Bandit-Based Random Mutation Hill-Climbing

Jialin Liu, Diego Pérez-Liébana and Simon M. Lucas
University of Essex, Colchester, UK
Email: {jialin.liu, dperez, sml}@essex.ac.uk

***Abstract*—The Random Mutation Hill-Climbing algorithm is a direct search technique mostly used in discrete domains. It repeats the process of randomly selecting a neighbour of a best-so-far solution and accepts the neighbour if it is better than or equal to it. In this work, we propose to use a novel method to select the neighbour solution using a set of independent multi-armed bandit-style selection units which results in a bandit-based Random Mutation Hill-Climbing algorithm. The new algorithm significantly outperforms Random Mutation Hill-Climbing in both OneMax (in noise-free and noisy cases) and Royal Road problems (in the noise-free case). The algorithm shows particular promise for discrete optimisation problems where each fitness evaluation is expensive.**

***Index Terms*—RMHC, bandit, OneMax, Royal Road**

I. Introduction

Evolutionary Algorithms (EA) have achieved widespread use since their developments in the 1950s and 1960s [1], [2], [3], [4], [5], [6], [7].

Their essence is relatively simple: to generate an initial set of candidate solutions at random, and then to iteratively improve the candidate set via a process of variation, evaluation and selection. They have been the subject of much analysis, development and a diverse range of applications. They have also spawned related approaches (i.e. methods which can be characterised by the outline description above) such as particle swarm optimisation, and have been extended for application to multi-objective optimisation.

This paper introduces a significant variation: the Bandit-Based Evolutionary Algorithm. Bandit algorithms [8], [9] have become popular for optimising either simple regret (the best final decision after a number of exploratory trials) or cumulative regret (best sum of rewards over a number of trials) in A/B testing.

A popular bandit algorithm is the Upper Confidence Bound (UCB) algorithm [10], [11] which balances the trade-off between exploration and exploitation. The UCB-style algorithms have achieved widespread use within Monte Carlo Tree Search (MCTS) [12], called UCT when applied to trees, the "T" being for Trees.

A wide literature exists on bandits [10], [11], [13], [14], [15], [16], [17] and many tools have been proposed for distributing the computational power over the stochastic arms to be tested. There are also some adaptations to other contexts: time varying as in [18]; adversarial [19], [20]; or involving the non-stationary nature of bandit problems in optimization portfolios. St-Pierre and Liu [21] applied the Differential Evolution algorithm [22] to some non-stationary bandit problem, which outperformed the classical bandit algorithm on the selection over a portfolio of solvers.

Browne et al. [12] noted the great potential for hybridising MCTS with other approaches to optimisation and learning, and in this paper we provide a hybridisation of an evolutionary algorithm with a bandit algorithm.

There are examples of using evolution to tune MCTS parameters [23], [24], [25]. Albeit robust, this application of EA is not widespread, due to the computational cost involved in performing fitness evaluations. It should be noted that Lucas et al. [25] made fitness evaluations after each rollout, so they could be rapidly optimised, albeit noisily.

The algorithm reported in this paper is a very different hybrid: it uses bandits to represent the state of the evolving system. This has some similarities with Estimation of Distribution Algorithms (EDAs) [26] but the details are significantly different.

To our knowledge, this is one of the very few times that this type of hybridisation has been attempted; the only other paper we are aware of in the same vein is Zhang et al. [27]. Zhang et al. used a bandit algorithm as a form of Adaptive Operator Selection: the variation operators used within the evolutionary algorithm were selected using a bandit-based approach, showing promising results.

In this paper we develop a bandit-based version of the Random Mutation Hill-Climbing (RMHC) algorithm, and compare the two methods, i.e., the original and the bandit-based algorithms.

To put this in some context, it should be noted that while the RMHC algorithm is very simple, it is often surprisingly competitive with more complex algorithms, especially when deployed with random restarts.

For instance, Lucas and Reynolds evolved Deterministic Finite Automata (DFA) [28], [29], using a multi-start RMHC algorithm with very competitive results, outperforming more complex evolutionary algorithms, and for some classes of problems also outperforming the state of the art Evidence-Driven State Merging (EDSM) algorithms.

Although Goldberg [30] used bandit models, they were used to help understand the operation of a Simple Genetic Algorithm. Our approach is different: we use them as the very basis of the algorithm. The bandit model provides a natural way to balance exploitation (sticking with what appears to be good) versus exploration (trying things which have not been sampled much).

In this paper we model the genome as an array of bandits. In the case where the genome is a binary string of length n,

978-1-5090-4601-0/17/$31.00

we model the state of the evolutionary system as an array of 2-armed bandits. If each element of the string can take on m possible values, then each element is represented by an m-armed bandit.

The main contribution in this work is this new bandit-based RMHC algorithm, together with results on some standard benchmark problems. The current objective is to compare the performance of these algorithms, as well as to analyze the method proposed, in test functions that have been extensively used in the past, in order to be able to draw comparisons with other techniques. Future work will include testing in more complicated test functions (such as the noisy Royal Road problem) and other scenarios (like evolution of game parameters). The new algorithm significantly outperforms the standard RMHC in both OneMax (in noise-free and noisy cases) and Royal Road problems in the noise-free case.

