Improved Efficiency of MOPSO with Adaptive Inertia Weight and Dynamic Search Space

Lee-Ping Pang, Sin-Chun Ng
The Open University of Hong Kong
Hong Kong
1022ping@gmail.com, scng@ouhk.edu.hk

ABSTRACT

In this paper, a new Multi-Objective Particle Swarm optimization algorithm (MOPSO) with adaptive inertia weight and dynamic search space is introduced for multi-objective optimization. The objective of the study is to investigate an efficient MOPSO to deal with large-scale optimization and multi-modal problems. The new adaptive inertia weight strategy allows the inertia weight to keep varying throughout the algorithm process, which helps the algorithm to escape from local optima. The dynamic search space design can avoid decision variables from continuously taking their extreme values, and therefore enhances the searching efficiency. The performance of the proposed algorithm was compared with three popular multi-objective algorithms in solving seven benchmark test functions. Results show that the new algorithm can produce reasonably good approximations of the Pareto front, while performing with a budget of 10,000 fitness function evaluations.

CCS CONCEPTS

• **Computing methodologies → Search methodologies;**

KEYWORDS

Multi-objective particle swarm optimization; inertia weight; exploration and exploitation; bound handing, searching efficiency

1 INTRODUCTION

Multi-Objective optimization problems (MOP) have more than one objective function to be optimized simultaneously [1]. A general MOP global minimum problem with n objectives is defined as [2]:

$$\textit{Minimize } f(x) = [f_1(x), f_2(x) \ldots, f_n(x)]$$
$$\text{subject to } g(x) \le 0,\ h(x) = 0$$

where $g(x) \le 0$ and $h(x) = 0$ represent the constraints.

GECCO '18 Companion, July 15–19, 2018, Kyoto, Japan

ACM ISBN 978-1-4503-5764-7/18/07…$15.00
https://doi.org/10.1145/3205651.3208229

Definition 1. (Pareto dominance) [3]:
x is said to dominate y
if $\forall i,\ f_i(x) \le f_i(y), \exists i,\ f_i(x) < f_i(y)$.

Definition 2. (Pareto optimal set) [3]:
$P^* = \{x \mid \neg\exists y \in \Omega, f(y) \prec f(x)\}$.

Definition 3. (Pareto front) [3]:
$PF^* = \{f(x) = [f_1(x), f_2(x) \ldots, f_n(x)] \mid x \in P^*\}$.

The concept of domination is defined as A dominates B if it is better in at least one objective and not worse in all other objectives. We aim to find a set of non-dominated "trade-off" solutions that represent the best possible compromises among the objectives instead of a single solution.

Particle Swarm Optimization (PSO) [3] is a population-based heuristic search technique that simulates the movements of a flock of birds which aim to find food. In a PSO algorithm, each particle represents a potential solution in objective space, and the population of particles is called a swarm. At each iteration, particles update their position according to its own experience (personal best, *pbest*) and its neighbors (global best, *gbest*). The non-dominated global best solutions are called "leaders", which is stored in an external archive. Particle movement is computed for the (t+1)th iteration as follows [3]:

$$x_i(t+1) = x_i(t) + v_i(t+1) \quad (1)$$

$$v_i(t+1) = W \times v_i(t) + C1r2 \times (pbest_i(t) - x_i(t)) + C2r2 \times (gbest(t) - x_i(t)) \quad (2)$$

where i = 1,2,3 ... , n is the index of particle, the position and velocity of the i-th particle at the t-th iteration is denoted as $x_i(t)$ and $x_i(t)$, respectively. At the t-th iteration, $gbest(t)$ is the global best position founded by the entire swarm while $pbest_i(t)$ is the personal best position found so far by the i-th particle. W is the inertia weight of the current particle, it controls the trade-off between global and local experience. C1, C2 are the acceleration coefficients, and r1 and r2 are random values uniformly distributing in the interval [0, 1].

In PSO, inertia weight (W) [4] plays a key role to balance between exploration and exploitation process in PSO. It determines the contribution rate of a particle's previous velocity to its velocity at the current time step. When W is small, the PSO is more like a local search algorithm. When W is large, the PSO is more like a global search method and it always tries to explore the new areas.

