2013 IEEE Congress on Evolutionary Computation
June 20-23, Cancún, México

Symmetry in Evolutionary and Estimation of Distribution Algorithms

Roberto Santana
Intelligent Systems Group
University of the Basque Country
San Sebastian, Guipuzcoa
Spain
Email:roberto.santana@ehu.es

R. I. (Bob) McKay
School of Computer Science
& Engineering
Seoul National University
Seoul, Korea
Email: rimsnucse@gmail.com

Jose Antonio Lozano
Intelligent Systems Group
University of the Basque Country
San Sebastian, Guipuzcoa
Spain
Email:ja.lozano@ehu.es

Abstract—**Symmetry has hitherto been studied piecemeal in a variety of evolutionary computation domains, with little consistency between the definitions. Here we provide formal definitions of symmetry that are consistent across the field of evolutionary computation. We propose a number of evolutionary and estimation of distribution algorithms suitable for variable symmetries in Cartesian power domains, and compare their utility, integration of the symmetry knowledge with the probabilistic model of an EDA yielding the best outcomes. We test the robustness of the algorithm to inexact symmetry, finding adequate performance up to about 1% noise. Finally, we present evidence that such symmetries, if not known a priori, may be learnt during evolution.**

I. Introduction

Symmetry is ubiquitous in nature – and in evolutionary computation (EC). It has been theoretically and experimentally studied in a number of ground-breaking papers over the past 15 years [1]–[7]. Yet it is noteworthy that the focus has been on Cartesian power spaces, and within Cartesian power spaces, on a specific form of symmetry, value symmetry. Unfortunately this has not always been clearly delineated. Most of the preceding papers present results of greater or lesser generality about symmetry. Yet none of them cover even the variable symmetry of Cartesian power spaces analysed by Munteanu and Rosa in [8], let alone the symmetries in the wider domains of EC – as different as permutation spaces or the unbounded function spaces of Genetic Programming (GP). In these domains, the important symmetries have rather more resemblance to variable symmetries than to value symmetries. This has the potential to lead to confusion about the role of symmetry in EC.

Much of the preceding literature [1], [3], [4], [6], [7] has focused on the difficulties that value symmetries pose for evolutionary, and specifically for Genetic Algorithms (GAs). Others [2], [5] have noted the relevance of estimation of distribution algorithms (EDAs) [9], on that basis that explicit probability models and explicit representation of symmetries may aid in overcoming these difficulties. Of course, this intuition may extend to variable symmetries.

In this paper, we aim to:

1) Clarify some of the meanings of the term ‘symmetry’, and their relevance to different evolutionary domains
2) Investigate whether specific forms of variable symmetry may be useful (rather than problematic) for optimisation
3) Experimentally investigate the performance of EDAs on variable symmetries in Ising problem spaces
4) Test the feasibility of learning variable symmetries

Section II introduces the background for this paper: defining symmetry formally, and discussing different kinds that may be relevant in EC. It introduces the Ising test problems, and briefly treats EDAs. Section III describes the algorithms we compare, the specific test instances, and the detailed parameter settings, the results being presented in section IV. Section V discusses the wider implications of the results, while section VI summarises the content and gives directions for the future.

II. Background and Theory

A. Mathematically Defining Symmetry

EC covers a wide range of domains, from the Cartesian power spaces of typical GAs, through permutation and other more complex domains, to the unbounded function domains of GP. If we are to study symmetry in a uniform framework across these wide-ranging domains, we need correspondingly general definitions.

For a domain D, a symmetry is any one-to-one and onto (bijective) mapping $\{\varsigma : D \to D\}$, also known as an automorphism. This definition generates a rich algebraic structure: the symmetry group of all automorphisms, $\mathfrak{S}_D$. However for EC, it is both too narrow and too general.

It is too narrow in not covering many closely related mappings in one important evolutionary domain, GP. Identities such as commutativity generate automorphisms. However other identities are perhaps more important (because of their role in bloat): e.g. idempotence ($x + 0 = x$) and inverse ($x - x = 0$). They generate mappings from the domain of expressions *onto* itself, but they are not *one-to-one*.

The set of onto mappings do not seem to have been heavily studied to date, perhaps because for finite domains they collapse to symmetries. Nevertheless, they have deep algebraic structure, forming a regular monoid. We plan to return to detailed study of these mappings in future work.

The definition is also too general because it considers only the set structure of the domain. In EC, we usually have more structure; it is usual to restrict attention to symmetries that preserve that structure. For example, in permutation spaces, one may restrict attention to rotations (preserving order), and/or reflections (preserving adjacency). More narrowly, much of EC focuses on Cartesian powers of simple domains ($\mathbb{R}^n, \mathbb{Z}^n, \{0,1\}^n$ and so on). We consider these next.

1) Symmetries in Cartesian Power Spaces: In a Cartesian power space A^n, there are two ways that symmetries can commute with the product structure:

1) Value symmetries are defined in terms of symmetries on the base domains. Every symmetry on the base space $\{\varsigma : A \to A\}$ generates a corresponding symmetry $\{\varsigma^n : A^n \to A^n\}$ defined by $\varsigma^n(a_1, \ldots, a_n) = (\varsigma(a_1), \ldots, \varsigma(a_n))$.
2) Variable symmetries map values between variables without affecting values. Each symmetry $s : \{1, \ldots, n\} \to \{1, \ldots, n\}$ of the indices of the Cartesian product generates a corresponding symmetry $\{\varsigma_s : A^n \to A^n\}$ defined by $\varsigma_s(a_1, \ldots, a_n) = (a_{s(1)}, \ldots, a_{s(n)})$. In this paper, we concentrate our attention on variable symmetries. We frame the discussion around symmetries exchanging pairs of subsets, but the discussion is readily extended to general variable symmetries.

