Evolutionary Diffusion Optimization, Part II: Performance Assessment

Kwok Ching Tsui and Jiming Liu
Department of Computer Science
Hong Kong Baptist University
Kowloon Tong, Kowloon, Hong Kong
email: {tusikc,jiming}@comp.hkbu.edu.hk

Abstract - A new population-based stochastic search algorithm called *Evolutionary Diffusion Optimization* (EDO) inspired by diffusion in nature has been proposed [10]. This article compares the performance of EDO with simulated annealing and fast evolutionary programming. Experimental results show that EDO performs better than SA and FEP in some cases.

I. Introduction

The objective of global optimization is to find a solution to a function which has the highest or lowest value depending on the task is maximization of minimization [1]. Consider a function $F(x)$ and $x = \{x_1, x_2, ..., x_y\}^T$ is a y-dimensional vector representing the parameters of function F. The optimal solution is represented by $F(x^*)$ such that

$$F(x^*) < F(x) \quad \forall x \tag{1}$$

Global optimization is difficult for at least two reasons:
1. *Unpredictability* The landscape of the function to be optimized in unknown. A search algorithm is likely to be trapped in suboptimal solutions, i.e. places where the objective function value is lower than the neighborhood areas but is not the optimal solution over the space of all possible solutions.
2. *Curse of Dimensionality* When changes are made to the value of one or more parameters, it is not immediately known how much contribution each of those changes are to the change in the function value. The dimensionality also affects the space and computational time required of the search algorithm. It also makes the exhaustive search, one which tries out all possible combinations of values of all parameters, unfeasible.

Many algorithms have been developed over the years to tackle the challenging task of global optimization [2]. Some well-known *stochastic search algorithm* such as simulated annealing [3], evolutionary algorithms [4], particle swarm [5], cultural algorithm [6] and ant colony optimization [7] have been applied to optimization tasks. With the exception of the SA, all the above search algorithms are population-based search because many candidate solutions are considered at the same time.

Diffusion can be observed every where in nature, ranging from molecules in a fluid to human beings around the world [8], [9]. A new population-based stochastic search algorithm called *evolutionary diffusion optimization* (EDO) has been proposed based on some observations of the natural diffusion [10]. Unlike the population-based algorithms mentioned above, the population size in EDO is not fixed because it is very 'lenient' towards poor performers in the population. Moreover, individual candidate solutions are changed based on information obtained in its proximity. The right to reproduce is also granted based on local competition. Initial analysis reveals that EDO is able to automatically maintain a good balance between exploration and exploitation. This article reports the results of several experiments attempted to assess the performance of EDO. These results are checked against two well-known algorithms, namely, simulated annealing and fast evolutionary programming [11].

II. The EDO Algorithm

Figure 1 contains a high level view of the steps in evolutionary diffusion optimization (EDO) and its major functions. This section briefly describes the data structure, the main features (local behavior) and global information adaptation. Details can be found in an accompanying paper [10].

A. Basic Data Structure

1. *Object Vector* each entity carries all the object variables of the function to be optimized and are tuned at the same time by EDO.
2. *Probability Matrix* The $m \times n$ matrix contains the likelihood estimate of success with respect to direction of motion. Parents and their offspring share the same probability matrix until an entity becomes a parent.
3. Fitness Fitness in EDO measures an entity's degree of success in the course of the search from the optimal solution and is equal to the function value.

B. Local Behaviors

B.1 Diffuse

Entities in EDO explores uncharted locations in the solution space by taking one of the following two forms of diffusion.
1. *Rational Move* An entity modifies its object values according to the values in the probability matrix and the step size parameter.
2. *Random Walk* An entity diffuses either towards the upper limit or the lower limit. The probability of taking random walk is given by:

$$P_{random\_walk} = e^{-\frac{lifespan-age}{\alpha}} \tag{2}$$

B.2 Reproduce

All entities having a fitness higher than their parents will replicate itself certain number of times. However, the actual number of offspring is limited by an entity's fitness and the total number of entities currently in the system. New entities are sent off to a new location by rational move and duplicates are eliminated. An inactive entity will be allowed to spawn new offsprings if all its offspring are dead, and its fitness is better than the current population average.