Multi-Objective Particle Swarm optimization (MOPSO) is a very competitive swarm intelligence algorithm for solving multi-objective optimization problems. However, there are two main issues affecting the efficiency of MOPSO. Firstly, when using MOPSO to solve multi-modal functions, it gets into local optima easily, and suffers from premature convergence. Secondly, in some cases, decision variables would continuously take the values of their extreme values, which does not contribute to the search [5]. This is because when a particle moves outside the bound, the decision variable takes the value of its corresponding boundary. To overcome the first issue, this paper proposes a new adaptive inertia weight strategy to adjust the inertia weight using the potential of algorithm being trapped into a local optima at the current iteration. To overcome the second issue, dynamic search space is introduced to avoid particles from going to the search space boundary.

2 PROPOSED ALGORITHMS

2.1 Adaptive inertia weight strategy

Inertia weight can balance the exploration and exploitation ability of PSO. When it is large, the algorithm has a higher chance to jump out of the local optima. When it is small, the algorithm enables more exploration. To enhance the capability of MOPSO to solve multi-modal problems, a new adaptive inertia weight strategy is introduced in this paper. The new strategy consider the search progress of the algorithm, using this information to dynamically adjust the inertia weight, so that the algorithm can escape from local optima.

To get an indicator of the search progress, the percentage of particles whose personal best (*pbest*) is dominated by any external archive members is calculated, which is called p(t). The higher p(t) indicates the current iteration has a higher potential that it is trapped into a local optimum. p(t) is calculated at the end of each iteration.

$$p(t) = percentage\ of\ particles\ whose\ pbest\ is\ dominated\ by\ any\ external\ archive\ member \quad (3)$$

When p(t) is high, it indicates that a dominance improvement is made at the previous iteration (a solution that dominates all previous solutions is found). At this iteration, inertia weight decreases to encourage more exploration of the new good leaders. When p(t) is medium, particles share information and converge to the global best, but the swarm is not trapped in local optima. When p(t) is low, the algorithm is potentially trapped in local optima, inertia weight will increase to encourage more global search, and escape from the local optima.

To balance local search and global search in MOPSO, we let W varies in [0.5, 0.9]:

$$W(t+1) = 0.9 - p(t) * 0.4 \quad (4)$$

where p(t) is the indicator value at iteration t, and W(t+1) is the inertia weight at iteration t+1.

Fig.1 and Fig.2 plots the Hypervolume and the corresponding inertia weight of a MOPSO with adaptive inertia weight using the test function ZDT4. When the Hypervolume improvement is small, it means that MOPSO is potentially trapped in local optima. Thus W increases to enable more exploitation.

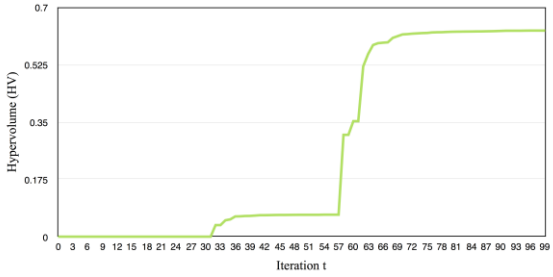

Figure 1: Hypervolume indicator at iteration t.

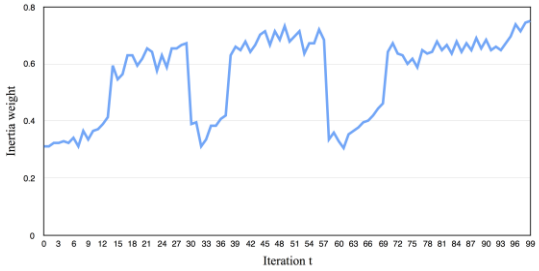

Figure 2: Variations of the inertia weight at iteration t.

2.2 Dynamic search space

In MOPSO, particles move in the search space guided by their own experiences and the acquired knowledge during the optimization process. One of the problems faced with PSO is that of maintaining the swarm within the feasible region. Researchers have proposed various bound handling methods, according to experimental studies, the nearest method is the most reliable methods for bound handling, which is when a particle moves outside the bound, the decision variable takes the value of its corresponding boundary [6]. However, the disadvantage of this method is that it is unable to solve some of the MOPs (e.g. DTLZ1, and DTLZ3 problems) when particles moves to its extreme values continuously [5].

A dynamic search space which the upper and lower bound of decision variables are changed with the number of iterations is introduced in the proposed algorithm. It can avoid particles from continuously going to the search space boundary, and therefore enhances the search efficiency.

