

# Self-Adaptive Genotype-Phenotype Maps: Neural Networks as a Meta-Representation

Luís F. Simões<sup>1</sup>, Dario Izzo<sup>2</sup>, Evert Haasdijk<sup>1</sup>, and A. E. Eiben<sup>1</sup>

<sup>1</sup> Vrije Universiteit Amsterdam, The Netherlands

<sup>2</sup> European Space Agency, The Netherlands

dario.izzo@esa.int, {e.haasdijk,a.e.eiben,luis.simoes}@vu.nl

**Abstract.** In this work we investigate the usage of feedforward neural networks for defining the genotype-phenotype maps of arbitrary continuous optimization problems. A study is carried out over the neural network parameters space, aimed at understanding their impact on the locality and redundancy of representations thus defined. Driving such an approach is the goal of placing problems' genetic representations under automated adaptation. We therefore conclude with a proof-of-concept, showing genotype-phenotype maps being successfully self-adapted, concurrently with the evolution of solutions for hard real-world problems.

**Keywords:** Genotype-Phenotype map, Neuroevolution, Self-adaptation, Adaptive representations, Redundant representations.

## 1 Introduction

Automated design of Evolutionary Algorithms (EAs) has been on the research agenda of the Evolutionary Computing community for quite some years by now. The two major approaches for finding good values for the numeric and/or symbolic parameters (a.k.a. EA components) are parameter tuning and parameter control, employed before or during the run, respectively. The current state of the art features several techniques to find 'optimal' values or instances, for all parameters and components, with one notable exception: the genotype-phenotype (G-P) mapping, a.k.a. the representation. In contrast to all other parameters related to selection, variation, and population management, there are only a handful of papers devoted to adapting representations. Considering the widely acknowledged importance of having a good representation, the lack of available techniques to 'optimise' it is striking. One could say that the challenge of tuning and/or controlling the representation in an EA is the final frontier in automated EA design.

In this paper we investigate the possibility of using adjustable genotype-phenotype maps for continuous search spaces. In particular, we propose neural networks (NN) as the generic framework to represent representations (i.e., NNs as a meta-representation). This means that for a given phenotype space  $\Phi_p \subset \mathbb{R}^n$  and a genotype space  $\Phi_g \subset \mathbb{R}^m$  the set of all possible representations we consider is the set of all NNs mapping  $\Phi_g$  to  $\Phi_p$ .

In the technical sections of the paper we investigate this approach, with regard to its expressiveness, and learnability. Informally, we consider a meta-representation expressive if it is versatile enough to define a wide range of transformations, granting it the potential power to restructure arbitrary fitness landscapes into more efficiently explorable genotype spaces for the underlying optimizer. As for learnability, we are pragmatic. All we want to demonstrate at this stage, is the existence of a learning mechanism that can change the NN during the run of the given EA, in such a way that the EA performance (measured by solution quality) is improved. Thus, in terms of the classic tuning-control division, we want to demonstrate the existence of a good control mechanism for NN-based representations.

## 2 Related Work

A fundamental theoretical result by Liepins & Vose [7], shows that “virtually all optimizable (by any method) real valued functions defined on a finite domain [are] theoretically easy for genetic algorithms given appropriately chosen representations”. However, “the transformations required to induce favorable representations are generally arbitrary permutations, and the space of permutations is so large that search for good ones is intractable”. Nevertheless, [7] still calls for research into approaches that adapt representations at a meta-level. In particular, [7] shows affine linear maps to provide sufficient representational power to transform fully deceptive problems into easy ones.

In the over two decades since [7], a vast body of work emerged, addressing ways to automatically adapt and control all sorts of Evolutionary Algorithm components [4,8]. The genetic representation, and its genotype-phenotype map, however, despite their recognized role as critical system components, have only sporadically been addressed in the literature. De Jong, in [2], considers that “perhaps the most difficult and least understood area of EA design is that of adapting its internal representation.”

In a series of papers culminating in [3], Ebner, Shackleton & Shipman proposed several highly-redundant G-P maps, the cellular automaton (CA) and random boolean network (RBN) mappings, in particular, being of special relevance to the present research. In them, chromosomes are composed of a dynamical system’s definition (CA or RBN rule table, and also the cell connectivity graph in the case of RBN), along with the system’s initial state. Decoding into the phenotype space takes place by iterating the dynamical system for a number of steps, from its initial state, according to the encoded rule table.