B.3 Aging

1. *Get Old* entities whose age is greater than the lifespan limit are eliminated from the system.
2. *Extended Life* Lifespan limit of an entity is extended by one iteration when it expires if its fitness is higher than the population average.

```
WHILE (number of entities > 0) and
      (number of iterations < limit)
    evaluate entity
    if (performance better than parent)
        positive feedback
        reproduce
        parent become inactive
    else
        negative feedback
        diffuse
        aging
    end-if
END-WHILE

Procedure Diffuse
    if rand() > P_rand_walk
        Random Walk
    else
        for each variable
            select step direction and size
        end-for
    end-if

Procedure Reproduce
    quota ← f(fitness)
    create new probability matrix
    for each offspring
        copy all variables
        create pointer to the new prob. matrix
        Diffuse
    end-for

Procedure Aging
    age ← age +1
    if fitness > parent's fitness * 1.25
        Remove
    else-if age > lifespan
        if fitness above average
            Keep
        else
            Remove
        end-if
    end-if
```

Fig. 1. Pseudocode for EDO

3. *Sudden Death* An unsuccessful entity will be eliminated if the entity's fitness is more than 25% of the parent's.
4. *Permanent removal* Any inactive entity not being rejuvenated will be removed permanently from the system.

B.4 Feedback

Two forms of feedback by offspring of an entity is used in EDO, which are used to update the probability matrix of an entity. The updated probability matrix is used by all offspring of the entity in question.
1. *Positive feedback* For positive reinforcement after a successful move, which may happen after taking many moves, the probabilities relating to the positive steps, negative steps and no change, are updated if the offspring's object variable is greater than, smaller than, or equals to the corresponding object variable of the parent.
2. *Negative feedback* Individual entities will update the parent's probability matrix after each move.
Note that all p' referred above are normalized using $\sum_i p'_i$ after updating.

C. Global Information Adaptation

EDO keeps several system-wide information that either controls the flow of the search or act as parameters to some of the features described above.

EDO adapts two of them to assist the optimization process. Firstly, the *Step_size* parameter, which is the unit of change effected on an object variable, is reduced according to the golden ratio if the current best has not been renewed for half of *Lifespan*, the maximum number of iterations an entity can stay in EDO without locating a 'better' location. Conversely, if the population has been improving continuously for some time (in number of iterations, again equal to half of *Lifespan*), the step size is increased according to the golden ratio (division).

The second adaptation strategy reduces the maximum population size by 0.1% at each iteration. This is to increase the selection pressure at the search advances so that only the highly successful entities are allowed to reproduce when it is very crowded.

III. Benchmark Problems

Four benchmark functions are chosen to test the EDO algorithm (see references within [11]). F1 and F2 are unimodal functions while F3 and F4 are multimodal functions. The unimodal functions are used to demonstrate the hill climbing capability of EDO and the multimodal functions are used to test EDO's ability to escape local minima.

