genotype space of the parasites is not toroidal: parasites with extreme variable values (-5.0 or 5.0) can only mutate in one direction.

Simulations are started with small, randomly created programs of max-

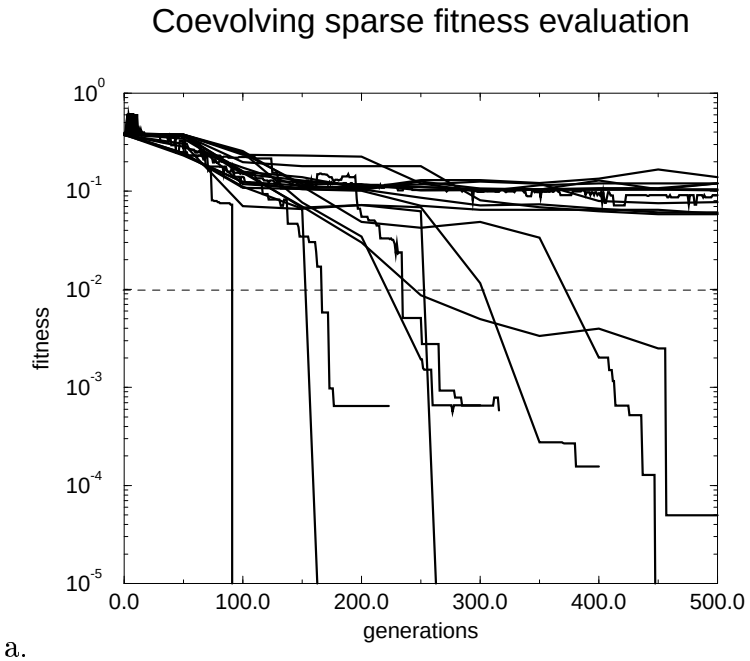
imum depth 3 and, in the case of the coevolutionary regime, parasites with the values X=0.2, Y=0.2. Simulations were stopped either when no correct solution was found within 500 time steps, or when a correct solution was found and retained in the population for 50 time steps. The demand for retention of the correct solution for 50 time steps was based on the idea that the coevolving evaluation scheme might produce a correct solution but could not keep it in the population due to lack of selection for complete correctness. As it turned out, this did not happen in any of the simulations; in fact solutions defined as correct were often (slightly) improved upon during these final 50 time steps.

3 Results

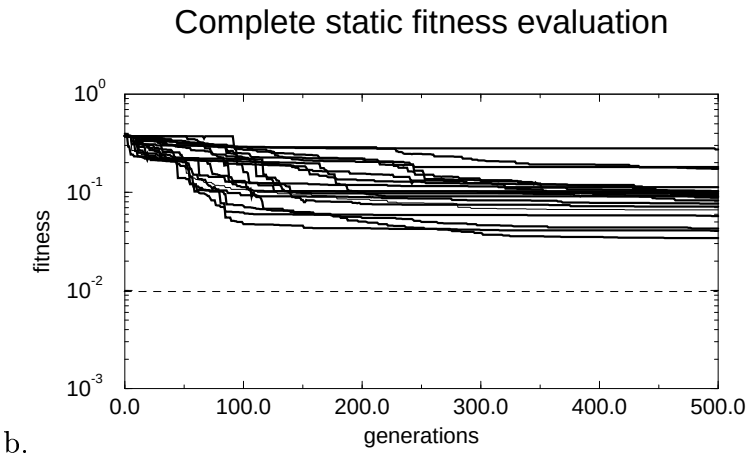
We report on 20 simulations for each of the two types of fitness evaluation schemes with the 2-D evolutionary target: complete static and coevolving sparse evaluation. During the simulations we recorded the fitness of the best solution based on the 'complete' problem set (the so-called total fitness) at that time step (best in the sense of total fitness) in order to compare the different evaluation schemes. The success rates of the two evaluation schemes are described in the following section. Thereafter, we will investigate results concerning the generalizability of the evaluated programs of the different evaluation schemes and describe the differences in their mutational stability.

3.1 Success Rates

Figure 2 shows the fitness curves of the best-of-generation solution for all simulations of the two evaluation schemes. Table 1 shows the percentage of simulations that produce correct solutions for the two evaluation schemes. The correct solutions found by the coevolving evaluation scheme come in two varieties. A number of solutions have fitness values ranging from 10^{-15} to 10^{-17}, while the others have fitness values ranging from 10^{-2} to 10^{-4}. These fitness values reflect whether the solution is an exact or only an approximate implementation of the target function. Solutions that implement the target function exactly can be easily rewritten in the same form as the target function. Although such 'perfect' solutions may still contain constants, these constants are not functional in the evaluation of the program. They are either multiplied by zero or two equal constants are subtracted from each



a.



b.

Figure 2: Fitness curves of the best-of-generation solution for coevolving a) and complete static problem evaluation b). Fitness is based on the 'complete' problem set that consists of 26×26 problems. The fitness curves that drop below 10^{-5} go to values between10^{-15} and 10^{-17}. The horizontal dotted lines give the value of the hit criterium (see text).

evaluation scheme	size of problem set	success ratio	mean number of nodes in final program
static	676	0%	68
coevolving	9 out of 676	45%	44

Table 1: The success ratios and mean size of the final solutions for the different evaluation schemes.

other. The constant 1.0 in the target function is formed by the term X/X or Y/Y. The fact that the fitness values are larger than zero is a consequence of the finite numerical precision of the fitness calculation. Solutions which approximate the target function (i.e. correct solutions with fitness values in the range 10^{-2}, 10^{-4}) still contain constants that affect the evaluation of the solution.

