Genetic Programming and Evolvable Machines (2020) 21:375–397
https://doi.org/10.1007/s10710-020-09379-0

A network perspective on genotype–phenotype mapping in genetic programming

Ting Hu[1,2] · **Marco Tomassini**[3] · **Wolfgang Banzhaf**[4]

Received: 14 October 2019 / Revised: 3 January 2020 / Published online: 29 January 2020

Abstract
Genotype–phenotype mapping plays an essential role in the design of an evolutionary algorithm. Variation occurs at the genotypic level but fitness is evaluated at the phenotypic level, therefore, this mapping determines if and how variations are effectively translated into quality improvements. In evolutionary algorithms, this mapping has often been observed as highly redundant, i.e., multiple genotypes can map to the same phenotype, as well as heterogeneous, i.e., some phenotypes are represented by a large number of genotypes while some phenotypes only have few. We numerically study the redundant genotype–phenotype mapping of a simple Boolean linear genetic programming system and quantify the mutational connections among phenotypes using tools of complex network analysis. The analysis yields several interesting statistics of the phenotype network. We show the evidence and provide explanations for the observation that some phenotypes are much more difficult to find as the target of a search than others. Our study provides a quantitative analysis framework to better understand the genotype–phenotype map, and the results may be utilized to inspire algorithm design that allows the search of a difficult target to be more effective.

Keywords Evolvability · Genotype–phenotype map · Networks · Neutrality · Redundancy · Robustness

1 Introduction

The genotype–phenotype map as a key aspect of natural evolution came to prominence in biology in the early 1990s. Alberch raised the issue in his attempt to create a framework for unifying evolutionary and developmental biology [1]. In particular, he raised attention to the fact that this map is much more complex and dynamic than previous understanding was willing to accept. In the intervening years, these aspects

✉ Ting Hu
ting.hu@queensu.ca

Extended author information available on the last page of the article

of complexity and dynamics have garnered substantial attention, couching the map between genotype and phenotype in terms of a network of regulatory elements that can control its different functions (see Davidson for an in-depth study [14]). In this view, genes form a network of regulatory entities where changes in the expression of one gene can be compensated by changes in other genes providing stability for the phenotype. The evolutionary process then molds the expression dynamics of this network by genetic variation which percolates to the corresponding phenotypes via the gene regulatory network. As Kirschner and Gerhard pointed out, this dynamic "buffering" layer for variation is a key aspect of the evolvability observed in natural evolutionary systems [27, 28].

In artificial evolutionary systems, the value of using a "buffer" layer between genotypes and phenotypes has also steadily gained prominence in research on evolutionary computation and artificial life. Indirect encodings [9], developmental processes [30, 34], epigenetic interactions [17] and regulatory networks [13] have been studied as ways to improve the evolvability of these artificial evolutionary systems.

Studying the context of one of the most simple genotype–phenotype maps available in nature embodied by RNA folding, Schuster, Fontana and others [44] have considered another network, namely that traversed during the evolutionary search process, where different genotypes form a complex network, with genotypes as nodes, and variation operator connecting those nodes via mutation. The phenotype network corresponding to this network is another network that can be studied at the same time. Here, the previously mentioned genotype–phenotype map has been abstracted into a physico-chemical model for the folding of linear RNA sequences into two-dimensional RNA folding shapes. Subsequent work has shown particular features of this map which are relevant for our study below [11, 41]. Following earlier work by Kauffman et al. [25], search on fitness landscapes has been studied using adaptive or random walks on these complex networks.

Here we adopt this view of search as an adaptive walk in the complex network provided by the representation of our problem, enabled by the operators working on this representation, and guided by the fitness attached to each of these representations. Notably, in evolutionary algorithms, the quality of a candidate solution is assessed based on its phenotype, i.e., how well the phenotype is able to produce a desired outcome judged by a fitness measure. Yet, the actual evolutionary search occurs in genotype space, where the encoding of candidate solutions is modified by mutation or recombination operations. Thus, how genotypes are mapped to phenotypes will substantially influence the search effectiveness of an evolutionary algorithm [15, 26].

Redundant genotype–phenotype maps are common in both natural [8, 43] and computational evolution [4, 22, 29, 42, 46], where multiple genotypes can map to the same phenotype. Such a *redundancy* is often unevenly distributed among phenotypes, where some phenotypes are over-represented, i.e. represented by many genotypes, and some are under-represented, i.e. represented by only a few [23, 42]. When the target phenotype is under-represented, its evolutionary search is often more difficult than having a genotypically over-represented target. This is intuitive since it can be more difficult to find one of the few genotypes that map to an under-represented target phenotype.

If the genotype-to-phenotype mapping is redundant, a mutation to a genotype may not change the phenotype it encodes, a phenomenon defined as *neutrality* [36, 48], and such mutations are called neutral mutations [18, 19, 21, 47]. Neutrality is facilitated by redundancy, but not guaranteed. For instance, there are cases where genotypes map to the same phenotype but are not mutationally connected, i.e., one genotype cannot be reached from the other through single point mutations, thus mutations that need to occur on the way from one to the other will need to alter the phenotype.

In contrast to neutral mutations, non-neutral mutations connect genotypes of distinct phenotypes. Such non-neutral mutational connections among phenotypes might also be heterogeneous [23, 38], i.e., a phenotype may not have the same likelihood of mutating to other phenotypes and thus may tend to "prefer" some phenotypes over others. The difficulty of finding a target phenotype is thus influenced not only by its genotypic abundance, but also by how mutational connections are distributed among different phenotypes.

In this article, we quantitatively measure the genotypic redundancy of phenotypes and the mutational connections among them, and take a network approach to analyze how these properties correlate with the difficulty of finding a target phenotype. This extends our research previously published in the EuroGP'19 conference proceedings [24], We use a linear genetic programming (LGP) algorithm for Boolean function search, and numerically characterize its genotype, phenotype, and fitness space. Using random sampling and random walks, we construct a phenotype network to depict the mutational connections among different phenotypes. Once a specific target phenotype is chosen, this changes the connectivity of the phenotype network since only non-deleterious mutations, i.e. mutations that do not decrease fitness, are allowed. We show that such changes can significantly influence the difficulty of finding a target.

The paper is organized as follows: Sect. 2 reviews our Boolean LGP system and defines basic notions for studying the complex networks. We then discuss evolvability and robustness as the two notions of major importance in this study (Sect. 3). Section 4 reports the results of our examination of genotype and phenotype networks and Sect. 5 discusses the consequences of our network view of the evolutionary search process.

2 Methods

2.1 A Boolean linear genetic programming algorithm

We use a linear genetic programming (LGP) algorithm for our empirical analysis. LGP is a branch of genetic programming and employs a sequential representation of computer programs to encode an evolutionary individual [7]. Such an LGP program is often comprised of a set of imperative instructions, which are executed sequentially. Registers are used to either read input variables (input registers) or to enable computational capacity (calculation register). One or more registers can be

designated as the output register(s) such that the final stored value(s) after the program is executed will be the program's output.

In this study, we use an LGP algorithm for a three-input, one-output Boolean function modeling application. Each instruction has one return, two operands and one Boolean operator. The operator set has four Boolean functions {AND, OR, NAND, NOR}, any of which can be selected as the operator for an instruction. Three registers $\mathtt{R}_1$, $\mathtt{R}_2$, and $\mathtt{R}_3$ receive the three Boolean inputs, and are write-protected in an LGP program. That is, they can only be used as an operand in an instruction. Registers $\mathtt{R}_0$ and $\mathtt{R}_4$ are calculation registers, and can be used as either a return or an operand. Register $\mathtt{R}_0$ is also the designated output register, and the Boolean value stored in $\mathtt{R}_0$ after an LGP program's execution will be the final output of the program. All calculation registers are initialized as FALSE before execution of a program. An LGP program can have any number of instructions, however, for the ease of simulation in this study, we determine that an LGP program has a fixed length of six instructions. An example LGP program is given as follows.

$$
\begin{aligned}
I_1 &: \mathtt{R}_4 = \mathtt{R}_2\mathtt{AND} \quad \mathtt{R}_3 \\
I_2 &: \mathtt{R}_0 = \mathtt{R}_1\mathtt{OR} \quad \mathtt{R}_4 \\
I_3 &: \mathtt{R}_4 = \mathtt{R}_4\mathtt{NAND} \quad \mathtt{R}_0 \\
I_4 &: \mathtt{R}_4 = \mathtt{R}_3\mathtt{AND} \quad \mathtt{R}_2 \\
I_5 &: \mathtt{R}_0 = \mathtt{R}_1\mathtt{NOR} \quad \mathtt{R}_1 \\
I_6 &: \mathtt{R}_0 = \mathtt{R}_3\mathtt{AND} \quad \mathtt{R}_0
\end{aligned}
$$

2.2 Genotype, phenotype, and fitness

The *genotype* in our evolutionary algorithm is a unique LGP program. Since we have a finite set of registers and operators, as well as a fixed length for all programs, the genotype space is finite. Specifically, considering an instruction, either of the two registers can be chosen as the return, and any two of the five registers can be chosen as the operands, and the operator is picked from the set of four possible Boolean functions. Thus, there are $2 \times 5 \times 5 \times 4 = 200$ unique instructions. Given the fixed length of six instructions for all LGP programs, we have a total number of $200^6 = 6.4 \times 10^{13}$ possible different programs. Although finite, the genotype space is enormous and is not amenable to exhaustive enumeration. Therefore, we conduct a simulation by randomly generating one billion LGP programs (≈ 15.6 ppm = 0.00156% of the genotype space) to approximate the genotype space.

