Weasel words, genetic algorithms and coarse acknowledge-ments

Victor Khomenko

The 'weasel words' example by Richard Dawkins is based on a simple genetic algorithm to demonstrate the likelihood of evolution due to natural selection. The algorithm rewards fine changes in single letters of a random sequence towards obtaining the desired phrase. The complexity of living things, however, is significantly more complex, irreducibly complex in many instances. Thus, fine changes such as the substitution of a single amino acid in a protein chains is unlikely to be rewarded by improved fitness. Improved fitness is only obtained by coarse changes in the overall structure, and often these changes will require moving through less optimal solutions (valleys between the peaks of Mt. Improbable). A more realistic genetic algorithm based on coarse acknowledgments shows that Mt. Improbable is truly impossible.

Introduction and previous work

In The Blind Watchmaker,[1] Richard Dawkins, a well-known proponent of the theory of evolution, proposes the following computer simulation allegedly proving the possibility of evolution. (Though Dawkins does not make such a strong assertion himself and even acknowledges some of the flaws in this simulation, other evolutionists do.) He takes the phrase 'methinks it is like a weasel' from Shakespeare's Hamlet and shows that it can be obtained by applying a sequence of mutations (random little changes) and selections (choosing the strings which better resemble the target phrase) to randomly generated strings of letters. An example of such a process is given in the second column of

Table 1 in the context of *genetic algorithms* described in the next section. Note that for simplicity any special symbols (such as spaces between the words) are disallowed and only strings of lower-case English letters of fixed length 23 are considered; thus the mentioned phrase, called in sequel the *target*, becomes 'methinksitislikeaweasel'.

There have been a number of creationist publications addressing Dawkins' example. They note the duping potential of such computer 'demonstrations':

> 'At the time (1986) it was fairly showy to have a computer program to demonstrate something and many readers were duped into thinking that the program had proved something, not realizing that a program will do whatever its programmer designs it to do.'[2]

The reader with a mathematical background can smell the rat in this alleged demonstration of evolution 'at work'; yet, it turns out to be not that trivial to pinpoint exactly where the flaw is! The following suggestions, among others, have been put forward in previous creationist publications:[2–4]

- **The answer is 'hardwired' into the program.** The target is predetermined, so no novel information arises during the simulation. Dawkins himself acknowledges that this is 'misleading in important ways'.[1] However, this is a plausible way to simulate the environment in a program. Note that the program uses the target in a 'black-box' manner, i.e. the only way for it to access the target is by *making queries of a certain type,* using a fixed *interface* (i.e. the format of queries), as shown in Figure 1. In this particular case, the program repeatedly passes the generated strings to the environment and gets back their 'fitness values' represented by natural numbers. All the models presented in this paper also fit into this framework; thus the main line of argument in this paper is not to criticize the top-level set-up of Dawkins' computer simulation, but to modify some of its low-level details (to make it better reflect the process of natural selection) and show that the problem becomes much harder indeed.
- **Accelerated mutation rate and perfect selection.** Real-world mutation rates are many orders of magnitude less than the one used in Dawkins' model, and the selection coefficient is 1.0 in every generation (i.e. the string closest to the target always survives), whereas in the real world a generous selection coefficient would be 0.01, i.e. its chance to survive would be just slightly better than those of other strings. These arguments, though completely valid, are not strong enough by themselves. Indeed, the evolutionist can claim almost arbitrary mutation rates (though mutation rates that are higher than the inverse of the length of the string would cause an *error catastrophe,* destroying useful information too quickly), motivating this by imperfection of replication mechanisms of primeval organisms, high radiation levels due to a different composition of the Earth's atmosphere and/or the Earth moving through a zone of space with high radiation, etc. Moreover, the mutation rate, as well as the 'perfectness' of selection, has only linear impact on the length of the evolution

process (unless the error catastrophe happens); in other words, if the mutation rate (or the selection coefficient) is reduced by the factor of 1000, the process of evolution would take about 1,000 times longer—the evolutionist operating in billions of years can cope with that. Nevertheless, these two arguments have an important advantage: they are essentially independent from to the other mentioned arguments, and can push up any time estimate derived from such arguments by a few orders of magnitude.

