Robustness, evolvability and phenotypic complexity: insights from evolving digital circuits

Nicola Milano1 · Paolo Pagliuca1 · Stefano Nolfi1

Received: 16 July 2018 / Revised: 31 October 2018 / Accepted: 19 December 2018 / Published online: 2 January 2019
© Springer-Verlag GmbH Germany, part of Springer Nature 2019

Abstract
We analyze the relation between robustness to mutations, phenotypic complexity, and evolvability in the context of artificial circuits evolved for the ability to solve a parity problem. We demonstrate that whether robustness to mutations enhances or diminishes phenotypic variability and evolvability depends on whether robustness is achieved through the development of parsimonious (phenotypically simple) solutions, that minimize the number of genes playing functional roles, or through phenotypically more complex solutions, capable of buffering the effect of mutations. We show that the characteristics of the selection process strongly influence the robustness and the performance of the evolving candidate solutions. Finally, we propose a new evolutionary method that outperforms evolutionary algorithms commonly used in this domain.

Keywords Evolvability · Robustness · Phenotypic variability · Phenotypic complexity · Evolutionary stagnation

1 Introduction
We investigate how the characteristics of the evolutionary algorithm influence the evolvability of candidate solutions, i.e. the propensity of evolving individuals to generate better solutions as a result of random genetic variations. This objective has been pursued by evolving digital circuits, which represent a classic domain of application for evolutionary algorithms and share important properties with natural systems such as proteins, RNA, regulatory circuits and metabolic networks (Wagner 2011; Raman and Wagner 2011).

Robustness and evolvability are fundamental properties of evolving systems (Wagner 2008). For the purpose of this paper we focus on robustness to mutations, defined as the capability of a system to preserve its functionalities after mutations. Robustness is necessary to successfully transfer adaptive capacities from parents to offspring in the presence of genetic variation. Moreover, we define evolvability as the propensity of a system to produce heritable phenotypic variations as a result of mutations. Evolvability is required to improve the capacities of candidate solutions across generations. These terms do not have a unique definition, for other usage (see: Houle 1992; Wagner and Altenberg 1996; Kirschner and Gerhart 1998; Bedau and Packard 2003; Earl and Deem 2004; Sniegowski and Murphy 2006; Wagner 2008; Masel and Trotter 2010).

We show that the selection of candidate solutions that are robust with respect to mutations can have negative or positive effects on evolvability, depending on how robustness is achieved. The effect is negative when robustness is achieved by selecting phenotypical simple solutions, in which the number of genes playing a functional role is minimal (De Visser et al. 2003). On the contrary, the effect is positive when robustness is achieved by selecting phenotypical non-minimal solutions that achieve robustness through redundancy or degeneracy (Tononi et al. 1999; Frei and Whitacre 2012).

Moreover, we show that the characteristics of the selection process strongly influence the robustness and the performance of the evolving candidate solutions.

The analysis of the evolutionary dynamics also provided the basis for the design of a new original algorithm, called Parallel Stochastic Hill Climber (PSHC), that outperforms the evolutionary algorithm commonly used in this domain.
2 Method

In this section we describe the digital circuits, the evolutionary algorithms, and the measures used to analyze the evolutionary dynamics.

2.1 The digital circuits

Digital circuits (Fig. 1) are systems that compute digital logic functions, such as the multiplication of digital numbers, by receiving as inputs two or more binary (Boolean) values and by producing as output one or more binary values. They are composed of multiple logic gates that receive two binary values (from the input pattern and/or from the output of other logic gates) in input and produce one binary value in output by computing an elementary logic function (OR, AND, NAND ext.) of the input. The logic function computed by a circuit depends on the functions computed by its constituent logic gates and by the way in which they are wired.

Digital circuits can be realized in hardware or simulated in a computer. In standard electronic digital circuits, the number and type of gates and the way in which they are wired is hardwired and hand-designed. In reconfigurable electronic digital circuits (such as the FPGA, see Balch 2003), instead, the logic function computed by each gate and the way in which gates are wired can be varied. In evolvable hardware applications or, more generally, in evolutionary circuits, the logic functions computed by each gate and the way in which gates are wired are encoded in artificial genotypes and evolved. Evolving circuits are selected on the basis of their fitness that is usually computed by measuring the extent to which the function computed by a circuit approximates a given target function (Thompson et al. 1999).

In our experiments we evolved simulated digital circuits with five inputs, 400 logic gates divided into 20 layers of 20 gates, and one output for the ability to compute a 5-bit even parity function (i.e. to produce as output 1 when there is an even number of 1 s in the input pattern and 0 otherwise). We choose this function since it constitutes a rather difficult problem for evolving circuits including OR, AND, NAND, and NOR logic gates (Miller and Thomson 2000).

As in several related works (Thompson et al. 1999; Hartmann and Haddow 2004; Raman and Wagner 2011), we choose to provide digital circuits with a fixed number of gates since this permits us to use a simple encoding schema. For alternative approaches allowing to evolve circuits having a variable number of gates, see for example Miller and Hartmann (2001) and Macia and Solé (2009). Notice however that the utilization of a fixed number of logic gates only limits the maximum size of the circuits. Indeed, as we will see, evolving circuits typically rely on a much smaller number of gates with respect to the limit, i.e. they include several non-functional gates that do not contribute to the function computed by the circuit itself (see for example the gate computing the state 6 in Fig. 1). In other words, the functional size of the evolving circuits can vary freely, within the upper limit imposed by the maximum number of gates. This also implies that the number of genes encoding phenotypical components (gates) playing a functional role can also vary freely during evolution.

Circuits are evaluated for the ability to map the $2^n$ possible input patterns into the corresponding desired outputs (i.e. 1 for input patterns with an even number of 1 and 0 otherwise). The fitness of the circuits is calculated on the basis of the following equation:

$$F = 1 - \frac{1}{2^n} \sum_{j=1}^{2^n} \left| O_j - E_j \right|$$  (1)
where $n$ is the number of inputs of the circuit, $j$ is the number of the input patterns varying in the range $[1, 2^n]$, $O_j$ is the output of the circuit for pattern $j$, $E_j$ is the desired output for pattern $j$.

