Customized Selection in Estimation of Distribution Algorithms

Roberto Santana, Alexander Mendiburu, and Jose A. Lozano

Department of Computer Science and Artificial Intelligence, University of the Basque Country (UPV/EHU). P. Manuel de Lardizabal, 20018, Guipuzcoa, Spain
{roberto.santana,alexander.mendiburu,ja.lozano}@ehu.es

Abstract. Selection plays an important role in estimation of distribution algorithms. It determines the solutions that will be modeled to represent the promising areas of the search space. There is a strong relationship between the strength of selection and the type and number of dependencies that are captured by the models. In this paper we propose to use different selection probabilities to learn the structural and parametric components of the probabilistic graphical models. Customized selection is introduced as a way to enhance the effect of model learning in the exploratory and exploitative aspects of the search. We use a benchmark of over 15, 000 instances of a simplified protein model to illustrate the gains in using customized selection.

Keywords: selection, estimation of distribution algorithms, optimization, customized selection.

1 Introduction

Since their inception most of the research on estimation of distribution algorithms (EDAs) [9,10,13] has been devoted to the analysis of the learning and sampling components of these algorithms. The characteristic feature of EDAs with respect to other evolutionary algorithms (EAs) is the use of probabilistic modeling to capture the most relevant features of the selected solutions. Therefore, learning and sampling steps are critical for these algorithms and research on these methods in EAs almost began with EDAs. A different situation arises for the selection methods used by EDAs. The selection schemes traditionally applied in these algorithms are essentially those widely applied in GAs.

Different approaches investigate how selection mediates the information about the fitness function that is passed to the probabilistic models. Among the research directions explored are: 1) Explicitly using fitness information in the construction of the probabilistic models to learn more accurate models [15,17,20] and 2) Explicitly modeling fitness information as part of the probabilistic model [7,11]. These research directions are very related since it has been shown that the explicit modeling of fitness information can produce more accurate representations of the interactions between the variables. Furthermore, some research on the role of selection has gone beyond the classical aim of optimization to investigate the effect of selection in recovering the original problem structure [2].

In this paper we propose customized selection as a new way to implement selection in EDAs. We start from the assumption that the role played by selection in EDAs is two-fold. As in GAs, selection should be able to capture information about the promising areas of search space. But in addition, the selection method should contribute to a meaningful and efficient representation by the probability model. Basically, this assumption states that, in EDAs, considering the choice of the selection method in accordance with the type of probability modeling applied can contribute to a more efficient search for the optimal solutions. Customized selection takes into account this assumption by splitting the information extracted during the selection step into: 1) Information used for structural learning and 2) Information used for parametric learning. We empirically evaluate this idea for Boltzmann and truncation selections.

2 Selection and Learning in EDAs

In this section we assume the reader is familiar with EDAs. Let $\mathbf{X} = (X_1, \dots, X_n)$ denote a vector of discrete random variables. We use $\mathbf{x} = (x_1, \dots, x_n)$ to denote an assignment to the variables. I denotes a set of indices in $\{1, \dots, n\}$ and X_I (respectively x_I) a subset of the variables of $\mathbf{X}$ (respectively $\mathbf{x}$) determined by the indices in I. p denotes a distribution, $p(x_I)$ the marginal probability for $\mathbf{X}_I = \mathbf{x}_I$, and $p(x_i \mid x_j)$ the conditional probability distribution of $X_i = x_i$ given $X_j = x_j$.