2) Symmetries and Fitness: In optimisation, we are only interested in symmetries to the extent that they may help or hinder us in solving a problem – and that requires them to have some relationship with the fitness function. Specifically, given a fitness function $F : D \to \mathbb{R}$ (or more generally, a fitness ordering $\prec_F \subseteq D \times D$), we are interested in symmetries ς that preserve fitness, i.e. $F(\varsigma(x)) = F(x)$ (or in general, $(\varsigma(x) \prec_F \varsigma(y)) \equiv (x \prec_F y)$).

3) Inexact Symmetries: In real life, fitness symmetries are often not exact (or even if they are exact, fitness evaluation may not be). That is, instead of $F(\varsigma(x)) = F(x)$ we have $F(\varsigma(x)) \approx F(x)$. Approximate symmetries have been studied for some time [10], [11, Chapter 6], but to not seem to have been previously studied in EC.

B. Symmetries in Evolutionary Computation

1) Value Symmetries: Naudts and Naudts [1] were the first to note the difficulties for GAs induced by value symmetries. A number of papers have continued this analysis, aiming to determine the underlying source of such problems [1], [3], [4], [6], [7]. Pelikan and Goldberg [2] and Peña et al. [5] investigated the use of EDAs on such problems, while a number of papers such as [12] and [13] have proposed specific genetic operators for specific symmetric problem domains.

2) Variable Symmetries: Variable symmetries appear to have been little studied in previous EC literature. The only explicit consideration we are aware of is that of Munteanu and Rosa [8], [14] who defined a form of variable symmetry limited to reflection, and discussed the ability of the inversion operator to deal with it. A number of other proposals for new operators – transposition [15], translocation [16], [17] – have been proposed for reasons of biological analogy or specific problem relevance, but are directly related to variable symmetry.

3) Other Uses of Symmetry: One other application of symmetry, not falling at all within Cartesian domains, comes from Stanley et al. [18]. They argued that neural network architectures evolved to contain symmetries[1] can solve problems with large numbers of inputs with these properties. HyperNEAT efficiently evolves very large neural networks that more resemble the neural connectivity patterns in the brain.

C. Estimation of Distribution Algorithms

In the empirical investigation of section IV, we use Tree-EDA, an evolutionary algorithm based on probabilistic trees, first introduced in [19]. This probabilistic model exhibits a good balance between complexity and efficiency. The bivariate dependencies it learns can be sufficient to solve many problems with interactions, and their computational expense is lower than in methods learning higher order dependencies.

D. Ising Problems

Ising problems are closely related to the Ising model originally defined by Lenz and Ising [20]. In general, an Ising problem consists of a directed graph $G = (V, E)$, in which every edge $e = (e_{\text{in}}, e_{\text{out}})$ has a permanently assigned interaction weight J_e. A configuration σ consists of an assignment to every vertex v of a spin $\sigma_v \in \{-1, 1\}$. There is also an external magnetic field at each vertex of magnitude h_v. The quantity of interest is the Hamiltonian

$$H(\sigma) = -\sum_{e \in E} J_e \sigma_{e_{\text{in}}} \sigma_{e_{\text{out}}} - \sum_{v \in V} h_v \sigma_v \qquad (1)$$

In an Ising problem, the aim is to maximise the value of the function H.[2] As is common, we restrict our attention to the case where the external magnetic field is zero, interactions are restricted to $\{-\frac{1}{2}, \frac{1}{2}\}$, and G is a finite regular square n-dimensional toroidal lattice with forward and backward links in each dimension. Since the edges are symmetric except for their weights, the total interaction contribution between vertices v_1 and v_2 is either -1, 0 or 1, so we can consider G as an undirected graph with weights drawn from $\{-1, 0, 1\}$. In the practical experiments, we will further restrict to the two-dimensional case and require that left-right and up-down interactions are equal, or in undirected graph terms, the possible interactions are drawn from $\{-1, 1\}$.

Two-dimensional Ising problems are particularly suitable for investigation of symmetries (and their interactions with modularity) because

1) In general, they are known to be NP-hard, so that problems of any required difficulty are available
2) 2D problems and solutions are readily visualised
3) Variable interactions are explicitly represented, they do not need to be deduced

[1]Although the authors do not designate all these regularities as symmetries, they fall under the more mathematical definition used here.

[2]Due to the $-$ sign in the Hamiltonian, an edge e with $J_e > 0$ will make a positive contribution to H if $\text{sign}(\sigma_{e_{\text{in}}}) \neq \text{sign}(\sigma_{e_{\text{out}}})$, and vice versa.

