A Comparison of Particle Swarm Optimisation Methods for Solving Constrained Optimisation Problems

E. J. van den Heever
Stellenbosch University, Department of Computer Science
22547134
22547134@sun.ac.za

***Abstract*—This paper compares two particle swarm optimisation algorithms for solving constrained optimisation problems, namely multi-guide particle swarm optimisation and new vector particle swarm optimisation. Four performance measures are used to compare the algorithms, namely reliability, consistency, efficacy, and efficiency. This research found that new vector particle swarm optimisation performs better than multi-guide particle swarm optimisation for all measures. Furthermore, new vector particle swarm optimisation is guaranteed to find solutions within the feasible region, which is not true for multi-guide particle swarm optimisation.**

Keywords—particle swarm optimisation, constrained optimisation problems, NVPSO, MGPSO

I. Introduction

Particle swarm optimisation (PSO) is a swarm intelligence algorithm and widely used technique for optimising benchmark functions [1, p. 289]. This paper compares the performance of two PSO algorithms for solving constrained optimisation problems (COPs). A COP consists of an objective function and a set of constraints used to define the feasible region. COPs can generally be described as follows:

$$\begin{aligned} &\text{minimise} && f(\boldsymbol{x}) \\ &\text{subject to} && g_i(\boldsymbol{x}) \leq 0, i = 1, \dots, n_g \\ &&& h_j(\boldsymbol{x}) = 0, j = 1, \dots, n_h \\ &&& x_{d\,\min} \leq x_d \leq x_{d\,\max}\ d = 1, 2, \dots, n \end{aligned} \tag{1}$$

where $\boldsymbol{x} = (x_1, x_2, \dots, x_n)$ is the vector of solutions such that $\boldsymbol{x} \in S \subseteq \mathbb{R}^n$. S is the n-dimensional space contained in the upper and lower bounds $[x_{d\,\min}, x_{d\,\max}], d = 1, 2, \dots, n$. n_g and n_h are the number of inequality and equality constraints respectively and the feasible region $F \subseteq S$ is the region of S such that all inequality and equality constraints are satisfied. A solution $\boldsymbol{x}$ is a feasible solution if it satisfies all constraints.

Scheepers *et al.* [2] presented a new PSO algorithm for optimising multi-objective optimisation problems (MOPs). The algorithm, named Multi-Guide Particle Swarm Optimisation (MGPSO), is a multi-swarm approach that makes use of one subswarm to optimize each objective. The velocity update equation includes an archive guide which facilitates convergence to a Pareto front of non-dominated solutions. This research converts COPs to MOPs by defining one objective function for all the constraints as follows:

$$p(\boldsymbol{x}_i) = \sum_{k=1}^{n_g+n_h} \lambda_k p_k(\boldsymbol{x}_i) \tag{2}$$

where

$$p_k(\boldsymbol{x}_i) = \begin{cases} \max\{0, g_k(\boldsymbol{x}_i)^\theta\}, & \text{if } k \in [1, \dots, n_g] \\ |h_k(\boldsymbol{x}_i)|^\theta, & \text{if } k \in [n_g + 1, \dots, n_h] \end{cases}$$

with θ a positive constant representing the power of the penalty. This research assumes that $\theta = 1$ and $\lambda_k = 1, \forall k \in [1, \dots, n_g + n_h]$. The boundary constrained MOP is then expressed as

$$\begin{aligned} &\text{minimise} && f(\boldsymbol{x}) = f\big(f_c(\boldsymbol{x}), p(\boldsymbol{x})\big) \\ &\text{subject to} && x_j \in [x_{j\,\min}, x_{j\,\max}], j = 1, \dots, n \end{aligned} \tag{3}$$

where f_c refers to the original constrained objective function, n is the number of decision variables, and $x_{j\,\min}$ and $x_{j\,\max}$ specify the range of valid variables that can be assigned to decision variable x_j.

Sun *et al.* [3] proposed a new vector particle swarm optimisation (NVPSO) for solving COPs. The algorithm makes use of a shrinkage coefficient to ensure that the dimensions of a particle remain within the upper and lower bounds of the problem. Additionally, a function $\phi(\boldsymbol{x})$ is used to determine whether a particle is within the feasible region. When a particle escapes the feasible region, one-dimensional search optimisation methods are used to produce a new position that is guaranteed to be in the feasible region.

This study evaluates the performance of MGPSO and NVPSO by comparing the consistency with which they find solutions, the reliability of the solutions found, the efficiency of the algorithm and the goodness of the solutions produced by the algorithm. This research found that NVPSO performs better than MGPSO for all measures. Additionally, the empirical analysis showed that NVPSO was guaranteed to produce solutions within the feasible region, while the probability that MGPSO produced a solution in the feasible region was dependent on the problem.

This paper is structured as follows: The second section gives necessary background for this paper, including a brief description of the new vector and multi-guide PSO algorithms. The third section outlines the methodology for this research, while the fourth section discusses the chosen empirical procedure. Section five presents the results and a discussion thereof. The sixth and final section provides a conclusion for the study.

II. Background

Particle Swarm Optimisation (PSO) is a nature-inspired stochastic optimisation algorithm introduced by Kennedy and Eberhart [4]. A swarm of particles is initialised within the

search space. Particles then move through the search space over time, using global and local information to inform their movement. The position of a particle represents a possible solution to an optimisation problem. This section describes two adaptations of PSO, multi-guide PSO (MGPSO) and new vector PSO (NVPSO), which can be used to solve constrained optimisation problems (COPs).

