Evolving Dynamics in an Artificial Regulatory Network Model

P. Dwight Kuo[1], André Leier[2] and Wolfgang Banzhaf[1]

[1] Department of Computer Science, Memorial University of Newfoundland, St. John's NL A1B 3X5, CANADA
[2] Department of Computer Science, University of Dortmund, D–44221 Dortmund, GERMANY

Abstract. In this paper artificial regulatory networks (ARN) are evolved to match the dynamics of test functions. The ARNs are based on a genome representation generated by a duplication / divergence process. By creating a mapping between the protein concentrations created by gene excitation and inhibition to an output function, the network can be evolved to match output functions such as sinusoids, exponentials and sigmoids. This shows that the dynamics of an ARN may be evolved and thus may be suitable as a method for generating arbitrary time–series for function optimization.

1 Introduction

It has been recognized that understanding the differences between species (and thus the key to evolution) lies in the DNA information controlling gene expression since only a tiny fraction of DNA is translated into proteins [1]. Regulation appears to be a very reasonable answer for a functional role for unexpressed DNA. According to Neidthardt [2] and Thomas [3], 88% of the genome of *E. Coli* is expressed with 11% suspected to contain regulatory information.

Since many evolutionary effects can be traced back to their regulatory causes, regulatory networks mediate between development and evolution and thus serve to help shape organism morphology and behavior [4]. Studying models of regulatory networks can help us to understand some of these mechanisms by providing lessons for both natural and artificial systems under evolution.

It has been previously shown that our regulatory network model is able to reproduce dynamic phenomena found in natural genetic regulatory networks [5]. One example is the ability to capture shifts in the onset and offset of gene expression (heterochrony) based on single bit–flip mutations. As such, this model can relate changes in time and intensity to tiny pattern changes on bit strings. This could possibly provide the algorithmic "missing link" between genotypes subject to constant evolutionary changes and the remarkably stable phenotypes found in the real world. In addition, this model has previously been shown to generate scale–free and small world topologies [6] and network motifs [7].

Recently, there has been significant interest in modelling regulatory networks in the evolutionary computation literature [5, 8–14]. Features of regulatory networks have been previously used in the context of optimization by [8, 10, 14]. However, these models have been explicitly designed for artificial ontogeny. Here we propose the use of a regulatory network framework as a general method for evolving arbitrary time series. Obtaining arbitrary functions through evolutionary means for the purpose of model optimization has been previously performed for flying [15], locomotion [16] and the inference of differential equations [17].

In addition, previous models of ARNs primarily use boolean representations of network dynamics [8, 9, 12, 13]. Here we show that an ARN model using differential equations (approximated as difference equations) can also display complex behaviors which may be selected by evolution.

Other ideas relating to genetic transcription have also previously been used in function optimization such as genetic–code transformations [18], gene expression [19, 20], gene signaling [21] and diploidity [22].

2 Artificial Regulatory Network Model

The ARN consists of a bit string representing a genome with direction (i.e. 5' $\rightarrow$ 3' in DNA) and mobile "proteins" which interact with the genome through their constituent bit patterns. In this model, proteins are able to interact with the genome most notably at "regulatory" sites located upstream from genes. Attachment to these sites produces either inhibition or activation of the corresponding protein. Therefore, these interactions may be interpreted as a regulatory network with proteins acting as transcription factors.

The genome itself is created through a series of whole length duplication / divergence events. Creation of a genome in such a manner has been shown to generate network topologies which have similarities to biological networks such as having scale–free and small world topology as well as network motifs [6, 7]. First, a random 32–bit string is generated. This string is then used in a series of length duplications similar to those found in nature [23] followed by mutations in order to generate a genome of length L_G. A "promotor" bit sequence of 8–bits was then arbitrarily selected to be "01010101". By randomly choosing "0"s and "1"s to generate a genome, any one–byte pattern can be expected to appear with probability $2^{-8} = 0.39\%$. Since the promotor pattern itself is repetitive, overlapping promotors or periodic extensions of the pattern are not allowed, i.e. a bit sequence of "0101010101" (10–bits) is detected as a single promotor site starting at the first bit. However regions associated with one gene may overlap with another should a promotor pattern also exist within a portion of the coding region of a gene.

