A Probabilistic Evolutionary Optimization Approach to Compute Quasiparticle Braids

Roberto Santana[1], Ross B. McDonald[2], and Helmut G. Katzgraber[2,3]

[1]Department of Computer Science and Artificial Intelligence,
University of the Basque Country (UPV/EHU), P. Manuel de Lardizabal,
20018, Guipuzcoa, Spain
[2]Department of Physics and Astronomy, Texas A&M University, College Station,
Texas 77843-4242, USA
[3]Santa Fe Institute, 1399 Hyde Park Road, Santa Fe, NM 87501, USA
roberto.santana@ehu.es, katzgraber@physics.tamu.edu

Abstract. This paper proposes the use of estimation of distribution algorithms to deal with the problem of finding an optimal product of braid generators in topological quantum computing. We investigate how the regularities of the braid optimization problem can be translated into statistical regularities by means of the Boltzmann distribution. The introduced algorithm obtains solutions with an accuracy in the order of 10^{-6}, and lengths up to 9 times shorter than those expected from braids of the same accuracy obtained with other methods.

Keywords: topological computing, quasiparticle braids, probabilistic graphical models, EDAs, braid optimization, Fibonacci anyons.

1 Introduction

The idea of using the theory of quantum mechanics to obtain computers potentially exponentially faster for certain applications, such as the factorization of prime numbers, arouses considerable interest and research efforts from the scientific community nowadays. In quantum computation, information is represented and manipulated using quantum properties. An obstacle for the construction of large quantum computers is the problem of quantum decoherence, that can be viewed as the loss of information of the quantum system due to the interaction with the environment. One possible solution to this problem is the design of quantum systems immune to quantum decoherence on a hardware level.

Topological quantum computing (TQC) [2,14] investigates quantum computing systems that, given the properties of quasiparticles they use, are not affected by quantum decoherence. The key idea of these systems is that quantum information can be stored in global properties of the system and thus affected only by global operations but not by local perturbations such as noise. In TQC, quantum gates are carried out by adiabatically braiding quasiparticles around each other. This braiding is used to perform the unitary transformations of a quantum computation.

G. Dick et al. (Eds.): SEAL 2014, LNCS 8886, pp. 13–24, 2014.

One of the essential questions to design a TQC is to find a product of braid generators (matrices) that approximates a quantum gate with the smallest possible error and, if possible, as short as possible to prevent loss [8]. The relevant question of minimizing the error of a TQC design can be posed as a braid optimization problem. Some optimization approaches to this question have been proposed. Exhaustive search [2] has been applied to search for braids of manageable size (up to 46 exchanges). Other methods such as the Solovay-Kitaev algorithm [3] provide bounds on the accuracy and length of the braids. However, they do not allow the user to tune the balance between the accuracy and the length as pioneered in [8] where the use of genetic algorithms (GAs) to find optimal braids is proposed. In this paper, we build on the GA approach introduced in [8] to solve the braid optimization problem.

We use the fitness function proposed in [8] and introduce a new representation, variation operators and enhancement procedures in the framework of estimation of distribution algorithms (EDAs) [10,7]. EDAs are evolutionary algorithms (EAs) that apply learning and sampling of distributions instead of classical crossover and mutation operators. Modeling the dependencies between the variables of the problem serves to efficiently orient the search to more promising areas of the search space by explicitly capturing and exploiting potential relationships between the problem variables.

2 Braids and Anyons

Qubits play in quantum computation a role similar to that played by bits in digital computers. A braid operation can be represented by a matrix that acts on the qubit space. These matrices are referred to as generators, and the quantum gate that a braid represents is the product of the generators that encode the individual braid operations.

Let σ_1 and σ_2 represent two possible generators. σ_1^{-1} and σ_2^{-1} respectively represent their inverses. Given a braid B, $len()$ is a function that returns the braid's length l (e.g. $B = \sigma_1\sigma_1\sigma_2\sigma_1^{-1}$, $l = len(B) = 4$).

Since the product of a matrix by its inverse reduces to the identity matrix, some braids can be simplified reducing their length. Therefore, we also define function $elen()$, that has a braid as its argument and returns the braid's *effective length* which is the length of braid after all possible simplifications have been conducted. For example, the effective length values of braids ($\sigma_1\sigma_1\sigma_1\sigma_1\sigma_1^{-1} = \sigma_1\sigma_1\sigma_1$) and ($\sigma_2^{-1}\sigma_1\sigma_1\sigma_1^{-1}\sigma_1^{-1}\sigma_2\sigma_1^{-1} = \sigma_1^{-1}$) are 3 and 1, respectively.