4) They have been intensively studied, so that there is a vast background literature to draw on

E. Variable Symmetries and Modularity

Variable symmetries are closely related to modularity; we consider here three formulations: additive separability, additive decomposability [21] and ϵ-modularity [22], [23]. None of the fitness functions we consider are additively separable, since there are interactions between every adjacent pair of variables – characteristic of Ising models with nonzero interactions. They are all additively decomposable, another general characteristic of Ising models, so this does not help to distinguish symmetric from asymmetric. However the relationship to Watson's more general ϵ-modularity [22], somewhat more rigorously defined in De Jong's [23], is more complex. After all, how could we use, in optimisation, the knowledge that two sets of variables $\vec{V} = \{V_1, \ldots, V_r\}$ and $\vec{W} = \{W_1, \ldots, W_r\}$ are symmetric? In an EDA, the obvious answer is that we know the two sets of variables follow the same distribution, since we can exchange their values without changing fitness. But in practice, this is only likely to be useful if we have a factorisation of the domain that allows us to separately and explicitly represent their distributions. And we are only likely to have a suitable such factorisation if V (and hence by definition W) are separate ϵ-modules.

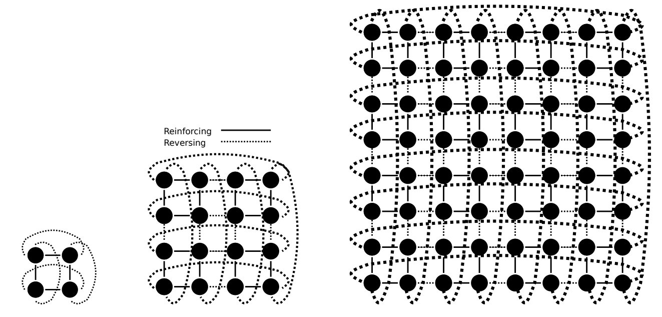
It might thus seem that variable symmetries contribute nothing more than ϵ-modularity. However this is a misconception, especially for EDAs. We can see this from the results in section IV, but also from a thought experiment about Ising lattices. By construction, local dependences in Ising lattices are stronger than longer-distance dependences, and increasingly so as the size of the neighbourhood increases. For example in random 2D lattices, the number of interconnections within a square region of diameter d increases as $2(d-1)^2$, while the number of connections to its exterior increases only as $4d$, and with a linearly more spread out (hence more disconnected) set of neighbours. Hence as d increases, the square is increasingly likely to be or contain an ϵ neighbourhood. The same argument applies to blocks of ϵ neighbourhoods, so that we automatically have a hierarchy of such neighbourhoods. The problem is that these neighbourhoods overlap, so that there are combinatorially many ways of forming hierarchies of neighbourhoods (and combinatorially many ways to start from the wrong decomposition, so that the ϵ-neighbourhoods do not nest properly over the whole space). The difficulty is how to choose, especially in algorithms that need to explicitly represent the dependence structure. If we know a symmetric decomposition of the problem, this immediately gives us guidance in structuring the problem. If we don't, symmetries may be easier to discover than modules.

In this context, what we gain from symmetry is:

1) Selection: Symmetries in Ising problems (and more generally) give use guidance as to which ϵ-modules to concentrate on.
2) Early Detectability: a small number of random samples, with exchanges of the values of V and W, can provide very high confidence that V and W are symmetric. In principle, symmetry should be detectable and able to be confirmed, early in search, whereas ϵ-modularity is only detectable (and hence explicitly usable) once search has approached within ϵ of the optimum.
3) Speedup: modularity of V and W allows us to learn them separately and in parallel; symmetry allows us to learn them together, using the same model, and hence further reduce computation time.

III. Methods

A. Test Problems



(a) Level 0 (b) Level 1 (c) Level 2

Fig. 1. **Recursive Levels of Ising Lattice Construction**

For the test problems, we used translation symmetries over random square toroidal 2-dimensional undirected $\{-1, 1\}$ Ising lattices. Each instance is recursively built up from a random initial lattice, the 'basic block' L_0, by a copying operation, and is characterised by four numerical values:

1) D_0: the diameter of L_0
2) N_C: the recursion count, the number of times the recursive operation – copying with translation – is applied in constructing the 'problem lattice'
3) N_F: the number of random flips applied to disrupt the symmetry of the (initially symmetric) problem lattice
4) S: the seed for the random number generator

In detail, L_0 is a square torus of horizontal and vertical connections (made toroidal by connecting the top row to the bottom, and the leftmost to the rightmost). It is characterised by the $2 \times D_0^2$ interconnections between horizontal and vertical neighbours, randomly sampled from $\{-1, 1\}$. The copying-with-translation operator is applied N_C times, starting with L_0. Each application consists of duplicating the whole of the previous lattice L_n both vertically and horizontally, creating a new lattice four times larger, then disconnecting the toroidal connections and re-using the same weights to re-connect the rightmost elements of the left lattice to the leftmost of the right (thus restoring the toroidal structure on the new, larger lattice L_{n+1}), with similar reconnections vertically. The operation is shown in detail in figure 1. It generates a lattice of size $D_0^2 \times 2^{2N_C}$. Finally (in noisy symmetry cases), N_F random locations in the recursively constructed lattice are flipped.

These test problems were chosen to satisfy $H_{L_{n+1}\max} \geq 4\times H_{L_n\max}$ (by replicating the optimal configuration of L_n). If we choose D_0 sufficiently small to reliably find $H_{L_0\max}$, we have a lower bound for the optimisation target for all problems. It is not an exact bound. Consider a 2*2 basic block L_0 in which the planar connections are -1 (reinforcing) and the toroidal connections are 1 (reversing) – figure 1a. All configurations σ have $H_\sigma = 0$ in L_0 (every pair of neighbours has a reinforcing planar and a reversing toroidal connection, which cancel). In its first replication L_1 – figure 1b – if all spins in the top left and bottom right copies are 1, and -1 in the other copies, all interactions make a positive contribution to H_σ. So this configuration has the maximum possible fitness, 32.[3]

B. Algorithms

1) Bayesian Tree Estimation of Distribution Algorithm: The core Bayesian Tree EDA is shown in Algorithm 1. Information about symmetry is incorporated, in our variants, in the evaluation phase (Step 4), the learning phase (Steps 6-9), and/or the sampling step (Step 10).

