






A Study of Direct and Indirect Encoding in Phenotype-Genotype Relationships

Clyde Meli¹ (✉) , Vitezslav Nezval¹, Zuzana Kominkova Oplatkova² ,
Victor Buttigieg³ , and Anthony Spiteri Staines¹

¹ Department of Computer Information Systems, University of Malta, Msida, Malta
{clyde.meli, tony.spiteri-staines}@um.edu.mt, vnez@cis.um.edu.mt

² Faculty of Applied Informatics, Department of Informatics and Artificial Intelligence,
Tomas Bata University in Zlín, Zlín, Czech Republic
oplatkova@utb.cz

³ Department of Communications and Computer Engineering, University of Malta,
Msida, Malta
victor.buttigieg@um.edu.mt

Abstract. This paper examines phenotype and genotype mappings that are biologically inspired. These types of coding are used in evolutionary computation. Direct and indirect encoding are studied. The determination of genotype and phenotype relationships and the connection to genetic algorithms, evolutionary programming and biology are examined in the light of newer advances. The NEAT and HyperNEAT algorithms are applied to the 2D Walker [41] problem of an agent learning how to walk. Results and findings are discussed, and conclusions are given. Indirect coding did not improve the situation. This paper shows that indirect coding is not useful in every situation.

Keywords: Indirect encoding · Direct coding · Genotype · Phenotype · Genetic algorithms · Evolutionary programming · Neuroevolution · NEAT · HyperNEAT

1 Introduction to Phenotype and Genotype Mappings

This paper deals with the study of the influence of direct and indirect encoding for the artificial neural network design by evolutionary computation, i.e. neuroevolution. Evolutionary computation meets terminology as phenotype (behaviour, physiology, the morphology of an organism) and genotype (genetic coding of the organism). The relationship between both terms has been the subject of various investigations, including [1–3]. Such phenotype mapping has been used to predict disease-related genes. Van Driel et al. [1] say that phenotypes can be used to predict biological interactions, which are the effect which two genes have on each other.

Many evolutionary computation systems, such as genetic programming (GP) work with direct encodings (sometimes the term coding is used). Program trees representing a computer program evaluated recursively are used as genotypes. They translate directly into solutions. Some hybrid systems operate on graphs so there may be a problem with

devising direct genetic operators. In a direct encoding [4], the genotype specifies every neuron and connection explicitly. Not all authors differentiate between direct encodings and structural encodings like [5]. Structural encoding implies that the encoding also holds information on connection weights. This is used when a genetic algorithm (GA) evolves Artificial Neural Network (ANN) parameters. In such an encoding, there are few constraints to the GA's exploration [5].

The minimal alphabet principle in Gas [6] specifies the selection of the smallest alphabet that permits a natural problem expression. This holds for direct encoding but for indirect encoding the search space can be reduced [7].

1.1 Direct Encodings for ANNs

Montana [8] reviewed how GAs can be used to represent and train ANNs. The genetic representation must include the network topology, real-valued weights associated with every link and real-valued bias associated with every node.

Specifically, an example of a direct and structural encoding of an ANN, as found in [8] is given in Fig. 1. Typically, the layout, such as the number of nodes in each layer and the number of hidden layers, is fixed, and not part of the encoding.