A. Multi-Guide Particle Swarm Optimisation

MGPSO is a multi-swarm particle swarm optimisation algorithm that enables swarms to share information through an archive. Each subswarm is assigned a different objective and the quality of particles in a subswarm is evaluated according to the objective function assigned to that subswarm. The personal best positions within a subswarm are referred to as local guides and are updated based on the objective function of that subswarm. MGPSO expands the velocity update equation of standard global best PSO to include an archive guide that facilitates the exchange of information between subswarms. The velocity update equation is defined as follows:

$$\begin{aligned}\boldsymbol{v}_i(t+1) = w\boldsymbol{v}_i(t) &+ c_1\,\boldsymbol{r}_1\big(\boldsymbol{y}_i(t) - \boldsymbol{x}_i(t)\big) \\ &+ \lambda_i c_2 \boldsymbol{r}_2\big(\hat{\boldsymbol{y}}_i(t) - \boldsymbol{x}_i(t)\big) \\ &+ (1-\lambda_i) c_3 \boldsymbol{r}_3\big(\hat{\boldsymbol{a}}_i(t) - \boldsymbol{x}_i(t)\big)\end{aligned} \tag{4}$$

where $\boldsymbol{v}_i(t)$ is the velocity of particle i at iteration t, w is the inertia weight, c_1, c_2 and c_3 are the acceleration coefficients, $\boldsymbol{r}_1$, $\boldsymbol{r}_2$ and $\boldsymbol{r}_3$ are random vectors with components sampled randomly from ~U(0, 1), $\boldsymbol{x}_i(t)$ is the position of particle i at iteration t, $\boldsymbol{y}_i(t)$ is the personal best position of particle i at iteration t, $\hat{\boldsymbol{y}}_i(t)$ is the neighbourhood best position of a particle i at iteration t, λ_i is the exploitation trade-off coefficient for particle i and $\hat{\boldsymbol{a}}_i(t)$ is global guide selected from the archive using tournament selection for particle i at iteration t. The position of particle i at iteration $t+1$, $\boldsymbol{x}_i(t+1)$ is calculated as

$$\boldsymbol{x}_i(t+1) = \boldsymbol{x}_i(t) + \boldsymbol{v}_i(t+1) \tag{5}$$

The exploitation trade-off coefficient, λ_i, controls the amount of influence that the archive guide has on the velocity of a particle. The archive implemented in MGPSO is a bounded archive that makes use of a crowding distance archive implementation, as described by Deb *et al.* [5]. When a new non-dominated solution is found by one of the swarms, and it is not dominated by any other solution in the archive, it is added to the archive, provided the archive is not full. If the archive is full, the most-crowded, non-dominated solution is removed from the archive and the newly found solution is added. The archive guide $\hat{\boldsymbol{a}}_i(t)$ is selected from a competition pool as the solution with the largest corresponding crowding distance in the archive. The competition pool is constructed by randomly selecting several entries from the archive, the exact number of which is defined by the tournament size.

The MGPSO algorithm does not need to be adapted to solve constrained optimisation problems, but rather, constrained optimisation problems need to be reformulated as multi-optimisation problems (MOPs) according to (2). MGPSO does not provide a single solution to an MOP, but rather, the archive is constructed to contain a selection of solutions that optimise the different objectives to varying degrees. Since a feasible solution for a COP is one for which none of the constraints are violated, the solution in the archive that produces the minimum value for the penalty function is the solution that should be chosen. If the penalty function defined by (2) does not evaluate to 0 for a given solution, then the solution is not a feasible solution. The selection of an optimal solution from the archive is done after the algorithm has run to completion. Algorithm 1 describes MGPSO.

Algorithm 1: Multi-Guide Particle Swarm Optimisation

for each objective $m = 1, \dots, n_m$ **do**
  Let f_m be the objective function;
  Create and initialise swarm S_m to contain $S.n_{s_m}$ particles
  for each particle $i = 1, \dots, S.n_{s_m}$ **do**
    // Initialise position $S_m.\boldsymbol{x}_i(0)$ uniformly
    $S_m.\boldsymbol{x}_i(0) \sim U(x_{i\,\min}, x_{i\,\max})$;
    // Initialise the personal best position as
    $S_m.\boldsymbol{y}_i(0) = S_m.\boldsymbol{x}_i(0)$;
    Determine the neighbourhood best position, $S_m.\hat{\boldsymbol{y}}_i(0)$;
    Initialise the velocity as $S_m.\boldsymbol{v}_i(0) = \boldsymbol{0}$;
    Initialise $S_m.\lambda_i \sim U(0,1)$;
  end for
end for
Let $t = 0$;
repeat
  for each objective $m = 1, \dots, n_m$ **do**
    for each particle $i = 1, \dots, S_m.n_s$ **do**
      if $f_m(S_m.\boldsymbol{x}_i) < f_m(S_m.\boldsymbol{y}_i)$ **then**
        $S_m.\hat{\boldsymbol{y}}_i = S_m.\boldsymbol{x}_i(t)$;
      end if
      for particles $\hat{\imath}$ with particle i in their neighbourhood **do**
        if $f_m(S_m.\boldsymbol{y}_i) < f_m(S_m.\hat{\boldsymbol{y}}_{\hat{\imath}})$ **then**
          $S_m.\hat{\boldsymbol{y}}_{\hat{\imath}} = S_m.\boldsymbol{y}_i$;
        end if
      end for
    end for
  for each objective $m = 1, \dots, n_m$ **do**
    for each particle $i = 1, \dots, S_m.n_s$ **do**
      Select a solution, $S_m.\hat{\boldsymbol{a}}_i(t)$, from the archive;
      Update $S_m.\boldsymbol{v}_i(t+1)$ using (4);
      Update $S_m.\boldsymbol{x}_i(t+1)$ using (5);
    end for
  end for
  $t = t + 1$;
until *stopping condition is true*

