NESTING DISCRETE PARTICLE SWARM OPTIMIZERS FOR MULTI-SOLUTION PROBLEMS

Masafumi Kubota and Toshimichi Saito
Faculty of Science and Engineering, Hosei University, 184-8584 Tokyo, Japan

1 INTRODUCTION

The particle swarm optimizer (PSO) is a simple population-based optimization algorithm and has been applied to various systems (Wachowiak et al., 2004) (Garro et al., 2009) (Valle et al., 2008): signal processors, artificial neural networks, power systems, etc. The PSO has been improved and evolved in order to expand the search function for various problems (Engelbrecht, 2005): multi-objective problems, multi-solution problems (MSP), discrete PSO, hardware implementation, etc. The MSP is inevitable in real systems and several methods have been studied (Parsopoulos and Vrahatis, 2004). The discrete PSO operates in discrete-valued search space (Engelbrecht, 2005), (Sevkli and Sevilgen, 2010) and has several advantages on reliable operation, reproducible results, robust hardware implementation, etc. However, there exist not many works of digital PSO for the MSP.

This paper presents the nesting discrete particle swarm optimizers (NDPSO) for the MSP. The NDPSO consists of two stages. The first stage is the global search. The whole search space is discretized into the local sub-regions (LSRs) that consist of subsets of lattice points in relatively rough resolution. Applying the particles with ring topology, the NDPSO tries to find the LSRs each of which includes an approximate solution. The second stage is the local search. Each subregion is re-discretized into finer lattice points and the algorithm operates in all the subregions in parallel to find all approximate solutions. Performing basic numerical experiment, the algorithm efficiency is investigated.

2 NESTING DISCRETE PSO

For simplicity, we consider the MSP in 2-dimensional objective functions $F(x) \geq 0$, $x \equiv (x_1, x_2) \in \mathbb{R}^2$ where the minimum (optimal) value is normalized as 0. $F$ has multiple solutions $x_i^s$, $i = 1 \sim M$: $F(x_i^s) = 0$, $x_i^s \equiv (x_{i1}^s, x_{i2}^s) \in S_0$. The search space is normalized as the center at the original with width $A$: $S_0 \equiv \{x\mid |x_1| \leq A, |x_2| \leq A\}$. As a preparation, we define several objects. The particle $x_i$ is described by its position $x_i$ and velocity $v_i$: $\alpha_i = (x_i, v_i)$, $x_i \equiv (x_{i1}, x_{i2})$, $v_i \equiv (v_{i1}, v_{i2})$ and $i = 1 \sim N$. The position $x_i$ is a potential solution. The personal best of the $i$-th particle, $pbest_i = F(x_{pbest_i})$, is the best of $F(x_i)$ in the past history. The local best, $lbest_i = F(x_{lbest_i})$, is the best of the personal best $pbest_i$ for the $i$-th particle and its neighbors. For example, the neighbor means the $i$-th and both sides particles in the ring topology. The global best $gbest$ is the best of the personal bests and is the solution of the present state.

Kubota M. and Saito T.
NESTING DISCRETE PARTICLE SWARM OPTIMIZERS FOR MULTI-SOLUTION PROBLEMS.
DOI: 10.5220/0003623102630266
Copyright © 2011 SCITEPRESS (Science and Technology Publications, Lda.)
The best AS1 is survived and the other subsets are removed. Although this paper uses square-shaped subsets, we construct subsets (LSR candidates) successively for all $i$ in order to include the target solution, we select top $\ell$ candidates to be included in the AS1 and $D_0$ is sampled on the discretized search space $D_0$. The target is the AS1 defined by

$$F(p) < C_1, \quad p \equiv (p_1, p_2) \in D_0$$

where $C_1$ is the criterion. The AS1 is used to make the LSRs each of which includes either solution. Let $t_1$ denote the discrete time in this stage and let $k_1$ denote the counter of AS1.

