Tianrungroj Yossathorn
03200437

Assignment 10
Solving Multi-objective Optimization Problem with Meta-Heuristic

Multi-objective Optimization Problem (MOP) is a problem with more than one objective function needed to be maximized or minimized. The goal is to find a frontier line on the plot of objective functions which also called Pareto front. Solutions on Pareto front are the most efficient solution, there is no solution that has better evaluation on every objective than the one on Pareto front. There are many methods to find or to approximate the Pareto front of the MOP. In this assignment, MOP is solved using two meta-heuristic algorithms, Particle Swarm Optimization (PSO) and Artificial Bee Colony Algorithm (ABC). To keep it simple, the number of objectives for MOP is fixed to two.

1 Particle Swarm Optimization (PSO)

Particle Swarm Optimization, or in this case Multiple Objective Particle Swarm Optimization (MOPSO) is a meta-heuristic based on the behavior of social animal, especially birds and fishes. These animals often travel in a group. Although leader of the group might be unclear, yet the group reaches their destination efficiently. The reason lies on the number of members in the group. If there are enough members, or ‘particles’, there should be at least one of them that could accidentally find a better spot and guide everyone else. The detailed will be explained in the following subsection.

The algorithm and implementation in this assignment mostly based on the first half of Kogiso’s study, the first item in the reference section.

1.1 Algorithm Explanation: PSO

First, let’s start with single objective optimization problem using PSO. The particles represent candidate solutions. The algorithm starts with randomly initialized particles within a search space or constraint. Then, these particles move iteratively according to a rule. Let’s say particles are represented by vectors $\boldsymbol{x}_{i=1\dots n}$. Personal bests are defined as vectors corresponding to each particle at its highest evaluation according to the problem’s objective function. For a particle x_i, at iteration t, its personal best represented by $\boldsymbol{x}_{p_i}^t$. Lastly, global best is a vector with highest evaluation among every particle up until the current iterations. At iteration t, global best is represented by $\boldsymbol{x}_g^t$. The movement rule of the next iteration $t+1$, for particle i is represented by velocity vector v_i^{t+1} and defined as follow, equation 3, and the movement is an addition of vectors in equation 4.

$$\boldsymbol{x}_{p_i}^t = \max_{t'=0\dots t} \boldsymbol{x}_{p_i}^{t'} \tag{1}$$

$$\boldsymbol{x}_g^t = \max_{\substack{\mathrm{i=1...n} \\ \mathrm{t'=0...}t}} \boldsymbol{x}_{p_i}^{t'} \tag{2}$$

$$\boldsymbol{v}_i^{t+1} = w\boldsymbol{v}_i^t + C_1 r_1\left(\boldsymbol{x}_{p_i}^t - \boldsymbol{x}_i^t\right) + C_2 r_2(\boldsymbol{x}_g^t - \boldsymbol{x}_i^t) \tag{3}$$

$$\boldsymbol{x}_i^{t+1} = \boldsymbol{x}_i^t + \boldsymbol{v}_i^{t+1} \tag{4}$$

Here in equation 3, there are 3 parameters w, C_1 and C_2, and 2 random variables r_1 and r_2. Parameter w represents the amount of inertia. Parameters C_1 and C_2 are acceleration constant toward personal best and global best respectively. Random variables r_1 and r_2 are scalar between 0 to 1 interval. They could be interpreted as a chance for exploitation or exploration of each particle.

The flow of algorithm is straight forward. In each iteration, personal best of each particle and global best is updated according to the previous value and the current particles' evaluation. Based on these values and defined parameters, the velocity vectors for each particle are calculated. Lastly, particles move according to the velocity vectors. The algorithm runs until the terminal condition, maximum number of iterations is reached for example. The global best is the best solution found by PSO for the single objective optimization problem.

1.2 Algorithm Explanation: MOPSO

Next, to use PSO on a multi-objective optimization problem, there are some modifications needed. In MOP, not only optimizing the objectives, spreading the candidate solutions (particles) across the Pareto front is also necessary. The particles after some iterations of normal PSO might be crowded at single spot on the Pareto front which isn't what we need. In this assignment, I used two method, the first method is sigma method, and the second method is the improved sigma method which I come up with myself to improve the algorithm on MOP with concave-down Pareto front.

