GLOBAL OPTIMIZATION USING LEVY FLIGHT

The Truyen Tran, Trung Thanh Nguyen, Hoang Linh Nguyen

Abstract

This paper studies a class of enhanced diffusion processes in which random walkers perform Levy flights which are frequently observed in statistical physics and biology. Levy flights are shown to have several attributes suitable for the global optimization problem, and those four algorithms are developed based on such properties. We compare new algorithms with the well-known Simulated Annealing on considerably hard test functions and the results are very promising

Keywords: *Levy Flight, optimization, algorithm.*

1. introduction

The optimization can be characterized, without lost of generality, as a process to minimize the objective function over a bounded or unbounded space. Of the widely used class of optimization algorithms called meta-heuristics, it is interesting to note that many of them imitate natural processes such as *Simulated Annealing* (SA - originated in physics) (S. Kirkpatrick, 1983), *Genetic Algorithm* (GA - imitating the Darwinian process of natural selection) (De Jong, 1975), *Ant Colony Optimization* (ACO – mimicking behavior of foraging ants) (Marco Dorigo, Vittorio Maniezzo, Alberto Colorni, 1996) and *Particle Swarm Optimization* (PSO – modeling the flocking and schooling by birds)(E. Eberhart, Y. Shi, 2001). In particular, SA and GA have long been known to be very practically successful although they never guarantee to reach optima within limited time.

This paper studies a group of stochastic processes that are frequently observed in physics and biology called Levy flights after Paul Levy, a French mathematician. The work is to realize the claim by Gutowski that Levy flights can be used in optimization algorithms. Based on those concepts, a new class of meta-heuristic algorithms called LFO (Levy Flights Optimization) is introduced. To our knowledge, similar work has not been introduced in literature, except for few attempts in natural sciences such as (Bardou, 1999) and (Viswanathan et al, 1999).

The rest of paper is organized as the following. Section 2 outlines some fundamentals of Levy flights. Section 3 discusses relevant background in meta-heuristic optimization and introduces four new LFO algorithms. Section 4 reports experiments to realize those algorithms and discusses the results. It introduces a framework that supports optimization algorithm implementations, comparisons and combinations as well as test problems used in experiments. The last section provides gives several possible outlooks in extending the work presented in the paper.

2. Levy flights

Extensive investigations in diffusion processes have revealed that there exist some processes not obeying Brownian motion. One class of processes is *enhanced diffusion*, which has been shown to comply with Levy flights. The Levy flights can be approximately characterized by following probability density function:

$$P(x) \sim |x|^{-1-\beta} \text{ as } x \to \infty \text{, where } 0 < \beta \le 2 \quad (1)$$

Note that with $\beta \le 0$, distribution in Equation (1) cannot be normalized, therefore it has no physical meanings although our computation needs not be concerned about such problem. For $0 < \beta < 1$, the expectation value does not exist.

For the interest of this paper, we consider a process of random walk with each step of length l with the probability of following form:

$$P(l) = \frac{\beta}{l_0 \left(1 + l / l_0\right)^{1+\beta}} \quad (2)$$

This is a normalized version of (Gutowski, 2001) with the scale factor l_0 added since it is more natural to think in term of physical dimension of given space. This form preserves the property of distribution in Equation (1) for large l but it is much simpler to deal with small l. The distribution is heavy-tailed, as shown on Figure 1.

It is not difficult to verify that l can be randomly generated as follows:

$$l = l_0 (\frac{1}{U^{1/\beta}} - 1) , \quad (3)$$

where U is uniformly distributed in the interval [0,1).