II. Test Problems

In this work, we consider two benchmark optimisation problems in a binary search space.

A. OneMax Problem

The OneMax problem [31] is a simple linear problem aiming at maximising the number of 1 of a binary string, i.e., for a given n-bit string $\mathbf{s}$

$$f(\mathbf{s}) = \sum_{i=1}^{n} s_i, \tag{1}$$

where s_i, denoting the i^{th} bit in the string $\mathbf{s}$, is either 1 or 0.

The complexity of OneMax problem is $O(n\log(n))$ for a n-bit string [32]. Doerr et al. proved that the black-box complexity with memory restriction one is at most $2n$ [33]. More lower and upper bounds of the complexity of OneMax in the different models are analysed [34], [35] and then summarised in Table 1 of [35]. In their elitist model, only the best-so-far solution can be kept in the memory. Our bandit-based RMHC stores the best-so-far solution in the noise-free environment and stores additionally its evaluation number in the noisy environment (detailed in Section III-C).

B. Noisy OneMax Problem

We modify the OneMax problem by introducing an additive noise with constant variance 1:

$$f'(\mathbf{s}) = f(\mathbf{s}) + \mathcal{N}(0,1), \tag{2}$$

$\mathcal{N}$ denotes a Gaussian noise. Thus, the noise standard deviation is of same order as the differences between fitness values. It is notable that our noise model is very different from the one in [36], which used (1+1)-EA and a one-bit noise. But it is the same as used in [37].

The influence of the noise strength on the runtime of (1+1)-EA for the OneMax problem corrupted by one-bit noise is firstly analysed by Droste [38]. In [38] and [36], the misranking occurs due to the change of exactly one uniformly chosen bit of $\mathbf{s}$ by noise with probability $p \in (0,1)$, where p is the noise strength. Thus, the noise acts before fitness evaluation and the evaluated individual (solution or search point) is possibly not the correct one. The individual is infected by noise in their model while in our model, the fitness function is infected by noise.

C. Royal Road Function

The Royal Road functions are firstly introduced by Mitchell et al. [39]. The function fitness gains only if all the bits in one block are flipped to 1[1]. The objective was to design some hierarchical fitness landscapes and studying the performance of Genetic Algorithms (GA). Surprisingly, a simple Random Mutation Hill-Climbing Algorithm outperforms GA on a simple Royal Road function, namely R_1 in [39], [40]. R_1 consists of a list of block-composite bit strings as shown in Fig. 1, in which '*' denotes a either 0 or 1. The fitness $R_1(\mathbf{x})$ is recalled as follows:

$$R_1 = \sum_{i=1}^{n} c_i \delta_i(\mathbf{x}), \tag{3}$$

where c_i is the order of $\mathbf{s}_i$ and $\delta_i = 1$ if $\mathbf{x} \in \mathbf{s}_i$, otherwise, $\delta_i = 0$.

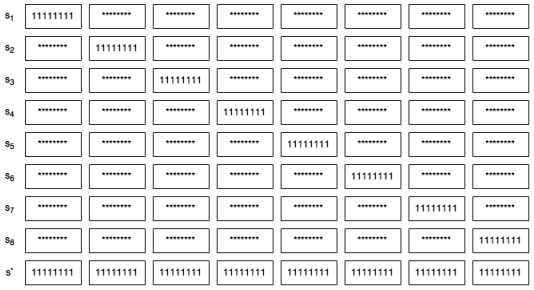


Fig. 1: Royal Road function R_1.

It's notable that, due to the landscapes in the Royal Road function and 1-bit mutation per generation, introducing noise to the fitness is not trivial. When introducing a noise with constant variance, with high probability, the mutated genome has an identical noise-free fitness value to the one of its ancestor. As a result, only the samples of introduced noise are compared.

III. Bandit-based RMHC

In contrast to the standard bandit terminology, where an arm is pulled to gain some reward, the purpose of our bandits is to select the element to mutate at each iteration of the algorithm.

We create an m-armed bandit for each gene of the genome that can take on m possible values. Each bandit works by recording how many times each arm has been pulled, i.e., the number of evaluations of each arm, and the difference in empirical reward between the previous fitness of the genome and the fitness obtained as a result of the selected mutation.

Note that instead of pulling an arm to gain some reward as in the normal bandit terminology, each bandit stores a state and has m arms where each arm $i \in \{1,\ldots,m\}$ stores

[1]Note that OneMax can be considered as a special case of the Royal Road function with a block size of 1.

the statistics of a transition: $T_i \in Transitions(S)$ with $|Transitions(S)| = m$. $Transitions(S)$ denotes the set of transitions at state S. Thus, for a genome of n genes, n multi-armed bandits are created, assuming they are independent.

Each bandit can have a different number of arms, depending on the problem and transition sets. In a n-dimensional OneMax problem or Royal Road function R_1, n 2-armed bandits are required.

For the rest of this paper we assume that $m = 2$ (i.e. we are dealing with binary strings). The extension to larger alphabets should be straightforward, as flipping bits would be replaced by pulling arms from each bandit.

