Swarm Intelligence (2024) 18:31–78
https://doi.org/10.1007/s11721-023-00232-5

Belief space-guided approach to self-adaptive particle swarm optimization

Daniel von Eschwege[1] · **Andries Engelbrecht**[1,2,3]

Received: 3 April 2023 / Accepted: 10 December 2023 / Published online: 31 January 2024

Abstract
Particle swarm optimization (PSO) performance is sensitive to the control parameter values used, but tuning of control parameters for the problem at hand is computationally expensive. Self-adaptive particle swarm optimization (SAPSO) algorithms attempt to adjust control parameters during the optimization process, ideally without introducing additional control parameters to which the performance is sensitive. This paper proposes a belief space (BS) approach, borrowed from cultural algorithms (CAs), towards development of a SAPSO. The resulting BS-SAPSO utilizes a belief space to direct the search for optimal control parameter values by excluding non-promising configurations from the control parameter space. The resulting BS-SAPSO achieves an improvement in performance of 3–55% above the various baselines, based on the solution quality of the objective function values achieved on the functions tested.

Keywords Self-adaptive · Particle swarm optimization · Belief space

1 Introduction

Particle swarm optimization (PSO) is an optimization algorithm modelled after the behaviour of birds in a flock (Kennedy and Eberhart, 1995) and belongs to the field of swarm intelligence (SI). PSO searches for a candidate solution by iteratively updating the positions of particles in a swarm. Position updates are informed by the best position a given particle has found, as well as by the best position found by the neighbourhood particles. PSO performance is greatly contingent on the selection of appropriate control parameter (CP) values which govern the search behaviour (Beielstein et al., 2002; Van den Bergh and Engelbrecht, 2006; Bonyadi

✉ Andries Engelbrecht
engel@sun.ac.za

Daniel von Eschwege
dvoneschwege@gmail.com

1 Department of Industrial Engineering, Stellenbosch University, Stellenbosch, South Africa

2 Computer Science Division, Stellenbosch University, Stellenbosch, South Africa

3 Center for Applied Mathematics and Bioinformatics, Gulf University for Science and Technology, Mubarak Al-Abdullah, Kuwait

and Michalewicz, 2016; Bratton and Kennedy, 2007). Control parameter configurations are usually selected as constant values, which is often not ideal for a specific problem (Van den Bergh and Engelbrecht, 2006; Jiang et al., 2007), because different objective function landscapes may benefit from varying degrees of exploration versus exploitation. However, tuning of control parameters for the problem at hand is computationally expensive and inefficient.

Alternatively, SAPSO algorithms have been proposed and attempt to adjust control parameters during the optimization process. Recent studies (Harrison et al., 2018a, 2016) have, however, shown that most SAPSO approaches introduce more parameters to which PSO performance is sensitive. These approaches also result in divergent behaviour, infeasible solutions, and small particle step sizes, and are generally ineffective at attaining better solutions. The self-adaptive process is further complicated by the fact that the previously optimal control parameter configuration may no longer be useful at the current moment (Harrison et al., 2018b).

The contribution of this study lies in the proposition of a SAPSO algorithm, which uses a cultural algorithm's belief space to adjust the PSO control parameters during the search, reducing the number of parameters to which PSO performance is sensitive, and improving the performance of the algorithm. The proposed BS-SAPSO performs 3% to 55% better than the various baselines in terms of objective function value solution quality, and (depending on the implementation) has two or one, instead of three runtime parameters which have to be adjusted.

Section 2 elaborates on PSO and BS, and Sect. 3 explains the design decisions pertaining to the BS-SAPSO. Section 4 explains the experimental procedure followed and the evaluation metrics used, and Sect. 5 relays the results obtained. Section 6 concludes the paper.

2 Background

This section elaborates on PSO itself, the effect of control parameter configurations, existing attempts at designing a SAPSO algorithm, metrics used to evaluate such algorithms, and the functioning of the belief space.

2.1 Particle swarm optimization

PSO makes use of population-based, stochastic search to find candidate solutions to an optimization problem by iteratively updating the positions of particles in a swarm. In addition to having a certain inertia, particles in the swarm update their positions in accordance with both the best positions they have personally found, and the best positions found by the particle neighbourhood. The latter are referred to, respectively, as the cognitive and social components of the velocity update rule (Shi and Eberhart, 1998; Kennedy and Eberhart, 1995):

$$v_{ij}(t+1) = \omega v_{ij}(t) + c_1 r_{1ij}(t)\left[y_{ij}(t) - x_{ij}(t)\right] + c_2 r_{2ij}(t)\left[\hat{y}_{ij}(t) - x_{ij}(t)\right] \tag{1}$$

where for particle i in dimension j at time t, $v_{ij}(t)$ is its velocity, $x_{ij}(t)$ is its position, $y_{ij}(t)$ is its personal best position, and $\hat{y}_{ij}(t)$ its neighbourhood best position. The inertia coefficient is denoted ω, and the cognitive and social coefficients are c_1 and c_2, respectively.

Stochasticity is introduced via the random constants r_{1j} and r_{2j}, sampled from a uniform distribution over (0,1). Position updates are governed by

$$\mathbf{x}_i(t+1) = \mathbf{x}_i(t) + \mathbf{v}_i(t+1) \tag{2}$$

2.2 Control parameter configurations

In order to optimally traverse the search space of the problem at hand, search trajectories should not diverge and also not repeat already explored paths cyclically. Furthermore, any optimization algorithm must manage a trade-off between exploration and exploitation; that is, searching as of yet unexplored areas, versus searching more thoroughly in locations which are already known to potentially yield good solutions. This trade-off is governed by three control parameters, namely the inertia weight, ω, the cognitive acceleration coefficient, c_1, and the social acceleration coefficient, c_2 (Engelbrecht, 2007). Constant control parameter values are often used, e.g. by selecting the generic values of $\omega = 0.729844$, $c_1 = 1.496180$ and $c_2 = 1.496180$ (Harrison et al., 2018a).

However, since more exploration is initially desirable, with exploitative behaviour towards the end of the search, one school of thought holds that PSO should find more optimal solutions if the control parameters can be adapted in a way which reflects this shift from exploration towards exploitation. An example approach would be to choose control parameters as follows [adapted from Sermpinis et al. (2013) to conform to $c_1 + c_2 > 4$ (Shi and Eberhart, 1998)]:

$$\begin{aligned}
\omega(t) &= 0.4\left(\frac{t-n_t}{n_t}\right)^2 + 0.4 \\
c_1(t) &= -3\frac{t}{n_t} + 3.5 \\
c_2(t) &= +3\frac{t}{n_t} + 0.5
\end{aligned} \tag{3}$$

where t is the current time step and n_t is the maximum number of timesteps. Equation (3) ensures a large c_1 and small c_2 initially, after which c_1 is decreased and c_2 increased. Since c_1 promotes exploration by increasing the personal component, whereas c_2 increases exploitation by increasing the social component, particles initially explore a lot and finally exploit more. The inertia weight ω is also decreased to shift from exploration to exploitation, so that particles initially take large steps to decrease the likelihood of becoming stuck in local minima and explore more of the search landscape, but eventually take small steps to search more thoroughly in the area which has been found to produce better solutions. A variant of Eq. (3) is the time-variant acceleration coefficients PSO (PSO-TVAC) (Ratnaweera et al., 2004), which is similar with exception of ω which is kept constant. Time-variant SAPSO algorithms therefore consider the total number of time steps used, and attempt to adjust the control parameters accordingly (Harrison et al., 2018a). The premise of time-variant SAPSO is therefore that a sufficient balance between exploration and exploitation can be found provided the computational budget limit.