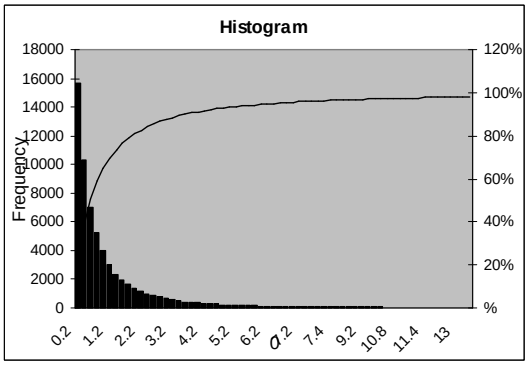

Figure 1: Flight length distribution (β = 1.5)

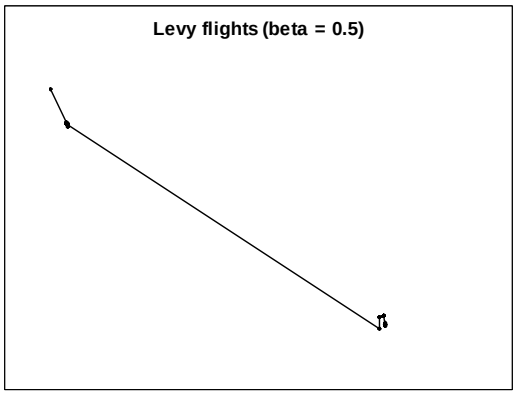

Figure 2: β = 0.5

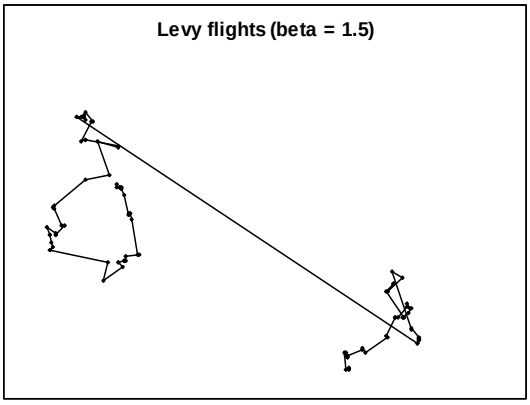

Figure 3: β = 1.5

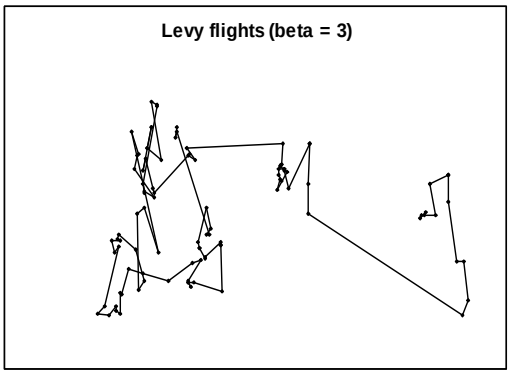

Figure 4: β = 3

Figures 2-4 depict Levy fights on 2-D landscape with l_0 = 1. Note that the scale of Figure 2 is much larger than that of Figure 4 (around order of 10^2). For small β, random walker tends to get crowded around a central location and occasionally jump a very big step to new location. As β increases, the probability of performing a long jump decreases. Note that Figure 4 shows the random walks as β = 3, which gets outside of the range given in Equation (1). However, for the computational purposes, we need not to strictly be consistent with physics laws.

3. algorithms

3.1. Search mechanisms

The optimization is a search process that has a goal of efficiently exploring the search space in order to find globally best solution. In other words, search defines how to move on the solution landscape towards the optima. Levy flights provide a basic mechanism to do so.

Let's start by noting that random walkers obeying Levy flights behave very much similar to those in evolutionary processes. For example, GA is based on an implicit assumption that similar good solutions may be generated using recombination (Balujia, Scott Davies) while occasional mutations may help explore unvisited regions. As analyzed in Section 2, such behaviors are already exposed well in Levy flights based processes. This paper shares the same suppositions, which proposed LFO algorithms are based on. The algorithms are described in the next sub-section.

We use the term "particle" to call the abstract entity that moves on the search landscape. The basic idea behind the algorithm is that from each position, the particle can "fly" to a new feasible position at a distance, which is randomly generated from Levy flights distribution. Many observations have revealed that often high quality optima are located "centrally" among local optima, similar to the picture of a "big valley" (Bose, 2000). If this conclusion generally holds in real-world applications, Levy flights process, together with other mechanisms hopefully help to a find number of local optima and then from those identifying the promising global region.