Algorithm 1. Tree-EDA

```
1   $D_0 \leftarrow$ Generate $N$ individuals randomly
2   $t = 0$
3   do {
4       Evaluate the individuals using the fitness function.
5       Assign a selection probability to each individual.
6       Create a compact population $D_t^S$ where copies of the same individual add up
        their probabilities $p_t^S$.
7       Calculate a probabilistic model using $D_t^S$ and $p_t^S$.
8       Compute the univariate and bivariate marginal frequencies $p_i^s(x_i|D_t^s)$ and
        $p_{i,j}^s(x_i, x_j|D_t^S)$ using $D_t^S$ and $p_t^S$
9       Calculate the mutual information using bivariate and univariate marginals.
10      Calculate the maximum weight spanning tree from the mutual information.
11      Compute the parameters of the model.
12      $t \leftarrow t + 1$
13      $D_t \leftarrow$ Sample $N$ individuals from the tree and add elitist solutions.
14  } until A stop criterion is met
```

Algorithm 1 shows the pseudocode of an EDA that uses the complete population to define the selection probabilities. This EDA learns a tree model. In this section we focus on the analysis of the selection procedure and leave the analysis of the model learning step for Section 4. In terms of the selection procedure, the main difference between Algorithm 1 and the typical EDA is that in the former, instead of selecting a subset of individuals based on their fitness, a vector of selection probabilities is computed over the complete population. The probabilistic model is learned from the vector and the population. The procedure described in steps 5 to 7 of Algorithm 1 was originally introduced in [17].

Using this way to implement the selection has two advantages: 1) When possible, the fitness information of the complete population is used. 2) Although the computation of the compact population is not essential for computation of the probabilistic model, it helps to make model learning faster, particularly when there are multiple copies of the same individuals in the population. A requirement for the application of this method is that model learning could be done directly on the probabilities. This can be easily done for most of the model learning methods [5,15,17,20].

There is an extra cost in finding the compact population but notice that for detecting that two solutions are different, it is sufficient to find a variable where they differ. Therefore, although comparison between pairs of solutions can have a worst case cost of n, this cost will depend on the homogeneity of the population and the expected cost of finding the compact population will be often much smaller than $Nlog(N)n$, where N is the population size.

3 Customized Selection

Probabilistic models learned by EDAs can be classified according to the type of learning they use into two groups: 1) Models that apply non-structural (parametric) learning. 2) Models that apply structural *and* parametric learning. We extend this classification to EDAs and talk of non-structural learning and structural-learning EDAs, understanding that the second class of algorithms also applies parametric learning of the models. Among non-structural-learning EDAs are the univariate marginal distribution algorithm (UMDA) [14] and other EDAs based on marginal product models [12]. Structural-learning EDAs include algorithms based on Bayesian networks and Markov networks.

The key idea of customized selection is to learn the structure and the parameters of the model from different selection probabilities. We assume that, in terms of population diversity, non-structural learning and structural learning may have different requirements for accurately modeling. For example, in truncation selection, we may need to have a selection threshold of 0.5 (half the population) to guarantee a dataset large and diverse enough from which to learn the model structure applying statistical tests. However, once the structure is learned, we would like to make the marginal probabilities to represent the characteristics of solutions of highest fitness, for instance, those included in the best 30% of the population. In this way, we combine learning a robust structure with non-structural learning more focused on the best solutions. In all selection methods currently used by EDAs, the same selected population is used for both tasks. We aim to split this process and investigate whether customized selection has an impact in the quality of the search of EDAs.

3.1 Customized Boltzmann and Truncation Selections

We use notation introduced in Section 2. In Boltzmann selection, the probability of each solution to be part of the selected population is computed according to the Boltzmann probability distribution $\hat{p}(\mathbf{x}) = \frac{e^{\frac{f(\mathbf{x})}{T}}}{\sum_{\mathbf{x}'} e^{\frac{f(\mathbf{x}')}{T}}}$, where $\sum_{\mathbf{x}'} e^{\frac{f(\mathbf{x}')}{T}}$ is the so-called partition function, and T is the temperature of the system that can be used as a parameter to smooth the probabilities. The partition function is computed using all the solutions in the current population and a probability of selection is associated to each solution.

