# Diversity Maintenance using a Population of Repelling Random-Mutation Hill Climbers

Rokas Volkovas
University of Essex
Colchester, UK
rv16826@essex.ac.uk

Michael Fairbank
University of Essex
Colchester, UK
m.fairbank@essex.ac.uk

Diego Perez-Liebana
University of Essex
Colchester, UK
dperez@essex.ac.uk

***Abstract*—A novel evolutionary algorithm, which can be viewed as an extension to the simple, yet effective, approach of the Random-Mutation Hill Climber (RMHC), is presented. The algorithm addresses the shortcomings of RMHC and its multi-individual parallel version through the introduction of a penalty term into the fitness function, which penalizes individuals in the population for being too similar, hence maintaining population diversity. The performance of the algorithm is evaluated on the deceptive trap and a set of SAT problems, comparing them to the Crowding EA. The results show that at a small cost of solution speed on simpler problems, the algorithm gains better capabilities of dealing with the issues of local maxima.**

## I. Introduction

Evolutionary Algorithms (EA) are known as some of the most generally applicable problem solvers in many different fields of research [1]. This reputation stems from their problem agnosticism, only requiring the knowledge of the solution structure and the ability to measure how good a specific solution is. With real-life evolutionary processes being the inspiration for the original algorithm concepts, they attempt to mimic how reproductive animals adapt to their environment over a number of generations through heredity and variation.

More specifically, the first incarnation [2] of the idea, from which many other algorithms were built upon, holds a number of possible chosen problem solutions–a population of individuals. Iteratively replacing the individuals with their better performing mutations then, in theory, produces higher and higher scoring individuals, or in other words, problem solutions of constantly increasing quality. In practice, however, this is commonly not the case in deceptive problems. Problems for which one would use a search algorithm tend to have vast amounts of possible solutions, with only some solutions being of noticeably higher quality than others, leading the population into suboptimal locations. This is known as the algorithm getting stuck at a local fitness-maximum of the search space, i.e. a point in the search space from which all minor perturbations decrease its fitness score.

Over the last few decades, many publications addressing the problem of local maxima within evolutionary algorithms have been produced. One of the proposed ideas was to maintain separate groups of individuals, allowing them to occasionally cross-over, which has been proven effective for some problems at the cost of additional parameter design. The contribution of this paper is an algorithm which enforces the separation of groups automatically, by having a population of individuals who prefer to explore areas further away from their brethren. This preference is enforced through the existence of a *repelling* penalty function, which is designed reduce the fitness of individuals that are too similar to each other. The algorithm is general enough to allow tweaking for specific needs, and can continue improving given prior knowledge or more computational power.

**Algorithm 1** RMHC

**Require:** individual $i$
1: **while** end condition not met **do**
2: $mutant \leftarrow Mutate(i)$
3: **if** $Evaluate(i) \leq Evaluate(mutant)$ **then**
4: $i \leftarrow mutant$
5: **end if**
6: **end while**

## II. Related Work

The Random-Mutation Hill Climber [3] (RMHC) is one of the simplest and oldest evolutionary algorithms. Pseudo-code for RMHC is given in Algorithm 1. The algorithm works by iteratively altering (mutating) a single solution slightly, and whenever the alteration yields an increase in fitness, the mutation is kept; otherwise the mutation is discarded. As usual in EAs, the end condition is either finding the optimal fitness value or, more likely, reaching the evaluation limit. An extension of RMHC is *Parallel RMHC* - population of RMHC behavior individuals (similar to parallel hill-climbing in [4]), which is the same as RMHC, but instead of improving just one individual, it maintains a population of individuals. The concept should not be confused with the application of *penalty functions* in constraint optimization problems, such as [5], where a penalty is applied for individuals entering the non-feasible region of a constraint optimization problem. The proposed algorithm is referred to as the Repelling RMHC for the remainder of this paper.