## 3 Neural Networks as Genotype-Phenotype Maps

We have chosen to use neural networks as the basis of our approach for two reasons. First, they are global function approximators. In principle, they are capable of expressing any possible G-P map (given a sufficiently large number of

hidden layer neurons). Second, they are learnable, even evolvable. There is much experience and know-how about ‘optimising’ neural nets by evolution [5,11].

Formally, and following Rothlauf’s notation [9, Sec. 2.1.2], we then have that, under an indirect representation scheme, evolution is guided by a fitness function  $f$  that is decomposed into a genotype-phenotype map  $f_g$ , and a phenotype-fitness mapping  $f_p$ . Genetic operators such as recombination and mutation are applied over genotypes  $x^g \in \Phi_g$  (where  $\Phi_g$  stands for the genotypic search space). A genotype-phenotype map  $f_g : \Phi_g \rightarrow \Phi_p$  decodes genotypes  $x^g$  into their respective phenotypes  $x^p \in \Phi_p$  (where  $\Phi_p$  is the phenotypic space), and fitness assignment takes place through  $f_p : \Phi_p \rightarrow \mathbb{R}$ , which maps phenotypes into fitness values. In summary, individuals in the population of genotypes are evaluated through  $f = f_p \circ f_g$ , the fitness of a genotype  $x^g$  being given by  $f(x^g) = f_p(f_g(x^g))$ . When considering an EA that searches in a continuous genotypic space, for solutions that decode into continuous phenotypes, we then have that  $\Phi_g \subset \mathbb{R}^m$ , and  $\Phi_p \subset \mathbb{R}^n$ , where  $m$  and  $n$  stand, respectively, for the dimensionalities of the considered genotypic and phenotypic spaces. A genotype-phenotype map is then a transformation  $f_g : \mathbb{R}^m \rightarrow \mathbb{R}^n$ .

Let  $\mathcal{N}$  be a fully connected, feedforward neural network with  $l$  layers and  $d^k$  neurons on its  $k$ -th layer ( $k = 1..l$ ). If  $\mathcal{L}^k$  is the vector representing the states of the  $d^k$  neurons in its  $k$ -th layer, then the network’s output can be determined through

$$\mathcal{L}_i^k = \sigma(b_i^k + \sum_{j=1}^{d^{k-1}} w_{ij}^k \mathcal{L}_j^{k-1}), i = 1..d^k,$$

where  $b^k$  represents the biases for neurons in the  $k$ -th layer, and  $w_i^k$  the weights given to signals neuron  $\mathcal{L}_i^k$  gets from neurons in the preceding layer. A sigmoidal activation function  $\sigma(y) = 1/(1 + e^{-y})$  is used throughout this paper. The network’s output,  $\mathcal{L}^l$ , is then uniquely determined through  $b$ ,  $w$ , and  $\mathcal{L}^1$ , the input vector fed to its input layer.

Without loss of generality, in the sequel we assume that the given phenotype space is an  $n$  dimensional hypercube. (If needed, the interval  $[0, 1]$  can be mapped with a trivial linear transformation to the actual user specified lower and upper bounds for each variable under optimization.) Using a neural network as a G-P map, we then obtain a setup where the number of output neurons  $d^l = n$  and the mapping itself is  $f_g : [0, 1]^{d^1} \rightarrow [0, 1]^{d^l}$ . To specify a given G-P mapping network we will use the notation  $\mathcal{N}(m_p, m_a)$ , where  $m_p$  and  $m_a$  are the map parameters<sup>1</sup> and map arguments, defined as follows. The vector  $m_p \in [-1, 1]^d$  contains the definition of all weights and biases in the network, while the vector  $m_a$  designates the input vector fed into the network. With this notation, we obtain a formal framework where genotypes are map arguments to the neural net and the representation is  $f_g = \mathcal{N}(m_p, \cdot)$ . Given a genotype  $x^g \in [0, 1]^{d^1}$ , the corresponding phenotype is  $f_g(x^g) = \mathcal{N}(m_p, x^g) \in [0, 1]^{d^l}$ .

