UNIVERSITY OF SUSSEX

A Comprehensive Investigation of Basic Computer-Based Optimization Methodologies

2020

University of Sussex – School of Engineering and Informatics

Abstract

In this short study, we perform a comparative scrutinization of two basic optimization methodologies. Through two simple modifications, we were able to ameliorate the performance of the most promising of the two, the hill-climber, to achieve a performance that is more comparable to that of a much more sophisticated EDA, than its vanilla implementation.

Contents

1. Motivation

There is almost no limit to the scope of real-world problems to which optimisation can be applied, but with so many algorithms available, using the one that is best suited to any given problem could lead to drastically increased performance, both in terms of the fitness achieved, and the time taken to do so. Depending on the problem in question, it's not difficult to see how these differences could translate to unrestricted costs in its respective real-world application. This is where the motivation for this project lies – its seemingly limitless scope, in conjunction with its scalable potential gain, depending on the nature of the problem in question.

My personal fascination with evolutionary algorithms stems from work of a similar nature I undertook during my undergraduate degree, and should this introductory piece spark a similar motivation in at least one other individual, I would see that as a worthwhile contribution to a compelling field that is still in its infancy - and it is perhaps here that a secondary motivation for this project lies.

2. Project Aims and Objectives

For this project I will be creating a MatLab script that will allow me to perform an in-depth analysis of various optimization methodologies, namely hill-climbers and gradient ascent, and how their performances compare to each other, as well then contrasting this to much more complex Estimation of Distribution Algorithms (EDA's) such as CMA-ES (covariance Matric Adaptation Evolution Strategy) and PSO (Particle Swarm Optimization). A particular focus will be placed on their abilities to escape suboptimal-local optima, as well as the time required to achieve an optimised solution. The goal of this is to determine exactly where the comparative strengths and weaknesses of each primitive algorithm lies - in an effort to categorise the problems for which each is best suited. Once these have been established, as a stretch goal I hope to be able to use the better performer of the two, in such a way that their key downfalls are largely negated, to hopefully demonstrate how a performance can be achieved that is more comparable to that of an EDA, as opposed to its own vanilla implementation, with only minimal modification.

During our investigation I hope to broaden the reader's (as well as my own) understanding of the principals behind computer-based optimization, making the otherwise potentially intimidating field of complex evolutionary algorithms more accessible to the layman.

3. Related Work

It is no secret that hill-climbers excel on a convex heuristic, that is one with a single global-optima, but once you introduce multiple local-optima, their success rate becomes partially dependent on their starting position. Although much research has gone in to solving this [1] [2] [3] through various methods such as the hybridization of hill-climbing with other stochastic methods [4], for the scope of our study we will be limiting our modifications to the variation of its maximum mutation size.

Gradient ascent is the second of the two primitive optimisation methodologies that I will be considering, it differs from the hill-climber in that it is totally deterministic. That is, it will always produce an identical result, given an identical starting position. In contrast, the hill-climber is stochastic - its movement is based on random mutations. Although gradient ascent has already been proven effective in a multitude of applications [5] [6], our work will differ by instead investigating whether this commendable performance is maintained in the situations where the more complex EDA's may suffer, and vice versa, or if they do indeed overlap, and if so, to what extent.

EDA's function similarly to GA's, with the key difference being that once the fittest members of the population are selected (in accordance with the implemented selection function), they construct a probabilistic model which attempts to estimate the probability distribution of the selected solutions. The new genotypes are then created by sampling this encoded distribution [7]. The majority of other metaheuristics use fixed problem-specific operators, whilst one significant advantage of EDAs is their ability to adapt them to suit the structure of the problem [8]. As demonstrated by Sastry et al [9], this can lead to significantly improved scalability on extremely large problems that cannot be solved with other techniques, however building the probabilistic models is often more time consuming than simple selection operators such as tournament selection, and therefore this may result in them being outperformed by our two more primitive optimisation techniques in certain scenarios.