Conversely, true self-adaptive approaches adjust control parameters based on introspective information derived during the search process. True SAPSO therefore seeks to adapt more specifically to the problem at hand, and to where in the search landscape a particle finds itself (Hashemi and Meybodi, 2011; Tanweer et al., 2015; Zhan et al., 2009; Jun and Jian, 2009). Adaptation of control parameter configurations is usually achieved by introduction of a governing equation which modifies control parameter configurations, and thus particle movement. The governing equation is mostly based on a behavioural tendency present in the swarm which is believed to contain information that can be exploited to improve performance (Harrison et al., 2018a). The premise of true SAPSO is therefore that the behaviour of particles contains information useful towards improving algorithm performance.

2.3 Convergence condition

A PSO particle is considered stable if it has convergent control parameters as specified by a derived stability condition (Poli, 2009; Poli and Broomhead, 2007). If the criterion,

$$c_1 + c_2 < \frac{24\left(1 - w^2\right)}{7 - 5w} \quad \text{and} \quad w \in [-1, 1] \tag{4}$$

holds for all particles' control parameters, the swarm is guaranteed to reach an equilibrium state. Note, however, that the criterion does not place a bound on the number of iterations required to reach this state, and also does not specify that velocities will not potentially assume very large values during the path to equilibrium.

2.4 Velocity clamping

Velocity clamping aims to prevent the explosion of particle velocities, which occurs when particles have sufficient inertia that their velocities grow without bound. Velocity explosion results in particles leaving the search space, often permanently remaining outside of the search boundaries (Oldewage et al., 2017). For a search space bounded by $[\boldsymbol{\ell}, \mathbf{u}]$, the velocity can be constrained in each dimension j to a fixed value as follows:

$$v_{i,j}^{t+1} = \begin{cases} v_{i,j}^{t+1} & \text{if } -v_{\max,j} \leq v_{i,j}^{t+1} \leq v_{\max,j} \\ v_{\max,j} & \text{if } v_{\max,j} < v_{i,j}^{t+1} \\ -v_{\max,j} & \text{if } v_{i,j}^{t+1} < -v_{\max,j} \end{cases} \tag{5}$$

where

$$v_{\max,j} = \delta\left(u_j - \ell_j\right), \quad \delta \in (0, 1) \tag{6}$$

Another velocity clamping method is to limit the magnitude of the velocity vector, i.e.

$$\mathbf{v}_i^{t+1} = \begin{cases} \mathbf{v}_i^{t+1} & \text{if } \left\|\mathbf{v}_i^{t+1}\right\| <= v_{\max} \\ \frac{v_{\max}}{\|\mathbf{v}_i^{t+1}\|}\mathbf{v}_i^{t+1} & \text{if } \left\|\mathbf{v}_i^{t+1}\right\| > v_{\max} \end{cases} \tag{7}$$

where

$$\begin{aligned} v_{\max} &= \delta \sqrt{\sum_{j=1}^{n} \left(u_j - \ell_j\right)^2} \\ &= \delta |\mathbf{u} - \boldsymbol{\ell}| \end{aligned} \tag{8}$$

with n-dimensional vectors $\mathbf{u} = \left[u_1, \ldots, u_n\right]^T$ and $\boldsymbol{\ell} = \left[\ell_1, \ldots, \ell_n\right]^T$.

Clamping velocities in all dimensions results in line search, whereas clamping per dimension has the disadvantage that it modifies the direction in which a particle is travelling. However, the random components $\mathbf{r}_1$ and $\mathbf{r}_2$ of the PSO change the particle's trajectory regardless of clamping, and therefore the additional modification of direction is not considered a problem.

Conversely, clamping by magnitude preserves the particle direction, but if the largest and smallest velocity components are on very different scales of magnitude, scaling all dimensions by some constant factor results in the smallest components becoming irrelevant.

2.5 Self-adaptive particle swarm optimization

Despite multiple SAPSO algorithms being published, many result in particles diverging towards infeasible search space, or premature convergence as a consequence of step sizes rapidly tending towards zero (Harrison et al., 2016). Most of the algorithms also introduce more control parameters than were present initially. These findings are corroborated by another paper (Harrison et al., 2018a), in which many of the 18 SAPSO algorithms analysed were found to demonstrate either divergence or premature convergence. Furthermore, in the attempt to suitably adjust the control parameters, ω, c_1 and c_2, merely three manage to decrease the number of parameters to which performance is sensitive, namely SAPSO by Li, Fuand and Zhang (SAPSO-LFZ) (Li et al., 2008), self-adaptive inertia weight PSO (SA-IWPSO) (Dong et al., 2008), and PSO with random acceleration coefficients (PSO-RAC) (Harrison et al., 2018a).

Note also that the SAPSO term is sometimes also used to refer to hyperheuristic adaptation strategies. In this case, instead of control parameter adaptation, a pool of candidate PSO variants is maintained, and the best performing variant is selected at each time step. Examples of such approaches are the heterogeneous PSO algorithms by Engelbrecht (2010) and Nepomuceno and Engelbrecht (2013), as well as the *Self-adaptive Particle Swarm Optimization-based Echo State Network for Time Series Prediction* approach by Xue et al. (2021). Such approaches are not considered in this paper, because hyperheuristic adaptation is fundamentally different to adjustment of the control parameters of an optimization algorithm.

2.6 Cultural algorithms

Cultural algorithms (CA) are evolution-inspired algorithms which maintain a belief space in parallel with a population space in order to engender dual inheritance of beneficial traits and information (Reynolds, 1994). The population space represents individuals, each with a set of behavioural traits, and the belief space represents beliefs which generalize on individual experiences. The population space may consist of a set of candidate solutions to the optimization problem, while the belief space maintains a set of beliefs about where in the search landscape the optimum resides. Any population-based metaheuristic can be used in the population space to find an optimal solution to the relevant optimization problem, with genetic algorithms (GAs) (Chahar et al., 2021) being generally used due to the analogy to nature with dual genetic and cultural inheritance.

As shown in Fig. 1, at each time step individuals in the population space are evaluated according to a fitness function, and the best individuals' beliefs are accepted to the belief space. The belief space is then used to influence the behaviour of individuals in the population space, thereby affecting the behaviour of particles in the population as it exists at later time steps. Since PSO particles' behaviour is determined by their control parameters, the belief space can steer the particles by influencing the control parameters. CAs therefore simulate the exchange of ideas between individuals, generally through mechanisms such as imitation, adaptation, or recombination. By allowing entities to learn from one another, CAs aim to leverage the collective intelligence of the population to find more optimal solutions.