- **Self-reproduction, a pre-requisite.** The model does not explain the origin of the first self-replicating entity, without which the whole process of evolution is a non-starter. Natural selection can operate on populations of such organisms but cannot aid in their formation, i.e. they could not have come into existence by means of gradual improvements of some simpler entity. In other words, extreme luck (which cannot be divided on a sequence of independent smaller 'lucks') must have been involved for it to appear by blind chance unless such an entity was very trivial. However, no simple self-replicating entity has been demonstrated so far, and the theories about what it could look like are at the stage of wild speculations. Strong suspicion arises that no trivial self-replicating entities can exist, and thus the 'molecule-to-man' evolution has a major problem already at the 'molecule' stage. Dawkins' simulation does not attempt to deal with this argument. Rather, its purpose is to show that once self-replicating entities have *somehow* appeared, the process of evolution is easy and smooth.
- **No irreducible complexities.** Only individual letters rather than combinations thereof are taken into account. This does not reflect the process of natural selection adequately, because many useful traits in living organisms are *irreducibly complex,* i.e. they fail completely if even one of their components is missing, which makes them resistant to any stepwise explanation of their origin.[5] Though evolutionists deny that irreducible complexities exist in living organisms, simulations that *a priori* exclude the possibility thereof simply beg the question.
- **No fitness valleys.** In Dawkins' simulation, an arbitrary string can be transformed into the target by stepwise improvement. This nice property can hardly be expected from the real-world natural selection: very often any such stepwise transformation must go via individuals with lesser fitness than that from which it has started. (The following analogy might be helpful: if a man stands on top of a small hill and wants to get any higher,

Table 1. *Experimental results: the original 'weasel words' model (the correct letters are shown in bold). The size of the population is 100.*

Generation	Fittest individuals	Fitness	Individual's №
0	d p q z **i** k f o b u c d h t h x d j g k j **e** **l**	3	21
	o o **t** g p k n f p f y c g f i **e** o **w** q r w w w		27
	z l o d s r **k** e u t g **t** n k n t **a** r j k p x i		48
	t a u q g **n** d m p h w z y **i** a y **a** b j r q g e		53
3	z l o d s o **k** e u **t** g t n k n j **a** r j k **s** x i	4	719
5	**m** a **t** **h** d k w u p f y c g r a **e** o **w** q r g w w	5	1033
7	n x r p y x **k** g **i** c b n n z **k** b **a** x x **a** q **e** r	6	1337
11	n x r p y x **k** g **i** c b c l z **k** w **a** r x **a** q **e** r	7	2101
	t a **t** **h** g **n** h n **i** v q z y **i** l y **a** **w** z m q g c		2160
14	**m** f **t** **h** d **n** **k** u p f y e l r a w o **w** q y g **e** w	8	2641
15	**m** f **t** **h** d **n** **k** **s** p f y e **l** r s w o **w** q y g **e** w	9	2819
20	**m** f **t** **h** d **n** **k** u p **t** y e **l** r a w o **w** **e** y u **e** z	10	3633
25	**m** f **t** **h** **i** **n** **k** n **i** f x e **l** **i** s w o **w** q y g **e** w	11	4497
	m p **t** **h** d **n** **k** r x f y **s** **l** **i** **k** w o **w** t y g **e** w		4647
28	**m** z **t** **h** d **n** **k** **s** p **t** s **s** **l** r a v o **w** q y s **e** z	12	5170
33	**m** z **t** **h** d **n** **k** **s** p t z **s** **l** r **k** g o **w** p r **s** **e** j	13	6010
35	**m** h **t** **h** d **n** **k** **s** d **t** z **s** **l** r **k** g o **w** t r **s** **e** **l**	14	6487
43	**m** w **t** **h** d **n** **k** r x f **i** **s** **l** **i** **k** **e** e **w** t l **s** **e** **l**	15	7826
50	**m** u **t** **h** d **n** **k** a x f **i** **s** **l** **i** **k** **e** x **w** **e** l **s** **e** **l**	16	9143
55	**m** i **t** **h** **i** **n** **k** **s** **i** **t** **i** k **l** r **k** b **a** **w** m r **s** **e** **l**	17	10143
66	**m** i **t** **h** **i** **n** **k** **s** **i** **t** **i** l **l** r **k** b **a** **w** **e** r **s** **e** **l**	18	12206
73	**m** i **t** **h** **i** **n** **k** **s** **i** **t** **i** k **l** c **k** **e** **a** **w** l **a** **s** **e** **l**	19	13670
81	**m** u **t** **h** **i** **n** **k** a **i** **t** **i** **s** **l** **i** **k** **e** x **w** **e** **a** **s** **e** **l**	20	15217
87	**m** u **t** **h** **i** **n** **k** a **i** **t** **i** **s** **l** **i** **k** **e** **a** **w** **e** **a** **s** **e** **l**	21	16426
114	**m** e **t** **h** **i** c **k** **s** **i** **t** **i** **s** **l** **i** **k** **e** **a** **w** **e** **a** **s** **e** **l**	22	21729
135	**m** e **t** **h** **i** **n** **k** **s** **i** **t** **i** **s** **l** **i** **k** **e** **a** **w** **e** **a** **s** **e** **l**	23	25843