The two evaluation schemes differ considerably in the number of problems evaluated per fitness evaluation; completely evaluated solutions compute $676/9 \approx 75$ times more problems than the coevolved solutions. Furthermore, the completely evaluated solutions are larger than the coevolved solutions (table 1); thus every single evaluation takes longer. On the other hand, the coevolution of problems takes some time too, as does the periodic computation of the total fitness of the coevolving solutions. Nevertheless, we found that simulations with complete fitness evaluation required 5 to 10 times more computer time than the simulations with coevolution.

3.2 Generalizability

We studied the generalization capabilities of the evolved solutions by increasing the sampling density of the program evaluation. The standard 'complete' static evaluation is based on 26×26 problems; the dense evaluation is based on 100×100 problems. Perfect solutions implement the target function exactly. Thus, plots of a perfect solution evaluated based on the dense set of problems are identical to plots of the target function (see fig. 1). Figures 3a and b show the standard (left) and dense (right) calculated evaluations of correct solutions from two simulations with the coevolving evaluation scheme. Neither solution is an exact implementation of the target function but is an approximation thereof. The right plot of fig. 3a shows that although the solution is not perfect it generalizes very well over the points that are not

included in the set of problems used in the simulation. The dense plot of fig. 3b shows that the solution does not generalize very well for a small subset of the dense set of problems (i.e. $X = 0$). Figure 3c shows the standard and dense calculated evaluations of an incorrect solution. Although the solution is not correct it is surprising to find that it nevertheless generalizes so well.

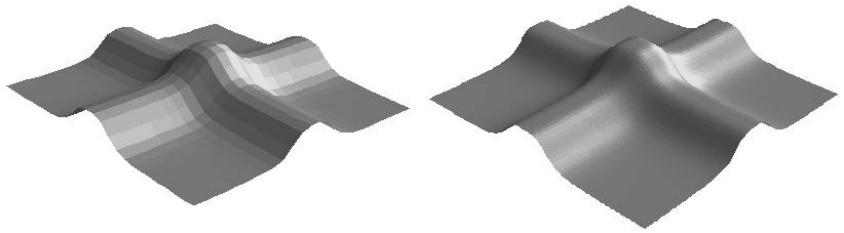
If we study the generalization capabilities of evolved solutions from simulations with the static evaluation scheme we get quite different results. Figure 4a and b show plots of standard (left) and dense (right) calculated evaluations of the best solutions of two simulations. Although the left plots show that the programs yield approximately correct values for the set of X,Y values on which they evolved, the right plots show that they generate absurd values for intermediate X,Y values.

The solutions produced by the static evaluation scheme seem to get trapped in the complexity of the functions they implement. In the process of further adaptation to the target function, due to the 'complete' sampling of the problem domain, the solutions are forced to conserve any adaptation already achieved. The 'complete' sampling results in a severe selection against the occurence of errors. Although the increase in the numerical complexity of the solutions gives the programs the opportunity to adapt to individual problems and thus increase their fitness the increased complexity reduces the generalizability of the solutions.

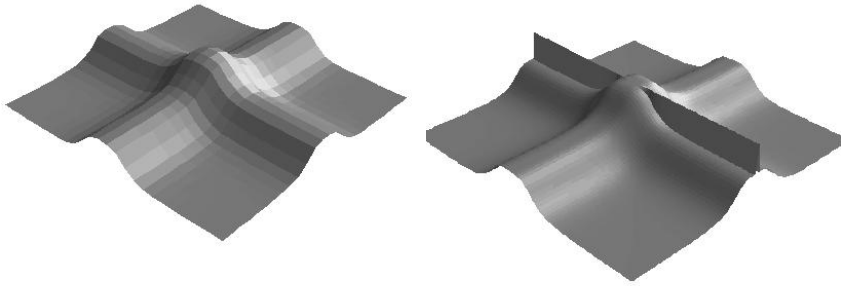
The solutions produced by the coevolving evaluation scheme have more freedom to make errors. As long as an error in the solution is not exposed by the local problems the solution can remain in the population. However, errors in solutions are simple evolutionary targets for the coevolving problems. Thus, sooner or later, errors in solutions will be selected against and these solutions will be expelled from the population.

3.3 Mutational stability

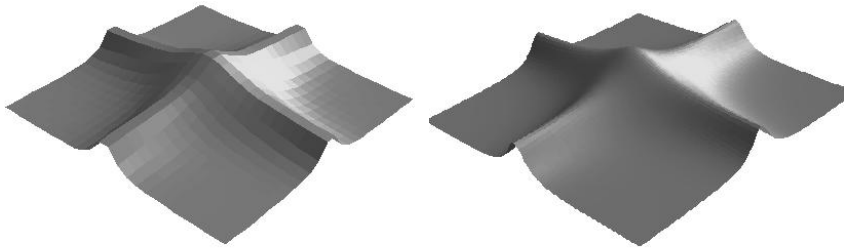
In order to study the effects that the two evaluation schemes have on the structures of the programs, we compare the final programs with 1000 of their one point mutants, i.e. programs that differ from the final program by one point mutation. The original and mutant programs are compared for all 676 problems in the standard problem set. If the absolute difference between the original program and the evaluated one-point mutant is less than the hit criterion for a certain problem (an absolute difference of less than 0.01) the mutant scores a hit on that problem. Thus, if many mutants score



a.



b.



c.