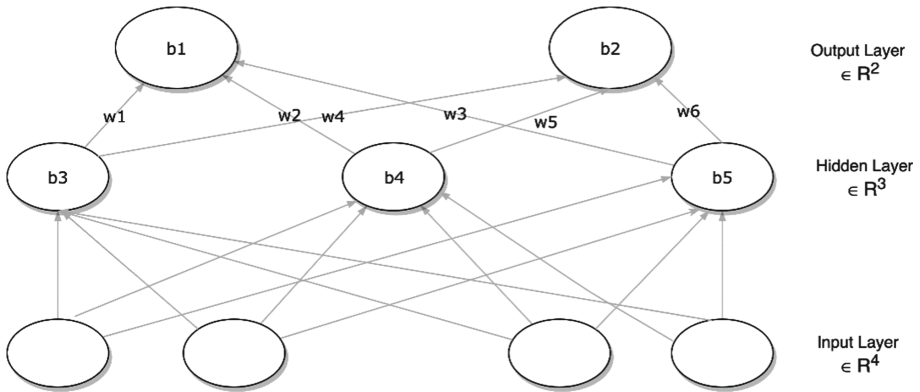


Fig. 1. ANN with one hidden layer

The ANN in Fig. 1 would be encoded as the following chromosome $(w_1, w_2, w_3, w_4, w_5, w_6, b_1, b_2, w_7, w_8, w_9, w_{10}, w_{11}, w_{12}, w_{13}, w_{14}, w_{15}, w_{16}, w_{17}, w_{18}, b_3, b_4, b_5)$.

This representation scheme degrades its performance of convergence as the network size is enlarged [9]. It is noted that using direct encodings, it is normally not possible to represent the ANN graphical structure geometrically [10]. This lack of geometry restricts ANNs from evolving brain-like structures.

1.2 Indirect Encodings for ANNs

[9] devised a genetic algorithm representation for ANNs, which represents connections and network topology. The existence of a connection is represented by ‘1’ and its absence

by a '0'. An extension of L-Systems [11] called Graph L-System, is used to generate graphs. Starting from an axiom of a 2×2 matrix, a set of deterministic rewriting rules are applied for edges and nodes represented by symbols. The connectivity matrix will enlarge every generation and the last generation will contain '1' and '0's representing feed-forward network connections as an upper-right triangle; only the bold connections are used, the rest are discarded. An example representation for the XOR problem [12] is given in Fig. 2 where the symbol S is a starting axiom and symbols A, B, C, D stand for nonterminal for the grammar graph generation based on Graph L-System. The details of the concept are described in [9] including rewriting rules to final '1' and '0's.

For the first line, '01100' represents a network with the top node not having a connection from itself, and only the next two nodes connecting to the top node.

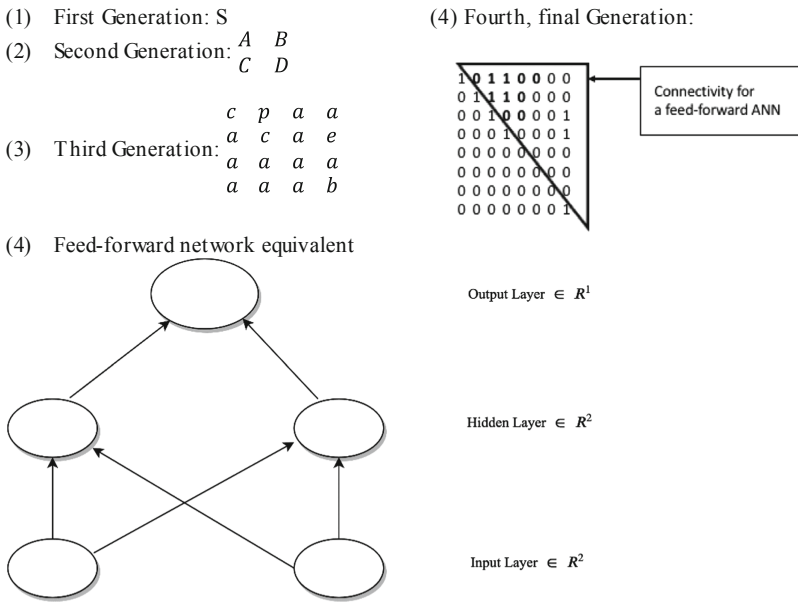


Fig. 2. 2-2-1 XOR Network Generations (based on an example from [9])

1.3 The Evolutionary Search Space of Indirect Encodings

Pigliucci [13] quotes Hartmann et al. [7] as having determined that indirect encoding dramatically reduced the evolutionary search space in evolutionary computation as opposed to the classical direct encoding. However, on a closer look, this is actually not something that Hartmann et al. claim as fact in all cases and situations. Indeed, they only claimed this holds for evolved digital circuits. So, further research may be required in this area to see whether indirect encoding would be beneficial in certain cases. The importance of this can be seen in the field of genetics, where [13] assumed that indirect encoding was found to be beneficial in all cases by an analogy that biological systems should

behave like simulated evolutionary models. On that unproven argument, it is claimed in [13] that the old metaphors of genetic blueprints and genetic programs are misleading or inadequate. [13] claims developmental or indirect encoding must be a promising basis to understand evolvability and the Genotype-Phenotype (G-P) mapping problem.