We let the first half swarm moves within a dynamic search space boundary, and another half moves within the normal search space boundary. When a particle moves outside the bound, the decision variable takes the value of its corresponding boundary. For the first half swarm at the first iteration, their position are set to the search space center, and the upper and lower bound of decision variable becomes:

$$upperbound = \frac{range}{2} + range * 0.1 \quad (5)$$

$$lowerbound = \frac{range}{2} - range * 0.1 \quad (6)$$

For every specified number of iterations, the upper and lower bound of decision variables increase or decrease by ($range * 0.1$). When half of the total iterations have completed, the upper

and lower bound of decision variables becomes their normal value.

The idea of dynamic search space is to first gather the particles at the search space center, and then expand the search area gradually. This can prevent the decision variables keep taking their extreme values, and therefore improve the efficiency of search.

2.3 The improved MOPSO Algorithm

The new algorithm is based on an improved MOPSO algorithm called OMOPSO [7], which adopts the concept of ε-dominance and crowding-distance information to identify the leader set. To improve the efficiency of the algorithm, a new adaptive inertia weight strategy and dynamic search space is applied.

Fig.3 shows the pseudo code of the proposed algorithm. First, the swarm is initialized. The non-dominated particles found in the swarm will be introduced into the leader set. The crowding factor of each leader is calculated. Later on, we gather the first half swarm to the search space center, and set the corresponding feasible search space. At each iteration, we calculate the indicator p(*t*), using this information to adjust the inertia weight. When the current iteration equals to ⅛, ¼, ⅜ or ½ of the total iterations, the feasible search space for the first half swarm will expand its size. For each particle, we perform the flight and apply the corresponding mutation operator. The resulting particles are evaluated and both the particles' memory and the archive are updated. The algorithm returns the archive as the approximation set found.

```
Begin
    Initialize swarm. Initialize leaders. Send leaders to ε-archive
    crowding(leaders), generation=0
    if (indexOfParticle < totalParticles*50%)
        set its position to the search space center
        set feasible search space
    end if
    while generation < maxGeneration do
        computeInertiaWeight()
        if (generation equals to specified number of generation and
            indexOfParticle < totalParticles*50%)
            expand the search space
        end if
        updatePosition()
        mutation()
        evaluation()
        updateLeaders()
        updateε-archive()
        generation++
    end while
    return results in ε-archive
End
```

Figure 3: Pseudo code of the complete algorithm.

3 PEFORMANCE COMPARSIONS

To validate our proposed algorithm, we performed comparison with respect to three state-of-the-art multi-objective algorithms: OMOPSO [7], SMPSO [8] and NSGA-II [9]. We evaluate the efficiency of the algorithms by comparing the convergence and diversity of solutions produced after performing 10000 function evaluations. For each test function, 30 independent runs are taken, and the following metrics [10] are adopted.

Hypervolume Indicator (HV): This measures the convergence and diversity of the obtained solutions. The closer are the solutions to the true Pareto fronts, the larger is the value of HV.

Generational distance (GD): This indicates the difference between obtained solutions and the true Pareto fronts.

Additive epsilon indicator (Iε+): This relies on the epsilon dominance concept, it measures how close the obtained solutions is to the true Pareto fronts.

ZDT (ZDT1, ZDT2, ZDT3, ZDT4, ZDT6) and DTLZ (DTLZ1, DTLZ3) benchmark test functions [11] are taken to verify the new proposed MOPSO, and two objectives are optimized. ZDT4 is a highly multi-modal function which has 21 local Pareto-optimal fronts.

All the algorithms have been implemented in Java using MOEA Framework [12]. For the three PSO-based algorithm, the population size and external archive size were both set to 100, C1 and C2 were assigned randomly between 1.5 and 2.0, r1 and r2 were assigned randomly between 0.0 and 1.0. For the new algorithm, the inertia weight varied in [0.5, 0.9]. For OMOPSO, the inertia weight was assigned randomly between 0.1 and 0.4. For SMPSO, the inertia weight was set to 0.1 and the mutation rate was set to 1/ (number of decision variables). For NSGA-II, the population size was set to 100, crossover rate was set to 0.9 and the mutation rate was set to 1/ (number of decision variables).