Figure 3: Three typical final solutions produced by coevolving fitness evaluation. The left plots are based on 26×26 evaluated problems, the right plots on 100×100 evaluated problems. a) and b) are two correct solutions that approximate the target function c) is an incorrect solution. All solutions generalize well on the 100×100 problems.

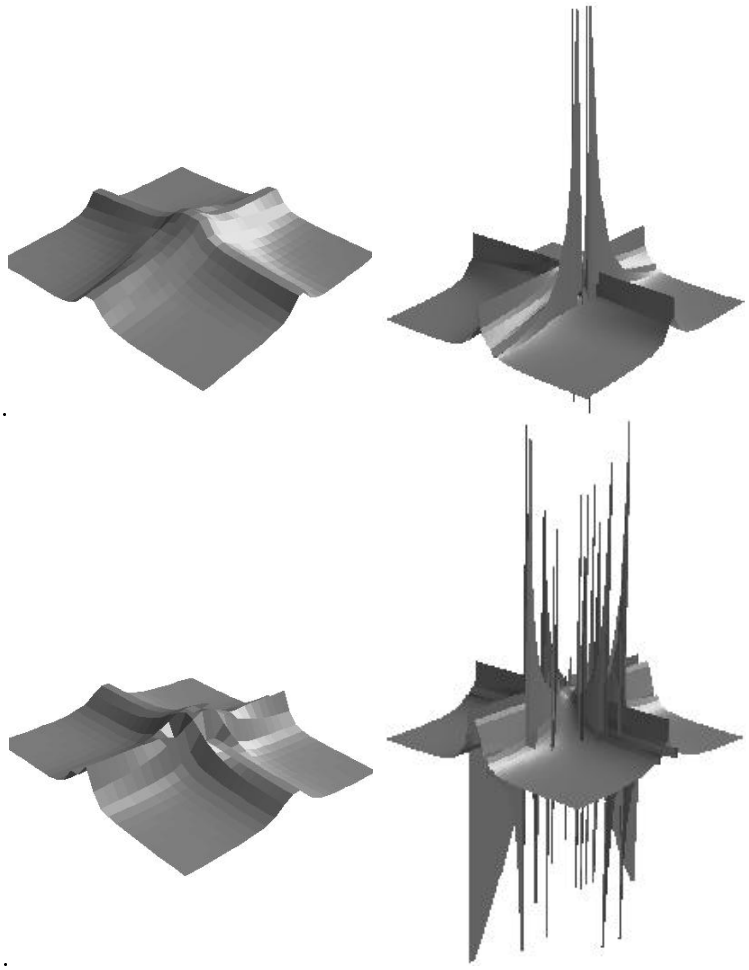


Figure 4: Two typical final solutions produced by static fitness evaluation. The left plots are based on 26×26 evaluated problems, the right plots on 100×100 evaluated problems. Neither solution is correct.

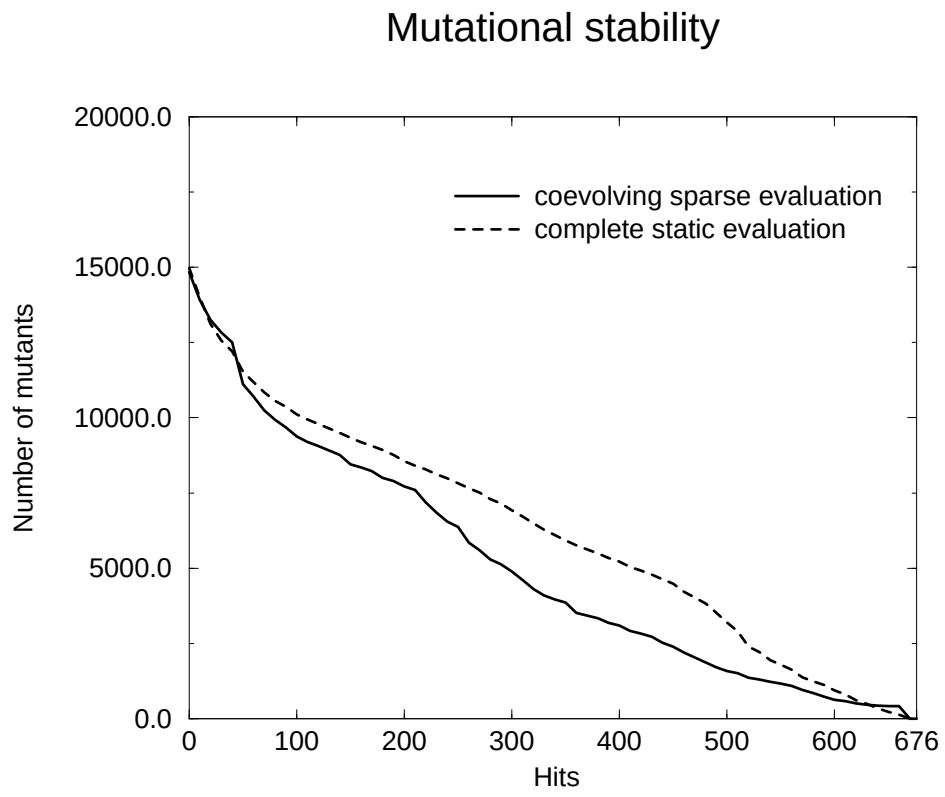


Figure 5: Histogram of the number of one-point mutants having at least x number of hits.

many hits the original program is phenotypically stable under mutations. For each evaluation scheme we looked at 1000 one-point mutants of the final program for all simulations. In figure 5 we plot the similarity between all 20,000 mutants and their respective final programs for the two evaluation schemes. On the horizontal axis we plot the number of problems for which a mutant scores a hit, or in other words: the number of problems for which the mutant is near identical to its original final program. On the vertical axis we plot the number of mutants that score at least x number of hits. It is clear that statically evaluated programs are more stable than the programs of the coevolving evaluation scheme (significance< 0.01 on the Kolmogorov-Smirnov test).