Another terms often used in meta-heuristics are *intensification/diversification* and (or) *exploitation/exploration*. It is generally agreed that a good search algorithm should maintain balance between the local exploitation and global exploration. Levy flights process does good job of this. Note in Equation (2), we can control the frequency and the length of long jumps by adjusting the parameters β and l_0, respectively . In the ideal case, the algorithm based on Levy flights should be able to dynamically tune the two parameters to best fit given landscape.

There are many ways to formulate an algorithm that employs such Levy-based distance. Firstly, Levy flights define a manageable move strategy, which can include small local steps, global jumps and (or) mixing between the two. As the Levy process can occasionally generate long jump, it can be employed in the multi-start model.

Secondly, one can use a single particle (as in Greedy, SA, Tabu Search) (F. Glover, M. Laguna, 1997) (Tabi, 2002) or a set of particle(s) (as in GA, Evolution Strategy, Genetic Programming, ACO and Scatter Search) (Christian Blum, Andrea Roll, 2003) moving over the search space. The third way, and probably the most promising, is to combine generic movement model provided by Levy flights with other known single-solution, or population-based meta-heuristics. Such combinations often result in more powerful hybrid algorithms than the originals (Talbi, 2002).

3.2 LFO algorithms

Here the author proposes four algorithms based on the idea of LFO (Levy Flights Optimization) and discussions in previous sub-section.

- Algorithm 1: Basic LFO (LFO-B)
- Algorithm 2: Hybrid LFO with Local Search (LFO-LS)
- Algorithm 3: Local Search with Multiple LFO Restarts (LFO-MLS)
- Algorithm 4: LFO with Iterated Local Search (LFO-ILS)

LFO-B Algorithm

This is quite *basic* algorithm which uses a set of particles at each "generation". Starting from one best-known location, the algorithm will generate a whole new generation at distances which are randomly distributed according to Levy flights. The new generation will then be evaluated to select the most promising one. The process is repeated until stopping criteria are satisfied. Current implementation of LFO-B is rather simple: only the best solution. This may open rooms for further improvements, i.e. with selection mechanisms borrowed from GA literature. The algorithm is quite closed to GA in a way that it iterates through generations of particle population, and at each generation it "selects" the good individual for next. However, there selection policy is quite simple: only the best of population survives and it can generate the whole new generation.

LFO-LS Algorithm

This is an extension of LFO-B algorithm, where before the selection is made each particle performs its own search to reach local optima. The selection procedure then locates the best local optima found so far, and Levy flights mechanism helps escape from such trap. It is open to implement the *local search* algorithm. Two simple ones are *first-improving* and *best-improving*. In the case of first-improving, the searcher moves to any neighbor that is better than the current. On the other hand, best-moving strategy requires evaluation of all possible (or at least enough) neighbors to select the best. It is observed that first-improving strategy is less computation expensive while leading to the same final solution as the best-improving (Mauricio G. C. Resende, Celso C. Ribeiro, 2002).

LFO-MLS Algorithm

This can be viewed as a sequential version of LFO-LS algorithm and actually behaves as a *Multi-start Local Search*. Here only one particle is used. In each iteration the particle tries local search until getting trapped in local optima. To escape from such trap, it will jump to new location by a step generated from Levy distribution. The jump is immediately accepted without further selection.

LFO-ILS Algorithm

This is very similar to LFO-MLS except for the local minima escape strategy. Instead of immediately accepting the Levy-based jump, the algorithm tries a number of jumps until a better solution is found. This behavior is identified as *Iterated Local Search*.

The four algorithms are detailed as follows:

```
procedure LFO_B()
init_position();
curr_val := objective_function();
best_val := curr_val;
best_position := curr_position();
while(stopping_criteria_not_met)
        while(jump_no_not_reach_maximum)
```

```
            flight_len =
            Levy_flights(base_len,beta);
            jump_randomly_at_distance(flight_l
            en);
            curr_val := objective_function();
            if(curr_val < best_val)
                  best_val := curr_val;
                  best_position :=
                  curr_position();
            end if
      end while
        return_to_best_known_position();
end while
end procedure
```