As shorthand for a considered NN architecture, we will use notation such as 30-5-10, to indicate a fully connected feedforward neural network with  $l = 3$

<sup>1</sup> “Map parameters” named by analogy with the strategy parameters (e.g., standard deviations of a Gaussian mutation) traditionally used in Evolution Strategies.

layers, having  $d^1 = 30$  neurons in its input layer,  $d^2 = 5$  neurons in the hidden layer, and  $d^3 = 10$  neurons in the output layer ( $\Phi_g = [0, 1]^{30}$ ,  $\Phi_p \subset \mathbb{R}^{10}$ ).

## 4 Expressiveness

The expressiveness challenge facing a representation of G-P maps, is that of ensuring the “language” used to represent representations supports the specification of widely distinct transformations between both spaces. Given our use of neural networks to represent G-P maps, and the knowledge that their expressiveness needs to be traded-off with their learnability, we will then address the following research question:

- what is the expressiveness retained by small to medium sized neural networks?

We will focus our analysis on two often-studied representation properties: *locality* and *redundancy* [9,1]. A representation’s locality [9, Sec. 3.3] describes how well neighboring genotypes correspond to neighboring phenotypes. In a representation with perfect (high) locality, all neighboring genotypes correspond to neighboring phenotypes. Theoretical and experimental evidence [9,1] support the view that high locality representations are important for efficient evolutionary search, as they do not modify the complexity of the problems they are used for. A redundant encoding, as the name implies, provides multiple ways for a phenotype to be encoded in the genotype. In [9, Sec. 3.1] different kinds of redundancies are identified, the advantages and disadvantages of each one being then subjected to theoretical and experimental study.

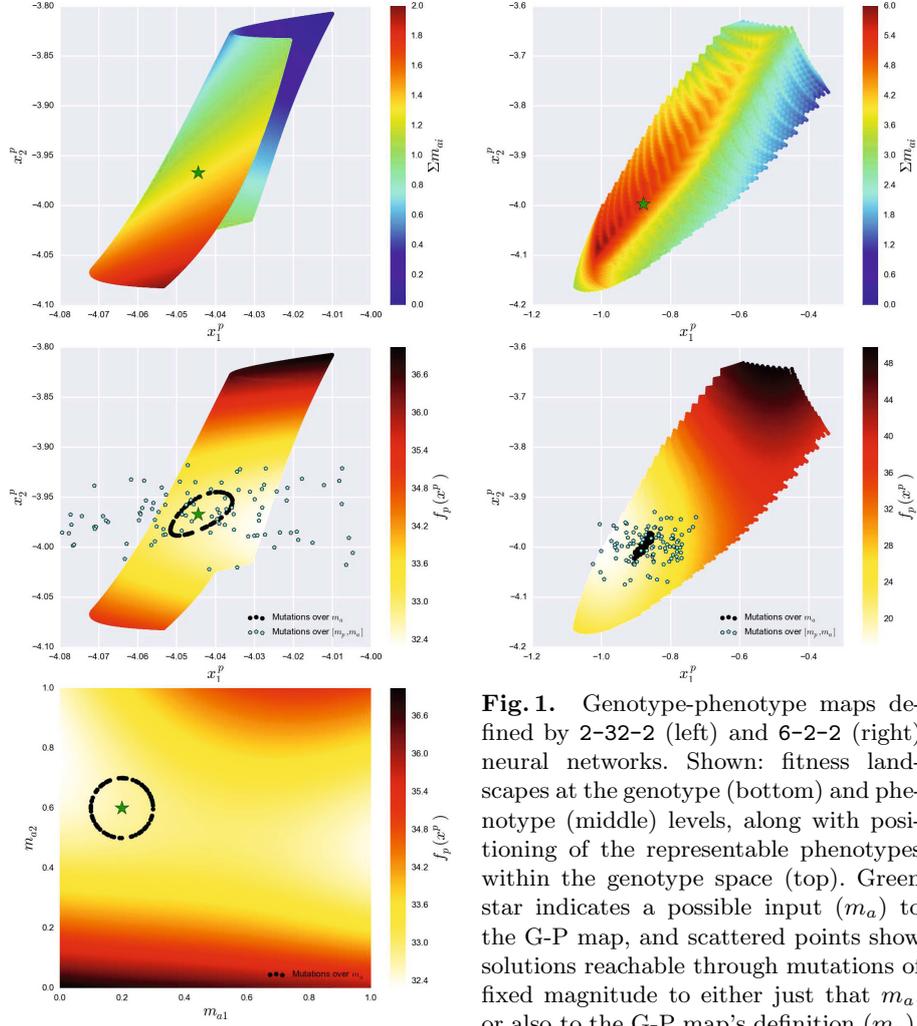
### 4.1 Map Characterization

We characterize here the expressive power of different NN architectures, in terms of the locality and redundancy of the G-P maps they can define. We conduct our analysis over the G-P map design space, by sampling random NN configurations within given architectures.