Let T represent the target matrix (gate to be emulated), the braid error is calculated as [8]: $\epsilon = |B - T|$ where the matrix norm used is $|M| = \sqrt{\sum_{ij} M_{ij}^2}$.

The problem of finding braiding operations that approximate gates is then reduced to finding a product chain of the reduced generators and their inverses that approximates the matrix representing the quantum gate. Two elements that describe the quality of a braid are its error ϵ and its length l.

Anyons appear as emergent quasiparticles in fractional quantum Hall states and as excitations in microscopic models of frustrated quantum magnets that

harbor topological quantum liquids [11]. Fibonacci anyons are the simplest anyons with non-Abelian braiding statistics that can give rise to universal quantum computation. Fibonacci anyon braids [2] only encompasses one-qubit gates. In such systems, the braid transition operators result in a phase change for the non computational state, and therefore it can be ignored. Overall, phases in the problem can also be ignored. Therefore the transition matrices can be projected onto SU(2) by a multiplication with $e^{\frac{i\pi}{10}}$, yielding for the generators

$$\sigma_1 = \begin{pmatrix} e^{\frac{-i7\pi}{10}} & 0 \\ 0 & -e^{\frac{-i3\pi}{10}} \end{pmatrix} \qquad \sigma_2 = \begin{pmatrix} -\tau e^{\frac{-i\pi}{10}} & -i\sqrt{\tau} \\ -i\sqrt{\tau} & -\tau e^{\frac{-i\pi}{10}} \end{pmatrix} \tag{1}$$

where $\tau = \frac{\sqrt{5}-1}{2}$.

In this paper we address the problem of finding a product of generator matrices for Fibonacci anyon braids. Although the methodology we propose can be extended to other braids, we focus on anyon braids since they are one of the best known in TQC [8,15]. As a target gate for computing the error we use $T = \begin{pmatrix} i & 0 \\ 0 & i \end{pmatrix}$.

3 Problem Formulation

Let $\mathbf{X} = (X_1, \ldots, X_n)$ denote a vector of discrete random variables. We use $\mathbf{x} = (x_1, \ldots, x_n)$ to denote an assignment to the variables. I denotes a set of indices in $\{1, \ldots, n\}$, and X_I (respectively x_I) a subset of the variables of $\mathbf{X}$ (respectively $\mathbf{x}$) determined by the indices in I.

In our representation for the quasiparticle braids problem, $\mathbf{X} = (X_1, \ldots, X_n)$ represents a braid of length n, where X_i takes values in $\{0, 1, \ldots, 2g-1\}$ and g is the number of generators. Given an order for the generators $\sigma_1, \sigma_2, \ldots, \sigma_g$, $X_i = j, j < g$ means that the matrix in position i is σ_{j+1}. If $X_i = j, j \geq g$, then the matrix in position i is $\sigma^{-1}_{(j-g)+1}$. For example, for generators shown in Equation (1), and $B = \sigma_1\sigma_1\sigma_2\sigma_2^{-1}\sigma_1^{-1}$, the corresponding braid representation is $\mathbf{x} = (0, 0, 1, 3, 2)$. Notice that this is a fixed length representation.

We are interested in the solution of an optimization problem formulated as $\mathbf{x}^* = argmax_{\mathbf{x}} f(\mathbf{x})$, where $f : S \to R$ is called the objective or fitness function. The optimum $\mathbf{x}^*$ is not necessarily unique. To evaluate the fitness function associated to a solution $\mathbf{x}$, firstly the product of braid matrices B is computed according to $\mathbf{x}$ and then the error ϵ is calculated from B as previously defined. The fitness function [8] is defined as:

$$f(\mathbf{x}) = \frac{1-\lambda}{1+\epsilon} + \frac{\lambda}{l} \tag{2}$$

where l is the braid's length, and λ serves to balance the two conflicting goals, i.e., having short braids or low approximation error. When $\lambda = 0$, braids are optimized only for the error and the function reaches its maximum value when this error is minimized.

We define functions $\hat{f}(\mathbf{x})$ and $\bar{f}(\mathbf{x})$ as two variations of function (2). Function $\hat{f}(\mathbf{x})$ is identical to $f(\mathbf{x})$, except that the effective length $\hat{l} = elen(B)$ is used instead of the braid's length. Function $\bar{f}(\mathbf{x})$ outputs the maximum value of the function for any of the braids contained in B that start from the first position, i.e. $\bar{f}(\mathbf{x}) = max_{\mathbf{y},\mathbf{y}\in\{(x_1),(x_1,x_2),\ldots,(x_1,\ldots,x_i),\ldots,(x_1,\ldots,x_n)\}} f(\mathbf{y})$.

