Investigating the Emergence of Phenotypic Plasticity in Evolving Digital Organisms

Jeff Clune, Charles Ofria, and Robert T. Pennock

Digital Evolution Lab, Michigan State University, 3114 Engineering Building, East Lansing, MI, USA
{jclune, ofria, pennock5}@msu.edu

Abstract. In the natural world, individual organisms can adapt as their environment changes. In most *in silico* evolution, however, individual organisms tend to consist of rigid solutions, with all adaptation occurring at the population level. If we are to use artificial evolving systems as a tool in understanding biology or in engineering robust and intelligent systems, however, they should be able to generate solutions with fitness-enhancing phenotypic plasticity. Here we use Avida, an established digital evolution system, to investigate the selective pressures that produce phenotypic plasticity. We witness two different types of fitness-enhancing plasticity evolve: *static-execution-flow* plasticity, in which the same sequence of actions produces different results depending on the environment, and *dynamic-execution-flow* plasticity, where organisms choose their actions based on their environment. We demonstrate that the type of plasticity that evolves depends on the environmental challenge the population faces. Finally, we compare our results to similar ones found in vastly different systems, which suggest that this phenomenon is a general feature of evolution.

1 Introduction

The field of evolutionary computation uses natural selection to automatically find solutions to engineering problems [1, 2]. Frequently, these solutions are on par or better than any human-produced solution [2]. However, most of these cases have involved static solutions to static challenges [3]. If the challenge suddenly changed even slightly, most organisms would not be able to adapt without further evolution. In other words, these organisms exhibit little or no within-life, or 'intralife', adaptation. Frequently, however, natural organisms or engineered solutions need to be robust enough to handle noisy and dynamic environments. One way of doing this is to evolve organisms in environments where the challenges they are presented with vary over time. Certainly, if our long-term goal is to evolve truly intelligent systems, we need to better understand how evolution can produce things that can intelligently adapt, initially in simple and then in increasingly sophisticated ways, to changing and novel situations.

F. Almeida e Costa et al. (Eds.): ECAL 2007, LNAI 4648, pp. 74 – 83, 2007.

2 Previous Work

One tactic researchers have used to evolve individuals that exhibit within-lifetime adaptation is to provide natural selection with a hand-written learning module, such as back propagation for neural networks. The researchers then investigate how evolution takes advantage of this ability to learn [4-8] and whether the results of learning are transferred to the genome via the Baldwin effect [9, 10]. Such research forces natural selection to use human-designed learning mechanisms, however, as opposed to discovering its own. A different approach—and the one taken here—is to use very simple systems to investigate how the ability to adapt could evolve on its own.

One clear sign that an organism is capable of adapting is if it behaves differently in two different environments. This kind of adaptive capacity is called *phenotypic plasticity*. Investigating this characteristic, Nolfi et al. evolved neural network brains for virtual robots in alternating light and dark environments [11]. The environments were constructed such that bodies, brains and behavior tuned to one environment would perform poorly in the other. They tested the evolved solutions by placing the same genome in the two different environments and observed whether they behaved differently. They found that in populations that evolved in alternately light and dark environments, the individuals were tuned to whichever environment they were placed in. Stanley et al. also used evolving neural networks to look at the evolution of phenotypic plasticity [3]. They compared experiments where the connection weights of neural networks never changed during the course of their life to those where evolution could create and modify connection weights during a lifetime. Such changes could be used to modify behavior in response to information sensed from the environment. The individuals were evolved in environments where each type of food randomly switched between nutritious and poisonous. Contrary to their expectations, they found that the individuals in the fixed-connection weight treatment discovered a simple solution that worked in all environments. The individuals executed the same code, but the inputs into that code were different because of the environmental differences, resulting in a different in behavior that was adapted to each environment. They thus possessed phenotypic plasticity in spite of a static execution flow. On the other hand, in the modifiable connection weight treatment the individuals developed phenotypic plasticity using a dynamic execution flow. In both cases the phenotypic plasticity enabled them to behave differently in the different environments. Ironically, the 'simple trick' evolution discovered in the fixed-connection weight experiment was more effective than the dynamic execution flow based strategy it discovered in modifiable connection weight treatments.

