A Particle Swarm Optimization Based on Multi Objective Functions with Uniform Design

Adel H. Al-Mter*1, Songfeng Lu *2, Yahya E. A. Al-Salhi *3, Arkan A. G. Al-Hamodi #4

*Research scholar, School of computer science Huazhong University of Science and Technology
Wuhan 430074, PRC

#Associate professor, School of computer science Huazhong University of Science and Technology
Wuhan 430074, PRC

Abstract—A Multi-objective problems occurs wherever optimal solution necessary to be taken in the presence of tradeoffs between more than one conflicting objectives. Usually the population’s values of MOPSO algorithm are random which leads to random search quality. Particle Swarm Optimization Based on Multi Objective Functions with Uniform Design (MOPSO-UD), is proposed to enhance the accuracy of the particles convergence and keep the versatility of the Pareto optimal solutions and used the Uniform design to resolve the randomize search problem of the original MOPSO algorithm also the execution time of MOPSO-UD is faster compared with multi-objective particle swarm optimization algorithm (MOPSO).

Keywords—Particle swarm optimization algorithm, Multi-objective optimization, MOPSO algorithm, Uniform Design, MOPSO-UD.

I. INTRODUCTION

Over the last few years, new technologies provided a Swarm Intelligence (SI), which it is a set of stochastic methods based on the collective behaviour of self-organizing, distributed, flexible, autonomous and dynamic systems. These systems have a population of simple computational parameters that able to understand and modify their local environment. This capability makes possible communication among the parameters, which pick up the changes in the environment that generated by the behaviour of their neighbours. There is no centralized control structure to establish how the parameters should behave local interactions among the parameters generally result to give a global behaviour, also without an express model of the environment. Particle Swarm Optimization Algorithm is swarm intelligence optimization algorithm [1]; it is a random searching algorithm [2]. The main objective idea of PSO algorithm is originated from sharing and updating the information among the particles in the process of space searching. Each individual particle can benefit from discovery and flight experience of the others. In PSO algorithms, the particles swarm randomly initialized in the searching space and each particle has initial position and its own velocity. The path of particle is updated through particle best position and the path of swarm is updated via global best location of the particles swarm. This makes particles move to the optimal solution [1]. The parameter results identification influenced directly by the Initial population and the initial value of the algorithm parameters in PSO method. Generally, covered the searching space is uncertain. If the initial space population does not contain the information of global optimal solution, in the process of iteration, the searching space face difficult to extend to the region where the solution of global optimal in the finite times. The premature convergence problem may happen [3]. Uniform Design can be the solution of the randomness problem. Its aim to scatter design points uniformly on an experimental region [4]. UD get the most information via smallest number of tests, and reduce the direct influence of the initial population and the initial value of algorithm parameters on the results of parameter identification greatly. In the reference [2], PSO algorithm based on uniform design (UD-PSO) is proposed by used combining the UD and PSO algorithm. This algorithm thinks about the uniformity of the particle swarm distribution and the memory characteristic of particles. In the reference [1], is proposed a novel algorithm, parallel multi-swarm PSO based on k-medoids and uniform design. During the last decade a new technique has become a promising research area as many optimization problems of the real world fall into this category it is called multi-objective optimization [5]. A lot of practical optimization problems often include the simultaneous satisfaction of more than one objective function which are usually in opposes. The real difficulty very often resides with the multiple conflicting objectives simultaneous optimization. Which no one single solution can to be an optimal solution. That means they have two or more of trade-off solutions rather than a single one to satisfy all opposing objectives [6]. Inspired by various backgrounds, an increasing the multi-objective optimization algorithms number [7]; however, the optimal solutions can be a set of solutions with a trade-off of the objectives [8]. The optimal set of the trade-off solutions is known as the
Pareto-optimal set from which solution for an application can be picked using a decision maker. The MOO is proverbially based on three approaches, i.e., a priori, interactive and a posteriori [5]. A priori approach is a preference-based strategy, which comprising of comparative preferences of the different objectives is used. However, it is a tough task to devise like a preference-based strategy without any knowing of the likely trade-off solutions. A priori approach examples are: goal programming, weighting method, lexicographic ordering, etc. The DM in the interactive approach explains its preference during the optimization process and converts the search direction to the sit preferred area where it intends to have the solution. Without knowing all the Pareto-optimal solutions it is very difficult for the DM to prefer the solution. In the third approach (posteriori approach), the optimization designed to obtain all the Pareto-optimal solutions from that the DM picks one. Many MOO algorithms from this approach as there are no need to know any prior knowledge about the relative objectives.