The problem of ensuring population diversity has been previously approached in a number of ways. They include niching [4]. which divides the population of individuals into groups and allows reproduction only or primarily within those groups. Another alternative proposed is tabu search [6], which

**Algorithm 2** Crowding

**Require:** Evaluation number $en$
**Require:** Tournament size $ts$
1: $genomes \leftarrow gn$ random individuals
2: **for** 1 to $en/2$ **do**
3: $parentA \leftarrow TournamentSelect(ts)$
4: $parentB \leftarrow TournamentSelect(ts)$
5: $childA, childB \leftarrow CrossOver(parentA, parentB)$
6: $childA \leftarrow Mutate(childA)$
7: $childB \leftarrow Mutate(childB)$
8: $Evaluate(childA)$
9: $Evaluate(childB)$
10: $CrowdingSelect(childA)$
11: $CrowdingSelect(childB)$
12: **end for**

**Algorithm 3** CrowdingSelect(child)

**Require:** Population $genomes$
**Require:** Crowd Size $cs$
**Require:** Large number $min$
1: **for** 1 to $cs$ **do**
2: $j \leftarrow Rand(|genomes|)$
3: $distance \leftarrow Distance(child, genomes[j])$
4: **if** $distance < min$ **then**
5: $min \leftarrow distance$
6: $replacedIndex \leftarrow j$
7: **end if**
8: **end for**
9: $genomes[replacedIndex] \leftarrow child$

prohibits individuals from mutating into previously explored configurations, thus forcing them to explore unseen areas of the fitness landscape. Crowding [7] maintains diversity using an individual replacement strategy, which favors the offspring who are the furthest from the current population. These techniques and many others [8] suffer from the drawbacks of requiring large amounts of memory, significant foresight from the designer or the reliance on the population initialization.

To gain insight into the quality of the introduced algorithm it needs to be compared against a known existing approach attempting to solve the same problem, which in this case is finding the optimal fitness via population diversity maintenance. The chosen algorithm to be compared against the Repelling RMHC in the select problems is the established Crowding algorithm [7], focused on ensuring the population is diverse through a steady state, tournament selection based approach. Generic cross-over procedure is used to create new individuals, which then replace other population members, which are the most similar, measured using a defined distance function. The version of the algorithm parts used is shown in Algorithms 2 and 3, which are reproduced from [9]. The $TournamentSelect(N)$ function returns the highest fitness individual from a subpopulation of $N$ individuals. Along with the distance function, the algorithm requires the specification of the crowd and tournament sizes.

A conceptual breakthrough in the EA field was Novelty Search [10], which strayed away from measuring the fitness of an individual directly to how it performs a certain task, but instead looking at how differently does it do it in comparison to the others within the population. Over time, it was established that its lack of generality, due to the need for a designer-crafted Quality Diversity function, hindered the concept as the selection of this function could be a problem no less difficult than the original [11][12]. Unsurprisingly, novelty search uses some type of diversity maintenance EA to ensure the difference between the individuals, such as the improved niching algorithm of MAP-Elites [13]. Thus, improvements of the diversity-maintenance methods at a lower level will lead to higher performance on more complex problems.

## III. Approach

The algorithm introduced in this paper is called *Repelling RMHC*. It can be seen as an extension to the Parallel version of RMHC; extended to include *repelling functions*, which are designed to push individuals away from each other. Similarities can be drawn to few of the many existing niching methods [4].

In short, if the fitness function of an individual $i$ in the RMHC algorithm can be defined in (1), then (2) shows the individual fitness in Repelling RMHC, where the individual subscript is the evaluation sequence number of the individual in the population, and $Eval$, $Fit$ and $Dist$ are short for $Evaluate$, $Fitness$ and $Distance$, respectively. The evaluation, as usual, depends on the specific application and the distance measure between two individuals has to be chosen prior to algorithm exploitation.

$$Fit(i_N) = Eval(i_N) \tag{1}$$

$$Fit(i_N) = Eval(i_N) - \sum_{n=1}^{N-1} Penalty(Dist(i_N, i_n), Fit(i_n)) \tag{2}$$

### A. Concept

The Repelling RMHC concept is inspired by the human rewarding behavior when applied to the appreciation of novelty, which is most obvious in the art world. Taking music as an example, once an artist creates a novel type of music arrangement, which is appreciated by many, all of the credit tends to go to that single artist and imitators get far less recognition, not to mention that plagiarism only generates infamy, i.e. negative recognition. More generally, the process can be visualized as a local peak of a fitness landscape of music, which the artist has reached and is now rewarded with recognition–the fitness equivalent. New individuals reaching the same hill are no longer rewarded due to the peak already being occupied, naturally encouraging exploring areas away from the ones already taken.