The promotor signals the beginning of a gene on the bit string analogous to an open reading frame (ORF) on DNA – a long sequence of DNA that contains no "stop" codon and therefore encodes all or part of a protein. Each gene is set to a fixed length of $l_{gene} = 5$ 32–bit integers which results in an expressed bit pattern of 160 bits. Genes can thus be created on the genome by complete

duplications of previously created genes, mutation, and / or combinations of the ending and starting sequences of the genome during duplication.

Immediately upstream from the promotor sites exist two additional 32–bit segments which represent the enhancer and inhibitor sites. As previously mentioned, attachment of proteins (transcription factors) to these sites results in changes to protein production for the corresponding genes (regulation). In this model, we assume only one regulatory site for the increase of expression and one site for the decrease of expression of proteins. This is a radical simplification since natural genomes may have 5–10 regulatory sites that may even be occupied by complexes of proteins [4].

Processes such as transcription, diffusion, spatial variations and elements such as introns, RNA–like mobile elements and translation procedures resulting in a different alphabet for proteins are neglected in this model. This last mechanism is replaced as follows: Each protein is a 32–bit sequence constructed by a many–to–one mapping of its corresponding gene which contains five 32–bit integers. The protein sequence is created by performing the majority rule on each bit position of these five integers so as to arrive at a 32–bit protein. Ties (not possible with an odd number for l_g) for a given bit position are resolved by chance.

Proteins may then be examined to see how they may "match" with the genome, specifically at the regulatory sites. This comparison is implemented by using the XOR operation which returns a "1" if bits on both patterns are complementary. In this scheme, the degree of match between the genome and the protein bit patterns is specified by the number of bits set to "1" during an XOR operation. In general it can be expected that a Gaussian distribution results from measuring the match between proteins and bit sequences in a randomly generated genome [4]. By making the simplifying assumption that the occupation of both of a gene's regulatory sites modulates the expression of its corresponding protein, we may deduce a gene–protein interaction network comprising the different genes and proteins which can be parameterized by strength of match. The bit–string for one gene is shown in Figure 1.

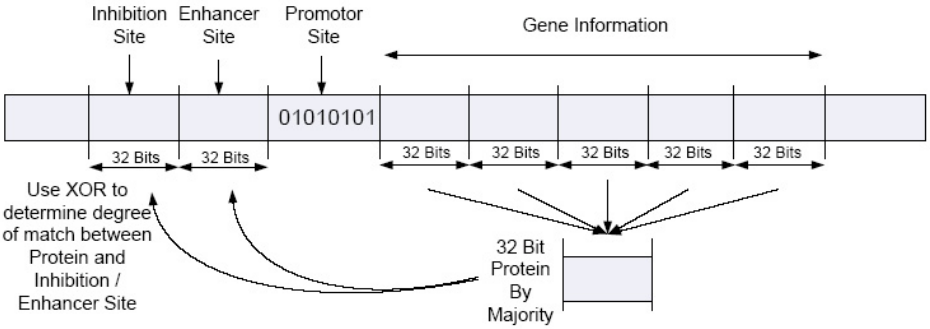


Fig. 1: Bit string for one gene in the ARN model.

The rate at which protein i is produced is given by:

$$\frac{dc_i}{dt} = \frac{\delta\left(e_i - h_i\right) c_i}{\sum_j c_j} \tag{1}$$

$$e_i, h_i = \frac{1}{N} \sum_j^N c_j \exp\left(\beta(u_j - u_{max})\right) \tag{2}$$

where e_i and h_i represent the excitation and inhibition of the production of protein i, u_j represents the number of matching bits between protein j and activation or inhibition site i, u_{max} represents the maximum match (in this case, 32), β and δ are positive scaling factors, and c_i is the concentration of protein i at time t. Note that the concentrations of the various proteins are required to sum to 1. This ensures that there is a competition between binding sites for proteins.