There are two techniques used for solving the MOO problems, the first one is classical search algorithm and the second one is evolutionary algorithm. CS algorithm is a point-by-point search method where just one solution is evaluated and updated to get a better solution at every iteration. The outcome of these techniques is a single solution. In EA, the candidate solutions improvement in successive iterations is basically done by using point-to-point search working method utilizing mass experiences from a set or neighbourhood, since EA uses the potential candidate solutions. Thus, it produces in its final population multiple near-optimal solutions. The multi-point search ability to get multiple solutions in a simulation operation makes EAs unique to solve MOO problems. There are many types of EAs, like Genetic Algorithm (GA), Ant Colony Optimization (ACO), Artificial Immune System (AIS), Tabu Search (TS) and Particle Swarm Optimization (PSO). GA has mostly used for the MOO. The latest literature surveys refer the PSO to be a competitor of the GA [9-12]. PSO has many advantages, like effective memory use, easy implementation, and solutions diversity maintenance [11,12]. Multi objective PSO (MOPSO) is aims to reach a better convergence and better diversity among the solutions [13]. Thus, the most researches on MOPSO are aimed to attainment of these two goals. The first goal (better convergence) can be achieved by determine a suitable guide in the particle swarm for each particle. Unlike a very preliminary work [14], all other works [15-24], follow the selection mechanism of local/neighbourhood best guide. The various types of selection mechanisms of local guide are: Sigma method [17,18], dynamic neighbourhood method [15,25], random selection and roulette wheel selection [20,26], non-dominated sorting PSO [16], and sub-swarm based method [19,23,24]. The second goal of MOPSO (better diversity among the solutions) can be achieved by using additional operators of EA, i.e., mutation/ turbulence operator [17,23,26], or by proper archiving of the group of elite/non-dominated solutions in sequential iterations. There are different archiving techniques, like, niche count [16], crowding distance assignment [16], A-dominance [18], and clustering technique [19,24]. However, the use of special archive techniques or additional EA operators poses additional computational strain.

The simultaneous optimization of the MOO can be done in many ways, like, lexicographic ordering [13], Pareto-dominance based approach [27], aggregating approach [28], and non-Pareto based approach [29]. Most of the MOO algorithms are based on the Pareto-dominance principle [13,29].

In this paper, a Particle Swarm Optimization Based on Multi Objective Functions with Uniform Design, the Pareto-based approach is used and we proposed a new modified on PSO algorithm where used Multi Objective Functions to decrease the execution time to get the optimal solution from more than one solution and used the Uniform Design to solve the randomize search problem. This paper used three techniques (PSO, MOO and UD) to get a new optimization algorithm which called (MOPSO-UD).

The organization of this paper is as follows; Section 2 is related works. Preliminaries are described in Section 3. In Section 4, we present the proposed method. Experimental results are given in Section 5. Section 6 is the conclusion. The appendix consists of the list of abbreviations and principal symbols used in this paper.

II. RELATED WORKS

This section gives some of related literatures for this paper. Literature [30] was the first effort to extend PSO algorithm to solve MOO problem. In their algorithm used Pareto dominance to generate the leaders list that guides the search. Literature [31] proposed multiple objective particle swarm optimization proposal which uses a geographically based approach and an external memory to maintain diversity. They were made the first comparisons between MOPSO with multi-objective evolutionary algorithms. After these works, a many PSO algorithms types to handle MOO problems have been developed [32]. Literature [33] used roulette wheel selection to select the global best particle from the trade-off solutions. Literature [34] select the global best particle by utilized a tournament niche method, and used Pareto dominance to update the local best particle. Literature [35] was rank all the particles by developed a preference order and thus to identified the global best particle. Literature [36] combined PSO with EA algorithm and used a tournament selection to select the global best particle from the external archive and select the individual best particle as the one with minimum strength Pareto fitness. Literature [37] introduced a multi-objective PSO algorithm enhanced
with a fuzzy logic-based controller. The authors evaluate search spaces by used a number of fuzzy rules further dynamic membership functions.