The genotype of evolving circuit is constituted by a vector of integer numbers encoding the function computed by each logic gate and the way in which gates are wired. This approach has been named Cartesian Genetic Programming (Miller and Thomson 2000; Miller et al. 2000). More specifically each genotype includes $400 \times 3 = 1200$ genes that specify the characteristics of the nodes and 1 additional gene that specify the identification number of the node that is used as output for the entire circuit. The inputs are indexed in the range $[1–5]$ and the nodes are indexed in range $[6-406]$. For each node, 1 gene bounded in the range $[1, 4]$ indicates the function of the node ($1 = \text{OR}$, $2 = \text{AND}$, $3 = \text{NAND}$, $4 = \text{NOR}$) and two genes bounded in the range $[1, 5+(L−1)\times20]$ indicate the indexes of two corresponding inputs of the node. Nodes are arranged in 20 layers that receive inputs only from the previous layers of nodes and from the 5 inputs. $L$ represents the layer of the corresponding logic gate. The value of the last gene, that encodes the node acting as output for the entire circuit, is bounded in the range $[6, 406]$. Mutations are realized by replacing each integer with a certain probability ($\text{MutRate}$) with a number randomly generated with a uniform distribution in the appropriate range.

We will use the term behavior to indicate the outputs produced by a circuit in response to each possible input pattern. Moreover, we will use the term functional size to indicate the number of gates actively contributing to the outputs produced by the circuit. Notice that circuits with the same fitness might differ at the level of the behavior produced. Indeed, circuits producing different outputs can produce the same number of correct and incorrect responses. Notice also that circuits displaying the same behavior might differ at the level of the circuit’s components. Indeed, circuits characterized by different types of gates or different wiring can produce the same outputs. The fact that the number of circuits that differ with respect to the type of the gates and/or the way in which the gates are wired is much greater than the number of circuits that differ at the level of the fitness implies that a large portion of genetic variations are neutral, i.e. they produce variations at the level of the circuit and/or at the level of behavior that do not alter the fitness of the circuit.

### 2.2 Evolutionary algorithms

To investigate the relation between the characteristics of the algorithm and the evolvability of the circuits, we compared three different algorithms.

The first is a $(1 + \lambda)$ evolutionary strategy (ES, Rechenberg 1973; Beyer and Schwefel 2002) that operates on the basis of a single parent, produces $\lambda$ offspring, and selects the best between the parent and the offspring as a new parent (see the pseudo-code below). We choose this algorithm since it is commonly used for the evolution of digital circuits (Miller et al. 2000) and, more generally, for the evolution of graph structures (Miller and Thomson 2000; Miller 2011).

To regulate the effect of the selection pressure, we added to the fitness of the evolving individuals a value randomly selected in the range $[-\text{Stochasticity}, \text{Stochasticity}]$ with a uniform distribution (see Jin and Branke 2005). We decided to use this technique in combination with a selection operator that always selects the best between parents and offspring, rather than probabilistic selection operators such as roulette wheels or tournament selection, since: (i) it allows to regulate the selective pressure quantitatively by varying a single parameter (see Back 1994), (ii) it permits to regulate the selective pressure from the maximum value, in which only the best candidate solutions between parents and offspring are selected, to a minimal value, in which the differential reproductive probability of better and worse individuals is minimal (see Back 1994), (iii) it is qualitatively similar to the stochastic variation of fitness caused by uncontrolled variations occurring in uncertain environmental conditions. An example of uncertain conditions is constituted by the experiments involving robots, in which the behavior displayed by the agent and, consequently, the fitness measure tend to vary during repeated evaluations even in apparently identical conditions as a consequence of small differences in the initial position and orientation of the robot, lighting conditions, etc. Previous demonstrations of the fact that the introduction of noise in the fitness measure promotes the evolution of better solutions are reported in (Bäck and Hammel 1994; Levitan and Kauffman 1994; Rana et al. 1996).
The second algorithm is a \((\mu + \mu)\) ES that operates with \(\mu\) parents, enables each parent to produce a single offspring, and select the \(\mu\) best individuals between the parents and the offspring as new parents (see the pseudo-code below). This type of algorithm is widely used for the evolution of neural networks (e.g. Pagliuca et al. 2018). Also in this case we use the \textit{Stochasticity} parameter to regulate the selective pressure.

Notice that in this method the offspring of individuals that are more robust to mutations have more chances to be selected than the offspring of individuals that are less robust. To illustrate this point, consider the case in which two parents have the same fitness and in which the former parent is more robust to mutations than the second parent. Moreover, imagine that the mutations received by offspring are neutral or counter-adaptive (none of the mutations produce an improvement of the fitness). The offspring of the former parent have a greater probability to preserve the fitness of the parent than the offspring of the latter parent. Consequently, the offspring of the parent that is more robust has a greater probability to be selected. Overall, this implies that the competition between the evolving individuals is regulated primarily by their fitness and secondarily (in the absence of adaptive mutations) by their robustness.

The competition between the individuals evolved with the \((1 + \lambda)\) method, instead, is not influenced by robustness to mutations. This is due to the fact that offspring originate from the same parent. Offspring might differ among themselves in terms of robustness as a result of the mutations that they received. However, mutations that alter the level of robustness without altering the fitness do not provide an adaptive advantage in the \((1 + \lambda)\) method since two offspring with the same fitness have exactly the same probability to be selected, irrespectively of whether the former is more robust of the latter or vice versa.