$$f_1(x) = \sum_{i=1}^{n} x_i^2 \tag{3}$$

$$f_2(x) = \sum_{i=1}^{n} \left(\sum_{j=1}^{i} x_j \right)^2 \tag{4}$$

$$f_3(x) = \sum_{i=1}^{n} [x_i^2 - 10cos(2\pi x_i) + 10)] \tag{5}$$

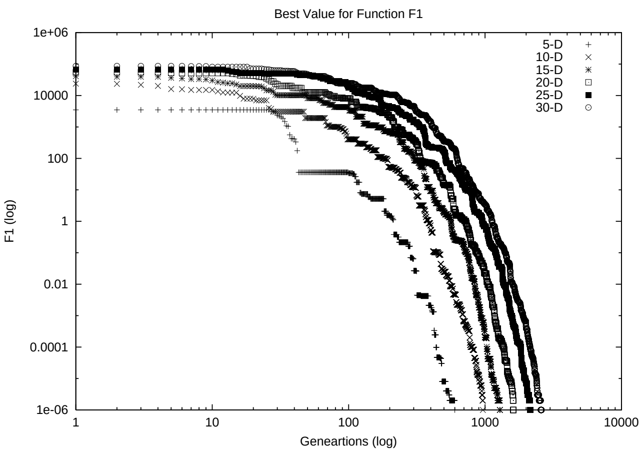
$$f_4(x) = \left[\frac{1}{500} + \sum_{j=1}^{25} \frac{1}{j + \sum_{i=1}^{2} (x_i - a_{ij})^6} \right]^{-1} \tag{6}$$

$$a_{ij} = \begin{pmatrix} -32 & -16 & 0 & 16 & 32 & -32 & \ldots & 0 & 16 & 32 \\ -32 & -32 & -32 & -32 & -32 & -16 & \ldots & 32 & 32 & 32 \end{pmatrix}$$

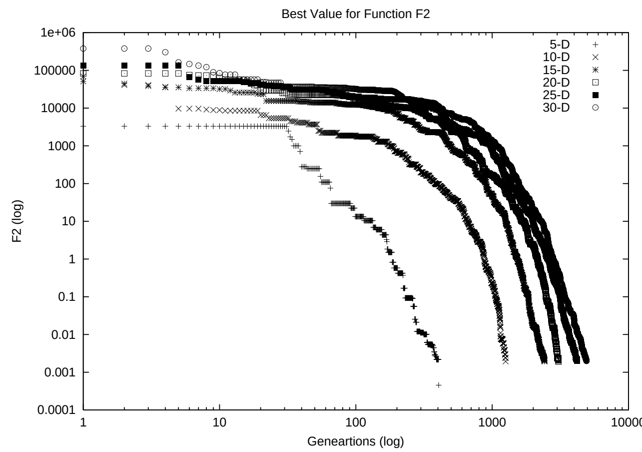
IV. Experimental Results

The EDO algorithm has been used to optimize the above four benchmark functions. Several forms of the functions, except function F4 (which has only two free parameters), are tested by increasing the number of free parameters. The intention is to test the performance and scalability of EDO.

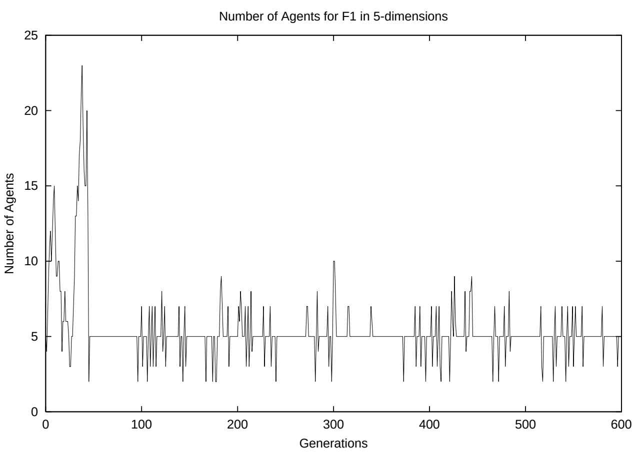
EDO has been tested on 6 F1 functions of 5 to 30 dimensions in increments of 5 dimensions. Figure 2(a) shows the development of the best solution over the iterations. In all the six experiments, EDO successfully found a solution with a value smaller than the target (1×10^{-6}). The time required increases from 600 to 4500 iterations