For any position (gene) at a given state S, there is one single possible action *flip* and two transitions, the next state will be

$$S' = \begin{cases} 1, & \text{if it's 0 at state } S \\ 0, & \text{otherwise.} \end{cases}$$

A. Urgency

At each iteration of the Bandit-based RMHC algorithm, the bandit agents manage the selection of the gene with maximal $urgency$ to mutate:

$$i^* = \underset{i \in \{1,2,\ldots,n\}}{\operatorname{argmax}} \; urgency_i. \tag{4}$$

The urgency of each bandit is derived from the standard UCB equation, except that we invert the normal use of the exploitation term, i.e. the first term in the RHS of Equation 5. Intuitively, this says that if a particular state of a bandit is already good, then its value should not be changed. The exploration term is there to ensure that as the total number of iterations N_i increases, so occasionally an apparently poorer option will be tried.

For any 2-armed bandit $i \in \{1, 2, \ldots, n\}$, the $urgency_i$ is defined as

$$urgency_i = -\max_{j \in \{0,1\}} \bar{\Delta}_i(j) + \sqrt{\frac{2\log(N_i + 1)}{N_i(j)}} + \mathcal{U}(1e^{-6}), \tag{5}$$

where N_i is the number of times the i^{th} bit is selected; $N_i(j)$ is the number of times the state j is reached when the i^{th} bit is selected; $\bar{\Delta}_i(j)$ is the empirical mean difference between the fitness values if the state j is reached when the i^{th} bit is selected, i.e., the changing of fitness value; $\mathcal{U}(1e^{-6})$ denotes a uniformly distributed value between 0 and $1e^{-6}$ which is used to randomly break ties.

This means that for each position in the bit string (i.e. for each gene) we have a simple bandit model that requires only 3 additional parameters for book-keeping: one parameter to model the fitness change when flipping a bit from one to zero, another one for the opposite flip, and one to count the number of times that a bandit has been selected ($N_i(j)$).

B. Noise-free case

Algorithm 1 presents the bandit-based RMHC in the noise-free case. To solve a noise-free problem, no resampling is necessary if the evaluation number and fitness value of the best-so-far genome can be saved. It is worth noting that, for problems in which computing the fitness value is difficult or requires high computational cost, saving the fitness of a solution is far less expensive than re-evaluating it again. For the further work, we are interested in applying our proposed approach to more difficult problems (such as game level generation and evaluation [41]). Our main interest is in improving the speed of convergence to approximately optimal states.

Algorithm 1 Bandit-based RMHC in the noise-free case.

Require: $n \in \mathbb{N}^*$: genome length
Require: $m \in \mathbb{N}^*$: dimension of search space
Randomly initialise a genome $\mathbf{x} \in \mathbb{R}^m$
$bestFitSoFar \leftarrow fitness(\mathbf{x})$
$N \leftarrow 1$ ▷ Total evaluation number
while time not elapsed **do**
  Select the element i^* to mutate using Eqs. 4 and 5
  $\mathbf{y} \leftarrow$ after mutating the element i^* of $\mathbf{x}$
  $Fit_{\mathbf{y}} \leftarrow fitness(\mathbf{y})$
  $N \leftarrow N + 1$ ▷ Update the counter
  if $Fit_{\mathbf{y}} \geq bestFitSoFar$ **then**
    $\mathbf{x} \leftarrow \mathbf{y}$ ▷ Update the best-so-far genome
    $bestFitSoFar \leftarrow Fit_{\mathbf{y}}$
  end if
end while
return $\mathbf{x}$

C. Noisy case

We now consider the noisy case. The Bandit-based RMHC in the noisy case is formalised in Algorithm 2. In the noisy case, the best-so-far genome requires multiple evaluations to reduce the effect of noise, this is called resampling.

The statistics of the best-so-far genome are stored, thanks to which, instead of comparing directly the fitness values of the offspring to the one of the best-so-far genome, the average fitness value of the best-so-far genome is compared at each generation. Therefore, the computational cost involved in the evaluation of the genome determines the computational cost of this algorithm.

Resampling has been proved to be a powerful tool to improve the local performance of EAs in noisy optimization [42], [43] and a variety of resampling rules applied to EAs in continuous noisy optimization are studied in [44].

IV. Experimental results

We apply first our proposed algorithm on the OneMax problem and the Royal Road function R_1 in a noise-free case, and then evaluate the performance of our algorithm on the OneMax problem with the presence of noise. Each experiment is repeated 100 times using randomly initialised strings.

A. OneMax

The results in noise-free OneMax problem of different dimensions is presented in Fig. 2. In the noise-free case, the average fitness evaluations used by bandit-based RMHC to

Algorithm 2 Bandit-based RMHC in the noisy case.