---

**Algorithm 1.** \((1 + \lambda)\) ES

1: genotype[0].generateRandomly()
2: genotype[0].fitness = genotype[0].evaluate()
3: \textbf{while} Evaluations < MaxEvaluations
4: \hspace{1em} genotype[0].noisefitness = genotype[0].fitness + rand(-Stochasticity, Stochasticity)
5: \hspace{1em} for o in range(1, \(\lambda+1\))
6: \hspace{2em} genotype[o] = genotype[0]
7: \hspace{2em} genotype[o].mutate()
8: \hspace{1em} genotype[o].fitness = genotype[o].evaluate()
9: \hspace{1em} genotype[o].noisefitness[o] = genotype[o].fitness + rand(-Stochasticity, Stochasticity)
10: \hspace{1em} \textbf{end for}
11: genotype.sort(key=noisefitness, order=descending)
12: \textbf{end while}

**Algorithm 2.** \((\mu + \mu)\) ES

1: \textbf{for} p in range(0, \(\mu\))
2: \hspace{1em} genotype[p].generateRandomly()
3: \hspace{1em} genotype[p].fitness = genotype[p].evaluate()
4: \hspace{1em} \textbf{end for}
5: \textbf{while} Evaluations < MaxEvaluations
6: \hspace{1em} for o in range(0, \(\mu\)):
7: \hspace{2em} genotype[o].noisefitness = genotype[o].fitness += rand(-Stochasticity, Stochasticity)
8: \hspace{2em} genotype[o + \(\mu\)] = genotype[o]
9: \hspace{2em} genotype[o + \(\mu\)].mutate()
10: \hspace{2em} genotype[o + \(\mu\)].fitness = genotype[o + \(\mu\)].evaluate()
11: \hspace{2em} genotype[o + \(\mu\)].noisefitness = genotype[o + \(\mu\)].fitness
12: \hspace{2em} += rand(-Stochasticity, Stochasticity)
13: \hspace{1em} \textbf{end for}
14: \hspace{1em} genotype.sort(key=noisefitness, order=descending)
15: \hspace{1em} \textbf{end while}
The third algorithm designed by us called Parallel Stochastic Hill Climber (PSHC) consists of a combination of a $(\mu+\mu)$ and a $(1+1)$ ES (see pseudo-code below). In this algorithm each parent is adapted through an $(1+1)$ ES for a certain number of Variations during which the parent or a varied version of the parent generates a single mutated candidate solution that is discarded or used to replace the previous candidate solution depending on whether or not it is outperformed by the original solution. The best candidate solution obtained during this variation phase is then used to replace the parent and, with a certain low probability (Interbreeding), the worst individual of the population. The combined usage of the two types of evolutionary strategies allows to combine the advantages of operating on a population of parents with the advantages that can be gained by reducing the level of competition between the members of the population.

Also in the case of this algorithm we regulate the selection pressure during the variation phase by adding to the fitness of the candidate solutions a value randomly selected in the range $[-\text{Stochasticity}, \text{Stochasticity}]$ with a uniform distribution. However, the selection of the best candidate solution obtained during the variation phase is made on the basis of the actual fitness (i.e. the fitness without noise). The rationale behind this choice is that it enables to select sub-optimal solutions temporarily, i.e. during the $(1+1)$ variation phase, but not during the $(\mu+\mu)$ selection phase. Selecting sub-optimal solutions reduces the risk to remain trapped in local minima but causes a reduction of performance that can be temporal or permanent. Retaining sub-optimal solutions temporarily only during the variation phase enables to reduce the risk to remain trapped in local minima while eliminating the risk to retain variation that produce a permanent reduction of performance.

Algorithm 3. Parallel Stochastic Hill Climber (PSHC)

```python
1: for p in range(0, \lambda)
2:     genotype[p].generateRandomly()
3:     genotype[p].fitness = genotype[p].evaluate()
4: end for
5: genotype.sort(key=fitness, order=descending)
6: while Evaluations < MaxEvaluations
7:     for p in range (0, \lambda)
8:         var-genotype[0] = genotype[p]
9:     for v in range(1, NVariations)
10:        var-genotype[v] = var-genotype[v-1]
11:        var-genotype[v].mutate()
12:        var-genotype[v].fitness = var-genotype[v].evaluate()
13:        if (var-genotype[v].fitness + rand(-Stochasticity, Stochasticity))
14:            >= var-genotype[v-1].fitness + rand(-Stochasticity, Stochasticity))
15:            do nothing
16:        else
17:            var-genotype[v] = var-genotype[v-1]
18:            var-genotype[v].fitness = var-genotype[v-1].fitness
19:        endif
20:    end for
21:    var-genotype.sort(key=fitness, order=descending)
22:    if (var-genotype[0].fitness >= genotype[p].fitness)
23:        genotype[p] = var-genotype[0]
24:        genotype[p].fitness = var-genotype[0].fitness
25:    endif
26: end while
27: genotype.sort(key=fitness, order=descending)
28: if (rand([0, 1.0]) < Interbreeding and var-genotype[0].fitness > genotype[\lambda - 1].fitness)
29:    genotype[\lambda - 1] = var-genotype[0]
30:    genotype[\lambda - 1].fitness = var-genotype[0].fitness
31: endif
32: end while
```
The PSHC is a form of island evolutionary algorithm (Whitley et al. 1998) in which the population is divided into a number of sub-populations and the individuals of each population are allowed to compete with the individuals of the other sub-populations only occasionally. In the case of the PSHC, however, the islands are constituted by single individuals that adapt on the basis of an \((1 + \lambda)\) evolutionary strategy instead of sub-populations in which individuals compete with the other members of the sub-population. This difference is important since, as pointed out above, the competition between evolving individuals influences the robustness and the evolvability of the agents. Indeed, experiments carried out by using a variation of the algorithm in which the islands were formed by 2, 5, or 10 individuals led to significantly worse results (see below).

The evolutionary process is continued until a maximum number of total evaluations are performed. This number was set to \(6 \times 10^6\). The rationale behind this is that the number of total evaluations are performed. This number was set to \(6 \times 10^6\). The rationale behind this is that the evaluation of candidate solutions constitutes the major computational cost. Consequently, fixing the total number of candidate solution evaluations permits to maintain the computational cost of experiments carried with different algorithms approximately constant. Each experiment is replicated 30 times with randomly different initial populations.