Algorithm 1: **Bayesian Tree EDA**

```
1  $D_0 \leftarrow$ Generate $M$ individuals randomly
2  $l = 1$
3  do {
4      Evaluate solutions in $D_{l-1}$
5      $D^s_{l-1} \leftarrow$ Select $N \leq M$ individuals from $D_{l-1}$
       according to a selection method
6      Compute the univariate and bivariate marginal
       frequencies $p^s_i(x_i|D^s_{l-1})$ and $p^s_{i,j}(x_i, x_j|D^s_{l-1})$ of
       $D^s_{l-1}$
7      Calculate the matrix of mutual information using
       bivariate and univariate marginals.
8      Calculate the maximum weight spanning tree from
       the matrix of mutual information.
9      Compute the parameters of the model.
10     $D_l \leftarrow$ Sample $M$ individuals (the new population)
       from the tree and add elitist solutions.
11 } until A stop criterion is met
```

We used four variants of the Bayesian Tree EDA to compare the effects of using the information about symmetries in more or less flexible ways. These variants require us to know in advance which are the symmetric components; they are readily extensible to other Cartesian power domains with variable symmetries. The variants investigated were:

1) EDA: No symmetry use: Tree-EDA as presented in Algorithm 1, evolving values for all $D_0^2 \times 2^{2N_C}$ variables.
2) EDA+basicblock: Trivial symmetry use: Only D_0^2 variables for one symmetric component are evolved, and are replicated 2^{2N_C} times before evaluation; selection and learning use only the original D_0^2 variables. This is not proposed as a useful algorithm, but to provide an upper bound on achievable performance.
3) EDA+block: Partial symmetry use: Before learning, each individual is deconstructed. All symmetric components of the solution are treated as copies of one single symmetric component. Thus the probabilistic model comprises only a fraction of the problem variables, but is learned using a population size which effectively is a multiple of the original population size. During the sampling phase, the same model is used to generate (independently) the values for each symmetric component.
4) EDA+blockperm: A separate model is learnt for each symmetric component using the original population size. During sampling, a random permutation of the models is used to sampled each symmetric component.

There is clearly a connection between variants 3 and 4 and translocation and transposition in GAs. To study this, we also implemented a number of variants of GAs:

1) GA+2ptxover: A traditional GA, with bit-flip mutation and two-point crossover.
2) GA+block: Replaces two-point crossover with a uniform blocked crossover, similar to the traditional uniform crossover operator but exchanging whole blocks of variables belonging to the same symmetric component between parents (in effect, the crossover mask is constrained to be uniform between blocks).
3) GA+blockperm: Uses a uniform blocked crossover with similar transverse exchanges, but blocks can be exchanged between different chromosome regions, giving a form of block-based translocation.
4) GA+transp: Replaces this with a mutation in which blocks are exchanged within the same parent, generating a form of block-based transposition.

All algorithms were implemented in Mateda2.0 [24].

C. Parameter Settings

TABLE I
PARAMETER SETTINGS

Parameter	Value	Parameter	Value
Problems			
D_0	4	N_C:	$0\ldots2$
N_F	0 1 2...64	S	$1\ldots100$
Algorithms			
Replications	100	Elite Size	1
Population	500	Generations	150
Selection Ratio	0.5		

Parameter settings are shown in Table I.

1) Problem Settings: We used 100 random replicates of the test problems, with random seeds $S = 1,\ldots,100$. We chose $D = 4$ as a compromise: small enough to reliably find $F_{\max}(L_0)$, but large enough (2^{32} configurations) that duplicates were unlikely (subsequently, we confirmed there were none). Potential symmetries (rotation, vertical, horizontal, diagonal reflection) made it difficult to test whether problems

[3]This large discrepancy depends heavily on the competition between internal and boundary interactions; it is likely to be much less for random basic blocks, and even more so as D_n increases with D_0 and N_C, because the total contribution from internal interactions increases quadratically, while that of boundary interactions increases only linearly.

might have been duplicated through these means, however the huge number of configurations renders this unlikely.

For noise free problems, we varied N_C, the recursion count, over the range $0 \dots 2$; experimentation beyond this was considered infeasible: the estimated run-time of the EDA comparator for $N_C = 3$ was around six weeks.

In the random noise experiments, the number N_F of bits flipped in experiments was systematically varied from a baseline of 1 (with 0 as a comparator), doubling each time, exponentially up to 64 bits. For these experiments, we used algorithm EDA+block with $N_C = 2$

2) Algorithm Settings: Each algorithm setting was run for 150 generations with 100 different random seeds, a population of 500, a truncation selection ratio of 0.5, and an elite of one.

