Algorithm Performance Comparison for Integer-Valued OneMax

Xiaoyue Li
Hasso Plattner Institute, Potsdam University
Potsdam, Germany
Xiaoyue.Li@hpi.de

Timo Kötzing
Hasso Plattner Institute, Potsdam University
Potsdam, Germany
Timo.Koetzing@hpi.de

ABSTRACT

Recently, several continuous-domain optimizers have been employed to solve mixed-integer black box optimization (MI-BBO) problems by adjusting them to handle the discrete variables as well. In this work we want to compare how these adjusted algorithms perform on purely discrete variables when compared with algorithms designed to handle discrete domains.

We use the algorithm $RLS_{\alpha,\beta}$ from the literature, which was analyzed theoretically on a specific problem class. We experimentally optimize the parameters α and β for further analysis.

Second, we make a comprehensive comparison of $RLS_{\alpha,\beta}$ with algorithms from the continuous domain on a generalization of OneMax to $\mathbb{Z}^n$. We find that $RLS_{\alpha,\beta}$ shows better performance on this discrete benchmark functions than the other algorithms.

Overall, these results show that adaptations of continuous algorithms are not suited for MI-BBO problems, and that research combining the strengths of both approaches is more promising.

CCS CONCEPTS

• **Mathematics of computing → Discrete optimization**.

KEYWORDS

CMA-ES; discrete search space; RLS

ACM Reference Format:
Xiaoyue Li and Timo Kötzing. 2024. Algorithm Performance Comparison for Integer-Valued OneMax. In *Genetic and Evolutionary Computation Conference (GECCO '24 Companion), July 14–18, 2024, Melbourne, VIC, Australia.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3638530.3654287

1 INTRODUCTION

In recent works, optimization algorithms meant for the continuous domain were extended to also handle discrete variables. In particular, the Covariance Matrix Adaptation Evolution Strategy (CMA-ES) were extended to handle integer-valued variables (see [6] and [9], respectively). The motivation is to design algorithms which can simultaneously handle continuous and discrete variables in an optimization problem, thus addressing Mixed-Integer Black Box Optimization (MI-BBO) problems.

So far, this effort came mostly from the side of continuous optimization experts. With this paper we want to add a view from discrete optimization. Concretely, we compare recent algorithms adapted to handle integer-valued variables with an algorithm developed specifically for discrete search spaces. In our analyses, we drop the continuous variables from the search space, since we specifically want to understand the performance on the discrete variables, and not directly develop a better algorithm for MI-BBO problems.

As the algorithm for discrete problems we consider a variant of Random Local Search (RLS) from [2].[1] This algorithm maintains a single current search point and adjusts it only in one dimension per iteration; only the offspring which is not worse than the parent is accepted. The magnitude of the change in a dimension is controlled by a step width parameter per dimension, which the algorithm self-adapts with a success rule. See Section 2 for details.

Also from [2] we get a generalization of OneMax, given as the ℓ_1-distance to the unknown optimum and start our analysis with this test function. We are particularly interested to see the impact of the initial distance to the optimum, but equally informative is the performance in dependence on the dimension of the problem.

The RLS with self-adaptive step size was mathematically shown to optimize generalized OneMax in $\Theta(n(\log n + \log r))$ function evaluations [2]. While this theorem gave some indication about what parameters are useful, these parameters have not been experimentally optimized. In Section 2.4 we show that (a) the RLS algorithm is robust to the exact choice of parameters with similar performance for similar parameter values; and (b) the parameter values $\alpha = 10$ and $\beta = 0.4$ lead to a very good performance on a specific mid-sized instance, which we fix for further analyses.

In Section 3 we compare the performance of CMA-ESwM and CMA-ESwR in terms of their median run time on the generalized OneMax fitness function. We conclude that the $RLS_{\alpha,\beta}$ is significantly better suited for the purely discrete search space.

Finally, we give a general discussion of our findings in Section 4 as well as concluding remarks. Before getting to the main part of the paper, we give details on problems and algorithms in Section 2.