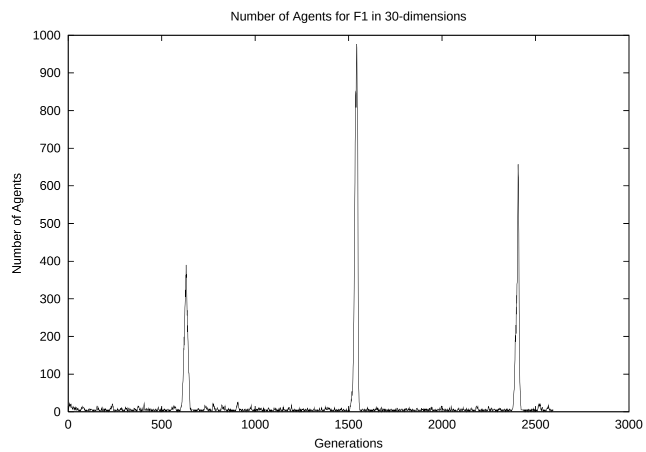
(a) Best solution F1 in 5- to 30-dimensions



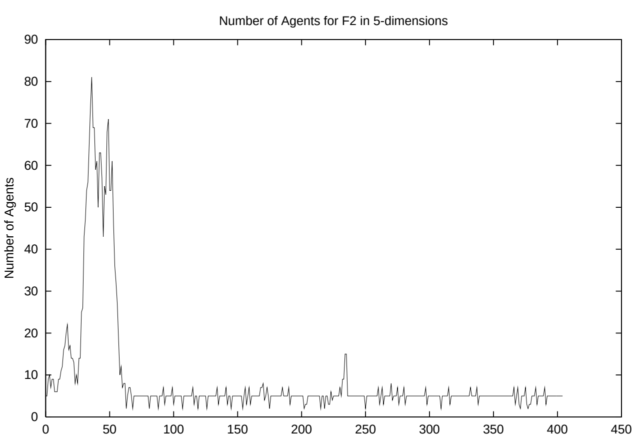
(b) Best solution F2 in 5- to 30-dimensions



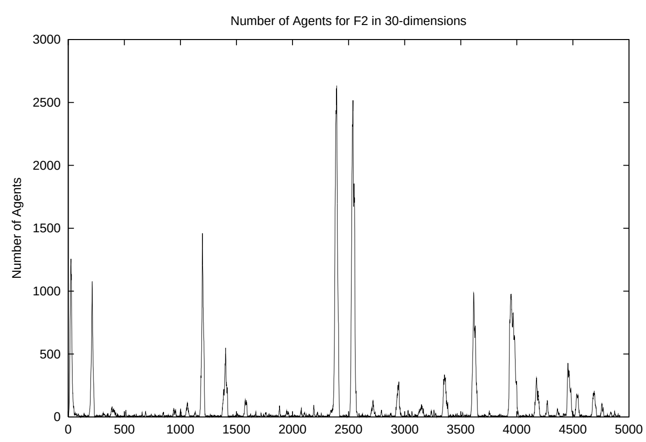
(c) Number of entities in 5-dimensions F1



(d) Number of entities in 30-dimensions F1



(e) Number of entities in 5-dimensions F2



(f) Number of entities in 30-dimensions F2

Fig. 2. Results of EDO on F1 and F2 in 5 to 30 dimensions

as the problem complexity increases from 5 dimensions to 30 dimensions. It can also be observed that lower dimension tasks (5-dimension) tend to result in more jumps, i.e. drastic drop in the best solution value after a period of no improvement. In contrast, the curves gradually become smoother as the dimension of the problem increases.

Figure 2(c) shows the number of entities present in the system at each iteration. During the initial stage (iteration 1-40), the number of entities has experienced two period of rapid increase. These are the periods when EDO discovered some solutions that are better than the randomly generated initial candidate solutions. However, when the offspring of these suboptimal solutions do not result in improvement, the number of entities decreases until a point when only the suboptimal solutions remains. The *solution rejuvenation* feature then kicks in to inject some candidate solutions to the population and preventing the search to end prematurely. The latest additions to the pool have a higher chance of success due to the negative feedback from previous failures. Thus the number of entities increases. This cycle of increase and decrease in population size continues until the maximum number of allowed iterations is reached. Figure 2(d) shows the same information for the 30-dimension F1 for comparison. Similar changes in the number of entities in the system can also be observed for the other four experiments.

