SCHOOL OF INFORMATICS

An Investigation of the Lion Optimization Algorithm

Liam Perston (S1867389) and Steinar Laenen (S1560045)
Natural Computing (INFR11161) Assignment
November 15, 2018

1 The Lion Optimisation Algorithm and its Inspiration

Many meta-heuristic optimisation algorithms take inspiration from various natural phenomena as a means of constructing an optimisation process. This assignment explores the prospects the Lion Optimisation Algorithm (LOA), proposed by Yazdani and Jola [3], and compares it with the Particle Swarm Optimisation (PSO) algorithm on various well-known benchmark functions.

Lion behaviour is the principle motivation for the development of the algorithm proposed by the authors. It takes into account various factors of the lifestyle of the felines, such as their social structures, solitary and cooperative habits, and survival tendencies in order to create a mechanism for exploring a search space.

The algorithm is population based; it encompasses a number of lions, each which is equivalent to a particle in the PSO algorithm, thereby representing a solution in the search space. At any moment, a lion is either part of a pride (a group of lions), or is a nomad whereby it acts alone. Prides engage in behaviours such as cooperative hunting, mating and roaming in order for exploration to take place. However prides have a focus on exploring specific regions of the search space. Having multiple prides thus ensures diversity in the regions explored. Alternatively, nomad lions only engage in randomly exploring the search space. It is, however, important to note that a lion may switch lifestyles; a male may be driven out of his pride if attacked by a stronger nomad male lion, and females may decide to migrate (based on a specified probability) between a pride and nomad social structure. Nomad lions add a stochastic element, and offer an escape for lions stuck in local optima. Lions remember their best visited position in order for the algorithm to progress from good solutions. Furthermore, the mating of lions aims to bring information exchange between them since their cubs inherit genes of the solutions from both parents. We recommend to look at the appendix (section 9) to get a complete and clearer idea of LOA.

There exists various improvements which would strengthen this algorithm's prospects for optimisation. Each lion has a gender and thus two different behaviours; only females explore new areas of the search space (via hunting), whereas males roam around previously explored areas as a means of exploitation via local search. While this does give a balance between exploration and exploitation, were both genders to encompass both such behaviours, the likelihood of finding better optima would increase since more of the search space would be explored and exploited. Secondly, the Opposition-Based Learning (OBL) method proposed by Tizhoosh [2] employed during hunting means that solutions are approached in a encircling-like manner, instead of directly. While this does serve as a strong method for exploring around a new region, it may however lead the hunter lions to become trapped inside local optima.

It is questionable whether lion behaviour is suitable for finding an good solutions in a search space; the inspiration seems far-fetched and the authors might have focused too much on the specifics of lion behaviour to put a name to the algorithm rather than developing a rigorous searching mechanism. Nonetheless, the main strength behind the algorithm comes from the mechanisms of lions' social behaviour. The swapping of lions between prides, the complex movements of lions as a result of hunting and the presence of nomad lions all contribute to the exploratory factors of the algorithm. Simultaneously, various subroutines, such as the hunting mechanism, offer strong exploitative local searching. Unfortunately, as a result of numerous ambiguities in the authors' explanations of various methods, implementation was tricky. Assuming our interpretation of the authors methods is correct, we expect the algorithm to work as well as is stated in the paper (discussed in sections 3, 4 & 6).

2 Key Formulae

2.1 Hunting

When a subset of female lions in a pride go hunting, they split up into three groups; the center group, which has the highest collective fitness, and a left and right group. The behaviour for the left and right group is governed by the following formula:

$$Hunter' = \begin{cases} rand((2 \times Prey - Hunter), Prey) \\ rand(Prey, (2 \times Prey - Hunter)) \end{cases} \tag{2.1}$$

And for the centre group by:

$$Hunter' = \begin{cases} rand(Hunter, Prey) & Hunter < Prey \\ rand(Prey, Hunter)) & Hunter > Prey \end{cases} \tag{2.2}$$

Where *Hunter* is the position of the hunter lion and *Prey* is the position of the hypothetical prey (which is the average position of all the hunters) and *rand(a,b)* takes a random uniform number between a and b. This form of OBL movement is a proven and effective method for optimization [2]. In essence, the lions encircle the prey and approach it from different directions, beneficial for exploration. If a *Hunter* obtains a stronger fitness while hunting by means of finding a better solution, the *prey* then escapes by the following formula:

$$Prey' = Prey' + rand(0,1) \times PI \times (Prey - Hunter) \tag{2.3}$$

Where PI is the improvement percentage of the hunter fitness. The prey moving away aims to prevent lion convergence.