Interestingly, developmental encoding itself was inspired by biological development [14]. More recently, Clune et al. [15] showed an indirect encoding outperforming direct encoding. On the contrary, Harding and Miller [16] showed that for lower complexity (in the sense of Kolmogorov) encoding patterns, direct encoding sometimes performs worse than indirect encodings.

Other studies utilising generative encodings include Clune and Lipson [17], Meli [18], Jacob and Rozenberg [11] (using the grammar by Lindenmayer [19]) and Kitano [9].

Kwasnicka and Paradowski [20] found that in a few generations, indirect encoding gave excellent solutions though the evolved networks were larger than the equivalent directly encoded ones. Da Silva et al. also demonstrated the benefit of indirect encoding [21] where the quality of the Particle Swarm Optimisation (PSO), GA and GP-based solutions for web services using using indirect encoding was higher than the equivalent baseline direct encoding approach for twelve out of thirteen datasets. Also Hotz [22] in a case study of lens shape evolution schemes showed that indirect encoding converged faster than direct ones.

Compared to the advantages stated in the above-mentioned papers, [23] found in a TETRIS problem that HyperNEAT, a tool for indirect encoding in ANNs, is superior to NEAT (a tool for direct encoding in ANNs) early in evolution, but this fizzles out eventually and NEAT performs better. Authors showed that HyperNEAT was better for raw but NEAT performed better for hand-designed features. The aim of this paper is to compare direct and indirect encoding on a problem to analyse the performance of evolution.

2 Indirect Encoding

Encodings in evolutionary programming are typically binary, real-valued, graph-based, computer code [24]. The selected representation affects the effectiveness of a genetic algorithm [25] and probably even other binary-coded evolutionary computation. Direct encoding is claimed to be ineffective [14]. Indirect encoding can vary. One form, developmental encoding, employs gene reuse. Used in evolutionary computation, this is referred to as embryogeny. This is called Artificial Embryogeny [14]. Some exciting research in indirect encoding involves ANNs. An ANN is represented as a binary tree with grammar rules generating the ANN as seen in [9].

In GP, one finds the use of indirect encodings like Cartesian GP (CGP) [26] and Grammatical Evolution [27]. They can also help in “encoding neutrality”. *Neutrality* is defined as a situation when small genotype mutations (*neutral mutations*) do not have an effect on the fitness of the expressed phenotype. Miorandi et al. [28] explain that a neutral mutation gives a substantial advantage for state space exploration. This can potentially increase the evolvability of a population providing further robustness.

Indirect encoding representation can allow optimisation to occur without restrictions, since “functional constraints are subsequently enforced during the decoding step” [21].

An issue which Ronald [29] finds with the developmental form of indirect encoding used mainly in hybrid systems is that some of these encodings do not address the entire search space. On the other hand, an alternate representation may address the entire search space, as the Millipede [18] representation does by combining or folding several points in the search space into one. Similarly, Della Croce et al. [30] in their GA, which solves the Traveling Salesperson Problem (TSP) employ a lookahead representation based on Falkenauer and Bouffoix's Linear Order Crossover (LOX). Another GP representation uses a block-oriented representation instead of the usual direct one [31]. It describes how a block diagram is built rather than the directly coded structure.

3 Summary of Research on Indirect Encoding

Research on indirect encoding has been summarized in two tables, Table 1 involving evolutionary computation with neural networks and Table 2 which involved the use of other evolutionary computation techniques.