The *phenotype* in our evolutionary algorithm is a Boolean relationship that maps three inputs to one output, represented by an LGP program, i.e., $f : \mathbf{B}^3 \rightarrow \mathbf{B}$, where $\mathbf{B} = \{\mathtt{TRUE}, \mathtt{FALSE}\}$. There are thus a total of $2^{2^3} = 256$ possible Boolean relationships. Having 6.4×10^{13} genotypes to encode 256 phenotypes, our LGP algorithm must have a highly redundant genotype-to-phenotype mapping. We define the *genotypic redundancy* of a phenotype as the total number of genotypes that map to it.

The *fitness* of an LGP program is dependent on the target Boolean relationship, and it is defined as the dissimilarity of the presented and the target

Boolean relationships. Given three inputs, there are $2^3 = 8$ combinations of Boolean inputs. The Boolean relationship encoded by an LGP program can be seen as a 8-bit string representing the outputs that correspond to all 8 possible combinations of inputs. Fitness is defined as the Hamming distance of this 8-bit output and the target output. For instance, if the target relationship is $f(\mathtt{R}_1, \mathtt{R}_2, \mathtt{R}_3) = \mathtt{R}_1\ \mathtt{AND}\ \mathtt{R}_2\ \mathtt{AND}\ \mathtt{R}_3$, represented by the 8-bit output string of 00000001, the fitness of an LGP program encoding the FALSE relationship, i.e., 00000000, is 1. Fitness falls into the range of [0, 8] where 0 is the perfect fitness and 8 is the worst, and is to be minimized.

2.3 Phenotype network

Point mutations to genotypes may change the encoded phenotypes from one to another. In the context of our LGP algorithm, a point mutation is to replace any one of the four elements, i.e., return, two operands, and operator, of an instruction in an LGP program. The mutational connections among pairs of phenotypes can be modeled using a *phenotype network*. In such a network, each node represents one of the 256 phenotypes that can be possibly encoded by the LGP genotypes. Two nodes (phenotypes) are directly connected by an edge if there exist at least one pair of underlying genotypes, one from each phenotype, that can be transitioned from one to the other through a single point mutation.

Since it is infeasible to enumerate all possible genotypes, sampling the mutational connections among phenotypes is also necessary. We assemble one million randomly generated LGP programs and allow each to take a 1000-step random walk in genotype space. All the phenotypes each random walker encountered are recorded in order to estimate the number of point mutations that can transition one phenotype to another. This random walk simulation yields a *undirected, weighted* phenotype network, where the weight of an edge is proportional to the number of sampled point mutations that can change the genotypes of one phenotype to that of the other phenotype.

Assigning a fitness to each phenotype and preventing deleterious mutations changes the reversible nature of point mutations, the weighted phenotype network into a *directed* graph and transforms the random walk on the network into an adaptive walk. We study adaptive walks by picking two target phenotypes with a considerable difference in their genotypic redundancies, given the consideration that whether a target phenotype is over- or under-represented by genotypic encodings may influence the difficulty level of finding that target [42] in an adaptive walk. The first target is phenotype 11110000 (decimal 240) which has a genotypic redundancy of 46,729,920, i.e., 4.673% of the one billion sampled genotypes. The second target is phenotype 10110100 (decimal 180) with only a genotypic redundancy of 86. Setting such different targets will render the corresponding directed, weighted phenotype networks different. Thus, we investigate a variety of network properties to compare these two networks.

2.4 Complex network analysis

Since we need a few concepts and methods from the field of network science [5, 35], we here collect some useful definitions to be referenced and used later.

Strength This term refers to the generalization of the vertex degree to weighted networks. It is defined as the sum s_i of weights of the edges from node i to its neighbors $\mathcal{N}(i)$,

$$s_i = \sum_{j \in \mathcal{N}(i)} w_{ij},$$

where w_{ij} is the weight of the edge connecting nodes i and j.

Disparity A given value of a node's strength can be obtained with very different values of edge weights. The contributing weights could be of about the same size or they could be very different. To measure the degree of heterogeneity of a node's edges *disparity* can be used. It is defined as follows:

$$Y_2(i) = \sum_{j \in \mathcal{N}(i)} \left(\frac{w_{ij}}{s_i} \right)^2.$$

If all the connections are of the same order then Y_2 is small and of order $1/k$ where k is the vertex degree. On the other hand, if there is a small number of high weight connections Y_2 is larger and may approach unity.

Centralities Network centrality measures are intended to characterize the importance of a node or an edge in a network and were first considered in social network contexts. There are several established centrality measures (see, e.g., [35]); here we use two of them: *eigenvector centrality* and *PageRank centrality*.

Eigenvector centrality tries to capture the idea that the more central the neighbors of a node are, the more central the node itself is. It can be expressed in terms of eigenvector solutions of eigenvalue systems based on the graph adjacency matrix, hence the name.

PageRank centrality is based on Google's PageRank algorithm [39], a key component of their search engine, which attributes to a web page a rank essentially given by the frequency with which a random walk hits the page in the long run. This frequency is higher for pages with many incoming links and provides a measure of the importance, or popularity, of a page.

Coreness The k-core of a network is its largest subgraph for which all vertex degrees are at least k. With a suitable visualization technique, knowledge of k-core decomposition of a network allows to highlight the best interconnected and the least interconnected parts in a network. This can be done by showing the k-cores as successive layers that go from highly connected at the center to less and less connected towards the periphery in a radial fashion.

Average shortest paths We use weighted and unweighted shortest paths between pairs of vertices. The average values of all two-point shortest paths in a graph give an idea of the typical distances between nodes.

Clustering coefficient The clustering coefficient $C(i)$ of a node i is defined as the ratio between the e edges that actually exist between the k neighbors of i and the number of possible edges between these nodes:

$$C(i) = \frac{e}{\binom{k}{2}} = \frac{2e}{k(k-1)}.$$

The clustering coefficient can be interpreted intuitively as the likelihood that two of node i's neighbors are also neighbors. The *average clustering coefficient* $\bar{C}$ is the average of $C(i)$ over all N vertices in the graph G, $i \in V(G)$: $\bar{C} = (1/N) \sum_{i=1}^{N} C(i)$.

Degree, strength, and edge weight distribution functions These discrete distributions give, respectively, the frequency of a given node degree, node strength, or edge weight in the network. These distributions are useful for evaluating whether they are, for instance, homogeneous or heterogeneous, unimodal or multimodal.

3 Evolvability and robustness

Evolvability and robustness are two important notions related to the genotype–phenotype mapping. Evolvability describes the ability of generating novel phenotypes in order to adapt to varying environments, while robustness refers to the resilience of remaining intact in response to perturbations and changes. Early work on evolvability [2, 3] in genetic programming was taken up in evolutionary biology [27, 40, 48, 50], and has since been further investigated in artificial evolutionary systems [2, 16, 21, 31] and is a prime example of "closing the feedback loop" between evolutionary computation and evolutionary biology.

The relationship of evolvability and robustness may appear antagonistic, however they have been shown correlated in both theoretical analyses [23, 49] and empirical observations [6, 33]. Robustness results from the redundant genotype to phenotype mapping, where changes to genotypes may not alter the phenotype. Thus, the evolutionary system is resilient to genetic perturbations. These neutral mutations, more importantly, provide an expansion in the genotypic space without being subject to selection pressure, and enable reaching more novel phenotypes faster when the system is actively adapting to a new environment.