IV. Results

A. Exact Symmetries

TABLE II
Mean Final Generation Best FItness and Mean Running Time (secs) vs Algorithm and Symmetry Level

Algorithm	Symmetry Levels)	Mean Final Best Fitness	Mean Running Time (secs)
EDA	0	20.9 ± 2.0	506 ± 290
	1	88.4 ± 8.7	5063 ± 2901
	2	325.5 ± 23.0	50911 ± 29011
EDA+	0	20.9 ± 2.0	506 ± 290
basicblock	1	83.8 ± 8.1	509 ± 290
	2	335.0 ± 32.3	5062 ± 2901
EDA+	0	20.9 ± 2.0	506 ± 290
block	1	88.3 ± 9.0	511 ± 290
	2	339.3 ± 35.1	5075 ± 2901
EDA+	0	20.9 ± 2.0	506 ± 290
blockperm	1	88.7 ± 9.0	512 ± 290
	2	340.7 ± 34.6	5078 ± 2900
GA+	0	20.9 ± 2.0	505 ± 290
2ptxover	1	88.4 ± 9.1	507 ± 290
	2	230.9 ± 13.8	5061 ± 2901
GA+	0	20.9 ± 2.0	504 ± 290
block	1	87.5 ± 8.7	507 ± 290
	2	201.8 ± 7.5	5061 ± 2901
GA+	0	20.9 ± 2.0	505 ± 290
blockperm	1	84.8 ± 7.8	507 ± 290
	2	182.1 ± 7.8	5061 ± 2901
GA+	0	20.9 ± 2.0	505 ± 290
transp	1	86.2 ± 9.4	507 ± 290
	2	263.9 ± 26.7	5061 ± 2901

We first consider the performance of the different algorithms on exactly symmetric data. Figure 2 shows the mean best-in-generation fitness (averaged over 100 runs) against generation for different numbers of symmetries, while the final-generation mean best fitness and computation time statistics are shown in table II.[4] Without symmetries, the performance of the algorithms are almost identical.

The performance of the uninformed EDA, which has to infer the problem structure from dependencies, is quite good – but at huge time cost. For two symmetry levels, the learning phase took around 95% of the overall time. For the other algorithms, the time was dominated by the cost of sampling (so all returned similar figures). At the other comparator extreme, EDA+basicblock runs quickly and with good performance, but achieves this by making the assumption that the overall solution is a replicate of the optimal solution to each block – which turns out to be incorrect.

The two block-based EDAs returned the best performance overall, but were not clearly distinguishable from each other. That they were able to find asymmetric solutions with higher fitness than the fully symmetric solutions is interesting. On the other hand, this implies that there must exist solutions in the level 2 problems with four times the fitness of the best found at level 1, but neither block-based EDA found them, suggesting that they are still too bound to the block structure, and further search capability (perhaps representing the dependencies between blocks) is required,

The four GA variants ran slightly faster, but performance was much lower. Simple 2-point crossover performed better than block-based crossover, even when the blocks were permuted, suggesting that translocation is not a useful approach to solving the problems. Transposition in combination with point mutation performed somewhat better, but still not comparably with the EDA-based methods.

B. Inexact Symmetries

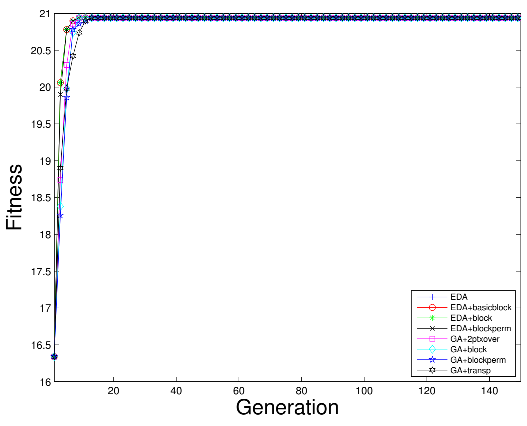
TABLE III
Mean Final Generation Best FItness vs Number of Flipped Interactions, EDA+block Algorithm, Symmetric 16×16 Lattices

# of Random Flips	Mean Final Best Fitness	p value
0	339.3 ± 35.1	
1	341.0 ± 36.4	0.89
2	333.9 ± 38.0	0.18
4	337.1 ± 36.2	0.78
8	337.7 ± 32.8	0.88
16	326.6 ± 31.1	0.01
32	315.0 ± 29.2	0.00
64	281.5 ± 32.7	0.00

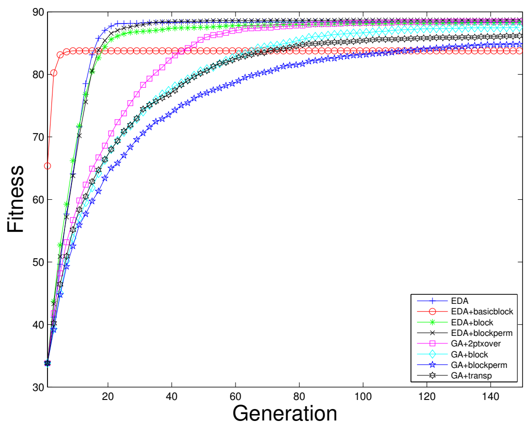
We investigated the robustness of the EDA+block algorithm to inexact symmetries – randomly-flipped interactions in the lattices. Figure 3 shows the mean best-in-generation fitness against generation for different numbers of flips, with the final-generation mean best fitness shown in table III.