Require: $n \in \mathbb{N}^*$: genome length
Require: $m \in \mathbb{N}^*$: dimension of search space
Require: $R \in \mathbb{N}^*$: number of resamplings
Randomly initialise a genome $\mathbf{x} \in \mathbb{R}^m$
$bestFitSoFar \leftarrow \frac{1}{R}\sum_{r=1}^{R} fitness(\mathbf{x})$
$M \leftarrow R$ ▷ Evaluation number of the best-so-far genome
$N \leftarrow R$ ▷ Total evaluation number
while time not elapsed **do**
    Select the element i^* to mutate using Eqs. 4 and 5
    $\mathbf{y} \leftarrow$ after mutating the element i^* of $\mathbf{x}$
    $Fit_{\mathbf{x}} \leftarrow \frac{1}{R}\sum_{r=1}^{R} fitness(\mathbf{x})$
    $Fit_{\mathbf{y}} \leftarrow \frac{1}{R}\sum_{r=1}^{R} fitness(\mathbf{y})$
    $N \leftarrow N + 2R$ ▷ Update the counter
    $averageFitness \leftarrow \frac{bestFitSoFar*M+Fit_{\mathbf{x}}*R}{M+R}$
    if $Fit_{\mathbf{y}} \geq averageFitness$ **then**
        $\mathbf{x} \leftarrow \mathbf{y}$ ▷ Update the best-so-far genome
        $bestFitSoFar \leftarrow Fit_{\mathbf{y}}$
        $M \leftarrow R$
    else
        $bestFitSoFar \leftarrow averageFitness$
        $M \leftarrow M + R$
    end if
end while
return $\mathbf{x}$

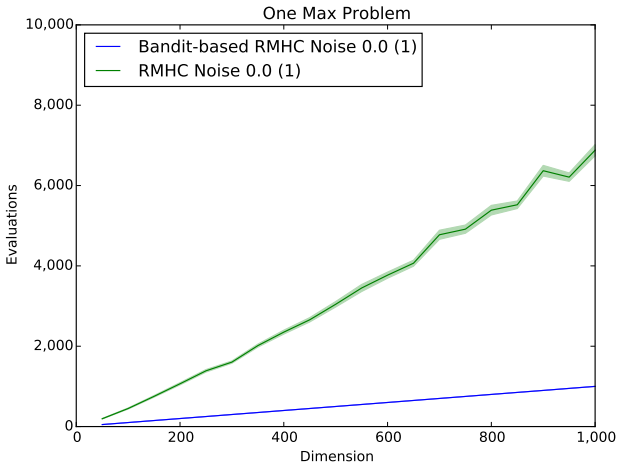


Fig. 2: Average evaluations required to find the optimal solution in the noise-free OneMax problem. Shadow area around the average shows the standard error of the measure.

solve the problem is close to the problem dimension, while the original RMHC required approximate 5 times more budget.

Fig. 3 illustrates the empirical average number of fitness evaluations required to reach the optimum value using RMHC and bandit-based RMHC in the OneMax problem of different dimensions with constant variance noise ($\sim \mathcal{N}(0,1)$). The resampling number in the noisy case is given between brackets. For comparison, the results in noise-free OneMax is also included (blue curves).

As is exhibited in the graph, with the presence of constant variance noise:

- Using RMHC, larger resampling number (10) leads to a faster convergence to the optimum (Fig. 3a) on high-dimension problems; when the problem dimension is low, resampling number equals to 3, 4 or 5 leads to a faster convergence to the optimum (Fig. 3c); the ratio of average fitness evaluations to the problem dimension increases noticeably when a small resampling number is used (Fig. 3c).
- Using bandit-based RMHC, the ratio of average fitness evaluations to the problem dimension remains stable when resampling is used (Fig. 3d); the total evaluation number scales almost linearly with the problem dimension; the optimal resampling number is 2 (red curve).
- As can be seen clearly from the figures, for an identical OneMax problem with the presence of noise, our proposed bandit-based RMHC significantly outperforms the standard RMHC by a very large margin - in some cases requiring a factor of ten fewer fitness evaluations.

B. Royal Road

Fig. 4 shows the empirical average fitness evaluations required to find the optimum of the noise-free Royal Road function using RMHC and bandit-based RMHC, respectively.

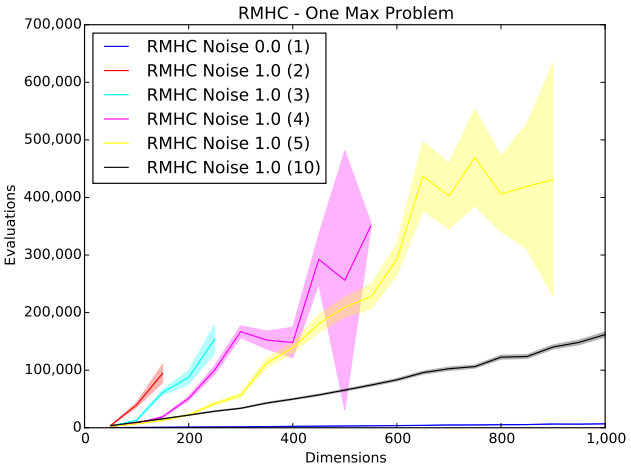
For a fixed length, bigger block size results in a harder problem and more fitness evaluations. The ratio of average fitness evaluations to the problem dimension increases with the problem dimension when a small resampling number is used (Fig. 3c). For an identical problem, bandit-based RMHC required far fewer fitness evaluations than RMHC to find the optimal solution. It can be seen from the curves that for an identical block size, using bandit-based RMHC, the total evaluation number scales linearly with the problem dimension.

In addition, to find the optimum string of 8 blocks of size 8, our bandit-based RMHC used half number of function evaluations than the RMHC used by Mitchell et al. [40], which was the most efficient algorithm in their experiments.