Constructing the phenotype network provides a quantitative characterization of evolvability and robustness in an evolutionary algorithm. Moreover, defining evolvability and robustness at different levels of genotype, genotype, fitness, and system helps elucidating their relationships. The *robustness* of a phenotype can be inferred using its genotypic redundancy [49]. An over-represented phenotype is considered robust to mutations since it has more encoding genotypes and random mutations to them will less likely change the phenotype.

Using the framework of the phenotype network, various network-based measurements have been proposed to quantify the *evolvability* of a phenotype, including node degree [49], disparity [10], and centralities [20, 38]. These network measures capture some aspects of phenotypic evolvability, i.e., starting from a specific phenotype, how easily can fitter phenotypes be reached? Both node degree and disparity

are found incapable of predicting such an ability given that they only provide the estimation of the direct neighborhood of a node, and fail to give long-term, multi-hop predictions [23]. Centralities, on the other hand, are computed based on the importance of a node that contributes to the global connectivity of the network, and provide a better long-term estimation of reaching other phenotypes. The weighted eigenvector centrality is found to have the best prediction power estimating the evolvability of a phenotype [20]. However, there is still a considerable discrepancy between the predicted and the observed quantities, and is found to be resulted by the mutational bias led by robust genotypes. Specifically, more robust genotypes are more likely to be visited through random mutations, therefore, they have a stronger influence on directing the transition to other phenotypes [23]. Thus, future attempts on quantifying the evolvability of a phenotype need to consider mutational connections both at the phenotypic and the genotypic levels. In this study, we further investigate the properties of the phenotype networks and relate them to previous findings on evolvability and robustness.

4 Results

4.1 Sampled genotype space

When we decode the one billion randomly generated genotypes, we find that 17 of the total 256 phenotypes are never sampled. The distribution of the genotypic redundancy of the remaining 239 sampled phenotypes is highly heterogeneous, as shown in Fig. 1. Moreover, due to the symmetry of Boolean relationships we see that many phenotypes have the same genotypic redundancy.

The most over-represented phenotype is 0, i.e., `FALSE`, which has over 108 million genotypes, while phenotype 255, i.e., `TRUE`, its symmetric counterpart, is the second most abundant with over 93 million genotypes. The asymmetry in count is due to the initialization of calculation registers, including the output register $\mathtt{R}_0$, to `FALSE` in all LGP programs prior to execution. In addition to the 17 phenotypes

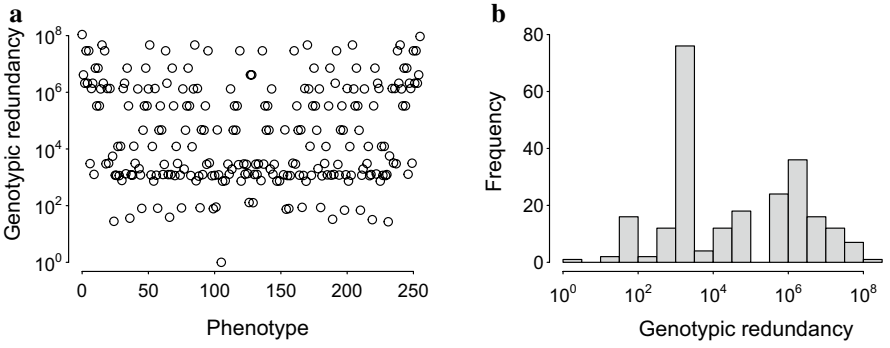

Fig. 1 **a** Scatter plot of and **b** distribution of the genotypic redundancy, in log scale, of sampled phenotypes using one billion randomly generated LGP programs

never sampled, under-represented phenotypes include 105, 231, 24, 219, 189, 36, and 66, none of which has more than 40 sampled genotypic encodings.

4.2 Undirected and weighted phenotype network

Using the assembly of one million 1000-step random walkers, the mutational connections among pairs of phenotypes can be approximated. 16 out of 256 phenotypes are never encountered, i.e., they are neither sampled in the one million randomly initialized walkers (programs), nor visited during the walks. These 16 phenotypes are isolated nodes in the phenotype network, 15 of which belong to the 17 never-sampled phenotypes discussed previously (Sect. 4.1). This also suggests that under-represented phenotypes are hard to reach by random walks.

The remaining 240 phenotypes are connected to form one component of the network, and there are 14,663 directly connected phenotype pairs, represented as edges, in the network. This yields an average node degree of 122. The network has an average shortest path 1.5 and a diameter as short as 3, which means that any pair of phenotypes can be reached from one to another by point mutations through no more than 3 hops in the phenotype network. The clustering coefficient of the network is high at 0.75; this is due to the fact that many nodes have neighbors that are themselves connected, giving rise to many closed triangles.

When we take a closer look at the distribution of mutational connections among phenotype pairs i.e., distribution of edge weights, it appears highly heterogeneous and ranges across orders of magnitude (Fig. 2a). The phenotype pair 0 and 255 has the highest number of mutational connections (5,673,803), and 6513 pairs only have one mutational connection. The distribution of these pairwise mutational connections is roughly monotonic, where the likelihood of having a greater number of connections decreases.

The phenotypes become less connected if we use an edge weight cutoff to filter out weakly connected edges. Figure 2b shows the number of nodes (excluding isolated nodes) in the phenotype network in relation to the edge weight filter cutoff. For

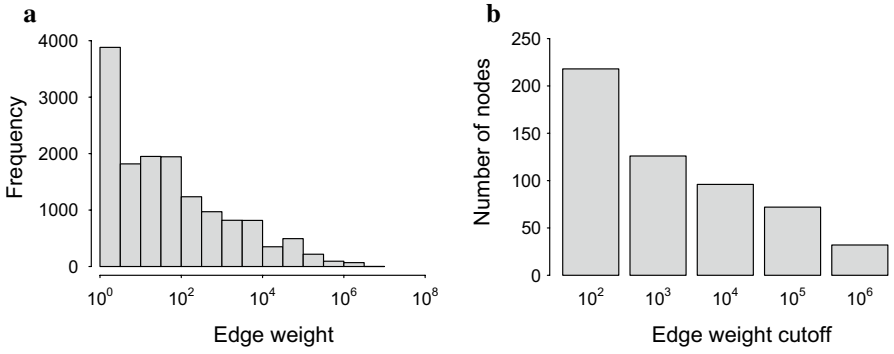

Fig. 2 **a** Distribution of the number of point mutations that can change one phenotype to another, i.e., edge weight in the phenotype network, and **b** the number of nodes (phenotypes) in the network using an edge weight filter

five different edge weight cutoffs, the network remains having only one connected component, however, its size decreases as the cutoff increases. This means that nodes are being detached from the largest component individually without forming a second connected component.

Figure 3 shows a visualization of the weighted phenotype network using an edge weight filter 10^5. There are 72 nodes and 382 edges in this network. The size of a node is proportional to the genotypic redundancy of its representing phenotype, and the width of an edge is proportional to its weight. The graph layout puts nodes with higher strengths in the center, and the two most central phenotypes are `FALSE` (0) and `TRUE` (255). We see that genotypically over-represented phenotypes (larger nodes) are more connected to others than under-represented ones.

Next, we show node properties and their correlations, as well as some global structures of the phenotype network, analyzed using the network measures defined in Sect. 2.4, in the following subsections.

4.2.1 Node properties and distributions

We look at a variety of node properties, including degree, strength, disparity, coreness, eigenvector centrality, and PageRank centrality (computed using $\alpha = 1$), in the phenotype network. Figure 4 shows the distributions of these node properties. Phenotype 0 and 255 have the highest node degree of 236, and phenotypes 22 and 104 only have a degree of 2. Strength, being a generalization of degree for weighted networks, has a shape that is qualitatively similar to the degree histogram, with a bimodal distribution.

The node disparity shows that most phenotypes have a low disparity, i.e., their links tend to have similar weights, and the distribution decays quickly. The coreness distribution shows that the majority of the phenotypes form a very dense core leaving only few peripheral nodes. The weighted centralities of eigenvector and PageRank share a very similar distribution. The majority of the nodes have similar and low centrality values and only very few nodes have significant higher centrality.