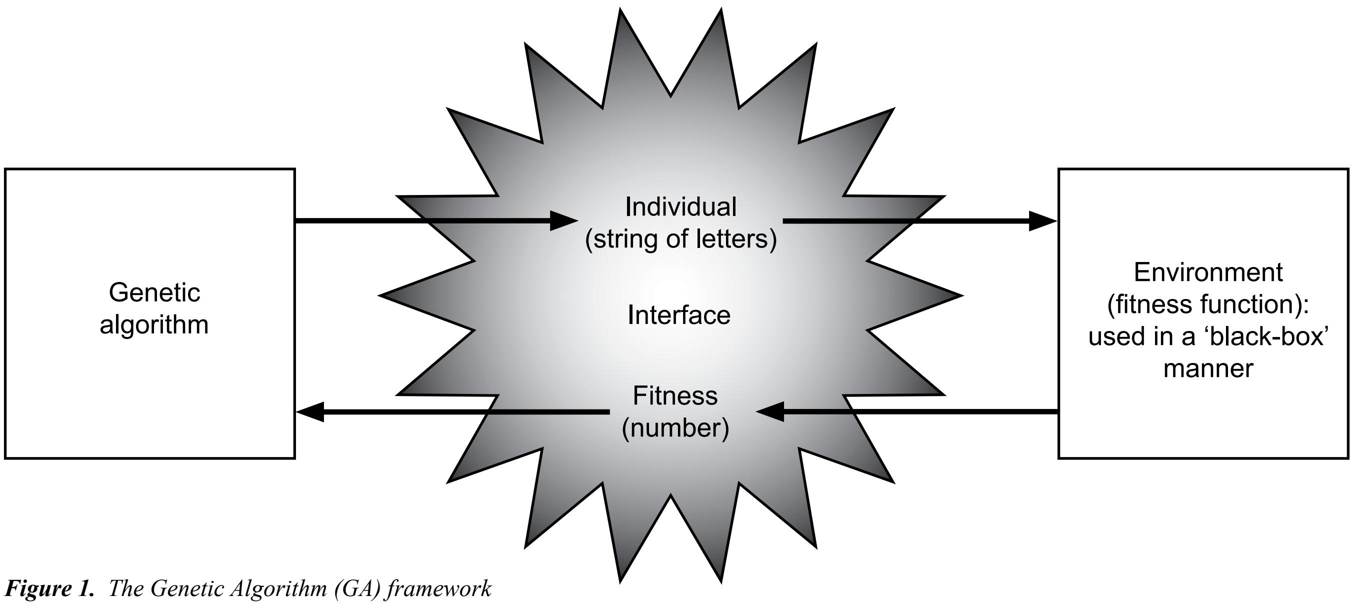

Genetic algorithm
Individual (string of letters)
Interface
Fitness (number)
Environment (fitness function): used in a 'black-box' manner

***Figure 1.** The Genetic Algorithm (GA) framework*

he has first to *come down* into a valley and then climb a higher hill.) Yet, individuals with such lower fitness are eliminated by natural selection, which makes crossing such 'valleys' highly problematic.

Of course, this is only a brief summary of the arguments that have been put forward; the adduced selection is just an attempt to collect and systematize the main points, with no pretence of completeness.

The rest of the paper revolves around the latter two arguments. Essentially, the 'irreducible complexities' are related to the idea of *coarse acknowledgements* (explained later in paper) by the environment, and the time taken for a certain trait to evolve is exponential in the 'coarseness' of the corresponding acknowledgement. Moreover, 'fitness valleys' manifest themselves as an additional problem of *multiextremity* (explained later in paper). Thus this paper offers no profoundly new concepts and arguments—its purpose is to expound and experimentally confirm the already existing ones and give new insights on them.[6]

Genetic algorithms

Dawkins' computer simulation can be seen as a process of solving a particular *optimisation problem,* namely finding a string resembling the target as much as possible (ultimately, with the target itself being the best solution), by a *genetic algorithm (GA).* There is a lot of literature devoted to GAs, the interested reader is referred to the overview by Thomas Bäck[7] and further references therein; here, only a brief description is given.

A GA attempts to find an individual (in this paper, individuals are strings) that is adapted to the environment (represented by a fitness function mapping individuals to their fitness values) as well as possible. A GA works by making a series of queries, passing an individual to the environment and receiving back its fitness value (a number), as shown in figure 1. This fitness value characterizes how well the individual is adapted to the environment, and a GA attempts to maximize it. In the case of Dawkins' example, the fitness is calculated as the number of letters in the given string which are 'on their places', i.e. coincide with the letters in the corresponding positions of the target; thus the fitness of the target itself is 23 whereas the fitness of the string 'xxtxxxxxitxxlikxxxxxxxxx' is 6.

Table 1 illustrates a typical run of a GA with the environment working in this manner.

Note that the environment is quite independent from the GA itself, and one can easily replace it by a completely different environment, provided that the latter has the same *interface,* i.e. the same format of queries. Thus a particular environment represents a particular optimisation problem, and vice versa, by choosing an appropriate environment one can apply the GA to the optimisation problem at hand. This important property will be used further in this paper, viz. the same GA will be applied to three different environments (including the one corresponding to Dawkins' model).