We use T as a way to influence the strength of selection. When $T \to \infty$, the models can not recover any information about the problem structure because all solutions are given the same probability. Similarly, when $T \to 0$ all the probability is concentrated in the point with highest function value in the population. Customized Boltzmann selection is implemented by using two different values of the temperature, Ts and Tp which will be associated to the structural and non-structural learning, respectively. Ts and Tp will bias the type and amount of information captured by the probabilistic models. In our experiments, we focus on the analysis of $Ts, Tp = 2^k$ for $k \in \{-3, -2, \cdots, 1, 2\}$.

In truncation selection, the best $M = \alpha N$ individuals of the population is selected, being $\alpha \in (0, 1]$. We define truncation selection on the complete population by associating a probability $\frac{1}{M}$ to the best M individuals and 0 to the rest. Customized truncation selection is implemented by defining two different truncation thresholds α_{T_s} and α_{T_p} for structural and non-structural learning, respectively. In usual application of truncation selection, $\alpha_{T_s} = \alpha_{T_p}$, but in customized selection these values can be different.

4 EDAs with Customized Selection

Customized selection can only be applied to EDAs that apply structural learning. We use Tree-EDA, an EDA that learns a tree probabilistic model and is similar to the ones introduced in [1] and [16]. The probability distribution of a tree is defined as $p_{\mathcal{T}}(\mathbf{x}) = \prod_{i=1}^{n} p(x_i|pa(x_i))$ where $Pa(X_i)$ is the parent of X_i in the tree, and $p(x_i|pa(x_i)) = p(x_i)$ when $pa(x_i) = \emptyset$, i.e. X_i is a root of the tree. The distribution $p_{\mathcal{T}}(\mathbf{x})$ itself will be called a tree model when no confusion is possible. We allow the existence of more than one root in the PGM (i.e. forests) although for convenience of notation we refer to the model as tree. Algorithm 1 shows the pseudocode of Tree-EDA.

We choose Tree-EDA to evaluate customized selection because it exhibits a good balance between the capacity of the probabilistic model to represent dependencies and the computational cost of learning and sampling the tree.

For comparison purposes we use UMDA. The univariate model used by this algorithm can be seen as a particular case of the tree when we have $pa(X_i) = \emptyset$ for all i.

As shown in Algorithm 1, the tree is learned using the Chow-Liu method [3] that calculates the maximum weight spanning tree from the matrix of mutual information. Notice that the mutual information is computed from the bivariate and univariate probabilities calculated upon marginalization of the selection probabilities of the compact population. When customized selection is used, the computation of the bivariate and univariate probabilities is done twice. The first time, from the selection probabilities calculated using T_s (respectively α_s for truncation selection). It is during this first step when the mutual information and the tree structures are computed. Then, during a second step, the univariate and bivariate probabilities are computed again, this time using T_p (respectively α_p for truncation selection), but only for the edges of the tree, i.e. a maximum of $n-1$ bivariate probabilities instead of $\frac{n(n-1)}{2}$.

5 HP Functional Model Protein

As a testbed we use an optimization problem defined on a simplified protein model. The HP model considers hydrophobic (H) residues and hydrophilic or polar (P) residues. A protein is considered a sequence of these two types of residues, which are located in regular lattice models forming self-avoided paths. Given a pair of residues, they are considered neighbors if they are adjacent either in the chain (connected neighbors) or in the lattice but not connected in the chain (topological neighbors).

The functional model protein is a "shifted" HP model that can represent native states that are not maximally compact [6]. An energy function that measures the interaction between topological neighbor residues is defined as $\epsilon_{HH} = -2$ and $\epsilon_{PP} = \epsilon_{HP} = \epsilon_{PH} = 1$. The functional model protein problem consists of finding the solution that minimizes the total energy and it is a NP-hard problem.

Figure 1 shows an example of a functional model protein instance. In our solution representation, for a given sequence and lattice, X_i represents the relative move of residue i in relation to the previous two residues. Taking as a reference the location of the previous two residues in the 2D lattice, X_i takes values in

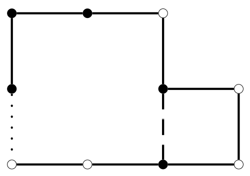

Fig. 1. One possible configuration of sequence $HHHPHPPHPP$ in the functional model protein. There is one HH interaction (represented by a dashed line), and one HP interaction (represented by a dotted line).