Translating this type of landscape navigation to an evolutionary algorithm requires the individuals to have the ability to find peaks as well as to feel the pressure to stray away from

ones already found by others. The peak seeking is covered by the default RMHC and the curiosity half is done by having each individual of the RMHC algorithm exhibiting individuals emit a predefined repelling function, modifying the fitness of all other individuals - applying a penalty for being nearby.

### B. Repelling RMHC

Fig. 1 visualizes the concept of repulsion as applied to an example optimization problem. The goal of this problem is to find a 10-bit string configuration with the highest fitness. The true fitness function curve is shown in light blue. It is a function dependent on the number of *active* bits, i.e. bits whose value is set to 1. The function has two fitness peaks at 2 and 7. Therefore a string with 7 active bits, regardless of their order, will have the optimal fitness.

In this example, a population of just two individuals is considered. Their genomes are initialized as uniformly random 10-bit strings. Assume the first randomly generated individual has 6 active bits, and corresponding fitness 1.25 (as indicated by the light blue continuous curve). By design, the individual will then *repel* the subsequently evaluated individuals away from its exact genome configuration. That is, upon evaluation of the second individual, its fitness value will be reduced from the true value by the repelling force of the first individual. The repelling force, in this case, is represented by a (Gaussian) penalty function, shown as a dashed orange line with its peak at the genome location of the first individual.

Concretely, if the second individual happened to also have 6 active bits, it would receive the fitness of 0 due to the superimposed reduction. The dotted black curve shows the updated fitness landscape of the second individual, which is equal to the orange dashed curve subtracted from the continuous light blue line. Note that the updated landscape now has its global optima at a different location, which is the core feature of the algorithm. The reductions for the following individuals would stack recursively. It is also important to notice, that the distance measure in this example was chosen to correspond to the fitness function, which in useful cases tends to be unknown, specifically to aid the explanation.

The way the superimposed penalty function is used in the algorithm can be likened to the approach used in [14], where the novelty of an individual and the fitness score are given weights, which are then adjusted to find the ideal ratio for the specific problem. Although related, they are not alternatives to each other and could possibly even be combined, as the weights in this case could control the strength of the penalty function.

The convenient effect of Repelling RMHC is that the impact of the algorithm can be readily scaled as desired, with one extreme finding the closest local peak and the other - spreading out across the fitness landscape as if it were flat. Intuitively, if the fitness landscape were to be completely flat, with any and all genomes receiving the same fitness value, the algorithm would converge to a uniformly distributed static population with equal distances between each other, each maximizing the penalized fitness functions. The statement holds only under the assumption that the equal distance spread is achievable taking into account the population density and landscape granularity.

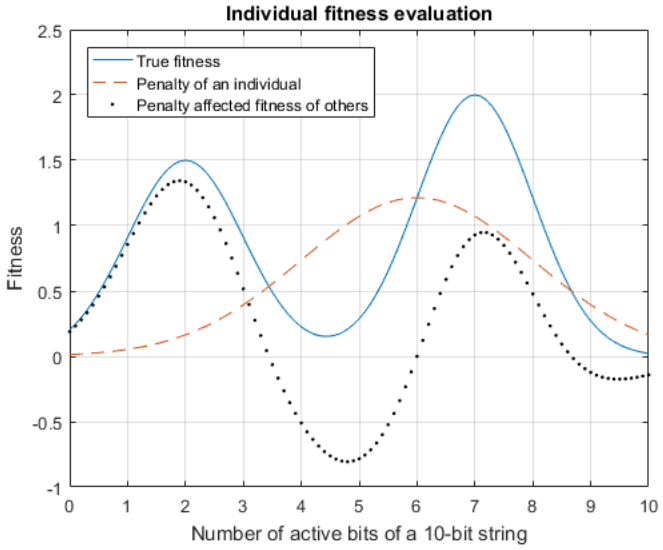


Figure 1: Example fitness function and associated penalty for an example fitness function and population of two individuals. The black curve (dotted) is the orange curve (dashed) subtracted from the blue curve (solid).

### C. Repelling function

Depending on the fitness landscape of a select problem, some choices of the repelling function will consistently produce higher quality results than others. It is possible that in some cases, due to this choice being poor, the optimal solution might appear as suboptimal due to the landscape morphing. On the other hand, the goal of the algorithm is not to find the optimal solution, but rather maintain a population of different, yet still fit individuals. Regardless of the chosen function and any of the other parameters, if the fitness landscape turns out to be mostly flat with sparse bumps, the repelling nature will have a higher chance of finding those bumps when compared to other local search methods due to its uniform spread property.

