Genetic Algorithm Niching by (Quasi-)Infinite Memory

Adrian Worring
Computational Biology & Simulation Group, TU Darmstadt
Darmstadt, Germany
adrian.worring@protonmail.com

Benjamin E. Mayer
Computational Biology & Simulation Group, TU Darmstadt
Darmstadt, Germany
mayer@cbs.tu-darmstadt.de

Kay Hamacher
Computational Biology & Simulation Group, TU Darmstadt
Darmstadt, Germany
hamacher@bio.tu-darmstadt.de

ABSTRACT

Genetic Algorithms (GA) explore the search space of an objective function via mutation and recombination. Crucial for the search dynamics is the maintenance of diversity in the population.

Inspired by tabu search we add a mechanism to an elitist GA's selection step to ensure diversification by excluding already visited areas of the search space. To this end, we use Bloom filters as a probabilistic data structure to store a (quasi-)infinite history.

We discuss how this approach fits into the niching idea for GAs, finds an analogy in generational correlation via epi-genetics in biology, and how the approach can be regarded as a co-evolutionary edge case of previous niching techniques.

Furthermore, we apply this new technique to an NP-hard combinatorial optimization problem (ground states of Ising spin glasses). We find by large-scale hyperparameter scans that our elitist GA with quasi-infinite memory consistently outperforms its respective standard GA variant without a Bloom filter.

CCS CONCEPTS

• **Theory of computation** → *Tabu search*; **Bloom filters and hashing**; Evolutionary algorithms; • **Mathematics of computing** → **Combinatorial optimization**; • **Computing methodologies** → **Genetic algorithms**.

KEYWORDS

Genetic Algorithms, Tabu Search, Combinatorial Optimization, Hybridization, Population management and niching

ACM Reference Format:
Adrian Worring, Benjamin E. Mayer, and Kay Hamacher. 2021. Genetic Algorithm Niching by (Quasi-)Infinite Memory . In *2021 Genetic and Evolutionary Computation Conference (GECCO '21), July 10–14, 2021, Lille, France.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3449639.3459365

1 INTRODUCTION

Since their inception Genetic Algorithms (GA)[15, 26] are of great interest to researchers of complex systems, computational sciences, and mathematics, as well as for practitioners who want to solve pragmatic optimization problems in a heuristic fashion.

While many aspects of evolutionary dynamics are quite well understood in principal [20], the quest for optimal evolutionary schemes and optimal (hyper-)parameters in practical applications is still ongoing. One way to broaden the applicability and improve the performance of GA and other evolutionary computing approaches is niching.

1.1 Niching & Previous Work

The niching idea [29] takes its motivation from ecology. Typically, (genetic) diversity on islands and other (geo-)spatially defined regions is higher [17]. A main distinction of niching approaches for GAs and other evolutionary algorithms – according to Mahfoud [23] – is spatial vs. temporal niching. Our approach laid out below (cmp. Sec. 2) couples the whole evolutionary history and thus falls into temporal niching.

Typically, one employs niching to have an evolutionary algorithm work in separated portions of the search space on (locally) optimal solutions and multi-objective optimization [11].

Biologically, though, niching also occurs whenever portions of a genome become inaccessible due to, e.g., epi-genetics (see Sec. 2).

1.2 Tabu Search & Previous Work

The Tabu Search [13] (TS) method stores individual configurations in memory and excludes them from future iterations – eventually excluding any revisit of a potential solution. This increases the volume of the explored search space and thus diversity. As the space requirements would increase, one typically implements a so-called tabu tenure, that is a maximum number of iterations a visited solution is kept.

1.2.1 Related Work on GAs. The overall idea to include some memorization mechanisms in GAs is not new: Glover et al. [14] discussed potential hybridization, while Kurahashi and Terano [21] introduced the idea of a tabu history in determination of parents during a GA's recombination phase. They proposed several tabu lists to achieve different tabu tenures and benchmarked this for some standard test function in heuristic optimization.

Antony and Jayarajan [1] added a tabu list to an GA with the explicit goal of diversifying the set of solutions. In data mining, a similar idea was proposed [22]. Mantawy et al. [24] hybridized GAs with TS and Simulated Annealing for resource allocation problems.

All this work, however, is characterized by finite tabu tenures, eventually restricted to a tiny subset of the search space. As soon as a visited configuration is beyond that, it is "forgotten" and can be revisited.

GECCO '21, July 10–14, 2021, Lille, France

ACM ISBN 978-1-4503-8350-9/21/07…$15.00
https://doi.org/10.1145/3449639.3459365

1.2.2 Related Work on Tabu Search. Hamacher [18] combined the idea of Tabu Search with a variant of Monte-Carlo based optimization, namely the Stochastic Tunneling approach [19, 31]. As the list of visited solutions grows during a run, the tabu list typically grows as well. The novelty in [18] is based on the usage of a Bloom filter (see below, Sec. 1.2.3) to work in constant space (and thus also in constant time for look-up).

1.2.3 Bloom Filter. A Bloom filter [8, 9, 27] is a probabilistic data structure to determine (approximate) set membership. In its original form two operations are available: 1) insertion of an element and 2) query whether an element is member of the represented set. Both operations take constant time. The required memory is also constant and can be chosen beforehand. When testing an element there is a probability of a false positives. The probability of a false negative is zero, though. Bloom filters can store an unlimited number of elements but the false positive rate increases the more are inserted.

To achieve this a Bloom filter has a bit array of size m and k different hash functions. An empty filter is initialized with all bits set to zero. To add an element the element e is run through the k hash functions resulting in a vector of hashes

$$\mathbf{v}_e = [h_1(e), \ldots, h_k(e)].$$

For each hash value $h_i(e)$ the bit at index $h_i \mod m$ in the bit array is set to 1. When an element is tested the hash functions are evaluated again and the bits at the positions $h_i \mod m$ are retrieved. If at least one is 0 the element was never entered to the filter before. If all are 1 the element has possibly been inserted.

It should be emphasized that the hash functions h_i do not retain a "neighborhood" property: although a pair e and e' might be "close" in search space the distance between $h_i(e)$ and $h_i(e')$ is not bounded at all.

2 TEMPORAL NICHING BY (QUASI-)INFINITE HISTORY

With the rise of epi-genetics, it is a well known biological fact that not only genetic information is transferred from generation to generation, but rather also the "availability" is encoded via, e.g., methylation patterns [12].