CAs have seen real-world application in the fields such as civil engineering, mechanical engineering, electrical engineering, and computer science, predominantly, however, in the latter two (Maheri et al., 2021). Specific examples of problems include the optimization of memory usage and improvement of computational efficiency, as well as fault detection (Pan et al., 2010). The CA has also seen use in the structural optimization, for example, of dome structures, which were optimized subject to various constraints, such as stress, displacement, and frequency (Jalili et al., 2019).

3 Design of belief space-guided self-adaptative mechanism

The purpose of this section is to relay the design of the BS-SAPSO algorithm. The section starts by describing the general architecture and interaction between the PSO and the BS, and then elaborates on specific design choices regarding selection, sampling, and updating of the belief and population spaces.

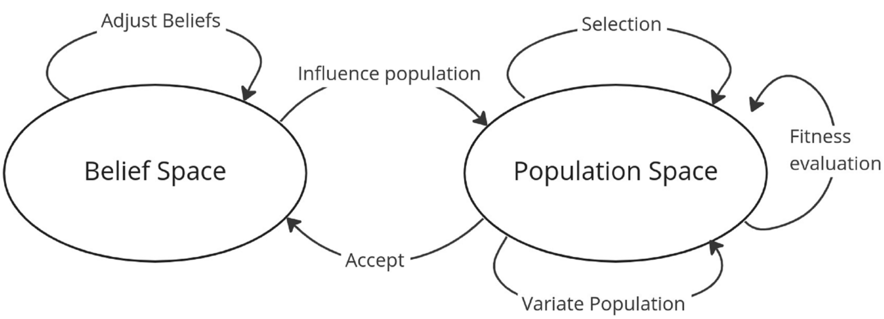


Fig. 1 Cultural algorithm (Jalili and Hosseinzadeh, 2014)

3.1 Architecture

In order to automatically tune the PSO control parameters, the PSO is augmented using the belief space concept, borrowed from CAs. The belief space represents the control parameter configurations believed to be best by the particles in the population space. Throughout the PSO search, certain particles affect the belief space, and in turn the belief space influences the control parameters of all particles in the population. An overview of the BS-SAPSO algorithm is given in Algorithm 1.

```
procedure BS-SAPSO(function, swarmSize, iterations)
    swarm, solutions ← InitializePSO(function, swarmSize)
    limitsBS ← w : [0, 1], c_1 : [0 : 4], c_2 : [0 : 4]
    BS ← InitializeBS(limitsBS)
    for i ← 1 to iterations do
        for j ← 1 to swarmSize do
            if UpdateCPTrigger(i) then
                CP ← UpdateCP(BS)
            end if
            particlePos ← PSOUpdateRule()
            swarm ← swarm + [particlePos]
            particleSol ← Evaluate(particlePos, function)
            solutions ← solutions + [particleSol]
        end for
        bestParticles, bestSolutions ← SelectParticles(swarm, solutions)
        if UpdateBSTrigger(i) then
            BS ← UpdateBS(bestParticles)
        end if
    end for
    return bestSolutions
end procedure

procedure InitializePSO(function, functionLimits, swarmSize)
    swarm ← []
    solutions ← []
    for i ← 1 to swarmSize do
        particlePos ← GenerateRandomPos(functionLimits)
        particleSol ← Evaluate(particlePos, function)
        swarm ← swarm + [particlePos]
        solutions ← solutions + [particleSol]
    end for
    return swarm, solutions
end procedure

procedure InitializeBS(limitsBS)
    BS[max] ← Max(limitsBS)
    BS[min] ← Min(limitsBS)
    return BS
end procedure
```

For brevity, *swarm* henceforth refers to the collection of particle positions, *solutions* to the collection of objective function values found by the particles at a given time step, *particlePos* to the position of a specific particle, and *particleSol* to the solution of a specific particle. Furthermore, *function* denotes the objective function being optimized, *CP* the control parameters with a mapping to their particles, and *BS* the belief space. Given Algorithm 1, the functions which are not self-evident, are the subjects of investigation in this paper:

- *Selection methods*, used to select the particles that will have their beliefs accepted into the belief space, explained in Sect. 3.2.
- *Update and sampling triggers*, to determine when the belief space is updated and control parameters are sampled. Given that a belief space update only affects the search if the control parameters are updated as well, UPDATEBSTRIGGER is taken to be the same as UPDATECPTRIGGER, with a discussion in Sect. 3.3.
- *Update methods*, used to specify how the chosen particles' beliefs affect the belief space. Here, UPDATEBS is performed as elaborated on in Sect. 3.4.
- *Sampling methods*, to determine how new control parameter values (beliefs) are sampled from the belief space to influence the population space, with UPDATECP explained in Sect. 3.5.

Ultimately, all of the abovementioned aspects influence the trade-off between exploration and exploitation, and therefore the performance of the algorithm. It is expected that if the belief space converges too quickly to a certain control parameter configuration, the algorithm will become too exploitative, and if the belief space does not converge quickly enough, the algorithm will become too explorative.

3.2 Selection methods

Selection methods determine which particles are allowed to influence the belief space, and the following selection methods are proposed:

- *Random selection*, where all particles have an equal probability of updating the belief space. While random selection is not expected to perform well, it serves to set a BS-SAPSO performance baseline.
- *Elitist selection*, where only the n_e particles with the best objective function values are allowed to influence the belief space, with n_e reduced by one on every update. Elitist selection postulates that since the particles have found better solutions, their control parameters are more likely to be correct for the function at hand.
- *Roulette wheel selection*, where particles are selected based on a probability proportional to their objective function values. While similar to elitist selection, the use of probabilities prevents the strict exclusion of worse-performing particles, but merely reduces the probability of their selection.
- *Rank selection*, where particles are ranked according to objective function value, and selected with a probability proportional to their ranking. This selection method is similar to roulette wheel selection, but prevents giving too much weight to the best particles in cases where the best solution quality is orders of magnitude larger than the worst.
- *Improvement selection*, where only the n_e particles which demonstrated the most improvement in objective function values are allowed to influence the belief space,

with n_e reduced by one on every update. While similar to elitist selection, the difference lies therein that value is placed on the improvement engendered by a control parameter configuration, rather than the actual value, which might have resulted from lucky initialization.

- *Improvement-magnitude selection*, where the n_e particles which showed the most improvement in objective function values are allowed to influence the belief space with a selection probability proportional to the objective function value improvement the particle underwent. Similar to improvement selection, but with probabilities instead of hard classes, this method gives more weight to particles that show more improvement, without strictly excluding the other particles.
- *Tournament selection*, where n_e particles are selected at random, after which particles are selected with a probability proportional to their objective function values. Tournament selection increases the randomness of the process, while retaining a degree of elitism after the random selection.

Generally, it is expected that methods which incorporate more randomness will lead to more exploration of the control parameter space, while methods which are more deterministic will lead to more exploitation.