B. New Vector Particle Swarm Optimisation

NVPSO is an adaptation of standard global best PSO that ensures that particles remain within the feasible region by employing a shrinkage coefficient and a function $\phi(\boldsymbol{x})$ that only evaluates to 1 when the solution does not violate any of the constraints. The velocity and position update equations used by NVPSO are the same as those proposed by Kennedy and Eberhart for standard global best PSO [4]. The velocity of a particle i at iteration $t+1$ is calculated as

$$\begin{aligned}\boldsymbol{v}_i(t+1) = w\boldsymbol{v}_i(t) &+ c_1\boldsymbol{r}_1\big(\boldsymbol{y}_i(t) - \boldsymbol{x}_i(t)\big) \\ &+ c_2\boldsymbol{r}_2\big(\hat{\boldsymbol{y}}_i(t) - \boldsymbol{x}_i(t)\big)\end{aligned} \tag{6}$$

and the position of particle i at iteration $t+1$ is calculated according to (5).

If the new position of a particle generated by (5) and (6) is not within the upper and lower bound constraints for each dimension, a shrinkage coefficient is introduced. The shrinkage coefficient guarantees that a particle is within the

upper and lower bounds of each dimension, and is calculated as follows:

$$\alpha = \frac{m_d - x_{i,d}(t)}{x_{i,d}(t+1) - x_{i,d}(t)}, d = 1, 2, \dots, n \tag{7}$$

where m_d denotes the lower and upper bounds of the d-th dimension. When the particle exceeds the upper bound, $m_d = x_{d\,\max}$ and when the particle exceeds the minimum, $m_d = x_{d\,\min}$. The minimum shrinkage coefficient, α' is selected as

$$\alpha' = \min\{\alpha_d\}, d = 1, 2, \dots, n$$

The new position of the i-th particle is then modified using the shrinkage coefficient as follows:

$$\boldsymbol{x}_i(t+1) = \boldsymbol{x}_i(t) + \alpha'\big(\boldsymbol{x}_i(t+1) - \boldsymbol{x}_i(t)\big) \tag{8}$$

The function $\phi(\boldsymbol{x})$ is used to determine whether a point is in the feasible region.

$$\phi(\boldsymbol{x}) = \prod_{i=1}^{n_g} e^{\max\{0, g_i(\boldsymbol{x})\}} \prod_{i=n_g+1}^{n_g+n_h} e^{\max\{0, |h_i(\boldsymbol{x})|\}} \tag{9}$$

All the positions where $\phi(\boldsymbol{x}) = 1$ are within the feasible region and any positions where $\phi(\boldsymbol{x}) > 1$ are outside of the feasible region. Particles that leave the feasible region are handled by letting the particle move to a position between its current position and the former position, according to (10).

$$\boldsymbol{x}_i(t+1) = \boldsymbol{x}_i(t) + \beta\big(\boldsymbol{x}_i(t+1) - \boldsymbol{x}_i(t)\big), \beta \in [0, 1) \tag{10}$$

When the new position of a particle is not in the feasible region, a value for $\beta \in [0, 1)$ is selected to ensure the minimal value of ϕ. Finding the value of β can be treated as a one-dimensional search problem. β is calculated such that

$$\min_{\beta \in [0,1)} \phi\big(\boldsymbol{x}_i(t+1)\big) = \min_{\beta \in [0,1)} \phi\left(\boldsymbol{x}_i(t) + \beta\big(\boldsymbol{x}_i(t+1) - \boldsymbol{x}_i(t)\big)\right) \tag{11}$$

Algorithm 2: *New Vector Particle Swarm Optimisation*

Create and initialize an n_x-dimensional swarm of particles within the feasible region
repeat
  for *each particle* $i = 1, \dots, n_s$ **do**
    if $\boldsymbol{x}_i \notin [x_{\min}, x_{\max}]$ **do**
      Calculate α_d using (7) for each dimension of particle i;
      Calculate α' as $\min\{\alpha_d\}, d = 1, 2, \dots, n$;
      Update the new position according to (8);
    end if
    Calculate $\phi(\boldsymbol{x}_i)$ using (9);
    if $\phi(\boldsymbol{x}_i) > 1$ **do**
      Update the position of particle i according to (10);
    end if
    // set the personal best position
    if $f(\boldsymbol{x}_i) < f(\boldsymbol{y}_i)$ **then**
      $\boldsymbol{y}_i = \boldsymbol{x}_i$;
    end if
    //set the global best position
    if $f(\boldsymbol{y}_i) < f(\hat{\boldsymbol{y}})$ **then**
      $\hat{\boldsymbol{y}} = \boldsymbol{y}_i$;
    end if
    update the velocity using (6)
    update the position using (5)
  end for
until *stopping condition has been met*