In short, we identify two types of phenotypic plasticity: static-execution-flow plasticity and dynamic-execution-flow plasticity. Which type prevails under natural selection depends on many evolutionary factors, such as the environmental challenge and or the genomic representation. Much of the research into evolving phenotypic plasticity *in silico* has been done with neural networks controlling simulated robots. But how general are the results? Can similar patterns be found in different systems? Here, we investigate the evolution of phenotypic plasticity in a vastly different setup: Avida, an experimental digital evolution system that maintains populations of self-replicating and evolving computer programs with a simplified (but Turing complete) assembly language as its genetic basis. These "digital organisms" have been shown to evolve

traits of significant complexity [18], but previous work has focused on constant environments and hence rigid solutions. We find that our preliminary results are surprisingly parallel to those from the neural net community and suggest fundamental biological principles that can be applied toward understanding the evolution of intelligence.

3 Methods

Avida is an established experimental evolution platform where digital organisms compete for space in a two-dimensional grid [12-17]. There is no explicit fitness function in Avida; instead, organisms compete for limited space and those that replicate the fastest are most successful. Organisms must copy their own genomes and then execute a divide instruction, to produce an offspring. The copy process is imperfect, however, which introduces the variation that fuels natural selection. In the experiments performed here, the copying of a given instruction occasionally results in a copy error (0.75% of the time). When this happens, an instruction is chosen at random from the available set (N=26) and written to the target location (see [13] for details on the instruction set). There are also insertion and deletion mutations, which introduce or delete an instruction at random in 5% of offspring. All mutations affect only the genome of the offspring. The population size is 3,600. While the genome size can vary, the ancestral organism is 100 instructions long. The ancestor starts with the ability to self-replicate, but is largely blank with 85 of its instructions set to a mostly neutral 'no operation' command.

Digital organisms can improve their speed of reproduction either by decreasing the number of instructions it takes to produce an offspring or by performing tasks that increase their metabolic rate (rate of executing instructions). The initial metabolic rate for any organism is approximately proportional to its genome length (see [12] for a more detailed explanation). This number is then doubled or halved each time a task is performed, depending on whether the task is rewarded or punished. The tasks in this experiment are the logic functions NOT and NAND. Each organism can input three 32-bit numbers. They can manipulate those numbers and output the result. If they output the logical bitwise negation of one of the numbers or the bitwise nand of any two, they have performed the NOT or NAND task, respectively. The manipulation of these numbers occurs as organisms push and pop them to stacks or move them between registers using instructions such as `push`, `pop`, `add` (combines the numbers in the two specified registers and places the result in a third), `shift-r` (bit shift right), etc.

In order to adapt to their environment, organisms need to be able to sense it. For these experiments, the typical Avida `IO` instruction, which simultaneously inputs and outputs a number, was changed to `IO-Feedback`. This new instruction is identical to `IO` except that it provides the organism with knowledge of the impact the output had on their metabolic rate. When `IO-Feedback` is executed, if the organism's metabolic rate increased (because it performed a task currently being rewarded), a 1 is placed on the top of its active stack. If its metabolic rate is diminished (because it performed a task being punished), a -1 is placed on the top of its active stack. If the output had no impact on its metabolic rate (because it was not the negation of one of

the three numbers or the nand of two of them), a 0 is placed on the top of its active stack. Organisms have flow-control instructions available that allow them to jump to, or skip over, sections of code in their genomes. An organism could, for example, execute a set of instructions that perform NOT and then repeat those instructions if the number atop their active stack is 1. The instruction set used here is Turing complete, meaning that it can perform any computable function. Therefore, any sophisticated conditional execution flow should in principle be able to evolve; the only remaining questions are whether natural selection discovers such complexity and, if it does, whether it will cost too much to be advantageous to the organism [12, 18].

4 Experiments and Results

To challenge the organisms to evolve plasticity, the population is alternately exposed to two different environments. In the Not+Nand- environment, performing NOT doubles an organism's metabolic rate and performing NAND halves it (the '+' indicates reward, the '-' indicates punishment). In the Not-Nand+ environment, the reverse is true. The environment shifts every 100 updates. Updates are the standard unit of time in Avida where each organism, on average, executes 30 instructions. Since in this experimental setup it normally takes organisms around 300 instructions to copy themselves, switching environments every 100 updates is equivalent to switching it approximately every ten generations. Because each trial lasts for 100,000 updates, there are 500 full cycles through the two alternating environments.