It is not true that the difference in stability depends on the fact that the coevolving evaluation scheme produces correct programs whereas the static

evaluation scheme does not. If we look only at the coevolved programs we find no difference in the stability of correct and incorrect programs. We also compared the mutational stability of ten incorrect programs for both evaluation schemes but found no significant difference between those and the results shown here. The difference in the mutational stability of the programs that have evolved under the different evaluation schemes does not depend on the difference in the size of the programs either. If we compare the mutational stability of the final programs for one evaluation scheme we find no correlation between size and stability.

3.4 Higher dimensional target functions

We also studied the 3-, and 4-D extensions of the 2-D evolutionary target which are simply extensions on the numerical function: $\frac{1}{(1+X^{-4})} + \frac{1}{(1+Y^{-4})} + \frac{1}{(1+Z^{-4})} + \ldots$, etc. In order to economize on computational time and resources we had to limit the problem domain to $X = -3.0, 3.0$ and $Y = -3.0, 3.0$. The interval was 0.4 again, thus limiting the number of values along one dimension to 16. In the 3-D case the total number of problems is 4096, in the 4-D case the total number is 65536. In the 3-D case we increased the lattice size to 100×100 cells, in the 4-D case to 150×150 cells. We found that the coevolutionary evaluation scheme did not find correct solutions in the smaller field of the 2-D target but it did in the larger lattices. Except with respect to these changes the model was identical to the 2-D model. With the 3-D target we ran 5 simulations of the static evaluation scheme and 10 runs of the coevolutionary evaluation scheme. The 4-D case was run only for the coevolutionary evaluation scheme since complete evaluation is extremely time consuming. These coevolutionary simulations were run for 500 time steps without complete evaluation. After that we performed a sparse evaluation on the basis of random sets of problems in order to get an indication of the total fitness of the solutions. Only if this indication was positive did we search for a correct individual.

In general the results of these simulations are very similar to those with the 2-D target. The static evaluation scheme does not produce any correct solutions in the 3-D case, whereas the coevolutionary evaluation scheme produces correct solutions in 5 out of 10 runs. The generalizability and mutational stability of the solutions in the 3-D case give similar results to the 2-D case. Statically evaluated solutions are much less generalizable and are

mutationally more stable than coevolved solutions. However we found in rare cases that coevolutionary evaluation may also evolve to a mutationally stable program. This happens, for instance, in very unsuccessful runs if the program yields a constant value independent of the problem values on the basis of which it is evaluated. Contrary to the mutationally stable programs that evolved under static evaluation the coevolved stable program is very generalizable since it simply yields a constant value.

We performed only three runs of the coevolutionary evaluation scheme with the 4-D evolutionary target, due to long simulation times. In these three runs we found two correct solutions. We have not attempted any further extensions of this evolutionary target since this would require an extensive increase in computational resources. It is clear, however, that coevolutionary fitness evaluation can still produce correct solutions under an increase in the dimensionality, and thereby in complexity, of the evolutionary target. Furthermore, the evolution of the different types of solutions is independent of the dimensionality of the evolutionary target. In all models we see that the coevolved solutions are more generalizable and mutationally less stable.

4 Discussion

In the previous section we showed that the evaluation based on small coevolving sets of problems and evaluation based on a large static set of problems differed in their success rate; the coevolving evaluation gives correct solutions in roughly half of the simulations, whereas 'complete' static evaluation does not produce correct solutions in any of the simulations. Even more importantly we showed that the types of solutions that evolved under the different evaluation schemes differ markedly. The coevolved solutions are more generalizable, less complex and mutationally less stable than the statically evaluated solutions. These differences between the solutions reflect the multiplicity of the coding of phenotypes in genotypes; genotypes which differ in generalizability, mutational stability and complexity can map to similar phenotypes, i.e. phenotypes that approximate the same evolutionary target. Although the degree to which the solutions differ in these properties may depend on model-specific properties, the qualitative difference is consistent in all simulations.

In the following sections we will discuss the results of the model. First we will discuss some aspects of the coevolution of solutions and problems in

terms of an evolutionary optimization process. Thereafter we will discuss the effect of the two evaluation schemes on the type of solutions that evolved.

4.1 Coevolution and optimization

Several authors have suggested that coevolutionary fitness evaluation is more successful than 'complete' static fitness evaluation because the coevolving problems sample the problem domain more efficiently. (Hillis, 1992; Paredis, 1994; Paredis, 1995). The idea is that the parasites sample particularly those problems in the problem domain that have not yet been solved by the solutions; as a result the fitness evaluation process becomes focused on 'hard' problems. In order to study the effect of this focusing on the coevolving problems in our model we also studied a variant of this model. In this variant the solutions are evaluated on nine problems (as in the coevolving model) which are randomly chosen from the 'complete' set at every fitness evaluation. Thus here the evaluation is sparse but does not coevolve. First, however, we will discuss briefly a few results from a similar but much simpler model in order to establish a baseline with respect to the efficiency of coevolutionary optimization. In this model the genotype-phenotype mapping is a one-to-one mapping; contrary to the previous model, the evolutionary target here can be represented in only one way in the genotype.