4. Intended Methodology

4.1 Software

I will use MatLab to carry out this investigation, as it is a programming environment with which I have previous relevant experience. Its ability to work with and operate on large matrices of data mean it is well suited to the optimisation-based task in hand - where many collections of related data points must be stored, visualised and compared. Should I wish to plot 3D fitness landscapes to visualise these results, this is another area in which it outshines other options such as a python implementation, due to extent of the in-built tools available to visualise and manipulate these plots. Since developing the EDA's is not the focus of this project, I will be using implementations provided by the academic and professional scientific resource Yarpiz [10].

4.2 Fitness Landscapes

For the purpose of this investigation I will be comparing the algorithms performance over various different fitness landscapes, with varying degrees of complexity. Those more complex landscapes will play host to more suboptimal local-optima. The rationale behind this is to allow for the different conditions necessary in order to investigate the various algorithms strengths and weaknesses regarding their ability to escape these local optima, and if their ability to do so comes at the cost of reduced performance where this behaviour isn't necessary.

4.2.1 Six-Hump Camel Function

The first function I wish to consider is the Six-Hump Camel Function [11], it has six local-optima, two of which are global. This may sound like an unlikely challenge for the primitive optimisation methodologies, but as you can see in Figure 1, the global optima are still fairly accessible due to the nature of the topology.

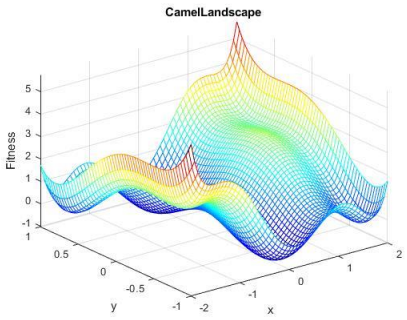


Figure 1 – The Six-Hump Camel Function

The function can be defines as:

$$f(x) = \left(4 - 2.1x^2 + \frac{x^4}{3}\right)x^2 + xy + (-4 + 4y^2)y^2$$

In order to cater for the gradient ascent algorithm, we must then calculate the following partial derivatives.

$$\frac{\partial f}{\partial x} = 2(x^5 - 4.2x^3 + 4x + 0.5y)$$

$$\frac{\partial f}{\partial y} = x + 16y^3 - 8y$$

These can then be used to calculate a gradient vector at the current point which will dictate the direction in which the algorithm will proceed for the next iteration.

4.2.2 Ripple Landscape

I will include the ripple landscape as part of my investigation, as it has local optima which completely encapsulate the global, meaning any algorithm must be able to consistently escape them in order to achieve a reliable success rate regardless of its initial position.

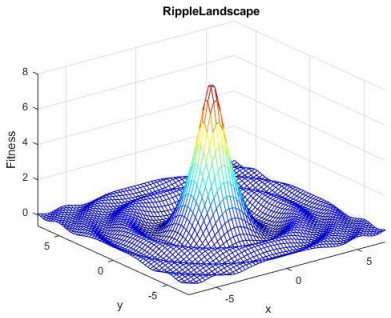


Figure 2 – The Ripple Function Plot

The function can be defined as:

$$f(x) = 8 \times cos\left(\frac{\frac{1}{4}(x^2+y^2)}{x^2+y^2+1}\right)$$

With the associated partial derivatives:

$$\frac{\partial f}{\partial x} = -\frac{4x\left((x^2+y^2+1)\sin\left(\frac{1}{4}(x^2+y^2)\right)+4cos\left(\frac{1}{4}(x^2+y^2)\right)\right)}{(x^2+y^2+1)}$$

$$\frac{\partial f}{\partial y} = -\frac{4y\left((x^2+y^2+1)\sin\left(\frac{1}{4}(x^2+y^2)\right)+4cos\left(\frac{1}{4}(x^2+y^2)\right)\right)}{(x^2+y^2+1)}$$

4.2.3 Modified DeJong Function

This function differs from the others in that it only has one optima. It is a modified version of the *first function of DeJong* [12], in that it's reflected in the *x*-*y* plane to reduce the optima from 4 to 1, as well as being positively translated in the *z* direction. The inclusion of this serves as a contrast to the Ripple Landscape, to offer those that are severely inhibited by suboptimal-local optima to potentially outperform their competition given a fitness landscape with contrasting characteristics.

