The Adaptationist Stance and Evolutionary Computation

Márk Jelasity
Research Group of Artificial Intelligence
József Attila University, Hungary
Szeged H-6720, Aradi vértanúk tere 1.
E-mail: jelasity@inf.u-szeged.hu

Abstract

In this paper the connections between the evolutionary paradigm called adaptationism and the field of evolutionary computation (EC) will be outlined. After giving an introduction to adaptationism we will try to show that the so called adaptational stance can be applied in EC as well as in biology and this application may have significant benefits. It will also be shown that this approach has serious, inherent limitations in both cases especially in the case of EC, because we lack the *language* which could be used to form the theories, but these representational limitations can be handled by devoting efforts to construct this language.

1 ADAPTATIONISM

This section introduces adaptationism, a strategy for understanding the products of evolution. We will discuss only biological evolution here; the discussion of the relationship with EC is given in Section 2. Adaptationism is a controversial question [8] and it seems that most of the misunderstandings can be originated from the insufficient and obscure definition of the paradigm.

1.1 BASIC NOTIONS

In this section we will try to make the assumptions of adaptationism clear and explicit and to show the role of the underlying language. First it seems that adaptationism requires the following principles in order to be applicable:

P_1 **(separation):** The separation of the organism under investigation and its environment is necessary. To give an example, it is not clear whether the ant colony or a single ant counts as an organism. To be more precise both approaches are valid; only the choice has to be fixed. If the organism is the ant then the ant colony is a part of the environment that has to be constant according to P_3. This assumption can be difficult to maintain though it might me meaningful if the time interval under investigation is relatively small. This is a very controversial question, see [2] and [16] for two quite different viewpoints.

P_2 **(constraints):** It is necessary to determine the constraints that give the space of the biologically feasible genotypes. These constraints help us exclude caws with machine-guns, birds with jet engines, etc. This principle is needed to allow the optimality condition to be well-defined.

P_3 **(frozen environment):** Optimality can be given only w.r.t. a fixed environment; this makes it possible to block the circular definition of fitness[1]. Since the organisms cannot change the environment, the number of their offspring is solely the function of their properties. We must pay the price for this however; no dynamic properties of the population or its interaction with the environment can be examined. Of course, this limit diminishes if we choose a larger entity such as a population or an even larger subsystem such as a food chain to be our organism.

P_4 **(one niche):** Beside P_2 it is also necessary to restrict the possible individuals further only to those occupying a particular niche; roughly speaking the possible organisms cannot be *too* different. Optimality can be defined only inside of a species; this ensures that a pig will not be compared to a lion. Both may well be optimal in their own way but it would be a serious mistake to consider a pig a bad quality lion or vica versa. This constraint seems to be simple but in fact it causes major difficulties especially when fossils are analyzed since it is hard to tell with respect to what the organism in question should be optimal (is it a bad pig or a bad lion?). The definition of niche is itself a problem.

P_5 **(improving fitness):** We need to assume that evolution improves fitness. This will be one of the motivations of the optimality assumption to be introduced soon.

P_6 **(single organism):** This is partly the consequence of P_1; at every time-step there is only one organism. It may seem to be counter-intuitive at first sight. But according to P_1 we have to decide what the definition of organism is. If we choose an *individual*[2] animal or plant then all the other members of its species become part of the environment and

[1] If the number of offspring define fitness then this fitness cannot be used to predict the number of offspring since the claim "Individuals with high fitness will have high number of offspring" becomes a tautology. The problem is that this claim is in the center of Darwinism

[2] This choice does not mean that we are interested in a particular individual. This only means that in every generation we are interested in one (hypothetical) individual which is subject to natural selection in its population.

according to P_3 they are not allowed to change while our organism changes. It is clearly a plain contradiction though for *short* time intervals this approach could give fairly good approximations; in fact such discretization methods are quite common in mathematics for example in numeric approximations of differential equations.