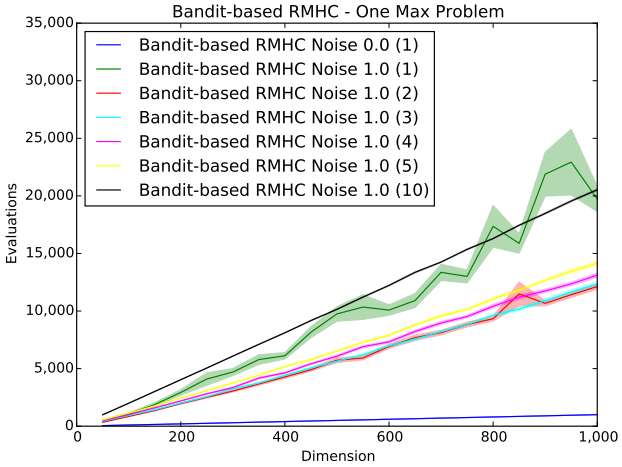
The Royal Road functions involve a harder credit assignment [45] problem than standard OneMax, an important aspect of sequential decision making. The reward for correctly mutating a bit is usually delayed, and dependent on many other correct bit settings.

Regarding credit assignment within the algorithm, the bandit-based RMHC uses *urgency* (Eq. 5) to model this, by attempting to track the fitness gained when switching a gene to a particular value. More use is made of the available information, leading to faster learning (see [46] and [47] for more analysis of the information rates of simple evolutionary algorithms).

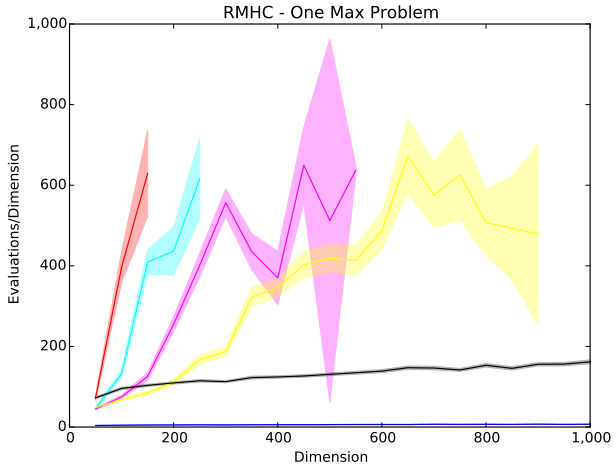
If this information was exploited in a way that was too naïve or too greedy, this could lead the algorithm to rapidly become stuck on poor values, especially for the noisy problems



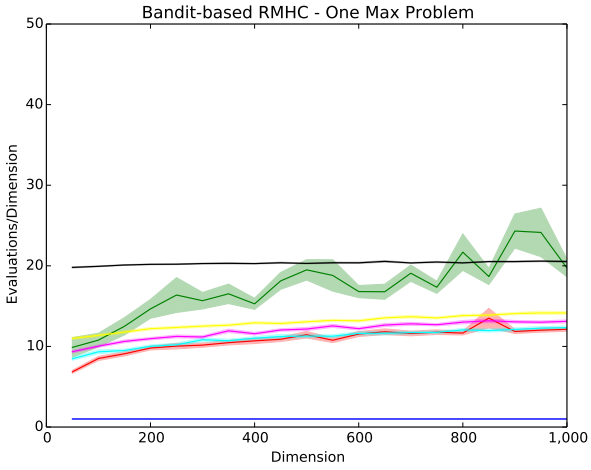
(a) Average evaluations used by RMHC. *y-axis: average #evals*.



(b) Average evaluations used by bandit-based RMHC. *y-axis: average #evals*.



(c) RMHC. *y-axis: average #evals/dim*.



(d) Bandit-based RMHC. *y-axis: average #evals/dim*.

Fig. 3: Performance of RMHC (left) and bandit-based RMHC (right) in the OneMax problem, without noise and with constant variance noise ($\sim \mathcal{N}(0,1)$). The y-axis range of RMHC ((a) and (c)) is **20 times** the ones of bandit-based RMHC ((b) and (d)). The resampling number in the noisy case is given between brackets. The standard error is shown as well as a faded area around the average. In the noisy case, our proposed bandit-based RMHC significantly outperforms RMHC. In the noisy case, the RMHC without resampling is not shown in because it could not solve the problem within $(1000 \times Dimension)$ function evaluations.

tested in this paper. However, the exploration term naturally counteracts such tendencies.

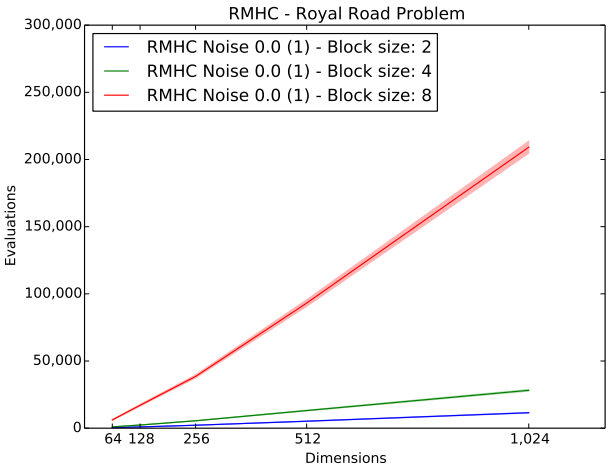
V. Conclusion and Further Work

This paper presented the first bandit-based Random Mutation Hill Climber (RMHC) - a simple but effective type of evolutionary algorithm. The algorithm was compared with the standard RMHC on the OneMax problem and Royal Road function. Tests were also made using a noisy OneMax problem together with resampling in each algorithm to ameliorate the effects of the noise.