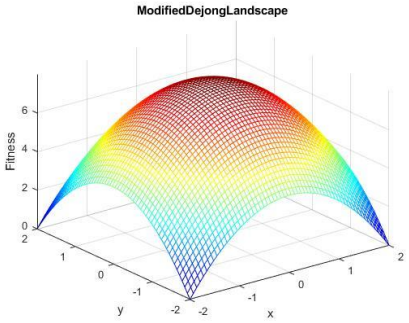


Figure 3 – Our modified DeJong function

The function can be defined as:

$$f(x) = -(x^2+y^2)+8$$

With the associated partial derivatives:

$$\frac{\partial f}{\partial x} = -2x$$

$$\frac{\partial f}{\partial y} = -2y$$

4.2.4 Complex Landscape

This function is unique in that it has large flat areas. It will be interesting to see how this impacts the performance of different algorithms, particularly in combination with the challenge posed by multiple local-optima.

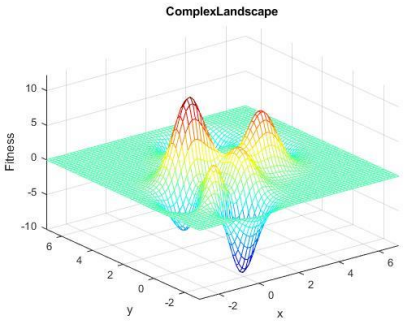


Figure 4 – The complex function plot

The function can be defined by:

$$f(x) = 4(1-x)^2 e^{(-(x^2)-(y+1))} - 15\left(\frac{x}{5} - x^3 - y^5\right) e^{(-x^2-y^2)} - \frac{1}{3} e^{(-(x+1)^2-y^2)} - 2(x-3)^7 - 0.3(y-4)^5 + (y-3)^9 e^{(-(x-3)^2-(y-3)^2)}$$

4.3 Terminating Conditions

In order to more accurately quantify the success of a given algorithm within a continuous fitness landscape, I will classify a significant majority (95%) of the global-optima as a 'good enough' solution, as one would do in a real-life situation. It must be noted that in a real-life application, the global-optima will likely be unknown, however by replicating this, we would be introducing another variable that would influence our results, that being our terminating conditions. Granted these conditions are key to most real-world situations, however they tend to be problem specific. Through the elimination of this extra variable, our results will be more dependent on the fundamentals of the underlying algorithms themselves, as opposed to the time-taken being greatly influenced by the specifics of the chosen conditions, the nature of which would be difficult to keep consistent across the different algorithms.

5. Results

5.1 Initial Primitive Analysis

Although the two more primitive algorithms share a common goal, the underlying discrepancies in their functionality lead to some interesting variation in performance. In many cases, the hillclimber has the capability to reach a height greater than that of the gradient ascent algorithm due to its stochasticity, and resultant ability to escape suboptimal local optima - these seem to be the Achilles heel of our implementation of gradient ascent, so to speak.

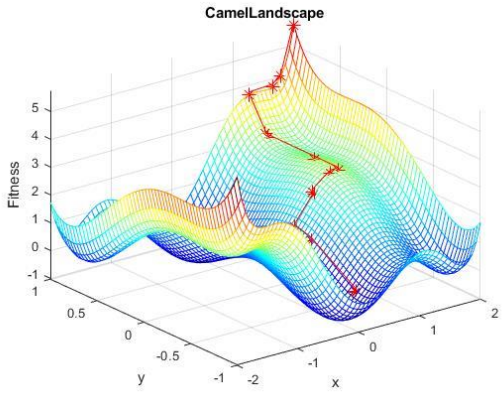


Figure 5.1 – Hill-climber demonstrating its inherent ability to escape local minima.