It is also important to note that in the cases of decaying repelling functions, such as the Gaussian function, care should be made with negative fitness values - pushing others away from negative fitness might cause attracting them instead, which is the reason for the experiments using functions with the minimum fitness being 0.

### D. Distance measure

Similarly as with the repulsive function type, the distance measure between two individuals can alter the effectiveness of the algorithm drastically. It can be argued that, much like the choice of Quality Diversity function in Novelty Search variations, the search for the optimal distance measurement type is possibly a problem of no less difficulty as the one being solved. However, a case can also be made for the argument that the distance measure is only polish of the core behavior. Thus,

**Algorithm 4** Repelling RMHC

**Require:** Number of evaluations $gi$
**Require:** Population size $gn$
$genomes \leftarrow gn$ random individuals
**for** 1 to $gi$ **do**
  $previous \leftarrow$ empty list of genomes
  **for all** $g$ in $genomes$ **do**
    $mutant \leftarrow Mutate(g)$
    $fitness \leftarrow Evaluate(mutant)$
    **for all** $p$ in $previous$ **do**
      $dist \leftarrow Distance(g, p)$
      $penalty \leftarrow Penalty(dist, Fitness(p))$
      $fitness \leftarrow fitness - penalty$
    **end for**
    **if** $Fitness(g) \leq fitness$ **then**
      $g \leftarrow mutant$
    **end if**
    $previous \leftarrow previous \cup g$
  **end for**
**end for**

the Euclidean distance between genomes, as opposed to the behavior of the phenotypes, is a reasonable choice, irrespective of the problem at hand.

### *E. Implementation*

Algorithm 4 shows the pseudo-code implementation of the Repelling RMHC. It requires the values of number of individuals (genomes) to be used, number of generations, the single bit mutation rate, and the definition of the $Penalty(distance)$ and $Distance(genomeA, genomeB)$ functions. The algorithm evaluates the objective fitness of each individual and then reduces this fitness by a penalty based on the distance from this individual to each one of the previously evaluated in that same generation.

An important deviation from the traditional EAs - which introduce new individuals into the population through replacing and recombination of the existing ones (a feature of the Crowding algorithm) - is that in the proposed approach the individuals are never removed from the population. The focus is shifted away from collecting the good parts of existing individuals to steering them into fit locations themselves.

## IV. Experimental Setup

For all experiments the single bit mutation chance was equal to $\frac{1}{genomeSize}$, the distance function was the Euclidean distance, defined by (3), where $a_n$ is the $n$'th bit value of individual $a$. The repulsion function was the Gaussian function, shown in (4) with $\sigma = genomeSize$ and $\mu = 0$, exponentially reducing the penalty as the distance between individuals got larger. Specifically, the penalty for each individual used is defined by (5); the $Penalty$ value is used to reduce the true fitness returned by $Evaluate$. When using the Crowding algorithm, the tournament size as well as crowding size were set to 2. The value was experimentally found to produce the best results when using the relatively small population sizes. The individual genome values were generated using random values from a uniform distribution. To evaluate the performance of the Repelling RMHC, two distinct testbeds were used.

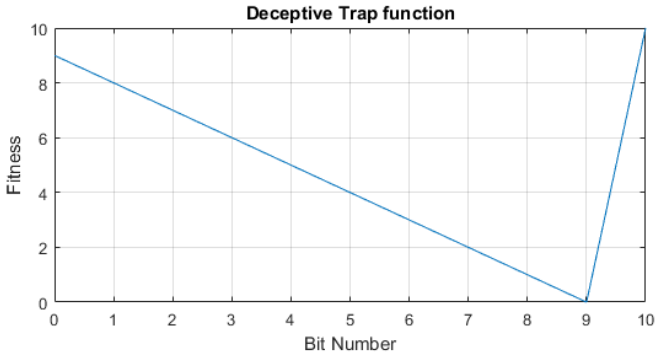


Figure 2: The deceptive trap function for a 10-bit string

$$Distance_{Euclidean}(a, b) = \sqrt{\sum_{n=0}^{N-1} (a_n - b_n)^2} \quad (3)$$

$$Gaussian(d) = \exp \frac{-(d - \mu)^2}{\sigma^2} \quad (4)$$

$$Penalty(i_N) = \sum_{n=1}^{N-1} Fit(i_n) \times Gaussian(Dist(i_N, i_n)) \quad (5)$$

### *A. Deceptive trap function*

One is a deceptive trap function, from a family of functions commonly used to showcase the vulnerabilities of EAs [15]. It has the property of positioning the optimal fitness value a single mutation away from the worst value, misleading the conventional EAs to the worse overall solution. The bit string version of the problem is presented in Fig. 2. For this problem the genome size (number of bits) was set to 10, population sizes of $1, 3, 4, 5, 8$ and the performance was averaged over a 100 runs of 1000 evaluations.