By such mechanisms generations temporally interact while the underlying (genetic) search space remains. The inclusion of such additional information and mechanisms allow for faster adaptation in biological systems. We will minimc this mechanism by excluding portions of the search space – namely, previously visited configurations – from future accessibility. This implements the Tabu Search idea and lends itself to niching in biology.

In the notion of Beyer and Schwefel [4] we will use this arising niching to diversify a GA population in order to cover the search space of our optimizatin problem more efficiently. *We thus leverage niching to increase population diversity for our single goal (to attain the global optimum)* and not as much a method to obtain several "good" minima.

Using Bloom filters helps us to achieve a (quasi-)infinite historical interaction of generations, partition the search space by the search history, and thus realizes genetic drift in peripatric speciation. We augment this with a minimal elitist element by allowing for 1% of individuals to be rescued from the emerging niches into a each new generation. Note, that due the elitist mechanism a configuration for a global minima will never enter the Bloom filter.

Our approach can be seen as an extreme form of *clearing* [28] to eliminate previously (non-elite) solutions.

3 OUR GA VARIANT WITH (QUASI-)INFINITE TEMPORAL COUPLING

Algorithm 1 describes our implementation. Solutions are implemented as fixed-length bit strings[1]. Single-point crossover is used as recombination operator (CrossOver) and mutations consist of a single bit flip (Mutate). The crossover locus and the position of the bit flip are drawn from a uniform random distribution. Multiple selection methods (Selection-Method) are implemented: Roulette Wheel Selection, (Exponential) Rank Selection, Truncation Selection and Tournament Selection. In the subsequent parts we restrict the choice Selection-Method to tournament selection as the best performing selection scheme in a preliminary hyperparameter scan. The n_{elite} individuals with the best fitness will be carried over to the next generation without undergoing recombination or mutation (thus implementing an elitist GA) or filtering by the Bloom filter.

A simple Bloom filter was implemented using the `MurmurHash3` hash function [2]. Due to this choice, the created hashes do not retain "neighborhood" as described above and the any false positives do not create a "forbidden region", but are (almost) independently distributed within the whole search space.

After selection, recombination and mutation each individual has the probability α_2 of being tested by the Bloom filter. If the individual is found in the filter it will be discarded and a new one will be selected. Keep in mind that there is a probability of false positives. So some will be discarded although they were never inserted into the filter[2]. Once the next generation has been constructed all individuals except for the n_{elite} with the highest fitness have the probability α_1 of being added to the Bloom filter.

For $\alpha_1 = \alpha_2 = 0$ we obtain a traditional "non-Bloom" GA.

The implementation is written in `C++` and uses `xoshiro256**` [6] as pseudo-random number generator.

The new method is efficient, whenever creation, mutation, and recombination is "cheap" in comparison to the evaluation of the objective function: the Bloom filter criterion might reject several dozens of configurations per step and only evaluates the objective function *after* a (new) individual has passed the Bloom filter criterion. Thus, whenever the computational time is dominated by the evaluation of the objective function the overhead we introduce is negligible as Bloom filters work in constant time and space.

4 APPLICATION

4.1 A Complex Test Instance

We apply our new technique to a combinatorial optimization problem which has multiple minima, exponential increase of the number

[1]This is the natural representation for our test problem in application Section 4. Note, that this design choice does not imply any restrictions of our Bloom filter idea in other applications or representations.

[2]This could potentially also hold for the (unique) global optimum. Pragmatically, one can either accept this as the algorithm still finds better solution than a "non-Bloom" variant or increase the Bloom hyperparameters to arbitrarily small false positive rates.

Algorithm 1 GA with Bloom filter; Bern(x) is a Bernoulli experiment with probability x of succeeding; note: for $\alpha_1 \cdot \alpha_2 = 0$ the algorithm implements a traditional GA.

$G \leftarrow n_{\text{pop}}$ random individuals
calculate the fitness of individuals in G
while target fitness or max generations is not reached **do**
  update statistics and sort G by fitness in ascending order
  for $i \leftarrow 0$ **to** $n_{\text{pop}} - n_{\text{elite}} - 1$ **do**
    if Bern(α_1) **then**
      add $G[i]$ to bloom filter
  $G_{\text{new}} \leftarrow n_{\text{elite}}$ best individuals of G
  while G_{new}.size $< n_{\text{pop}}$ **do**
    $parent_1, parent_2 \leftarrow$ SELECTION-METHOD(G)
    if Bern(ρ) **then**
      $child_1, child_2 \leftarrow$ CROSSOVER($parent_1, parent_2$)
    else
      $child_1, child_2 \leftarrow parent_1, parent_2$
    if Bern(μ) **then**
      $child_1 \leftarrow$ MUTATE($child_1$)
    if Bern(μ) **then**
      $child_2 \leftarrow$ MUTATE($child_2$)
    if Bern($1 - \alpha_2$) or $child_1$ is not in bloom filter **then**
      $G_{\text{new}} \leftarrow G_{\text{new}} \cup \{child_1\}$
    if Bern($1 - \alpha_2$) or $child_2$ is not in bloom filter **then**
      $G_{\text{new}} \leftarrow G_{\text{new}} \cup \{child_2\}$
  if G_{new}.size $> n_{\text{pop}}$ **then**
    remove last individual from G_{new}
  $G \leftarrow G_{\text{new}}$
  calculate fitness of individuals in G

of minima with the system size and potentially large barriers between local minima.

Spin glasses of the Ising type [5] fulfill all these required specifications. The Ising model restricts the variables to two discrete values ("up" and "down") mapped to integers +1 and −1, respectively. The energy function – also called the Hamiltonian – is[3]

$$\min E(\vec{s}) = \frac{1}{4} \sum_{<i,j>} J_{ij} s_i s_j \qquad \forall i \in \{1 \ldots N\} \ : \ s_i \in \{-1; +1\} \tag{1}$$

Here, we restrict the sum of spin pairs $< i, j >$ in Eq. 1 to a regular 2D grid of $N = L \cdot L$ spins[4] with the notation of $< i, j >$ as direct neighbors on the 2D grid. The $J_{ij} \sim \mathcal{N}(0, \sigma^2)$ are normally distributed around a vanishing mean with standard deviation σ and model the interaction strength between spin i and spin j. The vector describing the optimal state (lowest energy) of spins is $\vec{s}^* := \left(s_1^*, s_2^*, \ldots, s_N^*\right)$.

In general the energy minimization problem of Ising spin glasses is NP-hard [3]. In our special case, however, polynomial approaches exist and we can construct test instances via a readily available online service [30].