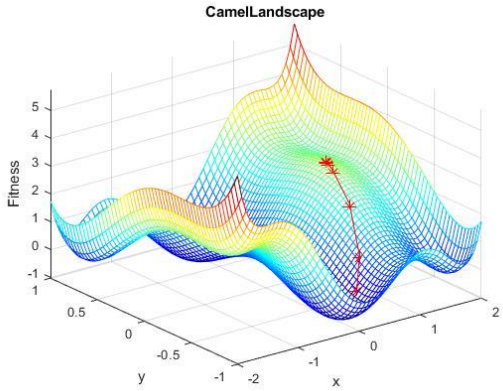
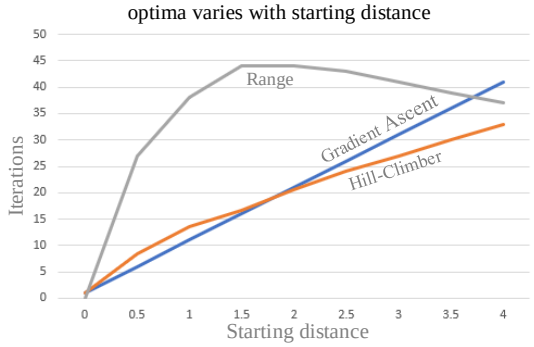


Figure 5.1 – Gradient ascent getting stuck in local minima.

In figures 5.1 and 5.2 above, you can see a side-by-side visualisation of the two algorithms' performance, when initiated from the same starting point. It's easy to see how the success of our gradient descent implementation is highly dependent upon where it is initialised. It must also be noted however, that the behaviour seen to be exhibited in figure 5.2 is not guaranteed - the same stochastic nature that allows for this path to be possible, of course means that there is a chance it won't be found by random mutation. In the cases where the gradient ascent approach is able to achieve global-optima, it will do so more consistently than the hill-climber as a result of its deterministic nature.

With regard to the comparative time required by each algorithm to achieve global-optima (in the cases where this is possible for the gradient ascent), this is somewhat dependent on the starting-distance from it. In the cases where a random starting point is in close proximity of the global optima, the gradient ascent's performance fairly consistently transcends that of the hillclimber. Contrary to this, the hillclimber is at an advantage in the cases where the starting-point is further afield, this can be credited to its ability to cover a much greater distance with each iteration. In figure 6 you can see the mean results from my experimentation over all landscapes, in which each hill-climber data point is averaged from 100 using our default parameters (GA: learningRate = 0.1, HC: maxMutate = 1) to account for its stochastic nature. I must however stress the level of variation existent amongst these, whilst the gradient ascent performed consistently as expected, the amount of iterations required by the hillclimber would vary hugely. The specifics of the amount of variation at each distance can be seen plotted by the line labelled "Range" in figure 6.

Figure 6 – Graph showing how time taken to reach global-optima varies with starting distance



I anticipate that the reason we see the range taper off, and even reduce slightly towards the higher end of the starting distances to be because the minimum amount of iterations required to cover that distance increases, yet the ceiling limiting how many are permitted has remained constant. This results in a narrower window of iteration counts within which a hill-climber instance can achieve the global minimum, and thus a reduced range for those that were successful.

5.2 Systematic Comparison

When testing the two algorithms systematically across a grid of initialization points, spanning the entire fitness landscapes of all 4 that were defined in section 4.2, the fundamental differences between them were exacerbated. They both achieved almost identical success rates when tasked with the Ripple, modified DeJong and Complex functions (the gradient ascent outshining the hill-climber by 2.4% on average), yet the successful instances of the gradient ascent achieved global-optima in only 41%, 37.5% and 71%of the iterations required by the hill-climbers, on the 3 landscapes respectively. When we turn our attention to the 6-Hump Camel Function however, the hill-climbers achieved a success rate of 86%, whilst only 13% of the gradient ascent instances were able to do so.

In section 5.1 we observed the differences in how the two behave when faced with local-optima, but this also goes to demonstrate the influence the nature of these local-optima has. The local-minima in the Ripple landscape inhibited the two equally, whilst those in the Camel Landscape did so significantly more for the gradient ascent - this is due to those in the Ripple landscape being far more isolated, such that neither had the ability to escape. Since gradient ascent's movement is a function of the current gradient at its current point, it struggles in flatter regions, including these lesser-defined local optima. For the purpose of clearly visualising this, employed the use of the Complex Landscape and plotted the concentration of points over all realisations as a colour-map, see figure 7 below.