For each number of flips, we performed a statistical test to check for difference in the performance of the algorithm with respect to the case when problems with exact symmetries are solved. We applied a Wilkinson rank sum test of the null hypothesis that the best solutions found by the algorithm for each class of problem, defined by the number of flips, corresponded to independent samples from identical continuous distributions with equal medians against the alternative that they do not have equal medians. The results of the test corroborated that for up to 8 flipped interactions (out of a total of 512 – i.e. below 2%) performance is unaffected. Beyond that, there is a definite deterioration. p-values are shown in table III.

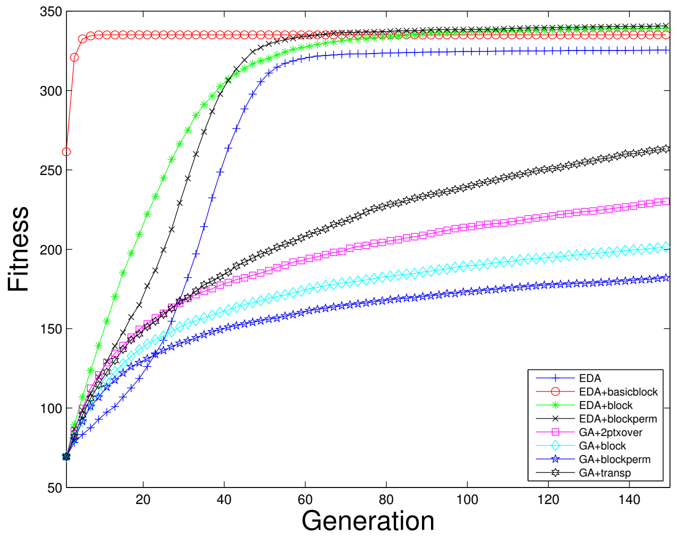
[4]TIme statistics in each generation are only recorded to the nearest second, creating the high regularity in these values.



(a) No Symmetries



(b) One Level of Symmetry



(c) Two Levels of Symmetry

Fig. 2. **Mean Best Fitness vs Generation for all Algorithms on Exact-Symmetry Problems, for Different Levels of Symmetry**

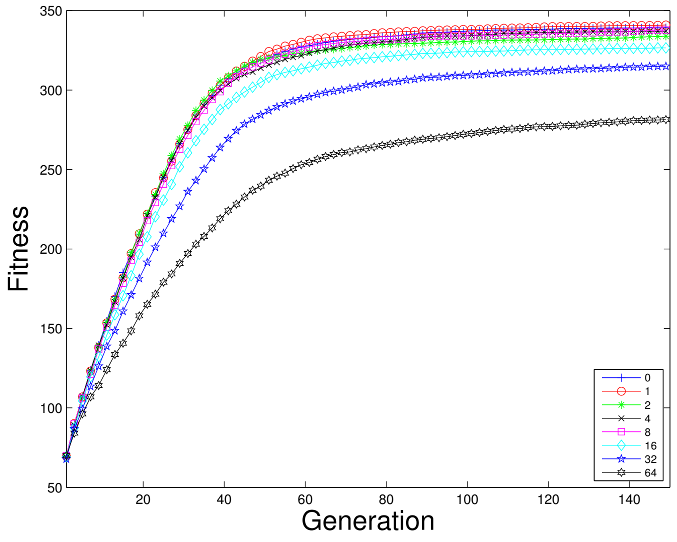


Fig. 3. **Mean Best Fitness vs Generation for EDA+block Algorithm with 0 - 64 Randomly Flipped Interactions in Symmetric 16×16 Lattices**

V. Discussion

A. Analysis of Results

Adding knowledge about variable symmetries seems more effective with EDAs than GAs – we have not, with a GA, been able to approach the EDA performance. This appears to validate our (and others') intuition, that the EDA model is an effective place to incorporate symmetry knowledge. On the other hand, the EDA results could still be improved. On a single level of symmetry, the best EDA method was able to achieve a mean fitness of 88.7, so that on the two-level problem, a fitness of at least 354.8 was achievable – but neither of the best EDAs could approach it, although they did exceed 16 times the base level performance. Possibly more carefully designed algorithms could improve this. Between the two algorithms, there was little to choose: block permutations improved the results very slightly, but any difference was indistinguishable from chance.

The block EDA algorithm demonstrated a relatively high level of robustness to failure of the symmetry assumption, generating reasonable results at more than 1% violation of this assumption. However performance declined rapidly thereafter, becoming quite poor at 64 random flips (i.e. over 10% violation of the symmetry assumption). However it is worth noting that even at this level of violation, performance is still better than any of the GA variants achieved even on the noise-free problem. Thus EDAs with knowledge about symmetry may be worth using with approximate symmetries even if the symmetry is quite badly violated. Of course, better alternatives are readily envisaged, in which a symmetry-aware EDA is at the core of the algorithm, but the algorithm is able to handle symmetry violations, either through a probabilistic model that explicitly models such violations, or through some form of local search from a symmetric core.

B. Learning Symmetries

EDAs explicitly learn the structure of the optimisation problem during evolution. That potentially includes the symmetry structure. There are two ways in which that information may be helpful, corresponding to the two equivalence relationships in variable symmetry – being in the same symmetric component, and playing the same role in different components. Mutual information may be able to tell us which variables belong to the same component (based on distance measures telling us about the relatedness between variables). On the other hand, variables with the same role in different components should have isomorphic dependency structures, and hence similar marginal distributions. Combining these two forms of information may allow us to find candidate symmetries in reasonable time. As we noted earlier, this is the hard part: once candidates have been detected, the cost of distinguishing true from false symmetries is generally low, because permuting a random set of values is unlikely to preserve the fitness value unless a true symmetry is involved.