**Setup.** The G-P map design space is explored by randomly sampling (with uniform probability) NN weights and biases, in the range  $[-1, 1]$ , thus providing the definition of map parameters,  $m_p$ . We follow by generating a large number of map arguments,  $m_a$  (10000, to be precise), in the range  $[0, 1]$ , according to a quasi-random distribution. Sobol sequences are used to sample the genotype space ( $m_a \in \Phi_g$ ), so as to obtain a more evenly spread coverage. The  $m_a$  scattered in the genotype space are subsequently mapped into the phenotype space. The Euclidean metric is used to measure distances between points within both the genotype and phenotype spaces.

The analysis conducted here is fitness function independent, but for illustration purposes (Figure 1), and for defining the phenotypic space in which distances will be measured, we consider the well known Rastrigin function<sup>2</sup>.

<sup>2</sup> Rastrigin:  $f_p(x_1^p, \dots, x_n^p) = 10n + \sum_{i=1}^n [(x_i^p)^2 - 10 \cos(2\pi x_i^p)]$ ,  $\Phi_p = [-5.12, 5.12]^n$ .



**Fig. 1.** Genotype-phenotype maps defined by 2-32-2 (left) and 6-2-2 (right) neural networks. Shown: fitness landscapes at the genotype (bottom) and phenotype (middle) levels, along with positioning of the representable phenotypes within the genotype space (top). Green star indicates a possible input ( $m_a$ ) to the G-P map, and scattered points show solutions reachable through mutations of fixed magnitude to either just that  $m_a$ , or also to the G-P map's definition ( $m_p$ ).

*Measuring locality.* We characterize the locality of representations definable by a given neural network architecture, by randomly sampling the space of possible network configurations ( $m_p$ ) in that architecture. A sample of 1000 points is taken, out of the 10000  $m_a$  given by the Sobol sequence (mentioned above), and a mutated version generated. A mutation is always a random point along the surface of the hypersphere centered on the original genotype, and having a radius equal to 1% the maximum possible distance in the  $\Phi_g$  hypercube. The mutated  $m_a$  is mapped into the phenotype space, and its distance there to the original point's phenotype measured. Given we wish to consider phenotype spaces having distinct numbers of dimensions, and importantly, given the fact that each different G-P map encodes a different subset of the phenotype space, it becomes important to normalize phenotypic distances, in a way that makes

them comparable. To that end, we identify the hyperrectangle that encloses all the phenotypes identified in the initial scatter of 10000 points, and use the maximum possible distance value there to normalize phenotype distances.

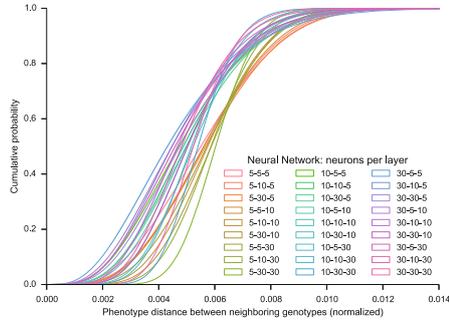
*Measuring redundancy.* To characterize a representation’s redundancy, we will want to relate phenotypes to the distinct genotypes capable of encoding them. In a representation with high locality, the extent to which dissimilar genotypes express similar phenotypes, provides an indication of its redundancy.