To verify that the differences in performance are not influenced by the parameters setting, we replicated the experiments by systematically varying the parameters and we compared the results achieved by different algorithms in the experiments performed with the optimal parameters.

### 2.3 Measures

**Robustness** to mutations, defined as the capability of circuits to preserve their functionality after mutations (Abdelhalim et al. 2011) is measured in two way: (i) by calculating the fraction of offspring that have a fitness equal or greater than the fitness of the parent (fraction of fitness preserving offspring), and (ii) by calculating the fraction of offspring generated by performing a single random mutation that have a fitness equal or greater than the fitness of the parent. To estimate robustness with good precision, the measures are calculated over \(10^4\) variations of the parent’s circuit. The first measure is influenced by the mutation rate that might vary in different experiments, while the second measure is independent of it.

**Phenotypic complexity**, defined as the complexity of the functional elements that constitute the system and the complexity of the way in which the elements are organized, is measured by counting the number of gates that contribute to generate the output of the circuit (functional gates). Functional gates correspond to the gate that constitute the output of the entire circuit and the gates that project connections to the output directly or indirectly, through other gates. Given that in our experimental setup genes encode the logic function performed by each gate and the way in which the gate is wired to the other gates, this measure also indicates the number of genes encoding functional properties of the system. The term phenotypic complexity does not have a unique definition (see for example see Adami 2002; Carlson and Doyle 2002; Crutchfield and Görnerup 2006; Hazen et al. 2007; Whitacre 2010). Moreover, phenotypic complexity is hard to define formally and to measure in the case of systems composed by heterogeneous elements and/or displaying multi-level organizations. The possibility to use a relatively straightforward measure in the case of our experiments, therefore, is due to the utilization of simple feed-forward digital circuits composed by homogeneous elements.

**Phenotypic variability**, defined as the propensity of circuits to vary phenotypically as a result of mutations, is measured (following Raman and Wagner 2011) by calculating the number of unique functions computed by circuits located in the genetic neighborhood of the original circuit to preserve their functionality after mutations (Abdelhalim et al. 2011) is measured in two way: (i) by calculating the fraction of offspring that have a fitness equal or greater than the fitness of the parent (fraction of fitness preserving offspring), and (ii) by calculating the fraction of offspring generated by performing a single random mutation that have a fitness equal or greater than the fitness of the parent. To estimate robustness with good precision, the measures are calculated over \(10^4\) variations of the parent’s circuit. The first measure is influenced by the mutation rate that might vary in different experiments, while the second measure is independent of it.

**Phenotypic complexity**, defined as the complexity of the functional elements that constitute the system and the complexity of the way in which the elements are organized, is measured by counting the number of gates that contribute to generate the output of the circuit (functional gates). Functional gates correspond to the gate that constitute the output of the entire circuit and the gates that project connections to the output directly or indirectly, through other gates. Given that in our experimental setup genes encode the logic function performed by each gate and the way in which the gate is wired to the other gates, this measure also indicates the number of genes encoding functional properties of the system. The term phenotypic complexity does not have a unique definition (see for example see Adami 2002; Carlson and Doyle 2002; Crutchfield and Görnerup 2006; Hazen et al. 2007; Whitacre 2010). Moreover, phenotypic complexity is hard to define formally and to measure in the case of systems composed by heterogeneous elements and/or displaying multi-level organizations. The possibility to use a relatively straightforward measure in the case of our experiments, therefore, is due to the utilization of simple feed-forward digital circuits composed by homogeneous elements.

**Phenotypic variability**, defined as the propensity of circuits to vary phenotypically as a result of mutations, is measured (following Raman and Wagner 2011) by calculating the number of unique functions computed by circuits located in the genetic neighborhood of the original

| Stochasticity | MutRate 0.01 | MutRate 0.02 | MutRate 0.03 | MutRate 0.04 |
|---------------|--------------|--------------|--------------|--------------|
| Stochasticity 0.00 | 46.66 (37.80) | 43.33 (33.91) | 33.33 (30.73) | 40.00 (32.65) |
| Stochasticity 0.01 | 46.66 (39.11) | 53.33 (35.83) | 36.66 (34.01) | 33.00 (33.41) |
| Stochasticity 0.02 | 30.00 (37.25) | 46.66 (37.53) | 43.33 (35.46) | 30.00 (32.48) |
| Stochasticity 0.03 | 43.33 (39.89) | 33.33 (38.25) | 53.33 (38.40) | 40.00 (35.62) |
| Stochasticity 0.04 | 30.00 (39.34) | 50.00 (36.24) | 30.00 (37.78) | 40.00 (38.74) |
| Stochasticity 0.05 | 50.00 (41.81) | 56.66 (42.87) | 53.33 (38.59) | 46.66 (37.95) |
| Stochasticity 0.06 | 43.33 (40.91) | 43.33 (36.69) | 50.00 (33.61) | 33.33 (33.11) |
| Stochasticity 0.07 | 30.00 (35.94) | 33.33 (34.34) | 30.00 (38.63) | 23.33 (37.23) |

Bold value indicates the best result obtained varying the parameters reported in tables

\(\mu\) was set to 20. The first number of each cell indicates the fraction of replications that achieved optimal performance. The numbers between parentheses indicate the average size of the evolved circuits

---

© Springer
circuit. This number is estimated performing for 10 times a 1000 steps function-preserving random walk from the original circuit. The random walk is realized by: (i) generating a mutated circuit, (ii) incrementing a counter if the varied circuit computes a function that differs from the functions computed by the original circuit and by previous varied circuits, (iii) preserving or removing the mutation depending on whether or not the varied circuit has the same fitness of the original circuit, (iv) repeating the previous three operations for 1000 steps.

2.4 Replicability

Readers might replicate all the experiments described in this paper by downloading and installing FARSA from “https://sourceforge.net/projects/farsa/” and the required experimental plugin and configurations files from http://laral.istc.cnr.it/res/complexevol/EvenParity.zip.