1.2.1 Sigma Method

The reason why normal PSO isn't good on MOP is because of the global best definition, one global best cannot be determined in MOP. Sigma-method aims to tackle this problem by dividing particles into groups, each group corresponded to an interval of sigma value. Particles with sigma values laying in the same interval belong to the same group. The sigma value of each particles can be found using the equation 5 below where $f1$ and $f2$ are MOP's objective functions. In each iteration of MOPSO, the velocity vector of a particle is determined using the global best of the group that particle belong to. There is also a chance for particles to move across the groups, so group assignment is also needed to be done in every iteration. The global best of each group is also referred as local best or guiding particle in this assignment.

$$\sigma = \frac{f_1^2 - f_2^2}{f_1^2 + f_2^2} \tag{5}$$

In Kogiso's study, there is no definition of how to determine the guiding particle. In each group, there could be multiple particles with rank 1 or in other words, particle that has no other particle with better evaluation on every objective function. To use the sigma method, one guiding particle per group is recommended as it saves the computational cost, so I defined fitness function *F* below in equation 6. The fitness function is defined as a negative value of the addition of the two objective functions. In each group of particles, a particle with the highest fitness function will be assigned as a guiding particle of the group. This fitness function is also being used to define personal best of each particle.

$$F(\boldsymbol{x}_i) = -f_1(\boldsymbol{x}_i) - f_2(\boldsymbol{x}_i) \tag{6}$$

1.2.2 Improved Sigma Method

After I tested the MOPSO with sigma method, it worked good on the first MOP, but on the second one which has a concave-down Pareto front, it didn't perform well. I tried modifying a little on the sigma method and the result was spectacular.

The modification that was to use a random guiding particle to calculate the velocity vector of each particles. To put it simply, the guiding particles are still assigned within each group of particles with similar sigma value. However, each particle uses a random guiding particle regardless of the group it belongs to. The reason that I came up with this modification is because I found that the initial batch of particles likely have a biased sigma value. Most of the initial batch of particles are belonging to the same group. In case of sigma method, the particles are less likely to move to another group. This problem is also amplified when the Pareto front of the MOP is concave-down. Detailed explanation will be discussed in the Result and Discussion section.

1.3 Implementation

In this section, the implementation of the MOPSO algorithms used in this assignment will be explained. The problems used to test each algorithm together with the evaluation methods will also be explained in the following subsections. The entire implementation was done from scratch using Python with NumPy library. One disclaimer needed to be clarified is that I have miss-typed one terminology in the code and figures. By 'guided particle', it was meant to be 'guiding particle'.

1.3.1 MOPSO implementation

The main algorithm of MOPSO was implemented under a class called *PSO*. The constructor method takes 3 arguments, a *Problem* instance, number of particles and sigma bins. The PSO parameters, *w*, *C1* and *C2*, are also allowed to be specified in the constructor but aren't required. In that case, the parameters will be set to 0.4, 2 and 2 respectively. Before the algorithm begins, the *PSO* instance need to be initialized using *init_PSO* method. Then, to start the algorithm, *train* method is needed to be called together with the number of iterations as an argument. The method will return the objective functions of every particles in the last iteration. Other attribute such as particles' position, guiding particles' position, fitness value of particles, are also accessible through the *PSO* instance.

The algorithm also repositions particles that have moved out of the constraint boundary using the method call *reposToConstraints*. The method repositions particles by moving them back to the closest coordinates within the constraint boundary.

During the training process, before the velocities vector are calculated, the guiding particles are assigned to every particle using *getGuided* method. For the normal version of MOPSO, *getGuided* method simply assigns the guiding particle based on the sigma value. In contrast, the improved version of MOPSO's *getGuided* method assigns the guiding particle to every particle randomly.

The *Problem* class is being used to pass the problem's objective functions and constraints to the *PSO* instance as an argument of the constructor mentioned above. Solutions to the problem are also included in the *Problem* class but doesn't being used in *PSO* class.

1.3.2 Multi-Objective Problems

Problems used to test the algorithms in this assignment are Multi-model MOP and Biased Pareto-optimal front problem. The former is represented by *prob1 Problem* instance while the latter is represented by *prob2 Problem* instance. The dimension (n) for both problems were set to 3. The alpha value for Biased Pareto-optimal front is adjustable. For this assignment, alpha value of 2 and 0.5 were tested.

1.3.3 Evaluation method

To compare between MOPSO method, range of the particles' sigma value is being used as a numerical metric. It can be calculated by simply subtracting the lowest particle's sigma value from the highest particle's sigma value.

Visualization is also being used to evaluate the performance of the algorithm.

1.4 Result and Discussion

The problem with normal MOPSO due to the biased sigma value initialization will be discussed. Then the following subsection will be the result and discussion of the normal MOPSO and the improved version on both problems. Parameters were fixed to the value shown in Table 1 for every experiment.