4.2.2 Node property correlations

Next, we look at how node properties correlate with the genotypic redundancy of its represented phenotype (see Fig. 5). Both degree and strength are strongly correlated with genotypic redundancy with a linear-log and a log-log correlation coefficient of $r^2 = 0.9642$ ($p < 10^{-16}$) and $r^2 = 0.9981$ ($p < 10^{-16}$). This suggests that over-represented phenotypes have more and stronger mutational connections to other phenotypes.

The positive correlation between disparity and redundancy has a coefficient of $r^2 = 0.3568$ ($p < 10^{-16}$). Coreness and genotypic redundancy also have a positive correlation. Please note that the maximal coreness value observed in the network is 95. Both eigenvector and PageRank centrality are positively correlated with genotypic redundancy, while only highly redundant phenotypes (with more than 10^6 representing genotypes) have greater centrality values than the rest.

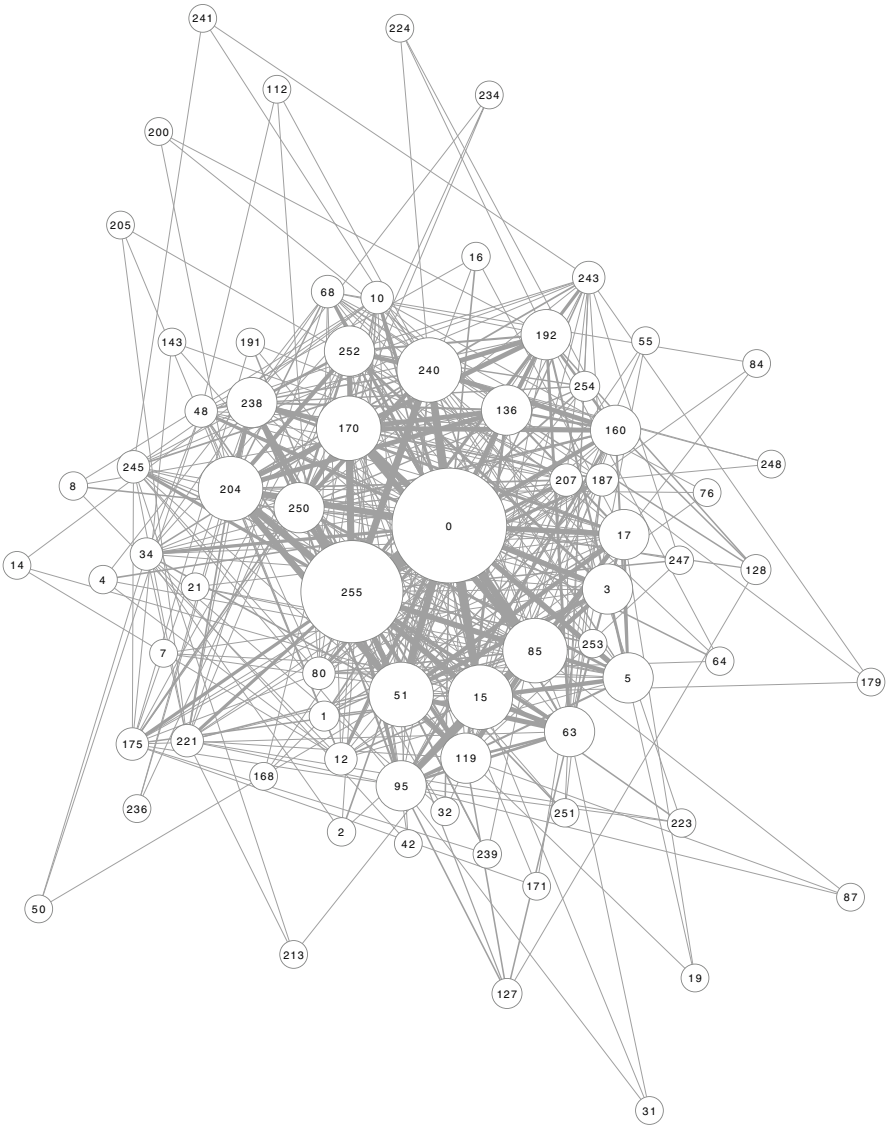

Fig. 3 The undirected and weighted phenotype network of the LGP Boolean algorithm using an edge weight filter $> 10^5$. The network includes 72 nodes (phenotypes) and 382 edges (mutational connections), resulting in an average node degree of 10.6. A phenotype is named using the decimal value of its 8-bit binary string. Node size is proportional to the genotypic redundancy of a phenotype, and edge width is proportional to the number of mutational connections between two phenotypes. The graph is visualized using software Cytoscape [45]

Recall that a phenotype with a greater number of underlying genotypes is considered more robust. Degree, strength, disparity, and centralities have been proposed in the literature as possible metrics to quantify evolvability. We find positive

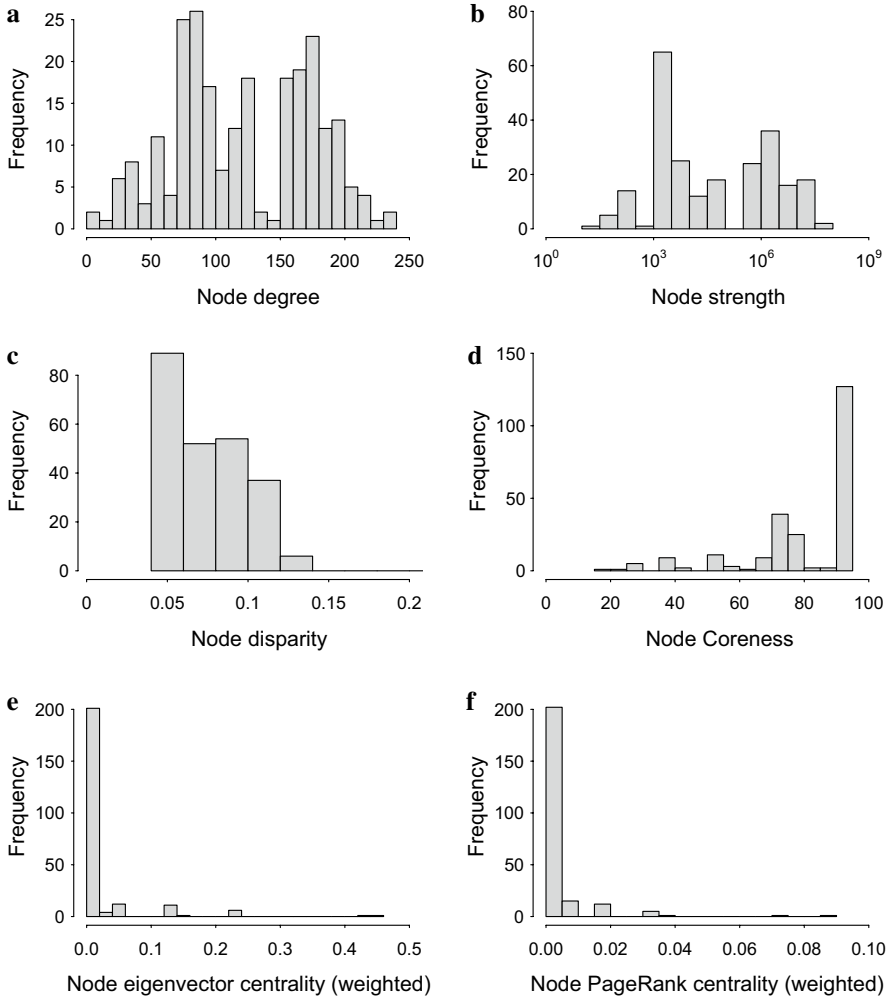

Fig. 4 Distribution of node **a** degree, **b** strength, **c** disparity, **d** coreness, **e** eigenvector centrality, and **f** PageRank centrality in the undirected weighted phenotype network

correlations of all these network measures with genotypic redundancy. Our results suggest that more robust phenotypes are also more evolvable, a correlation that concurs with observations from evolutionary biology.

4.2.3 Communities

Communities in a complex network can be loosely defined as clusters of nodes that are more strongly linked among themselves than with the rest of the network. A precise and unique definition cannot be given, which makes community detection a hard and somewhat ill-defined task for which heuristic methods must be used.

