Evolutionary Diffusion Optimization, Part I: Description of the Algorithm

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 - This article proposes a new population-based stochastic search algorithm called *Evolutionary Diffusion Optimization* (EDO) inspired by diffusion in nature. Each entity in EDO makes the decision to diffuse based on the information shared between its parent and its siblings. The behavior of EDO when solving a typical optimization problem is also discussed.

I. Introduction

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

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

The search for x^* can be viewed as the minimization of function F. Turning the sign in equation 1 around makes the search for x^* a maximization task. They can collectively be called global optimization tasks [1].

Many algorithms have been developed over the years to tackle the challenging task of global optimization [2], [3], [4] . There are several aspects to the challenge any new search algorithm will face. Firstly, the landscape of the function to be optimized is unknown. Unimodal functions can be monotonic in nature and the search is easy once the downhill 'direction' is found. Multimodal functions, on the other hand, have many suboptimal solutions where a search algorithm is likely to be trapped.

Secondly, there is usually no linear relationship between changes made to the function variables and the corresponding change in the function value.

This article proposes a new population-based stochastic search algorithm called *evolutionary diffusion optimization* (EDO) inspired by diffusion in nature. 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, candidate solutions are changed based on information obtained in its proximity. The right to reproduce is also granted based on local competition. An accompanying article compares in more detail the performance of EDO with Simulated Annealing and Fast Evolutionary Programming in optimizing for functions [5].

Similar to simulated annealing [6], EDO is a *stochastic search algorithm* as it employs an optimization policy that do not make the same decision all the time. It is also a population-based search algorithms as many candidate solutions are present during an iteration. Unlike evolutionary algorithms [7], particle swarm [8], cultural algorithm [9] and ant colony optimization [10] that keeps a fixed-sized population, the number of entities in EDO varies at the search progresses.

II. Diffusion in Nature

The study of human geography categorized reasons for human migration into push factors and pull factors [11], [12]. . Push factors relate to undesirable conditions such as poor living conditions, lack of job and overcrowding. Pull factors are those positive factors that attracts people to relocate, such as jobs and better living conditions. Two forms of migration can also be identified. Step migration refers to a series of local movements, such as moving from village to town, then to city. Chain migration refers to a more drastic change beyond the local region and is usually assisted by people who have already emigrated. The availability of information seems to be an important factor that help people to decide when and where to migrate.

The important lesson to learn for function optimization is that once the landscape of the function to be optimized is known, finding the optimal solution becomes trivial. The question is how to capture the *'trend'*. A search algorithm will need past experience to inform it of the possible successful moves (push factor) and unsuccessful moves (pull factor).

In addition, 'Knowledgeable' people can help others to decide whether to move or not. In essence, whoever has captured the 'trend' of the search space can help others to make 'better' moves by *sharing their knowledge*. On the other hand, as the human migration model shows, the chance of finding a solution can possibly be enhanced by making small *'step'* migrations.

Building on the above observations, the following section describes the evolutionary diffusion optimization that has facility to *learn the trend* of the search space, while *forming chains of information flow* to *facilitate small step exploration* of the search space.

III. The EDO Algorithm

A. Basic Data Structure

A.1 Object Vector

An entity in EDO represents a candidate solution to the optimization problem in hand. Whether a variable is modified by EDO during a particular iteration is dictated by a probability matrix, which contain probabilities for making changes as well as remain unchanged.

A.2 Probability Matrix

Each entity has a probability matrix associated with it. The matrix contains the likelihood estimate of success with respect to direction of motion. Specifically, the probability matrix is an $m \times n$ matrix representing the m object variables to be optimized and n possible steps of motions, i.e. $(n-1)/2$ steps towards the upper bound and the lower bound respectively, and remain stationary. At the beginning of the search, all entries are initialized to the $1/n$, which means it is equally likely to make any of the possible moves. The probability matrix will be continuously updated as a result of the local feedback behavior (see section III-B.4) of an entity. As the probabilities for various steps are updated, EDO begins to differentiate between 'good' and 'bad' directions of move with respect to the initial position of the ancestor. The probability matrix, therefore, carries some kind of 'heritage' throughout the family line of those entities that were present when the search begins.

Parents and their offspring share the same probability matrix. It is only when an entity becomes a parent then a probability matrix will be created for it, which is an exact copy of the parent's updated matrix. Sharing the probability matrix between parents and siblings enables entities from the same family to learn from each other's successes as well as failures. Details of this feedback sharing feature is discussed in more detail below.