2.2 Moving to Safe Place

The female lions in the pride that don't go hunting explore all the best visited positions. The direction in they move is decided by a tournament selection strategy. The paper uses a term K_j, which is the number of lions in pride j that have improved their position per iteration. If K_j is large, many lions improved their position, meaning they were far from an optimum. Thus K_j is a good indication for how close lions are to an optimum, and so is used for determining the tournament size. The tournament size for a single pride j is calculated:

$$T_j^{size} = max\left(2, ceil\left(\frac{K_j}{2}\right)\right) \tag{2.4}$$

After female lions select a position to visit, they move towards that position by the following equation:

$$Lion' = Lion + 2D \times rand(0,1) \times R1 + U(-1,1) \times tan(\theta) \times D \times R2 \tag{2.5}$$

Where R1 is the vector from the lions position to the chosen position, and R2 is orthonormal to R1. θ wasn't specified in the paper, but we took it to be a random number between $\frac{\pi}{6}$ and $-\frac{\pi}{6}$ in order to prevent the lion moving out of the search space.

2.3 Roaming around the Search Space

In each iteration, the male pride member and nomad lions (of both genders) roam randomly around the search space in the hope of finding better solutions. The respective subroutines for pride lions and nomads are identical, but roam differently defined spaces. Nomad lions decide to engage in roaming with the following probability:

$$pr_i = 0.1 + min\left(0.5, \frac{Nomad_i - Best_{nomad}}{Best_{nomad}}\right) i = 1,2,3... \tag{2.6}$$

Where Nomad_i and Best_nomad are the fitnesses of the *i*th and best current nomad respectively. Intuitively, this subroutine allows the worst (least fit) nomads to move out of an a poor area of the search space with the aim of moving them to an area with perhaps better prospects for containing an optima.

2.4 Lion Mating

Mating between lions was implemented in the algorithm as a means of improving the solutions represented by the lion over the iterations and to provide an opportunity for information exchange. Per iteration, females mate with a given probability, specified by the paper as 30%. Males are selected randomly, but a pride female may only mate with males in her pride, and nomad females may mate with any other nomad. Mating produces two offspring. The crossover of the parents genes are determined by the following equations:

$$Offspring_j1 = \beta \times FemaleLion_j + (1-\beta) \times \sum \frac{1}{\sum_{i=1}^{NR} S_i} \times MaleLion_j^i \times S_i \tag{2.7}$$

$$Offspring_j2 = (1-\beta) \times FemaleLion_j + \beta \times \sum \frac{1}{\sum_{i=1}^{NR} S_i} \times MaleLion_j^i \times S_i \tag{2.8}$$

Where $\text{Offspring}_\text{j}$1 and 2 are the position of the resulting two cubs in the j^{th} dimension, $\text{Female Lion}_\text{j}$ and $\text{Male Lion}_\text{j}$ are the position vectors of the respective parents, β is a random Gaussian variable with mean 0.5 and standard devation 0.1 and S_i is equal to 1 if the underlying male has been selected for mating, 0 otherwise. Effectively, these equations take the average male position of those who have been selected, and add it to the female position with the β variable included for nomalisation. The resulting position vectors are given to the ensuing cubs. Furthermore, each gene (dimension) of a cub can mutate.

3 Results Presented in the Paper

To test their algorithm, the authors evaluated LOA on 30 benchmark functions as specified in the CEC 2014 [1] competition. And is then compared to, at that time, 6 popular metaheuristic algorithms Invasive weed optimization (IWO) algorithm, biogeography-based optimization (BBO) algorithm, gravitational search algorithm (GSA) , hunting search (HuS) algorithm , bat algorithm (BA), Water wave optimization (WWO) algorithm. Why specifically these 6, and not for instance PSO, is not clarified by the authors.

In each run, the number of lions is set to 50, and they only evaluate the function in 30 dimensions. The maximum number of function evaluations (150,000) is set as a stopping condition. For each function, they evaluate the maximum score, minimum score, median score and the standard deviation.

On unimodal functions, the algorithm performs better on most of the algorithms that it is compared to. On multimodal functions it also performs much better than the algorithms that it is compared to.

On the group of hybrid functions, again, the authors emphasize that their algorithm performs better than the six other algorithms.

