INSENSITIVE DIFFERENTIAL EVOLUTION AND MULTI-SOLUTION PROBLEMS

Itsuki Handa and Toshimichi Saito
Faculty of Science and Engineering, Hosei University, 184-8584 Tokyo, Japan

Keywords: Swarm inteligence, Differential evolution, Multi-solution problems.

Abstract: This paper presents an insensitive differential evolution for multi-solution problems. The algorithm consists of global and local searches. In the global search, the algorithm tries to construct local sub-regions (LSRs) each of which includes either solution. In the local search, the algorithm operates on all the LSRs in parallel and tries to find all the approximate solutions. The algorithm has a key parameter that controls the algorithm insensitivity. If the insensitivity is suitable, the algorithm can construct all the LSRs before trapping into either solution and can find all the solutions. Performing basic numerical experiments where parameters are adjusted by trial-and-errors, basic performance of the algorithm is investigated.

1 INTRODUCTION

Differential evolution (DE) is a population-based search strategy in the evolutionary algorithms (Storn and Price, 1996) (Storn and Price, 1997) (Storn and Price, 1995). The DE has particles corresponding to potential solutions. The particle location is updated by a simple difference equation in order to approach the optimal solution (Takahama and Sakai, 2004)(Takahama and Sakai, 2006). The DE is simple, does not require differentiability of the objective functions and has been applied to various problems (non-convex, a multi-peak etc.). The engineering applications are many and include optimal parameter setting of circuits and systems: analog-to-digital converters (Lampinen and Vainio, 2001), digital filters (Luitel and Venayagamoorthy, 2008), switching power converters (Huang et al., 2004), etc.

This paper presents an insensitive differential evolution (IDE) for multi-solution problems (MSP). The IDE consists of the global and local searches. In the global search, the IDE tries to construct local subregions (LSRs) each of which includes either solution. The IDE has two key parameters: ε controls insensitivity for update of particle position and T_G controls switching timing to the local search. As the parameter ε is small, the algorithm operates similarly to classical DE where almost all particles tend to converge to either solution. In such a case, it is hard to maintain a diversity for successful construction of all the LSRs. As the parameter ε increases, the algorithm insensitivity increases and all the LSRs can be constructed before trapping into either solution. The global search is stopped at time T_G and the algorithm is switched into the local search. In the local search, the IDE operates on all the LSRs in parallel and tries to find all the approximate solutions. If the global search runs successfully and can construct all the LSRs, the local search can find all the solutions. In the algorithm, tuning of the two parameters ε and T_G is very important. Performing basic numerical experiments with trial-and-errors of the parameters tuning, we have investigated the algorithm performance in several typical measures such as success rate.

In several evolutionary optimization algorithms including DEs, escape from a trap of either solution is a basic issue of the MSPs. The traps relates deeply to local minima of unique solution problems. In order to avoid the trap, there exist several strategies including the tabu search (Li and Zhao, 2010). We believe that our insensitive method is simpler than existing methods and can be developed into an effective algorithm for the MSPs. This paper provides basic information to develop such an algorithm.

2 INSENSITIVE DIFFERENTIAL EVOLUTION

We define the algorithm IDE for m-dimensional objective functions.

Handa I. and Saito T..
INSENSITIVE DIFFERENTIAL EVOLUTION AND MULTI-SOLUTION PROBLEMS.
DOI: 10.5220/0003653002920295
In *Proceedings of the International Conference on Evolutionary Computation Theory and Applications* (ECTA-2011), pages 292-295
ISBN: 978-989-8425-83-6

$$f(\vec{x}) \geq 0,\ \vec{x} \equiv (x_1, x_2, \cdots, x_m) \in R^m \qquad (1)$$

where the minimum value is normalized as 0. This positive definite function has multiple solutions in the search space S_0:

$$f(\vec{x}_s^i) = 0,\ i = 1 \sim N_s,\ \vec{x}_s^i = (x_{s1}^i, \cdots, x_{sm}^i) \in S_0$$
$$S_0 = \{\boldsymbol{x} |\ |x_1| \leq A, \cdots,\ |x_m| \leq A\}$$