**Step 1.** Let $t_1 = 1$ and let $k_1 = 0$. Let $N_1$ particles form ring topology. The particle positions $x_i, i = 1 \sim N$, are assigned on $D_0$ randomly following uniform distribution on $D_0$ (Fig. 1 (a)). $p_i$, $pbest_i$ and $gbest_i$ are all initialized.

**Step 2.** The position and velocity are updated.

$$\dot{v}_i(t+1) = \omega \dot{v}_i(t) + r_1(x_{pbest_i} - x_i(t)) + r_2(x_{gbest} - x_i(t))$$

where $r_1$ and $r_2$ are random parameters in $[0, 1]$ and $[0, 2]$, respectively. If $x_i(t+1)$ exceeds $D_0$ then it is re-assigned into $D_0$. The parameters $\omega$, $r_1$, $r_2$, $c_1$ and $c_2$ are selected from lattice points to satisfy the condition $x_i(t) \in D_0$.

**Step 3.** The personal and local bests are updated:

$$x_{pbest_i} = x_i(t_1) \quad \text{if} \quad F_i(x_i(t_1)) > F_i(x_{pbest_i})$$

$$x_{lbest_i} = x_{pbest_i} \quad \text{if} \quad F_i(x_{pbest_i}) > F_i(x_{lbest_i})$$

**Step 4.** If $F(x_i) < C_1$ for some $i$ (Fig. 1 (b)) then $x_i$ is declared as the $k_1$-th AS1 and the counter number is increased: $p_i^{k_1} = x_i$ and $k_1 = k_1 + 1$. The position $x_i$ is declared as a tabu lattice point and is prohibited to revisit. $x_i$ is reset to a lattice point (Fig. 1(c)). $v_i$, $pbest_i$ and $lbest_i$ are all reset.

**Step 5.** Let $t_1 = t_1 + 1$, go to Step 2 and repeat until the maximum time step $t_{max}$.

In order to make the LSRs each of which is desired to include the target solution, we select top $K$ of the AS1s $p_1, \cdots, p^K$ such that $F(p_1) \leq \cdots \leq F(p^K) \leq C_1$. (If $k_1 < K$ then the top $k_1$ is used). Using the AS1s, we construct subsets (LSR candidates) successively for $i = 1 \sim K$: the $i$-th subset $S_i$ is centered at $p_1$ with area $2m_a d^2$: $S_i = \{x \mid |x_1 - p_1^1| < a_1, |x_2 - p_2^1| < a_1\}$, $a_1 = m_a d, i = 1 \sim K$ where $p_i = (p_i^1, p_i^2)$ is the $i$-th AS1 and $m_a$ is an integer smaller sufficiently than $m$. The $m_a$ determines area of $S_i$ in square shape. Although this paper uses square-shaped subspaces for simplicity, a variety of shapes should be tried depending on objective problems. If two or more subsets overlaps then the subset centered at the smallest AS1 is survived and the other subsets are removed.

If we obtain more than $M$ survived subsets, we select subsets centered at top $M$ of AS1s. We then obtain $M$ pieces of LSRs and reassign the notation $p_i \equiv (p_1^i, p_2^i)$ to the AS1 in the $i$-th LSR. If the LSRs include the target solutions $x_i^{*}, i = 1 \sim M$ then the global search is said to be successful.

Next, we discretize each LSR onto $m_2 \times m_2$ lattice points: $D_i = \{x \mid x_1 - p_1^i \in \mathbb{Z}_2, x_2 - p_2^i \in \mathbb{Z}_2\}$, $\mathbb{Z}_2 = \{l_{2,1}, \cdots, l_{m_2}\}$, $l_{2,k} = -(m_2 + \frac{1}{2})d + k d, d_1 \equiv 2m_2 d/m_2$. This $D_i$ is the $i$-th discretized LSR. $F(x)$ is sampled on $D_i$. The target is the AS2 defined by

$$F(q) < C_2, \quad q_i \equiv (q_1^i, q_2^i) \in D_i$$

where $C_2 < C_1$ is the criterion. The local search operates in parallel in $D_i$ (Fig. 1 (e)). $v_i$, $pbest_i$ and $gbest_i$ are all initialized.

