Seeker Algorithm

Nicolae Puica
nicolaerazvan.puica@studenti.unitn.it

July 2020

Abstract

Seeker Algorithm (SeekA) is a single-agent metaheuristic algorithm that exploits the Lévy Flights steps in order to make progress. In this work, the SeekA is simulated in a set of test functions in two-dimensional space and compared with three techniques. Results illustrate that with just one parameter tuning for the Lévy distribution the algorithm performs better in most of the cases.

1 Introduction

An interesting area in nature is animal foraging. The explanation of the strategies that an organism makes in order to search for food, has become of greater interest in the environmentalist, biology, and mathematics fields. Lévy flights, or walks, are characterized by rare but extremely long 'step' lengths [1]. This approach, as opposed to Brownian motion, encourages the exploration. While Brownian motion focuses on a restricted search environment, Lévy walker occasionally takes long jumps to new territory [2][3]. As an example in nature [4], predators living in small seas will move just a little in order to find prey, simulating a Brownian walk. Whereas predators living in oceans are forced to make larger moves in order to find prey, simulating a Lévy flight walk.

More often, metaheuristic algorithms focus on two different characteristics: intensification and diversification [5]. The first aims on searching at the local level by seeking around the best solution found until now, while diversification aims at searching at the global level efficiently exploring the search space.

In this work, Lévy Flights approach is exploited in order to find better positions. Section 2 will provide a deep understanding of the technique. Section 3 presents the whole algorithm procedure and how it works, while section 4 is entirely dedicated on the experiments. The latter provides the simulation results and discussion about them.

2 Lévy Flights

Mathematically, Levy flights [6] [7] [8] is a class of non-Gaussian random processes whose random walks are sampled with the probability density function (PDF) $L(s)$ decaying at large s. Often it is represented in terms of a simple power-law formula $L(s) \sim |s|^{-1-\beta}$ where $0 < \beta \leq 2$ is an index. A version of Levy distribution can be defined as following:

$$L(s, \gamma, \mu) = \begin{cases} \sqrt{\frac{\gamma}{2\pi}} \exp -\frac{\gamma}{2(s-\mu)}] \frac{1}{(s-\mu)^{3/2}} & \text{if } 0 < \mu < s < \infty \\ 0 & \text{if } s \leq 0 \end{cases} \tag{1}$$

where μ parameter is location or shift parameter, $\gamma > 0$ is scale (controls the scale of distribution) parameter.

$$F(k) = \exp[-\alpha|k|^{\beta}], \quad 0 < \beta \leq 2 \tag{2}$$

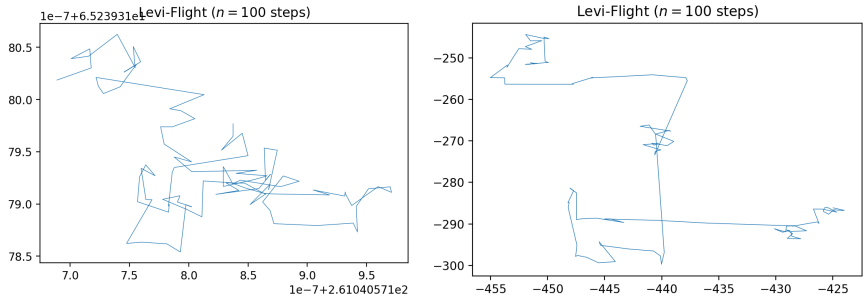
where α is a parameter within $[-1, 1]$ interval and known as scale factor. For β equal to 1, the equation becomes a Cauchy probability distribution.

$$F(k) = \exp[-\alpha|k|] \tag{3}$$

Another particular case is for β equal to 2, the distribution correspond to Gaussian distribution.

$$F(k) = \exp[-\alpha k^2] \tag{4}$$

The parameters α, β, γ and μ constitute the main elements to determine the distribution [9]. The parameter β determines the shape of the probability distribution, especially in the tail regions. Therefore, smaller β parameter causes the distribution to make longer jumps caused by longer tails, vice versa, bigger value makes shorter jumps. The sign of the scale parameter α indicates the slope direction, positive to the right, and negative to the left. When α is equal to 0, the distribution is symmetric. The last two parameters, width γ and the shift μ, determine the peaks of the distribution.