3.3 Update and sampling triggers

To maintain consistency of the effect between updating the belief space and sampling new control parameters, the same trigger is used for both UpdateBSTrigger and UpdateCPTrigger, and the following conditions that set the trigger are investigated:

- *Always:* setting the trigger always, i.e. on every time step, has the advantage of not introducing an additional control parameter, but will probably lead to rapid convergence of the belief space, because every consecutive belief space update necessarily reduces the range of acceptable control parameter values.
- *Time-variant:* the belief space can be updated according to a function of the current iteration, in an analogous fashion to how control parameters are updated by Eq. (3).
- *Fixed:* updating the belief space at fixed intervals p_f, instead of at every time step, is an option to prevent early convergence, but introduces an additional hyperparameter.
- *Stagnate:* the belief space can be updated when the best solution found by the swarm does not improve (i.e. stagnates) for p_s iterations, but this introduces an additional hyperparameter.

3.4 Update method

The belief space represents a range of acceptable control parameter values, and as such the limits of this range are updated based on the selected particles. Due to the large amount of variation introduced by the various selection methods in Sect. 3.2, UpdateBS is kept fixed by setting the belief space boundaries to the minimum and maximum control parameter values out of the group of selected particles.

3.5 Sampling method

In order to update control parameters based on the belief space, UPDATECP samples new control parameter values for each particle in the swarm from a uniform distribution between the belief space limits.

4 Experimental procedure

This section details the experimental procedure as pertaining to evaluation metrics and implementational details. Since many different facets of the algorithm are evaluated, a 'run' henceforth refers to a single execution of the BS-SAPSO on a single function, and an 'experiment' refers to running the BS-SAPSO algorithm for $r = 30$ runs on the whole function set, with a specific control parameter and algorithmic configuration.

4.1 Evaluation metrics

In order to analyse the performance and behaviour of the BS-SAPSO, a number of metrics are employed:

1. *Normalized global best solutions* The objective function values, albeit solutions, at the global best positions, are ultimately the indicator of how well an algorithm performs on a given optimization problem. Since functions often span vastly different magnitudes, normalized global best solutions are used. After all experiments are completed, the highest and lowest global best solutions which have been found throughout all experiments and all runs are used to normalize global best solutions to [0,1] for each experiment. Note that this does not affect the search, since the BS-SAPSO is executed on the function set as-is, after which the solutions are merely scaled to the same range to allow for calculating a score per experiment, by averaging over the global best solutions of the runs in that experiment. The scale to which is normalized is arbitrary, but unitary scaling results in easily interpretable results, where a lower value indicates a better solution.
2. *Average swarm diversity* provides information regarding the level of exploration and exploitation. Diversity is calculated using (Olorunda and Engelbrecht, 2008)

$$\mathcal{D} = \frac{1}{n_s} \sum_{i=1}^{n_s} \sqrt{\sum_{j=1}^{n_x} \left(x_{ij} - \bar{x}_j\right)^2} \tag{9}$$

   with the swarm centre at

$$x_j = \frac{\sum_{i=1}^{n_s} x_{ij}}{n_s} \tag{10}$$

   where n_s is the number of particles in the swarm, and is n_x the number of dimensions.
3. *Percentage of particles in infeasible space* If a particle violates the boundaries of feasible search space even in one dimension, it is considered as being in infeasible space.

Infeasible particles should not be considered when updating the best positions found as to not direct the search out of feasible space (Engelbrecht, 2013).
4. *Percentage of particles that are stable* A stable particle has convergent control parameter configurations, in accordance with the stability condition [refer to Sect. 2.3 in Eq. (4)].
5. *Average particle velocity* represents average step sizes, which have to decrease to achieve convergence, but should not tend towards zero too early in the search process, as to not get stuck in a local minimum. Average particle movement is calculated using (Harrison et al., 2018a)

$$\Delta(t+1) = \frac{1}{n_s} \sum_{i=1}^{n_s} \left\| \mathbf{x}_i(t+1) - \mathbf{x}_i(t) \right\| \tag{11}$$

4.2 Implementation

The PSO variant used is the inertia weight PSO (Shi and Eberhart, 1998) as given by Eqs. (1) and (2), and the neighbourhood of each particle is the whole swarm. The swarm is initialized with particles uniformly distributed within the feasible search space, and the belief space is initialized to the min-max ranges of $\omega \in [0, 1]$, $c_1 \in [0, 4]$ and $c_2 \in [0, 4]$. For each experiment (i.e. aspect which is investigated), $r = 30$ independent runs are performed over 31 minimization functions, which are given in Appendix C. The evaluation functions used to select the best algorithm can, however, not also represent function performance without bias, which is why a test set of 24 additional functions was introduced, discussed in Sect. 4.3, and listed in Appendix C.

Objective function value is only calculated for particles which reside within feasible space. Particles outside of feasible space have their objective values set to infinity, which, assuming minimization, automatically disqualifies those particles from updating the personal or global best known positions. A swarm size of $n_s = 30$ is used, together with a dimension space of $n_d = 30$ for each function, and $i_{max} = 5000$ time steps per run. *All plots and scores are the mean and standard deviation over all the functions and runs of a given experiment, thus* $(30 \times 45) + (30 \times 7) = 1560$ *runs per experiment.* Given the combination of algorithmic aspects investigated, a total of 98 experiments are performed and by implication 132,300 runs. The naming convention for experiments is (selection_method)(n_e)_(update_trigger)(p), in accordance with the explanations in Sect. 3.3. The performance of all experiments is compared in Table 9 in Appendix B according to the normalized global best solution quality achieved by that experiment, with explanation of the metrics in Sect. 4.1.

4.3 Benchmark function set

Lang and Engelbrecht (2021) proposed a benchmark set of 24 functions as *An Exploratory Landscape Analysis-Based Benchmark Suite*. This benchmark set was constructed following an intensive analysis of BBOB (Hansen et al., 2009) and CEC (Liang et al., 2013a, b, 2014; Wu et al., 2016) benchmarks. The analysis used self-organizing maps to cluster the functions according to their fitness landscape characteristics, and showed that many of the functions in the BBOB and CEC benchmarks did not differ significantly. The analysis also indicated gaps, where not all fitness landscape characteristics are sufficiently represented, with overemphasis on others (Cenikj et al., 2022; Lang and Engelbrecht, 2020a, b). Following the analysis, a more comprehensive benchmark set with wide coverage of fitness landscape characteristics was proposed, all the while containing many functions which are also used in the CEC, BBOB, and various other benchmark suites.

5 Results

This section explains how implementational details were varied across experiments, and presents the results obtained. Note that some plots contain gaps, for example, in Fig. 21, which appear when particles move so far out of the search space that the swarm diversity calculation in Eq. (9) results in numerical overflow.

5.1 Performance baselines

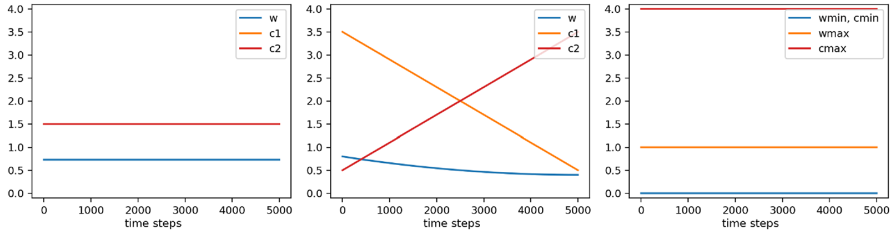
This section sets four performance baselines, where one is simply the CP-tuned inertia weight PSO, and the others are selected from the self-adaptive approaches which Harrison et al. (2018a) found to exhibit good search characteristics. Because this study surveyed the state of self-adaptive PSO, it sets a good starting point for comparison.