To prevent PSO algorithms from getting caught into local optimal, usually employed the diversity preserving techniques. A considerable technique being conducted to maintain the swarm variety in PSO, multi-swarm technique has attracted more attention. In latest years, many multiple swarms PSO algorithms have been developed for MOO. Literature [26] adopted different clustering techniques to divide the particles population into several swarms, aimed to maintain a better diversity. Literature [38] used k-means approach for clustering the particles into sets and developed a multi-swarm PSO for molecular docking problem. Literature [39] developed a strategy of dynamic population and integrated it with the multi swarm PSO to form an efficient algorithm for MOO. Many scams, like estimation of cell-based rank density, population growing and declining, also adjustment the local search, are aimed to improve the algorithmic performance. Literature [40] developed a cooperative multi-objective PSO algorithm. The algorithm composed of multiple sub-swarms which associate of each other with the ring topology. Literature [41] proposed a dynamic multi-swarm PSO for MOO. This divided the population dynamically into several small sized swarms and reassembly randomly every other a few generations. Literature [42] also applied dynamic multi-swarm PSO algorithm to large-scale portfolio optimization problem and obtained competitive performances. Literature [43] proposed a cooperative PSO algorithm which employs multi sub-swarms to solve the routing recovery problem. Literature [44] developed an archive based multi-swarm algorithm for many-objective problems. In this algorithm, different swarms have different achieving methods and contact with each other using the ring topology. Literature [45] proposed a hybrid competent multi-swarm approach for MOO. In their algorithm, the distribution algorithm estimation is combined with multi-swarm PSO.

III. PRELIMINARIES

A. PSO algorithm

In 1995 Dr. Eberhart and Dr. Kennedy developed an evolutionary computation technique that inspired from the social behaviour of birds [9]. PSO is initialized with a population of random solutions, called particles [46]. These particles are flying around the searching space with a velocity that is dynamically adjusted. These dynamical adjustments are dependent on the historical behaviours of itself and other particles in the population. \( x = (x_1, x_2, \ldots, x_d) \), \( x \) represents the \( i^{th} \) particle, the best individual solution of is \( p_i = (p_{i1}, p_{i2}, \ldots, p_{id}) \). The best solution of all particles is \( p_g = (p_{g1}, p_{g2}, \ldots, p_{gd}) \). \( V_i = (v_{i1}, v_{i2}, \ldots, v_{id}) \) is the velocity of particle \( i \) [47], the velocity changes are calculated as Eq.(1), for \( i^{th} \) particle \( j^{th} \) dimension:

\[
v_{ij}^{t+1} = v_{ij}^t + c_1 r_1 [p_{ij} - x_{ij}^t] + c_2 r_2 [p_g - x_{ij}^t]. \tag{1}
\]

The position of a particle \( (j) \) is calculated as following equation:

\[
x_{ij}^{t+1} = x_{ij}^t + v_{ij}^{t+1}, \quad j = 1, 2, \ldots, d. \tag{2}
\]

The first term among the three component terms in Eq.(1), is referred to habit or momentum, i.e., the propensity of a particle to keep the same direction which it has been traveling. The second term, which is scaled by the learning constant \( (c_1) \) and the random number \( (r_1) \), is referred to as guidance by self-knowledge. The third term, which is scaled by the learning constants \( (c_2) \) and the random number \( (r_2) \), is referred to as guidance by group knowledge [12]. The fitness of a particle is identifying by a predefined objective function. A particle’s \( p_g \) serves as a directory and the best neighbour is sentenced by the fitness evaluation. PSO is executed iteratively till attained the specified termination criterion, i.e., certain desired particle fitness or a specified maximum number of iterations. Several basic PSO modifications have been reported [10, 11, 12]. Between them, an efficient modification is that of a linearly decreasing inertia weight \( (\omega) \) to balance global and local search [10]. Lower inertia facilitates local search and a higher inertia represents a global search by increase the weights on the previous experience. In this inertia model, the velocity of the particle is updated by using Eq.(3). The inertia weight is multiplied with the particle velocity of the previous iteration.

\[
v_{ij}^{t+1} = w \cdot v_{ij}^t + c_1 r_1 [p_{ij} - x_{ij}^t] + c_2 r_2 [g_{ij} - x_{ij}^t]. \tag{3}
\]

The PSO performance depends on the sharing the information among the particles, using various neighbourhood topologies. The common types of neighbourhood are the local best and global best topologies [11]. In the local topology, a particle shares its experience just with its immediate neighbours. In the global topology, the particles shares information immediately results in initial faster convergence, with a great being trapped possibility in local minima.

The pseudo code of PSO algorithm is given below.