Given a randomly generated NN, and the set of 10000 solutions sampled in genotype space through a Sobol sequence (as previously described), we randomly select 200 of the obtained phenotypes. For each, we perform a k-Nearest Neighbor search, in phenotype space, for its 5 closest phenotypes. Having all our phenotypes been obtained from known genotypes, we are then able to map those nearest neighbors back to the genotypes that led to their expression. One data point in our analysis is then composed of the distances in both genotype and phenotype spaces, between a queried solution, and one of those neighbors. Analysis of one NN is in this setup then given by  $200 \cdot 5$  data points. To allow for comparisons of distances across NN architectures, phenotype distances are normalized over the maximum possible distance within the hyperrectangle enclosing the space of 10000 phenotypes obtained through the Sobol process. Additionally, also distances in genotype space are normalized, in this case over the maximum possible distance within the  $d^1$ -dimensional hypercube.

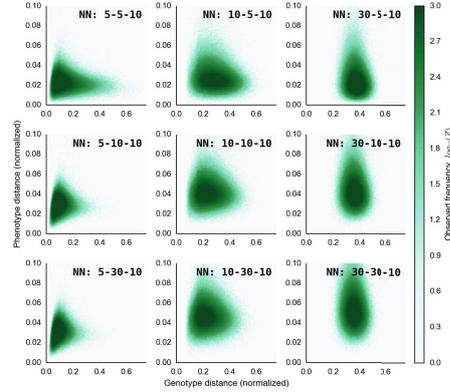
**Results.** Figure 1 shows the mappings defined by two randomly defined G-P maps (plots obtained by spreading an evenly spaced grid of  $10^6$   $m_a$  in genotype space, mapping them to phenotype space, and evaluating them). Top panels show the phenotypes expressible by each G-P map, color coded based on the  $m_a$  that led to their expression. The continuous color gradients observed in each case show that nearby genotypes are being decoded into nearby phenotypes, and are therefore evidence of two high locality representations (low locality would have resulted in a randomization of the colors present in small neighborhoods). In Figure 1 (top left) we see a genotype space that folds on itself as it gets mapped onto the phenotype space, leading *some* of its expressible phenotypes to become redundantly representable, as seen in the bottom and middle panels.

Figure 2 characterizes the locality of representations definable by different NN architectures. Each of the shown distributions was obtained by analyzing 1000 randomly generated G-P maps having that architecture, and thus represents a total of  $10^6$  measured phenotype distances. We consistently observe high locality representations resulting from *all* studied NN architectures: a mutation step of 1% the maximum possible distance in genotype space is in all cases expected to take us across a distance in phenotype space of at most  $\sim 1\%$  the maximum possible distance among phenotypes representable by the considered G-P map.

Figure 3 characterizes the redundancy of representations definable by different NN architectures. Each of the shown bivariate distributions was obtained by analyzing 1000 randomly generated G-P maps having that architecture, and thus



**Fig. 2.** Locality of representations expressible by different sized neural networks. Shown: empirical cumulative distribution functions of distances in phenotype space between pairs of neighboring genotypes.



**Fig. 3.** Redundancy of representations expressible by different sized neural networks. Shown: multivariate distribution relating a phenotype’s distance to one of its  $k$ -nearest neighbors, with the distance at which the pair lies in genotype space.

represents a total of  $10^6$  measured phenotype and genotype distances. We clearly see the number of neurons in the input layer as the primary factor determining the expected degree of redundancy in NNs sampled according to a given architecture. Nearest neighbors, which in all cases lie at approximately the same distance in phenotype space (on average, 2 to 6% the span of possible distances in  $\Phi_p$ ), turn out to be representable through genotypes at distances between themselves going on average from roughly 10, to 25, and 40% the span of possible distances in  $\Phi_g$ , as the number of input neurons grows from 5 to 30.

**Discussion.** The view emerging from these results is that NN-based G-P maps, as defined in Section 3, tend to generate high-locality representations, with a tuneable degree of expected redundancy.

Given a random G-P map, its genotype space will most likely decode into a limited region of the full phenotype space (as seen in Figure 1). This could be problematic if G-P maps were to be defined off-line, for unknown search spaces, where the risk of being unable to express the global optimum would be considerable. In an on-line adaptation scenario, however, the G-P map is instead at every point of the search devoting its resources to learning a representation of a momentarily relevant portion of phenotype space, while retaining the power to adapt itself, towards expression of newly identified superior phenotypic regions.

## 5 Learnability

In this section we want to establish the existence of a learning mechanism, that can change the NN-based G-P map during the run of a given EA, in such a way that the EA performance (measured by solution quality) is improved.