where $\boldsymbol{x}_s^i$ is the i-th solution ($i = 1 \sim N_s$) and N_s is the number of solutions. The search space is normalized as the center at the original with width A. The IDE has N pieces of particles whose search is characterized by the position vectors:

$$X = \{\vec{x}_1, \cdots, \vec{x}_N\},\ \vec{x}_i \equiv (x_{i1}, \cdots, x_{im}) \qquad (2)$$

where $i = 1 \sim N$. The vector $\vec{x}_i$ is a potential solution and is desired to approach either solution. The IDE consists of two stages. The first stage is the global search that tries to construct the LSRs each of which includes either solution. The second stage is the local search that tries to find the approximate solution in all the LSRs.

2.1 Global Search

Let t_1 denote the iteration number. The algorithm is defined by the following 5 steps.

Step 1 (Initialization). Let $t_1 = 0$. The particles $\vec{x}_i(0)$ ($i = 1 \sim N$) are initialized randomly in S_0.

Step 2 (Mutation). Three vectors $\vec{x}_{p1}(t_1)$, $\vec{x}_{p2}(t_1)$ and $\vec{x}_{p3}(t_1)$ are selected randomly from the set of particles X where $\vec{x_{p1}}(t_1) \neq \vec{x_{p2}}(t_1) \neq \vec{x_{p3}}(t_1)$ is assumed. A candidate vector $\vec{y}_i(t_1)$ is made by

$$\vec{y}_i(t_1) = \vec{x}_{p1}(t_1) + B(\vec{x}_{p2}(t_1) - \vec{x}_{p3}(t_1)) \qquad (3)$$

where B is the scaling parameter.

Step 3 (Crossover). Applying crossover for the candidate vector $\vec{y}_i \equiv (y_{i1}, \cdots, y_{im})$ and the parent $\vec{x}_i \equiv (x_{i1}, \cdots, x_{im})$, we obtain an offspring $\vec{c}_i$:

$$(c_{i1}, \cdots, c_{i(j-1)}) = (x_{i1}, \cdots, x_{i(j-1)})$$
$$c_{ij} = y_{ij}$$
$$(c_{i(j+1)}, \cdots, c_{im}) = \begin{cases} (y_{i(j+1)}, \cdots, y_{im}) : \text{ rate } P_c \\ (x_{i(j+1)}, \cdots, x_{im}) : \text{ rate } 1 - P_c \end{cases} \qquad (4)$$

where $i = 1 \sim N$ and P_c is the crossover probability. The crossover point j is selected randomly from all particle subscripts $\{1, \cdots, m\}$.

Step 4 (Survival). The parent $\vec{x}_i(t_1)$ is compared with the offspring $\vec{c}_i(t_1)$ and is updated as the following:

$$\begin{array}{ll} \vec{x}_i(t_1) = \vec{c}_i(t_1) & \text{if } f(\vec{c}_i(t_1)) < f(\vec{x}_i(t_1)) - \varepsilon \\ \vec{x}_i(t_1) = \vec{x}_i(t_1) & \text{if } f(\vec{c}_i(t_1)) > f(\vec{x}_i(t_1)) - \varepsilon \end{array} \qquad (5)$$

ε is a key parameter to control the insensitivity.

Step 5. Let $t_1 = t_1 + 1$, go to Step 2 and repeat until the maximum time limit T_G.

2.2 Local Subregions

The LSRs are constructed based on the set of updated particles

$$P = \{\vec{x}_1, \cdots, \vec{x}_N\},$$

Step 1. Let N_{max} denote the upper limit number of LSRs. Let i be the index of the LSR and let $i = 1$.