C. Choice of Control Parameters

The performance of MGPSO and NVPSO is dependent on the choice of control parameters, w, c_1, c_2 and c_3. Scheepers *et al.* [2] found that the choice of control parameters should be tuned for each MOP to achieve the best results using MGPSO. However, they also showed that order 1 and order 2 stability is achieved if the choice of control parameters satisfies the following equation:

$$0 < c_1 + \lambda c_2 + (1-\lambda)c_3 < \frac{4(1-w)^2}{1 - w + \dfrac{(c_1^2 + \lambda^2 c_2^2 + (1-\lambda)^2 c_3^2)(1+w)}{3(c_1 + \lambda c_2 + (1-\lambda)c_3)^2}}, |w| < 1 \tag{12}$$

Since NVPSO is an adaptation of the standard global best PSO, the stability conditions proposed by Engelbrecht [1, p. 307] can be used to ensure that particle trajectories are guaranteed to converge.

$$w > \frac{1}{2}(c_1 + c_2) - 1 \tag{13}$$

III. Methodology

This section describes the implementation of MGPSO and NVPSO for solving COPs. This research implements both variations of PSO in Java according to Algorithms 1 and 2. A stopping condition of 5000 iterations is used and swarm sizes of 40 are chosen. Note that for MGPSO, this means that subswarms have a size of 20, because MGPSO has two objectives when it is used to solve COPs. Particles are initialised with a velocity of $\mathbf{0}$. For MGPSO, particle positions are initialised randomly according to the following equation:

$$\boldsymbol{x}(0) = x_{d\,\min} + r_d(x_{d\,\max} - x_{d\,\min}), \forall d = 1, \dots, n \tag{14}$$

where $r_d \sim U(0, 1)$ is generated using Math.random() in Java. For NVPSO, particles are initialised to a position at the boundary of each dimension and then the position is adjusted randomly, one dimension at a time, until it is within the feasible region for the problem. The personal best position of each particle is initialised to its initial position, $\boldsymbol{x}(0)$.

This research evaluates MGPSO and NVPSO simulations for five different COPs, using 20 independent samples collected for each COP. The control parameters w, c_1, c_2 and c_3 were randomly generated such that they satisfied the stability conditions in (12) and (13) for MGPSO and NVPSO respectively.

IV. Empirical Procedure

This research aimed to compare the performance of two PSO algorithms on their ability to solve COPs. The algorithms were assessed on four performance measures, namely: the consistency of solutions found, the reliability with which the algorithm finds feasible solutions, the efficiency of the algorithm in finding solutions and the goodness of the solutions. Five COPs were selected from Liang *et al.* [6] to evaluate the performance of MGPSO and NVPSO in solving

constrained optimisation problems. g01 is a 13-dimensional problem with nine inequality constraints, g04 is a 5-dimensional problem with six inequality constraints, g07 is 10-dimensional problem with eight inequality constraints, g08 is a 2-dimensional problem with two inequality constraints, and g09 is a 7-dimensional problem with four inequality constraints. Note that these benchmark functions were converted to MOPs according to (2) and (3) before they were solved using MGPSO. Table I summarises the objective functions, dimensions and known solutions of the selected suite of benchmark functions.

TABLE I. SUITE OF BENCHMARK FUNCTIONS

Name	**Function Properties**		
	Objective Function	***Problem Dimensions***	***Known Solution***
g01	$f(\boldsymbol{x}) = 5\sum_{i=1}^{4} x_i - 5\sum_{i=1}^{4} x_i^2 - \sum_{i=5}^{13} x_i$	$0 \le x_i \le 1, i = 1, \dots, 9$ $0 \le x_i \le 100, i = 10, 11, 12$ $0 \le x_{13} \le 1$	$f(\boldsymbol{x}^*) = -15$
g04	$f(\boldsymbol{x}) = 5.3578547x_3^2 + 0.8356891x_1x_5 + 37.293239x_1 - 40792.141$	$78 \le x_1 \le 102$ $33 \le x_2 \le 45$ $27 \le x_i \le 45, i = 3, 4, 5$	$f(\boldsymbol{x}^*) = -30665.5$
g07	$f(\boldsymbol{x}) = x_1^2 + x_2^2 + x_1x_2 - 14x_1 - 16x_2 + (x_3 - 10)^2 + 4(x_4 - 5)^2 + (x_5 - 3)^2 + 2(x_6 - 1)^2 + 5x_7^2 + 7(x_8 - 11)^2 + 2(x_9 - 10)^{2^{x_{10}-7}} + 45$	$-10 \le x_i \le 10, i = 1, \dots, 10$	$f(\boldsymbol{x}^*) = 24.30621$
g08	$f(\boldsymbol{x}) = -\frac{\sin^3(2\pi x_1)\sin(2\pi x_2)}{x_1^3(x_1 + x_2)}$	$0 \le x_1 \le 10$ $0 \le x_2 \le 10$	$f(\boldsymbol{x}^*) = -0.09583$
g09	$f(\boldsymbol{x}) = (x_1 - 10)^2 + 5(x_2 - 12)^2 + x_3^4 + 3(x_4 - 11)^2 + 10x_5^6 + 7x_6^2 + x_7^4 - 4x_6x_7 - 10x_6 - 8x_7$	$-10 \le x_i \le 10, i = 1, \dots, 7$	$f(\boldsymbol{x}^*) = 680.6301$

To verify the implementation of MGPSO, it was tested on MOPs with known Pareto fronts. If MGPSO is implemented correctly, the plot of the objective function values contained in the archive at the end of the simulation should be representative of the Pareto front for the function. Fig. 1. shows the plots of archives generated using MGPSO for the zdt1, zdt2 and zdt3 benchmark functions proposed by Deb [7]. The plots in Fig. 1. are consistent with the known Pareto fronts for the respective functions, from which it can be concluded that MGPSO was implemented correctly to solve multi-objective optimisation problems.