A GA starts off by generating a *population* of fixed size of random individuals, and the environment evaluates them. Once the fitness of each individual is computed, the daughter population of individuals is created in the following way. The individuals from the current generation are replicated (perhaps, several times, according to their fitness, e.g. the ones with the fitness above the average can leave more descendants than the others). Then the *mutation* operator is applied to each generated individual; it changes (mutates) a small number of letters by replacing them by randomly generated ones. In this paper, the *mutation rate* (i.e. the probability of replacing a particular letter of an individual by a random letter) is taken to be 1/23 (the inverse of the sequence length is an often used value in GAs[7]); such a strategy allows the mutation operator to change either none or one or more letters (though it is exponentially unlikely to change many letters). The environment evaluates the individuals generated in this manner, and the fittest ones are retained to form the daughter population (i.e. the population is trimmed back to its initial size by discarding the inferior individuals). The whole process is repeated with this newly generated population, then with its daughter population etc., until some *termination criterion* is met (in this paper, the process is terminated after the target fitness is reached or after a million generations), and the fittest (among all the generations) individual is printed as the result.

Obviously, due to the randomness involved, a particular run of a GA is not guaranteed to find the best possible solution to the problem (i.e. one million iterations might be not enough for a GA to converge). In particular, experiments in this paper indicate that when the acknowledgments provided by the environment are coarse and the optimisation problem is multiextremal, the performance of GAs is severely crippled.

Fine and coarse acknowledgements

Suppose the target phrase is the key to the combination lock protecting a safe with a large amount of money. A burglar is attempting to open it by trying combinations at random. Somehow he has managed to get 90% of the combination right. Does that mean that he will take home 90% of the whole amount? Obviously not. Either he guesses the whole combination and gets the whole amount, or he gets nothing, no matter how close he is to the target. The expected number of combinations the burglar has to try to get access to the safe is 26^{23}, which is more than 3.5 x 10^{32}, so his chances of finishing by the time he dies of old age (or even within a trillion years) are pretty slim indeed. The combination lock illustrates the notion of *irreducible complexity*,[5] i.e. the task of finding the correct combination cannot be subdivided into *independent* simple tasks in such a way that solving one or several (but not all) such tasks yields some benefit.

In contrast, consider the variant of this problem where the safe is subdivided into 23 smaller safes, each protected by a one-letter combination lock (with the money equally distributed between the safes). The burglar's problem is much easier in this case. Indeed, there are only 26 possible combinations for each such lock, which can be checked quite quickly, so the burglar can break open all the safes within half an hour, and even if for some reason he has not managed to open all of them, he takes home some money. In this case, though the same combination of letters is used and the same amount of money is at stake, the problem is *reducibly complex,* i.e. can be subdivided into a series of independent simpler problems.

So, why exactly is the former problem irreducibly complex whereas the latter one not? What is the essential difference between these two problems? It does not take a rocket scientist to figure out that the main difference is that in the latter case *each* lucky guess is *acknowledged* (by rewarding the burglar with a certain amount) whereas in the former case only a *series* of 23 lucky guesses is acknowledged (and if even one guess is not correct the whole series is not acknowledged and not rewarded).

By subdividing the safe in different ways into smaller safes with simple locks one can build a series of problems ranging from trivial ones (as in the case of 23 one-letter locks) to intractable ones (e.g. a single 23-letters lock). For example, the lock can be split into 11 two-letter locks plus a single-letter one. Each two-letter lock has 26^2 = 676 possible combinations, so the burglar can solve the problem in slightly more than two hours (assuming that it takes him a second to try one combination). The problem with 7 three-letters locks plus a two-letters one is even more complicated. Each three-letters lock has 26^3 = 17576 combinations, so the burglar will have to spend the whole weekend working more than 17 hours a day. The problem with 5 four-letters locks plus a three-letters one will take more than a month of hard labour, and so on. As the number of letters in individual locks grows, the problem *very quickly* becomes intractable. Note that when the number of letters in each individual lock is small, short sequences of lucky guesses are acknowledged and so the corresponding problem is simple. Such acknowledgements will be called *fine.* As the number of letters in individual locks grows, longer sequences of lucky guesses are acknowledged, i.e. the acknowledgements become *coarser.*

Behe[5] adduces several examples of irreducibly complex systems in living organisms, such as the biochemistry of human vision and blood clotting, bacterial flagellum (a motor allowing bacteria to swim), a cell 'delivery system' transporting synthesised proteins to their proper place and a few other systems. These systems comprise several (often dozens or more) of proteins working together to accomplish a certain function; moreover, the absence of even one of them renders the whole system useless (or even harmful). That is, such a system could not have come into existence by gradual improvement of a simpler one, much like the opening combination of a lock cannot be obtained by gradual improvement of a random combination: all the systems' parts (all the correct letters in the combination) must be in place, and only then is there a sudden leap in functionality (the safe is opened). *Therefore, the acknowledgements provided by the environment for irreducibly complex systems are coarse. That is, the realistic environment cannot acknowledge (by increasing the returned fitness value – see Figure 1) the potentially positive changes until they show up, i.e. lead to an improvement in functionality.* However, in Dawkins' model every correct letter is acknowledged, i.e. all the acknowledgements are as fine as possible. To make the model more realistic, coarse acknowledgements should also be added.