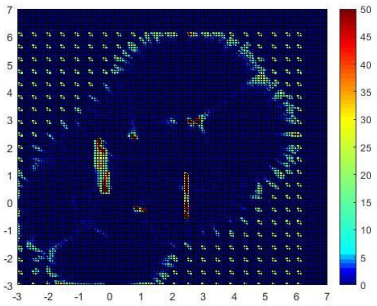


Figure 7.1 – Gradient Ascent colour-map

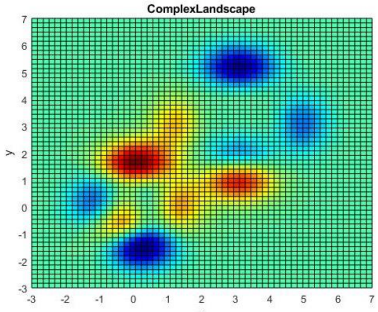


Figure 7.2 – Complex landscape topography

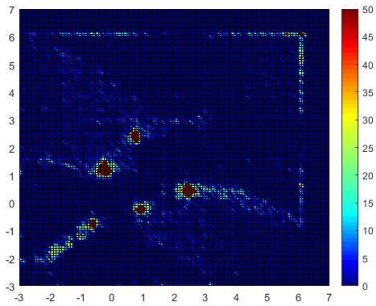


Figure 7.3 – Hill-Climber colourmap

Figure 7.2 is a useful point of reference to visualise what topography lies at any given coordinate, helping us to interpret the results seen in the two neighbouring plots. In figure 7.1 you can clearly see all the points where the algorithm has failed to make any headway, and all 50 iterations have landed on the same spot, leading to the high concentrations we see at each one, this clearly outlines the basins of attraction to each optima.

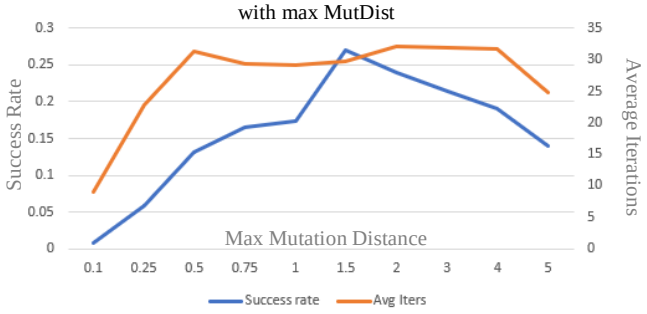
Whilst a high concentration exists on all 5 optima in figure 7.3, when we turn our attention to figure 10.1, we can see that the gradient ascent has high concentrations around 4 of the 5, with an additional point of interest at approximately (3,3) which is not present in figure 7.3. Upon comparison with figure 7.2, we can see that this is due to the surrounding troughs in the landscape. It would be tempting to then call this a suboptimal local-minima, however this would be erroneous because in addition to, and more critically than carrying a fitness of 0, there is a neutral gradient in the negative x direction leading to a genuine local-minima. This is why this particular concentration doesn't exist in figure 10.3, as hill-climbers don't require a gradient for movement.

5.3 Hill-Climber Parameter Tuning

We have now been able to characterise the problems for which each of the two primitive algorithms are best suited. The gradient ascent excels on simpler landscapes, achieving optimal finesses in a fraction of the iterations as the hill-climbers, however it suffers when dealing with more variable landscapes, let down by its inability to escape local-minima, as well as traverse problem spaces with little to no gradient. Of course, there are modifications we could make to ameliorate these issues, such as the implementation of stochastic gradient ascent for example, however for the scope of this investigation we will now be attempting to create a performance comparable to that of an EDA with *minimal* modification, as such, we will progress with the hill-climber.

Figure 8 depicts the effects of varying the maximum mutation distance on the complex landscape. For the purpose of explaining this correlation, I would like to partition the graph in to 3 sections. The lower end of MutDist - where both success rate and average iterations are low, the middle section in which both variables are high, and the upper end of MutDist in which both are once again low.