(a) $\beta = 2$ (b) $\beta = 1.5$

Figure 1: Plots of two different walks from a Lévy Flight search with $\beta = 2$ and $\beta = 1.5$ in a [-512, 512] domain

In Figure 1 can be observed how the walk is affected by changing the β parameter. The first, with a high β makes little steps sampled from the Lévy distribution, while the second plot makes in evidence the long jumps that explore more the search space.

3 Seeker Algorithm

Seeker Algorithm is a single-agent technique based on Lévy flights movement. More precisely, the agent exploits the steps to move in a better position until eventually the global minimum is reached. At each iteration, the agent will move accordingly to the following formula:

$$X_n = X_c + \alpha_1 \oplus Levy(\lambda) \tag{5}$$

where X_n is the next point, X_c is the current point and $\alpha_1 > 0$ is the scale step [10]. The above equation is the stochastic equation for the random walk. A random walk that is a Markov chain [11] where the next status/location only depends on the current location. The product $\oplus$ means entrywise multiplications. The Lévy flights provide a random walk, whereas the random step length is drawn from a Lévy distribution

$$Levy \sim u = t^{-\lambda}, \quad 1 < \lambda \leq 3 \tag{6}$$

which has an infinite variance with an infinite mean [12].

In the case of Seeker Algorithm, the agent will move to point X_n only if a better position is reached. If it is not the case, a second shot is made in the other direction starting yet from the current point:

$$X_{n'} = X_c - \alpha_1 \oplus Levy(\lambda). \tag{7}$$

Initially, α_1 has a large value in order to explore the objective function. At each iteration, α_1 will decrease by a certain percentage. This process is useful for a local exploration leading the agent towards the minimum of the respective basin. After a certain number of iterations, the value of α_1 will be restored to the initial one, encouraging the exploration once again. The latter mechanism is implemented by using again probability. With a small percentage (generally 1%) α_1 will return to its default value. Its initial value depends on the objective function bounds:

$$\alpha_1 = \max|lb| + \max|ub| \tag{8}$$

where lb and ub are respectively the lower and upper bounds.

The whole algorithm can be summarized in the following lines of code:

Algorithm 1: Seeker Algorithm

```
Generate the agent according to a uniform distribution
while (t < Max Generation) or (stop criterion) do
    make a Lévy flights step;
    if improvement then
        new point becomes the new record;
    else
        make the step in the other direction and check if improves;
    decrease α1 by 10%;
    with probability 1% restore α1 to default;
```

4 Experiments

4.1 Settings

The experiments were performed by comparing Seeker Algorithm with other three conventional techniques: simulated annealing (SA) [13][14][15], genetic algorithm (GA) [16] and Particle swarm optimization (PSO) [17]. The maximum evaluation number was 200.000 for all functions. All the algorithms are tested on objective functions with 2 dimensions. Other specifications are given below:

4.1.1 SA

The initial temperature of the algorithm is 100. The default temperature function used by SA is the temperature at any given step .95 times the temperature at the previous step [18]. This causes the temperature to go down slowly at first but ultimately get cooler faster.

4.1.2 GA

The population size adopted for the algorithm is 50. The single point crossover operation is 0.8. The mutation rate was set to 0.01. Stochastic uniform sampling was used for the selection phase [19].

4.1.3 PSO

The main parameters for the PSO are the weights that regulate personal and population relation. These weights were set both to 1.49. The adaptive inertia was set in the range [0.1,1.1] that are respectively the lower and upper bounds [20][21]. The swarm size was set to 30.

4.1.4 SeekA

In the experiments, just a parameter is chosen manually. This parameter is β. According to Xin-She Yang in the CuckooSearch (CS) Algorithm [10], the β

parameter for Levy-Flight is equal to 1.4.

4.2 Benchmark functions

In order to test the performance of the algorithms, it is important to make experiments on a different kind of benchmark function [22]. For this reason, the experiments were done on 10 different bounded objective functions with different characteristics such as unimodal, multimodal, separable, and non-separable. A function is multimodal if it has more than one local minima. Here the algorithm should explore, in order to escape from a local minimum. Another group of functions is the separable/non-separable functions. A p-variable separable function can be expressed as the sum of p functions of one variable.