4.1.1 Coevolution towards a simple linear evolutionary target

In this model we use bitstrings as the representation for the solutions, an arbitrarily chosen bitstring as evolutionary target, and a simple additive fitness function for the fitness evaluation. In other words this is a simple imitation function. Except with respect to these aspects this model is the same as the original model.

The complete set of problems here consists simply of all bits in the string, each specified by its position. In the complete static evaluation scheme[2] solutions are evaluated on the basis of all positions in the bitstring. The corresponding fitness landscape is smooth with one global maximum. In the coevolving evaluation scheme each parasite specifies three positions in the bitstring. Hosts are evaluated based on 9 parasites again, thus they are evaluated on the basis of 27 positions. We also studied random sparse

[2]Note that complete evaluation in this model is truly complete.

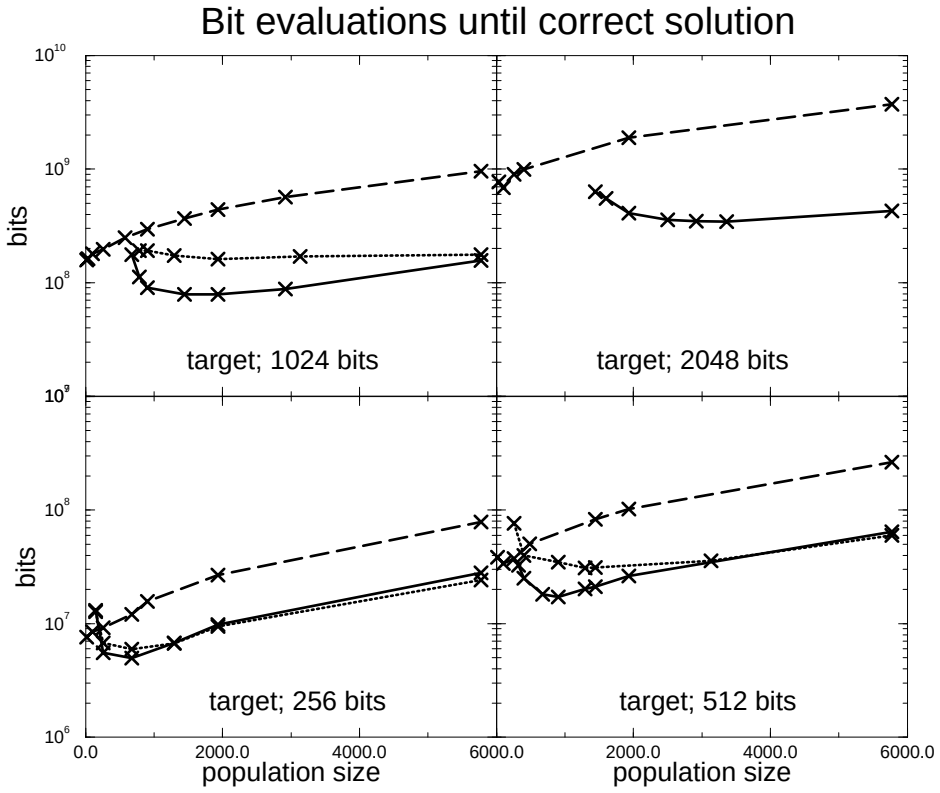


Figure 6: Coevolving (solid line), static (dashed line) and random (dotted line) fitness evaluation of a simple linear problem. Optimization time in terms of bit evaluations. Data are averaged over 5 runs.

evaluation. In this model each solution is evaluated based on 27 randomly chosen positions at every fitness evaluation. The latter two models have fitness landscapes that are more rugged compared to the first fitness landscape. The ruggedness reflects the sparse and dynamic nature of the fitness evaluation in these models.

We performed simulations of this evolutionary model for target strings of different lengths; 256, 512, 1024 and 2048 bits, each with several population sizes. The data are averaged over five runs. For all fitness evaluation regimes we record the evolutionary time required until a bitstring is found that is completely correct. Obviously in terms of evolutionary time the static evaluation scheme needs fewer steps than the coevolutionary sparse evaluation scheme. However, the number of bit evaluations, and thus the computational

cost per evolutionary step, differ greatly for the different evaluation regimes. For the coevolutionary and the random evaluation scheme the number of bit evaluations per evolutionary step is fixed. For the static evaluation scheme the number of bit evaluations is equal to the length of the target bitstring.

In fig. 6 we plot for each target string length the number of bit evaluations required until a correct bitstring is found. Now we see that for each target string length there is a population size for which the coevolutionary evaluation scheme is more efficient than the most efficient population size of the static evaluation scheme. In fact the range of population sizes for which this is true is quite large. The static evaluation scheme on the other hand is characterized by the very small range of relatively good population sizes. The random evaluation scheme shows the same trend as the coevolutionary runs for shorter string lengths. However, for longer strings random evaluation performs less efficiently. For the longest target string, i.e. 2048 bits, the random evaluation scheme does not find a correct solution in any of the runs.

For the shortest string length, i.e. 256 bits, a single random evaluation evaluates 10% of the complete problem set. Note that solutions are evaluated independently but selected with respect to each other. Also they have a certain lifetime during which they are evaluated on the basis of different random sets of problems. Thus, for shorter strings the random evaluation scheme resembles more and more the complete evaluation scheme. The parasites in the coevolving evaluation scheme on the other hand are much more structured, in time as well as in space. The temporal change in parasites is gradual via mutations. The spatial structuring is a result of local growth. Both properties lead to a variation in the evaluated problems, which is expected to be much smaller in the coevolutionary case than in the random evaluation case.