4 Probabilistic Modeling of Braids

To optimize the braid problem we use EDAs, a class of evolutionary algorithms that capture and exploit statistical regularities in the best solutions. EDAs assume that such regularities exist. As a preliminary proof of concept on the existence of such regularities, we investigate the Boltzmann distribution for braids of manageable size. A similar approach has been successfully applied to investigate the dependencies that arise in the configurations of simplified protein models [13] and conductance-based neuron models [12].

4.1 Boltzmann Distribution

We use complete enumeration to define a probability distribution on the space of all possible braids for $n = 10$. Using the fitness value as an energy function, we associate to each possible braid a probability value $p(\mathbf{x})$ according to the Boltzmann probability distribution. The Boltzmann probability distribution $p_B(\mathbf{x})$ is defined as

$$p_B(\mathbf{x}) = \frac{e^{\frac{g(\mathbf{x})}{T}}}{\sum_{\mathbf{x}'} e^{\frac{g(\mathbf{x}')}{T}}}, \tag{3}$$

where $g(\mathbf{x})$ is a given objective function and T is the system temperature that can be used as a parameter to smooth the probabilities.

In our approach, $p_B(\mathbf{x})$ assigns a higher probability to braids that give a more accurate approximation to the target gate. The solutions with the highest probability correspond to the braids that maximize the objective function. We use an arbitrary choice of the temperature, $T = 1$, since our idea is to compare the distributions associated to different fitness functions with fixed T.

Using the Boltzmann distribution we can investigate how potential regularities of the fitness function are translated into statistical properties of the distribution. In particular, we are interested in the marginal probabilities associated to the variables and the mutual information between pairs of variables.

4.2 Statistical Analysis of the Braids Space

Figure 1 shows the univariate probabilities computed from the Boltzmann distribution for functions f and $\bar{f}$, and 10 variables. The search space comprises $4^{10} = 1,048,576$ braids. Univariate probabilities for function $\hat{f}$ were also computed, they are similar to probabilities obtained for f, and due to space constraints we do not include figures for this function. p_1, p_2, p_3, and p_4 respectively

represent the univariate probabilities for braid generators λ_1, λ_2, λ_1^{-1}, and λ_2^{-1}. For all the functions, higher probabilities for p_3 indicate that λ_1^{-1} is more likely to be present in the best solutions. This is the type of statistical regularities that can be detected and exploited by EAs that learn probabilistic models.

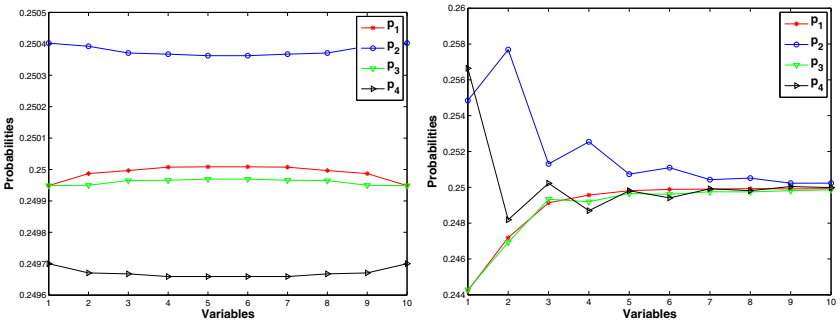


Fig. 1. Univariate probabilities of the Boltzmann distribution for: a) f, b) $\bar{f}$