On the group of composition functions, LOA does exhibit weaker performance than some of the other algorithms.

The authors conclude that the algorithm performs better than other popular metaheuristic algorithms. However, it must be noted again that they never specified why they compared their algorithm to these six MHOs specifically. Moreover, they do report much better results on must functions compared to the other 6, but they do not specify at what computational/temporal cost. If LOA is drastically slower, the question arises whether the high fitness weighs up against the high cost.

Furthermore, the authors could have analyzed the various sub-components of their algorithm to see how the behaviour changes when different subroutines are switched on or off. In this way, one could find the efficacy of the different parts of the algorithm, and detect redundant subroutines.

The authors do observe that in some cases the maximum/minimum value found of the algorithm is lower than the maximum/minimum of another MHO, or that the standard deviation is lower. However the paper does not provide any explanation as to what that could mean or why that perhaps is the case.

Overall, as the results are presented in the paper in its present form, it is unclear whether the authors picked the comparative MHOs because they knew they performed worse, or whether they are actually widely used. Secondly, the discussion does not go much further than just stating on which functions the score of LOA is better than other optimizers.

4 Reproduction

The length of the algorithm made it cumbersome to reproduce. Whereas other standard Meta Heuristic Optimizers (MHO) such as PSO are relatively simple to implement and vectorize, that is certainly not the case for LOA. To make coding simpler, we decided to implement the algorithm in Python, as this allowed us to object orientate the algorithm. The trade-off is that our implementation became significantly slow, such that we weren't able to test out implementation of some of the more complex benchmark functions in the paper.

Poor written English, ambiguous explanations and incomplete descriptions forced us to infer some parameters ourselves. For instance, the authors refer to distance multiple times (see for example equation 2.5), however, they never explicitly defined the distance metric they used. Initially, we assumed the distance metric to be the Euclidian distance. However, our algorithm quickly diverged out of bounds. We ended up setting the distance to be a unit of movement in the Euclidean distance space (Euclidian distance divided by maximum Euclidian distance in the space).

Furthermore, when female lions move to a safe place (section 2.2), the authors do not provide the value of θ. In a later routine, they set θ to be a random uniform number between $-\frac{\pi}{6}$ and $\frac{\pi}{6}$, so we copied that parameter range.

After mating takes place in each pride, the weakest males are driven out by stronger cubs. The authors did not specify how many lions should be driven out of the pride, the excess number of male lions in a pride, based on the maximum number of allowed lions (*parameter male_percentage*), to ensure the lion population does not explode.

Lastly, the authors of the papers sometimes refer to "the stronger lion", but it is unclear whether the stronger lion has a better *current* fitness or a better *best-ever* fitness. In each of these cases, we decided to infer for ourselves what was more logical.

The unimodal benchmark functions that we used are the shifted high conditioned elliptical (**f1**), shifted bent cigar function (**f2**) and the shifted cigar function (**f3**) as defined in [1] (see appendix section 9). The multimodal benchmark functions we test are the shifted Rosenbrock function (**f4**) and the shifted Rastrigin function (**f8**). As discussed in section 3, the algorithm does quite well on all these functions. In **f1** LOAs median score beats all the algorithms that it is compared to, in **f2** it loses to one and in **f3** it loses to two. So the algorithm has some qualitative form of 'betterness' measure on the unimodal functions. We picked these two multimodal functions (**f4** & **f8**) because LOA performs exceptionally well on them, and because they are computationally cheaper than other benchmark functions.

For simplicity, we also decided not to rotate the function as is done in the paper, because this also increases computation time.

5 PSO vs. LOA

Out of the standard canonical MHOs with the PSO, Genetical Algorithm, and Ant Colony Optimisation, it seemed to us that PSO is most similar to LOA. This mainly stems from the fact the the search space tested is continuous. This is not generally the case for using the other two flagship MHOs since they are more specialised towards combinatorial problems or discrete search space problems (although they can be translated into continuous spaces).

We can draw the following comparisons between the PSO and LOA algorithms:

Both effectively involve particles in a search space. In the PSO, each sample of the population is a particle. These particle move around the search space guided by forces from the other particles. In the LOA, each lion can be represented as a point in the search space. The lions thus have a defined evaluation function in the same way each particle can be evaluated, and as in the PSO, the subroutines of the algorithm which guided the movement of the lions is influenced by the positions of other lions in the search space (take when the male resident lion of a pride moves towards the areas found by other member of his pride for further exploration in the local region). The various attributes of the lion such as gender, social structure are physically irrelevant in the search space; they merely guide the directions in which the lions move and do not change the fact that each lions is represented by vector in the search space.