4.1 Experiment One

Our first experiment uses this setup to investigate whether the digital organisms will evolve phenotypic plasticity when their environment is uncertain. In this experiment, there are no restrictions on how often an organism can perform a task, so an organism's metabolic rate can be doubled or halved any number of times. In 13 of 50 trials, the final dominant organism alters the number of tasks it performs depending on which environment it is in. In 8 of these 13 trials, the flexibility results in a net positive reward in both environments. In the other 4 trials the plasticity is used to decrease the number of punished tasks being performed, but the punishments still exceed the rewards in one environment.

We next determined how these organisms are able to adapt to their environment. The same strategy is used in all 8 trials where the final dominant organism has a positive score in both environments. In one sense, these organisms are indeed 'adapting' because they end up performing different tasks in different environments. In another sense, however, they are not adapting at all; they execute the exact same series of instructions irrespective of what environment they are in. The organisms make no use of instructions that would make their execution flow conditional. Instead, they have discovered a string of instructions that results in behaviors that are tuned to the environment they find themselves in. Table 1 reports on which tasks are performed by the final dominant organisms from two example trials from experiment 1. The first organism does not adapt to its environment while the second does.

Table 1. The result of running the final dominant organism from two example trials from experiment 1 in two different environments. During evolution the organisms alternately encountered these two environments. The first organism always performs the same task set, which is beneficial in one environment and deleterious in the other. The second organism is able to adapt to the environment. It performs X NANDs and either 0 or >X NOTs, ensuring a net positive bonus. This same basic technique is used by the final dominant organism in all 8 trials in experiment 1 where the evolved plasticity resulted in positive fitness scores in both environments. Interestingly, the technique produces adaptation despite executing the same series of instructions. See text and Fig. 1 for an explanation of how.

Not+Nand- Environment			Not-Nand+ Environment			Static Execution Flow Plasticity?	Dynamic Execution Flow Plasticity?
NOTs	NANDs	Score	NOTs	NANDs	Score		
2	0	2	2	0	-2	No	No
102	51	51	0	51	51	Yes	No

The way the organisms are able to produce different numbers of tasks in different environments with the same series of executed instructions is simple and clever. The strategy involves putting different inputs into the same function to get different results. Using the information sensed from the environment as an input into a function, the resultant behavior can be modified based on the environment. The eight final dominant organisms that end up with positive rewards in both environments use simple variations on the theme shown in Figure 1.

This result emphasizes that evolution sometimes selects simple but effective solutions over complex, elegant ones. A sophisticated manipulation of execution flow was not needed to produce genomes that are adapted to both environments. However, many forms of sophisticated intelligence will require the ability to dynamically change the way that actions are determined. As such, it is worthwhile to determine the conditions under which dynamic execution flow phenotypic plasticity evolves (i.e. organisms that execute different programs in response to varying environmental conditions). To do so, we must understand how a static solution can be more fit than a dynamic solution. Why do organisms always perform a large number of NANDs? Why don't they regulate the performance of both tasks instead of just regulating NOT? It could be because it is easy to outcompete the current dominant static strategy by making a slightly better static strategy. If there is a population of organisms that do 0 or 20 NOTs and 10 NANDS, for a net reward of 10 in either environment, it is not uncommon to mutate to perform 0 or 22 NOTs and 11 NANDs for a net reward of 11. A mutation simply needs to make the organism run this loop one more time. Compare that to the challenge of setting up the necessary instructions to do the following: "if x, do instructions A,B,C; else, do D,E,F." This hypothesis motivates our second experiment.

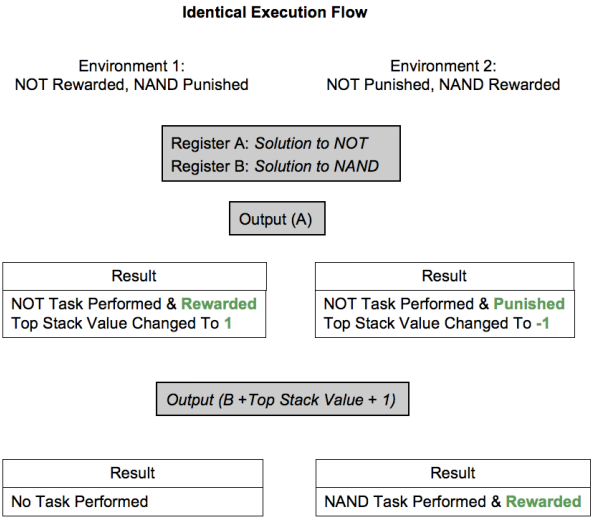

Fig. 1. The same series of instructions in two different environments leads to different results. Once the environment produces a difference in the stacks of the organisms, they can use it as an input to a function and produce two different results. If NOT is being rewarded, NAND is being punished. In environment 1, taking the correct answer for NAND and adding 2 to it produces a number that is no longer a correct answer for the NAND task (the 2 comes from 1 + the stack value of 1). Outputting this number does not result in either a reward or a punishment. In environment 2, adding 0 to the correct answer for NAND leaves it unchanged, and outputting this number yields the reward (the 0 comes from 1 + the stack value of -1).