3 Results

In this section we describe the results obtained and the analysis performed.

3.1 Digital circuits evolved with the \((\mu + \mu)\) ES

With the best combination of parameters (MutRate = 0.02, Stochasticity = 0.05, see Table 1), evolving circuits find optimal solutions in 56.66% of the replications.

The analysis of evolved circuits indicates that, overall, they are very robust with respect to mutations. Indeed, 35.12% of the offspring of the circuits evolved with the optimal parameters have a fitness equal or greater than their parent and 95.15% of the mutations are neutral (i.e., do not alter the fitness). This high robustness, however, is largely due to the fact that the size of the functional part of the evolving circuits is small, i.e., only about 10% of the 400 gates actively contribute to the generation of the outputs of the circuits (see Table 1). Indeed, by restricting the analysis to the mutations...
that affect the functional gates, the percentage of neutral mutations drops from 95.15 to 0.52%.

As shown in Fig. 2, the phenotypical complexity of the circuits, defined as the number of gates that contribute to generate the outputs of the circuits, increases during the initial phase of the evolutionary process (during which the fitness of the evolving circuits increases) and decreases during the successive phase (in which the fitness of the evolving circuits remains stable). This is the result of combined effect of the complexification of the individuals’ phenotype driven by adaptive variations and the simplification of the individuals’ phenotype driven by neutral variations.

The complexification originates as a consequence of the strong correlation between the fitness and the functional size of the circuits (Fig. 3, Spearman Test, rho 0.6359, phi < 10^{-20}, n = 600). The selection of circuits fitter than their ancestors, in fact, leads to the selection of circuits that are phenotypically more complex than their ancestors, on average.

The simplification of the individuals’ phenotype, instead, is caused by two factors. First, the probability to generate offspring with less functional gates is significantly higher than the probability to generate offspring with more functional gates (Wilcoxon ranksum test, p-value < 10^{-9}). Data obtained by analyzing the offspring of the 30 best circuits evolved in 30 replications performed with the (µ + µ) ES algorithm. For the purpose of this analysis, each circuit was allowed to generate 10^4 offspring. Secondly, as stated above, the competition between reproducing individuals leads to the selection of individuals that are robust with respect to genetic variations.

The fact that the phenotypic complexity increases during the initial phase of the evolutionary process in which the fitness of the evolving individuals also increases and then decreases during the successive phase is thus due to the fact that the complexification factor operates during the initial evolutionary phase only, while the simplification factor operates during the entire evolutionary process, especially during neutral evolutionary phases.

**Table 2** Performance of circuits evolved with the (1 + µ) ES in experiments carried out by using different mutation rate and stochasticity level

| MutRate | Stochasticity | N. offspring (λ) |
|---------|---------------|------------------|
| 0.01    | 0.00          | 96.66 (43.84)    | *[408,400] (44.54) |
| 0.02    | 0.02          | *[856,070] (48.91) | *[348,600] (42.38) |
| 0.03    | 0.04          | *[544,900] (42.76) | *[549,700] (42.49) |
| 0.04    | 0.06          | 86.66 (49.53)    | *[463,300] (40.54) |
|         |               | 43.33 (34.21)    | 0 (25.77)          |

Bold value indicates the best result obtained varying the parameters reported in tables.

| N. offspring (λ) | MutRate 0.01 | MutRate 0.02 | MutRate 0.03 | MutRate 0.04 |
|------------------|--------------|--------------|--------------|--------------|
| 5                | *[408,400]   | *[348,600]   | *[395,400]   |               |
| 10               | *[544,900]   | *[549,700]   | *[463,300]   |               |
| 20               | *[524,900]   | 96.66 (43.11)| 96.66 (39.96)|               |

**Table 3** Performance of circuits evolved with the (1 + λ) ES in experiments carried out by using different number of offspring (λ)

| N. offspring (λ) | MutRate 0.01 | MutRate 0.02 | MutRate 0.03 | MutRate 0.04 |
|------------------|--------------|--------------|--------------|--------------|
| 5                | *[408,400]   | *[348,600]   | *[395,400]   |               |
| 10               | *[544,900]   | *[549,700]   | *[463,300]   |               |
| 20               | *[524,900]   | 96.66 (43.11)| 96.66 (39.96)|               |

Bold value indicates the best result obtained varying the parameters reported in tables.

The mutation rate and stochasticity parameters were set to 0.03 and 0.0, respectively. All experiments found optimal solutions. The numbers indicated with square brackets represent the average number of evaluations that were necessary to find optimal solutions. The numbers between parentheses indicate the average size of the evolved circuits.
The presence of a strong positive correlation between the size and the phenotypic variability of the circuits (Fig. 5, Spearman Test, rho 0.8897, phi < 10^{-204}, n = 600) implies that the tendency to select phenotypically simple circuits leads to the selection of low evolvable individuals. In other words, the competition between reproducing individuals drives the evolving population toward a highly robust but low evolvable area of the search space. This area corresponds to the central and most connected region of the neural network in which the fraction of mutations that does not alter the fitness of the circuits is maximum (Wilke 2001), but in which the fraction of mutations that gives rise to different unique phenotypes is minimum.

A correlation between the size and the phenotypic variability of circuits has already been reported by Raman and Wagner (2011). In their case, the correlation was observed by comparing randomly generated circuits of different size that computed the same logic function. The correlation, therefore, seems to characterize all circuits, irrespectively of whether or not they were evolved and irrespectively of the function computed.

### 3.2 Digital circuits evolved with the (1 + λ) ES

With the best combination of parameters (MutRate = 0.02, Stochasticity = 0.05, see Tables 2, 3), evolving circuits find optimal solutions in all replications after 301400 evaluations, on the average. This performance is significantly better than the performance of the circuits evolved with the (μ + 1) ES (Wilcoxon ranksum test, p-value < 10^{-121}).