$\{0, 1, 2\}$. These values respectively mean that the new residue is located in one of the 3 possible directions (left, ahead, right) with respect to the previous two locations [8]. Therefore, values for X_1 and X_2 are meaningless. The locations of these two residues are fixed. A solution $\mathbf{x}$ can be seen as a walk in the lattice, representing one possible folding of the protein.

The codification corresponding to the configuration of the sequence shown in Figure 1a) is $\mathbf{x} = (0, 0, 2, 1, 2, 0, 2, 2, 1, 1)$. The objective function is computed as the opposite of the energy for feasible configurations.

In our representation there can be self-intersecting paths that correspond to unfeasible configurations. We use two ways to deal with these solutions: 1) Penalized fitness functions and 2) Repairing of solutions. We penalize self-intersecting solutions by dividing the energy by the number of the sequence's self-intersections. To repair solutions, a variation of the backtracking method introduced in [4] is applied.

As a problem benchmark, the functional model protein is a very interesting problem because, disregarding multiple representations of the same solution, the problem reaches the optimum on a unique configuration. We have selected a database of $15,575$ protein sequences ($n = 23$) [8] for which, the optimal value, the closest suboptimal value, and the number of configurations where this suboptimal value is reached have been previously determined. The complexity of the optimization problem can be very different between sequences.

6 Experiments

The aim of the experiments is determining if using customized selection can help to improve the results of the EDAs that apply Boltzmann and truncation selection. A second goal is to find an appropriate choice of the selection parameters. Finally, we investigate the effect of the number of local optima in the behavior of the EDAs for the different selection methods.

6.1 Experiment Settings

The population size for all EDAs was fixed to $N = 500$ and as termination criterion we used a maximum number of generations $N_g = 50$. Experiments were run with and without repairing of the solutions. In the second case, the fitness values of infeasible solutions were penalized.

For Boltzmann selection we investigate in detail the effect of using different probability distributions to learn the parameters and structure of the model. This is done by trying all combinations of $Ts, Tp = 2^k$, for $k \in \{-3, -2, \cdots, 1, 2\}$. For truncation selection, $\alpha_s, \alpha_p \in \{0.1, 0.2, 0.3, 0.4, 0.5, 0.6\}$. When using the UMDA, the selection parameters of the structural and non-structural learning have identical values. Therefore, for each type of selection there are 36 variants of Tree-EDAs, and 6 variants of UMDA. We run the EDAs 100 times for each of the $15,575$ instances. The total number of EDA runs for each type of selection method was $2 \times 100 \times 15,575 \times (36 + 6) = 130,830,000$ that were executed in a cluster of 575 computers.

6.2 Results for Customized Boltzmann Selection

We compare the algorithms in terms of the number of times that the optimum was found in 100 runs and in terms of the mean fitness value of the best solutions found in each of the 100 runs. The mean success rate of the different EDA variants from the $15,575$ instances is shown in Figure 2.

It can be seen in Figure 2a) that there are important variations in the success rate of Tree-EDA due to parameters Ts and Tp. The influence of the parameters can be critical for the behavior of the algorithm. Notably, Tree-EDA with $Tp > -1$ can not outperform the behavior of UMDA. Notice however, that UMDA is also very sensitive to the influence of parameter T. For all values of Tp, except $Tp = -3$, the number of times that Tree-EDA finds the optimum improves by selecting $Ts < Tp$. This means that, for a given selection strength applied to non-structural learning, a stronger selection applied to structural learning will likely improve the results. Figure 2b) shows how the mean success rate of Tree-EDA also increases by repairing the solutions. The same trend in the influence of the selection parameters is observed. Except for $Tp = -3$, the results of Tree-EDA improves by selecting $Ts < Tp$.