4.2 Experiment Two

In our second experiment we cap the number of tasks for which an organism can be rewarded or punished to 10. Because in the previous experiment it was easy to extend the static strategy indefinitely, organisms did not need to use dynamic execution flow to regulate both of their tasks. Instead, they only regulated the NOT task (see Table 1). With a cap of 10, every punished task takes away from a potential rewarded task. The only path to the maximal fitnesses is through regulation of both tasks. Aside from this cap, experiment 2 is identical to experiment 1.

The results in this setup are quite different from experiment 1. In experiment 2, the final dominant organism in 23 of 50 trials alters its task output based on which environment it is in. In 15 of these 23 trials, the final dominant organism achieves a net positive score in both environments. All but one of these 15 organisms employs dynamic execution flow (in contrast with 0 final dominant organisms using dynamic execution flow in experiment 1). The average replication speed (fitness) across all 50 trials is shown in Fig. 2a. An individual trial where plasticity evolved is shown in Fig. 2b. The breakdown of what tasks it performs in each environment is presented in Table 2.

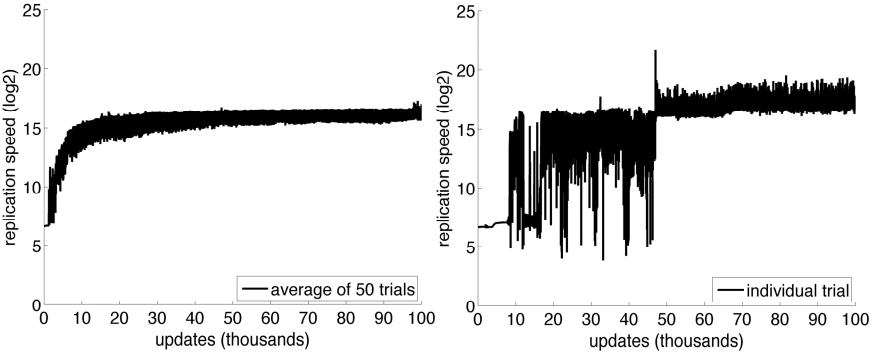

Fig. 2. The replication speed (fitness) for *(A)* the average of the 50 trials in experiment 2 in which the maximum number of times an organism can do a task (whether rewarded or punished) is set to 10, and *(B)* an example trial from this experiment where organisms have evolved to have high fitnesses across both environments (something that in this trial occurs about halfway through the experiment).

Table 2. The example organism from Fig. 2b. demonstrates a perfect ability to adapt to the two environments it is faced with. In environment 1, it gets rewarded 10 times for NOT (the maximum). In environment 2, it gets rewarded 9 times for NAND and punished once for NOT. It is not possible for an organism to perform better, as it must perform its first task without knowledge of which environment it is in. Thus, the first task performed will inevitably be punished in one of the two environments.

Not+Nand- Environment			Not-Nand+ Environment			Static Execution Flow Plasticity?	Dynamic Execution Flow Plasticity?
NOTs	NANDs	Score	NOTs	NANDs	Score		
16	0	10	1	32	9	Yes	Yes

The example organism from Table 2 (the final dominant from the experiment shown in Fig. 2b) attains the best score possible across the two environments. In Not+Nand- it ends up with 10 rewards (the max), and in Not-Nand+ it ends up with 9 rewards and 1 punishment. Since an organism must perform a NOT or NAND in order to determine which environment it is in, the best it can do is suffer just one poison in one of the two environments, as this organism does. Note: while all tasks performed above the cap of 10 (e.g 6 of the 16 NOTs) do not benefit the organism, performing them need not involve much or any extra cost for the organism. Evolution frequently produces designs that are 'good enough' instead of perfect. [19-24]

A question remains as to whether this organism's ability to adapt is derived from execution-flow plasticity. Is it changing which instructions it executes based on information from the environment? The answer in this case is yes. Fig. 3 presents a graphic representation of the instructions executed in the different environments by the final dominant organism from this case-study trial.

