A Crown Jewel Defense Strategy Based Particle Swarm Optimization

Ling Lin¹, Zhen Ji¹, Member, IEEE, Shan He², and Zexuan Zhu¹, Member, IEEE
¹Shenzhen City Key Laboratory of Embedded System Design
College of Computer Science and Software Engineering
Shenzhen University, Shenzhen, China 518060
²School of Computer Science
University of Birmingham, Edgbaston, Birmingham, B15 2TT, UK

Abstract—Particle swarm optimization (PSO) is a metaheuristic algorithm that is easy to implement and performs well on various optimization problems. However, PSO is sensitive to initialization due to its rapid convergence which leads to the lack of population diversity and premature convergence. To solve this problem, a jumping-out strategy named crown jewel defense (CJD) is introduced in this paper. CJD is used to relocate the global best position and reinitializes all particles’ personal best position when the swarm is trapped in local optima. Taking the advantage of CJD strategy, the swarm can jump out of the local optimal region without being dragged back and the performance of PSO becomes more robust to the initialization. Experimental results on benchmark functions show that the CJD-based PSO are comparable to or better than the other representative state-of-the-art PSO.

I. INTRODUCTION

Particle swarm optimization (PSO) was first introduced by Kennedy and Eberhart in 1995 [1] based on a social-psychological model of social influence and social learning. In PSO, a particle is defined to represent a potential solution of the optimization problem. A swarm of particles is initialized randomly and then each particle moves iteratively in the direction of its own personal best position and the global best position. Therefore, the particles discover optimal regions of the solution space through learning from the historical information of themselves and all the other particles. The moving direction, as known as velocity \(v^d\), and the position \(x^d\) of a particle are updated according to the formulas presented below:

\[
v^d = \omega \times v^d + c_1 \times r_1 \times (pbest^d - x^d) + c_2 \times r_2 \times (gbest^d - x^d)
\]

\[
x^d = x^d + v^d
\]

where \(w\) is the inertia weight introduced by Shi and Eberhart [2] to balance the global and local search, \(c_1\) and \(c_2\) are acceleration parameters, \(r_1\) and \(r_2\) are two random numbers in the range \([0, 1]\), \(pbest^d\) is the best position yielding the best fitness value in the historical search of the \(i\)th particle, and \(gbest^d\) is the global best position found by the whole swarm so far.

PSO is easy to implement and has been successfully applied in various optimization problems. However, it has a defect in solving complex multimodal problems for easily getting trapped in local optima. PSO converges quickly in that information transmits throughout the swarm rapidly, whereas the search scope is also shrinking rapidly, which leads to lack of population diversity and premature convergence. If the swarm is fortunately initialized in a good region, PSO will provide a preeminent solution, otherwise it will converge to an inferior solution and result in mediocre performance.

A lot of PSO variants have been proposed to solve the problem, whereas most of them are designed to change the way the particles fly [3]–[6]. In this paper, we propose a jumping-out strategy called crown jewel defense (CJD) to prevent the swarm from being trapped in local optima. Particularly, when the swarm stops improving for a certain period of time, CJD gives up the best positions found so far by the particles for the sake of higher possibility of acquiring a better solution and stabilizing the performance of the algorithm. CJD is applied to both global best and local best PSO. The global best version is based on PSO\(\omega\) [2] and the resultant algorithm is called PSOCJD. The local best version PSO is based on the von Neumann topology [8] and the resultant algorithm is named LPSOCJD. Experiments are performed on benchmark functions to validate the performance of the new algorithms.

The rest of this paper is structured as follows. In Section II, details of the CJD strategy are described. In Section III, the performance of the proposed algorithms on the test suites are demonstrated. Finally, the paper is ended with a short conclusion in Section IV.

II. METHODOLOGY

Before introducing the CJD strategy, we first define a success counter to detect whether the swarm is trapped in
A local optimum or not. If the swarm gets trapped, the CJD strategy will then be activated.