[3]Note, that the spin variables in alternative notation the sping variables are restricted to the values $-1/2$ and $+1/2$ and the prefactor $1/4$ in the Hamiltonian is omitted.
[4]2d grid with side length $L := \sqrt{N}$

4.2 Analysis of Results

4.2.1 Availability. All results in csv-format, the source code, as well as an interactive viewer for the parameter-hypercube are available on the Internet under http://biosrv0.compbiol.bio.tu-darmstadt.de/ga-bloom.

4.2.2 Metrics. We use multiple metrics to evaluate GA runs, the most important one being the relative error [25]

$$\Delta E_{\text{rel}} := \left| \frac{E_{\text{sol}} - E_{\text{ga}}}{E_{\text{sol}}} \right|, \tag{2}$$

where E_{ga} is the energy of a solution found by the GA and E_{sol} is the energy of the global optimum as determined by the previously mentioned online service [30]. Additionally, we looked at what percentage P of runs found the ground state.

For spin glasses an order parameter is the magnetization m of a configuration, that is the difference of number of up- vs. down-spins. Instead of m we just use the number of "up" spins (U) with is equivalent to the magnetization as $m = 2 \cdot U - L \cdot L$.

To assess the evolutionary dynamics we analyze the loss of diversity Blickle and Thiele [7] as the number of individuals that are not included into a new generation.

Finally, we also determined the selection intensity as defined by Blickle and Thiele [7]. It aims to measure the progress due to selection and is defined as

$$I_t := \left| \frac{\mu_{t+1} - \mu_t}{\sigma_t} \right|, \tag{3}$$

where μ_t is the mean fitness of generation t and σ_t is its standard deviation taken over the present population at t.

4.2.3 Parameter Scan. To keep the computational effort manageable the population size was fixed at 1,000 individuals, the number of elites at 10 and tournament selection[5] with a tournament size of 5 was used through out all runs.

The mutation rate μ, the recombination rate ρ, as well as the filtering parameters α_1 and α_2 were scanned on an equally-sized grid for the following values[6]

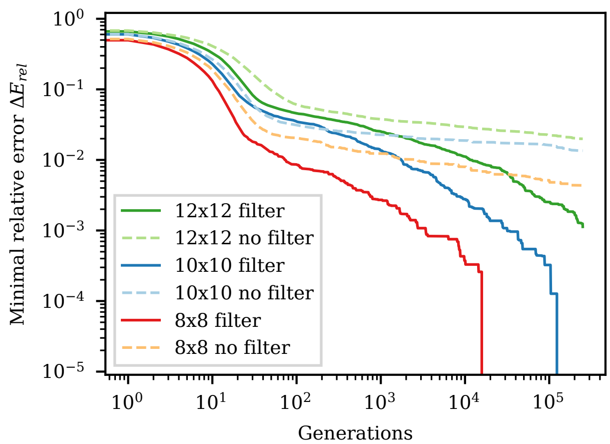
$$(\mu, \rho, \alpha_1, \alpha_2) \in \{0.1, 0.2, \ldots, 1.0\}^4.$$

All combinations as well as the case $\alpha_1 = \alpha_2 = 0$ which is a simple GA without filtering were tested on Ising spin glass models of three different sizes. First a scan with instances of size 8×8 and a Bloom filter of size 64MB with 7 hash functions limited to 30 000 generations was conducted. Then two more with 10×10 and 12×12 instances with a Bloom filter of size 256MB with 6 hash functions limited to 200 000 and 250 000 generations respectively were completed. For each parameter combination 100 runs with different spin glass instances were performed.

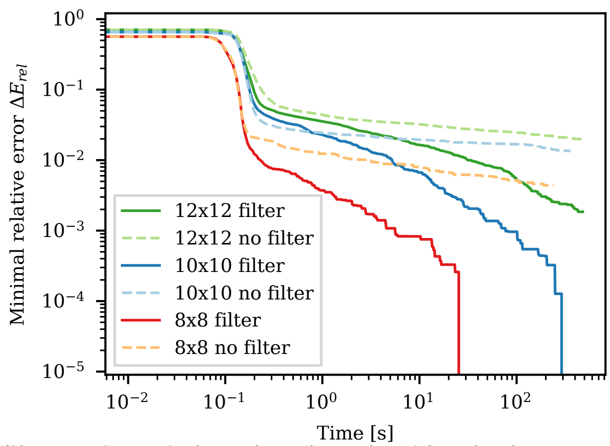
4.3 Overall performance

Fig. 1 compares the average relative error over the 100 runs for the best parameter choices with and without a Bloom filter, respectively. Clearly, the Bloom filter variant performs superior. The used parameters are listed in Tab. 1.

[5]Permutations were used to make sure every individual takes part in the same amount of tournaments [16].
[6]Note: for Tab. 2a we improved the resolution for α_1 further.



(a) Limited to 250 000 generations. Generation 0 is the initial random population.



(b) Limited to eight (12×12), six (10×10) and four (8×8) minutes.

Figure 1: Comparison of the average relative error of runs for the best parameter combinations found. Averaged over 100 runs. The parameters with the overall lowest minimal relative error were chosen. See Tab. 1.

We show the results for both the number of generations and with an overall computational time threshold on the abscissa. The last one being necessary as the computation under high rejection rate in the Bloom filter can take much longer than in the classical GA (cmp. Fig. 5). Note, that even in the case of our objective function of Eq. 1 where the evaluation time is negligible the (quasi-)infinite GA performs much better. This finding would be even more pronounced for objective functions that have a larger share of the overall runtime.

4.4 Filtering Parameters

Besides the relative error in Fig. 1b (as an average over all individuals and thus potentially highly skewed by a few "bad" actors) important insight can be gained from the analysis of how frequently individuals in the whole population reached the global optimum.

To examine the influence of the filtering parameters α_1 and α_2 on such a success rate, we show in Fig. 2 the fraction of runs P reaching the (known) global optimum as given by the particular spin state $\vec{s}^*$. The best parameter combination with Bloom filtering found the optimal solution in 81 out of 100 runs ($P = 81\%$) while the best result for the traditional variant[7] was $P = 5\%$.

Table 1: The best parameter sets for "non-Bloom" GAs ($\alpha_1 = 0$, $\alpha_2 = 0$) and GAs using Bloom filter ($\alpha_1 > 0$, $\alpha_2 > 0$) as determined by the mean minimal relative error for the system sizes. Whenever multiple parameter sets showed a vanishing error the configuration with the lowest median generation in which the ground state was found was chosen.