Table 1 to 3 tabulate the performance of all algorithms in terms of HV, GD and Iε+ in different test functions. A number highlighted in bold is the best results in such corresponding test function. The tables show that the proposed algorithm generally outperformed the other three MOP algorithms in all ZDT problems (except for ZDT6), according to all the indicators. For ZDT6 problem, the proposed algorithm obtains the best HV values, and the second best GD, Iε+ values. ZDT4 is a well-known problem characterized by having many local optima. We can observe that the proposed algorithm clearly outperformed the other three MOP algorithms for ZDT4, it has a high capability to solve multi-modal functions.

For the two DTLZ test problems, only the proposed algorithm and OMOPSO are able to converge to the true Pareto-front, their performance are comparable in terms of HV and Iε+. The rest of the MOP algorithms achieves a HV value very close 0, meaning that the solution sets obtained are outside the limits of the Pareto front. For the results in terms of GD, SMPSO outperformed the three other algorithms.

The proposed algorithm yielded five best mean values of HV, GD, and four best mean values of Iε+.

4 CONCLUSION

This paper has presented an improved multi-objective particle swarm optimizer with a new adaptive inertia weight strategy and bound handling method. We compared the performance of the new approach and three other popular MOP algorithms,

OMOPSO, SMPSO and NSGA-II with a budget of 10000 function evaluations, the new approach generally outperformed three other algorithms in terms of Hypervolume, Generational distance and Additive epsilon indicator in seven benchmark test problems. The proposed MOPSO is more efficient, also it is able to solve multi-objective problems with many local optima, and avoid decision variables getting the extreme values. As part of our future work, we intend to further enhance the diversity of solution sets.

Table 1: Performance comparison in HV

Test Function	HV	Proposed MOPSO	OMOPSO	SMPSO	NSGA-II
ZDT1	Mean	**6.31E-01**	6.24E-01	5.91E-01	6.02E-01
	Std.	1.71E-04	5.18E-03	7.64E-02	5.17E-03
ZDT2	Mean	**3.27E-01**	3.25E-01	3.08E-01	2.74E-01
	Std.	8.76E-05	2.06E-03	4.83E-02	4.70E-02
ZDT3	Mean	**4.84E-01**	4.75E-01	3.75E-01	4.66E-01
	Std.	3.91E-03	9.29E-03	7.70E-02	4.92E-03
ZDT4	Mean	**4.71E-01**	2.17E-01	2.59E-01	1.44E-01
	Std.	2.66E-01	2.7E-01	2.7E-01	1.46E-01
ZDT6	Mean	**3.99E-01**	3.99E-01	3.99E-01	1.07E-01
	Std.	3.00E-04	5.00E-04	4.57E-04	3.13E-02
DTLZ1	Mean	3.24E-01	0.00	**4.73E-01**	6.48E-02
	Std.	4.28E-02	0.00	5.83E-02	1.24E-01
DTLZ3	Mean	1.1E-01	0.00	**1.61E-01**	0.00
	Std.	2.52E-02	0.00	5.79E-02	0.00

Table 2: Performance comparison in GD

Test Function	GD	Proposed MOPSO	OMOPSO	SMPSO	NSGA-II
ZDT1	Mean	**2.91E-04**	9.78E-04	4.67E-03	2.67E-03
	Std.	5.92E-05	3.78E-04	7.98E-03	4.43E-04
ZDT2	Mean	**2.27E-04**	7.24E-04	4.17E-03	4.29E-03
	Std.	3.18E-05	2.21E-04	1.04E-02	1.22E-03
ZDT3	Mean	**2.09E-03**	3.27E-03	2.34E-02	4.20E-03
	Std.	8.30E-04	9.82E-04	3.44E-02	7.44E-04
ZDT4	Mean	**3.91E-02**	1.4E-01	6.86E-02	6.57E-02
	Std.	7.15E-02	2.47E-01	6.04E-02	3.32E-02
ZDT6	Mean	**8.23E-04**	8.35E-04	9.72E-04	5.57E-02
	Std.	3.90E-05	5.00E-04	4.57E-04	3.13E-02
DTLZ1	Mean	12.90	-	**7.1E-01**	3.64E-01
	Std.	8.16	4.52	2.73	2.74E-01
DTLZ3	Mean	49.20	-	**8.31**	-
	Std.	26.50	16.05	12.40	2.52