Table 1. Research involving indirect encoding with ANN or other neural networks

Paper	Indirect coding used
Clune et al. [15]	Evolution of Compositional Pattern Producing Networks (CPPNs) via Hypercube-based NeuroEvolution of Augmenting Technologies (HyperNEAT) algorithm. The latter outperformed direct encodings in three problem domains
Clune et al. [32]	Comparison of HyperNEAT and FT-NEAT algorithms, the former outperformed the latter in a quadruped problem
D'Ambrosio and Stanley [33]	Multiagent HyperNEAT outperformed multiagent Sarsa(λ)
Gillespie et al. [23]	HyperNEAT and NEAT comparison in TETRIS, NEAT overtakes HyperNEAT eventually
Hussain and Browse [5]	Evolving Neural Networks using Attribute Grammars
Kitano [9]	Artificial Neural Network (ANN) with GA and Graph L-System
Kwasnicka and Paradowski [20]	Neural Network with Direct and Indirect encoding, the latter does not reach the global optimum but gives good ANNs in few generations compared to direct encoding

Table 2. Research involving indirect encoding with other evolutionary computation techniques

Paper	Indirect coding used
Aickelin [35]	Indirect GA with a hill climber algorithm
Aickelin and Dowsland [36]	Indirect GA with a heuristic decoder and hybrid crossover operator
Brucherseifer et al. [31]	GP with block-oriented encoding
da Silva et al. [21]	PSO, GA and GP with four indirect representations
Haj-Rachid et al. [37]	GA comparisons of Indirect and Direct coding. Best direct encoding performance with PMX crossover and Inversion Mutation. Best indirect encoding performance with OX crossover with inversion mutation
Hartmann et al. [7]	Cartesian Genetic Programming using a symbolic netlist representation
Hotz [22]	Evolution strategies with indirect coding
Jacob and Rozenberg [11]	Genetic L-System Programming (GLP) Paradigm for development of L-Systems
Meli [18]	Genetic Algorithm (GA) with Millipede encoding
Thangavelautham and D'Eleuterio [38]	Artificial Neural Tissue (ANT) GP architecture with introns

3.1 Basic Processing of HyperNEAT

HyperNEAT is also known as Hypercube-based NEAT [10], where NEAT stands for NeuroEvolution of Augmented Topologies. The main idea in HyperNEAT is that it is possible to learn relationships when the solution is represented indirectly. It is a generative description of the connectivity of the ANN rather than searching for and tuning the connection weights of the ANN itself. Such approach is very important for evolution of large scale neural networks with huge amount of nodes and many more of connections.

HyperNEAT uses Compositional Pattern Producing Networks (CPPNs) which can produce augmented structures including possibility to use any activation functions via evolutionary process of genetic algorithm. Figure 3 briefly shows the steps involved in the HyperNEAT algorithm.

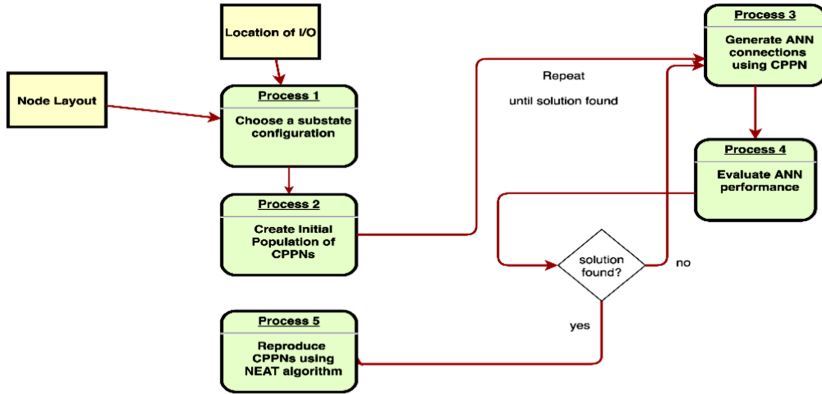


Fig. 3. HyperNEAT processing

4 Experiment

The experiment compared two direct and indirect encoding algorithms used for ANNs. The direct algorithm chosen was the NEAT algorithm [34], and the indirect algorithm used was HyperNEAT [10]. Recall that NEAT is used to evolve dense ANN network nodes and connections specifically. It uses a direct encoding because of the difficulty in obtaining extensive knowledge of how such encoding will be used and, such indirect search can be biased [34]. HyperNEAT extends the NEAT algorithm. It utilizes Compositional Pattern Producing Networks (CPPNs) to evolve ANNs using principles from the NEAT algorithm. These algorithms were preferred due to the availability of easily usable software, SharpNEAT¹.