On the other hand, for longer target strings the random evaluation performs less efficiently. The failure of the random evaluation scheme in the 2048 bit target string runs is due to the fact that the population of solutions cannot retain already acquired information about the target function. The average fitness of the population of solutions levels off fairly quickly without ever producing a completely correct solution. In fact for longer target string lengths the solutions rise above the error threshold (Eigen et al., 1989). It is not clear to what extend the coevolving parasites introduce extra information on which the evolutionary selection pressure can act. The results presented here show that the efficiency increase due to 'focused' evaluation (which happens in the coevolving but not in the random evaluation scheme)

is visible only for larger evolutionary targets.

A surprising result is that for both the linear and the nonlinear models (see 3.4) the coevolutionary evaluation scheme seems to work best if the population size is of the same order as the size of the 'complete' set of problems. We have not pursued this finding further but think that future studies should investigate this relation.

From this simple model we can draw several conclusion. First, evaluation on only a small part of the evolutionary target can nevertheless lead to integration of the complete evolutionary target in the solutions, given a sufficiently large population and local competition. Second, although the coevolving evaluation scheme needs much larger populations than the population in the static evaluation regime the difference in the computational cost nevertheless favors the coevolutionary evaluation scheme due to the sparse evaluation per individual per time step. Third, random sparse evaluation performs similarly to coevolutionary sparse evaluation with respect to smaller evolutionary targets. For large target strings random partial fitness evaluation *falls* over the error threshold. Coevolving evaluation on the other hand can still provide a high enough selection coefficient in these cases.

4.1.2 Sparse and dense random fitness evaluation; efficiency and focusing

In the original model we also studied a sparse random evaluation scheme. In the sparse random evaluation model the fitness of the solutions was evaluated on the basis of nine random problems. The random problems were chosen from the standard 'complete' problem set, and were chosen anew for every fitness evaluation. In 20 simulations 7 correct solutions were found; thus the difference between the success rate of this random evaluation scheme (35%) and that of the coevolving evaluation scheme (45%) is small. The final solutions of this random evaluation scheme are comparable to the final solutions of the coevolutionary evaluation scheme in the sense that both sets have good generalizing capabilities and are mutationally unstable. In the 2-D model the coevolving evaluation scheme is not much more efficient than the random evaluation scheme. In the previous section we showed that for relatively small evolutionary targets random fitness evaluation performs similarly to coevolutionary fitness evaluation with respect to optimization time. For larger targets coevolutionary evaluation outperforms random evaluation due to the fact that the coevolving parasites focus on hard problems.

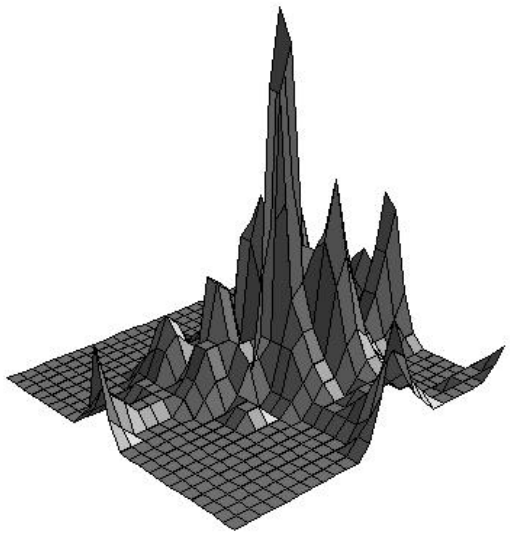


Figure 7: The coevolving problems focus on difficult regions in the problem domain. The height of each point depicts the total number of occurrences of that problem in the problem population during the evolution. The problems focus on the regions near X=0 and Y=0.

In fact, the random evaluation scheme in the original model with the 3-D evolutionary target cannot find correct solutions for this larger evolutionary target.

Although in the coevolutionary 2-D model the focusing of the parasites does not lead to a large increase in success rate, the selection of fit parasites does lead to focused sampling of the problem domain. Figure 7 shows that the coevolving problems focus on difficult regions during evolution. After allowing for an initial transient of 100 time steps we counted the number of times that a particular problem was present in the problem population and plotted these values for all problems in the domain. The plot shows that particularly the centre of the domain and the regions around X=0 and Y=0 are sampled by the coevolving problems. The effect on the success rate caused by the focusing of the parasites is probably negligible in the 2-D model because it is counteracted by the poorer variety of problems that are present in the parasite population, as mentioned previously.

The solutions that have evolved under the coevolutionary and random evaluation scheme are more generalizable and mutationally less stable than

the solutions evolved under the complete static evaluation scheme. However, the evolution of generalizable and mutationally unstable solutions does not necessarily result in correct solutions.

A second variant that we studied is an evaluation scheme in which the solutions are evaluated on the basis of 676 randomly chosen problems which are not elements of the 'complete' set of problems. These randomly chosen problems are uniformly distributed over the domain. In this case evaluation is not sparse but it does vary over time. The mutational stability of the programs produced in these simulations is even lower than the stability of the coevolved programs and the generalizability is similar to that of these programs. Nevertheless, only two correct solutions were found in 20 simulations. Thus, generalizability and mutational stability can evolve without increasing the success rate appreciably.