A.3 Fitness

Fitness in EDO measures an entity's degree of success in the course of the search for the optimal solution. It is also used as a basis to determine various local behaviors. For simplicity, the objective function value is used as fitness in EDO if the task is minimization. The reciprocal of the function value can be used as fitness if EDO is used in a maximization task.

B. Local Behaviorism

All entities in EDO are considered *active* and will engage in the behaviors of *diffuse* and *feedback*. However, when those entities that have diffused to a location with function value better than their parents, they will *reproduce* and give *feedback* again to their parents. At the end of these additional behaviors, these 'successful' entities will become *inactive* - will no diffuse any more. The following sections describe the aforementioned behaviors in more detail. Figure 1 shows the pseudocode of the major steps.

B.1 Diffuse

Entities in EDO explore uncharted locations in the solution space by diffusion. In the majority of time, the entities will choose a new location according to the probability distribution given by the probability matrix. As an entity becomes older and still have not located a 'better' place, it will probabilistically decide to take a random walk according to the following equation:

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

where α is a scaling factor that decides the degree to which random walk is to be exercised. If the entity remains stationery after diffusion, the process will be repeated. In more detail, the two types of diffusion an entity can take are described below.

1. *Rational Move* An entity modifies its object values by drawing a random number for each dimension of the object vector. Each random number is then used to choose between the set of fixed steps according to the probability matrix. A value equal to the product of the size of a step and the number of steps value is added to the entry in question in the object vector. This process is repeated until all dimensions of the object vector are covered. The updated object vector becomes the new position of the entity in the solution space.

2. *Random Walk* The direction of movement is first chosen uniformly randomly, i.e towards upper bound, towards lower bound and no move. In case a move is to be made, a new value between the limit in the chosen direction and the current value is then chosen randomly. The process ends when all the dimensions of the object vector are updated.

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

B.2 Reproduce

At the end of an iteration, the fitness of all active entities are compared with that of their parents, which have temporarily become stationary. All entities with higher fitness are allowed to reproduce by asexual reproduction. A reproducing entity will replicate itself the number of times. The new entities are then sent off to a new location by *rational move*. In the event duplicates resulted during reproduction, all except one are removed.

The number of offspring to reproduce is governed by the system-wide parameter and the following two rules:

1. *Differentiate Reproduction Quota* An entity is given the full quota to reproduce only if its fitness is 25% of the population average. Otherwise, two less than the quota is given for those in the lower 25%; one less than the quota applies to the rest.

2. *Restriction on Reproduction Quota* Further to the

above differential scheme, the reproduction quota is subjected to further restriction due to overcrowding.

$$quota' = \lfloor (quota * (pop_{max} - pop_{current})/pop_{max}) \rfloor \quad (3)$$

3. *Rejuvenate* An inactive entity will be allowed to spawn new offsprings if the following two conditions are satisfied:
(a) all its offspring are dead;
(b) its fitness is better than the population average.
The main idea behind *rejuvenate* is that the inactive entity has been receiving positive as well as negative reinforcement signals from its offspring (described below), its probability matrix contains the latest information regarding the neighborhood landscape. It would be a waste if this potentially useful information is discarded.

B.3 Aging

1. *Get Old* The normal lifespan of the entities is governed by a system-wide parameter. At the end of each iteration, the age of all entities is increased by one. Entities whose age is greater than the lifespan limit are eliminated from the system unless the following condition (extended life) is satisfied.
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.
3. *Sudden Death* An unsuccessful entity will be eliminated if the entity's fitness is more than 25% of the parent's.

$$(fitness/parent'sfitness) > 1.25 \quad (4)$$

4. *Permanent removal* Any inactive entity not being rejuvenated will be removed permanently from the system.

B.4 Feedback

All entities in EDO pass information back to their parents every time it has taken a 'good' move to a 'better' location or 'bad' move. This information allows the parent to update its probability matrix, which is shared among its offspring.
1. *Positive feedback* 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 equal to the corresponding object variable of the parent. The update rule is:

$$p'_{ij} = (p_{ij} + \delta)/(1 + \delta), \quad (5)$$

p_{ij} is the probability that relates relates to the ith object variable and jth step size.
2. *Negative feedback* Individual entities will update the parent's probability matrix after each move after each unsuccessful move. The parent in turn will update those probabilities in question using:

$$p'_{ij} = p_{ij} \times (1 - \delta). \quad (6)$$

This is a finer grain update than the positive reinforcement signal as it happens after each step. Moreover, the use of a multiplicative scaling factor ensures that the probability remains bigger than zero at all times. Note that all p' referred above are normalized using $\sum_i p'_i$ after updating.