MGPSO and NVPSO were evaluated using four performance measures. Note that when MGPSO produced infeasible solutions, the average and standard deviation was calculated over the feasible solutions only.

A. Reliability

The reliability of the algorithm was used as a measure of how likely the algorithm is to produce a feasible solution. By the definition of Algorithm 2, NVPSO is gauranteed to produce a feasible solution for constrained optimisation problems. However, the archive used in MGPSO is not gauranteed to contain a solution for which the penalty function (2) evaluates to 0. The reliability of MGPSO was calculated as the percentage of runs for which a feasible solution was produced.

B. Consistency

The consistency of solutions produced by an algorithm gives an indication of the likelihood that the algorithm will produce similar solutions over many independent runs. The standard deviation between final solutions produced by each algorithm describes the similarity of solutions and therefore, was used as a measure of consistency. The smaller the standard deviation between solutions, the more consistently the algorithm performed.

C. Efficacy

The goodness of solutions found by an optimisation algorithm gives an indication of how effective the algorithm is at solving a problem. This research evaluated the solutions

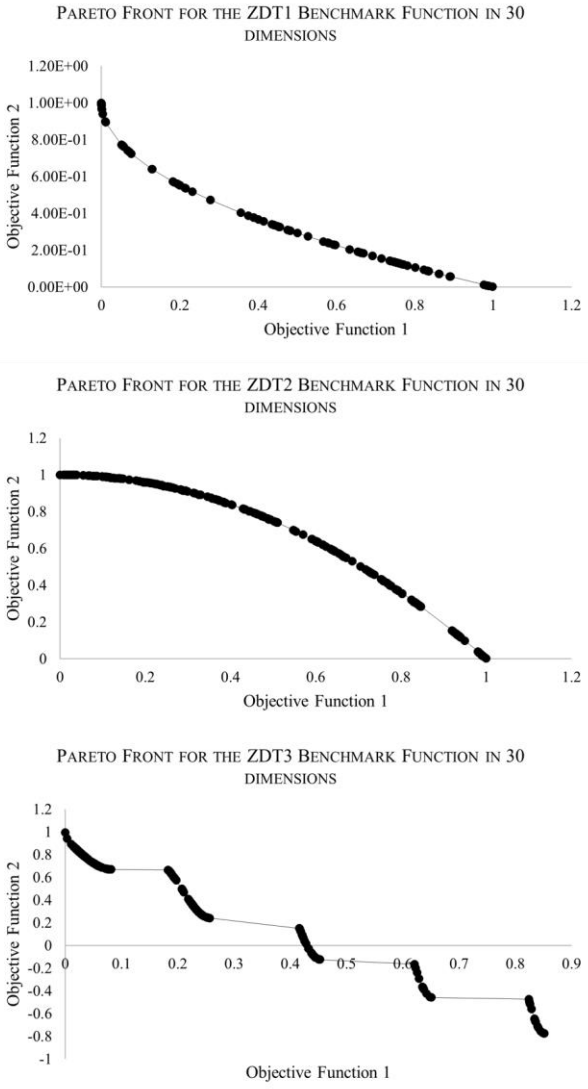

Fig. 1. Plots of the objective function values for the solutions contained in the archive after 5000 iterations of MGPSO on the zdt1, zdt2 and zdt3 benchmark functions.

produced in the best, average, and worst case as a measure of the efficacy of an algorithm.

D. *Efficiency*

The efficiency of an algorithm describes how quickly the algorithm produced a solution. This research measured the rate at which each algorithm converged on a solution. The faster an algorithm converged to a solution, the more efficient the algorithm was at solving the constrained optimisation problem.

V. Results & Discussion

This section presents empirical results and observations of the performance of MGPSO and NVPSO for solving the constrained optimisation problems described in Table I. Tables II and III contain the best, worst, and average solutions found after the last iteration of MGPSO and NVPSO respectively. As well as are the known minimums for each benchmark function and the standard deviations between the solutions found during the final iteration of each algorithm. Note that for MGPSO, the average and standard deviation are calculated across all feasible solutions, which may exclude some of the 20 independent runs of the algorithm. For the g04, g07 and g09 benchmark functions, all 20 independent runs of the MGPSO algorithm produced feasible solutions. However, for the g01 benchmark function, only four out of the 20 runs produced feasible solutions and for g08, 19 out of the 20 runs produced feasible solutions. From the definition of Algorithm 2, NVPSO is guaranteed to produced feasible solutions.