Algorithm 1: Basic LFO (LFO-B)

```
procedure LFO_LS()
init_position();
curr_val := objective_function();
best_val := curr_val;
best_position := curr_position();
while(stopping_criteria_not_met)
      while(jump_no_not_reach_maximum)
            flight_len =
            Levy_flights(base_len,beta);
            jump_randomly_at_distance(flight_l
            en);
            perform_local_search(curr_position
            ());
            curr_val := objective_function();
            if(curr_val < best_val)
                  best_val := curr_val;
                  best_position :=
                  curr_position();
            end if
      end while
        return_to_best_known_position();
end while
end procedure
```

Algorithm 2: Hybrid LFO with Local Search (LFO-LS)

```
procedure LFO_MLS()
init_position();
curr_val := objective_function();
best_val := curr_val;
best_position := curr_position();
while(stopping_criteria_not_met)
        perform_local_search(curr_position());
        curr_val := objective_function();
        if(curr_val < best_val)
              best_val := curr_val;
              best_position := curr_position();
        end if
        flight_len := Levy_flights(base_len,beta);
        jump_randomly_at_distance(flight_len);
end while
end procedure
```

Algorithm 3: Local Search with Multiple LFO Restarts (LFO-MLS)

```
procedure LFO_ILS()
init_position();
curr_val := objective_function();
best_val := curr_val;
best_position := curr_position();
while(stopping_criteria_not_met)
        perform_local_search(curr_position());
        curr_val := objective_function();
        if(curr_val < best_val)
              best_val := curr_val;
              best_position := curr_position();
        end if
        flight_len := Levy_flights(base_len,beta);
        while(not found better solution)
              jump_randomly_at_distance(fli
        ght_len);
        end while
end while
end procedure
```

Algorithm 4: LFO with Iterated Local Search (LFO-ILS)

4. EXPERIMENTS & RESULTS

4.1 Optimization framework

The authors developed a framework to ease the process of implementing and testing meta-heuristic optimization algorithms and applications. The framework was also aimed at providing a way for comparing and combining such algorithms.

There are three basic concepts forming the core of framework: *particle*, *landscape* and *optimizer*. The *particle* is a conceptual entity that can fly over the

search landscape. The *landscape* is a description of space containing all feasible solutions within problem-specific constraints. It is realized by search problems that defining search dimension, constraints and objective functions. The *optimizer* is recognized by search algorithms that control the movement of particles on given landscape.

There may be a single, or a set of particles managed by a *particle manager*. Each particle has *position* and *velocity*, which in turn consists of speed (scalar value in Euclidean space) and direction (a unit vector). The local movement can be either near continuous with real values (within machine dependent precision) or discrete with given number of *bits*.

A long term goal is the software should evolve to (*i*) a black-box optimization component which can be easily plugged into other container frameworks, or to (*ii*) a framework where applications can be built on top with minimal effort. At the time of this writing, a real commercial problem of forecasting had been solved using the framework using the second approach. The job was to best fit time-series data of telecommunications demands to several growth-curve models and then to automatically determine which model worked best for given data set.

The software was developed and tested in Windows environment (Visual C++ 6.0) but it was written entirely in ANSI C++ so it should not pose any problem in porting to other platforms.

4.2 Test problems

Test problems used to compare SA and four LFO algorithms are F_0 of Corana et al, F_2 (Rosenbrock's saddle) and F_5 (Shekel's foxholes) in well-known De Jong's five test function suite (De Jong, 1975), Rastrigin's F6 (D. Whitley, K. Mathias, S. Rana and J. Dzubera) and Keane's Bump (Keane).

F_0:

$$f_0(x_1,...,x_4)=\sum_{i=1}^{N}\begin{cases}(t_i\,\mathrm{sgn}(z_i)+z_i)^2 c d_i & \text{if } |x_i-z_i|<|t_i|\\ d_i x_i^2 & \text{otherwise,}\end{cases}$$

$$z_i=\left\lfloor\left|\frac{x_i}{s_i}\right|+0.49999\right\rfloor\mathrm{sgn}(x_i)s_i,$$

$$s_i=0.2, t_i=0.05, i=1,4,$$

$$d_i=\{1.0,1000.0,10.0,100.0\},$$

$$c=0.15,$$

$$-1000.0\le x_i\le 1000.0, i=1,..,4,$$

F_2:

$$f_2(x_1,...,x_N)=\sum_{i=1}^{N-1}(100(x_i^2-x_{i+1})^2+(1-x_i)^2),$$

$$-2.048\le x_i\le 2.048, i=1,...,N.$$

F_5:

$$f_5(x_1,x_2)=\frac{1}{\frac{1}{500}+\sum\frac{1}{j+\sum_{j=1}^{25}(x_i-a_{ij})^6}},$$

a_{j1} = {-32,-16,0,16,32,-32,-16,0,16,32,-32,-16,0,16,

32,-32,-16,0,16,32,-32,-16,0,16,32},

a_{j2} = {-32,-32,-32,-32,-32,-16,-16,-16,-16,-16,0,0,0,

0,0,16,16,16,16,16,32,32,32,32,32},

$-65.536 \le x_i \le 65.536$, $i = 1,2$.

F_6:

$$f_6(x_1,...,x_N) = 10N + \sum_{i=1}^{N}(x_i^2 - 10\cos(2\pi x_i)),$$
$$-5.12 \le x_i \le 5.12$$

Bump:

$$f_{BUMP}(x_1,...,x_N) = \frac{abs(\sum_{i=1}^{N}\cos^4(x_i) - 2\prod_{i=1}^{N}\cos^2(x_i))}{\sqrt{\sum_{i=1}^{N} i x_i^2}},$$

$$0 < x_i < 10$$

$$\prod_{i=1}^{N} x_i > 0.75, \qquad \sum_{i=1}^{N} x_i < 15n/2$$

with starting point is x_i = 5, $i = 1,\ldots,N$.

The Function F_0 is very difficult to minimize with 10^{20} local minima, and F2 is also considered to be hard despite of being uni-modal (Corana, 1987). We reject other De Jong's choices of F1, F3 and F4 because they have only one minimum so that the greedy type algorithms seem to perform better. F6 is interesting because of the sinuous component. According to Keane, the Bump is a seriously hard test-function for optimization algorithms because the landscape surface is highly "bumpy" with very similar peaks and global optimum is generally determined by the product constraint. The last feature makes Bump an excellent case to test the power of LFO algorithms under assumption of "big valley" structure of search landscape. Readers may wish to consult relevant references for more details on those test functions characteristics.

Among those functions, F_0 and F_5 have fixed dimension while the rest can be set freely so search space size can be scalable. In our tests, the dimension of 10 is set for F2 and F6 and of 50 for the Bump. Note that in the Bump problem, the original objective is to maximize the function f_{BUMP} so it is converted to 1-f_{BUMP} to keep consistency among all test functions.

4.3 Algorithms implementation

Five algorithms compared in this paper are SA and the four newly proposed LFO(s) described in Section 3. This sub-section provides greater details in algorithm realization, which are generally known to significantly affect algorithm efficiency. Such implementation can be classified into *move strategy, stopping criteria*, and *algorithm specificity*.

***Move strategy*:** each move is selected at random directions spanning in all dimensions of search space where move length is measured in Euclidean metric. In the case of infeasible move to the region outside bounded space, two strategies are used: (*i*) the move selection process is repeated until a feasible one is found, and (*ii*) the move is stopped at the edges. The first strategy is quite intuitive but it may result in many repetitions in case of long Levy flights. The second gives chance to explore the boundary regions, in which high quality solutions may be found.

***Stopping criteria*:** the major stopping criteria used in main algorithms are time and the quality of best solution found so

far (compared with known optimal one). The first criterion is more practical because in general we do not know the optimal solution before the search is made. Moreover, large-scale problems often require weeks or months to run while time budget for real-world projects is critically limited. The search algorithm, therefore, should be time-sensitive (Vinhthuy Phan, Pavel Sumazin, Steven Skiena, 2002), that it should make the best of allocated time.

Using time can have another advantage that it makes easier to compare efficiency of algorithms. Although there is another way to do comparison such as number of iterations or function calls, run time for each iteration does vary greatly in algorithm implementation. In addition, it is not obvious of how to count a function call in case of nested control (**while**, **for**) loops and in hybrid heuristics.