Let us introduce a notation for the most important components. Let O be the set of organisms that are biologically feasible according to P_2 and live in a particular niche according to P_4. Let f be the fitness function, i.e. a function of type $O \to \mathbf{R}^+$. For an organism $o \in O$ the number $f(o)$ gives the expected number of offspring of o. f is also a function of the environment but since it is constant according to P_3, it is not indicated. Let $\mathcal{G}$ be the finite[3] set of features that make it possible to describe the organism. Its elements are functions of type $O \to \mathbf{R}$. Predicates are possible as well, in that case the function has only two values: 0 for false and 1 for true. An example could be the length of the neck or the degree of flying ability.

1.2 CHARACTERIZATION OF FEATURES

I would like to emphasize as early as possible that the characterization will be *independent* of the *function* of the given feature in the organism. At this level nothing is said about the roles or the causal relationships of the elements of $\mathcal{G}$. For instance we can say that the neck-length of a giraffe is optimal without mentioning its function (which could be for example reaching leaves at the top of the trees).

The aim of this characterization is to decide whether a given feature is relevant or irrelevant w.r.t. the fitness and if relevant then it is optimal or not. Therefore there are three categories: irrelevant, optimal and suboptimal. The interesting question is of course to find a method to somehow classify $\mathcal{G}$ into these three classes.

First let us examine the case when the fossil record is available that show the development of the feature under investigation. P_5 will be heavily exploited. P_6 is used too to ensure that there is no variance to be taken into account (only as another feature).

In the case of the *divergent* behavior we can conclude that at the moment the feature at hand is in a developing stage and therefore is *suboptimal*. In the case of *random* behavior we classify the feature as *irrelevant*. The situation is somewhat more difficult however. The random behavior can be caused by a lot of factors. The first possible explanation is that the given feature is neutral; it is independent of the fitness of the organism. The second explanation is that the given feature changes according to some kind of dynamics that is out of the scope of our analysis. For example if our organism was chosen to be a species then some features may vary in accordance with evolutionary game theory [13]. A third explanation can be that our assumption P_3 about the constant environment is false.

In the case of *convergence* we say that the feature is *optimal*. There are lot of error possibilities however. The most important is that an irrelevant feature may converge as a result of genetic drift. Note that on the other hand it is impossible that relevant features show random behavior.

The situation without fossile record is more interesting since we do not have any ground to classify the features. The question is very sensitive since one has to decide which features are relevant and among the relevant features which ones are optimal. The question is also important since a lot of interesting features of earlier generations such as behavioral patterns or brain structure disappear almost completely. This is the point where adaptational stance comes in with the optimality principle.

P_7 **(optimality):** In the absence of evidence for the contrary it will be assumed that every relevant feature is optimal. This is nothing else than a method which is suggested as a replacement for coin tossing. Its power lies in the fact that in real-life cases optimal features are believed to be in majority. Another problem remains however: how do we decide which feature is relevant and which is not. As Dennett says, the sum of the number of eyes and the number of legs does not seem to be a relevant feature, but not much more is said about this issue. In Section 3 a detailed discussion of this question is given.

1.3 DEPENDENCY

As it has been shown the relevant and irrelevant features can be separated without referring to their functional role in the organism. So far the organism under investigation was handled as a *black box* i.e. we have not examined the causal relationship between the features. In other words *structure* has to be given to the set $\mathcal{G}$ in order to give an explanation and a complete description of the organism.

The structure of $\mathcal{G}$ will have the form of dependency relations. The concept of dependency has been mentioned already in Section 1.1. To discuss reverse engineering, this notion has to be made more precise. We will not give a formal definition but will try to make this term as clear as possible. If g_1 and g_2 are features than we say that g_2 depends on g_1 (and denote this relationship by $g_1 \to g_2$) if for any organism $o \in O$ it is possible to predict $g_2(o)$ from the value of $g_1(o)$ with some accuracy greater than zero. We will call this accuracy the *importance* or *weight* of the given dependency. Note that it is not necessarily possible to do such a prediction in the other direction; this dependency works like a (fuzzy) implication operator on predicates of the form "g is known".