### *B. SAT*

The second comparison problem was chosen to be the common NP-hard clause satisfiability problem (SAT), the state of the art in which is usually lead by stochastic local search algorithm with diversity maintenance [16]. The problem requires to find a suitable string of bits, which have to satisfy as many conjunctive clauses of disjunct bits. The set of problems used was the entire *Random-3-SAT Instances and Backbone-maximal Sub-instance* set from SATLIB benchmark problem library[1], which contains 500 instances of problems with 429 clauses of 100 variable 3-SAT problems, all fully satisfiable. The fitness function is equal to the number of clauses satisfied by a potential solution, 429 being the optimal. For evaluation

[1]http://www.cs.ubc.ca/ hoos/SATLIB/benchm.html

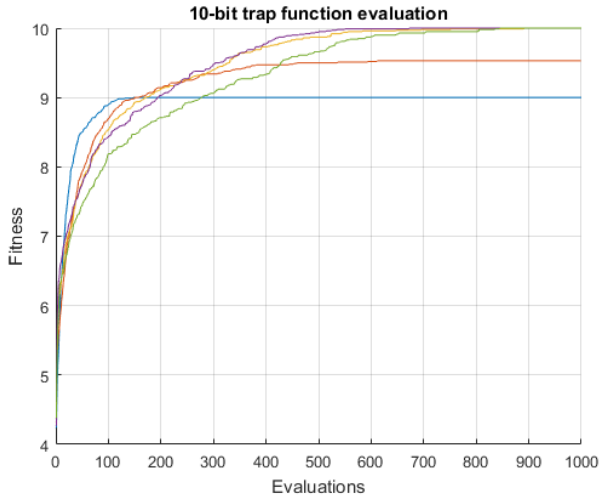


Figure 3: Algorithm comparison on the deceptive trap function. Legend numbers indicate the population size used. RMHC = REP1

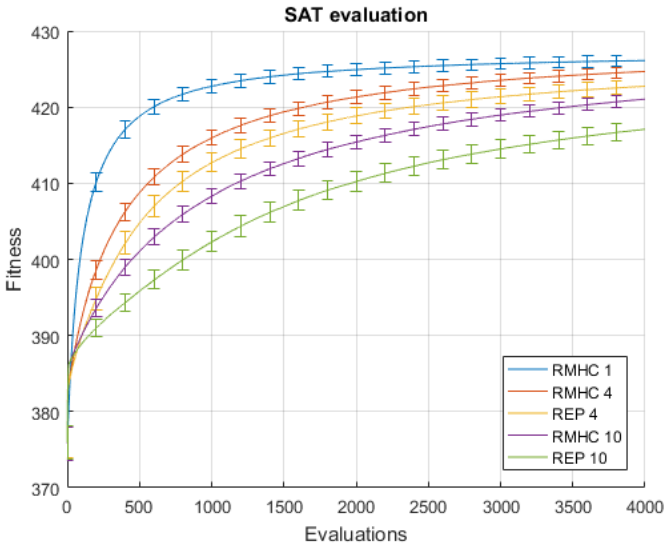


Figure 4: Parallel RMHC and the Repelling RMHC algorithm comparison on the SAT problems. Increasing population size reduces improvement speed.

each one of the configurations were run 10 times on each problem for an average performance on each evaluation within the evaluation budget of 4000. The configurations consisted of population sizes of 1, 4 and 10.

## V. Results and Discussion

Fig. 3 shows the performance comparison of the RMHC and Repelling RMHC algorithms applied to the 10-bit version of the deceptive trap function. RMHC here is not capable of finding the optimal value of 10, whereas the averaged Repelling RMHC curve keeps climbing to the optimal value and finds it after at most 700 evaluations in the 4 individual case, more in others. Whereas, Fig. 4 presents the comparison being made across the suite of SAT problems using different population sizes. Note that since the number of evaluations is kept constant, the larger population sizes imply proportionally fewer individual iterations for its members. The legend indicates the curve algorithm and the individual number used. Fig. 5 shows the Crowding algorithm performance under the same conditions. Note the different y axis scales.