Step 2. The global best particle $\vec{x_g}$ is selected:

$$f(\vec{x}_g) \leq f(\vec{x}_i) \ \text{ for all } i$$

Based on the global best, the i-th LSR is constructed

$$\text{LSR}_i = \{\ \vec{x}\ |\ \ \|\ \vec{x} - \vec{x}_g\ \| < r\}$$

Fig. 1 illustrates the LSR construction. After the LSR_i is constructed, all the elements in LSR_i are removed from P.

Step 3. Let $i = i + 1$, go to step 2 and repeat until the upper limit N_{max}.

2.3 Local Search

In the local search, the IDE operates on all the LSRs in parallel. We define the local search for the LSR_i. Let t_2 denotes the iteration number. The particles in LSR_i are denoted by

$$X_i \equiv \{\vec{x}_1(t_2), \cdots, \vec{x}_{N_i}(t_2)\}$$

The algorithm is defined by the following 5 steps.

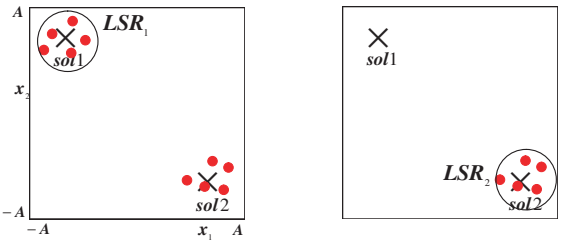

Figure 1: Construction of the LSRs.

Step 1 (Initialization). Let $t_2 = 0$.

Step 2. The mutation is defined by replacing t_1, X and N in Step 2 in 2.1 with t_2, X_i and N_i, respectively.

Step 3. The crossover is defined by replacing t_1, X and N in Step 3 in 2.1 with t_2, X_i and N_i, respectively.

Step 4 (Survival). The parent $\vec{x}_i(t_2)$ is compared with the offspring $\vec{c}_i(t_2)$ and is updated as the following:

$$\begin{array}{ll} \vec{x}_i(t_2) = \vec{c}_i(t_2) & \text{if } f(\vec{c}_i(t_2)) < f(\vec{x}_i(t_2)) - \varepsilon_2 \\ \vec{x}_i(t_2) = \vec{x}_i(t_2) & \text{if } f(\vec{c}_i(t_2)) > f(\vec{x}_i(t_2)) - \varepsilon_2 \end{array} \quad (6)$$

ε_2 is a key parameter to control the insensitivity in the local search. Let $f(\vec{x}_{Li})$ is the best in the LSR_i. The algorithm is terminated if

$$f(\vec{x}_{Li}) \leq C_1$$

where C_1 is an approximation criterion and $\vec{x}_{Li}$ is the approximate solution. Otherwise, go to Step 5.

Step 5: Let $t_2 = t_2 + 1$, go to Step 2 and repeat until the maximum time limit T_L.

3 NUMERICAL EXPERIMENTS

In order to confirm the algorithm efficiency, we have performed basic numerical experiments for a two dimensional function.

$$\begin{array}{c} f_1(\vec{x}) = -0.397 + (x_2 - \frac{5.1}{4\pi_1^2}x_1^2 + \frac{5}{\pi}x_1 - 6)^2 \\ +(1 - \frac{1}{8\pi})\cos(x_1) + 10 \\ S_0 = \{(x_1, x_2) \mid |x_1| \leq 15,\ |x_2| \leq 15\ \} \end{array} \quad (7)$$

f_1 has three solutions as illustrated in Fig. 2(a):

$$\begin{array}{l} \min(f_1(\vec{x})) = 0 \\ \vec{x}_s^1 \doteq (-\pi, 12.3),\ \vec{x}_s^2 \doteq (\pi, 2.28),\ \vec{x}_s^3 \doteq (9.42, 2.48) \end{array}$$

We have selected insensitive parameters (ε, ε_2) and the global search time limit T_G as control parameters. Other parameters are fixed after trial-and-errors: the number of particle N=30, the scaling parameter B=0.7, crossover probability P_c=0.9, approximation criterion $C_1 = 0.01$, the upper limit number of LSRs N_{max}=3, LSR radius r=1 and the maximum time step of the local search T_L=70.