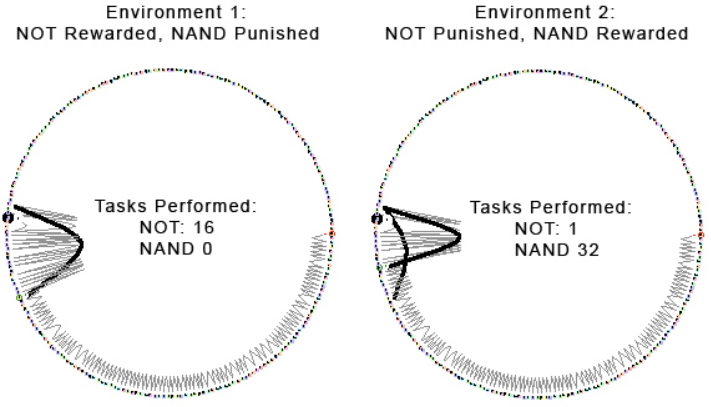

Fig. 3. A graphical representation of the instructions executed by the final dominant genome from the trial described in Fig. 2b and Table 2 when placed in environments 1 & 2. The small circles that make up the circumference represent each instruction in the genome of the organism. Arcs start on an executed instruction and land on the next one to be executed. Thin gray arcs indicate clockwise jumps and thick black arcs indicate counter-clockwise (backward) jumps. The height of the arc corresponds to the number of times that arc was traversed. Instructions that are never touched by an arc (as is the case with roughly half of the instructions in this organism) are not executed but do serve regulatory functions since they are traversed during replication. In both environments a series of instructions is executed before entering the looping area where tasks are performed. In environment 1, the organism repeats the same loop over and over, performing a NOT each time. In environment 2, the organism shortens the size of the loop after the first iteration and thus executes fewer instructions per subsequent iteration. This subset of instructions performs a NAND (and only a NAND) each iteration. The trigger to change the size of the loop is based on whether the NOT produced by the first iteration through the loop was rewarded or punished. The organism is thus able to change its behavior in response to the environment. It has evolved dynamic-execution-flow phenotypic plasticity.

5 Discussion and Conclusion

Our experiments demonstrate that natural selection will take advantage of simple static solutions that work across dynamic environments if they are available and advantageous. In our first experiment, a plastic solution using dynamic execution flow would have yielded high fitness values. Instead, selection reached high fitness values via a simpler static execution flow type of plasticity. Stanley et al. challenged static networks with dynamic environments and did not expect selection to discover a solution, but it did [3]. They tried a second experiment in which the network topologies could evolve over time. This simultaneously opened up the possibility of dynamic execution flow and made it difficult for static execution flow solutions to work. (Stanley, personal communication) In this second experiment, dynamic execution flow based plasticity evolved to produce highly fit organisms. Nofli et al. also challenged evolution with environments so different that, seemingly, only dynamic execution flow based strategies could work. Sure enough, they evolved [11]. A conclusion

is suggested: while natural selection will take advantage of simple static solutions to a dynamic environment if they exist, in environments where dynamic solutions are more likely to gain the highest fitness values, natural selection can employ them. The work of Stanley et al. and Nolfi et al. show that this principle holds for evolving neural nets. Our work shows that it holds in populations of evolving digital organisms that execute genomes consisting of a series of instructions.

That we find similar results in such vastly different systems lends credence to the idea that these results describe evolution in general. As we have seen, there are two types of phenotypic plasticity: static-execution-flow plasticity and dynamic-execution-flow plasticity. Whether one or both of them emerge depends on many factors. Here we demonstrated that the environmental challenge is one of those factors. Dynamic execution flow, which seems so powerful that one might expect it to always be advantageous, does not evolve merely because it can. Rather, in a given environment, evolution may opt for a simpler available static solution that will "do the trick," selectively speaking.

Understanding how evolution works is of utility for engineers who want to apply evolutionary methods for practical purposes. If one wishes to evolve a particular type of plasticity, one should do so using environments that make it actively advantageous for natural selection to produce it, rather than simply possible. Future research is needed to learn more about when natural selection results in these different types of plasticity. Hopefully, such knowledge will facilitate our efforts to evolve systems as complex and intelligent as those found in the natural world.