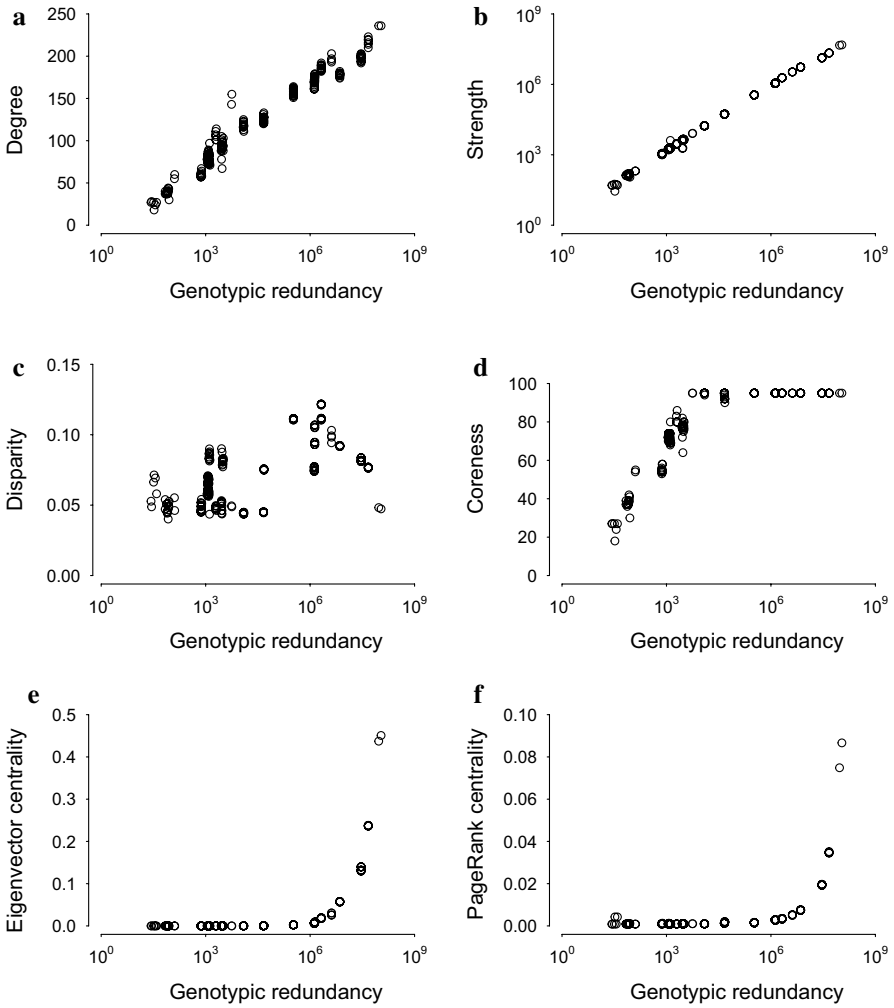

Fig. 5 Node **a** degree, **b** strength, **c** disparity, **d** coreness, **e** eigenvector centrality, and **f** PageRank centrality in relation to the genotypic redundancy of a phenotype

Nevertheless, several community detection algorithms have been proposed that work well in practice.

Here we use the methods implemented in the *igraph* R package [12], which also cover weighted networks. Before submitting our phenotype network G to a community detection algorithm some manipulations are necessary. In fact, the graph has a mean degree of about 122 which makes it a very dense network. Community detection algorithms typically do not work well, or at all, on such graphs. However, we note that edge weights in G span seven orders of magnitude (see Fig. 2a), which means that many links are comparatively very weak. Thus we have discarded weak network connections by cutting all edges with weights

below a threshold of $w_{ij} < 10^5$. As a consequence, some of the original nodes also become disconnected but we have ensured that all edges of the target phenotypes are kept, especially for target node 180, which would have become isolated otherwise, since all its edges have weights lower than the threshold. Community detection algorithms usually determine a partition of the network node. These partitions may overlap or not. Here we have used methods that produce a non-overlapping partition.

Modularity is a measure that estimates the cohesiveness of a partition found by a community detection algorithm with respect to a graph with the same degree distribution but with edges placed at random [37]. The community partition found with several community detection algorithms from *igraph* has a modularity value of about two, which is not very high but still significantly different from random. Figure 6a shows the communities found by the *Louvain* algorithm. It is important to note that the small community to which vertex 180 belongs is almost always found identically by all the different algorithms tried. Figure 6a clearly shows that node 180, together with its neighbors belonging to the same community, appears to be extremely difficult to reach, all the more taking into account that the intra-community and extra-community edges are weak. On the other hand, phenotype 240 is at the intersection of two bigger and well connected communities and thus it is intuitively reasonable that it should be easier to find.

The following Fig. 6b has been obtained with another community detection method, the *fastgreedy* algorithm, which is also in the *igraph* library. As said above, the results may differ because of the different constraints algorithms impose on the community definition but, again, vertex 180 belongs to a small and weakly connected cluster.

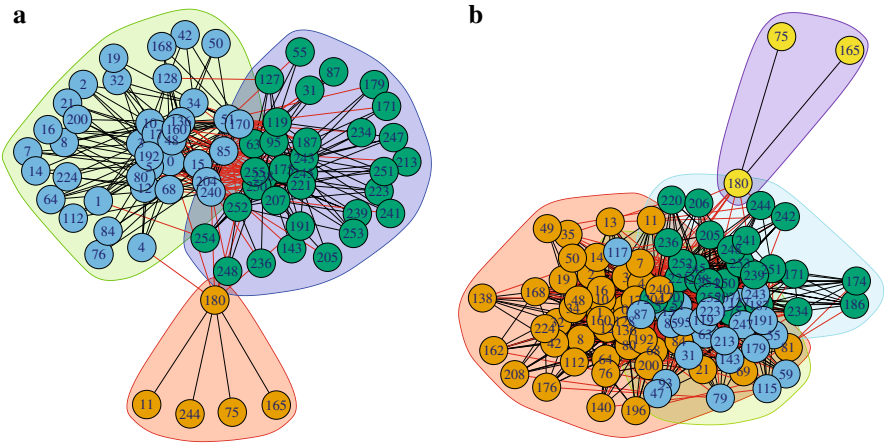

Fig. 6 Community structure of the edge-filtered undirected phenotypic network using two different partition algorithms (see text). In both figures, phenotype 180 clearly belongs to a small community that is very weakly connected to the rest of the network, while phenotype 240 is located in the center of the network and belongs to a larger and well connected community

4.3 Directed phenotype networks

When a fitness is assigned to each phenotype based on its Hamming distance to the target phenotype, the phenotype network becomes oriented since we only allow non-deleterious point mutations. Note that we only consider simple graphs in the current study, i.e., self-loops are excluded in our network analysis, in order to focus on the mutational connections among distinct phenotypes.

4.3.1 In- and out-degrees

A phenotype/node now has edges with two directions, pointing to its neighbors (out-edges) and being pointed from its neighbors (in-edges). Subsequently, in-degree and out-degree can be used to depict how many unique phenotypes can access or can be reached from a reference phenotype.

We often consider the search for a target phenotype as being effective if following fitter phenotypes will most likely lead to the target. That is, fitter phenotypes are expected to have more in-edges to attract mutations while less fit phenotypes should have more out-edges (see an example in Fig. 7).

Figure 8 shows the correlations of in- and out-degrees with the fitness of a phenotype in two directed phenotype networks with different targets. Using both targets, in-degrees are negatively correlated with fitness while out-degrees are positively correlated with fitness. Note that fitness is to be minimized. Phenotypes with better fitness will have less edges going out but more edges coming in, i.e., fitter phenotypes are easier to reach and harder to leave, which is intuitive and desirable since we hope reaching fitter phenotypes will be more likely leading to the path to the target. However, when we compare the correlations using different targets, it can be seen that using a relatively harder target (i.e., phenotype 180) results in weaker correlations of in-/out-degrees and fitness. This indicates that genotypically underrepresented targets are difficult to find partially because they render the guidance of the fitness gradient less effective. That is, reaching fitter phenotypes at a current step does not necessarily lead to better paths for finding the target.