A. Success Counter

Success counter (SC) is an integer used to record the number of substantive improvements obtained by the swarm in an observation period (OP). An OP is a predefined number of generations. SC is updated only if the personal best of the particles are improved to a certain extent. The calculation of improvement ipm and SC are formulated as follows in a minimization problem:

\[
ipm = \frac{pbestV_{old} - pbestV_{new}}{pbestV_{old}} \quad (3)
\]

\[
SC = \begin{cases} 
SC + 1 & \text{if } ipm \geq \delta \\
SC & \text{if } ipm < \delta 
\end{cases} \quad (4)
\]

where \(pbestV_{old}\) and \(pbestV_{new}\) stand for the personal best fitness value before and after each update, respectively. If a particle finds a better personal best position, \(pbestV_{new}\) is then assigned with the personal best value. It is notable that \(SC\) is increased only if the improvement \(ipm\) is greater than a predefined threshold \(\delta\). When the swarm gets trapped, the particles oscillate around the local optimum and the improvements tend to be negligible. In this case, \(SC\) should not increase. Therefore, the threshold \(\delta\) is introduced to distinguish a significant improvement of a particle reaching a better solution and the oscillation of a particle around the local optimum. After each \(OP\), if \(SC\) is not increased, the swarm is considered to be trapped in a local optimum and the CJD strategy kicks in.

B. Crown Jewel Defense

In business, crown jewel defense (CJD) is a takeover defense strategy in which the target company sells off its most valuable assets to a third party to become a less attractive acquisition target. In another word, the most valuable things are discarded for greater interest. This strategy can be applied in the circumstance when the swarm of PSO gets trapped in local optima. As shown in Fig.1 (a), trapped in a local optimum, the swarm will present an iterative oscillation around the local optimum. The personal best positions of the particles now become burdens that confine the particles in the local optimal region. Therefore, although they are the most valuable assets the swarm has found so far, they should be discarded to make the local optimum less attractive to the particles in the hope of yielding a better solution somewhere else. The CJD strategy is implemented with two steps:

1) Relocate the \(gbest\):

The first step of CJD strategy is simply archive the current position of \(gbest\), then reinitialize the velocity of the global best particle and randomly relocate \(gbest\) to a new position.

2) Discard \(pbest\) of all particles:

In order to let the swarm follow the new \(gbest\) and get out of the local optimal region, the particles need to discard their \(pbests\). As defined in (1), a particle’s movement is guided by both \(gbest\) and \(pbest\). If only the \(gbest\) is relocated, a particle will still be dragged back to its old region in the effect of \(pbest\), which makes the particle escape the local optimal region very slowly or even worse cannot get out of the region, as shown in Fig.1 (b). Therefore, following the relocation of \(gbest\), the \(pbests\) of all particles should be reinitialized, as shown in Fig. 1 (c), so that the particles will move out of the local optimal region and head toward the new \(gbest\) rapidly. The new \(gbest\) could be much worst then what it used to be, but the particles will be in a more promising region and have more opportunities to reach a superior solution, as shown in Fig.1 (d).

Note that the CJD strategy is different from reinitialization of the PSO for it only reinitializes the \(pbest\) values but not the particle positions. With reinitialization, the algorithm is simply rerun from scratch, which does not mean to improve the performance and the swarm would probably get trapped in the same location. CJD strategy is designed to learn a lesson form what it has searched and escape from where it has fallen. The effect of the CJD strategy is also illustrated in Fig.2 using one of the jumping-out processes of PSOCJD on the basic benchmark function 14 from [13].

C. PSOCJD and LPSOCJD

The performance of CJD strategy is investigated using both global and local best PSO. Based on the size of the neighborhood of a particle, PSO algorithms can be divided into two kinds: the global best and local best PSO. The global best PSO or \(gbestPSO\) updates the velocity and position by learning information from all the particles in the swarm. The local best PSO or \(pbestPSO\) differs from the \(gbestPSO\) in the
learning objects. Instead of learning from the best position of the whole swarm, the pbestPSO learns from particles within a predefined neighborhood, namely neighborhood topology.