4.1 Experimental Parameters

The open-source SharpNEAT neuroevolution C#.NET implementation was compiled and used unmodified to compare the two algorithms, NEAT and HyperNEAT, applied to the 2D Walker problem [41]. Settings were taken from [39], which compared the two algorithms applied to the T-Maze learning problem [39, 40] used HypersharpNEAT instead of SharpNEAT. The 2D Walker problem was chosen as it was the only problem available in SharpNEAT using both NEAT and HyperNEAT algorithms. The problem was investigated by [41], who used their implementation of NEAT, Covariance Matrix Adaptation Evolution Strategy (CMAES) and other deep reinforcement learning algorithms. The 2D Walker task involves an agent learning how to walk. [32] investigated a similar quadruped gait problem and concluded that HyperNEAT performed better than FT-NEAT, a directly encoded algorithm. They used the ODE physics simulator with a small population of 150, 1000 generations and 50 runs. The fitness function for 2D Walker is the mean hip position (if it is positive, otherwise 0) over five trials squared.

Every run consisted of 500 generations, the population size was 500, and 10% elitism was used. 50% of sexual offspring were not mutated. The asexual offspring “had 0.94

¹ Available at <https://sharpneat.sourceforge.io/>.

probability of link weight mutation, 0.03 chance of link addition, and 0.02 chance of node addition” [39]. Initial connections proportion was set to 0.05. Mutate connection weights was set to 0.89, mutate delete connection to 0.025. Connection weight range was five.

4.2 Results

Figure 4 below shows the results of running the NEAT and HyperNEAT algorithms using the SharpNEAT application for 500 generations, charting mean fitness.

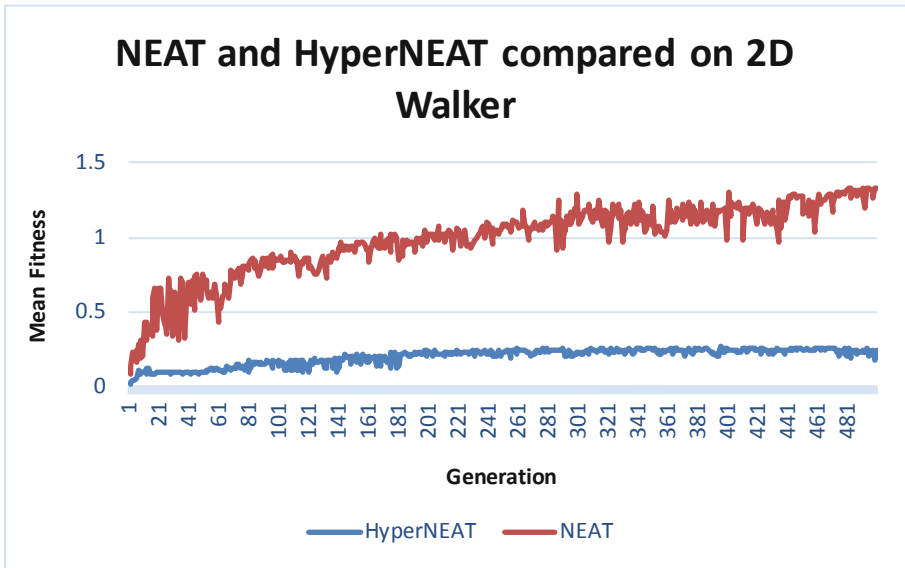


Fig. 4. NEAT vs HyperNEAT algorithm results

Interestingly, NEAT performed well on the task. It can be seen how the HyperNEAT algorithm using indirect coding does not manage to get good mean fitness values. Indeed all values are smaller than 0.3. On the other hand, the NEAT algorithm manages to get close to 1.3 mean fitness. By generation 200 the mean fitness for HyperNEAT is 0.239891, whereas the equivalent for NEAT is 0.96752. The largest mean fitness reached by HyperNEAT is 0.264622 at generation 396. This is never reached again by the algorithm. The largest mean fitness reached by NEAT is that of 1.333688, at generation 493.