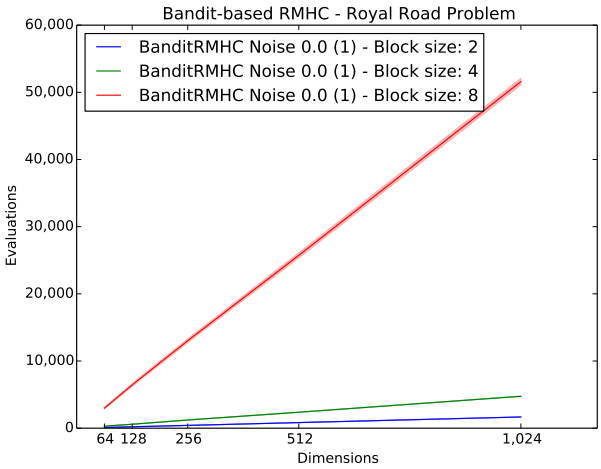
On noise-free and noisy OneMax problems and Royal Road function, our bandit-based RMHC algorithm significantly outperforms the RMHC, in some cases using a factor of ten fewer evaluations in the noisy case.

Furthermore, the fitness evaluations required by the bandit-based RMHC finding the optimal solution is an approximately linear function of the problem dimension when the resampling number is 2. For an identical Royal Road function 8 blocks of size 8), the bandit-based RMHC used half the number of function evaluations than the RMHC used by Mitchell et al. in [40], which was the most efficient algorithm in their experiments.

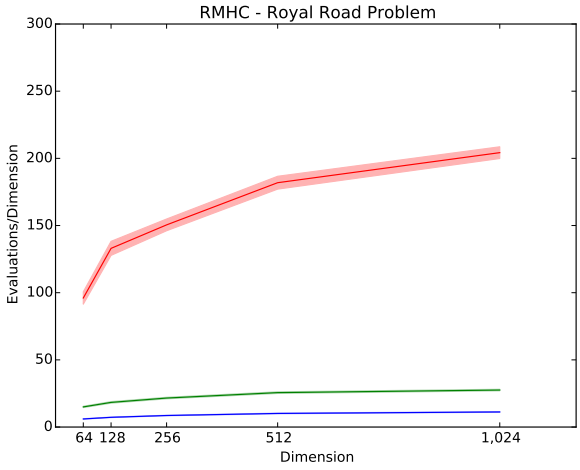
We obtain very promising results using this simple but effective bandit-based RMHC. An advanced version of the



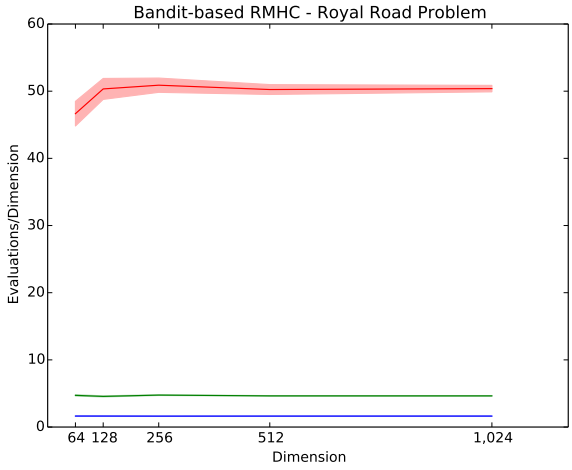
(a) Average evaluations used by RMHC. *y-axis: average #evals*.



(b) Average evaluations used by bandit-based RMHC. *y-axis: average #evals*.



(c) RMHC. *y-axis: average #evals/dim*.



(d) Bandit-based RMHC. *y-axis: average #evals/dim*.

Fig. 4: Performance of RMHC (left) and bandit-based RMHC (right) in the noise-free Royal Road problem with different block sizes. Note that the y-axis range of RMHC in (a) and (c) is **5 times** the ones of bandit-based RMHC in (b) and (d). The standard error is also given as a faded area around the average.

algorithm is designed for a large set of discrete and non-binary optimisation problems where each fitness evaluation is expensive and the fitness is possibly noisy due to some uncertainties, which is quite common in real-world applications [48], [49].

The main work in progress is considering different types of noise, such as variations in the input values or multiplicative noise on the output.

Acknowledgement

We are grateful to Qingfu Zhang (University of Essex, UK) for the valuable discussion.

References

[1] G. E. Box, "Evolutionary Operation: A Method for Increasing Industrial Productivity," *Applied Statistics*, pp. 81–101, 1957.

[2] G. J. Friedman, "Digital Simulation of an Evolutionary Process," *General Systems Yearbook*, vol. 4, no. 171-184, 1959.