In this study, the global best PSO is based on the PSOW whose update rules have been given in (1) and (2). The corresponding resultant algorithm is called PSOCJD for short. In PSOCJD, the boundary constraints are handled using reflecting scheme [14]. For local best PSO, the Von Neumann neighborhood [8] is employed and the resultant algorithm is named LPSOCJD. In LPSOCJD, the particle velocity is updated according to the following equation:

$$v_{it}^k = \omega \times v_{it}^k + c_1 \times r_1^d \times (pbest_i^d - x_{it}^d) + c_2 \times r_2^d \times (lbest_i^d - x_{it}^d)$$  \hspace{1cm} (5)

where lbest is the particle with best personal best value in the Von Neumann neighborhood of the particle being updated. The pseudo code of PSOCJD/LPSOCJD is given in Algorithm 1.

III. EXPERIMENTS

A. Benchmark Functions

Ten Benchmark functions selected from [15] and [17] (as listed in Table I) are used in the experimental tests. These functions consist of multiple local optima and are suitable for validating the effect of CJD strategy.

<table>
<thead>
<tr>
<th>Function Name</th>
<th>Dimensionality</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>CF1</td>
<td>10</td>
<td>[15]</td>
</tr>
<tr>
<td>CF2</td>
<td>10</td>
<td>[15]</td>
</tr>
<tr>
<td>CF3</td>
<td>10</td>
<td>[15]</td>
</tr>
<tr>
<td>CF4</td>
<td>10</td>
<td>[15]</td>
</tr>
<tr>
<td>CF5</td>
<td>10</td>
<td>[15]</td>
</tr>
<tr>
<td>CF6</td>
<td>10</td>
<td>[15]</td>
</tr>
<tr>
<td>CF7</td>
<td>10</td>
<td>[17]</td>
</tr>
<tr>
<td>CF8</td>
<td>10</td>
<td>[17]</td>
</tr>
<tr>
<td>CF9</td>
<td>10</td>
<td>[17]</td>
</tr>
<tr>
<td>CF10</td>
<td>10</td>
<td>[17]</td>
</tr>
</tbody>
</table>

B. Parameter Settings for PSOCJD and LPSOCJD

To optimize the test functions, the population sizes of both PSOCJD and LPSOCJD are set to 30. Since CJD strategy has increased the diversity, PSO can take a small \( \omega \) to accelerate the convergence. In both PSOCJD and LPSOCJD, \( \omega \) is set to 0.4. The acceleration parameters are configured as \( c_1 = 2 \) and \( c_2 = 2 \) for PSOCJD. In LPSOCJD, \( c_1 \) is set to 1 and \( c_2 \) is set to 2, so that particles can fly toward their neighborhood best particle quickly and the convergence is accelerated. The observation period \( OP \) is set to 3 for PSOCJD but 5 for LPSOCJD as the local best PSO converges slower than the global version. The improvement threshold \( \delta \) is equal to \( 1e-6 \).

C. Comparison with Other PSO

Four state-of-the-art PSO variants are used in the contrast experiment. These algorithms and their parameter settings are summarized in Table II (PS stands for particle size).

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Parameters</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>PSOW</td>
<td>PS=20, ( \omega=0.4, c_1=c_2=2 )</td>
<td>[15]</td>
</tr>
<tr>
<td>CLPSO</td>
<td>PS=20, ( c=1.49445, \omega_0=0.9, \omega_1=0.7, m=8 )</td>
<td>[6]</td>
</tr>
<tr>
<td>OLPSOG</td>
<td>PS=40, ( c=2, \omega=0.9, G=5 )</td>
<td>[16]</td>
</tr>
<tr>
<td>OLPSOL</td>
<td>PS=40, ( c=2, \omega=0.9, G=5 )</td>
<td>[16]</td>
</tr>
</tbody>
</table>

For fair comparison, all algorithms are subject to the maximum fitness evaluations (FEs) of \( 5e + 04 \). Each algorithm is independently run 50 times and the average performance is reported. Statistical comparisons of the algorithms on all the functions are conducted using the Wilcoxon signed-ranks test.
The test results of all algorithms on the ten functions are shown in Table III. All algorithms are ranked on in terms of the mean optimized function value.