L	μ	ρ	α_1	α_2	Relative Error ΔE_{rel}
8×8	1.0	0.2	0.0	0.0	0.00438
10×10	1.0	0.7	0.0	0.0	0.01366
12×12	1.0	0.2	0.0	0.0	0.01991
8×8	0.6	0.8	1.0	0.9	0
10×10	0.2	0.4	1.0	1.0	0
12×12	0.4	1.0	0.8	1.0	0.00113

The plots suggest that higher values of α_2 improve the performance significantly, while α_1 only has a minor effect.

The necessary maximum mutation rate for the non-filtering GA suggests the need for high "velocity" in sequence space. However, for the Bloom filter variant the parameter choices with high α_2 performed better with smaller mutation rates as shown in Fig. 2. It is also striking how drastically the performance improves with a value of $\alpha_2 = 0.9$ compared to $\alpha_2 = 0.7$. This suggests that we not necessarily have to keep track of all visited individuals (small α_1 are sufficient), but rather that it is always beneficial to pay attention to the historical knowledge we have ($\alpha_2 > 0$).

Tab. 2a shows this in more detail. Here, we scanned the α_1 parameter and extracted the best parameters α_2, μ, ρ as well as the metrics P and ΔE_{rel}. Clearly, using Bloom filtering improves upon P and ΔE_{rel}. Consistently, the best mutation rate μ tends to be smaller for larger αs. Furthermore, only higher α_2 values (0.8-1.0) were encountered. This suggests that – given some probability to memorize visited configurations – the procedure tends to leverage this information in its entirety (1.0) or at least to a large extent ($0.8 - 0.9$).

Furthermore, the switch from $\alpha_1 = 0$ (no Bloom filter) to $\alpha_1 = 0.01$ (1% coverage of generated individuals) reduces ΔE_{rel} by a factor $\approx 1/5$ and increases the success rate P by roughly an order of magnitude.

In Tab. 2b shows an orthogonal analysis: here we condition on α_2 and find the parameters α_1, μ, ρ that minimize ΔE_{rel}. In comparison, the broader range of α_1 values encountered here, suggests – together with the remarks from above on Tab. 2a – that it is not necessary to memorize always almost everything, but that it is always beneficial to leverage what the procedure "knows" about previously visited configurations.

[7] with parameters $\mu = 1.0$ and $\rho = 1.0$

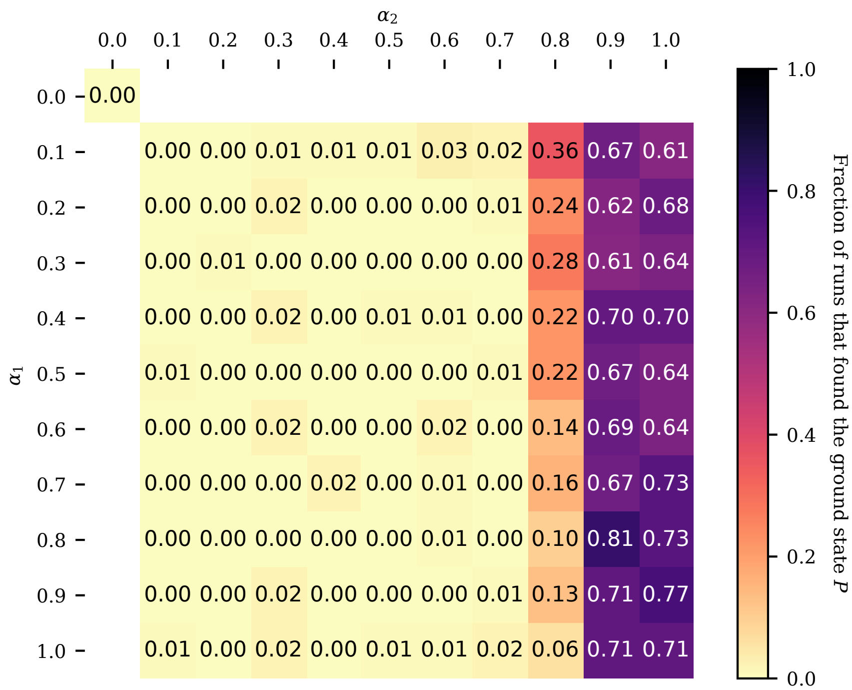


Figure 2: The fraction of runs P that found the ground state for 12×12 spin glasses for different values of α_1 and α_2 and $\mu = 0.5$ and $\rho = 1.0$. 100 runs with 250 000 generations were conducted per cell.

4.5 Larger Systems

In Tab. 1 we have already seen that the Bloom-filter GA performs better for various, small to intermediate system sizes. We also examined how well a good parameter combination is transferable to larger search spaces. The parameter choice with the lowest minimal relative error on 12×12 instances was used for instances from 6×6 up to 20×20. Again, for each system size 100 instances were generated. Fig. 3a shows the distributions and means of the minimal relative errors.

To assess the effect of the Bloom filter we also show results for a GA without a Bloom filter at the same $\mu = 0.4$ and $\rho = 1.0$ from Tab. 1. Again, the performance of the traditional GA is inferior, thus at least transferability of parameters is in question.

While from $L > 16$ onward we never encountered the exact global optimum within 250, 000 generations, the distribution of attained objective function values does not broaden very much, while at the same time the increase of the relative error is comparable small.

4.6 Computational Overhead

Adding elements to the Bloom filter and querying whether an element is in the filter requires additional computational effort of $\sim O(1)$. On top of these basic operations we encountered an additional effect: whenever an individual is discarded a new one needs to be derived via selection, recombination, and mutation. This could potentially have a negative effect on the run time of the code. To quantify this effect run times were recorded. Fig. 4 shows how the time per run and per generation is influenced by α_2.

Non-surprisingly, the time per generation increases with higher values of α_2. The total time of the runs also increases but after $\alpha_2 = 0.5$ more and more runs with a very low run time appear. These are runs that stopped early because they found the ground state[8]. That the GA with filter is able to compensate the additional computational burden is supported by Fig. 1b which shows that it performs better for a given amount of time. A comprehensive investigation of the interplay of shorter runtimes due to achieving the goal vs. longer runtimes for higher number of to-be-generated individuals due to, e.g., false positives is beyond the scope of this study, though.