The development of the best solution (see figure 2(b)) and the changes in the number of entities (see figure 2(e) and 2(f)) for F2 follow a pattern similar to that of F1. The best solutions achieved within the set time limit is around 0.002 and more iterations are generally required.

For the multimodal functions tested on EDO, the fitness landscape of F3 is more rugged that F4. Hence, better results are obtained from F4. The results of two different runs for F3 are shown in figure 3(a). In the 5-dimension F3 task, EDO found the optimal solution within 1000 iterations. The number of entities required follows a pattern similar to the situation when optimizing the unimodal function: the number of entities increase drastically when the search is homeing in to the optimal solution (beyond iteration 300, see figure 3(d)).

Increasing the dimension of the problem (F3) from 5 to 10 proves to be a difficult task for EDO. Figure 3(b) shows the changes in population average and best. While the search is able to locate suboptimal solutions and escaped from most of them, EDO fails to find the global optimal within the given time limit. Note that the population average fluctuates much more rapidly and sometimes even becomes very close to the population best. As the search progresses without making much improvement, the number of entities will decrease due to, for example, the expiry of lifespan. In the worst case, only those better performing parents remains. The only local behavior that can take effect in such a situation is 'solution rejuvenation', which will increase the number of entities in the population once again and will possibly cause the population average to increase as the new offspring diffuses from the parents' locations. The corresponding graph for the entity population size in figure 3(e) seems like a magnified version of figure 3(d). It is interesting to note that the period when EDO is making the best improvement (between iterations 500 and 1000), the population shows the least fluctuation and the number of entities is at the highest point.

EDO solved the easier F4 without any problem. Figure 3(c) shows the changes in the population average and best value over time. From the start to around 100 iterations, the population average is maintained at a relative stable level, well above the best value. However, as the entities struggle to escape the local minimum around 2 (y-axis), most of the entities were eliminated and a sudden drop in population average occurs. As EDO recovers, via *solution rejuvenation*, the population average reached the level similar to where it were before iteration 100. This cycle of fluctuation happens several times until the search reaches its final stage (between iteration 600 and 700). Notice from figure 3(f) that the number of entities required for this F4 run was very small except at the final stage where a drastic increase is observed. This probably is due to the convergence of most, if not all, entities around the global optimal. This convergent behavior has the side effect of causing the entities to reproduce. It is obvious that the absence of the downward going side of the spike is due to the search reaching the set target. If the search is allowed to proceed beyond this point, the number of entities will probably decrease due to the inability to find any solution better than the current best, which is the global optimal.

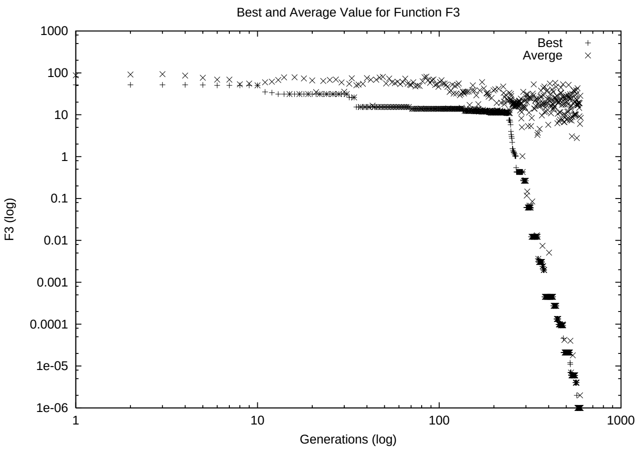
A. Performance Comparison

In order to benchmark the performance of EDO with some existing algorithms, some experiments were conducted on the benchmark functions of various complexity. The results comparing EDO to a simulated annealing algorithm (SA)[1] is presented in table I. The average number of steps (function evaluations) required to reach the target over 50 runs using different random seed is shown in the table along with the success rate, which measures the percentage of times when the method can reach the target. Both EDO and SA are considered failing to reach the target by comparing the function value achieved when they stopped, with the target. EDO stops either when the maximum number of iterations is reached, which is set at 5,000, or when the number of entities reach zero. SA stops when the maximum number of steps is reached, which is set at 2,000,000. The targets shown in table I are extremely close to the optimal solutions.