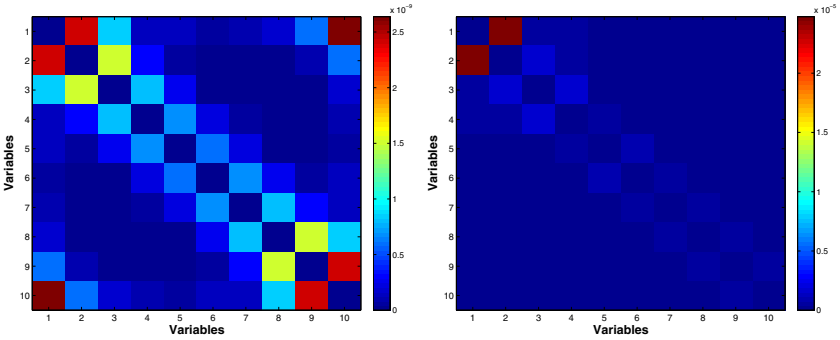
We compute the bivariate marginal distributions between every pair of variables and derive the values of the mutual information. The mutual information is a measure of statistical dependence between the variables and can serve to identify variables that are dependent. A strong dependence between two variables may indicate that their joint effect has a strong influence on the function and the optimizer should take into account this interaction. Figure 2 shows the mutual information computed for functions f and $\bar{f}$. It can be seen in Figure 2 that for the two functions the strongest dependencies are between adjacent variables, although for function f there is also a strong dependence between the first and the last variables. It can be also seen in Figure 2b) that the dependencies between adjacent variables decreases with the index for function $\bar{f}$.

Summarizing, the statistical analysis of the Boltzmann distribution shows that there are at least two types of regularities of the braid problem that are translated into statistical features. Firstly, there are different frequencies associated to the generators in the space of the best solutions. Secondly, there are strong dependencies between the variables, particularly those that are adjacent in the braid representation.

5 Estimation of Distribution Algorithms

EDAs use samples of solutions to learn a model that captures some of the regularities that may exist in the data. The pseudocode of an EDA is shown in Algorithm 1.

We work with positive distributions denoted by p. $p(x_I)$ denotes the marginal probability for $\mathbf{X}_I = \mathbf{x}_I$. $p(x_i \mid x_j)$ denotes the conditional probability



a) b)

Fig. 2. Mutual information computed from the Boltzmann distribution for: a) f, b) $\bar{f}$

Algorithm 1. Estimation of distribution algorithm

1 Set $t \Leftarrow 0$. Generate N solutions randomly.
2 **do** {
3 Evaluate the solutions using the fitness function.
4 Select a population D_t^S of $K \leq N$ solutions according to a selection method.
5 Calculate a probabilistic model of D_t^S.
6 Generate N new solutions sampling from the distribution represented in the model.
7 $t \Leftarrow t + 1$
8 } **until** Termination criteria are met.

distribution of $X_i = x_i$ given $X_j = x_j$. Three types of probabilistic graphical models are used: 1) Univariate model. 2) 1-order Markov model. 3) Tree model.

Univariate	1-order Markov	Tree
$p_u(\mathbf{x})$	$p_{MK}(\mathbf{x})$	$p_{\mathcal{T}}$
$\prod_{i=1}^{n} p(x_i)$	$p(x_1) \prod_{i=2}^{n} p(x_i \mid x_{i-1})$	$\prod_{i=1}^{n} p(x_i \mid pa(x_i))$

In the univariate model, variables are considered to be independent, and the probability of a solution is the product of the univariate probabilities for all variables. In the 1-order Markov model, the configuration of variable X_i depends on the configuration of its previous variable. In a probability distribution conformal with a tree, $Pa(X_i)$ 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. 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.

Univariate approximations are expected to work well for functions that can be additively decomposed into functions of order one (e.g. $g(\mathbf{x}) = \sum_i x_i$). However, other non additively decomposable functions can be easily solved with EDAs that

use univariate models (e.g. $g(\mathbf{x}) = \prod_i x_i + \sum_i x_i$) [9]. Therefore, it makes sense to test the univariate approximation for the braid problem. The 1-order Markov model captures only dependencies between adjacent variables, and the tree model can represent a maximum of $n-1$ bivariate dependencies. The computational cost of EDAs is mainly associated to the methods needed to learn and sample the models. The most complex EDA used in this paper is Tree-EDA which has a computational cost $O(n^2)$. Examples of EDAs that use univariate, 1-order Markov, and tree models are respectively presented in [10], [4] and [1,13] and details on the methods used to learn and sample the models can be obtained from these references.

5.1 Enhancements to the EDAs

We consider three enhancements to EDAs: 1) Use of a local optimizer. 2) Partial sampling. 3) Recoding.

As is the case of other EAs, EDAs can be enhanced by the incorporation of local optimizers. We use a greedy optimization algorithm that is applied during the evaluation of the population by the EDA. The algorithm starts from the solution generated by the EDA. In each iteration, the local optimizer evaluates all the $3n$ solutions that are different to the current solution in only one variable (the neighbor solutions). The next selected solution is the neighbor that improves the fitness of the current solution the most. The algorithm stops when none of the neighbors improves the fitness of the current solution.