In some cases more difficult relationships can be described. Computing a feature g may depend on values of more than one other features $g_1, \ldots, g_n$. Again, we define this relationship as a fuzzy formula on predicates like "g is known" using conjunction and implication and denote it by $g_1 \wedge \ldots \wedge g_n \to g$. If there are no other relationships of other kinds then we say that the database containing the dependencies is in Horn normal form or simply we have a Horn dependency database.

We are not interested in the details of the actual realization of the dependency of the features, i.e. the underlying physical, biological or any kind of laws. Only the pure statistical fact of the relationship is important. We are not interested in the nature or the effective procedure of computing the features

[3] Note that it is not the set of all possible features; it contains the actual features that are used (or will be used) by the biologists. It is awkward to emphasize that this set is finite since it is trivial. It is a habit however that some people might miss.

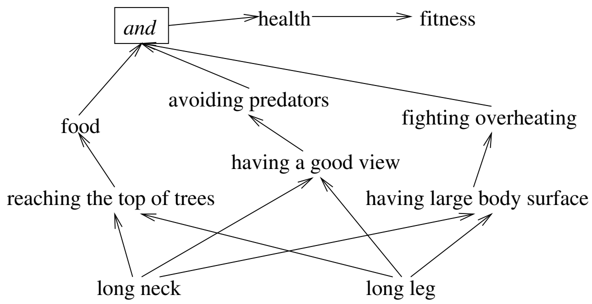

Fig. 1. A small and ad hoc subset of the feature-dependency network of a giraffe for illustration only.

either; we only assume that such a procedure exists. At the abstraction level of our model, these are all unimportant.

It is enough to give the dependencies between the features to give an explanation of the fitness of the organism since mating ability itself is a feature, and the fitness can be directly derived from the mating ability of our organism and the competing members of the population (that are part of the environment) and some other factors of the environment like predators. Since we have accepted P_3, it can be said that fitness is a feature itself i.e. $f \in \mathcal{G}$. The functional role of a feature is nothing else but the way fitness depends on it. Irrelevant features have no functional role at all because fitness does not depend on them (this is the definition of irrelevance) though they can (and probably will) depend on other, maybe relevant, features.

An example of such a dependency network for the giraffe is shown in figure 1. The *network* structure of the dependencies must be emphasized. Taking a look at the diagram, it is hard to accept that for example the functional role of the long neck of the giraffe is to reach the leaves on the top of the trees. It is one of its functions but it may have several other roles as well. In biological organisms it is typical for the components to have several different functional roles; good examples are hormones, vitamins, muscles (motion and heat generation) and so on. Of course it is necessary to weight the dependencies according to importance. There are more important and less important roles but this does not change the fact that fitness depends on each role so they must be part of the explanation of the structure of the organism.

From a historical point of view it is possible that the effect of a less important feature results in a diminishing selective pressure related to a more important feature and can converge only *after* the more important feature has converged. But this historical perspective is completely irrelevant when discussing the dependency relations in the case of an organism. This argument about the independence of syncronic and diachronic analysis is not new: in connection with the science of language Saussure [3] represents the same view which is now widely accepted.

1.4 REVERSE ENGINEERING

Roughly speaking, when trying to understand organisms via reverse engineering we try to reconstruct the dependency relations between the relevant features. The first difficulty is to find the relevant features. This problem will be discussed in detail later in many places from many directions. The second problem is that our dependency diagram is too simple; a classification of features along an additional direction needs to be introduced.

This dimension is approximately the level of abstraction. To understand this let us consider the typical way engineers solve a problem. In the first step they are given a functional description of the system to be designed. This includes its main purpose which is a high number of offspring in the case of biological organisms. This first specification is nothing else but requirements w.r.t. a set of high level, abstract features. The engineer then tries to reduce these features (or goals) into subgoals iteratively using more and more specific features while the design reaches a state where every subgoal can be implemented. This is the less abstract, physical state.

The actual procedure of finding a good design does not necessarily follow the above iteration; case based reasoning may play a major role for example but the result has the structure of features and the dependencies between the features. Some of the features represent low level physical properties some of them represent the highest level functional properties and there are features at intermediate levels of abstraction. Dennett distinguishes three levels: the physical, the organizational and the intentional levels [5]; we will need only the notion of physical features and the notion of top goal(s) which is essentially the first specification the engineer starts with. There are no clean boundaries, though: there is a practically continuous spectrum of abstraction along the implicational chain.