Figure 8 – How success rate and associated average iterations vary with max MutDist



Firstly, with a low mutation distance, the 'basin of attraction' of the global minima is significantly smaller, with only 50 iterations, only those that are initiated very close to the global-minima stand a reasonable chance of ever making it there, and so the success rate is low as a result, yet those that do succeed do so in few iterations, as there's less distance to cover. Each instance of the algorithm is able to travel further in the mid-range, widening this basin of attraction, as a result, the success rate rises and therefore so does the average iterations – as some have travelled from further afield. As we move towards the upper-end, a mutation distance that is too large effectively increases the search space, the probability of mutating by the small distances required to home in on the global-minima become ever smaller the closer it gets. This results in a reduced success rate, however those that are successful by chance, do so in few iterations due to the large step sizes being taken with each iteration.

5.4 Hill-Climber vs EDAs

Figure 9.1 – Hill-Climber (successful realisation)

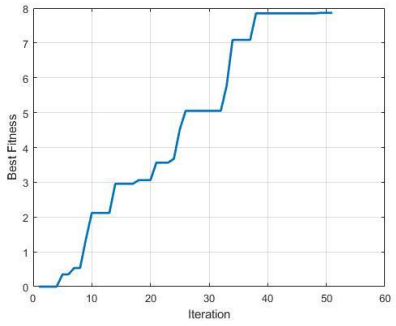


Figure 9.2 – PSO

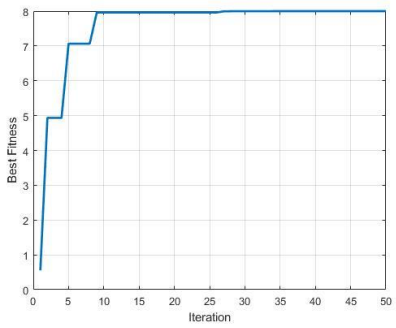
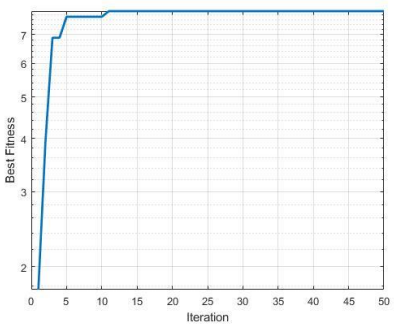


Figure 9.3 – CMA-ES



In Figures 9 you can see the performances of a hill-climber compared to our two EDA's (PSO and CMA-ES) on the Ripple Landscape. It must be noted however that figure 9.1 represents a marginal case for the hill-climber, that only accounts for an average of 15% of its realisations (as was the case to a similar degree on all landscapes with local-optima). In order to achieve a comparable performance, we are going to have to both increase this percentage drastically, as well as improve the rate at which the best fitness increases to more closely echo what we see in figures 9.2 and 9.3.

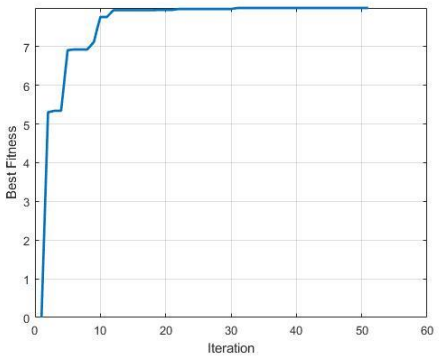
On the modified DeJong landscape, the EDA's still outperformed the hill-climber despite the lack of local-optima, in terms of the time required to achieve an optimal solution. CMA-ES and PSO proved to be far more computationally efficient, accomplishing all iterations in only 48% and 23% (respectively) of the time required by hill-climbers. Although the significance of this may seem limited here when all times were within a second, it's indicative of scalability issues for many real-world applications.