During the sampling step of an EDA, all variables are assigned their values according to the probabilistic model and the sampling method. For the EDA that uses the univariate model, variables are independently sampled. For 1-order Markov and tree, probabilistic logic sampling (PLS) [5] is used. In both methods, all variables are assigned the new values. However, for some problems with higher-order interactions using a base-template solution can be better than generating each new solution from scratch.

In partial sampling, a solution of the population is selected and only a subset of its variables are sampled according to the model. We use two variants of partial sampling I) Partial sampling where the number of variables to be modified is randomly selected between 1 and n. II) Partial sampling, where the number of variables to be modified is randomly selected between 1 and $\frac{n}{2}$.

Recoding consists in modifying the representation of the solution after the fitness evaluation. For functions $\hat{f}$ and $\bar{f}$ it is possible to recode the solution by eliminating redundant generators (e.g., pairs $\sigma_i \sigma_i^{-1}$). The rationale of using recoding is that meaningful variables will be located closer to the beginning of the braid. Since solutions have a fixed length, the last variables will be kept unused, i.e. garbage information. Therefore, we devised two ways to fill these gaps: I) Leaving the unused variables as they were in the original solution. II) Replacing the unused variables by a reverse copy of the variables used in the evaluation. The second variant intends to replicate information that has proved to be "informative" about the problem.

6 Experiments

The main objective of our experiments is to evaluate the capacity of the EDAs to find optimal solutions to the braid problem. We run experiments for $n \in \{50, 100, 150, 200, 250\}$ in order to evaluate the scalability of the algorithms. Increasing n may lead to obtain braids with a smaller error. A second objective is to compare different variants of the problem formulation and of the algorithm.

6.1 Experimental Settings

Each EDA is characterized by 5 parameters:

- Use of local optimizer. 0: Only EDA is applied, 1: EDA is combined with greedy search as described in Section 5.1.
- Type of function and representation. 0: Function f, 1: Function $\bar{f}$ without recoding, 2: Function $\bar{f}$ with recoding I, 3: Function $\bar{f}$ with recoding II.
- λ value. 0:0.0, 1:0.01, 2:0.05, 3:0.1.
- Sampling method. 0: Normal, 1: Partial sampling I, 2: Partial sampling II.
- Type of probabilistic model. 0: Univariate, 1: 1-order Markov, 2: Tree.

The experimental settings were selected to investigate different aspects that influence the behavior of the algorithm. The total number of variants of the algorithm was $2{\times}4{\times}4{\times}3{\times}3 = 288$. All the algorithms use truncation selection, in which the best 5% of the population is selected. EDAs that do not incorporate the greedy local search use a population size $N = 10000$. For these EDAs, the number of generations was dependent on n as $N_g = 15n$. Due to the large number of evaluations spent by the greedy search method, the population size for all hybrid EDAs was $N = 100n$ and the number of generations was fixed to $N_g = 100$. For each EDA variant, 100 experiments were run.

6.2 Best Solutions Found by EDAs

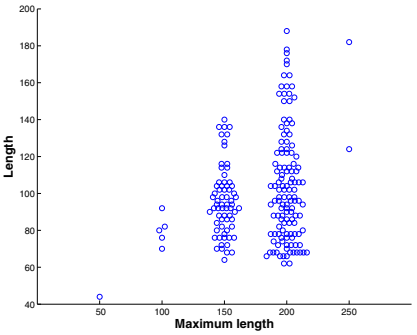
Figure 3a) and Table 1 respectively show the parameters that characterize the best braids found by the EDAs for each value of n, and the braids. In Figure 3a), we also show an estimate of the length of the braids ($O[log_{10}^{3.97}(1/\epsilon)]$) that would compute the Solovay-Kitaev algorithm [3] to obtain the same error ϵ of our best solutions. The lengths of our solutions clearly outperform these estimates. Figure 3b) shows the length of all the best solutions achieved for each value of n. It can be observed in Figure 3 that EDAs are able to find several braids with different lengths for $n = 150$ and $n = 200$.