1.1 Related Work

In [1], the authors propose a Generalized Evolutionary Strategy (GES) for solving a specific industry optimization problem, the optical filter optimization. This paper is the first one proposing a generalized heuristic approach for mixed integer optimization, named Mixed Integer Evolutionary Strategy (MIES).

Authors in [7] continue in the direction of MIES. They begin with a conceptual introduction of MIES and then present theoretical result in the optimal self-adaptation of step sizes and mutation rates on a specific model. The authors also summarize the MIES application, which prove the MIES on a general class of mixed integer nonlinear programming problems.

[1]Note that [2] refers to the setting of integer-valued variables as "multi-valued" variables. This phrase seems to imply a boundedness of the domain of the variables, which was true in that paper, but not necessarily in the present paper.

See also [8] for further discussions on evolutionary strategies for mixed-integer optimization.

2 PROBLEM AND ALGORITHMS

In this section we introduce the problem and algorithms we consider.

Specifically, we analyze the generalized OneMax problem proposed in [2], called integer-valued OneMax. This function extends the search space from $\{0,1\}^n$ to $\{0,\cdots,r-1\}^n$.

We formally define the integer-valued OneMax as follows, extending the definition from [2] slightly. For a given $R \in \mathbb{Z}^n$, we have the fitness function as

$$f_r : \mathbb{Z}^n \to \mathbb{Z}_{\geq 0}, x \mapsto \sum_{i=1}^{n} |x_i - R_i|.$$

We are interested in minimizing this fitness function.

In the following subsection, we introduce the algorithms for comparison. We consider two algorithms based on CMA-ES, followed by a version of RLS.

2.1 CMA-ES with Deterministic Rounding

Given that CMA-ES incorporates a self-adaptive step size, we explore whether the algorithm itself possesses the capability to handle the integer-valued OneMax problem through a straightforward treatment. The only modification made to CMA-ES involves the introduction of a deterministic rounding function to map the samples from the $\mathbb{R}^n$ space to the discrete $\mathbb{Z}^n$ space. No other aspects of the CMA-ES algorithm are altered. Notice in [5] the authors introduced an alteration of CMA-ES which implements box constraints to prevent the mean value from deviating from zero. This method lies in the sampling phase while CMA-ESwR does not modify the samples. The changes lies in the transformation of the objective function. Specifically, each candidate CMA-ES sample has its individual elements rounded to the nearest integer when calculating the fitness value.

2.2 CMA-ES with Margin

Based on CMA-ES, in [3] the authors proposed another approach called CMA-ESwM. This approach aims to address the issue of stagnation caused by the discretization by deterministic rounding. The authors acknowledge the existence of a method introduced in [5], which implements box constraints to prevent the mean value from deviating from zero. However, this method fails to completely eliminate stagnation in the negative domain. To overcome this limitation, the authors introduce a lower bound, referred to as a *margin*, for adjusting the controlling parameters of the Multivariate Gaussian Distribution (MGD) used for sampling the population pool and the mean value $\mathbf{m}$.

For CMA-ES and CMA-ESwM, we directly take the code uploaded by the authors of [3] to GitHub.[2]

For both variants of CMA-ES, we implemented the restart strategy as proposed in [4]. In order to provide a more informative comparison, we focus not only on the success rate but also on the runtime of various algorithms when attempting to find the optimum. To facilitate this, we adopt one of the strategies outlined in [4] to employ restarts, which involves resetting all sample values and doubling the population size λ.

[2]https://github.com/EvoConJP/CMA-ES_with_Margin

2.3 Random Local Search with Self-Adaptive Operator

The authors in [2] propose another perspective of analyzing the integer problem. They propose an algorithm called $RLS_{\alpha,\beta}$. The algorithm has a vector v called *velocity* which is the *step size* for the algorithm to make progress in each iteration. The vector $\mathbf{v}$ is controlled by parameters α and β, where $\alpha > 1$ increases the step size from $\mathbf{v}$ to $\alpha \cdot \mathbf{v}$ when the algorithm finds a better search point, while $0 < \beta < 1$ decreases the step size by a factor of β when the algorithm overshoots or does not find a better search point.