These results are in line with those reported in previous studies (Miller et al. 2000) and confirm the efficacy of the (1 + λ) ES in this domain. Notice also how, in the case of this method, the optimal results are achieved without Stochasticity, see Table 2. The experiments reported in Miller, Job, and Vassiley (2000) were also carried out without Stochasticity (the method used did not include the possibility to reduce the selection pressure through the addition of noise to the fitness).

At this point we should try to explain why the (1 + λ) ES largely outperforms the (μ + μ) ES.

The variability of the population is null in the case of (1 + λ) ES. Consequently, the superiority of this method cannot be due to the diversity of the population. The advantage is rather explained by the fact that this method leads to the selection of circuits that are less robust, but that have a greater phenotypic variability than the circuits evolved with the (μ + μ) ES (Table 4).

The lower performance of the circuits evolved with the (μ + μ) ES, on the other hand, is due to the fact that it drives the population toward a very robust but low evolvable region of the genetic space. As we pointed out above, the competition between evolving circuits, in the experiments carried out with the (μ + μ) ES, drives the population toward solutions that are very robust to genetic variations. Since the easiest way to achieve robustness consists in selecting solutions that are phenotypically simple and have a low phenotypic variability, the selection of solutions that are robust to mutations produces a reduction of phenotypic variability that, in turn, causes evolutionary stagnation.

| Table 4 | Comparison of the characteristics of the first circuits evolved with the (μ + μ) and the (1 + λ) ES that achieved optimal performance (i.e. the 17 out of 30 replications that achieved optimal performance in the case of the (μ + μ) experiments and 30 out of 30 replications in the case of the (1 + λ) experiments) |  |
| --- | --- | --- |
|  | (μ + μ) | (1 + λ) | Wilcoxon ranksum p-value |
| Phenotypic complexity (number of functional gates) | 61.0 | 61.8 | > 0.05 |
| Phenotypic variability | 3826.3 | 4826.0 | < 10^{-4} |
| Robustness (% offspring with a fitness equal or greater than the parent) | 23.7565 | 14.3 | < 10^{-40} |

| Table 5 | Performance of circuits evolved with the PSHC method in experiments carried out by varying mutation rate and stochasticity |
| --- | --- | --- | --- | --- | --- |
| MutRate 0.01 | MutRate 0.02 | MutRate 0.03 | MutRate 0.04 | MutRate 0.05 |
| Stochasticity 0.0 | *[413,270] (54.51) | *[275,300] (50.12) | *[361,730] (51.82) | *[692,800] (47.83) | 83.33 (42.88) |
| Stochasticity 0.01 | *[516,270] (50.43) | *[568,800] (56.36) | *[815,800] (52.96) | *[853,730] (50.26) | 80 (46.36) |
| Stochasticity 0.03 | *[537,600] (49.52) | *[798,630] (47.70) | *[809,450] (46.93) | 96.66 (45.20) | 83.33 (43.67) |
| Stochasticity 0.05 | *[1,024,530] (47.95) | *[1,065,420] (54.19) | *[1,160,500] (46.94) | *[1,160,500] (51.29) | 93.33 (51.29) | 73.33 (48.42) |

Bold value indicates the best result obtained varying the parameters reported in tables.
The fact that the $(1 + \lambda)$ ES is able to find optimal solutions in all replications without stochasticity (see Table 2) indicates that the stagnation phases affecting the circuits evolved with the $(\mu + \mu)$ ES is not caused only by the characteristics of the fitness surface (i.e., by the presence of local minima), but rather by the combined effect of the fitness surface and the tendency of the population to move toward areas of the genetic space containing highly robust but low-evolvable solutions.

We can schematize this process by considering all the candidate solutions with the same fitness as a series of nodes and the genetic variations transforming parents into offspring with the same fitness as links between the nodes. The nodes and the links form one or more neutral networks (Schuster et al. 1994; Van Nimwegen et al. 1999) constituted by connected candidate solutions. Robustness is higher in the central part of the network in which the number of connections between nodes is maximum and lower in peripheral parts of the network. On the other hand, phenotypic variability is lower in the central part of the network and higher in the periphery (see also Hu et al. 2012). Stagnation thus originates from the tendency of circuits evolved with the $(\mu + \mu)$ ES to move toward the central portion of the neutral network, which has a low phenotypic variability.

### 3.3 Digital circuits evolved with the PSHC algorithm

Table 5 shows the results of the experiment carried out by evolving circuits with the Parallel Stochastic Hill Climber (PSHC) algorithm. To analyze the role of the critical parameters, we carried out 20 series of experiments in which we systematically varied the mutation rate and stochasticity. The Variations parameter was set to 100 and the size of the population was set to 20. As can be see the best results are obtained without stochasticity and with the mutation rate parameter set to 0.02. Table 6 shows the results obtained in 7 additional series of experiments in which the Variations parameter was varied and the stochasticity and mutation rate parameters were set to 0.0 and 0.02, respectively (i.e., to the optimal values).

As can be seen, the PSHC algorithm allows the evolving circuits to find optimal solutions in all replications and, in the case of the best parameters (MutRate = 0.02 and Stochasticity = 0.0, Variations = 100), after only 275,300 evaluations. The results obtained with the PSHC algorithm are significantly better than the results obtained with $(\mu + 1)$ and $(1 + \lambda)$ algorithms reported above (Kruskal Wallis test p-value < $10^{-11}$).

As reported above, we the experiments carried out with a variation of the PSHC algorithm in which the population was divided in into 10, 5, or 2 sub-populations formed by 2, 4 or 10 individuals led to much worse results. Indeed, these experiments achieved optimal performance in only 84%, 62%, and 54% of the replications, respectively. This since the competition among individuals of

| Variations | 1 | 10 | 50 | 100 | 150 | 200 | 500 |
|------------|---|----|----|-----|-----|-----|-----|
|            | *|[710′300]|93.33|[764′200]|*|[275′300]|*|[275′300] |

 Bold value indicates the best result obtained varying the parameters reported in tables.