6.3 Behavior of the EDA Variants

We further investigate the behavior of the different EDA variants. Figure 4 shows the violin plots [6] with the distribution of the best values found in all the executions for: a) All EDA variants without local optimizer (14400 runs), b) All

n	l	ϵ	$log_{10}(\epsilon)$	$log_{10}^{3.97}(\frac{1}{\epsilon})$
50	44	4.8435×10^{-4}	-3.3148	116.47
100	70	8.3527×10^{-6}	-5.0782	633.37
150	64	8.3527×10^{-6}	-5.0782	633.37
200	62	8.3527×10^{-6}	-5.0782	633.37
250	124	3.5038×10^{-6}	-5.4555	841.82

a)



b)

Fig. 3. a) Parameters of the best braids found by the EDAs for each value of n. b) Length of the best solutions found for each value of n

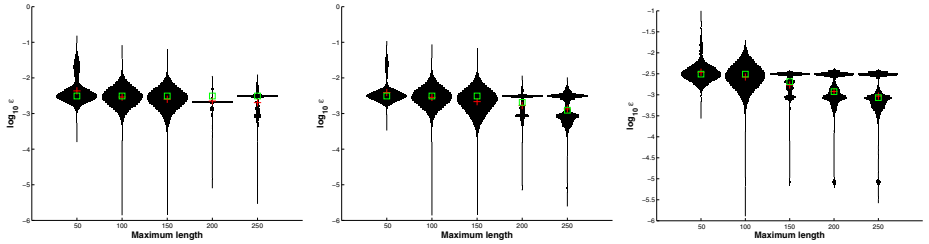
EDA variants that incorporate the greedy search (14400 runs), c) EDAs with local optimizer, recoding type II, and that use partial sampling type II (300 runs). Each violin plot shows a histogram smoothened using a kernel density with Normal kernel. The mean and median are shown as red crosses and green squares, respectively.

Table 1. Best braids found by the EDAs for each value of n

n	braid
50	$\sigma_1^{-1}\sigma_2^3\sigma_1^{-2}\sigma_2\sigma_1^{-1}\sigma_2\sigma_1^{-2}\sigma_2^3\sigma_1^{-1}\sigma_2\sigma_1^{-1}\sigma_2\sigma_1^{-2}\sigma_2\sigma_1^{-2}\sigma_2\sigma_1^{-1}\sigma_2^3\sigma_1^{-1}\sigma_2^2\sigma_1^{-1}\sigma_2\sigma_1^{-1}\sigma_2\sigma_1^{-2}\sigma_2^2\sigma_1^{-3}\sigma_2^2$
100	$\sigma_1\sigma_2^{-2}\sigma_1^4\sigma_2^{-1}\sigma_1\sigma_2^{-4}\sigma_1\sigma_2^{-1}\sigma_1^3\sigma_2^{-1}\sigma_1^2\sigma_2^{-2}\sigma_1^4\sigma_2^{-1}\sigma_1\sigma_2^{-4}\sigma_1\sigma_2^{-1}\sigma_1^6\sigma_2^2\sigma_1\sigma_2\sigma_1\sigma_2^2\sigma_1^3\sigma_2^5\sigma_1^{-1}\sigma_2\sigma_1^{-3}\sigma_2\sigma_1^3\sigma_2^5$
150	$\sigma_2\sigma_1^{-1}\sigma_2\sigma_1^{-1}\sigma_2^{-1}\sigma_1^2\sigma_2^{-1}\sigma_1\sigma_2^{-4}\sigma_1^2\sigma_2^{-2}\sigma_1^{-1}\sigma_2^{-1}\sigma_1^{-2}\sigma_2^{-4}\sigma_1^2\sigma_2^{-4}\sigma_1\sigma_2^{-1}\sigma_1^2\sigma_2^{-1}\sigma_1^{-1}\sigma_2\sigma_1^{-3}\sigma_2^{-1}\sigma_1\sigma_2^{-1}\sigma_1^3\sigma_2^{-1}\sigma_1\sigma_2^{-4}\sigma_1^2\sigma_2^{-4}\sigma_1^2\sigma_2^{-3}$
200	$\sigma_2^{-2}\sigma_1^4\sigma_2^{-1}\sigma_1\sigma_2^{-4}\sigma_1\sigma_2^{-1}\sigma_1^2\sigma_2^{-1}\sigma_1^3\sigma_2^{-1}\sigma_1\sigma_2^{-4}\sigma_1\sigma_2^{-1}\sigma_1^4\sigma_2^{-1}\sigma_1\sigma_2^{-2}\sigma_1^2\sigma_2^2\sigma_1\sigma_2^{-1}\sigma_1\sigma_2^{-1}\sigma_1^{-5}\sigma_2\sigma_1^{-1}\sigma_2^{-6}\sigma_1^{-2}\sigma_2^3$
250	$\sigma_1\sigma_2\sigma_1^{-1}\sigma_2\sigma_1^{-1}\sigma_2^{-1}\sigma_1\sigma_2^{-1}\sigma_1^{-3}\sigma_2^4\sigma_1^{-2}\sigma_2^4\sigma_1^{-1}\sigma_2\sigma_1^2\sigma_2\sigma_1^{-1}\sigma_2^4\sigma_1^{-2}\sigma_2^4\sigma_1^{-3}\sigma_2^{-1}\sigma_1\sigma_2^{-1}\sigma_1^{-1}\sigma_2^2\sigma_1^{-1}\sigma_2^{-1}\sigma_1^2\sigma_2^{-1}\sigma_1\sigma_2^{-4}\sigma_1^2\sigma_2^{-4}\sigma_1\sigma_2^{-1}\sigma_1^{-1}\sigma_2^{-1}\sigma_1^{-4}\sigma_2^{-1}\sigma_1^{-1}\sigma_2^{-1}\sigma_1^4\sigma_2^{-1}\sigma_1\sigma_2^{-4}\sigma_1^2\sigma_2^{-4}\sigma_1\sigma_2^{-2}\sigma_1^{-1}\sigma_2^5\sigma_1^{-1}\sigma_2\sigma_1^{-4}\sigma_2^2\sigma_1^{-4}\sigma_2^2\sigma_1^{-4}\sigma_2^2\sigma_1^{-1}$