It can be noted that the ARN model presented bears some resemblance to a recurrent neural network (RNN). In the ARN, genes and the match strength between inhibition / activation sites and proteins are analogous to the neurons and connection strengths in an RNN framework.

3 Optimization

By simulating the ARN model presented in the previous section, we obtain a dynamical view of the protein concentrations in the system. However, such a system has no assigned semantics – the protein concentrations have no meaning outside the system. In addition, since the protein concentrations are limited to sum to 1 (i.e. $\sum c_i = 1$), generation of some functions is excluded. In order to use such a system for the purpose of optimization, a mapping is required. An additional 64–bit sequence is randomly selected along the genome as a binding site for the desired output function. The first 32–bits specify the inhibition site while the second 32–bits specify the activation site. The proteins generated by the ARN are free to also bind to these additional regulatory sites. The levels of activation and inhibition are calculated in the same way as in Section 2, Equation 2.

However, instead of calculating a "concentration" of this site (which generates no protein of its own), the activity at this site is simply summed and used directly as an output function:

$$s(t) = \sum_i \left(e_i - h_i\right) \tag{3}$$

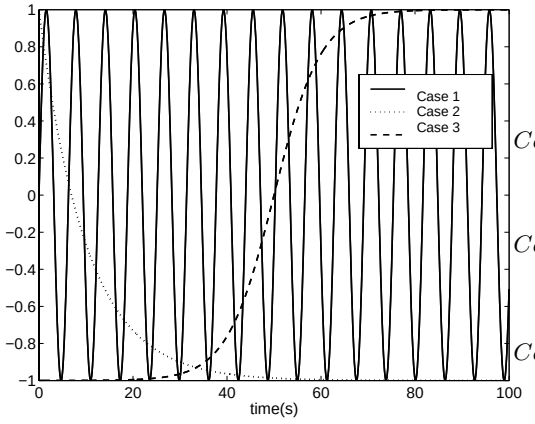
Subsequent normalization of $s(t)$ to between -1 and 1 generates the dynamics of the specific genome. Thus, the additional binding sites added to the genome may be thought of as a method with which to extract dynamics from the changes in

concentrations of the proteins in the ARN model. Further sites may be added to the genome for the extraction of additional signals.

In order to evolve solutions $s(t)$, a simple (50+100)–Evolutionary Strategy (ES) is used [24]. Genomes were generated by 10 duplication events per genome subject to 1% mutation (without selection) leading to individual genomes of length $L_G = 32768$. It has been shown that a mutation rate of 1% during the duplication / divergence process is sufficient to "rewire" parts of the topology of the network without making it completely random [6].

The number of genes in each genome is given by the number of promotor patterns present as was previously defined in Section 2. Each generation, 100 new individuals are created from the current population using a 1% single–point (bit-flip) mutation (i.e. on average, 328 mutations per genome). The fitness of these solutions was calculated and the best 50 of 150 (parents + children) proceed to the next generation. ES was stopped when the best solution found was not improved upon for 250 generations.

The objective is to minimize the fitness function calculated as the mean square error (MSE) between the desired function and the evolved function. The following cases were examined and are shown in Figure 2:



$$Case\ 1: \quad sin(t) \tag{4}$$

$$Case\ 2: \quad 2\exp(-0.1t) - 1 \tag{5}$$

$$Case\ 3: \quad \frac{2}{1+\exp(-0.2t+10)} - 1 \tag{6}$$

Fig. 2: Plot of the three fitness cases.

All solutions were generated with a time step, $dt = 0.1s$. The initial protein concentrations (the initial conditions for the differential equation) are set to be $\frac{1}{\#ofgenes}$ to remain within the simplex. In addition, the first 100 time steps (10s) are ignored. This is done in order to exclude the startup dynamics of the model. Thus, for calculation of the fitness function, the normalized output generated by the ARN model from time $t = 10...110s$ is compared with the fitness case $f(t)$ from time $t = 0...100s$.