Some functions are characterized by a small global minimum compared to the whole search space (Michalewicz). Some others, have the global minimum close to many local minima (Rastrigin) or even sitting on the bounds (Eggholder). For more details about the objective functions see Table 1.

No	Function	C	Range	Formulation
1	Goldstein	MN	[-2,2]	$f(x,y) = [1 + (x+y+1)^2(19 - 14x + 3x^2 - 14y + 6xy + 3y^2)]$ $[30 + (2x-3y)^2(18 - 32x + 12x^2 + 4y - 36xy + 27y^2)]$
2	Michalewicz	MS	[0,π]	$f(x) = -\sum_{i=1}^{d} \sin(x_i) \sin^{2m}\left(\frac{ix_i^2}{\pi}\right)$
3	Schwefel	MS	[-500,500]	$f(\mathbf{x}) = f(x_1, x_2, ..., x_n) = 418.9829d - \sum_{i=1}^{n} x_i sin(\sqrt{\lvert x_i \rvert})$
4	Sphere	US	[-5.12,512]	$f(\mathbf{x}) = f(x_1, x_2, ..., x_n) = \sum_{i=1}^{n} x_i^2$
5	Rastrigin	MS	[-5.12,5.12]	$f(x,y) = 10n + \sum_{i=1}^{n}(x_i^2 - 10cos(2\pi x_i))$
6	Rosenbrock	UN	[-30,30]	$f(x,y) = \sum_{i=1}^{n}[b(x_{i+1} - x_i^2)^2 + (a - x_i)^2]$
7	Zakharov	UN	[-5,10]	$f(\mathbf{x}) = f(x_1, ..., x_n) = \sum_{i=1}^{n} x_i^2 + (\sum_{i=1}^{n} 0.5ix_i)^2 + (\sum_{i=1}^{n} 0.5ix_i)^4$
8	Ackley	MN	[-32.768,32.768]	$f(\mathbf{x}) = f(x_1, ..., x_n) = -a.exp(-b\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}) +$ $-exp(\frac{1}{n}\sum_{i=1}^{n} cos(cx_i)) + a + exp(1)$ $a = 20, b = 0.2, c = 2\pi$
9	Eggholder	MS	[-512,512]	$f(x,y) = -(y+47)\sin\sqrt{\lvert \frac{x}{2} + (y+47) \rvert} - x\sin\sqrt{\lvert x - (y+47) \rvert}$
10	Dixon-Price	UN	[-10,10]	$f(\mathbf{x}) = (\mathbf{x}_1 - 1)^2 + \sum_{i=1}^{n} i(2\mathbf{x}_i^2 - \mathbf{x}_{i-1})^2$

Table 1: Optimization test functions used in the experiments. C: Characteristic, U: Unimodal, M: Multimodal, S: Separable, N: Non-Separable

4.3 Results

In the experiments, the SA, GA, PSO, and SeekA were compared on different benchmark functions described previously. Each experiment was repeated 100 times for each algorithm. It is a wide number that leads to a better understanding of the performance. The stopping criterion tolerance from the global minimum was set to 1e-5. The mean best values, standard deviations, and standard errors of means are given in Tables 2.

By looking at the results, Seeker Algorithm has most of the time 0 as standard deviation and standard error of the mean. This means that most of the time the algorithm converges to the global minimum. Furthermore, Seeker Algorithm outperforms over the other algorithms in several objective functions as Schwefel, Eggholder, and Rastrigin. Another important aspect of optimization