The inertia weight PSO baseline (PSO-IW) uses the constant control parameter configuration from Sect. 2.2, and the second (PSO-TVIW) uses the time variant configuration given by Eq. (3). The third is PSO-TVAC, explained in Sect. 2.2, and the fourth (PSO-RAC) samples random convergent control parameters per particle on every time step (Engelbrecht, 2022; Harrison et al., 2017), that is, the control parameter configurations conform to Poli's convergence criterion in Eq. (4). Table 1 gives the normalized global best solutions per baseline.

Figure 2 shows the CP values of the PSO baselines throughout the search. Figure 3 confirms that all particles are stable for the constant and random baselines, whereas for the time-variant baseline, initially all particles are unstable, which changes exactly halfway through the search, with the entire swarm then becoming stable. Figure 4 shows that for all baselines, initially almost all particles reside outside the feasible search space, but towards the end most are within feasible space. Figure 5 shows that the particle velocities of the constant baseline decrease smoothly, whereas particle velocities for the time-variant baseline explode initially, but then return to smaller values. The random baseline has a high average velocity, but also decreases smoothly. Figure 6 confirms the observations made in Fig. 5, as the swarm diversity is usually directly related to the average particle velocity.

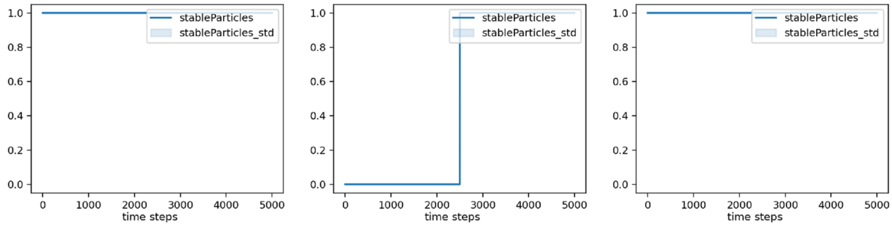
Table 1 Normalized global best solutions for performance baselines

	Evaluation	Testing
PSO-IW	0.1976	0.0808
PSO-RAC	0.2004	0.0910
PSO-TVAC	0.2543	0.1395
PSO-TVIW	0.2955	0.1756



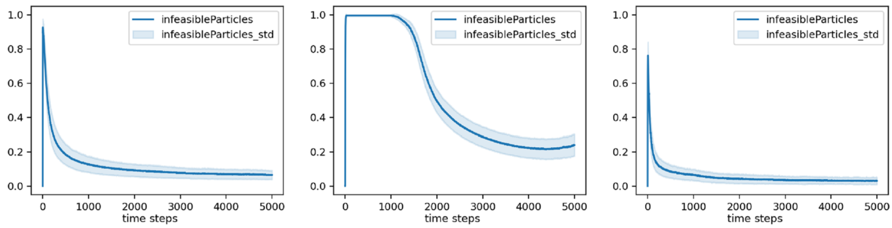
(a) Constant CP (b) Time-variant CP (c) Random CP

Fig. 2 Control parameters for baseline



(a) Constant CP (b) Time-variant CP (c) Random CP

Fig. 3 Stable particles for baseline

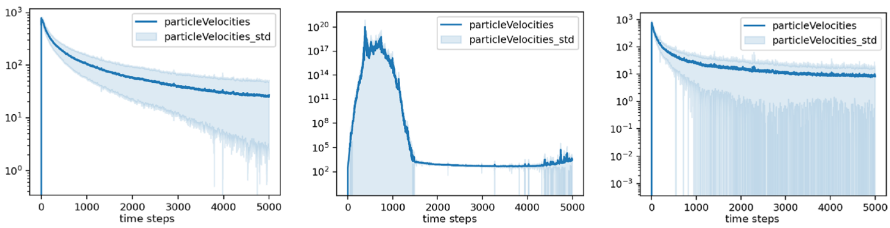


(a) Constant CP (b) Time-variant CP (c) Random CP

Fig. 4 Infeasible particles for baseline

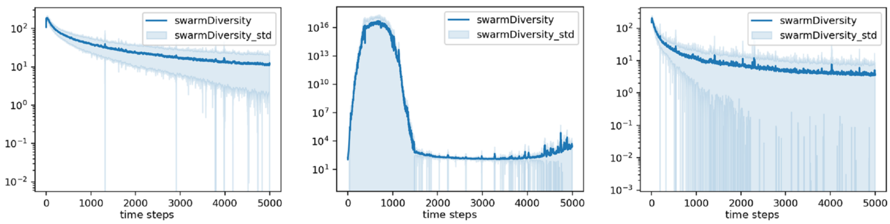
5.2 Belief space and control parameter updates on every time step

In this section, belief spaces are updated on every iteration and control parameters are sampled on every iteration. The most basic version of the BS-SAPSO algorithm (always_random) randomly selects particles to update the belief space on every time step. Thereafter, one aspect of the algorithm is changed at a time, in accordance with the explanations in Sect. 3.2. Table 2 gives the normalized global best solutions per experiment, and indicates that of the experiments which always update the belief space, elitist and improvement selection tend to perform better, followed closely by improvement-magnitude selection. Only a selected number of metrics are plotted, firstly due to space constraints, but also since most metrics within a section generally resemble each other across experiments. Since the values given are mean values over 45 functions with 30 runs each, the standard deviation is also given where applicable.



(a) Constant CP (b) Time-variant CP (c) Random CP

Fig. 5 Particle velocity vectors for baseline



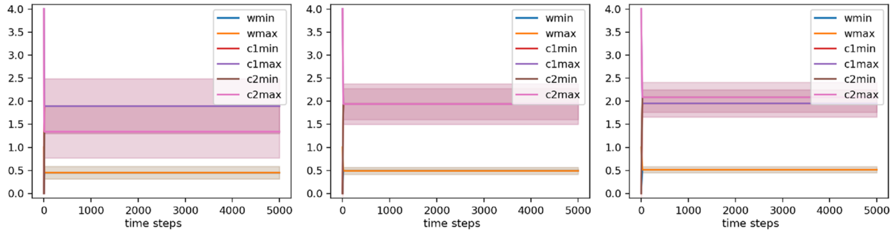
(a) Constant CP (b) Time-variant CP (c) Random CP