First, let us investigate the model where the acknowledgement is given only for the target sequence (this corresponds to a single 23-letters lock). The information content of a string of length 23 roughly corresponds to that of a

Table 2. *Experimental results for the model with acknowledgements of different 'coarseness' where only whole words and strings thereof have selective advantage (whole words occurring in the target are shown in bold). The size of the population is 100.*

Generation	Fittest individuals	Fitness	Individual's №
0	p n p s **a** n z w x i **a** f f **i t** t h n u c h y b u i n p v z b **a** k z l h w f **i s** n u x k g f o u j p g g h p e z y c **a i s** v d j m j n h n h	3	17 56 63
4	p n p x **a** y i w x **i s** b f **i t** s h n u c h y b	5	813
17	d u z v q q z z k t l p **a** c x x **i t i s a** b e p u z k q x z v i t l p **a** c x w **i t i s** o l e	9	3868 3886
1394	**l i k e** n f i y h p **a** w i l b l **i t i s** q z y	13	416039
1401	**l i k e a** f i y f p **a** w i l b l **i t i s** q z y	18	417524
135450	**l i k e a** o i u j n k x f y **i t i s l i k e** v	32	40543304
135457	**l i k e** o c m d j z k x f s **i t i s l i k e a**	48	40545001
369318	d d x z y f h **w e a s e l** y **i t i s l i k e a**	54	110549050
369321	d d j z y f **a w e a s e l** y **i t i s l i k e a**	61	110550277
397283	t s **l i k e a w e a s e l** g **i t i s l i k e a**	72	118910326
397290	**i s l i k e a w e a s e l** g **i t i s l i k e a**	85	118912517
1000000	(no improvement)	85	298648624

protein chain of length 25 (note that a protein chain can be represented by a 'string' in the 'alphabet' of 20 amino-acid residues, and $26^{23} \approx 20^{25}$), which is rather short compared with real proteins in organisms (which typically contain from about fifty to about three thousand amino-acid residues).[5] To accomplish this, the environment should assign to the target string the fitness value 1 (or any other positive value) and to all the other strings the value 0. The corresponding optimisation problem is hopeless for the GA,[6] as one can do no better than to explore strings at random. As there are $26^{23} > 3.5 \times 10^{32}$ such strings, the expected time needed to hit the target is 740,000 times greater than the age of the universe generally accepted by evolutionists (15 billion years), even if the computer generates and checks a billion strings per second. Clearly, one should not waste computer time experimenting with this model!

This shows that irreducibly complex proteins of realistic length could not have arisen by chance. Let us now solve an inverse problem and investigate what is the maximum length of an irreducibly complex substring that one can realistically expect to appear during a run of a GA. This is the purpose of the next proposed model, where acknowledgements range from the finest possible to coarser ones. This is accomplished by acknowledging every whole word occurring in the target phrase by adding the length of this word to the fitness value (this reflects the intuition that longer substrings carry more information). If several instances of the same word occur in the string, only one of them is acknowledged (intuitively, the other instances do not carry any *new* information; this also prevents the string from degrading to, e.g. 'a' repeated 23 times). Furthermore, in order to add also some reducible complexities and 'guide' the algorithm towards the target string, every substring (including the target phrase itself) composed of whole words is also acknowledged, by adding the length of this substring to the fitness value (this can be interpreted as synergy, i.e. the total contribution of several substrings towards the fitness might be greater than the sum of individual contributions; e.g. many traits in living organisms require a few proteins to work together). For example, the string 'xxxxitititislikexweasxx' has the fitness value 27 due to the words 'a', 'it' (occurs thrice, but counted only once), 'is', 'like', and the substrings 'itis', 'islike' and 'itislike'. One can easily show that the target (and only it) has the maximum possible fitness value 186.

It should be noted that most of the acknowledgements in this model are quite fine, as most of the words are very short. On the other hand, the acknowledgement for the word 'methinks' is quite coarse. This allows for an experimental investigation of the impact of 'coarseness' on the speed of evolution, by counting how many generations are needed for a given word to 'evolve'. Notice also that the word 'weasel' can potentially evolve 'around' the word 'a', so it is not, strictly speaking, irreducibly complex and can evolve in two stages which are separately acknowledged by the environment; however, the second stages would have quite a coarse acknowledgement.