Another stopping criterion is number of non-improvement moves. This can be either number of steps in neighborhood exploitation or number of jumps in global exploration. The first is used in greedy-like search, while the second applies for multiple restart type.

Algorithm specificity: in SA, the initial temperature T_0 is chosen as 10% of randomly generated initial solution while the stopping temperature T_s is fixed at 0.0001. Although there are no exact reasons for such choice, it is based on author's experience with SA so that transition probability is always 1 at beginning and very closed to 0 at the end of each run. The cooling schedule is the widely used power type:

$$T(t) = r^t T_0 ,$$

where $r = e^{\ln(T_s / T_0) / t_m}$, and t_m: maximum allocated run time. (4)

The idea is based on experimental observations and theoretical proof that SA finds very high quality solutions if the cooling is "slow" enough. The cooling rate r hence should not be fixed against time budget.

In all LFO algorithms, the power index β = 1.5. The number of jumps is set at 100 for LFO-B and LFO-MS algorithms, while the jump distance is limited at a half of largest size of search space's dimensions.

4.4 Results and Discussions

The tests were run on 2.5GHz Intel computer within given periods of time. As meta-heuristics in this paper are proposed for practical purposes, the exact numerical solutions (within machine's precision) are not very important. "Good enough" solutions (compared with known optima) were used as basis for setting run time. Every test was run repeatedly and best found solutions were sampled at relevant intervals and then averaged. Numbers of replications were 100 times for F0, F2, and F5, 20 times for F6 and 10 times for the Bump. Six algorithms' performances on the five test problems are presented in Figure 5 to Figure 9.