1) Preliminary Results on Mutual Information: We visualised the mutual information matrices generated in our experiments. The variable ordering used in the visualisations meant that the 4×4 symmetry blocks should lie within 4 of the main diagonal. A visualisation, from generation 10 of the first run, is presented in Figure 4. It is typical of all. It appears that true mutual dependences are detected relatively reliably, with few spurious artefacts, so that identification of the symmetry components may be feasible.

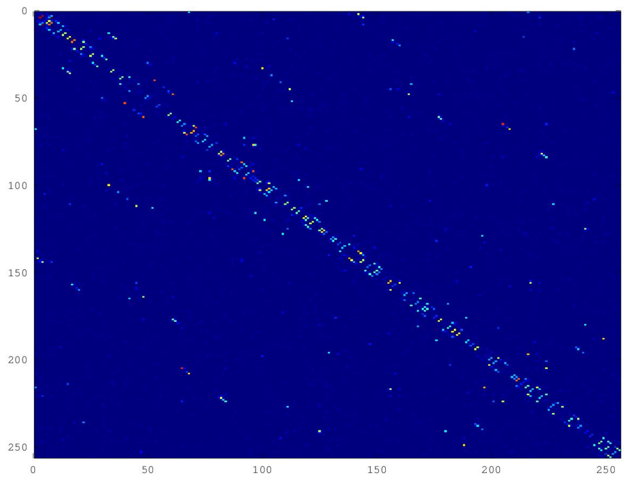


Fig. 4. **Mutual Dependence, EDA+block Algorithm, Symmetric** 16×16 **Lattice, Gen. 10** (light colours indicate significantly non-zero interactions).

C. Assumptions and Limitations

Our experiments relied on random basic components, combined with translation symmetry. To what degree can the results extend to other, different, problems?

Regarding the symmetries, we conducted similar experiments with reflection symmetry rather than translation symmetry; in terms of optimisation, the results were very similar. This suggests that the particular choice of symmetry is not particularly important for optimisation. On the other hand, the mutual dependence diagrams were much more confused, with many additional mutual dependencies detected, suggesting that learning the symmetries might be rather more difficult.

Regarding the basic components, random connections in those components generate little noise to interfere with the symmetry learning. It is possible that we might see different results with more structured components, especially if there were a less random structure to the variable interactions. On the other hand, it is also likely that more structured variable interactions would give rise to further symmetries of which these methods could take advantage.

VI. Conclusions

A. Future Work

1) Symmetries in Genetic Programming: This work had its genesis in consideration of the myriad symmetries of GP. Our awareness of the confusion of terminology stemmed from trying to understand how previous symmetry results could relate to GP, so that will be one focus of our future work

– to further elucidate the nature of epi-symmetries, and to attempt to apply this understanding to determine whether similar mechanisms could be created in EDA-GP.

2) Symmetries in other non-Cartesian spaces: There is potential for exploration of the role of symmetries in non-Cartesian spaces; this applies to abstract spaces such as permutations, and more concrete domains such as protein folding. Deeper investigation seems warranted.

3) Inexact Symmetries: Further work is needed to clarify the mathematics of approximate symmetries, and to relate this understanding to evolutionary domains. In parallel with this, experimental determination of the robustness of symmetry-based algorithms to small asymmetries will also be needed. In particular, we note that biology provides countless examples of inexact symmetries that may nevertheless be useful in search.

4) Symmetry Learning; Inexact Symmetries: In some domains symmetries are known ahead of time; in others they must be discovered: we may only suspect their possibility. Can we learn these symmetries in cost-effective ways?

B. Summary

In this paper, we have taken some preliminary steps toward clarifying the meaning of 'symmetry' in evolutionary computation. We have undertaken an investigation of the role of a range of evolutionary algorithms in some relatively difficult problems involving variable symmetry, suggesting that symmetry-aware EDAs can take advantage of these symmetries effectively, and that they can do so robustly even if the symmetries are not exact. Finally, we have gathered some preliminary evidence that such symmetries may be learnable if they are not known a priori.

Further study of symmetry has much to offer EC, beyond the confines of value symmetries on Cartesian spaces. We look forward to the adventure.

Acknowledgment

Bob McKay's work on this project was supported by the Basic Science Research Program of the National Research Foundation of Korea (NRF) funded by the Ministry of Education, Science and Technology (Project No. 2012S1A2A1A01031076). The work of Jose A. Lozano and Roberto Santana has been partially supported by the Saiotek and Research Groups 2007-2012 (IT-242-07) programs (Basque Government), TIN2010-14931, and COMBIOMED network in computational biomedicine (Carlos III Health Institute).

References

[1] B. Naudts and J. Naudts, "The effect of spin-flip symmetry on the performance of the simple GA," in *Parallel Problem Solving from Nature (PPSN V)*. Springer, 1998, pp. 67–76.

[2] M. Pelikan and D. Goldberg, "Genetic algorithms, clustering, and the breaking of symmetry," in *Parallel Problem Solving from Nature PPSN VI*, ser. Lecture Notes in Computer Science, M. Schoenauer, K. Deb, G. Rudolph, X. Yao, E. Lutton, J. Merelo, and H.-P. Schwefel, Eds. Springer Berlin Heidelberg, 2000, vol. 1917, pp. 385–394.