The experimental results for a typical run of the GA with an environment implementing the described fitness function are shown in table 2. The initial population happened to contain the words 'a', 'is' and 'it', albeit in different individuals, and in generation 4 an individual containing all these words has emerged. In generation 17 the combination 'itis' emerged in two individuals simultaneously. These individuals are very similar due to a recent common predecessor, which however is not related to the best individual in generation 4 (it is not shown in the table as it was less fit). It turns out that this common predecessor contained the word 'is', and a new instance of the word 'it' appeared in front it, so the process took advantage of reducible complexity of this word combination. Then the process did not show any improvement until generation 1,394, when the word 'like' appeared. Note that by that time 416,039 strings have been tried out, which is more than the expected number 26^4/(23-4+1) ≈ 22,849 of strings needed for this word to appear if they were generated at random. Such a delay is due to the attempt of the GA to preserve the substring 'itis' and an instance of the word 'a' by discriminating against any mutations damaging them and thus preventing other words from evolving on their places. Moreover, strings generated by the GA are not independent, as there is a strong correlation between pairs of successive generations. Soon afterwards 'a' was appended to 'like', producing the (reducibly complex) combination 'likea'.

The substring 'itis', being composed of short words, is quite mobile: if another instance of the substring 'it' appears in front of it, forming 'ititis', and then the 't' in the former 'it' is replaced by 's', forming 'itisis', the net effect is to move 'itis' two positions to the left. Similarly, it can move two positions to the right. However, such movements are not acknowledged because they do not increase the fitness (though they do not decrease it either). The only way to acknowledge such a movement is to adjoin the word 'methinks' to the left of 'itis' or the word 'like' to the right of it. However, 'methinks' is too long to wait for, and 'likea' has appeared on the wrong place, so new instances of 'like' are not acknowledged (and are more likely to be destroyed by mutations than 'likea' because their destruction would not reduce the fitness whereas the destruction of 'likea' would). That is why it took so long (further 134,049 generations) for the group 'itislike' to form. Soon afterwards 'a' was appended to it, producing the (reducibly complex) combination 'itislikea'. At this point, the former instance of 'likea' is no longer beneficial and quickly deteriorates due to mutations.

All the shorter words have now appeared in the correct combination, and there is no further progress for a very long time (233,861 generations), until the word 'weasel' emerges. By this time, 110,549,050 strings have been tried out, which is more than expected number 26^6/ (23-6+1) ≈ 17,161,988 if completely random strings were generated. The reasons for this delay are similar to those described for the first instance of the word 'like'.

Once the word 'weasel' has appeared, reducibly complex combinations involving this word start to emerge: in 3 generations 'a' is adjoined to its left, followed by 'like' in further 27,962 generations, and by 'is' in further 7 generations. Then the process fails to make any further improvement for the last 602,710 generations, in spite of the huge number of generated strings.

Several observations concerning this experiment are summarised below.

- Neither the target string nor even the word 'methinks' has 'evolved' within a million generations, in spite of the fact that almost three hundred million strings have been tried out. This is quite a pathetic performance, given that the latter is only 8 letters long—almost nothing compared to irreducible complexities encountered even in the simplest living organisms.
- Although relatively long reducibly complex substrings have evolved, the number of generations needed to produce an irreducibly complex substring exhibits an exponential in the substring's length trend (see the table below). This essentially rules out the possibility of evolution of even relatively short irreducibly complex substrings. In particular, in the performed experiments a word of length 6 has evolved (albeit it took large number of generations), and a word of length 8 has not emerged within a million generations. This is hardly surprising as the expected number of randomly generated strings needed for such a word to appear is $26^8/(23-8+1) > 1.3 \times 10^{10}$, which is more than an order of magnitude greater than the total number of generated individuals. Moreover, as already mentioned, the attempt of the GA to preserve other useful substrings and the strong correlation between the successive generations has a detrimental effect on the emergence of new irreducibly complex substrings.

Word length	1	2	4	6	8
Generation	0	0	1394	369318	—
Individual's №	1	17	416039	110549050	—

- Some of the useful substrings (both reducibly and irreducibly complex) had to evolve several times in different places, which further reduced the speed of evolution.
- The best individual generated by the GA has two copies of the substring 'islikea', and the future generations have to preserve them both, as destroying any of them significantly reduces the fitness. Moreover, many of the words have evolved in wrong places. These two factors virtually bury any hope for further improvement (indeed, no improvement has been shown for the last 602,710 generations).

In the performed experiments, the GA was run 100 times with different seed values of the random number generator, producing alternative versions of evolution, but no substantially different results have been achieved. In particular, neither the target string nor the word 'methinks' has ever emerged.

Due to the accelerated mutation rate and the perfect selection assumed in the proposed model, it must be a few orders of magnitude faster than what can be expected from nature; that is, the simulated million generations amount to many billions real-life generations. As the information content of a string of English letters is just slightly greater than that of a protein of comparable length, evolution is too slow to account for the existence of even relatively simple proteins employed in living organisms, not to mention living organisms themselves.