Fig. 2 (b)-(d) show particle movement in global search for ε=3.0 and T_G=30. At $t = T_G = 30$, the particles are divided into three swarms and the global search is stopped. The algorithm constructs three LSRs as shown in Fig. 2 (e) where each LSR includes either solution. The algorithm is switched into

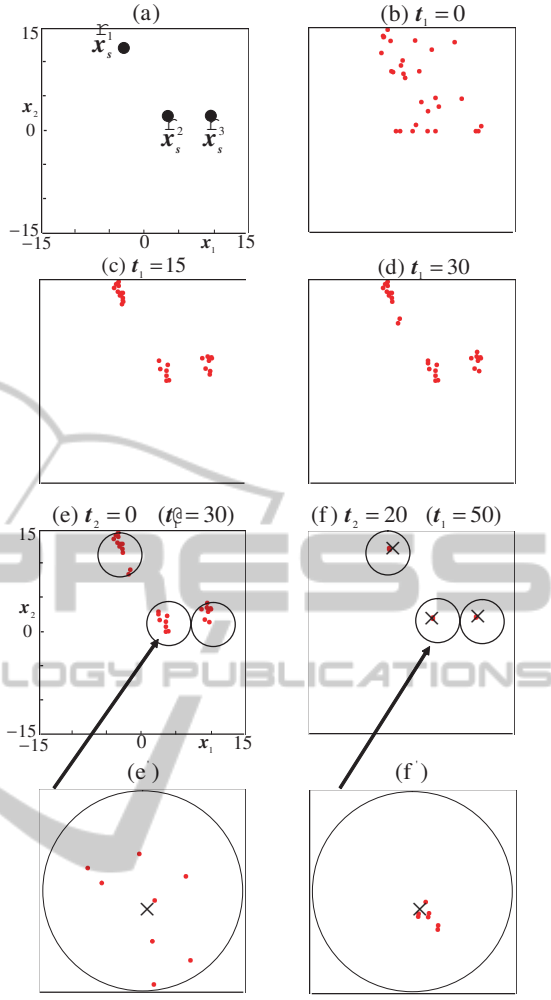

Figure 2: Particles movement for ε=3.0, T_G=30, ε_2=0.01 and T_L=70. (a) Solutions, (b) to (d) global search, (d) to (f') local search.

the local search with $\varepsilon_2 = 0.01$ and can find all the approximate solutions within the time limit $t_2 = T_L = 70$. Fig. 2 (f) shows that particles in each LSR converge to the each approximate solution. Figs. 2 (e') and (f') show enlargement of LSR_2 in Figs. 2 (e) and (f), respectively. Fig. 3 shows the local search process for ε_2=0.01 and T_L=70. The global search is said to be successful if the IDE can construct all the LSRs including either solution and local search is said to be successful if all the solutions are found.

Table 1 shows the success rate (SR) of global search for different initial particles positions, mutation and crossover in 50 trials. We can see that a good SR can be realized around $(\varepsilon, T_G) = (3, 30)$. The insensitive parameter ε plays an important role to construct the LSRs. If the parameter ε is too small then the IDE operation is the almost same as the standard DE and all the particles tend to be attracted to one so-

Table 1: Success rate (SR) of global search.

T_G \ ε	0	1	2	3	4	5
10	36	72	80	80	72	68
20	36	68	68	84	84	72
30	44	74	84	88	80	76
40	26	64	70	80	88	82
50	4	68	72	78	76	80

Table 2: SR of local search for successful global search (ε=3, $T_G = 30$).

ε_2	0.01	0.02	0.1	0.5	1.0	2.0	3.0
SR	100	100	60	10	1	0	0

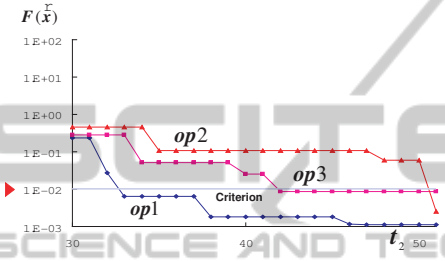

Figure 3: Local search process for ε_2=0.01 and T_L=70.