[3] W. Bledsoe, "The Use of Biological Concepts in the Analytical Study of Systems," in *OPERATIONS RESEARCH*, vol. 9. INST OPERATIONS RESEARCH MANAGEMENT SCIENCES 901 ELKRIDGE LANDING RD, STE 400, LINTHICUM HTS, MD 21090-2909, 1961, pp. B145–B146.

[4] H. J. Bremermann, "Optimization through Evolution and Recombination," *Self-organizing systems*, vol. 93, p. 106, 1962.

[5] I. Rechenberg, "Cybernetic Solution Path of an Experimental Problem," 1965.

[6] L. J. Fogel, *Artificial Intelligence Through Simulated Evolution.[By] Lawrence J. Fogel... Alvin J. Owens... Michael J. Walsh*. John Wiley & Sons, 1966.

[7] J. Reed, R. Toombs, and N. A. Barricelli, "Simulation of Biological Evolution and Machine Learning: I. Selection of Self-Reproducing Numeric Patterns by Data Processing Machines, Effects of Hereditary Control, Mutation Type and Crossing," *Journal of theoretical biology*, vol. 17, no. 3, pp. 319–342, 1967.

[8] J. C. Gittins, "Bandit Processes and Dynamic Allocation Indices," *Journal of the Royal Statistical Society. Series B (Methodological)*, pp. 148–177, 1979.

[9] D. A. Berry and B. Fristedt, *Bandit Problems: Sequential Allocation of Experiments (Monographs on Statistics and Applied Probability)*. Springer, 1985.

[10] T. Lai and H. Robbins, "Asymptotically Efficient Adaptive Allocation Rules," *Advances in Applied Mathematics*, vol. 6, pp. 4–22, 1985.

[11] P. Auer, N. Cesa-Bianchi, and P. Fischer, "Finite-Time Analysis of the Multiarmed Bandit Problem," *Machine learning*, vol. 47, no. 2-3, pp. 235–256, 2002.

[12] C. B. Browne, E. Powley, D. Whitehouse, S. M. Lucas, P. I. Cowling, P. Rohlfshagen, S. Tavener, D. Perez, S. Samothrakis, and S. Colton, "A Survey of Monte Carlo Tree Search Methods," *Computational Intelligence and AI in Games, IEEE Transactions on*, vol. 4, no. 1, pp. 1–43, 2012.

[13] P. Auer, "Using Confidence Bounds for Exploitation-Exploration Trade-offs," *The Journal of Machine Learning Research*, vol. 3, pp. 397–422, 2003.

[14] J.-Y. Audibert, R. Munos, C. Szepesvari *et al.*, "Use of Variance Estimation in the Multi-Armed Bandit Problem," 2006.

[15] S. Bubeck, R. Munos, and G. Stoltz, "Pure Exploration in Multi-Armed Bandits Problems," in *Algorithmic Learning Theory*. Springer, 2009, pp. 23–37.

[16] A. Garivier and O. Cappé, "The KL-UCB Algorithm for Bounded Stochastic Bandits and beyond," *arXiv preprint arXiv:1102.2490*, 2011.

[17] S. Bubeck and N. Cesa-Bianchi, "Regret Analysis of Stochastic and Nonstochastic Multi-Armed Bandit Problems," *arXiv preprint arXiv:1204.5721*, 2012.

[18] L. Kocsis and C. Szepesvári, "Discounted UCB," in *2nd PASCAL Challenges Workshop*, 2006.

[19] M. D. Grigoriadis and L. G. Khachiyan, "A Sublinear-Time Randomized Approximation Algorithm for Matrix Games," *Operations Research Letters*, vol. 18, no. 2, pp. 53–58, Sep 1995.

[20] P. Auer, N. Cesa-Bianchi, Y. Freund, and R. E. Schapire, "Gambling in a Rigged Casino: the Adversarial Multi-Armed Bandit Problem," in *Proceedings of the 36th Annual Symposium on Foundations of Computer Science*. IEEE Computer Society Press, Los Alamitos, CA, 1995, pp. 322–331.

[21] D. L. St-Pierre and J. Liu, "Differential Evolution Algorithm Applied to Non-Stationary Bandit Problem," in *2014 IEEE Congress on Evolutionary Computation (IEEE CEC 2014)*, Beijing, Chine, Jul. 2014. [Online]. Available: http://hal.inria.fr/hal-00979456

[22] R. Storn and K. Price, "Differential Evolution – A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces," *Journal of global optimization*, vol. 11, no. 4, pp. 341–359, 1997.

[23] A. Benbassat and M. Sipper, "Evolving Artificial Neural Networks with FINCH," in *Proceedings of the 15th annual conference companion on Genetic and evolutionary computation*. ACM, 2013, pp. 1719–1720.

[24] A. M. Alhejali and S. M. Lucas, "Using Genetic Programming to Evolve Heuristics for a Monte Carlo Tree Search Ms Pac-Man agent," in *Computational Intelligence in Games (CIG), 2013 IEEE Conference on*. IEEE, 2013, pp. 1–8.