From Table III the following facts can be observed:

- Both PSOCJD and LPSOCJD significantly outperform the other PSO on most of the functions. LPSOCJD is competitive with PSOCJD, but LPSOCJD obtains slightly better mean function values.
- The improvements on $f_1$, $f_2$ and $f_5$ are more obvious than the other functions.
- The variances of PSOCLD and LPSOCLD are also smaller than the other algorithms. Especially, on $f_1$, $f_3$, $f_4$, $f_6$ and $f_{10}$, the variances and mean values of both PSOCJD and LPSOCJD are in the same or lower order.

The results show that both PSOCJD and LPSOCJD are robust to different initializations. Compared with PSOω, the performance of PSOCJD has been greatly enhanced by CJD. That means CJD indeed helps the swarm to escape from the local optimal region and find a better solution. Although PSOCJD and LPSOCJD obtain better function values than CLPSO and OLPSO, yet their optimal solutions are on the same scale and the main contribution of the CJD strategy lies in the improvement of stability.

### D. Convergence Trace

The average convergence traces of the comparing algorithms are plotted in Fig. 3. Here, only the trace of PSOCJD is plotted because the converge property of PSOCJD and LPSOCJD are alike.

It’s shown in Fig. 3 that on most of the functions, PSOCJD converges faster than the other algorithms. At the beginning, the curves of PSOω and PSOCJD are close, but after the CJD strategy kicks in and the two curves of PSOω and PSOCJD are detached. The results in Fig. 3 indicate that when optimizing complex composite functions:

- PSOω is easily trapped in local optima and suffers from premature convergence.

---

**TABLE III**

**Experimental Results of All Algorithms, FEs= 5.0e + 4**

<table>
<thead>
<tr>
<th></th>
<th>PSOω</th>
<th>CLPSO</th>
<th>OLPSO</th>
<th>OLPSOL</th>
<th>PSOCJD</th>
<th>LPSOCJD</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean</td>
<td>2.33e+02</td>
<td>5.37e+01</td>
<td>4.80e+01</td>
<td>3.94e+01</td>
<td>7.1e-30</td>
<td>0.00e+00</td>
</tr>
<tr>
<td>Var.</td>
<td>2.09e+04</td>
<td>3.65e+03</td>
<td>5.70e+03</td>
<td>1.65e+03</td>
<td>2.53e-57</td>
<td>0.00e+00</td>
</tr>
<tr>
<td>Rank</td>
<td>6</td>
<td>5</td>
<td>3</td>
<td>4</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>Mean</td>
<td>2.50e+02</td>
<td>1.05e+02</td>
<td>8.52e+01</td>
<td>8.34e+01</td>
<td>1.56e+01</td>
<td>2.55e+01</td>
</tr>
<tr>
<td>Var.</td>
<td>2.20e+04</td>
<td>6.22e+03</td>
<td>7.73e+03</td>
<td>1.77e+03</td>
<td>7.78e+02</td>
<td>3.60e+00</td>
</tr>
<tr>
<td>Rank</td>
<td>6</td>
<td>5</td>
<td>3</td>
<td>4</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>Mean</td>
<td>3.31e+02</td>
<td>2.25e+02</td>
<td>1.60e+02</td>
<td>2.50e+02</td>
<td>1.21e+02</td>
<td>1.21e+02</td>
</tr>
<tr>
<td>Var.</td>
<td>3.22e+04</td>
<td>1.13e+04</td>
<td>2.34e+03</td>
<td>4.25e+03</td>
<td>3.51e+02</td>
<td>4.15e+02</td>
</tr>
<tr>
<td>Rank</td>
<td>6</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>Mean</td>
<td>2.50e+02</td>
<td>1.05e+02</td>
<td>8.52e+01</td>
<td>8.34e+01</td>
<td>1.56e+01</td>
<td>2.55e+01</td>
</tr>
<tr>
<td>Var.</td>
<td>2.20e+04</td>
<td>6.22e+03</td>
<td>7.73e+03</td>
<td>1.77e+03</td>
<td>7.78e+02</td>
<td>3.60e+00</td>
</tr>
<tr>
<td>Rank</td>
<td>6</td>
<td>5</td>
<td>3</td>
<td>4</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>Mean</td>
<td>3.31e+02</td>
<td>2.25e+02</td>
<td>1.60e+02</td>
<td>2.50e+02</td>
<td>1.21e+02</td>
<td>1.21e+02</td>
</tr>
<tr>
<td>Var.</td>
<td>3.22e+04</td>
<td>1.13e+04</td>
<td>2.34e+03</td>
<td>4.25e+03</td>
<td>3.51e+02</td>
<td>4.15e+02</td>
</tr>
<tr>
<td>Rank</td>
<td>6</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