When designing a suitable learning mechanism we face a couple of principal decisions, namely: which learning mechanism to use, and how to measure G-P maps' quality, so as to steer the learning mechanism towards superior ones? We answer these questions by employing a self-adaptation scheme, whereby the parameter vector that specifies a G-P map,  $m_p$ , is added to the chromosome and co-evolves with the solution ( $m_a$ ). By the very nature of self-adaptation, this option elegantly solves the second problem. In particular, we do not need to specify an explicit quality measure for the G-P map. Instead, the G-P map is evaluated implicitly: it is good if it leads to good solutions.

In neuroevolution, recombination-based EAs are known to face some difficulties when optimizing NNs. This is known as the competing conventions problem [5,11]. To avoid this problem, we decide to use a mutation-only EA.

Having made these choices, the issue of learnability addressed in this section can now be phrased as follows:

- can a self-adaptive mechanism within a mutation-only EA effectively learn useful G-P mappings that lead to increased EA performance?

## 5.1 Experimental Evaluation

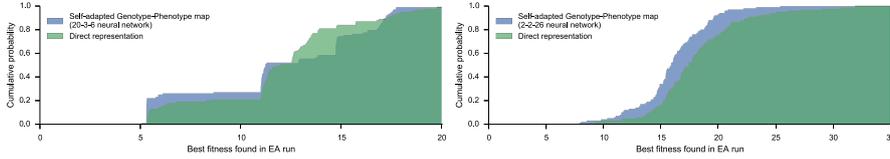
We experimentally evaluate here the performance achieved by EA setups that self-adapt G-P maps, through comparison with identical optimizer setups that work instead directly over the phenotype space.

**Setup.** Our experimental validation compares, for the same problem, optimization using a direct representation  $x^g = x^p$ , against optimization using an indirect representation, where a NN-based G-P map is self-adapted in the chromosome, together with its input,  $x^g = [m_p, m_a]$ .

We evaluate the different setups by searching for solutions to the Cassini 1 and Messenger\_full problems. These are difficult, real-world problems, of spacecraft interplanetary trajectory design. Their full specification can be found online, in the GTOP Database [10]<sup>3</sup>. In the indirect representation experiments we demonstrate the usage of distinct neural network architectures: in Cassini 1 we ask the optimizer to learn and exploit highly redundant encodings of the phenotype space, by using 20-3-6 G-P maps; In Messenger\_full we ask it instead to learn minimally redundant, lower dimensional projections of the problem's 26-dimensional phenotype space, by using 2-2-26 G-P maps.

We make use of the Improved Fast Evolutionary Programming (IFEP) algorithm introduced in [12], extended with the success rate based dynamic lower bound adaptation (DLB1) described in [6]. The optimizer was tuned as follows: population size  $\mu = 25$ , and tournament size  $q = 2$ . The strategy parameters in individuals' chromosomes, which parameterize their mutation operators, were initialized to values of  $\eta = 0.03$ , and had initial lower bounds of  $\eta_- = 0.015$ , adapted by DLB1, every 5 generations, using a reference success rate of  $A = 0.3$ .

<sup>3</sup> <http://www.esa.int/gsp/ACT/inf/projects/gtop/gtop.html>



**Fig. 4.** Comparison of direct and indirect representations on the Cassini 1 (left) and Messenger\_full (right) problems. Shown: empirical cumulative distribution functions of the best fitness value reached in an EA run.

In each IFEP run, evolution proceeded for a total of 5000 generations. IFEP has each parent generating two mutated offspring, one according to a Gaussian, and another according to a Cauchy distribution<sup>4</sup>. As such, 50 solutions were generated per generation, leading to a total of 250000 fitness evaluations per run. The indirect representation setups used bounds of  $m_{pi} \in [-1, 1]$ , and  $m_{ai} \in [0, 1]$ . For increased fairness in the comparison between direct and indirect representations, chromosomes were in both cases normalized at the optimizer level, into the unit hypercube,  $x^g \in [0, 1]^d$ , and scaled back at decoding and evaluation time.

**Results.** Figure 4 presents the distributions of best found solutions’ fitness values, in 100 independent EA runs performed under each optimization setup.