In Figure 4, the modes of the Normal distribution indicate the existence of a local optimum of the error with a very wide basin of attraction. This local optimum has value $log_{10}f(\epsilon) = -2.50785$ and the majority of the EDA runs can be trapped in this value. Differences between the EDAs due to the application of the greedy method can be appreciated for $n = 200$ and $n = 250$ (Figures 4a) and 4b)). Also, Figure 4c) reveals how a particular combination of the EDA's parameters can improve the results of the search.

There are a number of commonalities between the best EDA variants. Except in one case, all EDAs use recoding of type II. Similarly, except in one case, in all the variants $\lambda \in \{0.01, 0.05\}$. Except in two cases, the sampling method selected was partial sampling. The application of the local optimizer notably improved



a) b) c)

Fig. 4. Violin plots showing the distribution of the best values found in all the executions for: a) All EDAs variants without local optimizer (14400 runs), b) All EDAs variants with local optimizer (14400 runs), c) EDAs with local optimizer, recoding type II, and that use partial sampling type II (300 runs)

the results for $n \in \{150, 200\}$ but in terms of the best solution found it did not have an important influence for the other values of n.

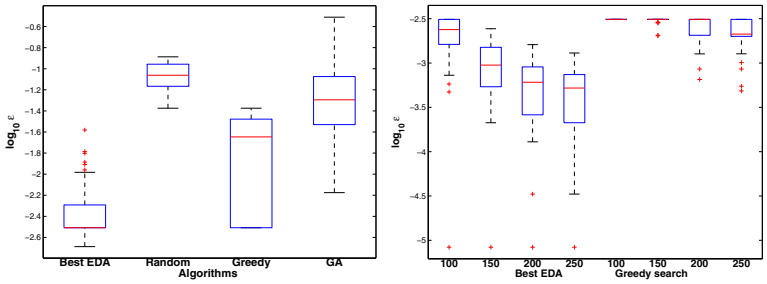
As a summary, we recommend to use an EDA that adds the greedy search, and uses partial sampling of type II and the 1-order Markov model since it is less complex than the tree and results achieved using the two models are similar.

6.4 Improvement Over Other Search Methods

As a final validation of our method, we compare our best EDA variant with the results achieved using a random search, the greedy local optimizer, and the GA introduced in [8]. For the random search, we randomly generated 10000 solutions and selected the best solution according to function $\bar{f}, \lambda = 0.01$. The same experiment was repeated 100 times to select the 100 "best" solutions for $n \in \{50, 100, 150, 200, 250\}$.

A similar procedure was followed for the greedy local search. The local optimizer was applied to each of the 10000 solutions until no improvement was possible. For the GA, we used the results of the 100 GA runs analyzed in [8]. Since these results were obtained using solutions of different length, and with a different number of evaluations, care must be taken to interpret the differences. We only compare the GA results with the other algorithms for $n = 50$. Similarly, the results of the random search were very poor for $n > 50$ and we only include them in the comparison for $n = 50$. Results are shown in Figure 5a). In the boxplots, the central mark is the median, the edges of the box are the 25th and 75th percentiles, the whiskers extend to the most extreme data points not considered outliers, and outliers are plotted individually.