4 Results

Tables 1, 2 and 3 summarize the results of 10 evolutionary runs each for the three fitness cases. Figures 3, 5 and 7 show the actual function generated by the best individual of each run for the three fitness cases. Figures 4, 6 and 8 show the progress of the best evolutionary run for each fitness case.

It is clearly shown that the ARN model accurately generates dynamics approximating the sinusoid (Figure 3), the exponential (Figure 5) and the sigmoid (Figure 7) functions with good accuracy for all runs. In all fitness cases and evolutionary runs, the MSE calculated was less than 0.00588654. Additional support for the success of these simulations can be seen in the final population fitness averages shown in Tables 1, 2 and 3. The average population fitness values (MSE) are relatively small with low standard deviation. This indicates that the population is such that all or virtually all individuals when simulated generate functions that closely approximate the respective objective functions.

Run #	*Best MSE*	*#Generations*	*#Genes*	*Avg. MSE(Pop.)*	*Avg. #Genes (Pop.)*
1	0.00101533	1235	154	0.00150(0.00013)	147.59(20.6)
2	0.00035992	557	36	0.00068(0.00012)	39.22(2.40)
3	0.00001843	758	100	0.00004(0.00001)	102.45(2.93)
4	0.00001732	721	96	0.00004(0.00001)	96.55(2.80)
5	0.00011328	617	97	0.00025(0.00006)	102.78(4.02)
6	0.00002073	825	104	0.00013(0.00005)	109.78(5.03)
7	0.00005429	465	108	0.00044(0.00018)	112.37(11.4)
8	0.00016598	879	177	0.00047(0.00022)	186.02(9.87)
9	0.00005034	575	195	0.00031(0.00012)	212.16(9.57)
10	0.00002219	987	39	0.00006(0.00001)	39.49(2.42)

Table 3: Results of 10 runs of (50+100)–ES on Case 3. Standard deviation in brackets.

5 Conclusions

It has been demonstrated that the dynamics of a differential equation based ARN model initially created through a duplication / divergence process can be evolved towards simple functions. This might suggest that such an approach may also be appropriate for generating arbitrary functions suitable for use in applications such as model optimization.

Due to the way in which the genes are detected on the genome, there are plentiful opportunities for individuals in the population to acquire neutral mutations. It has been previously shown that neutral mutation can be extremely beneficial in the context of evolution [25]. Since there may exist extensive non–coding regions of the genome, neutral mutations are free to be collected in such regions with new genes appearing suddenly when a new promotor pattern has been created through mutation. As well, each of the networks generated for each

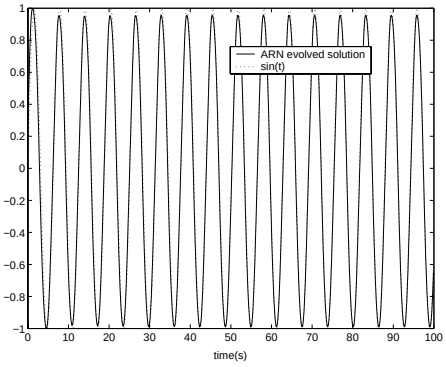


Fig. 3: Plot of best solution (run #8) compared to ideal solution for Case #1. The MSE is 0.000151746.

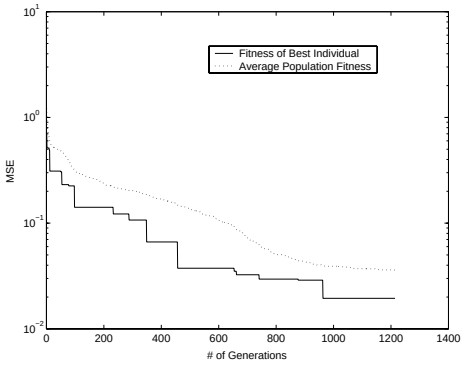


Fig. 4: Plot of the fitness of the best solution (run #8) and the average fitness using (50+100)–ES for Case #1.