**Step 2.** The position and velocity are updated.

$$\dot{v}_i(t_2 + 1) = \omega \dot{v}_i(t_2) + r_1(x_{pbest_i} - x_i(t_2)) + r_2(x_{gbest} - x_i(t_2))$$

where $r_1$ and $r_2$ are random parameter in $[0, c_1]$ and $[0, c_2]$, respectively. If $x_i(t_2 + 1)$ exceeds $D_i$ then it is reassigned into $D_i$. The parameters $\omega$, $c_1$ and $c_2$ are selected to satisfy $x_i(t) \in D_i$.

**Step 3.** The personal and global best are updated:

$$x_{pbest_i} = x_i(t_2) \quad \text{if} \quad F_i(x_i(t_2)) < F_i(x_{pbest_i})$$

$$x_{gbest} = x_{pbest_i} \quad \text{if} \quad F_i(x_{pbest_i}) < F_i(x_{gbest})$$

**Fig. 1:** Particles movement for two solutions. (a) Initialization, (b) the first AS1, (c) position reset, (d) two LSRs, (e) initialization for local search, (f) AS2.
Step 4. If \( F(x_{\text{glob}}) < C_2 \) for some \( i \) then we obtain one \( \text{AS2} \) and the algorithm is terminated.

Step 5. Let \( t_2 = t_2 + 1 \), go to Step 2 and repeat until the maximum time step \( t_{\text{max2}} \).

If the object is the \( \text{AS2} \) only, the particles are often trapped into either solution or local minima and are hard to solve the MSP. Our NDPSO tries to suppress the trapping by global search for the \( \text{AS1} \) with discretization (sampling) of the objective function.

### 3 NUMERICAL EXPERIMENTS

In order to investigate the algorithm capability, we have performed basic numerical experiments for the Himmelblau function with four solutions as illustrated in Fig. 2 (a):

\[
F_H(x) = (x_1^2 + x_2 - 11)^2 + (x_1 + x_2^2 - 7)^2
\]

\[
\text{min}(F_H(x_i^*)) = 0, \quad i = 1 \sim 4
\]

- \( x_1^1 = (-2.805118, 3.131312) \equiv \text{Sol1} \),
- \( x_1^2 = (3,2) \equiv \text{Sol2} \),
- \( x_1^3 = (-3.779310, -3.283185) \equiv \text{Sol3} \),
- \( x_1^4 = (3.584428, -1.848126) \equiv \text{Sol4} \)

\( S_0 = \{ x \mid x_1 \leq 6, \ x_2 \leq 6 \} \).

We have selected \( m_1, C_1 \) and \( t_{\text{max1}} \) as control parameters and other parameters are fixed after trial-and-error: \( N_1 = N_2 = 20, \omega = 0.7, \ c_1 = c_2 = 1.4, \ K = 30, \ m_1/(2m_a) = 8, \ m_2 = 32, \ t_{\text{max2}} = 50 \) and \( C_2 = 0.04 \). Fig. 2 (b) to (f) and Fig. 3 show a typical example of the global search process for \( m_1 = 64, C_1 = 5 \) and \( t_{\text{max1}} = 50 \). The NDPSO find the first \( \text{AS1} \) at \( t = 4 \) and find the other \( \text{AS1s} \) successively. At time limit \( t_{\text{max1}} \), the NDPSO can construct all the four LSRs successfully. Each LSR has \( 8^2 (8 = 2m_a = m_1/8) \) lattice points. Fig. 2 (g), (h) and Fig. 4 show the local search process where the NDPSO can find all the approximate solutions.