Table 3 shows the mean and maximum fitnesses reached in the last generation 500 averaged across runs for NEAT and HyperNEAT.

The Mann-Whitney U test was performed across the final generation runs for mean fitness since the data is not normally distributed. This showed that the difference is significant ($U = 0$; z-score is 5.39649, p -value is $<.00001$ and the result is significant at $p < 0.05$).

Table 3. Evaluation of mean and average of runs (Mean and Maximum fitnesses at Generation 500 for NEAT and HyperNEAT)

	Mean NEAT	Mean HyperNEAT	Max NEAT	Max HyperNEAT
Mean	1.1381	0.2925	2.5640	0.4874
Std Dev	0.4296	0.0701	1.6208	0.0588

The results clearly ascertain how indirect coding in the Walker problem does not help. Indeed, it appears to have stifled the evolution. Some other tests were made to see if HyperNEAT could be improved, e.g. using the larger number of generations used by [32] or interspecies crossover, however there was no improvement.

5 Conclusion

Most current research into indirect encoding involves GP, GAs or ANNs. This paper has covered the issues involving indirect coding which are important aspects of evolutionary computation. We looked at differences involving indirect and direct representation. Significant progress has been made in the theoretical and practical developments. More research is needed involving indirect coding with neural networks and other evolutionary computation techniques, resulting in better representations.

These negative results do not show that indirect encoding is better than direct coding for the 2D Walker problem. This shows an example where indirect encoding is hard to apply successfully and is less effective, reminiscent of [23].

Further research should clarify whether indirect encoding is better than direct encoding in some areas, as well as which categories of problems might be convenient for either type of encoding. It definitely cannot be claimed that indirect coding is always better. This will also ascertain the veracity of Pigliucci's claim that "old metaphors of genetic blueprints and genetic programmes are misleading or inadequate" [13].

Acknowledgments. This work was supported by financial support of research project NPU I No. MSMT-7778/2014 by the Ministry of Education of the Czech Republic, by the European Regional Development Fund under the Project CEBIA-Tech No. CZ.1.05/2.1.00/03.0089 and by resources of A.I. Lab research group at Faculty of Applied Informatics, Tomas Bata University in Zlin (ailab.fai.utb.cz).

References

1. Van Driel, M.A., Bruggeman, J., Vriend, G., Brunner, H.G., Leunissen, J.A.: A text-mining analysis of the human phenome. *Eur. J. Hum. Genet.* **14**, 535 (2006)
2. Meli, C.: Using a GA to determine genotype and phenotype relationships. In: *European Simulation and Modelling Conference 2007*. Westin Dragonara, St Julians (2007)