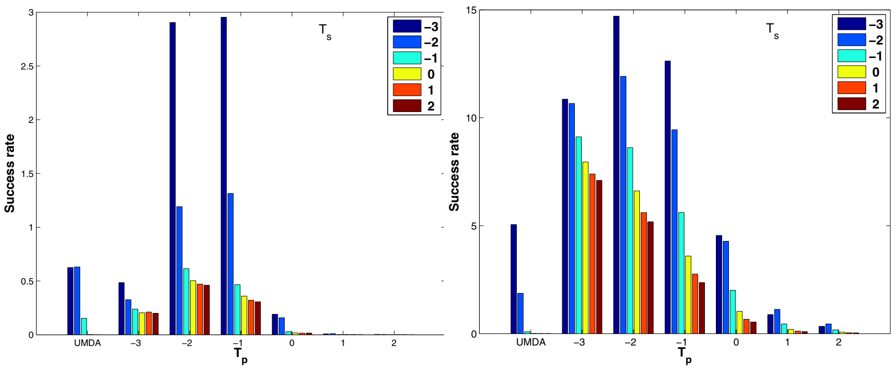

Fig. 2. Customized Boltzmann selection: Number of times the best solution was found for the different values of the temperature. a) Without repairing. b) When repairing is applied.

We also test, for each instance independently, for statistical differences between the EDAs that could be attributed to the use of customized selection. Fixing the value of Tp, we apply a multi-comparison test (p-value 0.05) for the six variants of Tree-EDA. The test uses as information the best result in each of the 100 runs for the corresponding instance. Among the six variants of Tree-EDA corresponding to the six values of Ts, we test which pairs of means are significantly different, and which are not. A test that can provide such information is called a multiple comparison procedure. Adjustment for multiple testing is applied using the Dunn-Sidak method [18], a procedure similar to, but less conservative than, the Bonferroni procedure.

Using the results of the test, we compute the number of times each Tree-EDA variant was significantly better (S_b) and significantly worst (S_w) than the Tree-EDA that uses $Ts = Tp$. For example, for $Tp = 1$, we compute the number of times that Tree-EDA ($Ts = i$) was significantly better than Tree-EDA ($Ts = 1$) for all $i \neq 1$. Similarly, we compute the number of times that Tree-EDA ($Ts = i$) was significantly worse than Tree-EDA ($Ts = 1$). The difference between these two numbers gives an idea of the appropriate choice for Ts in relation with Tp. Figure 3 shows the values of ($S_b - S_w$) for all values of Tp. A positive value for $Ts = i$ means that Tree-EDA improves its performance when it takes this value. Conversely, a negative value indicates a poorer behavior.

Figure 3 confirms the previous results obtained from the analysis of mean success rate. For Boltzmann distribution, improvements can be achieved by using, for structural learning, selection probabilities with a higher selection strength than that used for non-structural learning. We should learn the structure from a set of very good solutions but the selection strength can be relaxed at the time of learning the model's parameters.

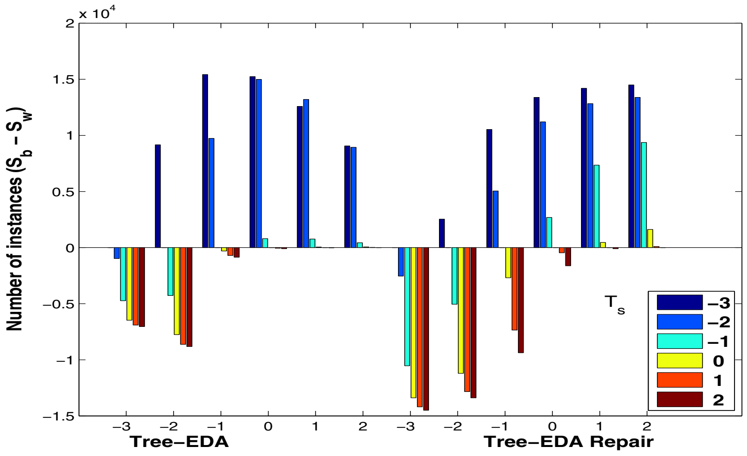

Fig. 3. Results of the statistical tests for customized Boltzmann selection

6.3 Results for Customized Truncation Selection

The analysis of customized truncation selection are conducted using the same methodology.