Fig. 6 Swarm diversity for baseline

Table 2 Normalized global best solutions for BS/CP updates on every time step

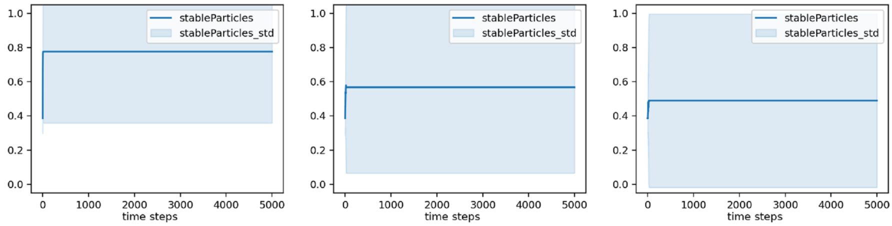
	Evaluation	Testing
always_elitist20	0.2764	0.1570
always_improve20	0.2801	0.1569
always_improve10	0.2813	0.1665
always_improve30	0.2838	0.1596
always_elitist10	0.2944	0.1770
always_improveMag30	0.2982	0.1799
always_elitist30	0.2984	0.1798
always_improveMag10	0.2989	0.1792
always_improveMag20	0.2989	0.1858
always_tournament10	0.3127	0.1890
always_tournament20	0.3150	0.1828
always_random	0.3209	0.2122
always_tournament30	0.3226	0.1858
always_roulette	0.3298	0.2088
always_rank	0.3305	0.2081

A tendency observed when plotting the belief space boundaries, consisting of $\{ (w_{min}, w_{max}), (c_{1min}, c_{1max}), (c_{2min}, c_{2max}) \}$, in Fig. 7, is that belief space convergence is almost instantaneous. This occurs because the belief space boundaries are updated on every time-step, which is too frequent to allow for exploration of the control parameter



(a) $n_e = 10$ (b) $n_e = 20$ (c) $n_e = 30$

Fig. 7 Belief space boundaries for elitist selection



(a) $n_e = 10$ (b) $n_e = 20$ (c) $n_e = 30$

Fig. 8 Stable particles for elitist selection

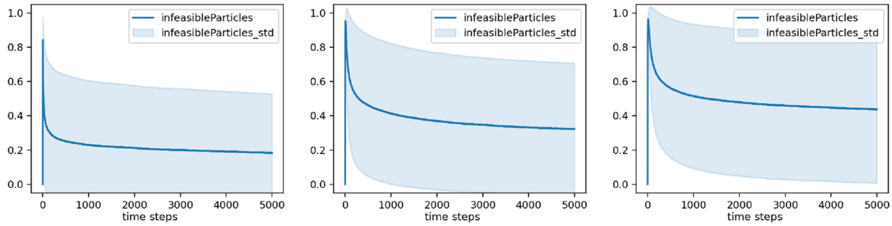
space. In Fig. 7, the minimum and maximum belief space boundaries therefore converge to the same value, and the plot shows a flat line.

Another observation is that the belief space boundaries tend to converge around the mean of the initial minimum and maximum belief space boundaries, implying that belief spaces assume statistically likely values rather than displaying self-adaptive behaviour. Belief space updates on every time step therefore do not seem to be a viable approach.

Generally, the number of stable particles decreases when more particles are allowed to influence the belief space (i.e. n_e is increased), as shown in Fig. 8, with the inverse being true for the number of infeasible particles in Fig. 9. The former is explained by larger belief space ranges being used for selection of new control parameter values if n_e is higher. Furthermore, for all experiments there are cases of exploding particle velocities, for example, as in Fig. 10, and thus exploding swarm diversity, as in Fig. 11, creating an unstable search process. All methods in Table 2 perform worse than the baselines in Table 1 and therefore need additional modification.

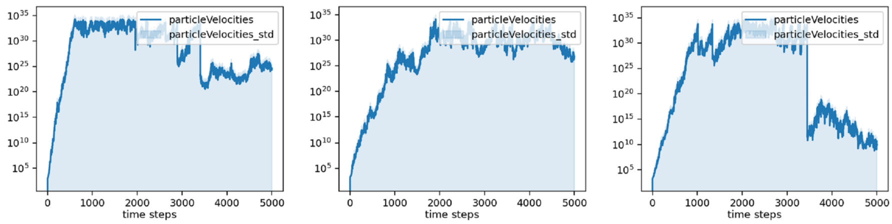
5.3 Belief space and control parameter updates at fixed intervals

All experiments in this section are based on the best performing experiments of Sect. 5.2 (that is, elitist and improvement selection), but use belief spaces which are updated at fixed intervals, with control parameters also sampled at the same fixed intervals.



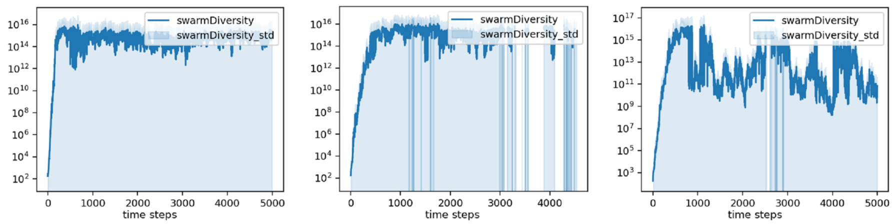
(a) $n_e = 10$ (b) $n_e = 20$ (c) $n_e = 30$

Fig. 9 Infeasible particles for elitist selection



(a) $n_e = 10$ (b) $n_e = 20$ (c) $n_e = 30$

Fig. 10 Particle velocity vectors for elitist selection



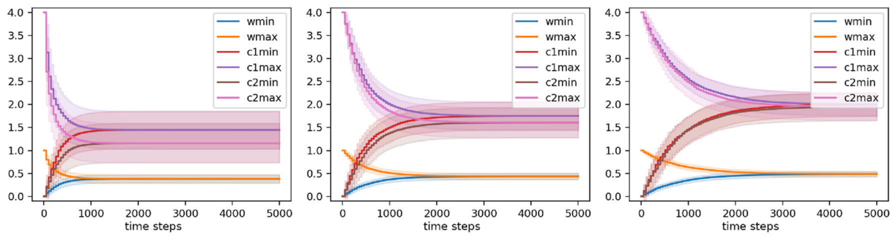
(a) $n_e = 10$ (b) $n_e = 20$ (c) $n_e = 30$

Fig. 11 Swarm diversity for elitist selection

Convergence is slower than before, as can be seen in the belief space boundaries in Fig. 12, and is achieved by fixing the belief space updates and control parameter sampling to longer intervals ($p_f = 20$ and $p_f = 50$). The delayed convergence improves performance, but increases the number of unstable particles, and introduces the additional parameter p_f. Furthermore, the improvement and elitist selection methods which initially use more particles for belief space updates (higher n_e) tend to perform better, as shown in Table 3. Finally, despite having a relatively stable swarm, demonstrated in Fig. 13, and low numbers of infeasible particles as shown in Fig. 14, particle velocities in Fig. 15 and swarm diversity in Fig. 16 explode similarly as in Sect. 5.2.