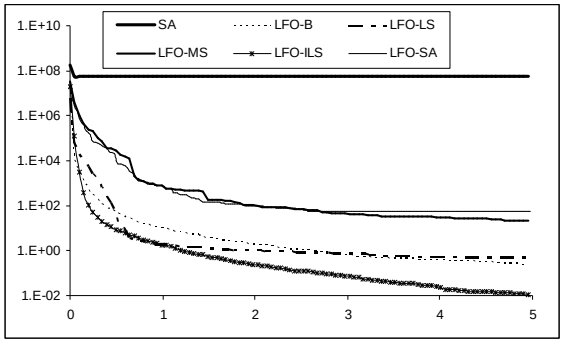

Figure 5: Test results for F0

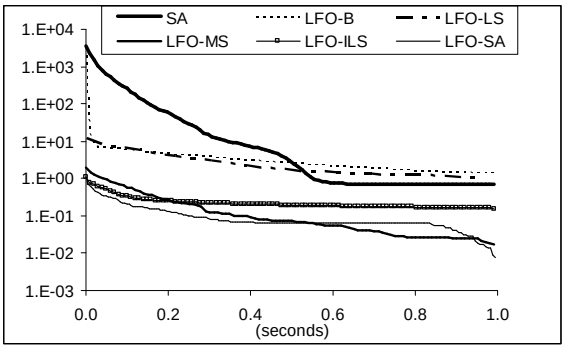

Figure 6: Test results for F2

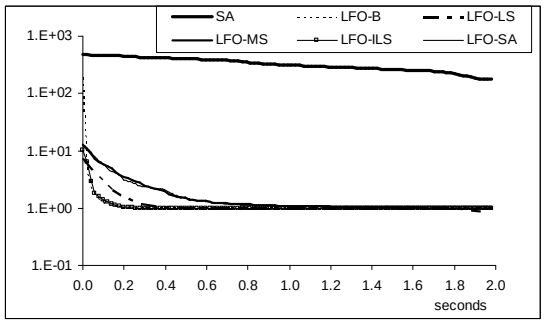

Figure 7: Test results for F5

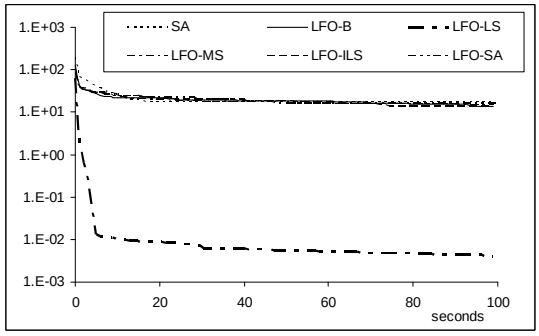

Figure 8: Test results for F6

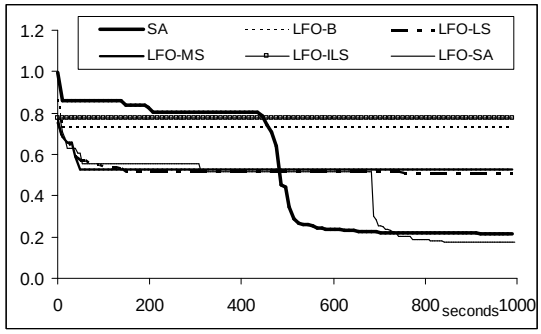

Figure 9: Test results for the Bump problem

In all cases, some of LFO algorithms outperform SA although SA appears to improve its performance in the long runs. This is a proven beauty of SA but it will be impractical if the required time is too long for daily activities. LFO algorithms tend to sample good quality solutions (compared to initial random solution) very quickly, in most cases within a second. The only exception is the Bump problem, where SA seems to work best after a significant time.

Although there are not enough representative test cases to statistically conclude on the power of LFO algorithms over SA, there are several interesting points to see. Firstly, according to (Viswanathan et al, 1999) Levy flights portrays well the distribution of flight lengths and times performed by foragers

observed in experiments. They also suggest that in the process of foraging over a given landscape, Levy distribution helps reduce the probability of returning previously visited locations, compared to normal distribution. This is particularly advantageous in memory-less algorithms like basic stochastic SA or GA, where there are no built-in mechanisms to avoid revisits.

Secondly, as mentioned early in this paper, LFO algorithms work under assumption that good quality solution can be found around (and among) local minima. Such property can be found in most test cases, where experiments have proved the algorithms' favor. However, in the Bump problem, such "big valley" hypothesis does not hold, i.e. the global minimum is located on search space boundary. This property can help explain why LFO algorithms fail to model the Bump landscape structure, and why they succeed in other cases.

Finally, we also implemented a simple hybrid algorithm called LFO-SA, which is a combination of LFO-MLS and SA separated in time. The idea is based on general observation that LFO algorithms often locate the good solution quickly while SA helps improve the quality in the long run. The hybrid LFO-SA appears to work as well as or better SA in all cases while keeping the similar search power to other LFO algorithms. It is obvious that LFO-SA behaves exactly like LFO-MLS in the short run (in F0, F2, F5 and F6, see Figure 5-Figure 8) and like SA in the long course (in Bump, see Figure 9).

5. conclusions & outlooks

The paper has proposed a set of optimization algorithms based on Levy flights, in which several hard problems of continuous variables were used to test against the well-known Simulated Annealing algorithm. Experiments have demonstrated that LFO (Levy Flights Optimization) could be superior to SA under some general assumptions.

The experiment of hybridization was meant to explore ways of combining two very different types of algorithms to seek further improvement. It was also based on intuition of known behavior of LFO-MS in short run and SA in long run. Perhaps, in order to be successful, the hybridization needs to be build upon a theoretical basis, which has not existed yet.

Algorithms implemented in this paper are rather basic, and there are rooms for further extension. For example, the two main parameters of mean Levy distance and the power index can be adjusted dynamically during the run. Or one may wish to extend the idea of Particle Swarm Optimization (PSO) in the way that the whole population keeps continuous flying while dynamically adjusting speed and direction based on information collected so far. Even each flying particle can be treated as an autonomous agent involving collective intelligent decision making. For those who are familiar with GA, it is possible to extend the practice of GA in the way that at each generation, we select a set of particles in stead of the best one to form the next generation.