No	Range	Function	Min		SA	GA	PSO	SeekA
1	[-2,2]	Goldstein	3	Mean	3.000001	3.810000	3	3.000004
				Std	0.000010	4.629058	0	0.000003
				SEM	0.000001	0.462906	0	0
2	$[0,\pi]$	Michalewicz	-1.8013	Mean	-1.792504	-1.801302	-1.801302	-1.801296
				Std	0.044150	0.000001	0.000001	0.000005
				SEM	0.004415	0	0	0
3	[-500, 500]	Schwefel	0	Mean	379.477016	16.581392	36.715909	0.000004
				Std	157.781261	41.303603	55.052886	0.000003
				SEM	15.778126	4.13036	5.505289	0
4	[-5.12,5.12]	Sphere	0	Mean	0.000022	0	0	0.000005
				Std	0.000058	0	0	0.000003
				SEM	0.000006	0	0	0
5	[-5.12,5.12]	Rastrigin	0	Mean	0.592625	0.328337	0.059707	0.000004
				Std	0.603423	0.510965	0.237516	0.000003
				SEM	0.060342	0.051096	0.023752	0
6	[-5, 10]	Rosenbrock	0	Mean	0.132091	0.381039	0.000669	0.000006
				Std	0.543049	1.252266	0.0034	0.000003
				SEM	0.054305	0.125227	0.000340	0
7	[-5, 10]	Zakharov	0	Mean	0.000028	0	0	0.000005
				Std	0.000081	0	0	0.000003
				SEM	0.000008	0	0	0
8	[-32.768, 32.768]	Ackley	0	Mean	0.153007	0.136212	0	0.000006
				Std	0.981908	0.696945	0	0.000002
				SEM	0.098191	0.069694	0	0
9	[-512,512]	Eggholder	-959.640662	Mean	-548.277338	-854.386626	-952.987175	-959.640660
				Std	189.718803	96.285003	29.083782	0.000003
				SEM	18.971880	9.628500	2.908378	0
10	[-10, 10]	Dixon-Price	0	Mean	0.000389	0	0.000004	0.000005
				Std	0.001618	0	0.000036	0.000003
				SEM	0.000162	0	0.000004	0

Table 2: Statistical results of 100 runs obtained by SA, GA, PSO and Seeker Algorithm. Mean: Mean of the Best Values, StdDev: Standard Deviation of the Best Values, SEM: Standard Error of Means

is the number of evaluation functions that an algorithm takes to get to the global minimum. The lower this value is, lower the computation execution to make during the search. In the case of the experiments, it can be observed that Seeker Algorithm outperforms in 6 out of 10 functions. In some cases, the function is even more than half less with respect to others in terms of function evaluations. This means that the algorithm has to refer to the objective function much less than the other algorithms to reach the global minimum. Seeker Algorithm does its best on Goldstein, Michalewicz, Sphere, Zakharov, Ackley, and Dixon-Price. Moreover, the standard deviation remains low in most of the cases. PSO wins in terms of function evaluation just ones. All the results can be seen in Table 3.

No	Function	SA	GA	PSO	SeekA	
		Mean	Mean	Mean	Mean	Std
1	Goldstein	1443	3494	1338	**589**	443
2	Michalewicz	1689	1374	749	**650**	795
3	Schwefel	1820	4811	1755	4360	5239
4	Sphere	1711	3042	1198	**306**	136
5	Rastrigin	1749	3529	1877	4870	4269
6	Rosenbrock	2154	7910	**2711**	3662	2521
7	Zakharov	1704	3118	1245	**337**	168
8	Ackley	1955	4118	1983	**1120**	680
9	Eggholder	1821	4868	1365	9724	13011
10	Dixon-Price	1815	3416	1411	**443**	235

Table 3: Solution costs in terms of evaluation number of SA, GA, PSO and Seeker Algorithm. Bold results indicate the best function evaluation with 100% of success rate.

The cases where an algorithm does not converge a 100% to the global minimum is not considered best in Table 3. Other measures were done by comparing the 4 algorithms in terms of success rate. Some algorithms may not converge to the global minimum leading the search in a local minimum. This experiment is important in order to understand how many times an algorithm can reach the global minimum.

In Table 4 can be observed that Seeker Algorithm reaches the global minimum all the times with no manual parameter tuning. This is not valid for other comparison algorithms. The other one needs configuration tuning as a different population size or different temperature reduction function in order to reach the global minimum.

5 Conclusion and future works

In the above sections, Seeker Algorithm was presented formally and a series of experiments were made to measure the performance of it. In most of the experiments, Seeker Algorithm overcomes the other by seeing the function much fewer times than the other.

The future works will be concentrating on exploiting more than one agent creating a population-based algorithm that uses collaboration to reach the global minimum. Another interesting work will be to measure the single-agent, multi-agent performance of the algorithm on multidimensional spaces and by expanding the test function set to regular and irregular once.