Figure 4 shows the mean success rate of Tree-EDA with and without repairing. The results of Tree-EDA improves by selecting $\alpha_s < \alpha_p$ for $\alpha_p > 0.1$. Also for Tree-EDA with repairing the results improve for $\alpha_s < \alpha_p$ for $\alpha_p > 0.1$, but the differences are not that clear. For truncation selection, $\alpha_s \in \{0.2, 0.3\}$ is the best choice for almost all values of α_p. Another remarkable fact that makes a difference with Boltzmann selection is that Tree-EDA is always better than UMDA when truncation selection is used.

The results of the statistical tests for customized truncation selection are shown in Figure 5. It can be seen in the figure that Tree-EDA with truncation selection exhibits a behavior similar to Tree-EDA with Boltzmann selection when the repairing procedure is not used. Nevertheless, when repairing is applied, there are fewer significant differences in the behavior of the algorithms. This fact can be also observed in Figure 5.

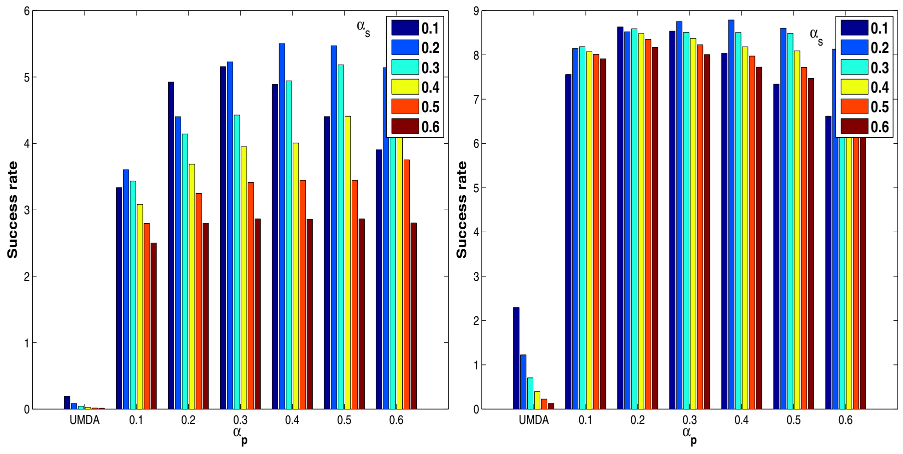

Fig. 4. Customized truncation selection: Number of times the best solution was found for the different values of the truncation parameter. a) Without repairing. b) When repairing is applied.