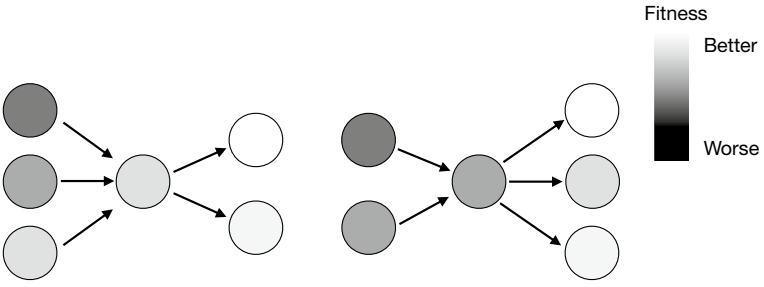

Fig. 7 Fitness in relation to in- and out-degrees. Fitness can better guide the search for the target if phenotypes (circles) with better fitness have higher in-degrees and lower out-degrees comparing with worse phenotypes

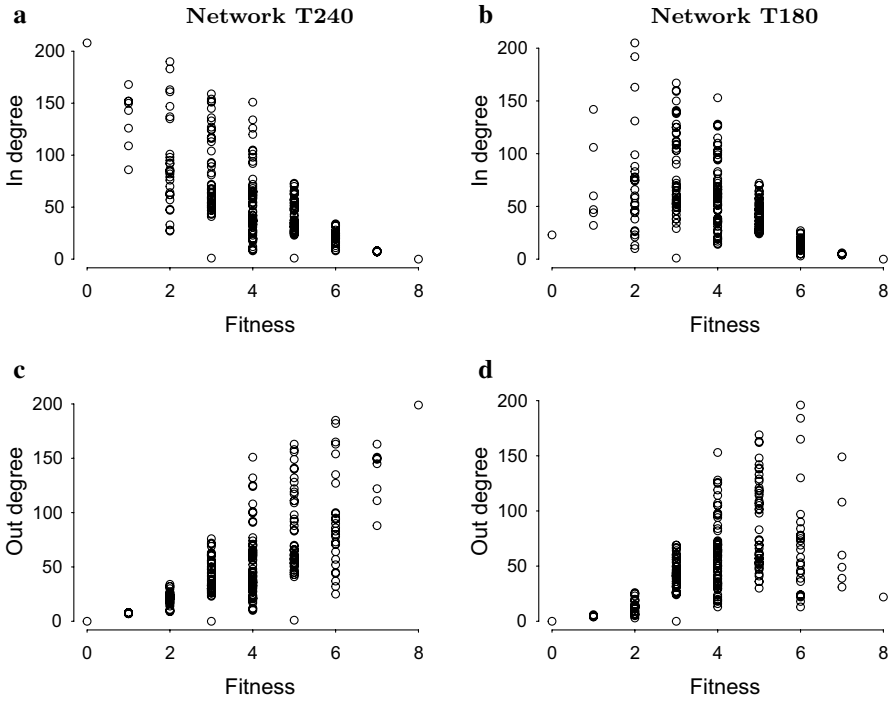

Fig. 8 Correlation of node in-degree (**a**, **b**) and out-degree (**c**, **d**) with fitness in the directed phenotype networks using an over-represented target 240 (**a**, **c**) and under-represented target 180 (**b**, **d**). In **a**, **b**, the negative correlations are with a coefficient of $R^2 = 0.462$ ($p < 2.2 \times 10^{-16}$), and $R^2 = 0.252$ ($p < 4.6 \times 10^{-16}$), respectively. In **c**, **d**, the positive correlations are with a coefficient of $R^2 = 0.461$ ($p < 2.2 \times 10^{-16}$), and $R^2 = 0.262$ ($p < 4.6 \times 10^{-16}$), respectively

4.3.2 Fitness correlation of neighboring phenotypes

Fitness correlation can give statistical information about the fitness assortativity of neighboring nodes in the network. A practical way for evaluating fitness correlation is given by the average fitness of neighbors $\bar{f}_{neighbor}(i)$ of a node i

$$\bar{f}_{neighbor}(i) = \frac{1}{|\mathcal{N}(i)|} \sum_{j \in \mathcal{N}(i)} f_j,$$

where f_i is the fitness of phenotype/node i.

From this quantity one can compute the average fitness of the neighbors $\bar{f}_{neighbor}(f)$ for nodes of the phenotypic network having fitness value f which is a good approximation to the fitness-fitness correlation:

$$\bar{f}_{neighbor}(f) = \frac{1}{N_f} \sum_i \bar{f}_{neighbor}(i),$$

where N_f is the number of nodes with fitness f.

Remembering that a low fitness value is better in our context, one can see from Fig. 9 that the neighboring fitness correlations are quite different when different phenotypes are used as the search target. Specifically, when the over-represented phenotype 240 is set as the target (see Fig. 9a), the fitness values of neighboring phenotypes do not correlate, meaning that a phenotype can be connected to phenotypes with any fitness values. However, as shown in Fig. 9b, when the target is under-represented, a bad phenotype that has a fitness value greater than four tends to have neighbors better than itself, while a good phenotype with a fitness value less than four tends to have neighbors worse than itself.

4.4 Random walks and adaptive walks

4.4.1 Random walks

Although the GP system searches the genotype space and not the much smaller phenotype space, it is still interesting to simulate random walk search in the latter to numerically confirm the above idea that some phenotypes are easy to find while others are hard. Nevertheless, it is worth noting that walking in the phenotypic network ignores the fact that it depends on the sampled genotypic space and thus may not adequately represent the actual difficulty since many steps are abstracted into a single step here and also depend on the sampling accuracy. Random walks on networks are reviewed in [32]. In an unweighted network, the probability for going from node i to node j is $p_{ij} = a_{ij}/k_i$, where k_i is the degree of i and a_{ij} is the corresponding entry in the graph adjacency matrix being 1 if nodes i and j are connected and 0 if they are not. The random walk we are interested in is biased, since edges of the undirected phenotype network are weighted. So we have to modify the probabilities accordingly, but the changes are minor: the transition probability from node i to node j through an edge $\{ij\}$ with weight $w_{ij} \geq 0$ now becomes $p_{ij} = w_{ij}/s_i$, where s_i is the *strength* of node i and is defined as the sum of the weights of the edges from i to its neighbors $\mathcal{N}(i)$ (see definitions in Sect. 2.4). These probabilities are well

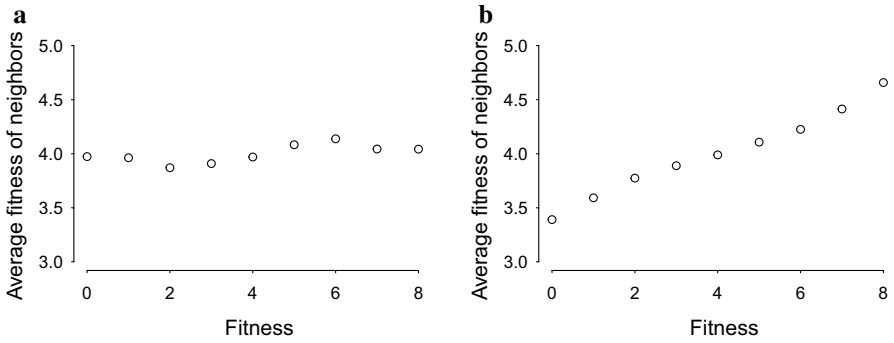

Fig. 9 Average neighbor fitness versus node fitness in the phenotypic network with the target phenotype **a** 240 and **b** 180

behaved since a connected node must have a positive finite strength, $p_{ij} \geq 0$, and $\sum_{j \in \mathcal{N}(i)} p_{ij} = 1$ and form the rows of the transition matrix $\mathbf{T}$.

The stationary distribution $\mathbf{p}^*$ of the random walks defined above can be found by solving the eigenvalue equation $\mathbf{p}^* = \mathbf{p}^* \mathbf{T}$ with an appropriate linear algebra algorithm. However, for large matrices there might be numerical difficulties that require specialized knowledge. Instead, here we shall follow a Monte Carlo simulation approach which, while it is slower, is very simple and gives the same results in the long time limit. It consists in performing a large number of random walk steps starting from each network node and recording each time a given target node is encountered. If the number of random walk steps is large enough, the final frequency with which the target has been hit will provide a very good approximation to its equilibrium occupancy probability.

For each of the two target phenotypes 180 and 240, we numerically simulate biased random walks in the original unfiltered network starting from all network nodes except the target nodes themselves. For each starting node we perform 10^5 random walk steps, for a total of $(N-1) \times 10^5 = 237 \times 10^5$ steps. For each of the phenotypes 180 and 240 we record the number of times it is found, i.e., the number of hits, and the mean number of steps to the first hit, when the node is found.