Concretely, the $RLS_{\alpha,\beta}$ is given in Algorithm 1. We provide the code on GitHub.[3]

Algorithm 1: $RLS_{\alpha,\beta}$ minimizing the integer-valued OneMax problem

```
1  Initialization: Choose $\mathbf{V} \in [1,10]^n$ u.a.r. $t = 0$;
2  Initialize $\mathbf{x}$ as an all 0 integer string;
3  while stopping criterion not met do
4      $\mathbf{y} \leftarrow \mathbf{x}$ ;
5      Choose $i \in [n]$ u.a.r.;
6      With probability $\frac{1}{2}$ let $\mathbf{y}_i \leftarrow \mathbf{x}_i + \lfloor \mathbf{V}_i \rfloor$ otherwise let
         $\mathbf{y}_i \leftarrow \mathbf{x}_i - \lfloor \mathbf{V}_i \rfloor$ ;
7      if $f(\mathbf{y}) < f(\mathbf{x})$ then
8          $\mathbf{V}_i \leftarrow \lfloor \alpha \mathbf{V}_i \rfloor$ ;
9      else
10         $\mathbf{V}_i \leftarrow max\{1, \beta \mathbf{V}_i\}$ ;
11     $\mathbf{x} \leftarrow \mathbf{y}$ ;
```

The authors in [2] also provide a theorem and a rigorous proof that the run time of $RLS_{\alpha,\beta}$ optimizing an integer-valued OneMax problem based on some constraints on the value of α and β is $\Theta(n(\log n + \log r))$. The theorem is presented as follows.

Theorem 1. *For constants a,b satisfying* $1 < a \leq 2, 1/2 < b \leq 0.9$, $2ab - b - a > 0$, $a + b > 2$ *and* $a^2b > 1$ *(one can choose, for example,* $a = 1.7$ *and* $b = 0.9$*) the expected run time of* $RLS_{\alpha,\beta}$ *in any generalized r-valued OneMax function is* $\Theta(n(\log n + \log r))$.

Furthermore, we notice for $RLS_{\alpha,\beta}$, there are no specific guidelines for parameter selection of α and β. Consequently, as a preliminary step documented in the next section, we conduct a series of experiments to determine a suitable pair of parameters for $RLS_{\alpha,\beta}$.

2.4 Grid Search for Parameter of RLS

To find the optimal combination of α and β in optimizing the integer-valued OneMax problem, we design a set of trial tests for $n = 40$ and $r = 1000$. We apply a grid search for $\alpha \in [2,20]$ with a step size of 1 and $\beta \in [0.1, 0.9]$ with a step size of 0.1. We let the algorithm run 200 times independently for each combination of α and β and

[3]https://github.com/SherryXiaoyueLi/RLS_ab

take the median value of the fitness evaluation as the run time for comparison. Results are given in Figure 1.

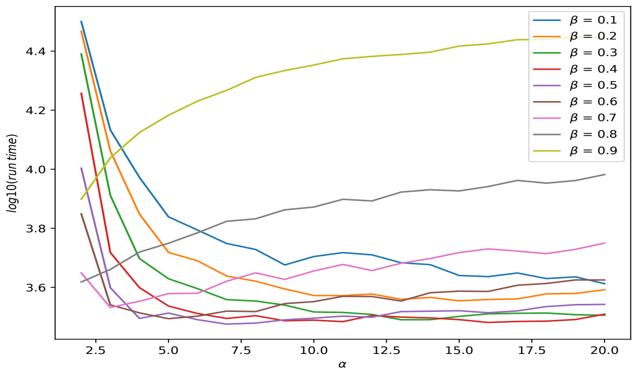

Figure 1: Run time for $RLS_{\alpha,\beta}$ under various α and β optimizing integer-valued OneMax when $n = 40$, $r = 1000$. Each line is the median value out of 200 independent runs.

From Figure 1 we notice first that a value of $\beta \geq 0.7$ is bad: the run time keeps increasing as α increasing and is significantly higher than the lower values of β. Then we see that $\beta \leq 0.3$ (lines in blue, orange, and green) does not help shorten the run time even with large α. Also when $\alpha \geq 10.0$ and $\beta \geq 0.4$, the run time does not show much of a difference. Therefore we select $\alpha = 10$ and $\beta = 0.4$ for all further experiments.