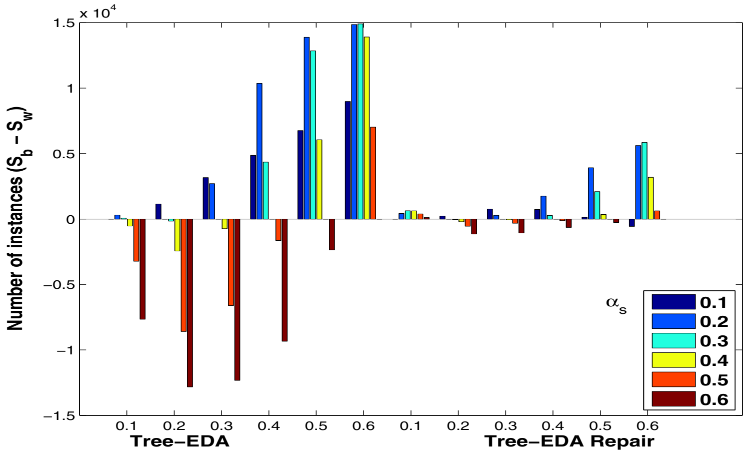

Fig. 5. Results of the statistical tests for customized truncation selection

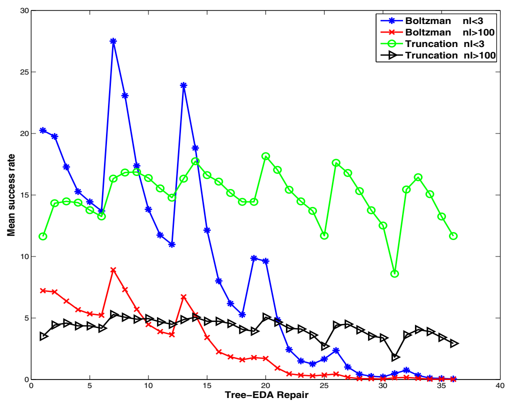

Fig. 6. Mean success rate of Tree-EDA with repairing for solutions grouped according to the number of local optima (nl)

To evaluate how sensitive are the introduced selection methods to the number of optima in the problem, we selected two sets of instances. The first group contains instances with one or two local optima (703 instances). The second group comprises instances with more than 100 local optima (685 instances). Using Tree-EDA with repairing we compute the mean success rate for the selection methods on all possible combinations of parameters Tp and Ts ($6 \times 6 = 36$). The results are shown in Figure 6 where nl is the number of local optima in addition to the global optimum. The main conclusion from the analysis of Figure 6 is that both selection methods are sensitive to the number of local optima. However, while the Boltzmann customized selection is able to outperform customized truncation selection for appropriate combination of parameters, the second selection method is more robust to the variation of the parameters.

7 Conclusions and Future Work

In this paper we have introduced customized selection in EDAs. We have shown that, by using different selection probabilities for structural and non-structural learning of the models, it is possible to increase the success rate of Tree-EDA for the functional model protein. Our results show that improvements are more important for Boltzmann selection than for truncation selection.

Beyond the improvements in optimization, customized selection opens new possibilities for research on the relationship between selection and model learning in EDAs. We can independently evaluate the effect of selection in the structural and parametric components of the graphical models. For instance, we can investigate the quality of the models as fitness surrogates by independently manipulating the different selection probabilities from which the model's components are learned.

There are optimization problems where some sets of variables make a higher contribution to the fitness. Evolutionary algorithms can fail in these situations when these *salient* building blocks converge before those with lower marginal fitness [19]. One possible extension of customized selection is the computation of marginal probabilities of different subsets of variables using different selection probabilities. In this way, marginal probabilities could be adjusted, according to different strengths of selection, to "synchronize" building blocks with different temporal-salience behaviors.

Acknowledgments. This work has been partially supported by the Saiotek and Research Groups 2013-2018 (IT-609-13) programs (Basque Government), TIN2013-41272P (Ministry of Science and Technology of Spain), COMBIOMED network in computational bio-medicine (Carlos III Health Institute), and by the NICaiA Project PIRSES-GA-2009-247619 (European Commission).

