Artificial Intelligence 791 Assignment 2 Option 2

Multi-Modal Multi-Guide Particle Swarm Optimization for Solving Multi-Objective Optimization Problems

Simeon Boshoff
Division of Computer Science
Stellenbosch University
Stellenbosch, South Africa
22546510@sun.ac.za

***Abstract*—Particle Swarm Optimization (PSO) is a good method for solving optimization problems in multiple dimensions, but only for single-objective functions. In this report, PSO algorithms that possess the ability to solve multi-objective problems, namely multi-guide PSOs (MGPSO), are analysed and compared to determine which algorithm performs best. An implementation of the inertia weight PSO is created in which multiple swarms each minimize a single objective function in a MOP respectively. These solutions are then exchanged between swarms via an archive, which contains non-dominated solutions found by all swarms. The swarms use tournament selection to randomly select a guide from the archive, which draws the swarms nearer to the Pareto optimal front. In order for respective swarms to converge, stability conditions need to be met, thus the parameters need to be selected carefully. This is then compared to an implementation of the Speciation MGPSO algorithm, to obtain benchmark results. This is achieved by running several simulations under different conditions and multi-objective functions, comparing the effectiveness of the methods. The results indicate that the Speciation MGPSO performs better for most, but not all test cases.**

***Index Terms*—MOPs, MGPSO, Speciation, Swarm, Multi-objective**

I. Introduction

Multi-modal multi-guide particle swarm optimization (MGPSO) is a computational method that can solve multi- and many-objective optimization problems (MOPs)[1]. The multi-modal MGPSO makes use of multiple swarms, where each swarm is tasked with independently optimizing an individual objective function within the MOP. The best positions found by the swarms are exchanged by use of an archive, which contains non-dominated solutions found by the swarms individually. As with the inertia weight PSO, swarm particles have inertia, cognitive and social components, but a fourth is added namely the archive guide component. The archive guide is randomly selected via tournament selection during every iteration and is used to guide the swarm to the current positions of the non-dominated solutions found within the archive. The potency of the guide is determined by a λ value, which dictates the balance between the social, and archive guide components.

The primary method of the MGPSO execution is by repeatedly updating each particle position by some velocity, which is dependent on the above-mentioned components. For each iteration, particles move around in n-dimensional space, finding more optimal solutions, and inserting non-dominated solutions into the archive. The goal is to obtain the Pareto optimal front, defined as a set of solutions that are non-dominated to each other but are superior to the rest of the solutions in the search space [2]. For MGPSO to converge, the parameters that influence the components need to abide by the convergence condition, further discussed in the background section.

To evaluate the performance of the inertia weight MGPSO, another state-of-the-art algorithm is used for comparison, namely Speciation. This method of PSO is similar to the inertia weight PSO but divides particles into sub-swarms (seeds) based on the proximity of particles to neighbours. Sub-swarms are created during every iteration, and allow particles to find the minima of multi-model optimization problems. The same convergence parameters are used for Speciation, as with normal inertia weight PSO. The grounds on which the two algorithms are compared, consist of the Inverted Generational Distance (IGD) to the Pareto optimal front of the function, as well as the Hypervolume.

II. Background

In the space of solving MOPs, many different algorithms have attempted to obtain optimal results, with the vector-evaluated PSO (VEPSO) being one of the more prominent candidates [3]. However, recent analysis of VEPSO has shown that the algorithm exhibits some flaws. The algorithm shows stagnation behaviour because it does not sufficiently exploit already found good candidate solutions [4]. The MGPSO aims to address the shortcomings of the VEPSO, by implementing the archive and archive guide component. In order for inertia weight particle swarms to behave optimally, we introduce stochasticity by multiplying each of the components in the velocity update by a uniform random value $r_i \in [0, 1]$.