Acknowledgments. The research for this paper was funded by the Cambridge Templeton Consortium, the National Science Foundation, and a fellowship to JC from the Quantitative Biology & Modeling Initiative at Michigan State University. We thank the anonymous reviewers and the members of the Evolving Intelligence and Digital Evolution Labs at Michigan State University, in particular Jeff Barrick, Sherri Goings, Dusan Misevic, Kaben Nanlohy, Brian Baer and Richard Lenski.

References

1. Holland, J.J.: Adaptation in Natural and Artificial Systems. University of Michigan Press, Ann Arbor (1975)
2. Koza, J., Keane, M., Streeter, M., Mydlowec, W., Yu, J., Lanza, G.: Genetic Programming: Routine Human-Competitive Machine Intelligence. Kluwer, New York (2003)
3. Stanley, K.O., Bryant, B.D., Miikkulainen, R.: Evolving Adaptive Neural Networks with and without Adaptive Synapses. In: IEEE Congress on Evolutionary Computation, Canberra, Australia, IEEE Press, Los Alamitos (2003)
4. Nolfi, S., Floreano, D.: Learning and Evolution. Autonomous Robots 7, 89–113 (2004)
5. Ackely, D.E., Littman, M.L.: Interactions between Learning and Evolution. In: Proceedings of the Second Conference on Artificial Life, Addison-Wesley, Reading (1991)
6. Belew, R.K., McInerney, J., Schraudolph, N.N.: Evolving Networks: Using the Genetic Algorithm with Connectionist Learning. CSE Technical Report CS89-174. University of California, San Diego (1990)
7. Whiteson, S., Stone, P.: Evolutionary Function Approximation for Reinforcement Learning. Journal of Machine Learning Research, 877-917 (2006)
8. Nolfi, S.: Learning and Evolution in Neural Networks. Adaptive Behavior 3, 5–28 (1994)

9. Baldwin, J.M.: A New Factor in Evolution. American Naturalist, 441-451 (1896)
10. Hinton, G.E., Nowlan, S.J.: How Learning Can Guide Evolution. Complex Systems, 495-502 (1987)
11. Nolfi, S., Miglino, O., Parisi, D.: Phenotypic Plasticity in Evolving Neural Networks, 146-157 (1994)
12. Ofria, C., Wilke, C.O.: Avida: A Software Platform for Research in Computational Evolutionary Biology. Artificial Life 10, 191–229 (2004)
13. Lenski, R.E., Ofria, C., Collier, T.C., Adami, C.: Genome Complexity, Robustness and Genetic Interactions in Digital Organisms. Nature 400, 661–664 (1999)
14. Ofria, C., Adami, C., Collier, T.C.: Design of Evolvable Computer Languages. IEEE Transactions on Evolutionary Computation, 420-424 (2002)
15. Misevic, D., Ofria, C., Lenski, R.E.: Sexual Reproduction Reshapes the Genetic Architecture of Digital Organisms. Proceedings of the Royal Society London, Series B 273, 457–464 (2006)
16. Adami, C., Ofria, C., Collier, T.C.: Evolution of Biological Complexity. Proceedings of the National Academy of Sciences 97, 4463–4468 (2000)
17. Goings, S., Clune, J., Ofria, C., Pennock, R.T.: Kin-Selection: The Rise and Fall of Kin-Cheaters. In: Proceedings of Artificial Life Nine, pp. 303–308 (2004)
18. Lenski, R.E., Ofria, C., Pennock, R.T., Adami, C.: The Evolutionary Origin of Complex Features. Nature 423, 139–144 (2003)
19. Darwin, C.: On the Various Contrivances by Which British and Foreign Orchids Are Fertilized by Insects. Murray, London (1862)
20. Dawkins, R.: The Selfish Gene. Oxford University Press, Oxford (1976)
21. Dawkins, R.: The Blind Watchmaker. Penguin, London (1986)
22. Gould, S.J.: The Panda's Thumb: More Reflections in Natural History. Norton, New York (1980)
23. Gould, S.J., Lewontin, R.C.: The Spandrels of San Marco and the Panglossian Paradigm: A Critique of the Adaptationist Programme. Proceedings of the Royal Society of London 205, 281–288 (1979)
24. Jacob, F.: Evolution and Tinkering. Science, 1161-1166 (1977)