5.5 Bridging the Performance Gap

By allowing for a marginal probability of downwards movements in the hill-climber, (much akin to the strategy employed by simulated annealing [13]) such that the probability of allowing such a movement is inversely proportional to height difference in a non-linear manner, we were able to achieve an average improvement in its success-rate of 10% over all landscapes (excluding the modified DeJong, as it's void of local-minima). Its performance on the modified DeJong function also did not seem to suffer as a result, I anticipated that on such a small landscape, the impact of an occasional downwards movement would fade in to insignificance in comparison to the highly stochastic nature of the algorithm. To test, this I scaled up the *x* and *y* axis domain 5-fold, as well as adjusting the maximum iteration count accordingly, and indeed found that the modified hill-climbers required on average 25.8% more iterations to achieve the same success rate. Since the iterations required to reach an optimal solution is a secondary concern after the success rate, this was a worthwhile trade-off.

The second, and much more key modification I chose to implement was the initialization of a population of hill-climbers, as opposed to an individual realisation. The results of which can be seen in figure 10, and it should be easy to see why the size of the population is a key parameter in all major evolutionary algorithms.

Figure 10 – Cumulative best fitness of a population of 50 hillclimbers



6. Concluding Remarks

After performing a comparative scrutinization of two basic optimization methodologies, through two simple modifications, we were able to ameliorate the performance of the most promising of the two, the hill-climber, to achieve a performance that is more comparable to that of a much more sophisticated EDA, than its vanilla implementation. It must be stressed that there is still a very substantial performance gap between the two, particularly in terms of the efficiency of each, those results achieved by the population of modified hill-climbers in Figure 10, were done so in 8 and 16 times what it took CMA-ES and PSO to achieve the same thing. This brief investigation serves as a proof of concept to demonstrate that the basis for some of the most state-of-the-art evolutionary algorithms can be mimicked by a combination of very simple techniques. In the future I hope to be able to further this line of work by performing an in-depth comparative analysis of the two EDA's that we used as a benchmark here, with the stretch goal of creating a hybrid approach that combines certain characteristics of both in an attempt to negate any relative detriments we may come to discover.

7. Bibliography

[1] P. Morris, "The breakout method for escaping from local minima," in *AAAI-93*, 1993.

[2] M. d. l. M. Deniz Yuret, Dynamic hill climbing: Overcoming the limitations of optimization techniques, Massachusetts Institute of Technology: Numinous Noetics Group, 1993.

[3] M. A. Al-Betar, "β -Hill climbing: an exploratory local search," in *Neural Computing and Applications*, Al-Huson, Irbid, Jordan, Springer London, 2017, p. 153–168.

[4] H. B. J.-M. Renders, "Hybridizing genetic algorithms with hill-climbing methods for global optimization: two possible ways," in *Proceedings of the First IEEE Conference on Evolutionary Computation. IEEE World Congress on Computational Intelligence*, Orlando, FL, USA, 1994.

[5] T. C. M. D. L. a. A. S. S. Alexey V. Gorshkov, “Photon storage in Λ-type optically dense atomic media. IV. Optimal control using gradient ascent,” *Phys. Rev. A,* vol. 77, no. 6, p. 15, 2008.

[6] R. C. E. R. Mark L.Williams, “Deterministic search for relational graph matching,” *Pattern Recognition,* vol. 32, no. 7, pp. 1255-1271, 1999.

[7] P. Larrañaga and J. A. Lozano, Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation, pringer Science & Business Media, 2001.

[8] M. P. Mark Haus, “An introduction and survey of estimation of distribution algorithms,” *Swarm and Evolutionary Computation,* vol. 1, no. 3, pp. 111-128, 2011.

[9] D. G. X. L. K. Sastry, “Towards billion bit optimization via parallel estimation of distribution algorithm, in: Genetic and Evolutionary Computation,” *GECCO-2007,* pp. 577-584, 2007.

[10] “Yarpiz,” Yarpiz, [Online]. Available: https://yarpiz.com/235/ypea108-cma-es. [Accessed Jan 2020].

[11] C. S. Marcin Molga, “Test functions for optimization needs,” Dnipropetrovs’k, Ukraine, 2005.

[12] K. D. Jong, “Learning with genetic algorithms: An overview,” *Machine Learning,* vol. 3, no. 2-3, pp. 121-138, 1988.

[13] C. D. G. J. M. P. V. S. Kirkpatrick, “Optimization by Simulated Annealing,” *Science,* vol. 220, no. 4598, pp. 671-680, 1983.