3 COMPARISON OF ALGORITHMS ON MULTI-VALUED ONEMAX PROBLEM

In this section, we present a comprehensive comparison of $RLS_{\alpha,\beta}$, CMA-ESwM, and CMA-ESwR in their performance in optimizing the integer-valued OneMax problem.

3.1 Comparison of Algorithms in Bounded Space

We notice that, for the CMA-ESwM, because of the *encoding function*, a discrete space is defined as a bounded set of integers. The encoding function defined by CMA-ESwM employs the parameter K_j, which represents the number of candidate integers for the j-th real value, and this parameter K_j is the bound. In the experimental section of [3], the authors defined the bound as $[-10, 11]^n$. We expand the boundary to $[-2 \cdot r, 2 \cdot r]^n$, where r is the target value for each component of our integer-valued OneMax objective.

In the initialization of $RLS_{\alpha,\beta}$, due to the bounded search space, we can use the prior knowledge about the boundary of the search space to initialize v. Therefore, we change the initialization of the velocity in Step 1 of Algorithm 1 to choose $v \in [r/4, r/2]^n$ uniformly at random. This implies that the algorithm will initially search the space in large steps and close in using smaller steps later. Note that, due to choosing the initial velocity randomly, the algorithm has no unfair advantage in finding the optimum, which might happen if the distance to the optimum is exactly the initial step size.

In order to assess performance, we conducted a comparative analysis using a similar methodology as in the previous section.

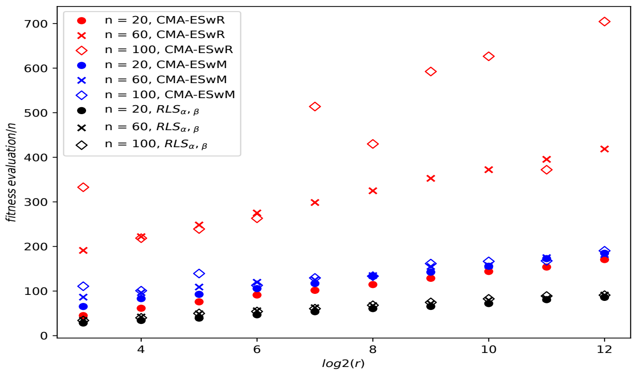

Figure 2: Comparison of run time for both CMA-ES variants and $RLS_{\alpha,\beta}$. Each marker is the median run time of 50 independent runs. The x-axis stands for $\log_2(r)$ and the y-axis stands for fitness evaluation normalized by n. Different colors stand for different algorithms and different marker style stands for different n.

We set $n \in \{20, 60, 100\}$ and $r \in \{2^j \mid j \in \{3, \ldots, 13\}\}$. As a result, we take the median value over 50 independent runs because the restart strategy of CMA-ESwM and CMA-ESwR leads to a significant spread in the run times, depending on whether the restart is initiated or not. To improve clarity, we normalize the run time by n. The results are presented in Figure 2.

From Figure 2, we see that $RLS_{\alpha,\beta}$ outperforms both variants of CMA-ES for all values of n and r. We also observe a linear relation between run time and $\log_2(r)$. In particular, for $n = 20$, the advantage of $RLS_{\alpha,\beta}$ over CMA-ESwR is a multiplicative factor of at least 2.1 and the multiplicative advantage over CMA-ESwR is 1.5 for small r. For larger r, starting at $r = 2^6$, the multiplicative advantage is already 2.0 and stays above 2 for all larger r.

For $n = 60$, the multiplicative advantage is 2.5 over CMA-ESwM and 5.5 over CMA-ESwR even when $r = 2^3$. As r increases, the advantage stays at an approximate multiplicative factor of at least 2.0 and 4.5 over CMA-ESwM and CMA-ESwR, respectively.