An important factor of the coevolving and the random evaluation scheme is that the solutions are evaluated on the basis of a very small subset of the 'complete' set of problems. This increases the freedom of the evolutionary process to traverse the space of possible solutions. The dense random evaluation scheme samples the evolutionary target to such an extent that solutions cannot make large mistakes easily. A clear example of the freedom in the evolutionary process is the following simulation taken from the coevolving evaluation scheme. During this simulation we traced the ancestry of the correct final solution. The final solution was perfect in the sense that it was a direct implementation of the target function $\frac{1}{(1+X^{-4})} + \frac{1}{(1+Y^{-4})}$. The parent of this final solution had as second term: $\frac{1}{(1+Y^{-3})}$, the grandparent had as second term: $\frac{1}{(1+Y^{-2})}$. The parent of the final solution has a total fitness of the order of 10^{15}. In fact, at the moment when the final solution was produced, the parent had a local fitness[3] about 10^{15}. Before that, however, the parent had a low local fitness of the order of 10^{-3}. This is the result of the sparse sampling of the problem domain. The grandparent and the parent solutions were produced at a moment when the local problems were at the edge of the domain, i.e. in the flat regions. In this part of the domain the transition of the second term from $\frac{1}{(1+Y^{-2})}$ to $\frac{1}{(1+Y^{-3})}$ does indeed result in lower fitness values. From that point onwards parasites mutated towards the centre of the domain, resulting in the exposure of the error in the second term. By that time the final solution had been produced with the correct second term.

[3]Local fitness is based on only the (nine) problems on the basis of which a solution is evaluated in the coevolutionary evaluation scheme.

Thus the sparseness of the evaluation helps (rather than hinders) the search process.

4.2 Side-effects of variable problem sampling

Clearly, generalizability is a very important property of evolving entities, be they solutions to optimization tasks or biological organisms. In our model we see a clear relation between the generalizability of the solution and the type of fitness evaluation we use. Generalizability appears to be a side-effect of the evolutionary dynamics resulting from fitness evaluation on the basis of a varying set of problems. Generalizability is of course a good strategy for coping with changing environmental conditions, here in the form of changing sets of problems. Thus, although there is no direct selection for generalizability, variable problem sampling may indirectly select for solutions which are generalizable.

Just as generalizability is not directly selected for in the coevolving and the random evaluation schemes, mutational stability is not directly selected for either. Nevertheless, solutions that are evaluated on the basis of coevolving or randomly selected problems are consistently mutationally less stable than statically evaluated solutions.

Mutational instability is a second possible strategy for counteracting changing environmental conditions. A mutationally unstable program is better able to adapt to new conditions than a mutationally stable program (this is also discussed by Thompson (1996) in the context of error correction by evolving entities). A decrease in genetic stability as a response to changing environmental conditions in RNA landscapes has been reported previously by Huynen et al. (1993). It is interesting to see that evaluation under variable "environmental" conditions (i.e. problem sets) produces solutions which implement the two strategies for coping with variable evaluation, namely mutational instability and generalizability.

Note that mutational stability and generalizability are two properties that render a solution robust. Mutational stability reflects genetic robustness; small changes in the genotype have little effect on the phenotype. Generalizability is a phenotypic measure of robustness; small changes in environmental input produce similar phenotypic responses. In our model we see that solutions are either robust in the sense of being generalizable, or they are robust in the sense of being mutationally stable, but not both. It is not clear whether mutational instability and generalizability are properties that are necessarily

linked. Even if they are not it might be that neither of these properties is attainable without the other.

4.3 Conclusion

With respect to evolutionary optimization processes we see that in our model the coevolution of problems and solutions does indeed yield better results than complete static fitness evaluation. This implies of course that sparse dynamic fitness evaluation can result in the complete integration of an evolutionary target. This is not at all a trivial finding. An important aspect of the success of coevolutionary evaluation is in fact the sparse sampling of the problem domain. We have shown that sparse sampling gives the solutions the opportunity to make large errors with respect to some problems as long as the solutions are not evaluated on the basis of these problems. This gives the evolutionary process greater freedom to explore the space of possible solutions.

More interesting than the difference in the success rate is our finding that the coevolutionary and the static evaluation schemes evolve different types of solutions. The coevolutionary evaluation scheme leads to solutions that are more generalizable, mutationally less stable and less complex than the solutions produced under the static evaluation scheme. The differences in the properties of the solutions that evolve under the different fitness evaluation schemes are side-effects of evolutionary dynamics; none of these properties has a direct effect on the fitness of the solutions and thus is not directly selected for. An evolutionary process which acts on genotypic representations that incorporate a many-to-one genotype-phenotype mapping can mould the genotypes in different ways. Nevertheless, the different genotypes can implement phenotypes that approximate the evolutionary goal to a similar extent. Such side-effects can occur only in genetic coding schemes that incorporate a multiple mapping from genotypes to phenotypes, as is the case in ,for instance, genetic programming or the natural genetic code, but is generally not the case in classical genetic algorithms. With respect to evolutionary optimization models which do incorporate such a multiple genotype-phenotype mapping this result suggests that a simple change in fitness evaluation can produce more generalizable solutions.

With respect to the biological evolutionary process, it is clear that neither the 'complete' static, nor the sparse variable set of problems serves as a good approximation of the fitness evaluation in nature. However, we con-

sider the static completeness and the variability of fitness evaluation as a continuous transition from static total sampling to variable sparse sampling. Our results suggest that this transition has a large impact on the genotypic structures that evolve. Static sampling leads to complex and mutationally stable solutions with low generalizing capabilities. Variable sparse sampling on the other hand leads to much less stable and simpler solutions with high generalizing capabilities. With regard to the natural evolutionary process, our results show that not all properties of evolving entities are the result of direct selection. Many properties, such as being robust on one level or the other, can be side-effects of the genotypic structuring that results from several aspects of the evolutionary process.