The problem of multiextremity for genetic algorithms

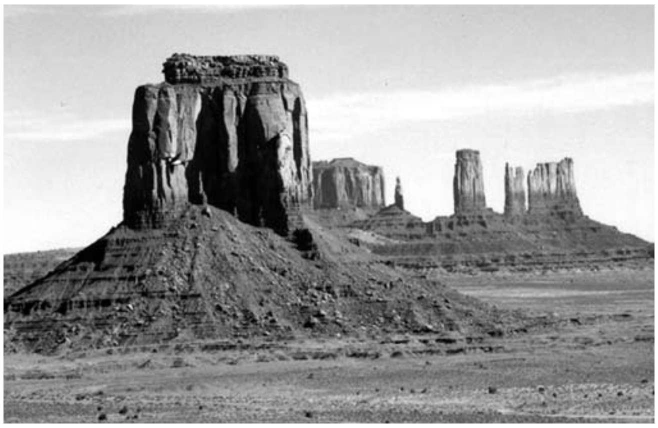
As was explained earlier, GAs are intended for solving optimisation problems, i.e. problems of finding the maximum values of functions. They do this by making a series of small random mutations ('steps') followed by selection, i.e. removal of the unfit.

The traditional visualization of such iterative optimisation strategies is hill climbing: the fitness value corresponds to the geographic altitude, and mutations to steps made by the climber. Suppose the climber makes a step in a random direction. If this step leads uphill (it is assumed that the climber can always determine his altitude), he continues climbing from the higher position. Otherwise, he simply discards this step and tries another direction. The hill climbing is being done in the night-time, so the climber does not have the global view of surroundings and has to move blindly; e.g. he cannot choose to cross a valley because he sees a better slope on its opposite end.

Of course, this analogy is quite crude; nevertheless, in spite of all its shortcomings, it is very popular and has proved to be helpful in practice, because it allows explaining optimisation strategies (as well as problems faced by them!) in vivid and intuitive terms. Dawkins also uses a similar analogy in one of his books, likening the process of evolution to climbing a high Mount Improbable:

> 'Mount Improbable rears up from the plain, lofting its peaks dizzily to the rarefied sky. The towering, vertical cliffs of Mount Improbable can never, it seems, be climbed. Dwarfed like insects, thwarted mountaineers crawl and scrabble along the foot, gazing hopelessly at the sheer, unattainable heights. They shake their tiny, baffled heads and declare the brooding summit forever unscalable.
>
> Our mountaineers are too ambitious. So intent are they on the perpendicular drama of the cliffs, they do not think to look round the other side of the mountain. There they would find not vertical cliffs and echoing canyons but gently inclined grassy meadows, graded steadily and easily towards the distant uplands. Occasionally the gradual ascent is punctuated by a small, rocky crag, but you can usually find a detour that is not too steep for a fit hill-walker in stout shoes and with time to spare. The sheer height of the peak doesn't matter, so long as you don't try to scale it in a single bound. Locate the mildly sloping path and, if you have unlimited time, the ascent is only as formidable as the next step. The story of Mount Improbable is, of course, a parable. We shall explore its meaning in this and the next chapters.'[8]

It should be noted, however, that this analogy is somewhat different from the one used in this paper, where neither vertical cliffs nor rocky crags pose any problem: it is feasible for a GA to make a sudden leap, significantly increasing the fitness in a single step. For example, the substring 'wexsel' is useless, yet a single mutation can change it into 'weasel', significantly increasing the fitness. (Thus the hill-climbing analogy should be amended: the climber walks upon a flat topographic map showing the altitudes as shades of colour, and by making a single step he is capable of 'scaling' a high vertical cliff.) The real problem is large tablelands (corresponding to coarse acknowledgements): the climber can go in circles for a long time not being able to discriminate between possible directions: whatever step he makes, he ends up at the same altitude, rendering natural selection useless.


Dawkins' concept of climbing Mt. Improbable is an inadequate analogy for the 'chasmatic' hurdles to be overcome by evolution.

The topography corresponding to the original Dawkins' model is a single mountain with gentle slopes covering the whole map; at each point (distinct from the top) it is possible to make a step increasing the altitude. This is obviously the easiest possible topography for the climber to cope with. On the contrary, the topography corresponding to the example with a combination lock is a single pole standing somewhere in a huge plateau (with the area exceeding that of the Earth's surface by the factor of 200,000,000,000,00 0,000): the climber would have to roam about blindly until he bumps into it. He is *extremely* lucky if he finds it within a trillion years!