References

1. Baluja, S., Davies, S.: Using optimal dependency-trees for combinatorial optimization: Learning the structure of the search space. In: Fisher, D.H. (ed.) Proceedings of the 14th International Conference on Machine Learning, pp. 30–38. Morgan Kaufmann, San Francisco (1997)
2. Brownlee, A.E.I., McCall, J., Shakya, S.K.: The Markov network fitness model. In: Shakya, S., Santana, R. (eds.) Markov Networks in Evolutionary Computation, pp. 125–140. Springer (2012)
3. Chow, C.K., Liu, C.N.: Approximating discrete probability distributions with dependence trees. IEEE Transactions on Information Theory 14(3), 462–467 (1968)
4. Cotta, C.: Protein structure prediction using evolutionary algorithms hybridized with backtracking. In: Mira, J., Álvarez, J.R. (eds.) IWANN 2003. LNCS, vol. 2687, pp. 321–328. Springer, Heidelberg (2003)
5. Echegoyen, C., Mendiburu, A., Santana, R., Lozano, J.A.: On the taxonomy of optimization problems under estimation of distribution algorithms. Evolutionary Computation 21(3), 471–495 (2013)
6. Hirst, J.D.: The evolutionary landscape of functional model proteins. Protein Engineering 12, 721–726 (1999)
7. Karshenas, H., Santana, R., Bielza, C., Larrañaga, P.: Multi-objective optimization based on joint probabilistic modeling of objectives and variables. IEEE Transactions on Evolutionary Computation 18(4), 519–542 (2014)
8. Krasnogor, N., Blackburne, B.P., Burke, E.K., Hirst, J.D.: Multimeme algorithms for protein structure prediction. In: Guervós, J.J.M., Adamidis, P.A., Beyer, H.-G., Fernández-Villacañas, J.-L., Schwefel, H.-P. (eds.) PPSN VII. LNCS, vol. 2439, pp. 769–778. Springer, Heidelberg (2002)
9. Larrañaga, P., Karshenas, H., Bielza, C., Santana, R.: A review on probabilistic graphical models in evolutionary computation. Journal of Heuristics 18(5), 795–819 (2012)
10. Lozano, J.A., Larrañaga, P., Inza, I., Bengoetxea, E. (eds.): Towards a New Evolutionary Computation: Advances on Estimation of Distribution Algorithms. Springer (2006)

11. Miquélez, T., Bengoetxea, E., Larrañaga, P.: Evolutionary computation based on Bayesian classifiers. International Journal of Applied Mathematics and Computer Science 14(3), 101–115 (2004)
12. Mühlenbein, H., Mahnig, T., Ochoa, A.: Schemata, distributions and graphical models in evolutionary optimization. Journal of Heuristics 5(2), 213–247 (1999)
13. Mühlenbein, H., Paaß, G.: From recombination of genes to the estimation of distributions I. Binary parameters. In: Voigt, H.-M., Ebeling, W., Rechenberg, I., Schwefel, H.-P. (eds.) PPSN IV. LNCS, vol. 1141, pp. 178–187. Springer, Heidelberg (1996)
14. Mühlenbein, H., Schlierkamp-Voosen, D.: The science of breeding and its application to the breeder genetic algorithm (BGA). Evolutionary Computation 1(4), 335–360 (1994)
15. Munetomo, M., Murao, N., Akama, K.: Introducing assignment functions to Bayesian optimization algorithms. Information Sciences 178(1), 152–163 (2008)
16. Pelikan, M., Mühlenbein, H.: The bivariate marginal distribution algorithm. In: Roy, R., Furuhashi, T., Chawdhry, P. (eds.) Advances in Soft Computing - Engineering Design and Manufacturing, pp. 521–535. Springer, London (1999)
17. Santana, R.: Factorized distribution algorithms: Selection without selected population. In: Proceedings of the 17th European Simulation Multiconference ESM 2003, Nottingham, England, pp. 91–97 (2003)
18. Šidák, Z.: Rectangular confidence regions for the means of multivariate normal distributions. Journal of the American Statistical Association 62(318), 626–633 (1967)
19. Thierens, D., Goldberg, D.E., Pereira, A.G.: Domino convergence, drift, and the temporal-salience structure of problems. In: Proceedings of 1998 IEEE International Conference on Evolutionary Computation, Anchorage, AK, pp. 535–540 (1998)
20. Valdez-Peña, I.S., Hernández-Aguirre, A., Botello-Rionda, S.: Approximating the search distribution to the selection distribution in EDAs. In: Proceedings of the Genetic and Evolutionary Computation Conference GECCO 2009, pp. 461–468. ACM, New York (2009)