4.7 False Positive Rate

How much time the selection of the next generation takes depends on how often an individual is discarded because it is in the Bloom filter. This in turn could also depend on the false positive rate of the filter. Runs with different Bloom filter sizes and numbers of hash functions were started to assess this effect. Fig. 5 shows the average time per run dependent on the false positive rate at the end of the run given by[9]

$$\varepsilon \approx \left(1 - \left(1 - \frac{1}{m}\right)^{kn}\right)^k, \tag{4}$$

where m and k are the Bloom filter parameters and n is the number of inserted individuals. The run times do not increase below an

[8]This can only happen as we know the ground state in our test application.
[9]Eq. 4 is only a good approximation for sufficiently high m with low k. The true false positive rate is strictly higher [10].

Table 2: The parameter choices with the lowest mean relative error ΔE_{rel} for scans with 12×12 spin glasses and conditioned on α_1 (part a) and α_2 (part b).

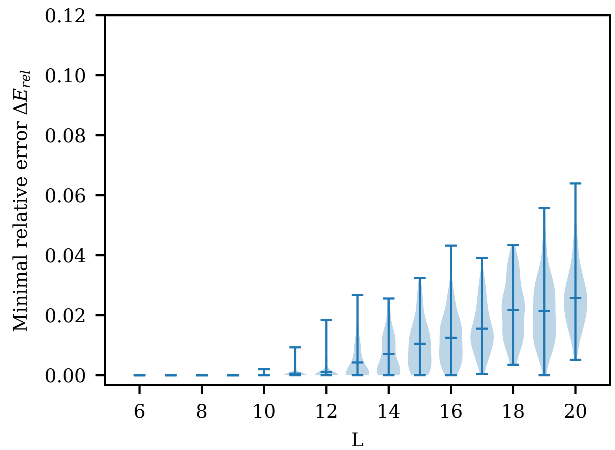
(a) Scan over α_1 (1st column), ΔE_{rel} minimized over α_2, μ, ρ.

				ΔE_{rel}		spins up U		
α_1	α_2	μ	ρ	mean	std	mean	std	P[%]
0.0	0.0	1.0	0.2	0.01991	0.01359	72.899	6.207	5
0.01	0.8	0.6	0.4	0.00444	0.00685	71.484	6.069	48
0.03	1.0	0.7	0.1	0.00297	0.00530	73.077	6.017	56
0.05	0.9	0.7	0.1	0.00221	0.00407	72.803	6.334	65
0.07	1.0	0.7	0.5	0.00238	0.00512	71.353	5.894	64
0.09	1.0	0.6	0.8	0.00225	0.00428	72.205	5.531	67
0.1	1.0	0.5	0.2	0.00190	0.00365	72.937	5.876	67
0.2	0.9	0.6	0.7	0.00188	0.00365	72.071	6.243	71
0.3	1.0	0.6	0.3	0.00156	0.00358	71.147	5.902	75
0.4	0.9	0.6	0.5	0.00130	0.00268	71.266	5.909	74
0.5	0.9	0.4	0.3	0.00153	0.00327	72.709	5.894	71
0.6	1.0	0.3	1.0	0.00143	0.00403	71.888	6.134	76
0.7	1.0	0.3	0.5	0.00147	0.00381	72.977	5.907	75
0.8	1.0	0.4	1.0	0.00113	0.00288	72.645	6.190	77
0.9	1.0	0.3	0.5	0.00147	0.00340	71.811	5.893	74
1.0	1.0	0.3	0.5	0.00116	0.00284	72.347	6.403	80

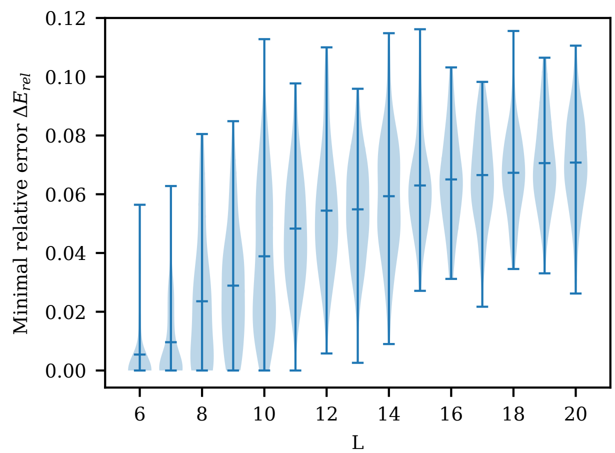
(b) Scan over α_2 (1st column), ΔE_{rel} minimized over α_1, μ, ρ.

				ΔE_{rel}		spins up U		
α_2	α_1	μ	ρ	mean	std	mean	std	P[%]
0.0	0.0	1.0	0.2	0.01991	0.01359	72.899	6.207	5
0.1	0.3	1.0	0.2	0.01916	0.01295	73.307	5.848	6
0.2	0.6	0.9	0.4	0.01847	0.01262	71.813	6.167	3
0.3	0.3	1.0	0.5	0.01700	0.01123	72.103	5.871	5
0.4	0.1	0.9	0.6	0.01521	0.01155	72.161	5.704	14
0.5	0.2	0.9	0.3	0.01151	0.01008	73.191	5.274	15
0.6	0.1	0.8	0.2	0.00790	0.00795	73.035	5.937	30
0.7	0.4	0.8	0.5	0.00363	0.00526	73.370	6.104	50
0.8	0.3	0.7	0.2	0.00225	0.00443	71.915	5.998	67
0.9	0.8	0.5	1.0	0.00114	0.00280	71.664	5.969	81
0.91	1.0	0.5	0.7	0.00137	0.00316	71.098	6.129	70
0.94	0.6	0.3	1.0	0.00135	0.00281	72.262	5.972	70
0.97	0.3	0.6	0.6	0.00134	0.00270	72.753	6.377	72
0.98	1.0	0.4	0.6	0.00110	0.00312	72.385	6.250	81
0.99	0.4	0.5	0.3	0.00109	0.00298	71.492	6.135	81
0.995	0.7	0.5	0.7	0.00118	0.00302	71.662	6.131	79
0.999	1.0	0.4	0.7	0.00111	0.00269	72.181	6.227	77
1.0	0.8	0.4	1.0	0.00113	0.00288	72.645	6.190	77

approximate false positive rate of about 0.7, then it starts to increase dramatically. To incoporate this insight, we have chosen the Bloom parameters k, m (based on the maximal n for a population) in all of the above analysis.



(a) With Bloom filter.



(b) Without Bloom filter.