Now reverse engineering can be defined more clearly. If normal engineering proceeds from the top goals to the physical features then reverse engineering goes in the other direction: from the observable physical features it reconstructs the top and intermediate goals. There is one more kind of reverse engineering. In this case the to goals are also known, only the intermediate features have to be discovered. Good examples of the first kind are the mysterious tools of ancient cultures found by archaeologists. Here scientists have to find out the purpose of these tools and in most of the cases all they know is the actual physical properties of the objects.

An interesting question is that in the case of the problem of explaining physical features of animals such as long neck or big ears which kind of reverse engineering is involved. The top goal of different organisms is the subject of debate, for example what is the top goal of an ant? However in certain cases there are available top goals (which are actually subgoals in the strict sense) e.g. flying. Determining the dependency relationships between the physical features of birds and their observable flying ability is an example.

Optimality and dependency Dependency is a property of features as functions while optimality is a property of a particular feature *value*. However while reverse engineering an organism we usually have to rely only on particular feature values when we determine the dependency relations. The case when the optimality or irrelevance of features is known is the simpler. Here we ignore irrelevant features and try to guess (extrapolate) the optimal value of suboptimal features so we can rely on optimal relevant values. When trying to find the function of a physical feature we assume that this function is such that the physical feature is a good implementation of it. For example if we observe strong wings we assume that their function is flying. Note that without the optimality condition it would be possible that the wing is simply irrelevant and not used for anything or suboptimal and used for digging for example or even harmful (as the legs of snakes). When optimality is not known the optimality assumption P_7 has to be used (see Section 1.2) according to the adaptational stance.

2 ADAPTATION IN EC

First let us take a look at the notions defined in Section 1.1. The easiest is P_2 which requires that the constraints of the possible organisms should be taken into account. In EC this is not a problem since one of the first steps of any application is the exact definition of the search space i.e. O. The particular methods of constraint handling and coding is not relevant here only the fact that the problem is handled properly.

P_1 and P_3 are trickier but still easier than in the case of biological evolution. In EC it is typical to have a population of solutions in every time step (generation) and usually an objective function is defined over the possible solutions. The expected number of offspring (i.e. fitness) of the members of the population is given by the objective function values of the other members of the population thereby fitness depends on the objective function *and* the actual population. Therefore we face the same problem of determining the boundaries of the organism and keeping the environment constant.

The later goal is easier since the environment outside the population is constant in most of the applications though nowadays the applications in dynamic environments are becoming more and more important. This means that the earlier goal reduces to deciding whether the individual or the population should be the organism to study. The later choice seems more reasonable though the arguments in connection with P_6 given in Section 1.1 apply here as well. Anyway, the usefulness of having large populations is not proven; there are cases where one-element populations perform best. A typical example is [6].

The notion of the fitness function f is also cleaner in EC. It is very interesting that in EC there is a tradition of calling the objective function the fitness function. At first sight this results in a dissonance between biological and computational terminology but in the light of the restrictions expressed by P_3 we saw that even in biology fitness is taken as a kind of objective function; adaptationists emphasize the objective nature of fitness. In EC to be an adaptationist all we have to declare that the good old tradition should be continued.

$\mathcal{G}$ also has traditions in EC especially in genetic algorithms (GAs). In GAs the solutions need to be encoded so that genetic operators which are defined problem independently could be applied to them. This encoding is analogous to the DNA sequence in which mutation is very similar in the case of every living organism. The views expressed in this question usually take the form of problems of encoding which is essentially nothing else but defining some atomic elements of $\mathcal{G}$ (see e.g. [15]). These are the physical features as introduced in Section 1.4. The other features called building blocks are some simple combinations of these primitives [7]. Though these building blocks can be regarded as functional properties this approach has a number of well known difficulties and limitations as will be discussed in Section 3.