References

[Balujia00] Shummeet Balujia, Scott Davies, "Probabilistic Modeling for Combinatorial Optimization", *Statistical Machine Learning for Large-scale Optimization, Neural Computing Surveys* 3, 1-58, 2000

[Bardou99] Francois Bardou. "Cooling gases with Levy lights: using the generalized central limit theorem in physics." *Mini-proceedings: Conference on Levy processes: theory and applications*, Aarhus 18-22 january 1999, MaPhySto Publication (Miscellanea no. 11, ISSN 1398-5957),O. Barndor-Nielsen, S.E. Graversen and T. Mikosch (eds.). (1999):

[Blum03] Christian Blum, Andrea Roll, "Metaheuristics in Combinatorial Optimization: Overview and Conceptual Comparison," *ACM Computing Surveys (CSUR)*, Volume 35 Issue 3, Sep 2003

[Bose00] Kenneth D. Bose, "A Review of Iterative Global Optimization", *Statistical Machine Learning for Large-scale Optimization, Neural Computing Surveys* 3, 1-58, 2000

[Corana87] A. Corana, M. Marchesis, C. Martini and S.Ridella, "Minimizing multimodal functions of continuous variables with the "simulated annealing" algorithm", *ACM Transactions on Mathematical Software (TOMS)*, Volume 13 Issue 3, 1987.

[De Jong75] K. A. De Jong, "An analysis of the behavior of a class of genetic adaptive systems," *Ph.D. dissertation*, Univ. Michigan, Ann Arbor, 1975.

[Dorigo96] Marco Dorigo, Vittorio Maniezzo, Alberto Colorni. "The Ant System: Optimization by a colony of cooperating agents", *IEEE Transactions on Systems, Man, and Cybernetics Part B: Cybernetics*, Vol.26, No.1, pp. 1-13, 1996.

[Eberhart01] E. Eberhart, Y. Shi, "Particle swarm optimization: developments, applications and resources", *Proceedings of IEEE International Conference on Evolutionary Computation*, pp. 81-86, 2001.

[Glover97] F. Glover, M. Laguna, "Tabu Search", *Modern Heuristic Techniques for Combinatorial Problems*, 1997.

[Gutowski01] M. Gutowski. "Levy fights as an underlying mechanism for global optimization algorithms", *V Domestic Conference on Evolutionary Algorithms and Global Optimization*, May 30th - June 1st, 2001, Jastrz\c{e}bia G\'ora (Poland).

[Kean03] A.D.Kean, "A Brief comparison of some evolutionary optimization methods", URL: http://www.soton.ac.uk/~ajk/brunel.ps, download 30 Dec, 2003.

[Kirkpatrick83] S. Kirkpatrick, C. D. Gelatt, Jr., M.P. Vecchi, "Optimization by Simulated Annealing," *Science*, Number 4598, 13 May 1983

[Phan02] Vinhthuy Phan, Pavel Sumazin, Steven Skiena. "A Time-Sensitive System for Black-Box Combinatorial Optimization (2002)." *Algorithm Engineering and Experiments*, LNCS 2409, pp. 16-28, Springer-Verlag, 2002:

[Resende02] Mauricio G. C. Resende, Celso C. Ribeiro, "Greedy Randomized Adaptive Search Procedures," *AT&T Labs Research Technical Report*, Sept. 2001. Revision 2, Aug. 29, 2002. To appear in "State of the Art Handbook in Metaheuristics", F. Glover and G. Kochenberger, eds., Kluwer, 2002 (2002):

[Rosen92] Bruce Rosen, Lester Ingber. "Genetic Algorithms and Very Fast Simulated Reannealing: A Comparison." *Mathematical and Computer Modeling*, 16(11), 1992, 87-100

[Talbi02] EG. Talbi, "A Taxonomy of Hybrid Metaheuristics," *Journal of Heuristics*, pp. 541-564. Volume 8, Number 6, November 2002

[Viswanathan99] G. M. Viswanathan, Sergey V. Buldyrev, Shlomo Havlin, M. G. E. da Luzk, E. P. Raposok & H. Eugene Stanley. "Optimizing the success of random searches", *Nature*, Vol 401, 28 October 1999.

[Whitley95] D. Whitley, K. Mathias, S. Rana and J. Dzubera, "Building better test functions", *Proceedings of the 6th International Conference on Genetic Algorithms* pp: 239 – 247, 1995, ISBN:1-55860-370-0