Acknowledgements:
The authors thank Ms. S.M. McNab for linguistic advice. They also thank the referees for constructive criticism and helpful suggestions. The investigations were supported by the Life Sciences Foundation (SLW), which is subsidized by the Netherlands Organization for Scientific Research (NWO).

References

Altenberg, L. (1994). The evolution of evolvability in genetic programming. In Kinnear, Jr., K. E., editor, *Advances in Genetic Programming*, chapter 3, pages 47–74. MIT Press.

Angeline, P. J. and Pollack, J. B. (1993). Competitive environments evolve better solutions for complex tasks. In Forrest, S., editor, *Proceedings of the 5th International Conference on Genetic Algorithms*, pages 264–270, San Mateo, CA. Morgan Kaufmann.

Collins, R. J. and Jefferson, D. R. (1991). Selection in massively parallel genetic algorithms. In Belew, Richard K.; Booker, L. B., editor, *Proceedings of the 4th International Conference on Genetic Algorithms*, pages 249–256, San Diego, CA. Morgan Kaufmann.

Eigen, M., McCaskill, J., and Schuster, P. (1989). The molecular quasi-species. *Adv Chem Phys*, 75:149–263.

Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization, and Machine Learning*. Reading. Addison-Wesley, Massachusetts.

Hightower, R. R., Forrest, S., and Perelson, A. S. (1995). The evolution of emergent organization in immune system gene libraries. In Eshelman, L. J., editor, *Proceedings of the 6th International Conference on Genetic Algorithms*, pages 344–350, San Francisco. Morgan kaufmann Publishers.

Hillis, W. D. (1992). Co-evolving parasites improve simulated evolution as an optimization procedure. In Langton, C. G., Taylor, C., Farmer, J. D., and Rasmussen, S., editors, *Artificial Life II*, volume X of *Sante Fe Institute Studies in the Sciences of Complexity*, pages 313–324. Addison-Wesley, Santa Fe Institute, New Mexico, USA.

Holland, J. H. (1992). *Adaption in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence*. MIT Press.

Husbands, P. (1994). Distributed coevolutionary genetic algorithms for multi-criteria and multi-constraint optimisation. In Fogarty, T., editor, *Evolutionary Computing, AISB Workshop Selected Papers*, volume 865 of *Lecture Notes in Computer Science*, pages 150–165. Springer-Verlag.

Huynen, M. and Hogeweg, P. (1994). Pattern generation in molecular evolution: exploitation of the variation in rna landscapes. *Journal of Molecular Evolution*, 39:71–79.

Huynen, M., Konings, D., and Hogeweg, P. (1993). Multiple coding and the evolutionary properties of rna secondary structure. *Journal of Theoretical Biology*, 165(2):251–267.

Juillé, H. and Pollack, J. B. (1996). Co-evolving intertwined spirals. In Fogel, L. J., Angeline, P. J., and Baeck, T., editors, *Evolutionary Programming V: Proceedings of the Fifth Annual Conference on Evolutionary Programming*, pages 461–467. MIT Press.

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

Mahfoud, S. W. (1995). A comparison of parallel and sequential niching methods. In Eshelman, L. J., editor, *Proceedings of the 6th International Conference on Genetic Algorithms*, pages 136–143, San Francisco. Morgan kaufmann Publishers.

Paredis, J. (1994). Steps towards co-evolutionary classification neural networks. In Brooks, R. A. and Maes, P., editors, *Proceedings of the 4th International Workshop on the Synthesis and Simulation of Living Systems $ArtificialLifeIV$*, pages 102–108, Cambridge, MA, USA. MIT Press.

Paredis, J. (1995). Coevolutionary computation. *Artificial Life*, 2:355–375.

Potter, M. A. and De Jong, K. A. (1994). A cooperative coevolutionary approach to function optimization. In Davidor, Y., Schwefel, H.-P., and Manner, R., editors, *Proceedings of the Third Conference on Parallel Problem Solving from Nature*, volume 866 of *Lecture notes in computer science*, pages 249–257, Jerusalem, Israel. Springer-Verlag.

Potter, M. A., De Jong, K. A., and Grefenstette, J. J. (1995). A coevolutionary approach to learning sequential decision rules. In Eshelman, L. J., editor, *Proceedings of the 6th International Conference on Genetic Algorithms*, pages 366–372, San Francisco. Morgan kaufmann Publishers.

Rosin, C. D. and Belew, R. K. (1997). New methods for competitive coevolution. *Evolutionary Computation*, 5(1):1–29.

Schuster, P. (1989). Optimization of rna structure and properties. In Perelson, A. S. and Kauffman, S. A., editors, *Molecular evolution on rugged landscapes:proteins, RNA and the immune system*, pages 47–71. Addison Wesley.

Sims, K. (1994). Evolving 3D morphology and behavior by competition. In Brooks, R. A. and Maes, P., editors, *Proceedings of the 4th International Workshop on the Synthesis and Simulation of Living Systems $ArtificialLifeIV$*, pages 28–39, Cambridge, MA, USA. MIT Press.

Thompson, A. (1996). Evolutionary techniques for fault tolerance. In *Proceedings of UKACC International Conference on control 1996 (CONTROL'96)*, London, UK. The Institution of Electrical Engineers, IEE conference publications.

Vafaie, H. and De Jong, K. A. (1996). Genetic algorithms as a tool for restructuring feature space representations. In *Proceedings of the International Conference on Tools with AI*, Los Alamitos, CA. Institute of Electrical and Electronics Engineers, IEEE Computer Society Press.