A. Multi-guide particle swarm optimization

The MGPSO makes use of 5 components when updating particle velocity, each with a constant component weight:

- w: Inertia
- c_1: Cognitive (Scaled by r_1)
- c_2: Social (Scaled by r_2)
- c_3: Archive (Scaled by r_3)
- λ: Social-Archive balance

These components need to satisfy the following condition, in order to converge on the positions of non-dominated solutions in the archive:

$$c_1+\lambda c_2+(1-\lambda)c_3 < \frac{4\left(1-w^2\right)}{1-w+\frac{\left(c_1^2+\lambda^2c_2^2+(1-\lambda)^2c_3^2\right)(1+w)}{3(c_1+\lambda c_2+(1-\lambda)c_3)^2}} \quad (1)$$

With $|w| < 1$.

The MGPSO execution pseudo code is contained in Algorithm 1:

Algorithm 1 Multi-guide Particle Swarm Optimization (MGPSO) [3]

for each objective $m = 1, ..., n_m$ **do**
  Let f_m be the objective function;
  Create and initialize a swarm, S_m, to contain $S.n_{sm}$ particles
  for each particle $i = 1, ..., S.n_{sm}$ **do**
    Initialize position $S_m.x_i(0)$ uniformly within a predefined hypercube of dimension n_x;
    Initialize the personal best position as $S_m.y_i(0) = S_m.x_i(0)$;
    Determine the neighbourhood best position, $S_m.\hat{y}_i(0)$;
    Initialize the velocity as $S_m.v_i(0) = \mathbf{0}$;
    Initialize $S_m.\lambda_i \sim U(0,1)$;
  end for
end for
Let $t = 0$;
repeat
  for each objective $m = 1, ..., n_m$ **do**
    for each particle $i = 1, ..., S_m.n_s$ **do**
      if $f_m(S_m.x_i) < f_m(S_m.y_i)$ **then**
        $S_m.y_i = S_m.x_i(t)$
      end if
      for particles $\hat{i}$ with particle i in their neighbourhood **do**
        if $f_m(S_m.y_i) < f_m(S_m.\hat{y} - \hat{i})$ **then**
          $S_m.\hat{y} - \hat{i} = S_m.y_i$
        end if
      end for
      Update the archive with the solution $S_m.x_i$;
    end for
  end for
  for each objective $m = 1, ..., n_m$ **do**
    for each particle $i = 1, ..., S_m.n_s$ **do**
      Select a solution $S_m.\hat{a}_i(t)$, from the archive using tournament selection;
      $S_m.v_t(t+1) = wS_m.v_t(t) + c_1r_1(S_m.y_i(t) - S_m.x_i(t)) + S_m\lambda_i c_2 r_2(S_m.\hat{\lambda}_t(t) - S_m.x_i(t)) + (1 - S_m\lambda_i)c_3r_3(S_m.\hat{a}_i(t) - S_m.x_i(t))$
      $S_m.x_i(t+1) = S_m.x_i(t) + S_m.v_i(t+1)$
    end for
  end for
  $t = t + 1$
until stopping condition is true =0

1) Crowding distance: For the MGPSO to explore the whole true Pareto front and not just a section, crowding distance needs to be used to spread out solutions in the archive. When the archive is full, and an attempt is made to insert a new non-dominated solution, the element with the smallest crowding distance is removed from the archive, and the new element is added. This ensures that solutions that are more spread out across the search space are favoured. For the ejection of the most crowded particle, the archive needs to be sorted according to crowding distance. Figure 1 illustrates how crowding distance is calculated:

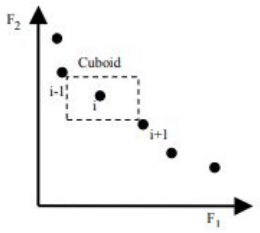

Fig. 1. Crowding distance calculation [4]

B. Speciation PSO

Speciation PSO is based on the inertia weight PSO, and thus makes use of the same components for particle velocity update:

- w : Inertia
- c_1 : Cognitive (Scaled by r_1)
- c_2 : Social (Scaled by r_2)

The components need to satisfy the convergence condition below, to converge on an optimal solution.

$$c_1 + c_2 \leq \frac{24\left(1-w^2\right)}{7-5w} \quad (2)$$

With $|w| < 1$.