TABLE II. Experimental Results for MGPSO

Name	Results for MGPSO at $t = 5000$				
	Known Minimum	*Best Solution*	*Worst Solution*	*Average solution*	*Standard Deviation*
g01	-15	-3.30062	-1.43260	-2.23946	0.819814
g04	-30665.5	-30500.2	-29840.5	-30262.4	158.9911
g07	24.30621	37.59871	79.69111	51.66558	10.97471
g08	-0.09583	-0.09496	-0.00205	-0.06741	0.025966
g09	680.6301	692.8229	954.9944	719.5378	54.81573

TABLE III. Experimental Results for NVPSO

Name	Results for NVPSO at $t = 5000$				
	Known Minimum	*Best Solution*	*Worst Solution*	*Average solution*	*Standard Deviation*
g01	-15	-14.5	-12.2765	-13.7585	0.670431
g04	-30665.5	-30665.5	-30665.5	-30665.5	2.16×10^{-10}
g07	24.30621	24.32078	25.75837	24.70225	0.327995
g08	-0.09583	-0.09583	-0.09583	-0.09583	3.18×10^{-17}
g09	680.6301	680.6301	680.6309	680.6302	2.21×10^{-4}

Fig. 2. shows the average global best position found over time for MGPSO and NVPSO for the g01 benchmark function. Note that for MGPSO, the global best position is the minimum feasible solution contained in the archive at each iteration. Fig. 2. shows that MGPSO only identified the first feasible solutions after 955 iterations. From the plot, it is evident that NVPSO quickly converges on a solution that is close to the known minimum of the g01 benchmark function. By contrast, MGPSO takes longer to converge on a solution, and the solution that it converges on is much further from the known minimum of the g01 benchmark function. This suggests that NVPSO is both more efficient as it converges faster, and more effective as it produces a better solution than MGPSO. From Tables II and III, it is evident that even in the worst case, NVPSO produces a better solution than MGPSO. Additionally, MGPSO only produced feasible solutions for 20% of the independent runs, making it less reliable than NVPSO for the g01 benchmark function. Table II shows a larger standard deviation for the g01 benchmark function than Table III, indicating that NVPSO converged to the same solution more consistently than MGPSO. Therefore, on the g01 benchmark function, NVPSO produces better results, is more efficient, more reliable, and performs more consistently than MGPSO.

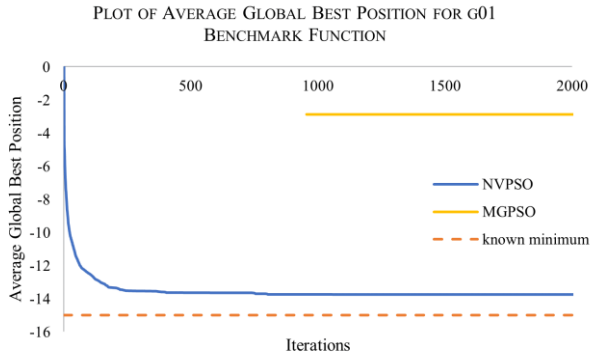

Fig. 2. Plot of the average best solution found across 20 independent runs of MGPSO and NVPSO on the g01 benchmark function.

Fig. 3. shows the average global best position over 20 independent runs of the NVPSO and MGPSO algorithms for the g04 benchmark function. The plot indicates that, although both algorithms converge on solutions quickly, NVPSO converges to a solution that is closer to the known minimum of the g04 benchmark function. MGPSO produced feasible solutions for all 20 runs on the g04 benchmark functions, from which it can be concluded that MGPSO performs as reliably as NVPSO on the g04 benchmark function. From Tables II and III, it is evident that NVPSO produces solutions that are closer to the known minimum in the best, worst, and average case. Additionally, NVPSO produces a better solution in the worst case than the solution found by MGPSO in the best case, which indicates that NVPSO solves the g04

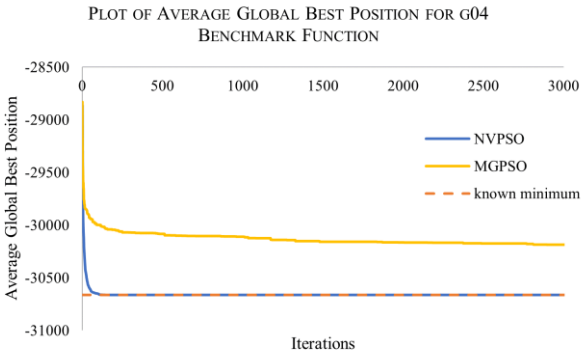

Fig. 3. Plot of the average best solution found across 20 independent runs of MGPSO and NVPSO on the g04 benchmark function.

problem more effectively than MGPSO. The large standard deviation between results found using MGPSO indicates that MGPSO does not perform consistently on the g04 benchmark problem. NVPSO however, has a small standard deviation between solutions, suggesting that it performs very consistently on the NVPSO problem. Therefore, although both algorithms perform with a similar efficiency and reliability, NVPSO solves the g04 COP more effectively and performs more consistently than MGPSO.

Fig. 4. contains a plot of the average solutions found over time by MGPSO and NVPSO for the g07 benchmark function. MGPSO produced feasible solutions during every run, making it equally as reliable as NVPSO for the g07 problem. However, Fig. 4. Shows that the first feasible solutions were only identified after 24 iterations of MGPSO. The plot shows that NVPSO converges on a solution in fewer iterations that MGPSO, making it more efficient for the g07 problem. Additionally, NVPSO converges on solutions that are closer to the known minimum for the problem, which indicates that NVPSO is more effective than MGPSO for the g07 problem. From Tables II and III, it is evident that NVPSO produces much better solutions than MGPSO in the best, worst and average case. Furthermore, the best solution found using MGPSO is still worse than the worst solution found using NVPSO. Lastly, the large standard deviation between MGPSO solutions indicates that MGPSO does not perform consistently on the g07 benchmark function, especially when compared to NVPSO, which has a very small standard deviation between results. Although both algorithms have the same reliability for the g07 problem, NVPSO performed more consistently, efficiently, and effectively.