Figure 3: The minimal relative error of the GA after 250 000 generations on $L \times L$ Ising spin glass instances. Averaged over 100 runs. Parameters that performed well on 12×12 instances were used.

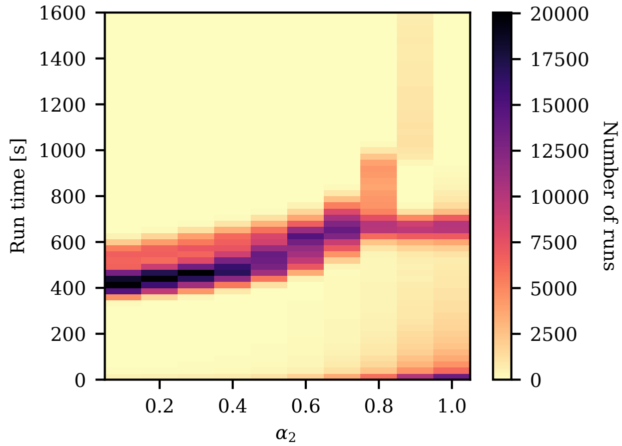
4.8 Evolutionary Dynamics

Coming back to our motivation on increasing diversity and improving the search performance of the GA dynamics of Sec. 2, we show in Fig. 6a the time-averaged selective pressure $I := \frac{1}{T}\sum_{t=1}^{T} I_t$ of Eq. 3. While the sorting order has no particular meaning, this mode of display allows for a direct comparison of the overall selective pressures measured.

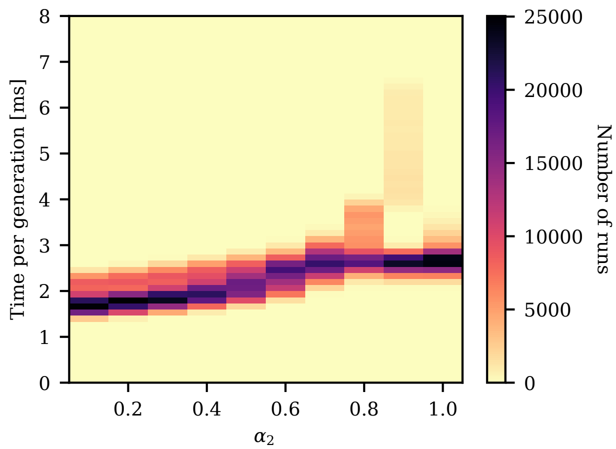
Clearly, an increase in α_1 leads to higher selective pressure in the population. Note, the pronounced difference between the cases $\alpha_1 = 0$ (no Bloom filter) and $\alpha_1 = 0.01$ (only one out of one hundred individuals gets inserted into the Bloom filter, thus, roughly 10 individuals per generation) is already sufficient for the selective pressure to act upon the population.

This pronounced difference is also observable in ΔE_{rel} as a function of time (generations) in Fig. 6b.

In Fig. 7 we show the loss of diversity according to Blickle and Thiele [7] for various parameter choices under varying α_1 and α_2,



(a) Total time.



(b) Average time per generation.

Figure 4: The run time of the GA dependent on α_2 for 12×12 spin glasses.

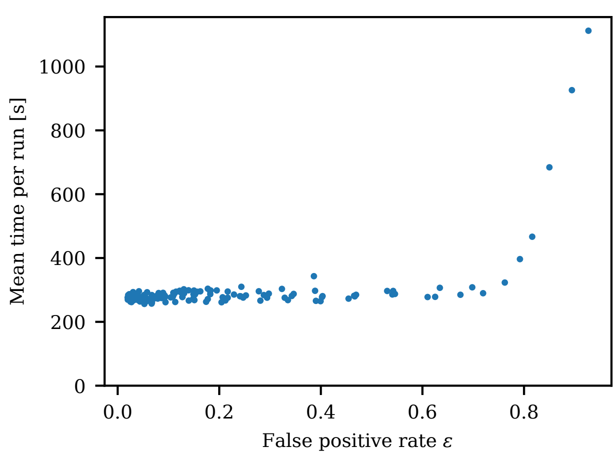
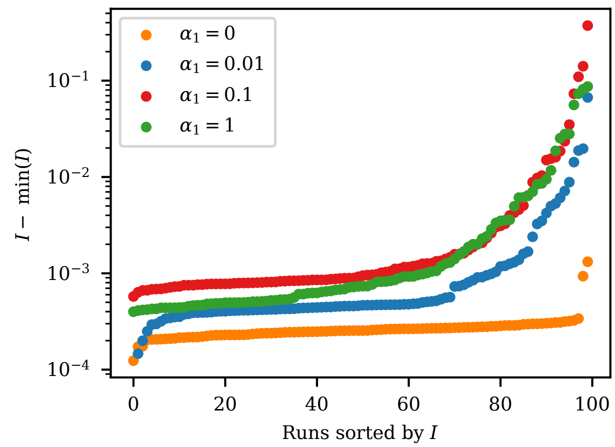
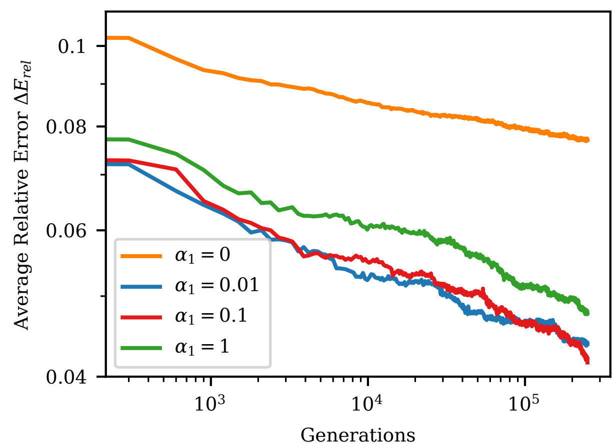


Figure 5: The average run time of the GA dependent on the false positive rate after 250 000 000 individuals have been inserted (12×12 spin glass).



(a) Time-averaged Selective Pressure I of the 100 runs. Values are sorted in ascending order.