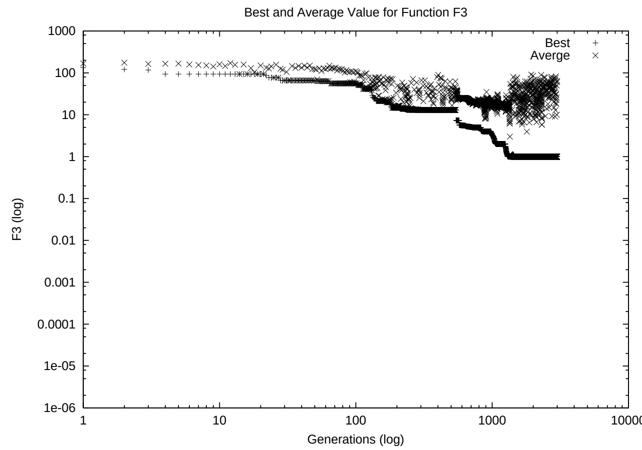
The results show the general trend that more complex problems requires more time (iterations) to solve. However, multimodal functions tend to be more difficult to solve than unimodal ones. It can be observed from table I that EDO performs better than SA in F1, F2 with lower dimensions and F4. As the complexity of the problem increase, EDO tends to fall behind. Moreover, EDO tends to be less robust than SA as it fails to reach the target in more cases than SA. While one has to wait until the maximum number of steps is reached to decide whether SA has reached the target, EDO generally stops within few tens of hundreds of iterations. Therefore, EDO allows a better chance to restart the search using another random seed.

The results achieved by EDO and the SA are also compared to that obtained using an EA algorithm - the Fast Evolutionary Programming (FEP) [11]. The average steps over 50 runs are presented in table II. The number of steps for FEP is calculated by multiplying the number of iteration reported by the number of individuals in the population in each iterations. The results show that EDO performs better than FEP in F1, and F4 while SA perform better than FEP in all cases except F4. It should be

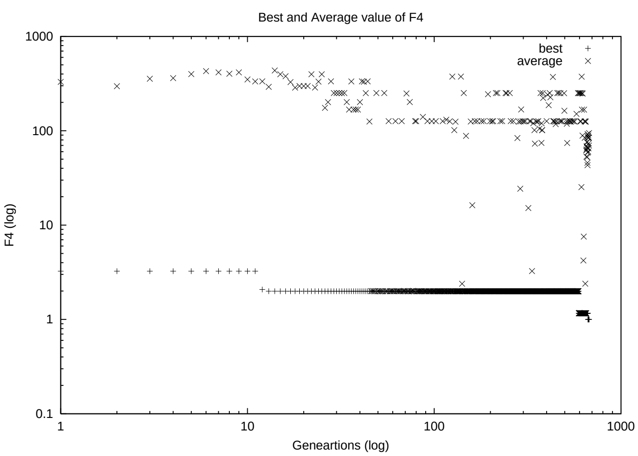
[1] source code obtained from http://www.taygeta.com/annealing