The speciation PSO uses the same algorithm as the inertia weight PSO, but with the following added to create sub-swarms:

Algorithm 2 Speciation PSO pseudo code [1]

1: Generate an initial swarm with randomly generated particles.
2: Evaluate all particles in the swarm.
3: Sort all particles in descending order of their fitness value.
4: Assign each species seed identified as the n_best to all particles identified in the same species.
5: Adjust each particle's velocity and position
6: If termination criterion is not met, go to step 2. =0

Sub-swarms are created on the bases of forming new neighbourhoods (seeds) of particles that are in close proximity of each other. Algorithm 3 explains how these seeds are determined:

Algorithm 3 Determining species seeds [1]

Let L_{sorted} = all particles sorted in decreasing order of fitness.
Let S_{seed} = {}
while L_{sorted} contains *unprocessed particles* ***do***
  Get the first unprocessed particle x_L *in* L_{sorted}
  Let found = FALSE
  for *each particle* x_s *in* S_{seed} ***do***
    if $d(x_S, x_L) \leq r_s$ ***then***
      found = TRUE
      Break from loop
    end if
    if *(not found)* ***then***
      let $S_{seed} = S_{seed} \cup \{x_L\}$
    end if
    Mark x_L *as processed*

C. The Speciation MGPSO

To evaluate the performance of the multi-modal MGPSO, it is compared to the Speciation MGPSO. This makes use of the same algorithm as the multi-modal MGPSO, but adds the creation of sub-swarms (species) at every iteration.

D. Purpose of study

For this assignment, the impact of using a multi-modal MGPSO algorithm instead of an inertia weight MGPSO is investigated by use of comparing the inverted generational distance and hypervolume per iteration of each algorithm to the Pareto optimal fronts of the respective multi-objective functions. The goal is to determine which works best.

III. Implementation

The data obtained in this experiment is calculated by the execution of a Java program, with simulates the movement of the particles, and manages the archive. The archive at any iteration is stored in a .csv file, which is then read by a Python3 script. The purpose of the Python3 script is to visualise the data, as well as to calculate the performance metrics. The reason why two different programs are used is because Java performs much faster than Python3 when it comes to simulating huge amounts of movements. Each simulation is run multiple times, in order to obtain consistent results.

For the sake of this study, two evaluation metrics are used, noted below:

- Inverted Generational Distance
- Hypervolume

A. Inverted Generational Distance (IGD)

The IGD performance indicator inverts the generational distance and measures the distance from any point in Z to the closest point in A.

$$\mathrm{IGD}(A) = \frac{1}{|Z|}\left(\sum_{i=1}^{|Z|} \hat{d}_i^p\right)^{1/p}$$

where $\hat{d}_i$ represents the Euclidean distance $(\mathrm{p} = 2)$ from z_i to its nearest reference point in A [6]. The IGD calculation is illustrated in Figure 2:

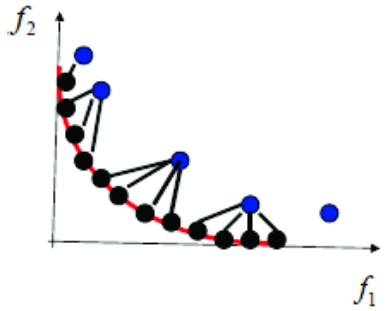

Fig. 2. Inverted generational distance illustration [6]

Even though IGD can give qualitative data surrounding the performance of an MGPSO, it does not reflect the crowding of the solutions.

B. Hypervolume

Unlike IGD, the Hypervolume performance indicator does not require the real Pareto front to be known, instead, it makes use of a reference point **r**, which enables the calculation of area/volume between the point and all solutions in the archive.

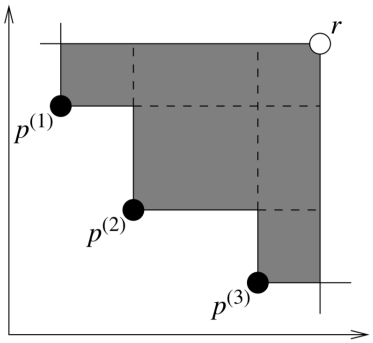

Fig. 3. Hypervolume illustration [7]

The figure shows a two-objective example where the area which is dominated by a set of points in the archive is shown in grey. This gives qualitative data on the spread of the archive elements.