<table>
<thead>
<tr>
<th>Particle Swarm Optimization Algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. // initialize all particles.</td>
</tr>
<tr>
<td>2. for each particle i do</td>
</tr>
<tr>
<td>3. for each dimension d in D do</td>
</tr>
<tr>
<td>4. //initialize the position and velocity of all particle’s</td>
</tr>
<tr>
<td>5. ( v_{ij} = r[0,1] )</td>
</tr>
<tr>
<td>6. end for</td>
</tr>
<tr>
<td>7. //initialize Particle best position</td>
</tr>
<tr>
<td>8. ( p_i = x_i )</td>
</tr>
<tr>
<td>9. //update the global best position</td>
</tr>
</tbody>
</table>

---

**SSRG International Journal of Computer Science and Engineering (SSRG-IJCSE) – volume 3 Issue10 – October 2016**

ISSN: 2348 – 8387  www.internationaljournalssrg.org  Page 22
10. If \( f(p_i) < f(p_g) \) then  
11. \( p_g = p_i \)  
12. end if  
13. end for  
14. repeat  
15. for each particle \( i \) do  
16. // update the particle’s best position  
17. If \( f(X_i) < f(p_i) \) then  
18. \( p_i = X_i \)  
19. end if  
20. // update the global best position  
21. If \( f(p_i) < f(p_g) \) then  
22. \( p_g = p_i \)  
23. end if  
24. end for  
25. // update particles velocity and position  
26. for each particle \( i \) do  
27. for each dimension \( d \) do  
28. calculate the velocity and position  
29. end for  
30. end for  
31. // advance iteration  
32. \( t = t + 1 \)  
33. until \( t > \text{max iterations number} \)

### C. Uniform Design

Briefly in this section, we describe one of the important experimental design methods called Uniform Design. It was proposed in 1981 by Professor Kaitai Fang and Yuan Wang [50,51]. The Uniform design has been successfully used in many fields, such as computer sciences, natural sciences, system engineering and survey design pharmaceutics etc. The main objective idea of UD is to sample a small group of points from a given group of points, such that sampled points are uniformly distributed. Suppose \( n \) factors and \( q \) levels per factor. The UD selects \( q \) combinations out of \( q^n \) possible combinations after \( n \) and \( q \) are given. Selected \( q \) combinations are given by a uniform array \( U_{ij} = [U_{ij}]_{i \times n} \), where \( U_{ij} \) level of the \( j \)th factor in the \( i \)th combination, which can be calculated by the following formula:

\[
U_{ij} = (i\sigma^{i-1} \mod q) + 1,
\]

where, the parameter \( \sigma \) should be chosen by the user [50,51].

### IV. Proposed Method

In this paper, A Particle Swarm Optimization Based on Multi Objective Functions with Uniform Design, we proposed MOPSO-UD algorithm to decrease the running time of original MOPSO algorithm and employee the uniform design in multi objective swarm optimization to develop the work of multi objective PSO algorithm and resolve the randomize searching problem. The MOPSO-UD algorithm steps are the following:

**Step1:** Initialize swarm, velocities and best positions of each particle in the swarm and empty external archive (\( A \)).

**Step2:** For each particle

**Step2.1:** Select archive (\( A \)) members.

**Step2.2:** Update velocity and position.

**Step2.3:** Evaluate new position.

**Step2.4:** Update best position and archive (\( A \)).

**Step3:** If stopping conditions not be satisfied repeat step (2).

The first step is to initialize the population of the swarm which the first values of the particles velocities and positions are random and the external archive set (\( A \)) is empty. Next step is selecting the archive members; then update the velocities according to Eq. (7).
where, the new velocity is \( v_{1j}^{t+1} \) and the old velocity is \( v_{1j}^{t} \), where the parameter of inertia weight coefficient is \( w \), which points out the particle's ability to maintain the previous speed. \((c_2)\) and \((c_2)\) are acceleration coefficients used to represent the degree of tracking solitary optimal and over all optimal respectively. \((x_{1j}^{t})\) is the current position of the particle \((j)\), the parameters \((p_{1j})\) and \((g_{1j})\) are the best particle \((j)\) position and the leader position of each particle \((j)\) respectively. The velocity update equation consists of three components, including components of the previous velocity, a cognitive and a social. They are mainly controlled by the inertia weight parameter and two acceleration coefficients. Uniform parameters are given in Eqs. (8) and (9)

\[
U_1 = (i \ast \sigma^{i-1} \mod p_{1j}) + 1, \\
U_2 = (i \ast \sigma^{i-1} \mod g_{1j}) + 1. \tag{9}
\]

where, \((i)\) is the particle and the parameter \((j)\) is maximum iterations , \((p_{1j} \cdot g_{1j})\) are same as the parameters of Eq. (7) and \((\sigma)\) is detected by the user. To update the positions of each particles follow Eq. (2), then evaluate the new positions of the particles, after that update both of best positions and the archive, updating operation of the best position of the individual particle is by compare the current best position with the new position if the new position is better then will change current position by new position otherwise the particle keep the current position and updating the archive using non-dominate solutions that explained in definition 1.1 to update the members. Finally, terminate the search process when the stopping conditions are satisfaction.