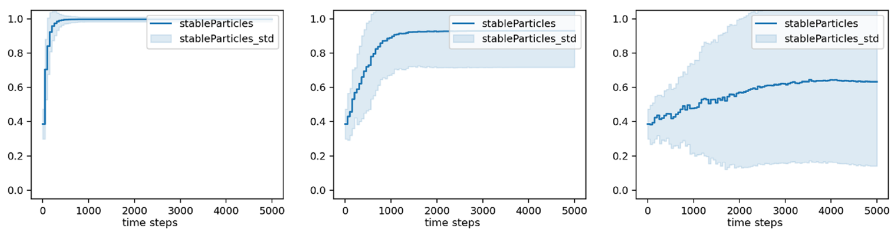
Table 3 Normalized global best solutions for BS/CP updates at fixed intervals

	Evaluation	Testing
fixed50_improve30	0.1922	0.0852
fixed50_elitist30	0.2077	0.0946
fixed20_improve30	0.2090	0.0882
fixed20_improve20	0.2112	0.0957
fixed20_elitist30	0.2169	0.0963
fixed10_improve30	0.2181	0.0939
fixed50_improve20	0.2186	0.0954
fixed20_elitist20	0.2231	0.1024
fixed10_elitist30	0.2232	0.1014
fixed10_improve20	0.2281	0.1049
fixed50_elitist20	0.2305	0.1031
fixed10_elitist20	0.2380	0.1150
fixed50_improve10	0.2600	0.1258
fixed50_elitist10	0.2604	0.1240
fixed20_improve10	0.2677	0.1387
fixed20_elitist10	0.2723	0.1427
fixed10_improve10	0.2743	0.1450
fixed10_elitist10	0.2787	0.1542



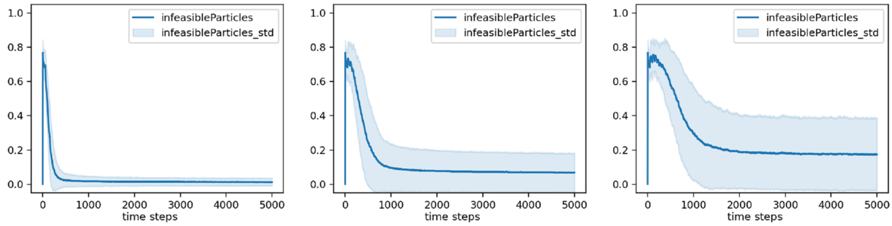
(a) $n_e = 10$ (b) $n_e = 20$ (c) $n_e = 30$

Fig. 12 Belief space boundaries for improvement selection at $p_f = 50$ time steps



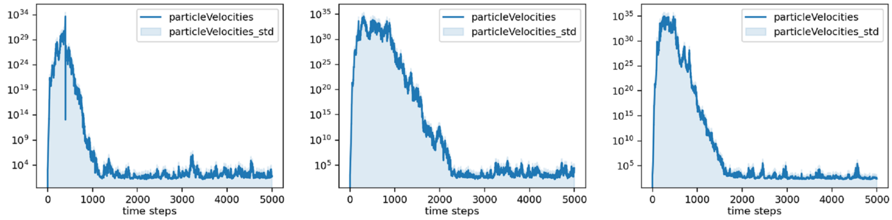
(a) $n_e = 10$ (b) $n_e = 20$ (c) $n_e = 30$

Fig. 13 Stable particles for improvement selection at $p_f = 50$ time steps



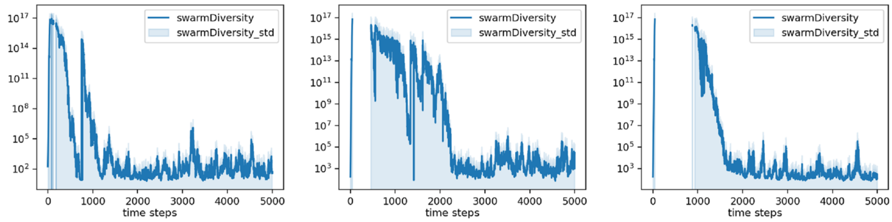
(a) $n_e = 10$ (b) $n_e = 20$ (c) $n_e = 30$

Fig. 14 Infeasible particles for improvement selection at $p_f = 50$ time steps



(a) $n_e = 10$ (b) $n_e = 20$ (c) $n_e = 30$

Fig. 15 Particle velocity vectors for improvement selection at $p_f = 50$ time steps



(a) $n_e = 10$ (b) $n_e = 20$ (c) $n_e = 30$

Fig. 16 Swarm diversity for improvement selection at $p_f = 50$ time steps

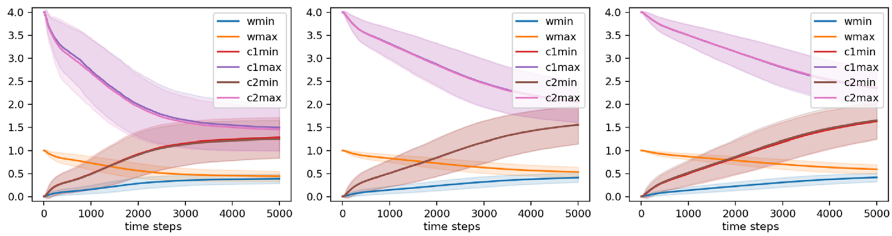
5.4 Belief space and control parameter updates upon stagnation of global best solution

All experiments in this section are similar to those in Sect. 5.3, but use belief spaces which are updated when the global best solution does not improve for a certain number of iterations, with control parameters sampled at the same time steps (Figs. 17, 18, 19, 20).

Delaying convergence by updating the belief space and control parameters when the objective function value stagnates for longer intervals ($p_s = 20$ and $p_s = 50$) improves performance compared to performance reported in Sect. 5.2. Furthermore, improvement and elitist selection methods which initially use more particles for belief space updates (higher n_e) tend to perform better. Velocity explosion is, however, especially bad here, and exceeds the plot limits regularly, as depicted in Fig. 21 (Table 4).

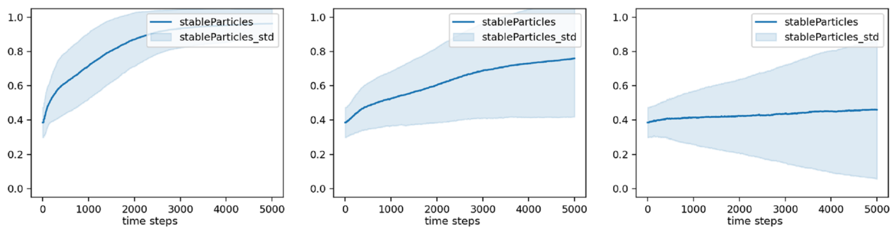
Table 4 Normalized global best solutions of for BS/CP updates on stagnation

	Evaluation	Testing
stagnate20_improve30	0.2157	0.0966
stagnate10_elitist30	0.2188	0.0969
stagnate10_improve30	0.2208	0.0960
stagnate10_improve20	0.2228	0.1009
stagnate20_improve20	0.2253	0.1012
stagnate20_elitist30	0.2255	0.1006
stagnate20_elitist20	0.2270	0.1052
stagnate50_improve20	0.2270	0.1033
stagnate10_elitist20	0.2287	0.1088
stagnate50_improve30	0.2297	0.1003
stagnate50_elitist30	0.2317	0.1047
stagnate50_elitist20	0.2320	0.1050
stagnate50_improve10	0.2455	0.1204
stagnate50_elitist10	0.2480	0.1198
stagnate20_improve10	0.2487	0.1321
stagnate10_improve10	0.2498	0.1344
stagnate10_elitist10	0.2501	0.1425
stagnate20_elitist10	0.2512	0.1343