IV. Empirical process

A. Selected Benchmark Functions

The performance of both multi-modal MGPSO and speciation MGPSO will be measured by using five different benchmark functions. To create a diverse problem set for testing, a variety of benchmark functions is obtained from Zitzler, Deb and Thiele [8] to gauge performance. n is the number of dimensions of the function.

1) ZDT1:

$$
\begin{aligned}
&g(x) = 1 + 9(\sum_{i=2}^{n} x_i)/(n-1) \\
&F_1(x) = x_1 \\
&F_2(x) = g(x)[1 - \sqrt{x_1/g(x)}] \\
&x \in [0,1]
\end{aligned} \tag{3}
$$

2) ZDT2:

$$
\begin{aligned}
&g(x) = 1 + 9(\sum_{i=2}^{n} x_i)/(n-1) \\
&F_1(x) = x_1 \\
&F_2(x) = g(x)[1 - (x_1/g(x))^2)] \\
&x \in [0,1]
\end{aligned} \tag{4}
$$

3) ZDT3:

$$
\begin{aligned}
&g(x) = 1 + 9(\sum_{i=2}^{n} x_i)/(n-1) \\
&F_1(x) = x_1 \\
&F_2(x) = g(x)[1 - \sqrt{x_1/g(x)} \\
&\quad - x_1/g(x)sin(10\pi x_1))] \\
&x \in [0,1]
\end{aligned} \tag{5}
$$

4) ZDT4:

$$
\begin{aligned}
&g(x) = 91 + \sum_{i=2}^{n} [x_i^2 - 10cos(4\pi x_i)] \\
&F_1(x) = x_1 \\
&F_2(x) = g(x)[1 - \sqrt{x_1/g(x)}] \\
&x_1 \in [0,1], x_i \in [-5,5] i = 2, ..., 10.
\end{aligned} \tag{6}
$$

5) ZDT6:

$$
\begin{aligned}
&g(x) = 1 + 9[(\sum_{i=2}^{n} x_i)/(n-1)]^{0.25} \\
&F_1(x) = 1 - exp(-4x_1)sin^6(6\pi x_1) \\
&F_2(x) = g(x)[1 - (f_1(x)/g(x))^2] \\
&x \in [0,1]
\end{aligned} \tag{7}
$$

B. Simulation parameters

Values for w, $c1$, $c2$, and $c3$ are randomly selected, and then λ is selected such that the convergence condition (1) is satisfied. However, this can lead to underwhelming results, illustrated in Figure 4. Due to the stochastic nature of the MGPSO, it can occur that the generated parameters satisfy the condition, but result in poor behaviour.

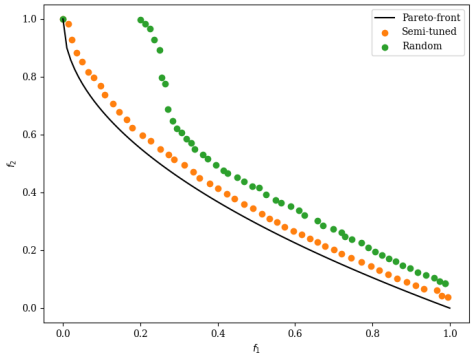

Fig. 4. Pareto front for semi-tuned and random parameters

To improve consistency of results, parameter values are chosen which are relatively close the tuned parameters, found in [3]. Thus parameter bounds are slightly refined, shown in the following table:

$w \sim$	uniform[0.4, 0.5]
$c_1 \sim$	uniform[1.5, 2.1]
$c_2 \sim$	uniform[0.8, 1.4]
$c_3 \sim$	uniform[1.5, 2.1]

TABLE I
PARAMETER VALUES

In this study, the simulation will be run across:

- 2 swarms.
- 50 archive size.
- 10000 iterations.
- 30 dimensions (10 dimensions for ZDT4 and ZDT6).
- 2 tournament size.