Results are shown in Table 1. From the number of hits and the first hit times, it is apparent that phenotype 180 is much harder to find than phenotype 240. Furthermore, if we exclude the first neighbors of the target node as starting nodes in the random walk, it becomes even more difficult, comparatively, to find phenotype 180 (figures after the comma in Table 1). We can also see that phenotype 240 is very often found directly from a starting node that is a first neighbor given that 240 has 223 neighbors while 180 only has 41 connections.

4.4.2 Adaptive walks

In contrast with random walks where fitness doesn't play a role, *adaptive walks* can be performed in a phenotype network when each phenotype is attributed a fitness value. Adaptive walks can be defined in several ways [25] but given that the walk is at a certain phenotype i at step t, the basic idea is to move to a neighboring node k at step $t+1$, provided that the fitness of k is better than i's fitness; if no neighbor has a better fitness the walk terminates at i.

Our implementation of adaptive walks is somewhat different from the usual ones as it takes into account the weight of the edge w_{ij} between phenotypes i and j. This is

Table 1 Average number of hits and first hitting times for random walks having nodes 180 and 240 as targets. 237×10^5 random walks steps are performed in total

	Target Phenotype 180	Target Phenotype 240
Number of hits	2, 0	97,465, 5862
First hitting time	910,306, –	10, 10

The figures after the commas refer to the same quantities when the first neighbors of nodes 180 and 240 are excluded as starting nodes for the walk

because this weight stands for the number of single-point mutations that transform a genotype belonging to phenotype *i* into a genotype for phenotype *j* in one step, and thus represents the potential frequency of such a transition. Our algorithm for an adaptive walk can be summarized as follows:

1. choose a starting node *a* and make it the current node *current*
2. sort the neighbors of *current* in decreasing order of their link weights and place them in the list *weights*
3. if it exists, set *current* to the first node *n* in *weights* whose fitness *f*(*n*) is better than *f*(*current*) and go to 2; otherwise stop

where 2 and 3 are repeated for a prefixed number of steps. This is a deterministic algorithm that needs to be run only once for each starting node.

We applied the above algorithm to the directed and weighted networks of target phenotypes 180 and 240. Those phenotypes are the single global optima of their respective networks in terms of fitness. We did a first computational experiment by starting an adaptive walk from each network node except the target phenotype for each of 180 and 240 networks. The results were the following: target phenotype 180 was reached 27 times out of 237 adaptive walks. On the other hand, phenotype 240 was the end state of 39 out of 239 walks. We see that, even taking fitness into account, 180 is more difficult to reach by hill-climbing than 240. At this point it has to be noted that most phenotypes are degenerate local optima, i.e., there is more than one phenotype having the same fitness at the end of an adaptive walk. By contrast, the globally optimal phenotypes 180 and 240 are single.

We next did another numerical simulation, this time only using as starting nodes those that have fitnesses above a certain level. Recall that the optimal fitness is zero and the worst value is eight, i.e., the walk starts from a distribution of "worse" nodes here, and so the walk in general requires a longer time to find its way to better fitness nodes. Excluding all nodes whose fitness is lower than 5 as starting points for the walks, phenotype 240 is reached 8 times, while 180 is hit only on three occasions. Restricting the starting node set further to only those that have fitness 6 or higher results in node 240 being reached three times and node 180 zero times. In conclusion, even when there is fitness guidance in the search, phenotype 240 is always easier to find than phenotype 180.

5 Discussion

The genotype–phenotype mapping plays an essential role in evolvability, since the variations characterizing the search occur in the genotype space, but the quality or behavior of a system can only be observed and evaluated at the phenotypic level. We argue that some target phenotypes are more difficult to find than others, not only because they are most likely under-represented in genotypic space, but also because the mutational connections between phenotypes are altered through setting different target phenotypes.

In this study, we took a network approach and quantitatively analyzed the distribution of mutational connections among phenotypes and how this distribution is changed with different phenotypes set as target. Using a Boolean LGP algorithm, we sampled the genotypic and phenotypic spaces and constructed a phenotype network to characterize the distribution of mutational connections among phenotypes. By setting two different phenotypes as the target, one genotypically over-represented and one under-represented, we compared the properties of the resulting directed, weighted phenotype networks.

Similar to many GP systems, our Boolean LGP algorithm has a highly redundant mapping from genotypes (6.4×10^{13}) to phenotypes (256). Such redundancy is heterogeneously distributed among phenotypes, with the most abundant phenotype possessing about 10% of the entire genotype space while some other phenotypes never appeared in our samples. By examining the undirected, weighted phenotype network, we found that genotypically more abundant phenotypes have more access to different phenotypes and more tendency to mutate into certain neighboring phenotypes. These findings are in line with reported observations in evolutionary biology, where the genotypic redundancy of a phenotype is often considered as its robustness, and most robust phenotypes are often found more evolvable.

We chose two phenotypes, 180 and 240, as targets, and observed that in addition to having considerably different degrees, 41 and 233, the two phenotypes have very different community structures. This suggests that target 180 is much more difficult to find, not only because it was connected to fewer neighbors, but also because it is located in a small and distant community. It was also interesting to see that in the directed phenotype network resulting from setting 180 as target, fitness was less effective at guiding evolution, since fitness and in-/out-degree of a phenotype are less correlated, i.e., reaching a fitter phenotype at a current step would not necessarily lead to more promising paths to finding the target.

The search performance of an evolutionary algorithm can vary considerably with different problem instances. Our study provides a quantitative investigation into this issue using complex network analysis. That a specific target is hard to reach can have multiple explanations: (1) the target is under-represented in genotypic space; (2) the target is connected to only a few phenotypes in phenotypic space; (3) the target belongs to a small community distant from the rest of the phenotypes in that space; and (4) setting the target has wired the connections among phenotypes in a way that renders following fitter phenotypes in order to reach the target a less effective strategy.

We hope our observations can be found useful to inspire more intelligent search mechanisms that are able to overcome these challenges. For instance, can we use network-based metrics to supplement fitness evaluation such that the search in the genotypic space is better guided towards a genotypically under-represented and weakly connected target? Is the genotype–phenotype map itself evolvable? Can we evolve genotype–phenotype maps that allow finding rare phenotype targets more easily?

Acknowledgements This research was supported by the Natural Sciences and Engineering Research Council (NSERC) of Canada Discovery Grant RGPIN-2016-04699 to T.H., and the Koza Endowment

fund provided to W.B. by Michigan State University and supported by its BEACON Center for the Study of Evolution in Action.