We evaluate the global search by success rate (SR) that means rate of finding all the LSRs in 100 trials for different initial states. Table 1 shows the SR of global search for \( m_1 \) and \( C_1 \). For \( m_1 = 32, 64 \) and 128, the LSR has \( 4^2, 8^2 \) and \( 16^2 \) lattice points, respectively in the global search. The LSR is divided into \( m_2 \times m_2 \) lattice points for the local search. We can see that \( C_1 \) is important for finding LSRs. As \( C_1 \) increases the SR increases and tends to saturate. For smaller \( C_1 \), the NDPSO operates like standard analog PSO in principle and tends to trap local solution/minimum. For larger \( C_1 \), the NDPSO has possibility to find a suitable \( \text{AS1} \) before the trapping. As \( m_1 \) increases, the resolution becomes higher and the SR tends to increase; however, the computation cost also increases of course: there exists a trade-off between the SR and computation cost.

We evaluate the local search by the SR and the average number of iteration (ITE) of finding all the
AS2s in 100 trials for different initial states. Table 2 shows the SR/#ITE of local search after the successful global search. We can see that the NDPSO can find all the AS2 speedily.

<table>
<thead>
<tr>
<th></th>
<th>C1 = 3</th>
<th>C1 = 5</th>
<th>C1 = 10</th>
<th>C1 = 20</th>
<th>C1 = 30</th>
</tr>
</thead>
<tbody>
<tr>
<td>m1 = 32</td>
<td>20</td>
<td>36</td>
<td>59</td>
<td>79</td>
<td>84</td>
</tr>
<tr>
<td>m1 = 64</td>
<td>32</td>
<td>53</td>
<td>74</td>
<td>92</td>
<td>94</td>
</tr>
<tr>
<td>m1 = 128</td>
<td>46</td>
<td>61</td>
<td>80</td>
<td>91</td>
<td>95</td>
</tr>
</tbody>
</table>

Table 2: SR/#ITE of local search of $F_H$ for $t_{max1} = 50$.

Table 3: SR of global search of $F_H$ for $t_{max1} = 50$.

<table>
<thead>
<tr>
<th></th>
<th>C1 = 3</th>
<th>C1 = 5</th>
<th>C1 = 10</th>
<th>C1 = 20</th>
<th>C1 = 30</th>
</tr>
</thead>
<tbody>
<tr>
<td>m1 = 32</td>
<td>100/5.30</td>
<td>100/5.60</td>
<td>100/5.46</td>
<td>100/5.55</td>
<td>100/5.57</td>
</tr>
<tr>
<td>m1 = 64</td>
<td>100/5.01</td>
<td>100/5.34</td>
<td>100/5.41</td>
<td>100/5.43</td>
<td>100/5.49</td>
</tr>
<tr>
<td>m1 = 128</td>
<td>100/5.08</td>
<td>100/5.20</td>
<td>100/5.36</td>
<td>100/5.42</td>
<td>100/5.55</td>
</tr>
</tbody>
</table>

Table 3 shows the SR in global search for $t_{max1}$ and $m_1$. The SR increases as $t_{max1}$ increases. For $m_1 = 64$ and 128, the SR saturates and $t_{max1} = 50$ (or 100) is sufficient for reasonable results. The parameter $t_{max1}$ can control the SR and computation costs.

4 CONCLUSIONS

The NDPSO is presented and its capability is investigated in this paper. Basic numerical experiments are performed and the results suggest the following.

1. The parameters $m_1$ and $C_1$ can control roughness in the global search that is important to find all the LSRs successfully. Higher resolution encourages trapping and suitable roughness seems to exist.

2. Parallel processing of the local search in LSRs is basic for efficient search. If LSRs can be constructed, the AS2s can be found speedily and steadily.

3. The discretization is basic to realize reliable and robust search in both software and hardware.

Future problems are many, including analysis of search process, analysis of role of parameters, comparison with various PSOs (Engelbrecht, 2005) (Miyagawa and Saito, 2009) and application to practical problems (Valle et al., 2008) (Kawamura and Saito, 2010).

REFERENCES