V. Results

In this section, the simulation results and observations are discussed in detail, showing the efficiency of a multi-modal MGPSO as well as a speciation MGPSO. Firstly, the multi-modal MGPSO algorithm is assessed on ability to solve the multi-objective problems, with iterations, swarm size and archive size taken into account. Next, the multi-modal MGPSO is compared to the speciation MGPSO on the basis of IGD and hypervolume.

A. Multi-Modal MGPSO

The multi-modal MGPSO has exhibited good, consistent behaviour with regards to solving multi-objective programming problems, and obtain results close to the true Pareto front. To illustrate the performance metrics, the following figures show how the MGPSO finds the true Pareto front, the IGD over time as well as the hypervolume:

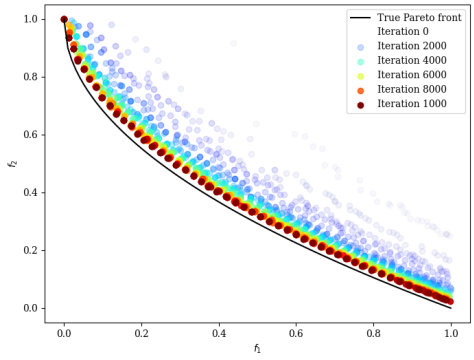

Fig. 5. Archive values per 2000 iterations. ZDT1

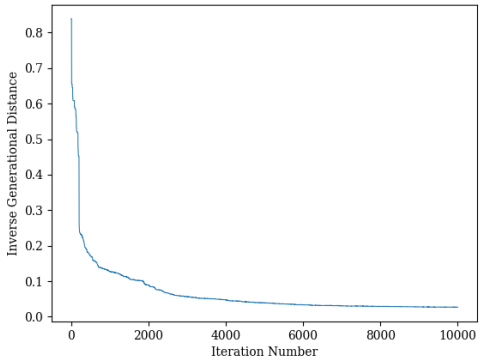

Fig. 6. IGD per iteration. ZDT1

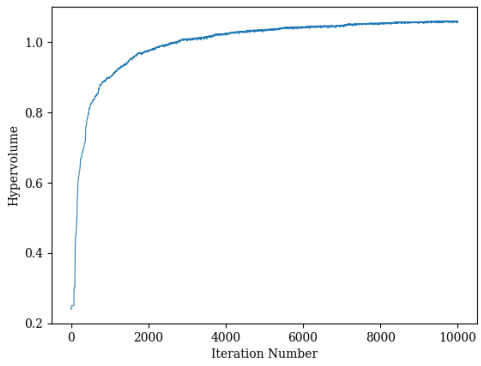

Fig. 7. Hypervolume per iteration. ZDT1

When evaluating IGD, lower is better. This means that the objective of the MGPSO is to obtain a low IGD as fast as possible. When evaluation hypervolume, higher is better, since hypervolume measures the area of the archive points relative to the reference.

Next, the ability of the inertia weight MGPSO to find the Pareto fronts of ZDT2, ZDT3, ZDT4 and ZDT5 is shown:

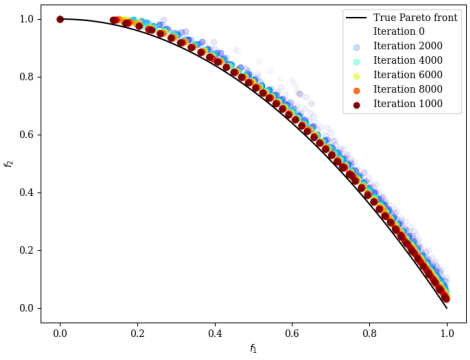

Fig. 8. Archive values per 2000 iterations. ZDT2

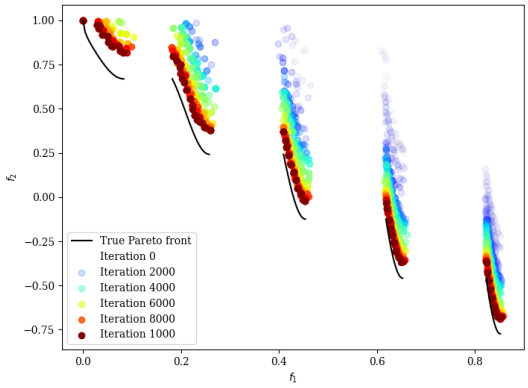

Fig. 9. Archive values per 2000 iterations. ZDT3

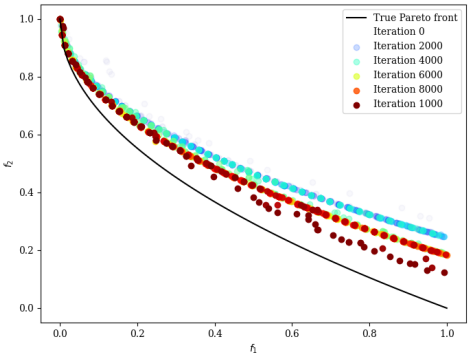

Fig. 10. Archive values per 2000 iterations. ZDT4

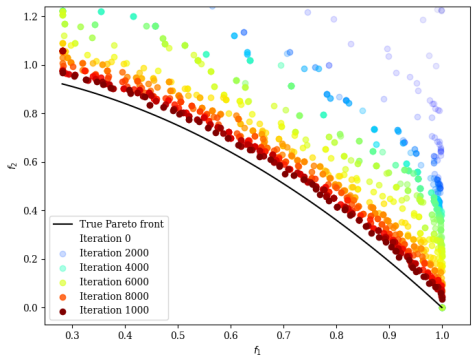

Fig. 11. Archive values per 2000 iterations. ZDT6

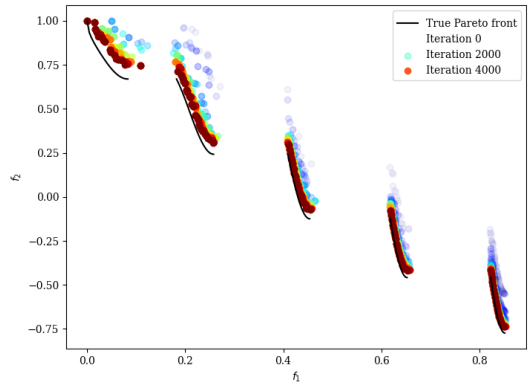

Fig. 13. Archive values per 2000 iterations. ZDT3 (Archive size = 100)

As can be observed, the true Pareto front for some of the selected benchmark functions is relatively easy to obtain, whilst others, such as ZDT4 (Figure 10) are more difficult. This creates a good metric for comparing the performance of two MGPSOs. Due to the nature of the functions, as well as the number of dimensions the simulations are performed on, "relatively" close to the Pareto front counts as a success. To illustrate the effect of archive size, a comparison will be drawn between $n = 10$ and $n = 100$, using ZDT3. The same random seed is used for both experiments, to eliminate stochastic anomalies.

As Figures 12 and 13 show, the capacity of the archive does not have a notable effect on the ability of the MGPSO to find the true Pareto front, however, it enables the particles to exhibit more exploratory behaviour, since the archive guide is selected from a more diverse group.

B. Comparing inertia weight MGPSO with speciation MGPSO

To make a conclusive comparison between the inertia weight MGPSO and speciation MGPSO, 50 trials are run across the 5 benchmark functions. To determine which MGPSO is better, the lowest IGD obtained, as well as the highest hypervolume is taken into consideration. The test results were obtained by generating graph similar to the ones below, multiple times:

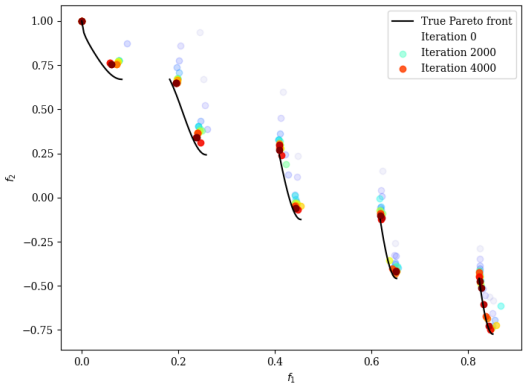

Fig. 12. Archive values per 2000 iterations. ZDT3 (Archive size = 10)