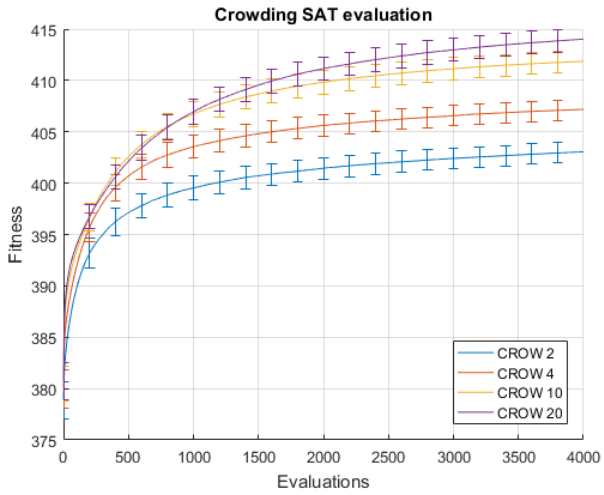


Figure 5: Crowding algorithm performance on the SAT problem set using different population sizes. Does not reach fitness levels above 415 (429 being optimal).

### A. Trap function

The Repelling RHMC is shown to be capable of overcoming the local maxima in the deceptive trap function as indicated by Fig. 3 with any number of individuals larger than 2. Using 3 individuals (the REP3 curve), the algorithm finds the optimal solution 50% of the time after 400 evaluations, 100% when using 4 and given 700 evaluations. This data shows that a population of repelling RMHCs outperforms a population of non-interactive where on deceptive trap-like function.

From the statistical point of view, there are $2^{bits}$ combinations that can be evaluated - 1024 in the case of 10 bits. That is randomly guessing the solution has a $\frac{1}{1024}$ chance of getting the optimal solution and if RMHC does not get it on initialization, it is doomed to converge to the suboptimal value (disregarding the even smaller chance of mutating into the desired configuration). The Repelling RMHC is not repelled by the low probability, because as soon as one individual starts moving towards the suboptimal value, the others attempt to move away, eventually finding the optimal value in less than 1024 evaluations in the experiment cases.

In none of the experimental cases (not shown), the Crowding algorithm was capable of converging to a higher fitness value than the suboptimal 9. This behavior can be readily attributed to the reliance on crossover. Crossover is known to be beneficial only when the individuals carry parts of a more desired solution in their genomes [17]. The required modularity of active bits is absent in the second best solution.

### B. SAT

Considering the Repelling RMHC, the performance benefit in the trap function is paid for using the performance drop seen in the evaluation of the SAT set in Fig. 4. In this figure, comparing the fitness curves of only the RMHC configurations, it can be seen that giving a single agent more evaluations is more beneficial than polishing multiple ones at the same time - a feature of a problem without a more noisy landscape. Specifically, RMHC1 outperforms RMHC4, which in turn outperforms RMHC10.

Once repulsion is added, the fitness drops further, as indicated by the REPN curves. This is likely due to the good solutions being close to each other, which could potentially be overcome by either increasing the population size or altering the repulsion function parameter, indicating the drawbacks of the algorithm when applied more generally. Also, the optimal solution of $429$ (known to exist), on average, is never found within the given evaluation budget, but the search space now consists of $2^{429}$ combinations. It is interesting to note that even with the fitness drop, the worst scoring configuration of $10$ individuals in the Repelling RMHC case, on average outperformed the best case of the Crowding algorithm, using population size of $20$, seen in Fig. 5.

### C. Limitations

It is important to note that because of the population spread over the entire landscape, adding additional individuals is guaranteed to increase the chance of finding the optimal solution, which would not be the case when stuck at local maxima. However, even though the results show that the algorithm can be useful when applied to certain types of problems, such as the deceptive trap function, it is important to recognize the properties of the approach that enabled it to find the optimal value despite the local maxima issue and how they can both be beneficial and detrimental in some scenarios. Specifically, the effectiveness of the algorithm depends upon the choice of the newly introduced parameters, controlling just how repelling do the individuals find their neighbors.