Next, both particles and lions remember their best visited value. In every iteration of the PSO algorithm, each particle has a force enacting upon it which draws it towards its best remembered position (or global best position). One of the effects of this is to try and encourage futher exploration of that particular local area. In the LOA,

the best visited positions of the lions making up a pride are marked as the *territory* of the pride. The significance of this is that the pride's resident male lion explores a selected area of the territory while the female pride member undertake hunting. This is done, as in the PSO, to try and explore the local regions further in the hope of finding a better solution.

Next, the LOA is essentially just several particle swarms that exchange information, with some nomad particles in addition. In the PSO, the swarm exchanges information between the particles in that one of the factors influencing the direction in which each particles move is towards the best position found by all particles in the swarm. One of the aims of this effect is that, after a certain amount of time, the particles will converge on the best globally found area. If we represent each pride as a swarm, it can be seen that the members of the pride are drawn towards good solutions found by the other pride member in the *moving towards a safe place* and *pride resident male roaming* subroutines. It is important to note that these tendencies are with the intention of further exploring the area, not necessarily to converge upon it. However in the LOA the nomad lions serve as a way for the pride members to switch lifestyle in the hope that they will be drawn out of local optima, since other nomad females may be added to the pride and thus bring new information about the state space to the rest of the pride members.

Next, the PSO is easier to vectorize, whereas LOA isn't (at least we did not manage to do in the available time), which means that our implementation of PSO runs much faster. The representation of the particles in the search space can be implemented as a matrix of vectors referring to their position and velocity. As a result, the manipulation of the particles over the algorithm iterations is relatively straightforward and can even be parallelised (on a per iteration basis). This, however, is certainly not the case for the LOA. The complexities of the various subroutines of the algorithm often lead to dependencies of information within each iteration. For example in order for a female nomad lion to migrate into a pride, there must be spaces in the pride because of previous female pride members leaving. These subroutines take place within each iteration. Therefore as a result of such intra-iteration dependencies, our implementation did not parallelise the algorithm, which overall resulted in excessive run times when compared with our PSO implementation on equivalent evaluation benchmarks (See table 6.5 in the results section).

Finally, the PSO algorithm is much simpler than the LOA. There exist no complex social structures, behaviour patterns and hunting techniques. The complex design of the LOA meant our implementation was long-winded, requiring almost 1000 lines of code to be written. This compares with the PSO taking up approximately 100 lines of Python. It can be seen in the following sections whether such complexities actually added any significant value to the algorithm's prospect as an optimization mechanism.

6 Results

In this section we present the results of our evaluation of LOA. As was done in [3], we set the termination criterium for PSO and LOA to be 150.000 function evaluations. Since the paper sets the population size to 50, and for each iteration every lion evaluates its position at least once, we generously set the number of iterations to be 3000. To obtain a more accurate comparison we set the PSO population size to 50 as well and run the PSO for the same number of iterations.

For each benchmark function we run the algorithm 60 times, and obtain the best values found (ever). In the result tables (table 6.3 and 6.4) we report the maximum score found over all runs, the minimum score, the median score and the standard deviation of all scores)

Furthermore we also report the experimentally obtained running time of the algorithm (table 6.5). Although quantitatively this number depends on factors such as the type of processor used, programming language and other factors, we hope to show that the qualitative difference becomes clear.

6.1 Preconditions

Table 6.1: Function Parameters

MHO	Parameter	Value	Parameter	Value
LOA	Number of prides	4	Sex Rate	0.8
	% nomad	0.2	Mating prob.	0.3
	% roaming	0.2	Immigration rate	0.4
	Mutate prob.	0.2		
PSO	$\alpha 1$	0.5	$\alpha 2$	0.5
	ω	0.94		
Both	particles	50	iterations	3000
	runs	60		

Parameters used for LOA and PSO. LOA parameters were obtained from [3]. PSO parameters were obtained from brief experimentation.