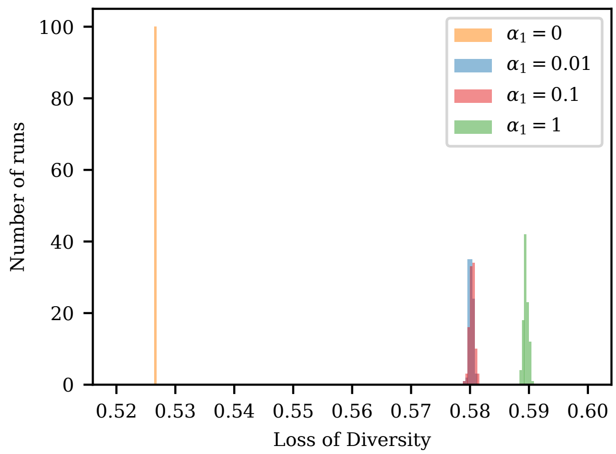


(b) Average relative error ΔE_{rel} of the populations of 100 runs in each generation for the same configurations as in Fig. 6a.

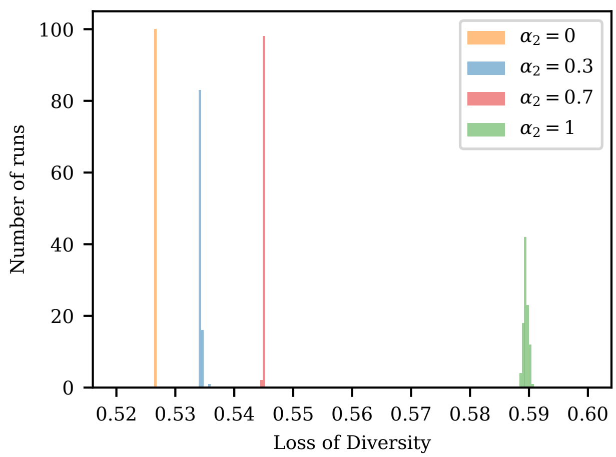
Figure 6: a) Time-averaged selective pressure I with T generations for each of 100 runs and b) average relative error ΔE_{rel} of each generation for an Ising spin glass of size 12×12 and representative α_1 values (with other parameters taken from Tab. 2a).

respectively. We observe a systematic change with the increase of α_1 and α_2.

Taken together, the following picture emerges: the selective pressure grows with increasing α_1; furthermore the loss of diversity increases, too. While this may at first be in contradiction to our initial motivation on increasing diversity, these findings can be reconciled by consulting Tabs. 2a and 2b and Fig. 6b: we see that neither α_1 nor α_2 have any (systematic) effect on the dispersion of the number of up-spins in the elitist configurations at the end of individual runs U, while the populations for different α_1s are more broadly dispersed (at least in ΔE_{rel} values). This detail implies



(a) Different α_1. $\alpha_2 = 1$.



(b) Different α_2. $\alpha_1 = 1$.

Figure 7: Histograms of the loss of diversity over 100 runs with $\mu = 0.5$, $\rho = 0.5$.

that the configurations are still as diverse as in the non-Bloom case. However, we loose *in each selection step* diversity and have *in each selection step* a higher selective pressure. This implies that the creation of new individuals for a new generation must introduce a higher diversity. Then – in the next generation's selection step – the diversity is reduced again based on higher selective pressure and loss of diversity in a more diverse population – induced by the usage of a population-focused Bloom filter.

5 CONCLUSIONS

We implemented an (quasi-)infinite memory into a GA. We have discussed the relationship of this GA variant to the notion of niching and – most importantly – to the biological concept of epi-genetics.

Practically, we rely on Bloom filters that work in constant time and space to avoid pitfalls of other heuristics with finite memory such as tabu search.

We have scanned parameters extensively and analyzed the performance and search dynamics under various measures. The picture that emerges is a nuanced one of the interplay of higher diversity at the creation of a new population intertwined by higher and more restrictive selection.

We find that in a test application (Ising spin glasses) for which exact solutions are independently derived our new GA variant performs better, both in relative error and – most striking – near perfect solution probability in less time and number of generations. We increase this ratio from 5% for a traditional GA to up to 81% for an optimal choice of parameters. Without optimization of parameters such as the mutation and recombination rate, we still improve up to some 60% (cmp. Fig. 2).

To this end, a high filtering parameter α_2 of at least 0.8 was needed for the GA to perform best. The recombination rate ρ and α_1 had only minor effects on the results. This implies that an optimization of those hyper-parameters is not necessarily worth the effort. α_2 itself describes how effectively the previously acquired knowledge is leveraged. The other parameter α_1, however, showed a somewhat less pronounced tendency towards 1. In fact, superior performance is attained as soon as $\alpha_1 \geq 0.01$. The hyperparameter α_1 determines how many individuals per generation are inserted into the Bloom filter.

We furthermore show how loss of diversity and selective pressure are geared towards a more diverse population by the Bloom-filter-based history.

Our new GA variant requires a fixed amount of additional memory for the Bloom filter. A single generation takes longer to be generated. Still the time to reach an improved objective function value is shorter with the filter. Though a Bloom filter can theoretically store (quasi-)infinte elements, one must know beforehand for how many generations the GA will run to chose an appropriate filter size.

In the future, we plan to perform more tests to evaluate whether the good performance transfers to other combinatorial optimization problems – potentially under different representations of the genotpyes.

ACKNOWLEDGMENTS

Computations were conducted on the Lichtenberg high performance computer of the TU Darmstadt, Germany.

REFERENCES

[1] Sneha Antony and J.N. Jayarajan. 2015. T-GEN: A Tabu Search Based Genetic Algorithm for the Automatic Playlist Generation Problem. *Procedia Computer Science* 46 (2015), 409–416. https://doi.org/10.1016/j.procs.2015.02.038 Proc. Int. Conf. Inf. Comm. Tech, ICICT 2014, 3-5 Dec. 2014 at Bolgatty Palace & Island Resort, Kochi, India.

[2] Austin Appleby. 2016. Murmurhash3. (2016). https://github.com/aappleby/smhasher/wiki Accessed: 2021-01-25.

[3] F. Barahona. 1982. On the computational complexity of Ising spin glass models. *J. Phys. A: Mathematical and General* 15, 10 (oct 1982), 3241–3253.

[4] Hans-Georg Beyer and Hans-Paul Schwefel. 2002. Evolution strategies – A comprehensive introduction. *Natural Computing* 1, 1 (01 Mar 2002), 3–52. https://doi.org/10.1023/A:1015059928466

[5] K. Binder and A.P. Young. 1986. Spin glasses : Experimental facts, theoretical concepts, and open questions. *Rev. Mod. Phys.* 58, 4 (1986), 801–976.