[25] S. M. Lucas, S. Samothrakis, and D. Perez, "Fast Evolutionary Adaptation for Monte Carlo Tree Search," in *Applications of Evolutionary Computation*. Springer, 2014, pp. 349–360.

[26] P. Rolet and O. Teytaud, "Bandit-Based Estimation of Distribution Algorithms for Noisy Optimization: Rigorous Runtime Analysis," in *Learning and Intelligent Optimization*. Springer, 2010, pp. 97–110.

[27] K. Li, A. Fialho, S. Kwong, and Q. Zhang, "Adaptive Operator Selection with Bandits for a Multiobjective Evolutionary Algorithm based on Decomposition," *Evolutionary Computation, IEEE Transactions on*, vol. 18, no. 1, pp. 114–130, 2014.

[28] S. M. Lucas and T. J. Reynolds, "Learning DFA: Evolution versus Evidence Driven State Merging," in *Evolutionary Computation, 2003. CEC'03. The 2003 Congress on*, vol. 1. IEEE, 2003, pp. 351–358.

[29] ——, "Learning Deterministic Finite Automata with a Smart State Labeling Evolutionary Algorithm," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 27, no. 7, pp. 1063–1074, 2005.

[30] D. E. Goldberg *et al.*, *Genetic Algorithms in Search Optimization and Machine Learning*. Addison-wesley Reading Menlo Park, 1989, vol. 412.

[31] J. D. Schaffer and L. J. Eshelman, "On Crossover as an Evolutionarily Viable Strategy," in *ICGA*, vol. 91, 1991, pp. 61–68.

[32] S. Droste, T. Jansen, and I. Wegener, "Upper and Lower Bounds for Randomized Search Heuristics in Black-Box Optimization," *Theory of computing systems*, vol. 39, no. 4, pp. 525–544, 2006.

[33] B. Doerr and C. Winzen, "Memory-Restricted Black-Box Complexity of OneMax," *Information Processing Letters*, vol. 112, no. 1, pp. 32–34, 2012.

[34] B. Doerr and C. Doerr, "Black-Box Complexity: from Complexity Theory to Playing Mastermind," in *Proceedings of the 15th annual conference companion on Genetic and evolutionary computation*. ACM, 2013, pp. 617–640.

[35] C. Doerr and J. Lengler, "OneMax in Black-Box Models with Several Restrictions," *Algorithmica*, pp. 1–31, 2016. [Online]. Available: http://dx.doi.org/10.1007/s00453-016-0168-1

[36] C. Qian, Y. Yu, and Z.-H. Zhou, "Analyzing Evolutionary Optimization in Noisy Environments," *Evolutionary computation*, 2015.

[37] J. Liu, M. Fairbank, D. Pérez-Liébana, and S. M. Lucas, "Optimal resampling for the noisy onemax problem," *arXiv preprint arXiv:1607.06641*, 2016.

[38] S. Droste, "Analysis of the (1+ 1) EA for a Noisy OneMax," in *Genetic and Evolutionary Computation–GECCO 2004*. Springer, 2004, pp. 1088–1099.

[39] M. Mitchell, S. Forrest, and J. H. Holland, "The Royal Road for Genetic Algorithms: Fitness Landscapes and GA Performance," in *Proceedings of the first european conference on artificial life*. Cambridge: The MIT Press, 1992, pp. 245–254.

[40] M. Mitchell and J. H. Holland, "When will a Genetic Algorithm Outperform Hill-Climbing?" 1993.

[41] A. Khalifa, D. Perez-Liebana, S. M. Lucas, and J. Togelius, "General Video Game Level Generation," *Proceedings of the 2016 Genetic and Evolutionary Computation Conference (GECCO'16)*, 2016.

[42] D. V. Arnold and H.-G. Beyer, "A General Noise Model and its Effects on Evolution Strategy Performance," *Evolutionary Computation, IEEE Transactions on*, vol. 10, no. 4, pp. 380–391, 2006.

[43] H.-G. Beyer, *The Theory of Evolution Strategies*. Springer Science & Business Media, 2013.

[44] J. Liu, "Portfolio Methods in Uncertain Contexts," Ph.D. dissertation, INRIA, 12 2015.

[45] R. S. Sutton, "Temporal Credit Assignment in Reinforcement Learning," 1984.

[46] S. M. Lucas, "Investigating Learning Rates for Evolution and Temporal Difference Learning," in *Computational Intelligence and Games, 2008. CIG'08. IEEE Symposium On*. IEEE, 2008, pp. 1–7.

[47] ——, "Estimating Learning Rates in Evolution and TDL: Results on a Simple Grid-World Problem," in *Computational Intelligence and Games (CIG), 2010 IEEE Symposium on*. IEEE, 2010, pp. 372–379.

[48] K. Kunanusont, R. D. Gaina, J. Liu, D. Perez-Liebana, and S. M. Lucas, "The n-tuple bandit evolutionary algorithm for automatic game improvement," in *Evolutionary Computation, 2017. CEC'17. The 2017 Congress on*. IEEE, 2017.

[49] J. Liu, J. Togelius, D. Perez-Liebana, and S. M. Lucas, "Evolving game skill-depth using general video game ai agents," in *Evolutionary Computation, 2017. CEC'17. The 2017 Congress on*. IEEE, 2017.