Table 6.2: Tested Benchmarks

Type	ID	Function	f*
Unimodal	**f1**	Shift High Cond.Elliptic	100
	f2	Shift Bent Cigar	200
	f3	Shift Discus	300
Multimodal	**f4**	Shift Rosenbrock	400
	f8	Shift Rastrigin	800

Functions used for testing LOA and PSO. f is global minimum. Function definitions can be found in the appendix (section 9) See section 4 why these were chosen.*

Table 6.3: Results Unimodal

		PSO	LOA 1	LOA 2
f1	max	1.00e+02	3.34e+08	3.90e+05
	min	1.00e+02	1.11e+08	2.43e+04
	median	1.00e+02	1.95e+08	1.45e+05
	std	8.41e-14	4.65e+07	1.32e+05
f2	max	2.00e+02	1.60e+10	1.85e+03
	min	2.00e+02	5.49e+09	2.00e+02
	median	2.00e+02	9.89e+09	6.83e+02
	std	3.09e-14	2.06e+09	4.96e+02
f3	max	3.00e+02	1.13e+05	1.30e+03
	min	3.00e+02	4.31e+04	3.00e+02
	median	3.00e+02	7.13e+04	5.29e+02
	std	4.58e-14	1.55e+04	3.20e+02

Results of the PSO and LOA of the experiments that were run. Hyper parameters that were used are specified in table 6.1. LOA 1 column are the results of our implementation of LOA. LOA 2 are the results as reported in [3].

For the unimodal functions, PSO outperforms both our (LOA 1) implementation and [3]'s (LOA 2) implementation of LOA. PSO finds the global optimum in each run. Further we can notice that our implementation performs worse than the original implementation in the paper.

Table 6.4: Results Multimodal

		PSO	**LOA 1**	**LOA 2**
f4	max	8.94e02	1.67e+03	5.45e02
	min	5.33e02	7.67e+02	4.00e02
	median	7.09e02	1.11e+03	4.26e02
	std	8.38e01	2.11e+02	4.81e01
f8	max	8.60e03	4.75e+03	8.11e02
	min	4.38e03	2.75e+03	8.00e02
	median	6.14e03	3.38e+03	8.02e02
	std	9.85e02	4.06e+02	3.81e00

Results of PSO and both implementations of LOA on the multimodal benchmark functions. Hyperparameters used are specified in table 6.1. LOA 1 column are the results of our implementation of LOA. LOA 2 column are the results as presented in [3]

For multimodal functions, the LOA 2 implementation outperforms the PSO implementation. This result was not reported in the paper. The LOA 1 implementation performs worse than bot PSO and LOA 2 on **f4**. On **f8**, LOA 1 performs better than PSO, but still worse than LOA 2.

Poor performance on the unimodal functions of our algorithm indicates that the exploratory component of this algorithm is too large to find a global optimum. However, in the multimodal case it seems to escape local optima more easily and find better values than the PSO.

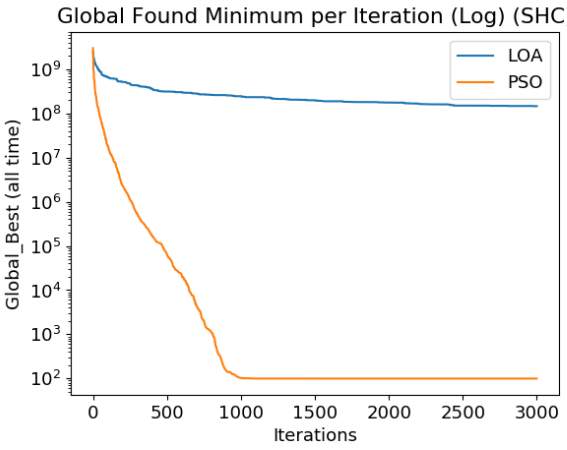

Figure 1 - LOA compared to PSO on the average global best found fitness per iteration on **f1** *Plots for* ***f2*** *and* ***f3*** *are qualitatively similar.*

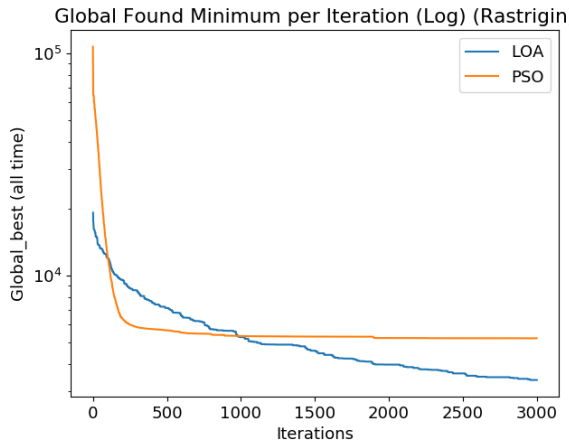

Figure 2 - LOA compared to PSO on the average global best found fitness per iteration on ***f8****. Here we see that LOA performs better than PSO. PSO gets stuck in a local optimum, whereas LOA is still decreasing.*

We can see that even though our implementation of LOA performs very poorly on unimodal functions, it does seem to perform better than PSO on multimodal functions. This is likely due to the strong exploratory component of the algorithm.