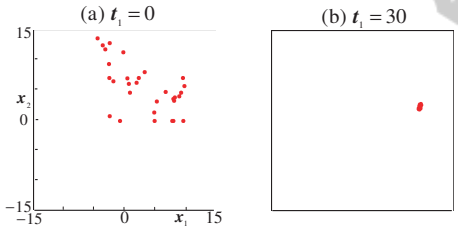

Figure 4: Particles movement of global search for ε=0.

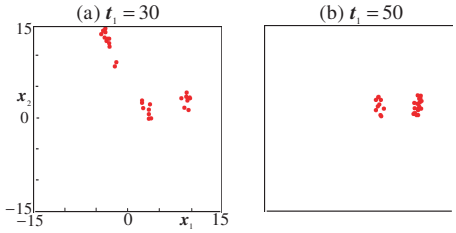

Figure 5: Particles movement ($t_1 \geq 30$) if the global search continues after Fig. 2 (d).

lution as suggested in Fig. 4 for $\varepsilon = 0$. In this case, the IDE can not find all the three solutions. If ε is too large then to IDE is too insensitive to find LSRs. The global search limit T_G is also important. If the global search continues after the time limit T_G then particles in either swarm tend to be attracted to other swarms as suggested in Fig. 5. If ε is too small or T_G is too large, the particles converge to one solution. Table 2 shows the SR of local search for 50 trials where ε=3, T_G=30 and the SR is calculated for successful global search. The SR=100 is achieved for small ε and the SR decreases as ε_2 increases.

4 CONCLUSIONS

A basic version of the IDE is presented and its performance is investigated in this paper. In the global search, the IDE can construct the LSRs successfully if the key parameters ε and T_G are selected suitably. In the local search, the IDE can find the desired approximate solution almost completely if ε_2 is selected suitably.

Future problems are many, including analysis of search process, analysis of insensitive parameters effects, automatic adjustment of key parameters, application to various functions, comparison with existing algorithms and application to engineering problems.

REFERENCES

Huang, H., Hu, S., and Czarowski, D. (2004). Harmonic elimination for constrained optimal pwm. In *Proc. Annual Conf. IEEE Ind. Electron. Soc.*, pages 2702–2705.

Lampinen, H. and Vainio, O. (2001). An optimization approach to designing otas for low-voltage sigma-delta modulators. In *Proc. of WCCI*, pages 1665–1671.

Li, C. and Zhao (2010). The hybrid differential evolution algorithm for optimal power flow based on simulated annealing and tabu search. In *Proc. of IEEE*, pages 1–7.

Luitel, B. and Venayagamoorthy, G. (2008). Differential evolution particle swarm optimization for digital filter design. In *Proc. of IEEE*, pages 3954–3961.

Storn, R. and Price, K. (1995). Differential evolution - a simple and efficient adaptive scheme for global optimization over continuous spaces. In *ICSI Technical Report, International Computer Science Institute.*

Storn, R. and Price, K. (1996). Minimizing the real functions of the icec'96 contest by differential evolution. In *Proc. of ICEC*, pages 842–844.

Storn, R. and Price, K. (1997). Differential evolution - a simple and efficient heuristic for global optimization over continuous spaces. In *Journal of Global Optimization*, pages 341–359.

Takahama, T. and Sakai, S. (2004). Constrained optimization by combining the α constrained method with particle swarm optimization. In *Proc. of Joint 2nd International Conference on Soft Computer and Intelligent Systems and 5th International Symposium on Advanced Intelligenct Systems*.

Takahama, T. and Sakai, S. (2006). Constrained optimization by the ε constrained differential evolution withgradient-based mutation and feasible elites. In *Proc. of WCCI*, pages 308–315.