Principle P_4 which requires that only a single niche should be studied is maybe the most problematic. There is a significant amount of work in the field of niche and species formation in the field of EC, [11] and [4] are two examples. The common problem is that all these methods operate with a distance measure defined over O and this distance measure typically depends on the atomic elements of $\mathcal{G}$. This makes the whole procedure ad hoc in the sense that the actual encoding of a given problem is not necessarily optimal. In fact it is always possible to find a distance measure such that fitness has only one optimum i.e. it is unimodal. For example the difference between the fitnesses is such a distance measure. This makes $\mathcal{G}$ even more interesting.

Finally P_5 and P_6 has to be mentioned. P_5 usually holds in EC if the applied selection mechanism is elitist which means that the best member of every generation will be the member of the next generation. The elitist strategy usually performs quite well so it is typically applied. P_6 can be interpreted similarly as in the case of biological evolution.

It should be clear by now that there are no basic incompatibilities between EC and the adaptationist stance. All that remains is to take a look at the possible applications of the *methods* of the adaptationist stance such as optimality analysis and reverse engineering.

2.1 CHARACTERIZATION OF FEATURES

In this area EC has a major advantage namely that it is possible to perform virtually any number of experiments and thereby collecting as many "fossils" as necessary. It is possible to predict with a much greater degree of confidence if a feature is optimal or not by performing statistical experiments since for optimal features convergence to the same value should occur in the majority of the experiments. Of course premature convergence and other well known effects can alter the results. The most difficult problem is to ensure that the algorithm should stay in the same niche every time the experiment is run.

There is another advantage: a hypothesis about the characterization of a feature can be tested experimentally. For example if a feature is suggested to be irrelevant it is easier to vary its value while leaving the other features unchanged and calculate the fitness of the resulting solutions.

2.2 REVERSE ENGINEERING

The dependency relations between features have the same status as their optimality since they were also defined in statistical terms. Any number of experiments can be performed to test and refine the hypothesis. In spite of the fact that there are much greater possibilities in testing the dependency relations between any kind of features (which are the result of reverse engineering) there is very little work in the literature that would describe such reverse engineering results. The reason probably is that the success of reverse engineering which usually results in a deeper understanding of the engineering problem at hand and so also faster and better algorithms than plain EC algorithms is usually considered a *failure* of EC for some reason. People seem to forget that the way the faster and better algorithm are developed is largely dependent on the performance of EC algorithms; in fact it is impossible without them.

In this paper it is attempted to show that it may be useful to at least try to understand the outcome of the optimization process, the solutions suggested by EC algorithms since these solutions form a good starting point to a reverse engineering process that makes it possible to translate the information accumulated by evolution into engineering knowledge. If this process results in better heuristics then *it is the success of EC* and the engineers of course.

A simple example of the successful applications of reverse engineering is [10]. There are more sophisticated examples as well such as network design [1]. Unfortunately we cannot discuss these due to the lack of space.

3 REPRESENTATIONAL BOTTLENECK

The language of biology is particularly rich and in a way it is very close to natural language. The terms (features) used to describe organisms such as color, shape, organs and body-parts like heart, lungs and arms and behavioral patterns like aggression are typically understandable by anyone. Biology inherited a large, detailed terminology of describing the living world. This may be a result of our own evolution and the evolution of our culture; animals and plants have been around us since language and culture emerged and have been playing a crucial role in our lives as food, enemy, building material and so on ever since. Our sensory and cognitive system is likely to be specialized in describing living organisms, among other things.

This makes the job of finding problems in biology relatively easy since the features to be explained *are there*. The situation is radically different in EC. The description of solutions lacks even the simplest terms and usually reduced to the encoding of the solution. This means that we talk only about *genes* as if biologists could describe a monkey only with the help of its DNA sequence. The insufficiency of this description may be evident to some of the readers but actually in EC the practice is accepting that the encoding provides us with a language sufficient for describing the solutions. This is motivated by the need of developing a domain independent theory of EC, and this need may be originated from the analogy with other optimization methods like the method of steepest descent or the different Newtonian iterations. The problem is that the application area of EC is *much* larger than any of these restricted methods. This is why domain specific information plays a more important role.