3. Fogel, D.B.: Phenotypes, genotypes, and operators in evolutionary computation. In: Proceedings of the 1995 IEEE International Conference on Evolutionary Computation (ICEC 1995), pp. 193–198 (1995)
4. Galushkin, A.I.: *Neural Networks Theory*. Springer, Heidelberg (2007). <https://doi.org/10.1007/978-3-540-48125-6>
5. Hussain, T.S., Browse, R.A.: Evolving neural networks using attribute grammars. In: 2000 IEEE Symposium on Combinations of Evolutionary Computation and Neural Networks. Proceedings of the First IEEE Symposium on Combinations of Evolutionary Computation and Neural Networks (Cat. No. 00), pp. 37–42 (2000). <https://doi.org/10.1109/ECNN.2000.886217>
6. Goldberg, D.E.: *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley (1989)
7. Hartmann, M., Haddow, P.C., Lehre, P.K.: The genotypic complexity of evolved fault-tolerant and noise-robust circuits. *Biosystems*. **87**, 224–232 (2007)
8. Montana, D.J., Davis, L.: Training feedforward neural networks using genetic algorithms. In: *IJCAI*, pp. 762–767 (1989)
9. Kitano, H.: Designing neural networks using genetic algorithms with graph generation system. *Complex Syst.* **4**, 461–476 (1990)
10. Gauci, J., Stanley, K.O.: Autonomous evolution of topographic regularities in artificial neural networks. *Neural Comput.* **22**, 1860–1898 (2010)
11. Jacob, C.: Genetic L-system programming. In: Davidor, Y., Schwefel, H.-P., Männer, R. (eds.) *PPSN 1994*. LNCS, vol. 866, pp. 333–343. Springer, Heidelberg (1994). https://doi.org/10.1007/3-540-58484-6_277
12. Zhao, Y., Deng, B., Wang, Z.: Analysis and study of perceptron to solve XOR problem. In: *The 2nd International Workshop on Autonomous Decentralized System*, 2002, pp. 168–173 (2002) <https://doi.org/10.1109/IWADS.2002.1194667>
13. Pigliucci, M.: Genotype-phenotype mapping and the end of the ‘genes as blueprint’ metaphor. *Philos. Trans. R. Soc. Lond. B Biol. Sci.* **365**, 557–566 (2010)
14. Stanley, K.O., Miikkulainen, R.: A taxonomy for artificial embryogeny. *Artif. Life*. **9**, 93–130 (2003)
15. Clune, J., Stanley, K.O., Pennock, R.T., Ofria, C.: On the performance of indirect encoding across the continuum of regularity. *IEEE Trans. Evol. Comput.* **15**, 346–367 (2011)
16. Harding, S., Miller, J.F.: A comparison between developmental and direct encodings. Presented at the *GECCO 2006* (Updated version) (2006)
17. Clune, J., Lipson, H.: Evolving three-dimensional objects with a generative encoding inspired by developmental biology. In: *ECAL*, pp. 141–148 (2011)
18. Meli, C.: Millipede, an extended representation for genetic algorithms. In: *International Journal of Computer Theory and Engineering*. IACSIT PRESS, Rome, Italy (2013)
19. Lindenmayer, A.: Mathematical models for cellular interactions in development. I. Filaments with one-sided inputs. *J. Theor. Biol.* **18**, 280–299 (1968)
20. Kwasnicka, H., Paradowski, M.: Efficiency aspects of neural network architecture evolution using direct and indirect encoding. In: Ribeiro, B., Albrecht, R.F., Dobnikar, A., Pearson, D.W., Steele, N.C. (eds.) *Adaptive and Natural Computing Algorithms*, pp. 405–408. Springer, Vienna (2005). https://doi.org/10.1007/3-211-27389-1_98
21. da Silva, A.S., Mei, Y., Ma, H., Zhang, M.: Evolutionary computation for automatic web service composition: an indirect representation approach. *J. Heuristics*. **24**, 425–456 (2018)
22. Hotz, P.E.: Comparing direct and developmental encoding schemes in artificial evolution: a case study in evolving lens shapes. In: *Proceedings of the 2004 Congress on Evolutionary Computation* (IEEE Cat. No. 04TH8753), vol. 1, pp. 752–757 (2004). <https://doi.org/10.1109/CEC.2004.1330934>