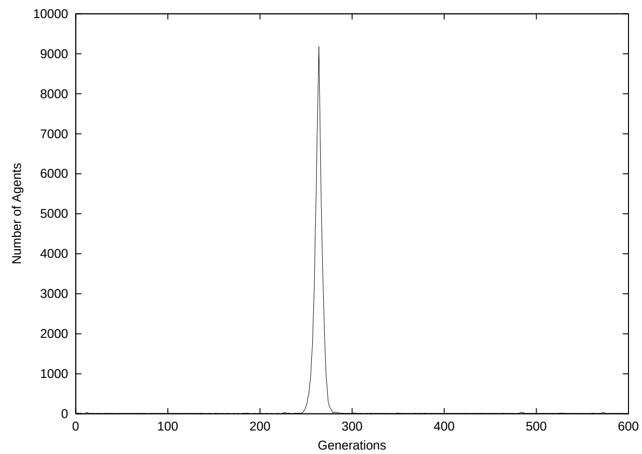
(a) Best/average: 5-D F3



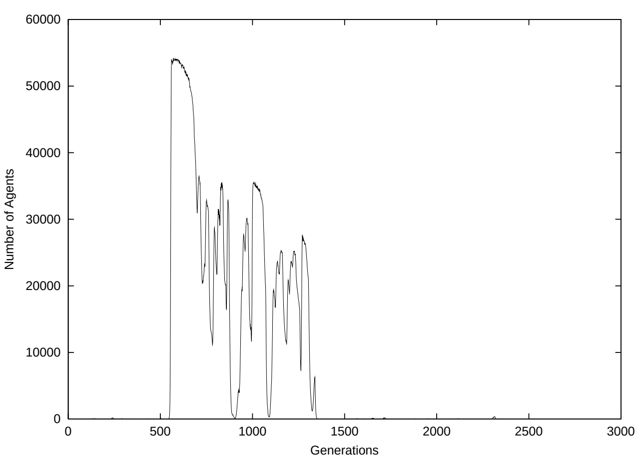
(b) Best/average: 10-D F3



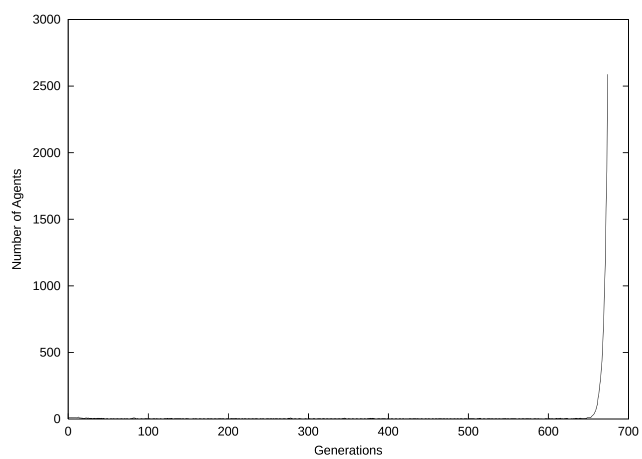
(c) Best/average: F4



(d) Entities: 5-D F3



(e) Entities: 10-D F3



(f) Entities: F4

Fig. 3. Best and average value, and number of entities used in each iteration for F3 and F4.

TABLE I

Average number of steps (over 50 runs) required by EDO and SA to optimize the benchmark functions of various complexity and the success rate of both methods

	Target	EDO			SA		
		Average	Std Dev.	Success	Average	Std. Dev.	Success
F1_5	1.0×10^{-6}	2,000	1,168	96%	5,235	2,828	100%
F1_10		3,240	5,017	94%	26,355	12,144	100%
F1_15		6,816	10,234	88%	61,209	28,316	100%
F1_20		17,277	31,174	84%	112,261	58,460	100%
F1_25		26,683	24,031	84%	178,062	82,119	92%
F1_30		167,094	878,892	84%	235,071	103,478	100%
F2_5	2.0×10^{-3}	1,705	1,472	90%	13,701	72,864	100%
F2_10		7,710	11,875	96%	9,302	27,483	96%
F2_15		29,328	38,789	90%	10,725	16,801	90%
F2_20		103,910	141,580	94%	68,189	173,688	82%
F2_25		227,959	490,607	90%	35,689	50877	92%
F2_30		2,224,903	4,558,546	94%	91,098	181,442	82%
F3_5	1.0×10^{-6}	15,687,788	5,812,865	74%	101,468	61,699	94%
F4	0.998004	12,172	48,182	90%	78,441	251,730	58%

TABLE II

The average number of steps (over 50 runs) required by EDO and SA to reach the target achieved by FEP

	target	EDO	SA	FEP
F1_30	5.7×10^{-4}	49,205	32,947	150,000
F2_30	1.6×10^{-2}	4,537,359	23,411	500,000
F3_5	0.14	6,799,883	1,594	40,000
F4	1.22	907	79,085	10,000

pointed out that FEP was reported to have reached that target in all 50 runs while both EDO and SA did not. This shows that FEP is more robust than both of them.