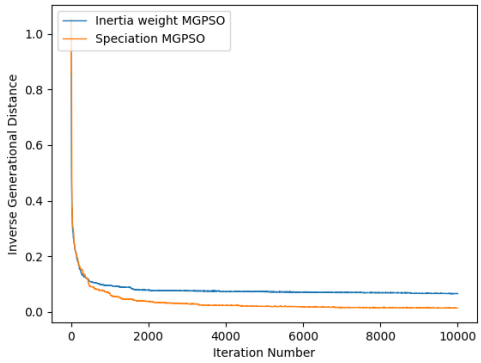

Fig. 14. Speciation vs Inertia Weight MGPSO. IGD. ZDT3 (D = 30)

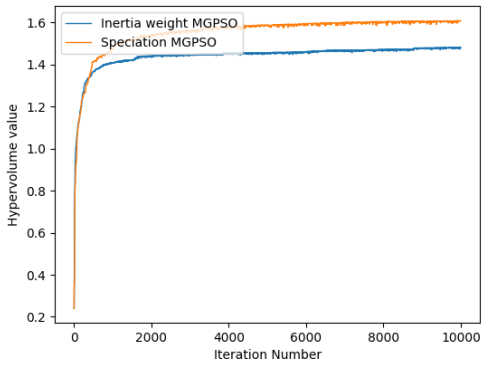

Fig. 15. Speciation vs Inertia Weight MGPSO. Hypervolume. ZDT3 (d = 30)

In the case above, the Speciation MGPSO shows to obtain a smaller IGD value, due to the independent exploration of neighbourhoods. The inertia weight MGPSO briefly reaches a smaller IGD value, but does not explore enough to find the exact Pareto front.

C. Trial results

10 trials were run per benchmark function, and the following results have been obtained:

	ZDT1	ZDT2	ZDT3	ZDT4	ZDT6	Total
IW MGPSO	Wins	Loses	Loses	Wins	Loses	2
S MGPSO	Loses	Wins	Wins	Loses	Wins	3

TABLE II
TRIAL VALUES

VI. Conclusion

The results indicate that Speciation MGPSO is better for the majority of the benchmark functions, however, due to the high amount of variability introduced by velocity update parameter randomization, it can not be concluded that Speciation MGPSO is overall superior. If the parameters were tuned, more consistent behaviour would lead to more conclusive results. Both algorithms show astounding ability to solve multi-objective programming problems, and do not converge when the true Pareto front is not obtained - in some cases, adding more iterations leads to the front being reached. The crowding distance calculations added to the archive succeeded in ensuring that archive elements are spread out well, and that the true Pareto front is covered in its entirety. Overall, we can conclude that the inertia weight MGPSO is a good method for solving MOO problems, and modifying it to include Speciation shows to improve performance in most cases.

References

[1] A.P. Engelbrecht, “Computational Intelligence, An Introduction. Second Edition” University of Pretoria, South Africa, pp. 343–360, 2007.
[2] M. Akbari, P. Asadi, M.K. Besharati Givi, G.Khodabandehlouie, “Artificial neural network and optimization”. Definition of a Pareto set. Chapter 13.
[3] C. Scheepers, A.P. Engelbrecht, C.W. Cleghorn, “Multi-guide particle swarm optimization for multi-objective optimization: empirical and stability analysis”, August 2019, pp. 1–11.
[4] W. Matthysen, A.P. Engelbrecht, K.M. Malan,“Analysis of stagnation behavior of vector evaluated particle swarm optimisation.” In Proceedings of the IEEE swarm intelligence symposium. IEEE. 2013
[5] Kalyanmoy Deb, Associate Member, IEEE, Amrit Pratap, Sameer Agarwal, and T. Meyarivan, “A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II”, 2002, p. 4.
[6] “Inverted Generational Distance Figure”, M. Sulaiman, 2015
[7] “Hypervolume illustration”, Pymoo.com, https://pymoo.org/misc/indicators.html
[8] N. Chase, M. Rademacher, E. Goodman, “A Benchmark Study of Multi-Objective Optimization Methods”, Michigan State University, East Lansing, MI, pp. 5–22