C. Global Information and its 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. The four parameters are:
• *Offspring_to_reproduce*: Subject to other restrictions, this is the maximum number of offspring any reproducing entity is allowed to spawn at one time.
• *Max_pop_size*: This is the upper limit of the number of entities, both active and inactive together, an EDO session can keep.
• *Lifespan*: This duration, in iterations, given to an entity to perform search
• *Step_size*: The amount of change to an object variable. It is used in combination with the probability matrix to decide what should the actual change be. For example, if the two steps toward the upper limit were chosen based on the probability distribution in the probability matrix, a value equal to two times of *Step_size* parameter will be added to the object variable in question.

EDO has some adaptive behavior to assist the optimization process. Firstly, the *Step_size* parameter is reduced according to the golden ratio if the current best has not been renewed for half of *Lifespan* (in number of iterations). The rationale behind this reduction is that the entities may be in the neighborhood of a minimum. Therefore, finer steps are required for careful exploitation. 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).

EDO also reduces the maximum population size by 0.1% at each iteration to increase the selection pressure as the search advances.

IV. Discussion

Using the most basic form (two-dimensional) of $F_1(x) = \sum_{i=1}^{n} x_i^2$ as a testbed, this section will present the behavior of the more important features of EDO outlined above.

The basic form of the evolutionary diffusion optimization (EDO) consists of local behaviors such as diffusion, reproduction, aging and feedback. Some advanced features such as random walk during diffusion, step size adaptation and solution rejuvenation are also available to EDO. Various combinations of the basic EDO with the advanced features are used to optimize the two-dimensional version of F1 for 200 iterations. Figure 2 shows the best and average function value for the basic EDO together with one or more of the three advanced features.

The basic EDO is able to a find a 'good enough' solution, which has a value less than 1, in just 10 iterations (see figure 2(a)). This is a period when the diversity in terms of the variety in the parameter values is at the highest. The population converges gradually during this period. As the diversity decreases, EDO tends to try out small variations to the candidate solutions in the population and attempts to update the probability matrix. As the population moves together closer to the optimal solution, a fixed step size will gradually move the candidate solutions across the optimal point and over to the other size of the 'valley' in the function value landscape. In combination with the fact that EDO accepts all new points into the solution pool, the population average will diverge at certain point. In addition, whatever direction biasing information learned in the probability matrix will suddenly become irrelevant. A new 'learning' phase then commences (between iterations 40-50). This cycle of learning and re-learning can also be observed at later stage of the search (between iterations 100 and 200).

Another point worth noting is that new best solution is not discovered at every iteration. One might expect that once a trend has been learnt and encoded into the probability matrix, any new entity having the latest probability matrix will simply slide downwards towards to the optimal solution. This is not found in any of the plots in figure 2. This is because new candidate solutions are generated using the probability matrix and the step size. However, this fixed size is becoming larger and larger relative to the distance between the best solution and the optimal solution. With a fine tuning mechanism such as the adaptive step size, this problem is eliminated (see figure 2(b)). It can also be observed that the adaptive step size feature produces a population that is converging at a faster rate than the basic EDO. This is another evidence of efficiency gain in utilizing the probability matrix.

The random walk feature (figure 2(c)) seems to be making a more progressive improvement on the best solution than the basic EDO. A distinctive feature of random walk is that it produces a non-converging, if not diverging, population. Notice that the tail of the population average graph is diverging and the search finished before the set limit, at 159 iterations, as the number of entities has dropped to zero. This is somewhat undesirable as the search has become more unpredictable but this side effect vanishes when random walk is combined with other features.

The solution rejuvenation feature (figure 2(d)) alone performs almost exactly as the basic EDO. It seems this feature is not making significant contribution in the optimization of a unimodal function.

The combination of any two advanced features shows some interesting results when optimizing the two-dimensional F1. Adaptive step size continue to show the best performance and shadowed the contribution of solution rejuvenation (figure 2(e)). A converging population similar to, but slight different from, the one in figure 2(b) is also observed.