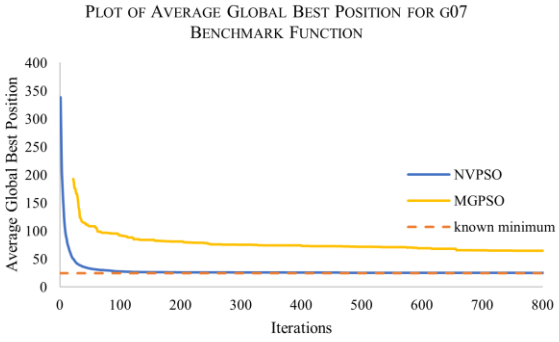

Fig. 4. Plot of the average best solution found across 20 independent runs of MGPSO and NVPSO on the g07 benchmark function.

Fig. 5. shows the average global best position over time for MGPSO and NVPSO on the g08 COP. The figure shows that NVPSO converges on a solution faster than MGPSO, and therefore solves the g08 problem more efficiently. Furthermore, NVPSO ultimately converges on a solution that is closer to the known minimum for the problem than the solution found by MGPSO, suggesting that NVPSO solves the gg08 problem more effectively than MGPSO. This is confirmed by the results contained in Tables II and III, which show that NVPSO performs better than MGPSO in the best, worst, and average cases. While the standard deviations across solutions are small for both algorithms, the standard deviation of solutions found by NVPSO is smaller than that of MGPSO, indicating that NVPSO performs more consistently on the g08 problem. Note that MGPSO produced feasible solutions for 19 of the 20 independent runs, giving it a reliability rate of 95% on the g08 problem, which is less than the 100% reliability of NVPSO. Therefore, although both variations of PSO produce solutions that are close to the known minimum, with small standard deviations, NVPSO is still more reliable, effective, and efficient that MGPSO for the g08 benchmark function.

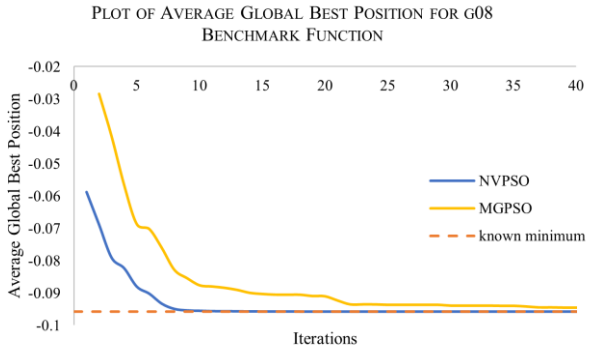

Fig. 5. Plot of the average best solution found across 20 independent runs of MGPSO and NVPSO on the g08 benchmark function.

Fig. 6. shows a plot of the average best position found over 20 iterations of MGPSO and NVPSO for the g09 COP. Fig. 6. indicates that NVPSO performs more efficiently than MGPSO on the g09 problem because it converged to a solution in fewer iterations. Additionally, the average solution found by NVPSO is closer to the known minimum for the problem, from which it can be concluded that NVPSO performs more effectively on the g09 problem. This is confirmed by Tables II and III, which show that NVPSO finds solutions very close to the known minimum in the best, worst, and average case and that there is a small standard deviation between solutions. The solutions found by MGPSO in the best, worst, and average cases are worse than those found by NVPSO and the standard deviation between solutions is notably larger. This indicates that the performance of NVPSO on the g09 benchmark function is more consistent, efficient, and effective than the performance of MGPSO. NVPSO and MGPSO are equally reliable for the g09 benchmark problem, as they both produced reliable solutions for all 20 independent runs of the algorithm.

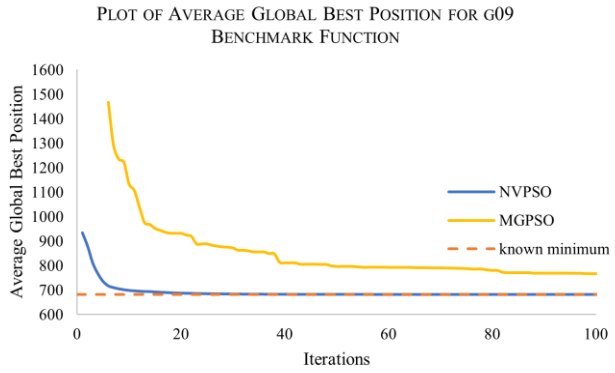

Fig. 6. Plot of the average best solution found across 20 independent runs of MGPSO and NVPSO on the g09 benchmark function.

Fig. 7. shows the standard deviation between the best solutions identified over time by the MGPSO and NVPSO algorithms respectively. The first plot suggests that MGPSO quickly converges to very similar solutions on the g01 and g08 benchmark functions. This is confirmed by the relatively small standard deviations for these problems contained in Table II. The standard deviations for the g07 and g09 problems converge after approximately 1500 iterations, indicating a very large variation in solutions during the early iterations of the algorithm. Furthermore, the large final standard deviations contained in Table II suggest that MGPSO produces solutions with a large variance after 5000 iterations of the algorithm, and therefore does not perform consistently on these problems. For all 5000 iterations of the algorithm, MGPSO produced solutions for the g04 problem with a very large standard deviation. This indicates that MGPSO converges to very different solutions for the g04 problem over independent runs and does not perform consistently on the g04 problem.