Using the 100 best solutions for each of the four algorithms, a multiple comparison test was applied to test the null hypothesis that samples corresponding to every pair of algorithms are drawn from the same population. The multiple comparison test uses the Tukey's honestly significant difference criterion. Every pair-wise comparison is based on the Kruskal-Wallis test, a nonparametric



a) b)

Fig. 5. a) Comparison between the Best EDA variant, the random search, the greedy local search and the GA for $n = 50$. b) Comparison between the Best EDA variant and the greedy local search $n \in \{100, 150, 200, 250\}$

version of the classical one-way ANOVA. The significance criterion was $\alpha = 0.05$. The application of the test identified statistical differences between each pair of algorithms and they were ranked as: 1) Best-EDA, 2) Greedy, 3) GA, 4) Random.

The results of the comparison between the EDA and the greedy search for $n > 50$ are shown in Figure 5b). The application of the Kruskal-Wallis test ($\alpha = 0.05$) found significant differences between the EDA and the Greedy algorithm for all n. Furthermore, it can be seen in Figure 5b) that as n increases the algorithm is able to scale and find better solutions.

7 Conclusions

In this paper we have proposed for the first time the use of probabilistic modeling of the search space to address the problem of approximating a quantum gate as a product of braid generators. We have shown that some of the problem characteristics can be translated into statistical regularities of the Boltzmann distribution. This result indicates that capturing and exploiting statistical regularities emerges as a sensible approach to the quasiparticle braid problem.

In a second step we have shown the effectiveness of EDAs to find short braids that provide accurate approximations. The best braids obtained with our EDAs have lengths up to 9 times shorter than those expected from braids of the same accuracy obtained with the Solovay-Kitaev algorithm and had not been previously reported to be found by the GA approach.

Acknowledgments. R. Santana 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). H. G. Katzgraber acknowledges support from the NSF (Grant No. DMR-1151387).

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 (1997)
2. Bonesteel, N.E., Hormozi, L., Zikos, G., Simon, S.H.: Braid topologies for quantum computation. Physical Review Letters 95(14), 140503 (2005)
3. Dawson, C.M., Nielsen, M.A.: The Solovay-Kitaev algorithm. arXiv preprint quant-ph/0505030 (2005)
4. De Bonet, J.S., Isbell, C.L., Viola, P.: MIMIC: Finding optima by estimating probability densities. In: Mozer, et al. (eds.) Advances in Neural Information Processing Systems, vol. 9, pp. 424–430. The MIT Press, Cambridge (1997)
5. Henrion, M.: Propagating uncertainty in Bayesian networks by probabilistic logic sampling. In: Lemmer, J.F., Kanal, L.N. (eds.) Proceedings of the Second Annual Conference on Uncertainty in Artificial Intelligence, pp. 149–164. Elsevier (1988)
6. Hintze, J.L., Nelson, R.D.: Violin plots: a box plot-density trace synergism. The American Statistician 52(2), 181–184 (1998)
7. 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)
8. McDonald, R.B., Katzgraber, H.G.: Genetic braid optimization: A heuristic approach to compute quasiparticle braids. Physical Review B 87(5), 054414 (2013)
9. Mühlenbein, H., Mahnig, T., Ochoa, A.: Schemata, distributions and graphical models in evolutionary optimization. Journal of Heuristics 5(2), 213–247 (1999)
10. Mühlenbein, H., Paaß, G.: From recombination of genes to the estimation of distributions I. Binary parameters. In: Ebeling, W., Rechenberg, I., Voigt, H.-M., Schwefel, H.-P. (eds.) PPSN 1996. LNCS, vol. 1141, pp. 178–187. Springer, Heidelberg (1996)
11. Read, N., Rezayi, E.: Beyond paired quantum Hall states: parafermions and incompressible states in the first excited Landau level. Physical Review B 59(12), 8084 (1999)
12. Santana, R., Bielza, C., Larrañaga, P.: Conductance interaction identification by means of Boltzmann distribution and mutual information analysis in conductance-based neuron models. BMC Neuroscience 13(suppl 1), P100 (2012)
13. Santana, R., Larrañaga, P., Lozano, J.A.: Protein folding in simplified models with estimation of distribution algorithms. IEEE Transactions on Evolutionary Computation 12(4), 418–438 (2008)
14. Sarma, S.D., Freedman, M., Nayak, C.: Topological quantum computation. Physics Today 59(7), 32–38 (2006)
15. Xu, H., Wan, X.: Constructing functional braids for low-leakage topological quantum computing. Physical Review A 78(4), 042325 (2008)