Table 1 MOPSO Parameter settings

Parameters	Value
W	0.4
C1	2
C2	2
num_particle	10000
sigma_bins (6bins)	`[-1.0, -0.93, -0.766, -0.174, 0.5, 0.94, 1.0]`
max_iter	100

1.4.1 Biased sigma value initialization

To visualize the biased sigma problem, the plot of the two objectives value of initial particles is shown in Figure 1 below. The objective functions in this case are from the second problem, Biased Pareto-optimal from problem. From the figure, almost every particle has relatively high objective value for the second objective function compare to the first. This means that most of them have a very high sigma value and in the grouping process, they will all be assigned to the same group.

In case of normal PSO algorithm, where the particles are less likely to move to other group, this problem is expected to worsen the performance. On the other hand, the improved version where particles are allowed to move across the group, should perform better. The detailed comparisons for both normal and improved PSO are shown in the next subsections.

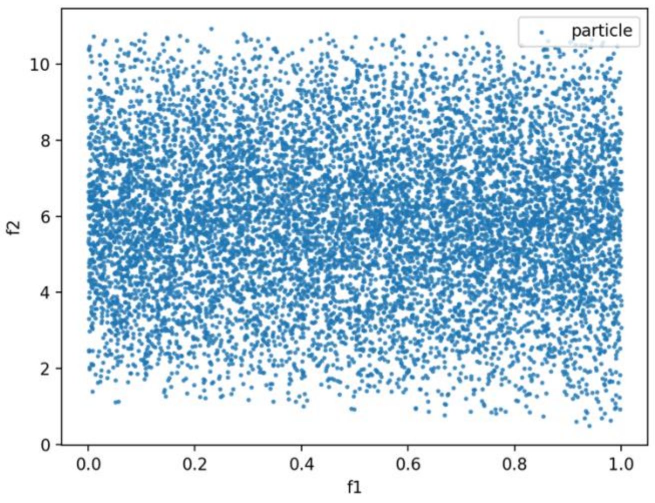

Figure 1 Plot of the two objective values of the initial 10000 particles.

1.4.2 Multi-model MOP

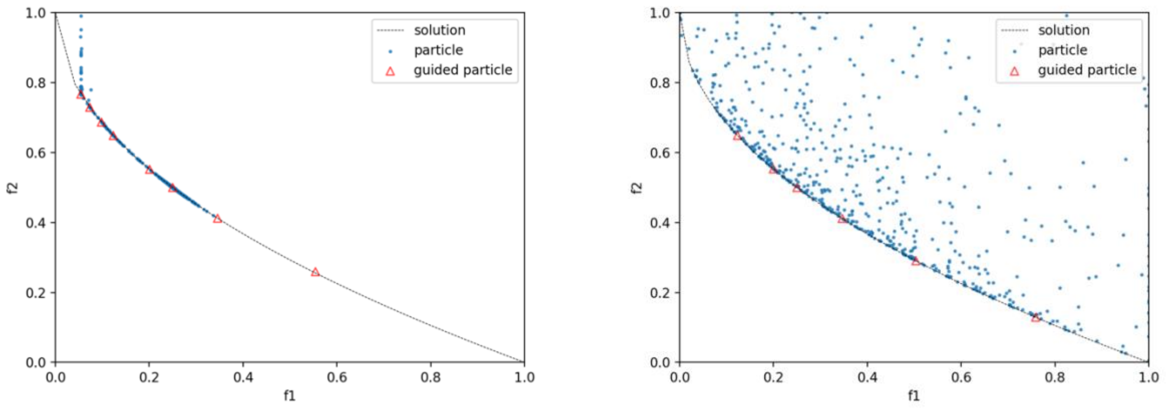

Figure 2 Plots of Multi-model MOP objective values of particles in the last iteration for normal (left) and improved (right) MOPSO

The result from both normal and improved PSO for the first problem, Multi-model MPO is shown in Figure 2. The dotted curve represents the actual Pareto front for the problem. Blue dots represent particles while red triangles represent the guiding particles.

On the left plot of Figure 2, as expected, the normal PSO's particles at the final iteration gather at one spot and the guiding particles, red triangle, cover only a small region of the Pareto front. The reason for particles gathering at one spot lies in the fitness function definition. The spot that particles are moving toward has the highest fitness value.

On the right plot of Figure 2, particles and guiding particles are more distributed across the Pareto front. This is because of the random guiding particle assignment that increases variety in particles' velocity while maintaining the tendency of moving toward the Pareto front.

1.4.3 Biased Pareto-optimal front problem

The second problem, Biased Pareto-optimal front problem, has two variations, concave down and concave up, depend on the parameter alpha.