Run #	*Best MSE*	*#Generations*	*#Genes*	*Avg. MSE(Pop.)*	*Avg. #Genes (Pop.)*
1	0.001445217	731	47	0.00287(0.000765)	45.31(5.72)
2	0.001165628	381	74	0.00316(0.000780)	76.92(3.42)
3	0.000614281	1214	105	0.00114(0.000147)	117.59(4.57)
4	0.000747053	835	234	0.00291(0.000817)	244.00(13.2)
5	0.001861556	428	63	0.00326(0.000684)	75.08(9.34)
6	0.000640149	1077	101	0.00186(0.000347)	102.49(4.08)
7	0.001561523	315	26	0.00440(0.000847)	32.78(5.55)
8	0.000151746	1040	124	0.00058(0.000131)	135.63(6.32)
9	0.000519559	933	71	0.00134(0.000341)	92.88(53.2)
10	0.000846462	858	55	0.00270(0.000449)	48.57(3.22)

Table 1: Results of 10 runs of (50+100)–ES on Case 1. Standard deviation in brackets.

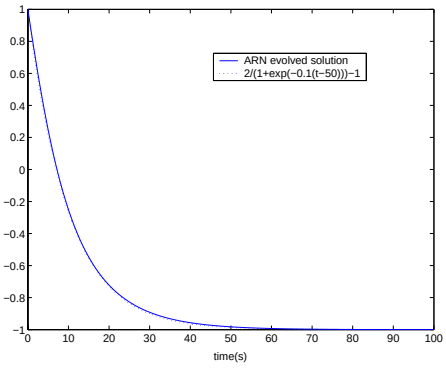


Fig. 5: Plot of best solution (run #3) compared to ideal solution for Case #2. The MSE is 0.00363873.

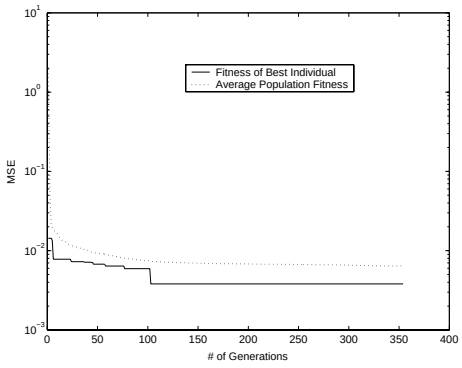


Fig. 6: Plot of the fitness of the best solution (run #3) and the average fitness using (50+100)–ES for Case #2.

Run #	*Best MSE*	*#Generations*	*#Genes*	*Avg. MSE(Pop.)*	*Avg. #Genes (Pop.)*
1	0.00411971	708	133	0.00447(0.000134)	142.83(5.88)
2	0.00478168	642	166	0.00554(0.000250)	185.95(13.5)
3	0.00363873	354	27	0.00641(0.000553)	52.22(7.00)
4	0.00441011	359	20	0.00660(0.000610)	31.95(7.38)
5	0.00381064	747	97	0.00505(0.000303)	106.81(5.71)
6	0.00402240	877	63	0.00464(0.000180)	58.83(4.17)
7	0.00426413	501	128	0.00574(0.000354)	116.14(8.75)
8	0.00537858	287	176	0.00661(0.000458)	164.40(11.1)
9	0.00511630	466	58	0.00688(0.000563)	54.26(3.73)
10	0.00588654	519	45	0.00643(0.000171)	45.65(3.10)

Table 2: Results of 10 runs of (50+100)–ES on Case 2. Standard deviation in brackets.