Table 6.5: Runtime per iteration

MHO	Dimension	Mean	std
PSO	30	1.46e-04	1.84e-05
	40	1.45e-04	1.09e-05
	50	1.98e-04	2.00e-05
	100	3.90e-04	4.31e-05
LOA	30	1.30e-01	3.59e-02
	40	1.38e-01	2.56e-02
	50	1.88e-01	4.23e-02
	100	3.28e-01	8.72e-02

Time per iteration for LOA and PSO evaluated on ***f1****. Average time was taken over 3000 iterations. Parameters used are identical to table 6.1. Only the dimension is varied.*

In Table 6.5 we see that the time it takes to perform one iteration of the algorithm is 3 orders of magnitude larger than one iteration of PSO. This limits this review's ability to perform any extensive optimization on parameters that have to be inferred from the paper, and thus might result in worse performance of this paper's implementation.

7 Discussion

From the results of our implementation presented in the section above, it can be seen that our implementation performs significantly worse than both that of the author who proposed the algorithm, and the PSO. There are various factors behind this.

Firstly take the complexity of the algorithm coupled with the pervasive ambiguity of the paper. It is likely that our implementation did not exactly match that undertake by the authors, even though we did our best to infer from what was written and used logical judgments when faced with our own choices. As a result of this, our algorithm was likely more prone to making mistakes.

Secondly, take our implementation decisions. We thought it best to object-orientate our implementation to make the development process easier, and to aid with debugging. This did mean that the runtime of the algorithm was unfortunately increased for the evaluation functions. While this did not directly affect the algorithm performance in terms of accuracy, long run times mean that experimenting with different implementations was tricky, and limited in what we could attempt. Indirectly this lead us to develop what effectively is a sub-par implementation. Secondly, it meant that parameters which were not explicitly defined in the paper had to be determined using common sense and logical judgment. The values chosen were likely not optimal for running the algorithm, and thus we achieved a worse performance. Were the algorithm faster to run, we would have made an attempt to optimise the parameters in the hope of improving its performance.

The only benchmark function that LOA performs better on than our implementation of PSO is **f8**. **f8** contains many local optima, and it seems that LOAs exploratory components seem to escape them, whereas PSO doesn't (figure 2).

One of the improvements which we could make would be to change the searching process of the algorithm, and incorporate some features to make it more similar to the PSO. For example, the roaming of nomad lions in our implementation is done randomly. Nomads move to a completely random position in the search space with no heuristics. This is a poor decision choice since it is very likely than the new position is no closer to an optima than the current. Instead, we could include a tendency for the lion to move towards it best visited position, as particles do in the PSO. While this does not guarantee finding a good solution in the process, the superior performance of the PSO means that it would increase the chances of doing so. If more time were available, we would have made an attempt to decrease the algorithm runtime as far as possible. This could be achieved by parallelising the various steps in each iterations which contained no dependencies. Also we could have tried to represent the populations as a matrix of vectors. The resulting manipulations of this over the algorithms would not only be far more memory-efficient but also faster, since object orientation encoding induces overheads in runtime. This however, would have been challenging given the underlying complexity and intricacies of the LOA, and would have lengthened the development process. Finally, we could have implemented our programme in an alternative language to Python, e.g. C or C++. Given its interpreted nature of runtime, coupled with its single threaded usage, Python operates much more slowly than other mainstream languages Furthermore, these languages offer support for parallelisation via various libraries such as OpenMP. Such an approach would further decrease runtime.

Were we to have achieved faster runtimes by changing our implementation, an parameter optimisation would become feasible, and thus the performance of the underlying algorithm would likely increase.

8 Conclusion

The LOA presented by the authors took inspiration from the behavioural patterns of Lions. Since various naturally inspire meta-heuristic optimisers have been shown to work with strong performance in many applications, it was interesting to explore whether incorporating such behaviours in an algorithm proved equally stellar results.