The hill climbing analogy helps to illustrate a formidable problem of multiextremity with stepwise optimisation strategies. Suppose the climber happens to be in the vicinity of some small hill. Not having a global view of the surroundings, he does not know that this hill is too small to be worthy of pursuing and scales it. Once at the top, he discovers that the only possible way is downhill and thus he gets confined there, with no hope of further progress! In other words, it is very difficult for a GA to cross a valley, because doing that means decreasing fitness at the initial stage of the process. Yet, individuals with such lower fitness are eliminated by natural selection. (Some GAs use crossovers and/or mutations considerably changing the string, which potentially allow them to escape from the top of a hill; however, such big changes can be considered as huge random steps in the dark: it gets vanishingly unlikely to jump to a higher hill if the number of high hills is small compared to the size of the search space.) Note that in the original Dawkins' model the problem of multiextremity does not show up because the corresponding topography has no hills except the target. However, in slightly less artificial settings it clearly manifests itself: this is exactly what happened with the example in

Table 2: the process has reached a point from which it is impossible to progress without first substantially decreasing the fitness.

Thus, in order to climb Mount Improbable one has to be lucky enough to start in its vicinity rather than in the vicinity of some small hill. But how to do this if it is surrounded by Dwarf Hills, Pigmy Knolls, Midget Hillocks and Mole Mounds? The climber risks scaling one of those, burying the hope of further progress due to the need to cross a valley. Again, a lot of luck is required!

However, it should be noted that this argument is less strong than that about coarse acknowledgements, because it heavily depends on the 'topography' of the optimisation problem, which is extremely hard to describe for realistic environments and living organisms. Though in the performed experiments the problem of multiextremity did have a significant impact on the final results, it still remains to be shown that this is the case for the processes in nature. That is, one has to show (quantitatively!) that there are a lot of low-quality maximums and very few with the quality comparable to that of existing living organisms, and so the probability of reaching one of such high-quality maximums is negligible.

Conclusion

GAs are sometimes adduced as an argument supporting evolution: they 'work' in some practical applications, so the theory of evolution which had inspired them must be correct. However, the experiments in this paper show that GAs are no silver bullet: they perform satisfactorily only if the acknowledgements are fine and there is no need to cross valleys. This is indeed the case for the original Dawkins' model: it converges within a few hundred generations. As soon as the acknowledgements are even slightly coarser, the performance deteriorates dramatically; in particular, the word 'methinks' of length 8 has not 'evolved' within a million generations. While relatively long reducibly complex substrings can indeed emerge, the number of generations required for a given irreducibly complex one to appear grows exponentially with the 'coarseness' of the corresponding acknowledgement. Many features encountered in living organisms are irreducibly complex[5] and thus the corresponding acknowledgements are coarse; this makes the probability of their emerging as a result of evolution vanishingly small.

Another problem with GAs is that they have difficulties with crossing fitness valleys. This problem does not exist in the original Dawkins' model but clearly manifests itself in less artificial settings.

Of course, both Dawkins' model and the ones proposed in this paper are simple rather than precise. (Any computer model attempting to describe multiple generations of organisms is bound to be crude!) However, the model with coarse acknowledgements is more realistic as it better reflects the process of natural selection. It demonstrates that the process of evolution is much slower than Dawkins' model suggests. And, in addition to all this complexity, there is still the major problem how the first self-replicating entity came about.

It seems appropriate to finish with the following quote from Behe's book:

> 'The knowledge we now have of life at the molecular level has been stitched together from innumerable experiments in which proteins were purified, genes cloned, electron micrographs taken, cells cultured, structures determined, sequences compared, parameters varied, and controls done. Papers were published, results checked, reviews written, blind alleys searched, and new leads fleshed out.
>
> 'The result of these cumulative efforts to investigate a cell—to investigate life at the molecular level—is a loud, clear, piercing cry of 'design!' The result is so unambiguous and so significant that it must be ranked as one of the greatest achievements in the history of science.'[5]

References

1. Dawkins, R., *The Blind Watchmaker*, Penguin Books, London, 1990.
2. Ey, L. and Batten, D., Weasel, a flexible program for investigating deterministic computer 'demonstrations' of evolution, *TJ* **16**(2): 84–88, 2002.
3. Truman, R., Dawkins' weasel revisited, *TJ* **12**(3): 358–361, 1998.
4. Gitt, W. and Wieland, C., Weasel words, *Creation* **20**(4): 20–21, 1998.
5. Behe, M., *Darwin's Black Box*, Simon & Schuster, 1998.
6. The software used in the experiments is available for download from <homepages.cs.ncl.ac.uk/victor.khomenko/home.formal/tools/weaseloid>.
7. Bäck, T., Evolutionary computation: a guided tour, *Bulletin of the European Association of Theoretical Computer Science* **77**: 132–166, June 2002.
8. Dawkins, R., *Climbing Mount Improbable*, Penguin Books, London, 1996.

Victor Khomenko has been active in creationist ministry since 2002. He obtained MSc in Applied Mathematics and Computing Science from Kiev Taras Shevchenko University and PhD in Computing Science from the University of Newcastle upon Tyne. His areas of interests include verification and synthesis of software and hardware.