23. Gillespie, L.E., Gonzalez, G.R., Schrum, J.: Comparing direct and indirect encodings using both raw and hand-designed features in tetris. In: Proceedings of the Genetic and Evolutionary Computation Conference, pp. 179–186. Association for Computing Machinery, Berlin (2017). <https://doi.org/10.1145/3071178.3071195>
24. Kicinger, R., Arciszewski, T., De Jong, K.: Evolutionary computation and structural design: a survey of the state-of-the-art. *Comput. Struct.* **83**, 1943–1978 (2005)
25. Caruana, R.A., Schaffer, J.D.: Representation and hidden bias: Gray vs. binary coding for genetic algorithms. In: Machine Learning Proceedings 1988, pp. 153–161. Elsevier, Amsterdam (1988)
26. Miller, J.F., Thomson, P.: Cartesian genetic programming. In: Poli, R., Banzhaf, W., Langdon, W.B., Miller, J., Nordin, P., Fogarty, T.C. (eds.) EuroGP 2000. LNCS, vol. 1802, pp. 121–132. Springer, Heidelberg (2000). https://doi.org/10.1007/978-3-540-46239-2_9
27. Ryan, C., Collins, J.J., Neill, M.O.: Grammatical evolution: evolving programs for an arbitrary language. In: Banzhaf, W., Poli, R., Schoenauer, M., Fogarty, T.C. (eds.) EuroGP 1998. LNCS, vol. 1391, pp. 83–96. Springer, Heidelberg (1998). <https://doi.org/10.1007/BFb0055930>
28. Miorandi, D., Yamamoto, L., De Pellegrini, F.: A survey of evolutionary and embryogenic approaches to autonomic networking. *Comput. Netw.* **54**, 944–959 (2010). <https://doi.org/10.1016/j.comnet.2009.08.021>
29. Ronald, S.: Robust encodings in genetic algorithms: a survey of encoding issues. Presented at the (1997). <https://doi.org/10.1109/ICEC.1997.592265>
30. Della Croce, F., Tadei, R., Volta, G.: A Genetic algorithm for the job shop problem. *Comput. Oper. Res.* **22**, 15–24 (1995). [https://doi.org/10.1016/0305-0548\(93\)E0015-L](https://doi.org/10.1016/0305-0548(93)E0015-L)
31. Brucherseifer, E., Bechtel, P., Freyer, S., Marenbach, P.: An indirect block-oriented representation for genetic programming. In: Miller, J., Tomassini, M., Lanzi, P.L., Ryan, C., Tettamanzi, A.G.B., Langdon, W.B. (eds.) EuroGP 2001. LNCS, vol. 2038, pp. 268–279. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-45355-5_21
32. Clune, J., Beckmann, B.E., Ofria, C., Pennock, R.T.: Evolving coordinated quadruped gaits with the HyperNEAT generative encoding. In: 2009 IEEE Congress on Evolutionary Computation, pp. 2764–2771 (2009). <https://doi.org/10.1109/CEC.2009.4983289>
33. D’Ambrosio, D.B., Stanley, K.O.: Scalable multiagent learning through indirect encoding of policy geometry. *Evol. Intell.* **6**, 1–26 (2013). <https://doi.org/10.1007/s12065-012-0086-3>
34. Stanley, K.O., Miikkulainen, R.: Evolving neural networks through augmenting topologies. *Evol. Comput.* **10**, 99–127 (2002). <https://doi.org/10.1162/106365602320169811>
35. Aickelin, U.: An indirect genetic algorithm for set covering problems. *J. Oper. Res. Soc.* **53**, 1118–1126 (2002)
36. Aickelin, U., Dowsland, K.: An indirect genetic algorithm for a nurse scheduling problem. *Comput. Oper. Res.* **31**, 761–778 (2008). [https://doi.org/10.1016/S0305-0548\(03\)00034-0](https://doi.org/10.1016/S0305-0548(03)00034-0)
37. Haj-Rachid, M., Ramdane-Cherif, W., Chatonnay, P., Bloch, C.: Comparing the performance of genetic operators for the vehicle routing problem. *IFAC Proc.* **43**, 313–319 (2010). <https://doi.org/10.3182/20100908-3-PT-3007.00068>
38. Thangavelautham, J., D’Eleuterio, G.M.T.: A coarse-coding framework for a gene-regulatory-based artificial neural tissue. In: Capcarrère, M.S., Freitas, A.A., Bentley, P.J., Johnson, C.G., Timmis, J. (eds.) ECAL 2005. LNCS (LNAI), vol. 3630, pp. 67–77. Springer, Heidelberg (2005). https://doi.org/10.1007/11553090_8
39. Risi, S., Stanley, K.O.: Indirectly encoding neural plasticity as a pattern of local rules. In: Doncieux, S., Girard, B., Guillot, A., Hallam, J., Meyer, J.-A., Mouret, J.-B. (eds.) SAB 2010. LNCS (LNAI), vol. 6226, pp. 533–543. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-15193-4_50

40. Soltoggio, A., Bullinaria, J.A., Mattiussi, C., Dürr, P., Floreano, D.: Evolutionary advantages of neuromodulated plasticity in dynamic, reward-based scenarios. In: Proceedings of the 11th International Conference on Artificial Life (Alife XI), pp. 569–576. MIT Press (2008)
41. Zhang, S., Zaiane, O.R.: Comparing deep reinforcement learning and evolutionary methods in continuous control (2017). <https://arxiv.org/abs/1712.00006>