The initial stage of the the random walk and adaptive step size combination (figure 2(f)) is very similar to the random walk only search. But the later stage of the search using both features shows a very steady improvement. However, the population is no longer converging and less fluctuation is observed. It can be concluded that step size adaptation helps to exploit existing solutions while random walk is exploring new areas in the solution space. This is the most important findings of this initial testing.

The combination of random walk and solution rejuvenation (figure 2(g)) shows a profile similar to random walk alone. However, the population value at the second half of the search fluctuations instead of simply diverging. This is due to the fact that the few good solutions are not eliminated from the system and allowed to reproduce beyond the time when are freely active entities have become extinct.

When all three advanced features are used, EDO achieves the smallest value and the population shows a slight converging behavior (figure 2(h)). If the function value is chosen as the evaluation criteria of the advanced features when acted alone, adaptive step size achieves the best solution followed by random walk. When more than one feature is considered, all three advanced features together has a slight advantage over adaptive step size alone.

Figure 3 shows the distribution of entities used to optimize the 2 dimensional version of F1. It shows the behavior of the entities from a different perspective. Five entities were positioned randomly on the solution space - three with x-value around -80, one around -20 and one around zero - but none is within the central 20 by 20 zone around the origin. As the search progresses to iteration 10, more and more entities appears inside the central zone. The population size increase is due to the success of the entities around there, which *reproduces* quickly; providing some positive feedback to their parents. This trend continues between iteration 10 and iteration 80. Note that there also are entities throughout the solution space. They were put there by *rational moves* with by step size or by *random moves*.

The diameter of the occupied area increases between iteration 10 and 30, and shrinks between iteration 30 and 80. The entities have learnt the 'direction' of the fitness landscape in the first period and zoom in to the origin (the optimal solution) in the second period. The third observation worth mentioning is the increase in the number of entities along the axes, which can be considered suboptimal solutions as one of the two variables is in its lowest value. This is another evidence to show that learning has happened.

The number of entities is at its highest point around iteration 80 but dropped drastically in the next 10 iterations. This is because entities not around the origin or along the axes have much higher fitness than the population average. The second phase of the search sees entities along the axes are eliminated as they are occupying a suboptimal position. As the entities continue to move towards the origin, entities that are further away from the origin are eliminated: In iteration 120, four of the five entities are located within the range ± 0.5 around the origin. In iteration 160, all five entities are inside the ± 0.06 range. In iteration 200, four of the six entities are inside the range ± 0.02. The last iteration, 220, has 11 of its 14 entities inside the ± 0.01 range.

V. Related Work

Several well-known search algorithms share some common features with EDO. This section attempts to highlight some of their major differences.

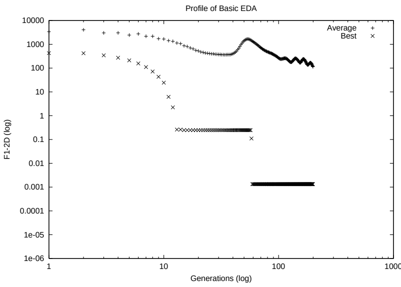
A. Simulated Annealing

Apart from the difference in population size, both SA and EDO maintains a schedule for modifying the system parameters. In SA, the temperature of the 'cooling' process is predetermined to go down gradually. The counterpart in EDO is the modification to the step size. However, EDO's step size can go up as well as down depending on the progress over a period of time.

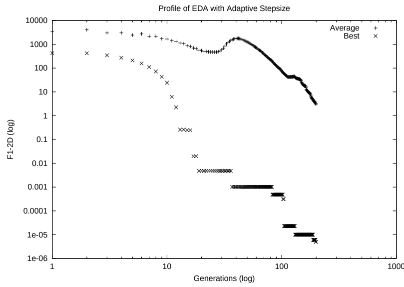
Another major difference between SA and EDO lies in the way they accept worse candidate solution as the starting point of the next iteration. While SA will accept this candidate based on certain probability, EDO will always accept it. The aging mechanism in EDO allows the system to get rid of poor performers after they are consistently performing badly over a period of time.