[6] David Blackman and Sebastiano Vigna. 2018. Scrambled Linear Pseudorandom Number Generators. *CoRR* abs/1805.01407 (2018). arXiv:1805.01407 http://arxiv.org/abs/1805.01407

[7] Tobias Blickle and Lothar Thiele. 1995. *A comparison of selection schemes used in genetic algorithms.* Technical Report TIK-Report 11. Computer Engineering and Networks Laboratory – Swiss Federal Institute of Technology (ETH), Gloriastrasse, Zürich, CH. 279–284 pages.

[8] Burton H. Bloom. 1970. Space/Time Trade-Offs in Hash Coding with Allowable Errors. *Commun. ACM* 13, 7 (jul 1970), 422–426. https://doi.org/10.1145/362686.

362692

[9] James Blustein and Amal El-Maazawi. 2002. *Bloom filters. A tutorial, analysis, and survey.* Technical Report. Dalhousie University, Halifax, Nova Scotia. 1–31 pages. https://cdn.dal.ca/content/dam/dalhousie/pdf/faculty/computerscience/technical-reports/CS-2002-10.pdf; accessed 04/06/2021.

[10] Prosenjit Bose, Hua Guo, Evangelos Kranakis, Anil Maheshwari, Pat Morin, Jason Morrison, Michiel Smid, and Yihui Tang. 2008. On the false-positive rate of Bloom filters. *Inform. Process. Lett.* 108, 4 (2008), 210 – 213. https://doi.org/10.1016/j.ipl.2008.05.018

[11] Kalyanmoy Deb. 2001. *Multi-objective optimization using evolutionary algorithms.* John Wiley & Sons, Chichester New York.

[12] Cathérine Dupont, D. Randall Armant, and Carol A. Brenner. 2009. Epigenetics: definition, mechanisms and clinical perspective. *Seminars in reproductive medicine* 27, 5 (Sep 2009), 351–357. https://doi.org/10.1055/s-0029-1237423 19711245[pmid].

[13] Fred Glover. 1989. Tabu Search—Part I. *ORSA Journal on Computing* 1, 3 (1989), 190–206. https://doi.org/10.1287/ijoc.1.3.190

[14] Fred Glover, James P. Kelly, and Manuel Laguna. 1995. Genetic algorithms and tabu search: Hybrids for optimization. *Comp. & Op. Res.* 22, 1 (1995), 111–134. https://doi.org/10.1016/0305-0548(93)E0023-M

[15] David E. Goldberg. 1989. *Genetic Algorithms in Search, Optimization, and Machine Learning.* Addison-Wesley, Reading, Massachusetts.

[16] David E. Goldberg, Bradley Korb, and Kalyanmoy Deb. 1989. Messy Genetic Algorithms: Motivation, Analysis, and First Results. *Complex Syst.* 3, 5 (1989).

[17] Natalie Graham, Daniel S. Gruner, Jun Y. Lim, and Rosemary G. Gillespie. 2017. Island ecology and evolution: challenges in the anthropocene. *Environmental Conservation* 44, 4 (2017), 323–335. https://doi.org/10.1017/s0376892917000315

[18] K. Hamacher. 2019. Hybridization of Stochastic Tunneling with (Quasi)-Infinite Time-Horizon Tabu Search. In *Hybrid Metaheuristics*, M. J. Blesa Aguilera, C. Blum, H. Gambini Santos, P. Pinacho-Davidson, and J. Godoy del Campo (Eds.). Springer International Publishing, Cham, 124–135.

[19] K. Hamacher and W. Wenzel. 1999. The Scaling Behaviour of Stochastic Minimization Algorithms in a Perfect Funnel Landscape. *Phys. Rev. E* 59, 1 (1999), 938–941.

[20] John H. Holland. 1992. *Adapation in Natural and Artifical Systems.* MIT Press, Cambridge, Massachusetts.

[21] Setsuya Kurahashi and Takao Terano. 2000. A Genetic Algorithm with Tabu Search for Multimodal and Multiobjective Function Optimization. In *Proceedings of the 2nd Annual Conference on Genetic and Evolutionary Computation (GECCO'00).* Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 291–298.

[22] F. M. Lopes and A. T. R. Pozo. 2001. Genetic algorithm restricted by tabu lists in data mining. In *SCCC 2001. 21st International Conference of the Chilean Computer Science Society.* 178–185. https://doi.org/10.1109/SCCC.2001.972646

[23] Samir W. Mahfoud and Samir W. Mahfoud. 1995. A Comparison of Parallel and Sequential Niching Methods. In *In Proceedings of the Sixth International Conference on Genetic Algorithms.* Morgan Kaufmann, 136–143.

[24] A. H. Mantawy, Y. L. Abdel-Magid, and S. Z. Selim. 1999. Integrating genetic algorithms, tabu search, and simulated annealing for the unit commitment problem. *IEEE Trans. Power Sys.* 14, 3 (1999), 829–836. https://doi.org/10.1109/59.780892

[25] B. E. Mayer and K. Hamacher. 2014. Stochastic Tunneling Transformation during Selection in Genetic Algorithm. In *Proceedings of the 2014 Annual Conference on Genetic and Evolutionary Computation (GECCO '14).* Association for Computing Machinery, New York, NY, USA, 801–806. https://doi.org/10.1145/2576768.2598243

[26] Melanie Mitchell. 1996. *An introduction to genetic algorithms.* MIT Press, Cambridge, Massachusetts.

[27] Michael Mitzenmacher. 2002. Compressed bloom filters. *IEEE/ACM Trans. Netw.* 10, 5 (2002), 604–612.

[28] A. Petrowski. 1996. A clearing procedure as a niching method for genetic algorithms. In *Proc. IEEE Int. Conf. Evol. Comp.* IEEE, 798–803. https://doi.org/10.1109/ICEC.1996.542703

[29] Ofer M. Shir. 2012. Niching in Evolutionary Algorithms. In *Handbook of Natural Computing*, Grzegorz Rozenberg, Thomas Bäck, and Joost N. Kok (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 1035–1069. https://doi.org/10.1007/978-3-540-92910-9_32

[30] C. Simone, M. Diehl, M. Jünger, P. Mutzel, and G. Reinelt. 1995. Exact ground states of Ising spin glasses: New experimental results with a branch-and-cut algorithm. *J. Stat. Phys.* 80 (1995), 487.

[31] W. Wenzel and K. Hamacher. 1999. A Stochastic tunneling approach for global minimization. *Phys. Rev. Lett.* 82, 15 (1999), 3003–3007.