The mutation rate and stochasticity parameters were set to 0.2 and 0.0, respectively. The number of parents ($\lambda$) was set to 20. The first number of each cell indicates the fraction of replications that achieved optimal performance. The experiments in which all replications found optimal solutions are indicated with an asterisk followed by the average number of evaluations that were necessary to find optimal solutions indicated with square brackets. The numbers between parentheses indicate the average size of the evolved circuits.

### Table 6 Performance of circuits evolved with the PSHC method in experiments carried out by with different number of variations

| Variations | 1 |
|------------|---|
|            | *|[710′300] |

| 10         |
|------------|
| 93.33      |

| 50         |
|------------|
| *|[764′200] |

| 100        |
|------------|
| *|[275′300] |

| 150        |
|------------|
| *|[454′200] |

| 200        |
|------------|
| *|[398′410] |

| 500        |
|------------|
| *|[584′650] |

### Table 7 Comparison of the characteristics of the first circuits evolved with different algorithms that achieved optimal performance (i.e. the 17 out of 30 replications that achieved optimal performance in the case of the $(\mu + \mu)$ experiments and the 30 out of 30 replications that achieved optimal performance in the case of the $(1 + \lambda)$ and PSHC experiments)

|                  | $(\mu + \mu)$ | $(1 + \lambda)$ | PSHC | Kruskal Wallis p-value |
|------------------|---------------|-----------------|------|-----------------------|
| Phenotypic complexity (number of functional gates) | 61.0 | 61.8 | 71.6333 | 0.0081 |
| Phenotypic variability | 3826.3 | 4826.0 | 4198.5 | 0.0063 |
| Robustness (% offspring with a fitness equal or greater than the parent) | 23.75 | 14.3 | 18.79 | 0.0027 |
The comparison between the first circuits that achieved optimal performance (Table 7) and the optimal circuits obtained at the end of the evolutionary process (Table 8) shows how the evolving circuits tend to become progressively more robust with respect to genetic variations during phases in which the fitness remain stable. Indeed, the percentage of offspring that preserve the same fitness of their parents increases significantly in all cases (Wilcoxon ranksum test, p-value < 10^{-15}). The increased robustness is achieved by reducing the phenotypic complexity and, consequently, by reducing the phenotypic variability.

Overall, these results indicate that the advantage of the PSHC in this problem domain is due to its ability to limit the effects of the competition between reproducing individuals, that favours the selection of phenotypically simple individuals characterized by low phenotypic variability, and to its ability to maximize the variability of the population.

4 Conclusion

In this paper we investigated how the characteristics of the evolutionary algorithm influence the evolvability of candidate solutions, i.e. the propensity of evolving individuals to generate better solutions as a result of random genetic variations. This objective has been pursued by evolving digital circuits which represent a classic domain of application for evolutionary algorithms and which share important properties with natural systems such as proteins, RNA, regulatory circuits and metabolic networks.

The results indicate that in this domain, which is characterized by a high level of neutrality, (1 + λ) ES largely outperform (μ + μ) ES. The analysis of the evolutionary dynamics indicates that this difference is due to competition for robustness to mutations among evolving individuals. When individuals compete for robustness and when robustness is achieved simply by minimizing the number of genes playing a functional role, as in the case of the circuits evolved with the (μ + μ) ES, evolution tends to select individuals located in high robust regions of the genetic space which are characterized by low phenotypic variability and, consequently, low evolvability. When the evolving individuals

|                      | (μ + μ) | (1 + λ) | PSHC | Kruskal Wallis p-value |
|----------------------|---------|---------|------|------------------------|
| Phenotypic complexity (number of functional gates) | 48.94   | 44.83   | 43.86 | > 0.05                 |
| Phenotypic variability (individual)               | 2617.7  | 2211.1  | 2418.3| 0.0375                 |
| Robustness (% offspring with a fitness equal or greater than the parent) | 33.7    | 27.34   | 39.80 | 0.0058                 |

Data calculated by using the first optimal individual circuit of the last generation. In the case of the (μ + μ) experiments, we included only the replications that achieved optimal performance.
do not compete for robustness to mutations, as in the case of the circuits evolved with the $(1+λ)$ ES, evolution selects individuals that are less robust but have a higher phenotypic variability and evolvability. Overall, these results show how neutrality and robustness to mutations do not necessarily enhance evolvability. They can also reduce evolvability and cause evolutionary stagnation, as also observed by Ancel and Fontana (2000).

Whether robustness to mutations enhances or diminishes phenotypic variability and evolvability depends on whether robustness is achieved through the development of parsimonious (phenotypically simple) solutions minimizing the number of genes playing functional roles, or through phenotypically more complex solutions capable of buffering the effect of mutations. Robustness to mutations of the latter type can evolve as a correlated side effect of the evolution of robustness to environmental variations (De Visser et al. 2003), which cannot be improved through phenotypic simplification. Therefore, whether robustness to mutation enhancement or reduce evolvability might depend on whether or not robustness to mutation is combined with robustness to environmental variations. For evidences supporting this hypothesis in the context of evolving digital circuits, see Milano and Nolfi (2016).

The comprehension of the effects that the competition between evolving individuals has on robustness and evolvability allowed us to design a new evolutionary algorithm, named Parallel Stochastic Hill Climber (PSHC), which outperforms the other two methods considered. This is achieved by limiting the negative effects that the competition among evolving individuals can have on evolvability while preserving the advantage provided by the utilization of a diversified population.

The analysis reported in this paper has been restricted to algorithm operating on the basis of asexual reproduction. The impact of sexual reproduction and of other genetic operators on the evolutionary dynamics should be investigated in future studies.