As we have seen over the course of this investigation, this is unfortunately not the case. Even though there have been various limitations which constrained our investigation, such as the pervasive ambiguities in the paper, the complexities of the algorithm leading to a challenging reproduction, and a limit in the computational power available for our development and testing, we still maintain that this algorithm does not serve as legitimate competition to the main canonical MHOs.

Our results are testament to this, we could not reproduce the results of the authors, and even the author results do not provide to match that of a standard PSO implementation on the same unimodal problems. LOA does perform better than PSO on multimodal problems. While it is true that various aspect of the algorithm aim to give a strength in both and balance between exploratory and exploitative action via the various subroutines, it so happens that these did not prove fruitful. We therefore conclude that the simplicity of the PSO algorithm, combined with the fact that its theoretical aspects have been reasonably well explored in academia (eg. bounds on parameters), make it superior in various optimisation problems. While we credit the authors for taking inspiration from nature in and developing such a creative and dexterous optimisation mechanism, we do not recommend its implementation as a means of finding optimal solutions within a search space.

References

[1] P. N. Suganthan, N. Hansen, J. J. Liang, K. Deb, Y. P. Chen, A. Auger, and S. Tiwari. Problem Definitions and Evaluation Criteria for the CEC 2014 Special Session and Competition on Single Objective Real-Parameter Numerical Optimization. Technical Report May, 2014.

[2] Hamid R. Tizhoosh. Opposition-based learning: A new scheme for machine intelligence. In *Proceedings of the International Conference on Computational Intelligence for Modelling, Control and Automation and International Conference on Intelligent Agents, Web Technologies and Internet Commerce Vol-1 (CIMCA-IAWTIC'06) - Volume 01*, CIMCA '05, pages 695–701, Washington, DC, USA, 2005. IEEE Computer Society.

[3] Maziar Yazdani and Fariborz Jolai. Lion Optimization Algorithm (LOA): A nature-inspired metaheuristic algorithm. *Journal of Computational Design and Engineering*, 3(1):24–36, 2016.

9 Appendix

9.1 Evaluated Benchmark Functions

$$f_1(\mathbf{x}) = \sum_{i=1}^{D} (10^6)^{\frac{i-1}{D-1}} \mathbf{x}_i^2 \tag{9.1}$$

$$f_2(\mathbf{x}) = x_1^2 + 10^6 \sum_{i=2}^{D} x_i^2 \tag{9.2}$$

$$f_3(\mathbf{x}) = 10^6 x_1^2 + \sum_{i=2}^{D} x_i^2 \tag{9.3}$$

$$f_4(\mathbf{x}) = \sum_{i=1}^{D-1} (100(x_i^2 - x_{i+1})^2 + (x_i - 1)^2) \tag{9.4}$$

$$f_8(\mathbf{x}) = \sum_{i=1}^{D} (x_i^2 - 10cos(2\pi x_i) + 10) \tag{9.5}$$

9.2 Pseudocode for LOA. Figure taken from [3]

Lion Optimization Algorithm pseudo code

1. Generate random sample of Lions N_{pop}(N_{pop} is number of initial population).
2. Initiate prides and nomad lions

   i. Randomly select %N (Percent of lions that are nomad) of initial population as nomad lion. Partition remained lions into P (P is number of prides) prides randomly, and formed each pride's territory.
   ii. In each pride %S (Sex rate) of entire population are known as females and the rest as males. This rate in nomad lions is inversed.

3. For each pride do

   i. Some randomly selected female lion go hunting.
   ii. Each of remained female lion in pride go toward one of the best selected position from territory.
   iii. In pride, for each resident male; %R (Roaming percent) of territory randomly are selected and checked.
   %Ma (Mating probability) of females in pride mate with one or several resident male. → *New cubs become mature.*
   iv. Weakest male drive out from pride and become nomad.

4. For Nomad do

   i. Nomad lion (both male and female) moving randomly in search space.
   %Ma (Mating probability) of nomad Female mate with one of the best nomad male. → *New cubs become mature.*
   ii. Prides randomly attacked by nomad male.

5. For each pride do

   i. Some female with I rate ((Immigrate rate)) immigrate from pride and become nomad.

6. Do

   i. First, based on their fitness value each gender of the nomad lions are sorted. After that, the best females among them are selected and distributed to prides filling empty places of migrated females.
   ii. With respect to the maximum permitted number of each gender, nomad lions with the least fitness value will be removed.

If termination criterion is not satisfied, then go to step 3