Table 3: Performance comparison in Iε+

Test Function	Iε+	Proposed MOPSO	OMOPSO	SMPSO	NSGA-II
ZDT1	Mean	**8.37E-03**	1.39E-02	3.94E-02	2.81E-02
	Std.	1.75E-03	3.97E-03	5.37E-02	3.51E-03
ZDT2	Mean	**7.89E-03**	1.06E-02	4.39E-02	1.16E-01
	Std.	2.19E-03	1.40E-03	8.89E-02	1.97E-01
ZDT3	Mean	**1.46E-02**	2.86E-02	1.59E-01	4.54E-02
	Std.	5.15E-03	1.60E-02	8.66E-02	5.41E-02
ZDT4	Mean	**2.67E-01**	7.47E-01	5.45E-01	6.08E-01
	Std.	4.9E-01	8.28E-01	4.59E-01	3.07E-01
ZDT6	Mean	1.12E-02	**1.04E-02**	1.22E-02	4.75E-01
	Std.	2.66E-03	3.05E-03	1.64E-03	6.97E-02
DTLZ1	Mean	4.22E-01	-	**4.52E-02**	1.07
	Std.	1.14E-01	8.71	1.24E-01	7.38E-01
DTLZ3	Mean	6.64E-01	-	**2.69E-01**	-
	Std.	1.3E-01	3.97E+01	3.39E-01	5.31

REFERENCES

[1] Mishra, E. A. K., Mohapatra, E. Y., & Mishra, E. A. K. (2013). Multi-Objective Genetic Algorithm: A Comprehensive Survey. *International Journal of Emerging Technology and Advanced Engineering*, 3(2), 81-90.

[2] Kumar, V., & Minz, S. (2014). Multi-objective particle swarm optimization: An introduction. *SmartCR*, 4(5), 335-353.p.346

[3] Reyes-Sierra, M., & Coello, C. C. (2006). Multi-objective particle swarm optimizers: A survey of the state-of- the-art. *International journal of computational intelligence research*, 2(3), 287-308.

[4] Shi, Y., & Eberhart, R. (1998, May). A modified particle swarm optimizer. *In Evolutionary Computation Proceedings, 1998. IEEE World Congress on Computational Intelligence., The 1998 IEEE International Conference* on (pp. 69-73). IEEE.

[5] Durillo, J. J., García-Nieto, J., Nebro, A. J., Coello, C. A. C., Luna, F., & Alba, E. (2009, April). Multi-objective particle swarm optimizers: An experimental comparison. In *International Conference on Evolutionary Multi-Criterion Optimization* (pp. 495-509). Springer, Berlin, Heidelberg.

[6] Agarwal, D., & Sharma, D. (2016). Experimental Study on Bound Handling Techniques for Multi-objective Particle Swarm Optimization. In *Innovations in Bio-Inspired Computing and Applications* (pp. 555-564). Springer, Cham

[7] Sierra, M. R., & Coello, C. A. C. (2005, March). Improving PSO-based multi-objective optimization using crowding, mutation and∈-dominance. In *International Conference on Evolutionary Multi-Criterion Optimization* (pp. 505-519). Springer, Berlin, Heidelberg.

[8] Nebro, A. J., Durillo, J. J., Garcia-Nieto, J., Coello, C. C., Luna, F., & Alba, E. (2009, March). Smpso: A new pso-based metaheuristic for multi-objective optimization. In *Computational intelligence in miulti-criteria decision-making, 2009. mcdm'09. ieee symposium on* (pp. 66-73). IEEE.

[9] Deb, K., Agrawal, S., Pratap, A., & Meyarivan, T. (2000, September). A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II. In *International Conference on Parallel Problem Solving From Nature* (pp. 849-858). Springer, Berlin, Heidelberg.

[10] Coello, C. A. C., Lamont, G. B., & Van Veldhuizen, D. A. (2007). *Evolutionary algorithms for solving multi-objective problems* (Vol. 5). New York: Springer.

[11] Shi, Y., & Eberhart, R. C. (1999). Empirical study of particle swarm optimization. In *Evolutionary computation, 1999. CEC 99. Proceedings of the 1999 congress on* (Vol. 3, pp. 1945-1950). IEEE.

[12] Hadka, D. MOEA Framework (2017). http://www.moeaframework.org/