V. Conclusions

This article has described two sets of experiments on four benchmark functions - two unimodal and two multimodal. EDO performs better than SA in two of them while SA excels in the rest two. Similar conclusion can be drawn from the results between EDO and FEP. However, EDO seems to be less robust than SA in that it fails to reach the target in more times than SA among those 50 runs for each case. In all cases, FEP was reported to have found a solution.

Central to the operation of EDO is the probability matrix. Further work is planned to fully explore the usefulness of the current representation and to investigate whether there are better alternative representations. It is also worth investigating how far back should the feedback goes. Ideas from some reinforcement algorithms such as the bucket brigade algorithm and the Q-learning algorithm, which take into account all feedback in future but with different weight, may be useful here.

Despite efforts by *diffusion* and *reproduction* to eliminate duplicates, EDO still requires an extremely large number of iterations to solve some problems. It warrants a closer look at the way EDO generates and manages its population of entities. Perhaps some kind of clustering algorithm that group 'similar' entities together will help EDO to allocate its new entities in a better way. Some researchers suggest maintaining a set of system parameters, such as step size, for a specific segment of the search space as it will be more suitable for the landscape. This idea ties in well with the clustering issued mentioned above.

References

[1] Aimo Torn and Antanas Zilinskas, *Global Optimization*, Sringer-Verlag, 1989.

[2] Reiner Horst and Panos M. Pardalos, Eds., *Handbook of Global Optimization*, Kluwer Academic Publishers, 1995.

[3] S. Kirkpatrick, C.D. Gelatt, and M.P. AndVecchi, "Optimization by simulated annealing," *Science*, vol. 220, no. 4598, pp. 671 – 680, 1983.

[4] Thomas Bäck, David B Fogel, and Zbigniew Michalewicz, Eds., *Handbook of Evolutionary Computation*, Institute of Physics Publishing and Oxford University Press, 1997.

[5] James Kennedy, "The particle swarm: Social adaptation of knowledge," in *Proceedings of the 1997 International Conference on Evolutionary Computation*, Piscataway, NJ, 1997, pp. 303–308, IEEE Serice Center.

[6] R. G. Reynolds, "An introduction to cultural algorithms," in *Proceedings of the 3rd Annual Conference on Evolutionary Programming*, A. V. Sebald and L. J. Fogel, Eds. 1994, pp. 131–139, World Scientific, River Edge, NJ.

[7] Marco Dorigo, Vittorio Maniezzo, and Alberto Colorni, "The ant system: Optimization by a colony of cooperative agents," *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, vol. 26, no. 1, pp. 1–13, 1996.

[8] Sarah Witham Bednarz, "Mission geography," http://geog.tamu.edu/sarah/humangeog/migration8.html, 2000.

[9] E. L. Cussler, *Diffusion : Mass Transfer in Fluid Systems*, Cambridge University Press, 1997.

[10] Kwok Ching Tsui and Jiming Liu, "Evolutionary diffusion optimization, part i: Description of the algorithm," in *Proceedings of CEC'02*, 2002, to appear.

[11] X. Yao, Y. Liu, and G. Lin, "Evolutionary programming made faster," *IEEE Transactions on Evolutionary Computation*, vol. 3, no. 2, pp. 82–102, July 1999.

[12] David H. Wolpert and William G. Macready, "No free lunch theorems for optimization," *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 1, pp. 67–82, 1997.