## VI. Conclusion and future work

A novel approach to diversity maintenance using repelling functions between individuals is presented. The algorithm attempts to push all of the individuals away from each other, with the hopes of them ending up the fitness landscape peaks. It is shown to be effective on the deceptive trap function with a severe local maxima problem, but also exhibits some performance drops on a specific set of SAT problems. The benefits of the approach causing the population to spread out across the fitness landscape, thus potentially providing alternative solutions to the same problem, are identified along with the drawbacks of extra parameters required for operation.

The paper however does not address the value of changing the newly exposed parameters to explore how the behavior changes for the presented problems or introducing a generational evolutionary system, which can be looked at in the future. At the same time, each individual is exploring a different landscape, however in this incarnation that information is not being used - its usefulness is unknown. Additionally, it should be useful to apply the algorithm to more specific problems, potentially within the field of artificial creativity, see how the solutions it finds differ from each other in a way easier for humans to grasp. For example, applying the strategy to a dynamic structure neural network producing sprites to visualize the difference between the different solutions found by this algorithm. In conclusion, the paper serves more as an introduction of the algorithm itself, showing its concept and the most useful features and potential applications when a fit and diverse population is required, rather than a thorough comparison of its performance and diversity maintenance capabilities against more of the other existing algorithms, which without a doubt would shed more light on its behavior.

## Acknowledgment

This work was funded by the EPSRC CDT in Intelligent Games and Game Intelligence (IGGI) EP/L015846/1.

## References

[1] Thomas Back. *Evolutionary algorithms in theory and practice: evolution strategies, evolutionary programming, genetic algorithms*. Oxford university press, 1996.

[2] Nils Aall Barricelli. Symbiogenetic evolution processes realized by artificial methods. *Methodos*, 9(35-36):143–182, 1957.

[3] Stuart Russell and Peter Norvig. Artificial intelligence: a modern approach. 1995.

[4] Samir W Mahfoud. Niching methods for genetic algorithms. *Urbana*, 51(95001):62–94, 1995.

[5] William Crossley, Edwin Williams, William Crossley, and Edwin Williams. A study of adaptive penalty functions for constrained genetic algorithm-based optimization. In *35th Aerospace Sciences Meeting and Exhibit*, page 83, 1997.

[6] Fred Glover and Manuel Laguna. *Tabu Search∗*. Springer, 2013.

[7] K. A. de Jong. *An Analysis of the Behavior of a Class of Genetic Adaptive Systems*. PhD thesis, Department of Computer and Communication Science, University of Michigan, 1975.

[8] Deepti Gupta and Shabina Ghafir. An overview of methods maintaining diversity in genetic algorithms. *International journal of emerging technology and advanced engineering*, 2(5):56–60, 2012.

[9] Faustino J Gomez. Sustaining diversity using behavioral information distance. In *Proceedings of the 11th Annual conference on Genetic and evolutionary computation*, pages 113–120. ACM, 2009.

[10] Joel Lehman and Kenneth O Stanley. Exploiting open-endedness to solve problems through the search for novelty. In *ALIFE*, pages 329–336, 2008.

[11] Steijn Kistemaker and Shimon Whiteson. Critical factors in the performance of novelty search. In *Proceedings of the 13th annual conference on Genetic and evolutionary computation*, pages 965–972. ACM, 2011.

[12] Elliot Meyerson, Joel Lehman, and Risto Miikkulainen. Learning behavior characterizations for novelty search. In *Proceedings of the 2016 on Genetic and Evolutionary Computation Conference*, pages 149–156. ACM, 2016.

[13] Jean-Baptiste Mouret and Jeff Clune. Illuminating search spaces by mapping elites. *arXiv preprint arXiv:1504.04909*, 2015.

[14] Giuseppe Cuccu and Faustino Gomez. When novelty is not enough. *Applications of evolutionary computation*, pages 234–243, 2011.

[15] Kalyanmoy Deb, Jeffrey Horn, and David E Goldberg. Multimodal deceptive functions. *Complex Systems*, 7(2):131–154, 1993.

[16] Dave AD Tompkins and Holger H Hoos. Ubcsat: An implementation and experimentation environment for sls algorithms for sat and max-sat. In *International conference on theory and applications of satisfiability testing*, pages 306–320. Springer, 2004.

[17] Richard A Watson and Jordan B Pollack. A computational model of symbiotic composition in evolutionary transitions. *Biosystems*, 69(2):187–209, 2003.