Finally, for $n = 100$, when $r = 2^3$, the multiplicative advantage is 3.0 over CMA-ESwM and 9.7 over CMA-ESwR. And as r increases, starting from $r = 2^6$ the multiplicative advantage stabilizes around 2.0 over CMA-ESwM and 7.0 over CMA-ESwR, respectively.

Furthermore, both CMA-ESwM and $RLS_{\alpha,\beta}$ appear to scale linearly with n (blue and black markers in different shapes do not show much of a difference), while CMA-ESwR does not. This is because the restart times for CMA-ESwR depend on the values of n and r. This shows that CMA-ESwR is more sensitive to the restart strategy compared to CMA-ESwM. Moreover, this demonstrates that the currently selected restart strategy, while aiding in the search for the optimal solution, may not be well-suited for analysis.

In general, even for small instance sizes, the advantage of $RLS_{\alpha,\beta}$ is a multiplicative factor of at least 1.5. For larger r, starting at $r = 2^6$, the multiplicative advantage is already about 2 and increases to 2.5 for $r = 2^{13}$. In some extreme cases, for example when $n = 100$, the multiplicative factor against CMA-ESwR is more than 9, and against CMA-ESwM it is more than 3, even when $r = 2^3$. This

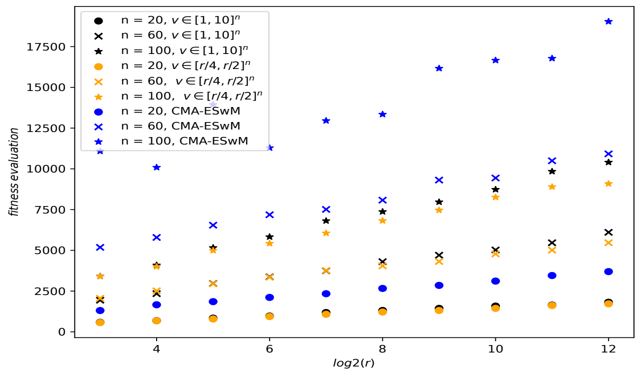

Figure 3: Comparison of run times for the $RLS_{\alpha,\beta}$ and CMA-ESwR with differently initialized v. The x-axis stands for $\log_2(r)$ and the y-axis stands for fitness evaluation. Each marker is the median run time of 50 independent runs, which different styles stands for different n and different colors stand for differently initialized v.

highlights the suitability of $RLS_{\alpha,\beta}$ for effectively optimizing the integer-valued OneMax problem.

3.2 Another Perspective: the Unbounded Search Space

Another perspective to be considered for comparing algorithms is that of $\mathbb{Z}^n$ as the search space, i.e. the discrete space with no boundary.

As previously discussed, CMA-ESwM needs a bound for the *encoding function* which means it is not applicable for the problem defined on the unbounded search space. In contrast to this, CMA-ESwR will have no difference in the bounded and unbounded space since CMA-ES has no boundary for samples and the deterministic rounding function mapping the sampled value in $\mathbb{R}^n$ to $\mathbb{Z}^n$ will also have no restrictions. Therefore, results will be the same as the results shown in Figure 2.

As for $RLS_{\alpha,\beta}$, the main difference is in the initialization of the velocity v. Since it is an unbounded space, the algorithm cannot know what a suitable initial velocity would be. Therefore, the initialization of v needs to be independent of r. We design groups of experiments accordingly.

We select the same parameter setup as for the bounded space but differently initialized v. For the unbounded search space, we initialize $v \in [1, 10]^n$ uniformly at random. The rest is the same as before. Results are given in Figure 3.

From Figure 3 we see that, when $r > 2^7$, $RLS_{\alpha,\beta}$ with initialized $v \in [r/4, r/2]^n$ has a shorter run time than $RLS_{\alpha,\beta}$ with initialized $v \in [1, 10]^n$, as indicated by the non-overlapping black and orange markers. However the observed discrepancy in performance is minor. This suggests that for $RLS_{\alpha,\beta}$, possessing prior knowledge regarding the optimum within the bounded and unbounded search spaces has a negligible impact on the performance of $RLS_{\alpha,\beta}$. This intuitively comes from the exponential increase in velocity while the velocity is too low (see [2]), while the main bottleneck of the optimization lies in the later parts of the optimization process.