The elephant picture function When it comes to reverse engineering and giving an explanation of the solutions developed we need to find features to create a model of the particular problem class. To see this let us give an example: the elephant picture function. In this problem the organisms are two-dimensional bitmaps where the physical features are the pixels of the picture. The fitness of a given picture is the degree of resemblance to an elephant; any kind of elephant in any position as illustrated by Figure 2.

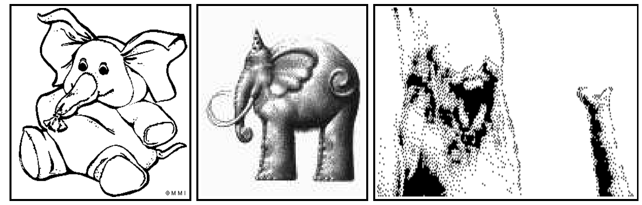

Fig. 2. Examples of bitmaps with a good fitness.

The fitness is independent from a particular pixel since the degree of resemblance of a negative picture is the same as of the original. The elephant picture function does depend on lower-level features such as lines, curves and similar basic components but these are still too abstract to depend on particular pixels; we need another abstraction level closer to the physical features. To convince those who are still in doubt imagine that the positions of the pixels of the picture are mixed by a *deterministic* and *invertible* algorithm so the elephant cannot be recognized anymore by looking at the picture (see Figure 3). Let the fitness function be the original

Fig. 3. An elephant picture and one of its permutations.

elephant picture function applied to the result of the inverse of the above transformation on the given bitmap. The most predictive high-level feature is still resemblance to elephants *before* mixing the pixels. If one cannot "decode" the image then it is almost impossible to predict fitness.

On the other hand an EC algorithm could probably find a good solution of the elephant picture problem provided that somehow this function was available. Interactive applications i.e. applications that use the human user as a fitness function indeed exist; one example is the iterative evolution of textures where the fitness is a kind of artistic or aesthetic value [9].

Finally let us note that the features used in biology to describe organisms are practically independent of the level of DNA as well since it would be extremely hard to predict the shape or behavior of an *unknown* animal from its DNA only. Genes connected to higher level features are often described and known species with similar DNA can be used but this process proceeds in a top-down fashion: the features are described first and then the corresponding genes are looked for.

Similar problems Self-organizing maps (SOMs) are devices that are capable of finding structure or a clustering in their input data-set. One good example is Kohonen's phoneme recognizer [12]. SOMs are usually based on the physical features of these vectors only so in many cases it has serious problems finding clusters that are similar in some important sense but are not in the physical level (e.g. elephant pictures). In the case of instance based learning and case based reasoning (see e.g. [14]) it is well known that one of the main problems is the selection of the distance measure. To find a good distance measure, one has to *understand* the problem domain to be able to tell the difference between important and irrelevant features or at least to find features to start with. Just like in SOMs for the interesting problems it is not sufficient to rely only on the physical, lowest level features like pixels in an image or elements of a vector as usually done.

The spaceship picture function To illustrate that the elephant picture function is indeed a very serious problem let us give another example: the spaceship picture function. The situation is like with the elephant pictures but the difference is that e.g. 500 years ago nobody had an idea about spaceships. The interesting thing is that the function did exist nevertheless though no one could predict its values. Engineers of that time would have been in great trouble when trying to understand a spaceship function computing machine. Note that no low-level "decoding" would have helped them in this case as we have seen in Section 3. The problem is that spaceships can be very different and some of them are very similar to aircrafts, saucers etc.

What the engineers lacked 500 years ago is the knowledge about spaceships, space, artificial flying, science fiction movies etc. To be short the term spaceship was not part of their language. I assume that there are more unknown than known terms: this is what I call the representational bottleneck. The situation in the abstract domains such as flow control or combinatorial problems that are likely to have a rich and complex yet undiscovered structure is even worse.

4 CONCLUSIONS

It was demonstrated that the adaptational stance and reverse engineering is strongly connected. Adaptational stance is a strategy of reverse engineering that might fail in certain kinds of circumstances but it seems to be useful in many cases. It was shown that reverse engineering is applicable and in fact it has been applied in EC. The bottleneck of this method however is the language that is available for describing the abstract problem domains. As we have seen, biology has a much larger and effective vocabulary partly inherited from natural language though this vocabulary is constantly growing as other sciences develop so adaptationist explanations in biology also have the representational bottleneck.