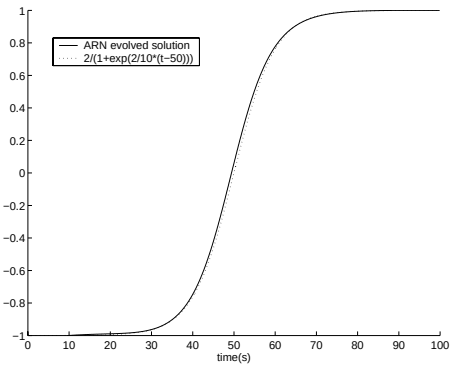


Fig. 7: Plot of best solution (run#4) compared to ideal solution for Case #3. The MSE is 0.0000173162.

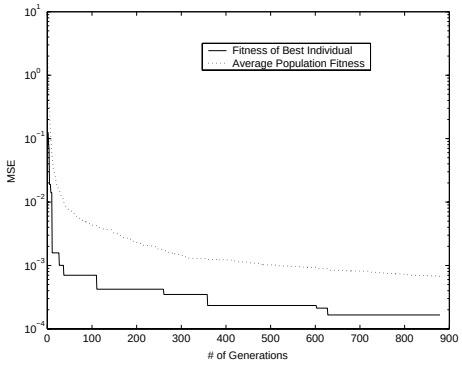


Fig. 8: Plot of the fitness of the best solution (run#4) and the average fitness using (50+100)–ES for Case #3.

fitness case contains a different topology (number of genes). Therefore, due to the quality of solutions, it may be inferred that there are many different networks which can give good approximations to each of the fitness cases.

Unfortunately, it is difficult to determine how the number of genes in the genome affects the evolvability or richness of dynamics in the system. It is an open question within this framework how the number of genes affects the ability of the system to generate functions of a given type. Another interesting area of further inquiry is to determine the minimum number of genes required for a given function. Further studies investigating the evolvability and ability of the ARN model to represent other classes of functions as well as arbitrary functions are necessary before the use of regulatory networks in function optimization can be systematically and fully realized. However, this contribution is a first step in that direction.

6 Acknowledgements

The authors would like to thank William Langdon and the reviewers for helpful comments.

References

1. Hood, L., Galas, D.: The digital code of DNA. Nature **421(6921)** (2003) 444–448
2. Neidhardt, F.C.: Escherichia Coli and Salmonella Typhimurium. ASM Press, Washington, DC (1996)
3. Thomas, G.H.: Completing the E.Coli proteome: a database of gene products characterised since completion of the genome sequence. Bioinformatics **15(10)** (1999) 860–861
4. Banzhaf, W.: On the dynamics of an artificial regulatory network. In Banzhaf, W., Christaller, T., Dittrich, P., Kim, J.T., Ziegler, J., eds.: Advances in Artificial Life – Proceedings of the 7th European Conference on Artificial Life (ECAL). Volume 2801 of Lecture Notes in Artificial Intelligence., Springer–Verlag (2003) 217–227
5. Banzhaf, W.: Artificial regulatory networks and genetic programming. In Riolo, R.L., Worzel, B., eds.: Genetic Programming Theory and Practice. Kluwer (2003) 43–62
6. Kuo, P.D., Banzhaf, W.: Scale–free and small world network topologies in an artificial regulatory network model. Ninth International Conference on the Simulation and Synthesis of Living Systems (ALIFE) **(in press)** (2004)
7. Banzhaf, W., Kuo, P.D.: Network motifs in artificial and natural transcriptional regulatory networks. Journal of Biological Physics and Chemistry **(in submission)** (2004)
8. Watson, J., Wiles, J., Hanan, J.: Towards more relevant evolutionary models: Integrating an artificial genome with a developmental phenotype. In: Proceedings of the Australian Conference on Artificial Life (ACAL). (2003) 288–298
9. Hallinan, J., Wiles, J.: Evolving genetic regulatory networks using an artificial genome. In Chen, Y.P.P., ed.: Second Asia-Pacific Bioinformatics Conference (APBC2004). Volume 29 of CRPIT., Dunedin, New Zealand, ACS (2004) 291–296