V. RESULTS

The performance of MOPSO-UD algorithm has been compared with the original version of MOPSO algorithm. The experiments are given in this section.

A. Experimental setting

In this section, the performance of MOPSO-UD algorithm is examined and implemented in Matlab [R2013b]. During the numerical experiments, both the Particle Swarm Optimization Based on Multi Objective Functions with Uniform Design and the original version of MOPSO algorithm were run with random initial population values and repeated for ten times and the running trials implemented with two hundred particles. The inertia weight \((w)\) is 0.7 and the inertia weight damping rate equal to one. The acceleration coefficients \((c_1)\) and \((c_2)\) set to two. They are the parameters of cognition and social model of MOPSO-UD and MOPSO respectively. Deap-benchmarks test multi objective functions used to test both of the algorithms. The multi objective test functions that used are ZDT1, ZDT2, ZDT3, ZDT4 and ZDT6. The formulates are shown in Table 1.

<table>
<thead>
<tr>
<th>Test functions name</th>
<th>Formula</th>
</tr>
</thead>
<tbody>
<tr>
<td>ZDT1 ( g(x) = 1 + \frac{1}{n-1} \sum_{i=2}^{n} x_i )</td>
<td>( f_1(x) = x_1 )</td>
</tr>
<tr>
<td>( f_2(x) = g(x) \left( 1 - \frac{x_1}{g(x)} \right) )</td>
<td>( f_2(x) = g(x) \left( 1 - \frac{x_1}{g(x)} \right)^2 )</td>
</tr>
<tr>
<td>ZDT2 ( g(x) = 1 + \frac{1}{n-1} \sum_{i=2}^{n} x_i )</td>
<td>( f_1(x) = x_1 )</td>
</tr>
<tr>
<td>( f_2(x) = g(x) \left( 1 - \frac{x_1}{g(x)} \right)^2 )</td>
<td>( f_2(x) = g(x) \left( 1 - \frac{x_1}{g(x)} \right)^2 )</td>
</tr>
<tr>
<td>ZDT3 ( g(x) = 1 + 10(n - 1) + \sum_{i=2}^{n} \left( x_i^2 - 10 \cos(4 \pi x_i) \right) )</td>
<td>( f_1(x) = x_1 )</td>
</tr>
<tr>
<td>( f_2(x) = g(x) \left( 1 - \frac{x_1}{g(x)} \right)^2 )</td>
<td>( f_2(x) = g(x) \left( 1 - \frac{x_1}{g(x)} \right)^2 )</td>
</tr>
<tr>
<td>ZDT4 ( g(x) = 1 + 9 \left( \frac{\sum_{i=2}^{n} x_i}{n-1} \right)^{0.25} )</td>
<td>( f_1(x) = 1 - \exp(-4x_1) \sin^6(6x_1) )</td>
</tr>
<tr>
<td>( f_2(x) = g(x) \left( 1 - f_1(x)/g(x) \right)^2 )</td>
<td></td>
</tr>
</tbody>
</table>
According to the proposed algorithm, and as mentioned in Eqs. (7), (8) and (9) the execution time has been decreased compare with previous method. From Table 2, the obtained results by applying “particle swarm optimization based on multi objective functions with uniform design” for all Deap-benchmarks test multi objective functions that used in this paper considerably shows the speed is faster than the original version of MOPSO algorithm, which the executions time is less than original version of MOPSO algorithm.

VI. CONCLUSIONS

Multi-objective function is more convenient to deal with the real problems of the life because most of these problems involve at least two objectives. In this paper, A Particle Swarm Optimization Based on Multi Objective Functions with Uniform Design, we proposed MOPSO-UD algorithm to employ the uniform design in multi objective swarm optimization to develop the work of multi objective PSO algorithm for resolving the randomize searching problem and this decreased the run-time of the algorithm as shown in Table 2,

UD distributes the experimental points uniformly in the whole test domain. Simulation results show MOPSO-UD has a better stability and global convergence; which it is the most important conclusion.

Appendix

This appendix consists of the list of abbreviations used in this paper.

List of abbreviations:
PSO: particle swarm optimization.
PSO-MOU: particle swarm optimization based on multi objective functions with uniform design.
MOPSO: multi objective particle swarm optimization.
SI: swarm intelligence.
UD: uniform design.
UD-PSO: particle swarm optimization algorithm based on uniform design.
MOO: multi objective optimization.
DM: decision maker.
EA: evolutionary algorithm.
CS: classical search.

ACKNOWLEDGMENT

The authors gratefully acknowledge the support from the Natural National Science Foundation of China under (Grant No. 61173050).

REFERENCES