It is also important to emphasize that the performance of an EC algorithm may well be very good even if reverse engineering is not successful and domain specific knowledge is not available. This is the main power of evolution: it only works with the physical features and does not care about any explanations because explanations or dependency models of features of different abstraction levels are simply tools for us human beings to handle predictions with our limited cognitive capacity. It is very much like the relationship between axioms and theorems in mathematics. Axioms have the status of physical features; knowing the axioms means knowing everything[4]. Theorems are needed only because it exceeds our abilities to tell the truth value of a formula directly from the axioms. This is why the representational bottleneck is a problem though the physical features (encoding of a solution in EC) are known; explanations need higher level features because of our cognitive limits.

An EC algorithm does not provide *final* solutions to problems; it is a ledder leading to an understanding of the problem domain making reverse engineering possible and thus leading to faster and better specific algorithms. The representational bottleneck problem should be solved by scientific research in each domain. Since experiments can be repeated any time with any settings this research is much easier than in biology though the vocabulary to start with may be poor. The contribution of EC to the new algorithms is essential since without known good solutions reverse engineering is impossible therefore the success of such results is definitely a success of EC as well.

Acknowledgements

This work was supported by FKFP 1354/97.

References

1. L. A. Cox, Jr, L. Davis, L. L. Lu, D. Orvosh, X. Sun, and D. Sirovica. Reducing the costs of backhaul networks for PCS companies using genetic algorithms. *Journal of Heuristics*, 2(1):1–16, 1996.
2. R. Dawkins. *The Selfish Gene*. Oxford University Press, Oxford, 2nd edition, 1989.
3. F. de Saussure. *Cours de linguistique générale*. Payot, Paris, 1939.
4. K. Deb and D. E. Goldberg. An investegation of niche and species formation in genetic function optimization. In J. D. Schaffer, editor, *The Proceedings of the Third International Conference on Genetic Algorithms*. Morgan Kaufmann, 1989.
5. D. C. Dennett. *The Intentional Stance*. MIT Press, 1990.
6. A. E. Eiben and J. K. van der Hauw. Graph coloring with adaptive genetic algorithms. *Journal of Heuristics*, 4(1), 1998.
7. D. E. Goldberg. *Genetic algorithms in search, optimization and machine learning*. Addison-Wesley, 1989.
8. S. J. Gould and R. Lewontin. The spandrels of San Marco and the Panglossian paradigm: A critique of the adaptationist programme. In *Proceedings of the Royal Society*, volume B205, pages 581–598, 1979.
9. A. Ibrahim. *GenShade: An Evolutionary Approach to Automatic and Interactive Procedural Texture Generation*. PhD thesis, Texas A&M University, May 1998.
10. M. Jelasity. A wave analysis of the subset sum problem. In T. Bäck, editor, *Proceedings of the Seventh International Conference on Genetic Algorithms*, pages 89–96, San Francisco, California, 1997. Morgan Kaufmann.
11. M. Jelasity and J. Dombi. GAS, a concept on modeling species in genetic algorithms. *Artificial Intelligence*, 99(1):1–19, 1998.
12. T. Kohonen. The "neural" phonetic typewriter. *IEEE Computer*, pages 11–22, March 1988.
13. J. Maynard Smith. *Evolution and the Theory of Games*. Cambridge University Press, Cambridge, 1982.
14. T. M. Mitchell. *Machine Learning*. McGraw-Hill, 1997.
15. N. J. Radcliff. The algebra of genetic algorithms. *Annals of Maths and Artificial Intelligence*, 1994.
16. D. S. Wilson and E. Sober. Reintroducing group selection to the human behavioral sciences. *Behavioral and Brain Sciences*, 17(4):585–654, 1994.

[4] It is not always true because of Gödel's theorem but practically this is the case in traditional maths. Theorems that may be true or false in different models are very hard to find.