Concave down (alpha=2)

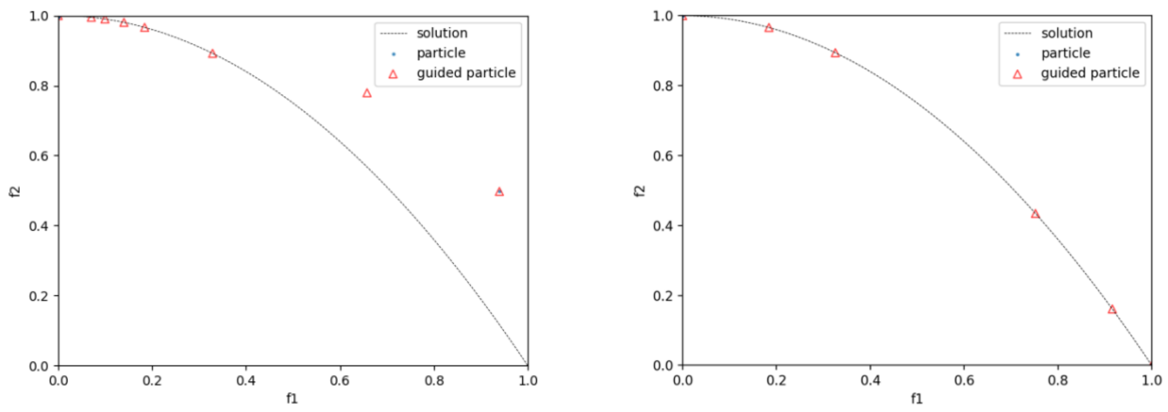

Figure 3 Plots of Biased Pareto-optimal front (concave down) objective values of particles in the last iteration for normal (left) and improved (right) MOPSO

As shown in Figure 3, for both plots, the particles converged at the top of the pareto front. Again, this is because of the fitness function definition. However, the guiding particles of the normal PSO didn't reach the Pareto front. The reason might be that that guiding particle came from other sigma group and their personal best position was keeping it away from the Pareto front. This isn't the case for the improved MOPSO because there is a higher chance of particles to move across from another sigma group which also allows them to perform a PSO with each other locally.

Concave up (alpha=0.5)

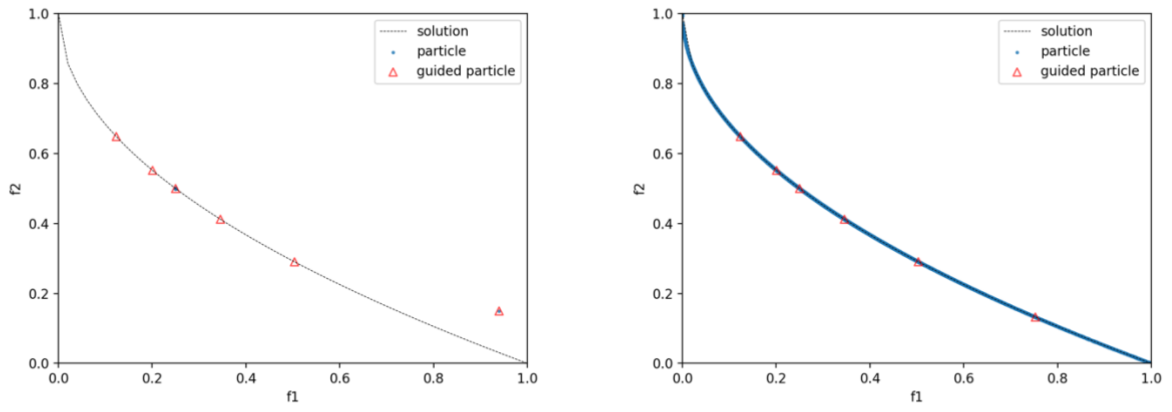

Figure 4 Plots of Biased Pareto-optimal front (concave up) objective values of particles in the last iteration for normal (left) and improved (right) MOPSO

Similar to the concave down variation of the problem and Multi-model MOP, particles gather at one spot in where the fitness function is the highest for the normal MOPSO. Also, the guiding particles distributed widely but does not converged at the marginal region. In contrast, the improved MOPSO performed spectacularly. Particles converged and distributed evenly across the Pareto front.

1.4.4 Numerical evaluation

Table 2 Numerical metrics: Range of the particles' sigma value

	Normal MOPSO	Improved MOPSO
Multi-model MOP	1.522859	1.999999
Biased Pareto-optimal front (concave up)	1.562449	1.999998
Biased Pareto-optimal front (concave down)	1.880524	1.999998