Both Cassini 1 and Messenger\_full are minimization problems. In Cassini 1 we see a median fitness of 11.9 being found with a direct representation, and 11.2 with an indirect one. Peak performance was however lower on the indirect representation runs: 5.3, against 5.1 for the direct representation. In Messenger\_full the indirect representation improved median performance from 17.4, to 15.9, as well as peak performance (7.9, against 8.8). We see also in it a considerable improvement to worst case performance (from 32.0 to 25.4).

**Analysis.** Extending chromosomes with the definition of their own G-P maps, naturally places a significant burden on top of the optimization process: Cassini 1 goes from being a 6-dimensional optimization problem in the direct representation case, to a 107-dimensional one when simultaneously learning a 20-3-6 G-P map. Similarly, the Messenger\_full problem goes from 26 to 86 dimensions when adding a 2-2-26 G-P map. Still, as seen in Figure 4, EA performance is matched, or even surpassed, by the G-P maps’ addition.

Back in Section 4 we saw in Figure 1 (middle panels), regarding mutation over vectors containing  $[m_p, m_a]$ , that it is possible to conduct a robust search simultaneously over the G-P map’s definition, and its inputs: mutated offspring tend to encode phenotypes in the vicinity of those of their parents. The results reported in this section show that such variation, in an evolutionary setting,

<sup>4</sup> The lognormal self-adaptation of strategy parameters employed by EP is not used in the indirect representation setups to adapt the (also self-adapted) map parameters. Instead, their variation takes place through the Gaussian (or Cauchy) mutation.

indeed allows for the self-adaptation of G-P maps to take place concurrently with the search for problem solutions.

## 6 Conclusion

We investigated the usage of neural networks as a meta-representation, suited to the encoding of genotype-phenotype maps for arbitrary pairings of fitness landscapes and metaheuristics that are to search on them.

Small to moderately sized feedforward neural networks were found to define, on average, high locality representations (where structure of the phenotypic fitness landscape is locally preserved in the genotype space), and having a degree of redundancy tuneable through the number of neurons in the input layer.

An exploration into the feasibility of evolving genotype-phenotype maps, concurrently with the problem solution, showed this to be a viable approach.

**Acknowledgements.** Luís F. Simões was supported by FCT (Ministério da Ciência e Tecnologia) Fellowship SFRH/BD/84381/2012.

## References

1. Correia, M.B.: A study of redundancy and neutrality in evolutionary optimization. *Evolutionary Computation* 21(3), 413–443 (2013)
2. De Jong, K.: Parameter Setting in EAs: a 30 Year Perspective. In: Lobo, et al. (eds.) [8], pp. 1–18
3. Ebner, M., Shackleton, M., Shipman, R.: How neutral networks influence evolvability. *Complexity* 7(2), 19–33 (2001)
4. Eiben, A.E., Hinterding, R., Michalewicz, Z.: Parameter Control in Evolutionary Algorithms. *IEEE Transactions on Evolutionary Computation* 3(2), 124–141 (1999)
5. Floreano, D., Dürr, P., Mattiussi, C.: Neuroevolution: from architectures to learning. *Evolutionary Intelligence* 1(1), 47–62 (2008)
6. Liang, K.H., Yao, X., Newton, C.S.: Adapting Self-Adaptive Parameters in Evolutionary Algorithms. *Applied Intelligence* 15(3), 171–180 (2001)
7. Liepins, G.E., Vose, M.D.: Representational issues in genetic optimization. *Journal of Experimental & Theoretical Artificial Intelligence* 2(2), 101–115 (1990)
8. Lobo, F.G., Lima, C.F., Michalewicz, Z. (eds.): *Parameter Setting in Evolutionary Algorithms*. SCI, vol. 54. Springer, Heidelberg (2007)
9. Rothlauf, F.: *Representations for Genetic and Evolutionary Algorithms*, 2nd edn. Springer, Heidelberg (2006)
10. Vinkó, T., Izzo, D.: Global optimisation heuristics and test problems for preliminary spacecraft trajectory design. ACT technical report GOHTPPSTD. European Space Agency, the Advanced Concepts Team (September 2008)
11. Yao, X.: Evolving artificial neural networks. *Proceedings of the IEEE* 87(9), 1423–1447 (1999)
12. Yao, X., Liu, Y., Lin, G.: Evolutionary Programming Made Faster. *IEEE Transactions on Evolutionary Computation* 3(2), 82–102 (1999)