References

1. P. Alberch, From genes to phenotype: dynamical systems and evolvability. Genetica **84**, 5–11 (1991)
2. L. Altenberg, The evolution of evolvability in genetic programming, in *Advances in Genetic Programming*, (MIT Press, Cambridge, MA, 1994), pp. 47–74
3. L. Altenberg. Genome growth and the evolution of the genotype-phenotype map, in W. Banzhaf and F. Eeckman, eds., *Evolution and Biocomputation*, volume 899 of *Lecture Notes in Computer Science*. (Springer, 1995), pp. 205–259
4. W. Banzhaf, Genotype–phenotype mapping and neutral variation—a case study in genetic programming, in *Parallel Problem Solving from Nature, volume of 866 Lecture Notes in Computer Science*, ed. by Y. Davidor, H.-P. Schwefel, R. Manner (Springer, Berlin, 1994), pp. 322–332
5. A.-L. Barábasi, *Network Science* (Cambridge University Press, Cambridge, 2016)
6. J.D. Bloom, S.T. Labthavikul, C.R. Otey, F.H. Arnold, Protein stability promotes evolvability. Proc. Nat. Acad. Sci. **103**(15), 5869–5874 (2006)
7. M.F. Brameier, W. Banzhaf, *Linear Genetic Programming* (Springer, Berlin, 2007)
8. P. Catalan, A. Wagner, S. Manrubia, J.A. Cuesta, Adding levels of complexity enhances robustness and evolvability in a multilevel genotype–phenotype map. J. R. Soc. Interface. **15**(138), 20170516 (2018)
9. J. Clune, K.O. Stanley, R.T. Pennock, C. Ofria, On the performance of indirect encoding across the continuum of regularity. IEEE Trans. Evolut. Comput. **15**(3), 346–367 (2011)
10. M.C. Cowperthwaite, E.P. Economo, W.R. Harcombe, E.L. Miller, L.A. Meyers, The ascent of the abundant: how mutational networks constrain evolution. PLoS Comput. Biol. **4**(7), e1000110 (2008)
11. M.C. Cowperthwaite, L.A. Meyers, How mutational networks shape evolution: lessons from RNA models. Annu. Rev. Ecol. Evol. Syst. **38**, 203–230 (2007)
12. G. Csardi, T. Nepusz, The igraph software package for complex network research. InterJ. Complex Syst. **1695**, 1–9 (2006)
13. S. Cussat-Blanc, K. Harrington, W. Banzhaf, Artificial gene regulatory networks—a review. Artif. lLfe **24**(4), 296–328 (2019)
14. E.H. Davidson, *The Regulatory Genome: Gene Regulatory Networks in Development and Evolution* (Elsevier, Amsterdam, 2010)
15. J.A.G.M. de Visser, J. Krug, Empirical fitness landscapes and the predictability of evolution. Nat. Rev. Genet. **15**, 480–490 (2014)
16. M. Ebner, M. Shackleton, R. Shipman, How neutral networks influence evolvability. Complexity **7**(2), 19–33 (2002)
17. A. Fontana, Epigenetic tracking: biological implications, in *European Conference on Artificial Life*, (Springer, 2009), pp. 10–17
18. E. Galvan-Lopez, R. Poli, An empirical investigation of how and why neutrality affects evolutionary search, in M. Cattolico, ed., *Proceedings of the Genetic and Evolutionary Computation Conference*, (2006), pp. 1149–1156
19. T. Hu, W. Banzhaf, Neutrality and variability: Two sides of evolvability in linear genetic programming, in *Proceedings of the 18th Genetic and Evolutionary Computation Conference (GECCO)*, (2009), pp. 963–970
20. T. Hu, W. Banzhaf, Quantitative analysis of evolvability using vertex centralities in phenotype network, in *Proceedings of the 25th Genetic and Evolutionary Computation Conference (GECCO)*, (2016), pp. 733–740
21. T. Hu, W. Banzhaf, Neutrality, robustness, and evolvability in genetic programming, in R. Riolo, B. Worzel, B. Goldman, B. Tozier, eds., *Genetic Programming Theory and Practice XIV*, chapter 7, (Springer, 2018), pp. 101–117

22. T. Hu, W. Banzhaf, J.H. Moore, The effect of recombination on phenotypic exploration and robustness in evolution. Artif. Life **20**(4), 457–470 (2014)
23. T. Hu, J. Payne, W. Banzhaf, J.H. Moore, Evolutionary dynamics on multiple scales: a quantitative analysis of the interplay between genotype, phenotype, and fitness in linear genetic programming. Genet. Program. Evolv. Mach. **13**(3), 305–337 (2012)
24. T. Hu, M. Tomassini, W. Banzhaf, Complex network analysis of a genetic programming phenotype network, in *Proceedings of the 22nd European Conference on Genetic Programming (EuroGP)*, volume 11451 of *Lecture Notes in Computer Science*, (2019), pp. 49–63
25. S. Kauffman, S. Levin, Towards a general theory of adaptive walks on rugged landscapes. J. Theor. Biol. **128**(1), 11–45 (1987)
26. D.B. Kell, Genotype-phenotype mapping: genes as computer programs. Trends Genet. **18**(11), 555–559 (2002)
27. M. Kirschner, J. Gerhart, Evolvability. Proc. Natl. Acad. Sci. **95**, 8420–8427 (1998)
28. M. Kirschner, J.C. Gerhart, *The Plausibility of Life: Resolving Darwin's Dilemma* (Yale University Press, New Haven, 2006)
29. J. D. Knowles, R. A. Watson, On the utility of redundant encodings in mutation-based evolutionary search, in *Parallel Problem Solving from Nature—PPSN VII*, volume 2439 of *Lecture Notes in Computer Science*, (2002), pp. 88–98
30. J.R. Koza, D. Andre, M.A. Keane, F.H. Bennett III, *Genetic Programming III: Darwinian Invention and Problem Solving*, vol. 3 (Morgan Kaufmann, Burlington, 1999)
31. R.E. Lenski, J.E. Barrick, C. Ofria, Balancing robustness and evolvability. PLoS Biol. **4**(12), e428 (2006)
32. N. Masuda, M.A. Porter, R. Lambiotte, Random walk and diffusion in networks. Phys. Rep. **716**, 1–58 (2017)
33. R.C. McBride, C.B. Ogbunugafor, P.E. Turner, Robustness promotes evolvability of thermotolerance in an RNA virus. BMC Evolut. Biol. **8**, 231 (2008)
34. J.F. Miller, W. Banzhaf, Evolving the program for a cell: From French flags to Boolean circuits, in *On Growth, Form and Computers*, ed. by S. Kumar, P. Bentley (Academic, New York, 2003), pp. 278–301
35. M.E.J. Newman, *Networks: An Introduction* (Oxford University Press, Oxford, 2018)
36. M.E.J. Newman, R. Engelhardt, Effects of selective neutrality on the evolution of molecular species. Proc. R. Soc. B **265**(1403), 1333–1338 (1998)
37. M.E.J. Newman, M. Girvan, Finding and evaluating community structure in networks. Phys. Rev. E **69**, 026113 (2004)
38. K. L. Nickerson, Y. Chen, F. Wang, T. Hu, Measuring evolvability and accessibility using the Hyperlink-Induced Topic Search algorithm, in *Proceedings of the 27th Genetic and Evolutionary Computation Conference (GECCO)*, (2018), pp. 1175–1182
39. L. Page, S. Brin, R. Motwani, T. Winograd, *The pagerank citation ranking: Bringing order to the web* Technical report, Stanford InfoLab (1999)
40. J.L. Payne, A. Wagner, The causes of evolvability and their evolution. Nat. Rev. Genet. **20**, 24–38 (2019)
41. R. Rezazadegan, C. Barrett, C. Reidys, Multiplicity of phenotypes and RNA evolution. J. Theoret. Biol. **447**, 139–146 (2018)
42. F. Rothlauf, D.E. Goldberg, Redundant representations in evolutionary computation. Evolut. Comput. **11**(4), 381–415 (2003)
43. S. Schaper, A.A. Louis, The arrival of the frequent: how bias in genotype–phenotype maps can steer populations to local optima. PLoS One **9**(2), e86635 (2014)
44. P. Schuster, W. Fontana, P .F. Stadler, I .L. Hofacker, From sequences to shapes and back: a case study in RNA secondary structures. Proc. R. Soc. Lond. Ser. B Biol. Sci. **255**(1344), 279–284 (1994)
45. P. Shannon, A. Markiel, O. Ozier, N.S. Baliga, J.T. Wang, D. Ramage, N. Amin, B. Schwikowski, T. Ideker, Cytoscape: a software environment for integrated models of biomolecular interaction networks. Genome Res. **13**, 2498–2504 (2003)
46. T. Smith, P. Husbands, M. O'Shea, Neutral networks and evolvability with complex genotype-phenotype mapping, in J. Kelemen, P. Sosik, eds., *Proceedings of the European Conference on Artificial Life*, volume 2159 of *Lecture Notes in Artificial Intelligence*, (Springer-Verlag, 2001), pp. 272–281

47. E. van Nimwegen, J.P. Crutchfield, M.A. Huynen, Neutral evolution of mutational robustness. Proc. Natl. Acad. Sci. **96**(17), 9716–9720 (1999)
48. A. Wagner, Robustness, evolvability, and neutrality. Fed. Eur. Biochem. Soc. Lett. **579**(8), 1772–1778 (2005)
49. A. Wagner, Robustness and evolvability: a paradox resolved. Proc. R. Soc. B **275**(1630), 91–100 (2008)
50. G.P. Wagner, L. Altenberg, Perspective: Complex adaptations and the evolution of evolvability. Evolution **50**(3), 967–976 (1996)

Affiliations

Ting Hu[1,2] · Marco Tomassini[3] · Wolfgang Banzhaf[4]

Marco Tomassini
marco.tomassini@unil.ch

Wolfgang Banzhaf
banzhafw@msu.edu

[1] School of Computing, Queen's University, Kingston, ON, Canada

[2] Department of Computer Science, Memorial University, St. John's, NL, Canada

[3] Faculty of Business and Economics, Information Systems Department, University of Lausanne, Lausanne, Switzerland

[4] Department of Computer Science and Engineering, and BEACON Center, Michigan State University, East Lansing, MI, USA