[3] C. Van Hoyweghen, B. Naudts, and D. E. Goldberg, "Spin-flip symmetry and synchronization," *Evolutionary Computation*, vol. 10, no. 4, pp. 317–344, 2002.

[4] C. V. Hoyweghen, D. E. Goldberg, and B. Naudts, "From twomax to the Ising model: Easy and hard symmetrical problems," in *In Proc. of the Genetic and Evolutionary Computation Conference (GECCO 2002*. Morgan Kaufmann, 2002, pp. 626–633.

[5] J. Peña, J. Lozano, and P. Larrañaga, "Globally multimodal problem optimization via an estimation of distribution algorithm based on unsupervised learning of Bayesian networks," *Evolutionary Computation*, vol. 13, no. 1, pp. 43–66, 2005.

[6] A. Rogers, A. Prügel-Bennett, and N. R. Jennings, "Phase transitions and symmetry breaking in genetic algorithms with crossover," *Theoretical Computer Science*, vol. 358, no. 1, pp. 121–141, 2006.

[7] S.-S. Choi, Y.-K. Kwon, and B.-R. Moon, "Properties of symmetric fitness functions," *Evolutionary Computation, IEEE Transactions on*, vol. 11, no. 6, pp. 743 –757, dec. 2007.

[8] C. Munteanu and A. Rosa, "Symmetry at the genotypic level and the simple inversion operator," in *Proceedings of the aritficial intelligence 13th Portuguese conference on Progress in artificial intelligence*, ser. EPIA'07. Berlin, Heidelberg: Springer-Verlag, 2007, pp. 209–222.

[9] P. Larrañaga and J. A. Lozano, Eds., *Estimation of Distribution Algorithms. A New Tool for Evolutionary Computation*. Boston/Dordrecht/London: Kluwer Academic Publishers, 2002.

[10] J. Rosen, "Approximate symmetry," in *Group Theoretical Methods in Physics*, ser. Lecture Notes in Physics, A. Janner, T. Janssen, and M. Boon, Eds. Springer Berlin Heidelberg, 1976, vol. 50, pp. 608–608.

[11] ——, *Symmetry Rules: How Science and Nature are Founded on Symmetry ; with 4 Tables*, ser. The frontiers collection. Springer London, Limited, 2008. [Online]. Available: http://books.google.es/books?id=EEycWgMs-MUC

[12] S. Fischer and I. Wegener, "The Ising model on the ring: Mutation versus recombination," in *Proc. of the Genetic and Evolutionary Computation Conference (GECCO 2004). LNCS 3102*. Springer, 2004, pp. 1113–1124.

[13] M. Dietzfelbinger, B. Naudts, C. Van Hoyweghen, and I. Wegener, "The analysis of a recombinative hill-climber on H-IFF," *Evolutionary Computation, IEEE Transactions on*, vol. 7, no. 5, pp. 417 – 423, oct. 2003.

[14] C. Munteanu and V. Lazarescu, "Simple inversion genetic algorithm: An effective optimization strategy," in *Proceedings of the International ICSC / IFAC Symposium on Neural Computation (NC 1998). Vienna, Austria*, M. Heiss, Ed. ICSC Academic Press, International Computer Science Conventions, Canada / Switzerland, 1998, pp. 151–156.

[15] A. Simoes and E. Costa, "Transposition: a biologically inspired mechanism to use with genetic algorithms," in *Fourth International Conference on Neural Networks and Genetic Algorithms*, 1999, pp. 612–619.

[16] M. Oates, D. Corne, and R. Loader, "Skewed crossover and the dynamic distributed database problem," in *Artificial Neural Nets and Genetic Algorithms*. Springer Vienna, 1999, pp. 280–287.

[17] M. S. Voss, C. M. Foley *et al.*, "Evolutionary algorithm for structural optimization," in *Proceedings of the Genetic and Evolutionary Computation Conference*, vol. 1, 1999, pp. 678–685.

[18] K. Stanley, D. D'Ambrosio, and J. Gauci, "A hypercube-based encoding for evolving large-scale neural networks," *Artificial Life*, vol. 15, no. 2, pp. 185–212, 2009.

[19] S. Baluja and S. Davies, "Using optimal dependency-trees for combinatorial optimization: Learning the structure of the search space," in *Proceedings of the 14th International Conference on Machine Learning*, D. H. Fisher, Ed. San Francisco, CA.: Morgan Kaufmann, 1997, pp. 30–38.

[20] E. Ising, "Contributions to the theory of ferromagnetism (beitrage zur theorie des ferromagnetismus," Ph.D. dissertation, University of Hamburg, 1924.

[21] R. K. Thompson and A. H. Wright, "Additively decomposable fitness functions," University of Montana, Tech. Rep., 1996.

[22] R. A. Watson, "Compositional evolution," Ph.D. dissertation, Brandeis University, 2002.

[23] E. De Jong, D. Thierens, and R. Watson, "Hierarchical genetic algorithms," in *Parallel Problem Solving from Nature-PPSN VIII*. Springer, 2004, pp. 232–241.

[24] R. Santana, C. Bielza, P. Larrañaga, J. A. Lozano, C. Echegoyen, A. Mendiburu, R. Armañanzas, and S. Shakya, "Mateda-2.0: A MATLAB package for the implementation and analysis of estimation of distribution algorithms," *Journal of Statistical Software*, vol. 35, no. 7, pp. 1–30, 2010. [Online]. Available: http://www.jstatsoft.org/v35/i07