The second plot shows the standard deviation of solutions over time when the problem suite is solved using NVPSO. For all problems, the standard deviation converges to a very small value within 130 iterations. This indicates that NVPSO produces very similar solutions for each problem, and therefore, performs consistently on the suite of benchmark functions. Additionally, the fast convergence rate shows that NVPSO is efficient in solving this suite of problems. Like MGPSO, the second plot indicates that NVPSO performs best on the g01 and g08 benchmark functions, converging to a solution almost immediately. Similarly, NVPSO performed worse on the g07 and g09 benchmark functions, taking longer to produce very similar solutions than for the g01 and g09 problems. NVPSO performed worst on the g04 benchmark

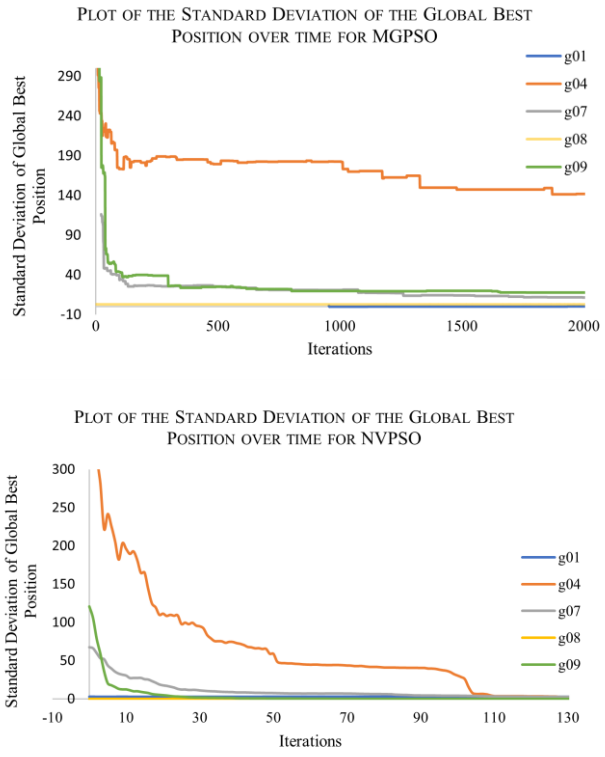

Fig. 7. Plots of the standard deviations across solutions found over time by MGPSO and NVPSO respectively.

function, like MGPSO. However, note that NVPSO still converged on solutions with a small standard deviation within 150 iterations, whereas MGPSO never converged to a small standard deviation for the g04 problem. This indicates that despite performing worst on the g04 problem, NVPSO still produced a significantly more consistent and efficient solution than MGPSO.

The results indicate that the reliability of MGPSO depends on the COP, while NVPSO is consistently reliable. For all the functions contained in the problem suite, NVPSO performed more consistently and produced better solutions, making it the more effective algorithm. The efficiency of the algorithm was found to depend on the problem. Both algorithms performed with similar efficiencies on the g01 problem. However, NVPSO performed more efficiently on the remaining problems.

VI. Conclusion

Although multi-guide particle swarm optimisation (MGPSO) performs reliably for three out of the five benchmark problems, it still produces solutions that are consistently worse than those produced by new vector particle swarm optimisation (NVPSO). In addition to this, MGPSO produces results with a large standard deviation for all benchmark functions, indicating that MGPSO does not consistently converge to the same solution. NVPSO, however, produces solutions with a small standard deviation across independent runs, verifying that the algorithm consistently produces similar results. Furthermore, NVPSO produces better results on average, and in the best and worst cases for all the benchmark functions, making it the more effective algorithm. Lastly, the plots of average best solutions over time indicate that NVPSO converges to a solution more quickly than MGPSO, making it a more efficient algorithm for solving constrained optimisation problems. Therefore, it can be concluded that new vector particle swarm optimisation solves constrained optimisation problems more accurately, reliably, efficiently, and consistently than multi-guide particle swarm optimisation.

References

[1] A. P. Engelbrecht, “Particle Swarm Optimisation,” in *Computational Intelligence. An Introduction.*, Pretoria, John Wiley & Sons, Ltd, 2007, pp. 289 - 341.

[2] C. Scheepers, A. P. Engelbrecht and C. W. Cleghorn, “Multi-Guide Particle Swarm Optimisation for Multi-Objective Optimisation: Empirical and Stability Analysis,” *Swarm Intelligence*, 2018.

[3] C.-l. Sun, J.-c. Zeng and J.-s. Pan, “A New Vector Particle Swarm Optimisation for Constrained Optimisation Problems,” *2009 International Joint Conference on Computational Sciences and Optimization*, pp. 485-488, 2009.

[4] J. Kennedy and R. Eberhart, “Particle Swarm Optimisation,” *Proceedings of ICNN'95 - International Conference on Neural Networks*, pp. 1942-1948, 1995.

[5] K. Deb, A. Pratap, S. Agarwal and T. Meyarivan, “A Fast and Elitism Multiobjective Generic Algorithm: NSGA-II,” *Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182-197, 2002.

[6] J. Liang, T. P. Runarsson, E. Mezura-Montes and P. N. Suganthan, “Problem Definitions and Evaluation Criteria for the CEC 2006 Special Session on Constrained Real-Parameter Optimisation,” *Nangyang Technological University, Singapore, Tech. Rep. 41*, 2006.

[7] K. Deb, “Multiobjective Genetic Algorithms: Problem Difficulties and Construction of Test Problems,” *Evolutionary Computation*, vol. 7, no. 3, pp. 205-230, 1999.