10. Bongard, J.: Evolving modular genetic regulatory networks. In: Proceedings of the IEEE 2002 Congress on Evolutionary Computation, IEEE Press (2002) 1872–1877
11. Hotz, P.E.: Genome–physics as a new concept to reduce the number of genetic parameters in artificial evolution. In: Proceedings of the IEEE 2003 Congress on Evolutionary Computation, IEEE Press (2003) 191–198
12. Willadsen, K., Wiles, J.: Dynamics of gene expression in an artificial genome. In: Proceedings of the IEEE 2003 Congress on Evolutionary Computation, IEEE Press (2003) 199–206
13. Reil, T.: Dynamics of gene expression in an artificial genome: Implications for biological and artificial ontogeny. In Floreano, D., Nicoud, J.D., Mondada, F., eds.: Advances in Artificial Life – Proceedings of the 5th European Conference on Artificial Life (ECAL). Volume 1674 of Lecture Notes in Computer Science., Springer–Verlag (1999) 457–466
14. Bongard, J.C., Pfeifer, R.: Evolving complete agents using artificial ontogeny. In Hara, F., Pfeifer, R., eds.: Morpho–functional Machines: The New Species (Designing Embodied Intelligence). Springer–Verlag (2003) 237–258
15. Augustsson, P., Wolff, K., Nordin, P.: Creation of a learning, flying robot by means of evolution. In Langdon, W.B., Cantú-Paz, E., Mathias, K., Roy, R., Davis, D., Poli, R., Balakrishnan, K., Honavar, V., Rudolph, G., Wegener, J., Bull, L., Potter, M.A., Schultz, A.C., Miller, J.F., Burke, E., Jonoska, N., eds.: GECCO 2002: Proceedings of the Genetic and Evolutionary Computation Conference, Morgan Kaufmann Publishers (2002) 1279–1285
16. Dittrich, P., Burgel, A., Banzhaf, W.: Learning to move a robot with random morphology. In Husbands, P., Meyer, J.A., eds.: Proceedings of the First European Workshop on Evolutionary Robotics. Volume 1468 of Lecture Notes in Computer Science., Springer-Verlag (1998) 165–178
17. Cao, H., Kang, L., Chen, Y., Yu, J.: Evolutionary modeling of systems of ordinary differential equations with genetic programming. Genetic Programming and Evolvable Machines **1(4)** (2000) 309–337
18. Kargupta, H., Ghosh, S.: Toward machine learning through genetic code–like transformations. Genetic Programming and Evolvable Machines **3(3)** (2002) 231–258
19. Kargupta, H.: The gene expression messy genetic algorithm. In: Proceedings of the IEEE 1996 Congress on Evolutionary Computation, IEEE Press (1996) 814–819
20. Eggenberger, P.: Evolving morphologies of simulated 3d organisms based on differential gene expression. In Harvey, I., Husbands, P., eds.: Proceedings of the 4th European Conference on Artificial Life (ECAL), MIT Press (1997) 205–213
21. Goldberg, D.E., Korb, B., Deb, K.: Messy genetic algorithms: Motivation, analysis and first results. Complex Systems **3(5)** (1989) 493–530
22. Yoshida, Y., Adachi, N.: A diploid genetic algorithm for preserving population diversity – pseudo–meiosis GA. In Davidor, Y., Schwefel, H.P., Männer, R., eds.: The Third Conference on Parallel Problem Solving from Nature(PPSN). Volume 866 of Lecture Notes in Computer Science., Springer–Verlag (1994) 36–45
23. Wolfe, K., Shields, D.: Molecular evidence for an ancient duplication of the entire yeast genome. Nature **387(6634)** (1997) 708–713
24. Beyer, H.G., Schwefel, H.P.: Evolution strategies: A comprehensive introduction. Natural Computing **1(1)** (2002) 3–52
25. Yu, T., Miller, J.: Neutrality and the evolvability of boolean function landscapes. In: Proceedings of the 4th European Conference on Genetic Programming (EuroGP). Volume 2038 of Lecture Notes in Computer Science., Springer–Verlag (2001) 204–217