B. Evolutionary Algorithms

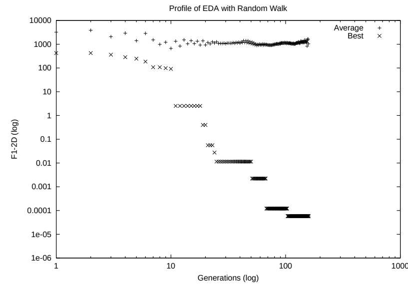
There are many strands in the class of evolutionary algorithms, which are mostly population-based. However, most of them maintain a fairly common framework based on the 'survival of the fittest' principle. They differ in the



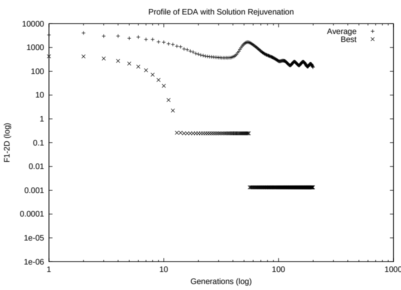
(a) Basic EDO



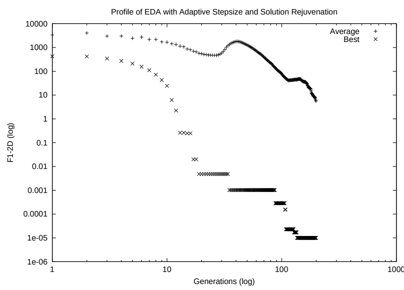
(b) with adaptive stepsize



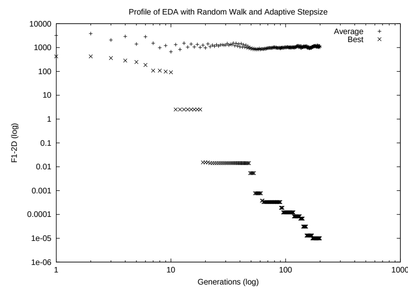
(c) with random walk



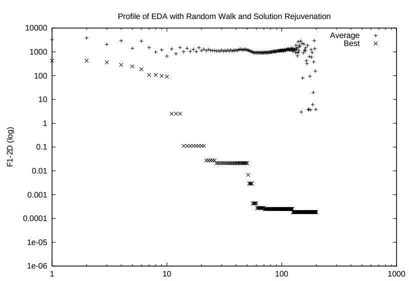
(d) with solution rejuvenation



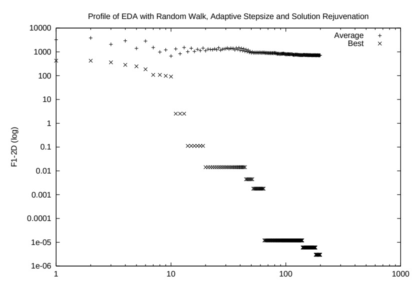
(e) with adaptive stepsize and solution rejuvenation



(f) with random walk and adaptive stepsize



(g) with random walk and solution rejuvenation



(h) with random walk, adaptive stepsize and solution rejuvenation

Fig. 2. Profile of the development of a solution on F1 using various forms of EDO

choice of the so called genetic operators and in implementation of the algorithm. EDO is different from EA in a few places.

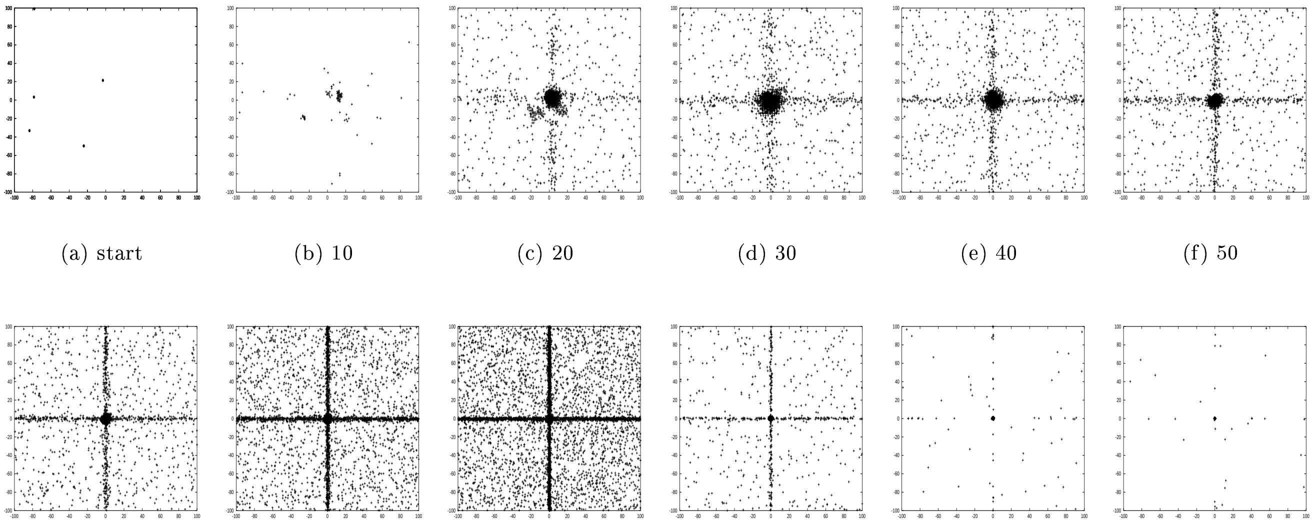
Firstly, the population size of EA is usually fixed while EDO maintains an open-ended approach. Such an approach allows the weaker entities that survive to the next iteration to wander to a better position, and hence free from being trapped by local suboptimal.