No	Function	SA	GA	PSO	SeekA
1	Goldstein	**100**	97	**100**	**100**
2	Michalewicz	99	**100**	**100**	**100**
3	Schwefel	4	86	69	**100**
4	Sphere	**100**	**100**	**100**	**100**
5	Rastrigin	47	69	94	**100**
6	Rosenbrock	97	88	**100**	**100**
7	Zakharov	**100**	**100**	**100**	**100**
8	Ackley	96	96	**100**	**100**
9	Eggholder	1	5	91	**100**
10	Dixon-Price	**100**	**100**	**100**	**100**

Table 4: Success rates (%) of SA, GA, PSO and Seeker Algorithm. Bold results indicate the best success rate achieved.

References

[1] Viswanathan. G. Fish in Lévy-flights foraging. *Nature*, 465:1018–1019, 2010.

[2] Viswanathan, G., Buldyrev, S., Havlin. S. et al. Optimizing the success of random searches. *Nature*, 401:911–914, 1999.

[3] E P Raposo, S V Buldyrev, M G E da Luz, G M Viswanathan, and H E Stanley. Lévy flights and random searches. *Journal of Physics A: Mathematical and Theoretical*, 42(43):434003, oct 2009.

[4] Humphries, N., Queiroz, N., Dyer. J. et al. Environmental context explains Lévy and Brownian movement patterns of marine predators. *Nature*, 465:1066–1069, 2010.

[5] M. Lozano and C. García-Martínez. Hybrid metaheuristics with evolutionary algorithms specializing in intensification and diversification: Overview and progress report. *Computers & Operations Research*, 37(3):481 – 497, 2010. Hybrid Metaheuristics.

[6] Aleksei Chechkin, Ralf Metzler, Joseph Klafter, and Vsevolod Gonchar. *Introduction to the Theory of Lévy Flights*, pages 129 – 162. Researchgate, 09 2008.

[7] Lars Fredriksson. A brief survey of lévy walks : with applications to probe diffusion, 2010.

[8] Ilya Pavlyukevich. Lévy flights, non-local search and simulated annealing. *Journal of Computational Physics*, 226(2):1830 – 1844, 2007.

[9] Yang Xin-She. *Nature-Inspired Metaheuristic Algorithms*. Researchgate, 07 2010.

[10] X. Yang and Suash Deb. Cuckoo search via lévy flights. In *2009 World Congress on Nature Biologically Inspired Computing (NaBIC)*, pages 210–214, Dec 2009.

[11] L. Rabiner and B. Juang. An introduction to hidden markov models. *IEEE ASSP Magazine*, 3(1):4–16, 1986.

[12] Xin-She Yang and Suash Deb. Multiobjective cuckoo search for design optimization. *Computers & Operations Research*, 40(6):1616 – 1624, 2013. Emergent Nature Inspired Algorithms for Multi-Objective Optimization.

[13] Scott Kirkpatrick, C. Gelatt, and M. Vecchi. Optimization by simulated annealing. *Science (New York, N.Y.)*, 220:671–80, 06 1983.

[14] Peter J VAN Laarhoven, Ohki, Yasutsugu, and Demsew Teferra. Simulated annealing: Theory and applications. *STOR*, 01 2020.

[15] L. Ingber. Simulated annealing: Practice versus theory. *Mathematical and Computer Modelling*, 18(11):29 – 57, 1993.

[16] John H. Holland. *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*. MIT Press, 1992.

[17] J. Kennedy and R. Eberhart. Particle swarm optimization. In *Proceedings of ICNN'95 - International Conference on Neural Networks*, volume 4, pages 1942–1948 vol.4, 1995.

[18] David Johnson, Cecilia Aragon, Lyle McGeoch, and Catherine Schevon. Optimization by simulated annealing: An experimental evaluation. part i, graph partitioning. *Operations Research*, 37:865–892, 12 1989.

[19] Haldurai Lingaraj. A study on genetic algorithm and its applications. *International Journal of Computer Sciences and Engineering*, 4:139–143, 10 2016.

[20] Lijun Sun, Xiaodong Song, and Tianfei Chen. An improved convergence particle swarm optimization algorithm with random sampling of control parameters. *Journal of Control Science and Engineering*, 2019:1–11, 06 2019.

[21] Dongshu Wang, Dapei Tan, and Lei Liu. Particle swarm optimization algorithm: an overview. *Soft Computing*, 01 2017.

[22] Dervis Karaboga and Bahriye Akay. A comparative study of artificial bee colony algorithm. *Applied Mathematics and Computation*, 214(1):108 – 132, 2009.