---

1. PSOCJD and LPSOCJD have the same mean value, therefore, the ranks is calculated as $(1 + 2)/2 = 1.5$ in the interest of fairness.

2. The differences of the mean values are statistical significant at level $\alpha = 0.05$ in Wilcoxon signed-ranks test between PSOCJD and the four contrast algorithm.

3. The differences of the mean values are statistical significant at level $\alpha = 0.05$ in Wilcoxon signed-ranks test between LPSOCJD and the four contrast algorithm.
Fig. 3. Average convergence curve of the algorithms over 50 runs on functions 1 to 10
Algorithm 1 PSOCJD/LPSOCJD

1: BEGIN
2: Initialize the particle swarm;
3: Update the best particle;
4: Initialize SC = 0;
5: Initialize GEN = 0;
6: while FEs < MaxFEs do
7: for Pid = 1 to SwarmSize do
8: Update particle velocity according to (1) for PSOCJD or (5) for LPSOCJD;
9: Update particle position according to (2);
10: Calculate particle fitness;
11: if PBestValue is improved then
12: Update pbest position and value;
13: Calculate ipm according to (3);
14: Update SC according to (4);
15: end if
16: if GBestValue is improved then
17: Update gbest particle id;
18: end if
19: end for
20: GEN = GEN + 1
21: if (GEN mod OP == 0) and (SC == 0) then
22: Reinitialize gbest particle velocity;
23: Update gbest particle position according to (2);
24: Update gbest particle’s pbest position and value;
25: for The rest of the swarm do
26: Reinitialize pbest position and value;
27: end for
28: end if
29: end while
30: END

- CLPSO and OLPSO increase the population diversity at the cost of convergence speed. Their curves present a slow but continued convergence trace.
- PSOCJD maintains quick convergence and meanwhile increases the population diversity. It attains satisfactory solution in a limited computational budget.

IV. CONCLUSION

In this paper, a CJD jumping-out strategy, which aims at increasing diversity and preventing PSO from converging to an inferior solution, was introduced to both global and local best PSO. The main idea of CJD is to discard the personal best value obtain when the swarm get trapped in a local optimal region and free the particles from the local optimum. The comparison studies were carried out on 10 composite benchmark functions between PSO with CJD and the other four state-of-the-art PSO variants. The test results show that CJD greatly enhances the performance of PSO. The convergence traces on the test functions demonstrated that PSO with CJD are able to increase the diversity while maintaining the quick convergence speed. Nevertheless, CJD still has a few problems to be addressed in the future work.

For example, the performance of PSO with CJD on higher dimensionality are yet to be explored, rules of thumb are needed to set the threshold δ and the observation period OP, and also more sophisticated heuristic is required for the relocation of particles.

REFERENCES