References

Abdelhalim L, Blachon S, Selbig J, Nikolosky Z (2011) Robustness of metabolic networks: a review of existing definitions. Biosystems 106(1):1–8

Adami C (2002) Sequence complexity in Darwinian evolution. Complexity 8:49–57

Ancel LW, Fontana W (2000) Plasticity, evolvability, and modularity in RNA. J Exp Zool Part B Mol Dev Evol 288:242–283

Back T (1994) Selective pressure in evolutionary algorithms: a characterization of selection mechanisms. In: Proceedings of 1st IEEE conference evolutionary computation, Jun. 27–29, 1994, pp 57–62

Bäck T, Hammel U (1994) Evolution strategies applied to perturbed objective functions. In Proceedings of the international conference on evolutionary computation. pp 40–45

Balch M (2003) Complete digital design. McGraw-Hill, New York

Bedau MA, Packard NH (2003) Evolution of evolvability via adaptation of mutation rates. Biosystems 69:143–162

Beyer HG, Schwefel HP (2002) Evolution strategies—a comprehensive introduction. Nat Comput 1(1):3–52

Carlson JM, Doyle J (2002) Complexity and robustness. PNAS 99:2538–2545

Crutchfield JP, Görenerup O (2006) Objects that make objects: the population dynamics of structural complexity. J R Soc Interface 3:345–349

De Visser JA et al (2003) Perspective: evolution and detection of genetic robustness. Evolution 57(9):1959–1972

Earl DJ, Deem MW (2004) Evolvability is a selectable trait. PNAS 101:11531–11536

Edelman GM, Gally JA (2001) Degeneracy and complexity in biological systems. Proc Natl Acad Sci USA, 98(13):763–768

Frei R, Whitacre J (2012) Degeneracy and networked buffering: principles for supporting emergent evolvability in agile manufacturing systems. Nat Comput 11(3):417–430

Hartmann M, Haddow P (2004) Evolution of fault tolerant and noise-robust digital designs. IEE Proc Comput Digit Tech 151:287–294

Hazen RM, Griffin PL, Carothers JM, Szostak JW (2007) Functional information and the emergence of biocomplexity. Proc Natl Acad Sci 104:8574–8581

Houle D (1992) Comparing evolvability and variability of quantitative traits. Genetics 130:195–204

Hu T, Payne JL, Banzhaf W, Moore JJ (2012) Evolutionary dynamics on multiple scales: a quantitative analysis of the interplay between genotype, phenotype, and fitness in linear genetic programming. Genet Program Evolvable Mach 13(3):305–337

Jin Y, Branke K (2005) Evolutionary optimization in uncertain environments: a survey. IEEE Trans Evol Comput 9(3):303–317

Kirschner M, Gerhart J (1998) Evolvability PNAS 95:8420–8427

Levitan B, Kauffman S (1994) Adaptive walks with noisy fitness measurements. Mol Divers 1(1):53–68

Macia J, Solé RV (2009) Distributed robustness in cellular networks: insights from synthetic evolved circuits. J R Soc Interface 6(33):393–400

Masel J, Trotter MV (2010) Robustness and evolvability. Trends Genet 26(9):406–414

Milano N, Nolfi S (2016) Robustness to faults promotes evolvability: insights from evolving digital circuits. PLoS ONE 11(7):e0158627

Miller J, Hartmann M (2001) Evolving messy gate for fault tolerance: some preliminary findings. In: Proceedings 3rd NASA workshop on evolvable hardware, pp 116–123

Miller JF, Thomson P (2000) Cartesian genetic programming. In: Poli R, Banzhaf W, Langdon WB, Miller J, Nordin P, Fogarty TC (eds) Lecture Notes in Computer Science 1802 Genetic programming. Springer, Heidelberg

Miller JF, Job D, Vassilevy VK (2000) Principles in the evolutionary design of digital circuits. J Genet Progr Evolv Mach 1(1):8–35

Miller JF, Thomson P (2000) Cartesian genetic programming. In: Proceedings of the third european conference on genetic programming (EuroGP), vol 1820. Springer, Berlin, pp. 121–132

Miller JF, Thompson A, Thomson P, Fogarty T (eds) Lecture Notes in Computer Science 1802 Genetic programming. Springer, Heidelberg

Rivera D, Taborsky M (2003) Principles of supporting emergent evolvability in agile manufacturing systems. Proc Natl Acad Sci USA, 98(13):763–768

Rana S, Whitled LD, Cogswell R (1996) Searching in the presence of noise. In: Voigt HM (ed) Parallel problem solving from nature. Springer
Evolutionary Intelligence (2019) 12:83–95

Lecture Notes in Computer Sciences, 1141. Springer, Berlin, pp 198–207

Rechenberg I (1973) Evolutionstrategie—Optimierung technischer Systeme nach Prinzipien der biologischen Evolution. Frommann-Holzboog, Stuttgart

Schuster P, Fontana W, Stadler PF, Hofacker IL (1994) From sequences to shapes and back: a case study in RNA secondary structures. Proc R Soc Lond B 255:279–284

Sniegowski PD, Murphy HA (2006) Evolvability Curr Biol 16:831–834

Sekanina L (2004) Evolvable computing by means of evolvable components. Nat Comput 3(3):253–292

Thompson A, Layzell P, Zebulum R (1999) Explorations in design space: unconventional electronics design through artificial evolution. IEEE Trans Evol Comput 3(3):167–196

Tononi G, Sporns O, Edelman GM (1999) Measures of degeneracy and redundancy in biological networks. Proc Natl Acad Sci USA 96:3257–3262

Van Nimwegen E, Crutchfield JP, Huynen M (1999) Neutral evolution of mutational robustness. PNAS 96:9716–9720

Wagner A (2008) Robustness and evolvability: a paradox resolved. Proc R Soc B 275:91–100

Wagner A (2011) The origins of evolutionary innovations: a theory of transformative change in living systems. Oxford University Press, Oxford

Wagner GP, Altenberg L (1996) Perspective: complex adaptations and the evolution of evolvability. Evolution 50:967–976

Whitacre JM (2010) Degeneracy: a link between evolvability, robustness and complexity in biological systems. Theor Biol Med Model 7:6

Whitley D, Rana S, Heckendorn RB (1998) The island model genetic algorithm: on separability, population size and convergence. J Comput Inf Technol 7:33–47

Wilke CO (2001) Adaptive evolution on neutral networks. Bull Math Biol 63:715–730

Publisher’s Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.