4 DISCUSSION AND CONCLUSION

$RLS_{\alpha,\beta}$ is specific to the discrete domain and employs a clever step size adaptation. Thus, it is not surprising that it outperforms other algorithms which are not meant for this domain. However, we believe that RLS offers an excellent base line on what performance can look like for discrete problems.

While discrete problems require smaller and smaller steps to approximate local optima, the setting in the discrete domain is different: We want to find the optimum, which might be far away. Our analyses have shown that a lot of algorithms suffer just from moving the optimum away from the start point (while an optimum near the starting point is common for continuous optimization). We believe that any algorithm targeting a discrete domain needs to be able to follow these long ascents to the optimum.

Algorithms such as CMA-ESwM and CMA-ESwR add features which are reminiscent of features in discrete-domain algorithms, and try to mimic corresponding behavior. We show that using the original discrete algorithms employ these strategies more effectively. Thus, we believe that, instead of making continuous algorithms behave like discrete algorithms, one should focus building directly on the strengths of discrete algorithms.

Clearly, our analysis only on an integer-valued function is not the same as a truly mixed-integer problem. With this work, we want to take the first step of seeing how different algorithmic approaches fare on the discrete part of the search space.

ACKNOWLEDGMENTS

This work is supported by grant FR 2988/17-1 by the German Research Foundation (DFG).

REFERENCES

[1] Thomas Bäck and Martin Schütz. 1995. Evolution Strategies for Mixed Integer Optimization of Optical Multilayer Systems. In *Evolutionary Programming IV – Proc. Fourth Annual Conf. Evolutionary Programming*. The MIT Press, 33–51.
[2] Benjamin Doerr, Carola Doerr, and Timo Kötzing. 2017. Static and Self-Adjusting Mutation Strengths for Multi-valued Decision Variables. *Algorithmica* 80 (07 2017), 1732–1768. https://doi.org/10.1007/s00453-017-0341-1
[3] Ryoki Hamano, Shota Saito, Masahiro Nomura, and Shinichi Shirakawa. 2022. CMA-ES with Margin: Lower-Bounding Marginal Probability for Mixed-Integer Black-Box Optimization. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO '22)*. Association for Computing Machinery, New York, NY, USA, 639–647. https://doi.org/10.1145/3512290.3528827
[4] Nikolaus Hansen. 2009. Benchmarking a BI-Population CMA-ES on the BBOB-2009 Function Testbed. In *Proceedings of the 11th Annual Conference Companion on Genetic and Evolutionary Computation Conference: Late Breaking Papers (GECCO '09)*. Association for Computing Machinery, New York, NY, USA, 2389–2396. https://doi.org/10.1145/1570256.1570333
[5] Nikolaus Hansen. 2011. *A CMA-ES for Mixed-Integer Nonlinear Optimization.* Technical Report. INRIA.
[6] Koki Ikeda and Isao Ono. 2023. Natural Evolution Strategy for Mixed-Integer Black-Box Optimization. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO '23)*. Association for Computing Machinery, New York, NY, USA, 831–838. https://doi.org/10.1145/3583131.3590518
[7] Rui Li, Michael Emmerich, Jeroen Eggermont, Thomas Bäck, Martin Schütz, Jouke Dijkstra, and Johan Reiber. 2013. Mixed integer evolution strategies for parameter optimization. *Evolutionary Computation* 21, 1 (2013), 29–64.
[8] Günter Rudolph. 2012. *Handbook of Natural Computing: Theory, Experiments, and Applications.* Springer-Verlag, Berlin-Heidelberg, Germany, Chapter Evolutionary Strategies, 673–698.
[9] Yohei Watanabe, Kento Uchida, Ryoki Hamano, Shota Saito, Masahiro Nomura, and Shinichi Shirakawa. 2023. (1+1)-CMA-ES with Margin for Discrete and Mixed-Integer Problems. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO '23)*. Association for Computing Machinery, New York, NY, USA, 882–890. https://doi.org/10.1145/3583131.3590516