Secondly, a variable mutation rate, realized by the diffusion step size, allows EDO to migrate smoothly between exploration and exploitation. Moreover, most EA give high priority to making small changes rather than large changes. The latter may be able to use fewer time steps for search if larger steps are possible.

Thirdly, selection in EA is a population-wide operation where all entities in the population are sorted according to fitness. However, selection in EDO is entirely local, which compares only the performance of an entity and its parent. This local selection process is computationally less costly and offers better chance for parallel implementation.

Fourthly, the probability matrix in EDO allows entities to capture the trend of the fitness landscape and points further search towards a potentially fruitful direction. This information is again local to an entity and only shared by an entity's offspring. In contrast, any trend in EA can only be captured by observing all entities in the population.

Fifthly, the chance to reproduce and remain in the population decreases with the entity's fitness in the case of EA. In contrast, the notion of success is limited to the local sense in EDO. This allows more diversity in the population. Finally, random walk plays a stronger role in EDO



(a) start (b) 10 (c) 20 (d) 30 (e) 40 (f) 50

(g) 60 (h) 70 (i) 80 (j) 90 (k) 100 (l) 110

Fig. 3. Distribution of entities in the solution space for 2-D F1

than in EA as an instrument to escape suboptimal.

C. Particle Swarm

Particle swarm is similar to EDO in that individual particles make their own local decisions without consulting any other particles except one. In more details, individual particles requires two pieces of information to make the decision regarding the next position to assume: the velocity (direction and speed) of the last move and that of the overall best particle. Therefore, some kind of global information is always required. Although communication between entities of the same lineage is required in EDO, it does not depend on any global information.

D. Ant Colony Optimization

ACO is an optimization algorithm based on the principle of autocatalysis, where shared knowledge is built incrementally and is implicit. Knowledge sharing in EDO is explicit and local, and is achieved via the probability matrix, which is used and updated by all offspring of an entity.

E. Cultural Algorithm

Human being passes experience and knowledge to the next generation and beyond by means of culture. This form of knowledge sharing has been captured in the cultural algorithm. EDO also takes the view that knowledge gained by one entity is useful to others. As such, direction of change is passed to others via the probability matrix, which is consumed by entities within the 'family'. This is particularly useful in a population-based search algorithm where many sites in the fitness landscape are being explored in parallel as direction information in one site is usually irrelevant to another site.

VI. Conclusions

This article has described a new local search algorithm called *evolutionary diffusion optimization* (EDO) for global optimization. Initial experiments reveal that EDO is able to automatically maintain a good balance between exploration and exploitation. More experiments need to be run to assess the performance of EDO in terms of scalability and robustness, some of which are reported in an accompanying article [5].

References

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

[2] Reiner Horst and Hoang Tuy, *Global Optimization : Deterministic Approaches*, Springer-Verlag, 1990.

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

[4] Jonas Mockus, *Bayesian Approach to Global Optimization : Theory and Applications*, Kluwer Academic, 1989.

[5] Kwok Ching Tsui and Jiming Liu, "Evolutionary diffusion optimization, part II: Performance assessment," in *Proceedings of CEC'02*, 2002, to appear.

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

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

[8] 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.

[9] 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.

[10] 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.

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

[12] W. A. V. Clark, *Human Migration*, Sage Publications, 1986.