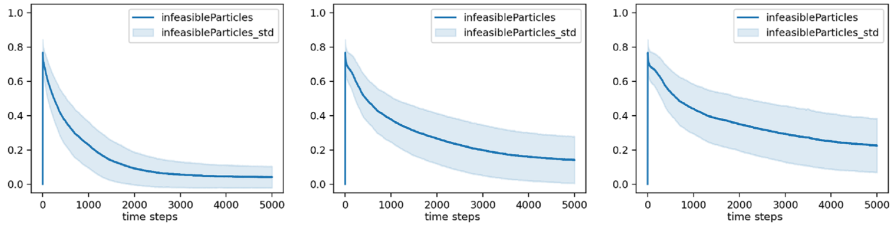
(a) $n_e = 10$ (b) $n_e = 20$ (c) $n_e = 30$

Fig. 17 Belief space boundaries for improvement selection at $p_s = 20$ time steps



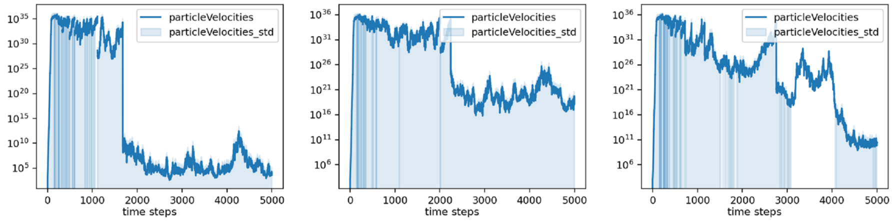
(a) $n_e = 10$ (b) $n_e = 20$ (c) $n_e = 30$

Fig. 18 Stable particles for improvement selection at $p_s = 20$ time steps



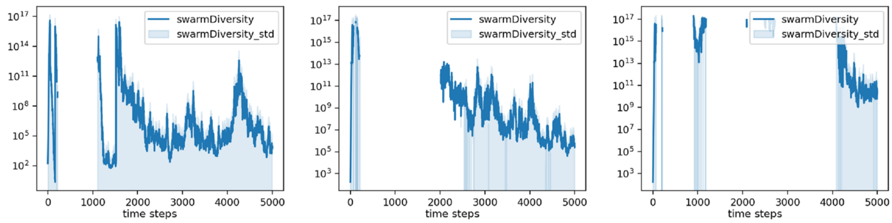
(a) $n_e = 10$ (b) $n_e = 20$ (c) $n_e = 30$

Fig. 19 Infeasible particles for improvement selection at $p_s = 20$ time steps



(a) $n_e = 10$ (b) $n_e = 20$ (c) $n_e = 30$

Fig. 20 Particle velocity vectors for improvement selection at $p_s = 20$ time steps



(a) $n_e = 10$ (b) $n_e = 20$ (c) $n_e = 30$

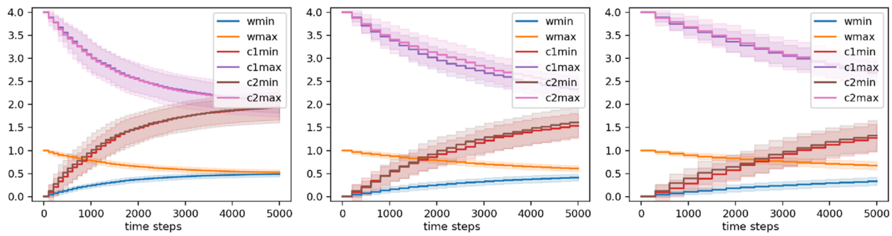
Fig. 21 Swarm diversity for improvement selection at $p_s = 20$ time steps

5.5 Belief space and control parameter updates at delayed intervals

Considering the results of Sects. 5.3 and 5.4, a clear tendency is that performance improves if larger time step intervals are used with high values of n_e. Therefore, this section repeats the experiments from these sections, but with even larger time step intervals, and with n_e set to the maximum of $n_e = n_s = 30$. Figure 22 show a slower belief space convergence, accompanied by less stable particles and more infeasible particles in Figs. 23 and 24. Particle velocities and swarm diversity still explodes, as shown in Figs. 25 and 26, respectively. The best performing methods in Table 5 outperform the baselines in Table 1, but only marginally.

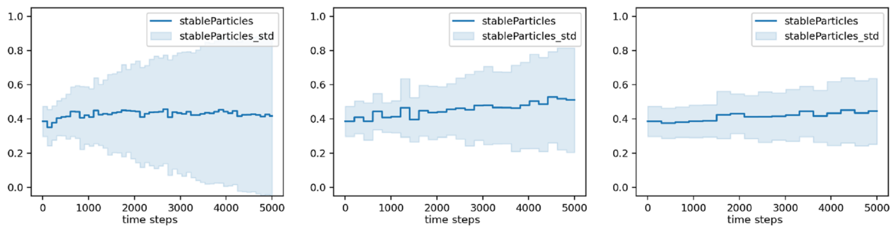
Table 5 Normalized global best solutions for BS/CP updates at delayed fixed intervals

	Evaluation	Testing
delayed_fixed100_improve30	0.1919	0.0900
delayed_fixed100_elitist30	0.2023	0.0959
delayed_fixed200_improve30	0.2050	0.0996
delayed_fixed200_elitist30	0.2064	0.0998
delayed_fixed300_elitist30	0.2195	0.1081
delayed_fixed300_improve30	0.2213	0.1084
delayed_stagnate100_improve30	0.2345	0.1098
delayed_stagnate100_elitist30	0.2398	0.1093
delayed_stagnate200_elitist30	0.2483	0.1168
delayed_stagnate200_improve30	0.2488	0.1173
delayed_stagnate300_improve30	0.2555	0.1208
delayed_stagnate300_elitist30	0.2560	0.1211



(a) $p = 100$ (b) $p = 200$ (c) $p = 300$

Fig. 22 Belief space boundaries for improvement selection for $n_e = 30$

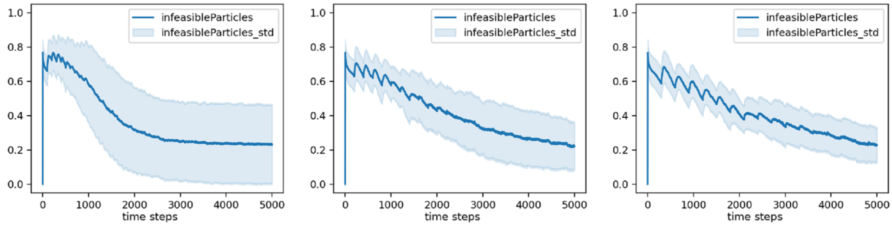


(a) $p = 100$ (b) $p = 200$ (c) $p = 300$

Fig. 23 Stable particles for improvement selection for $n_e = 30$

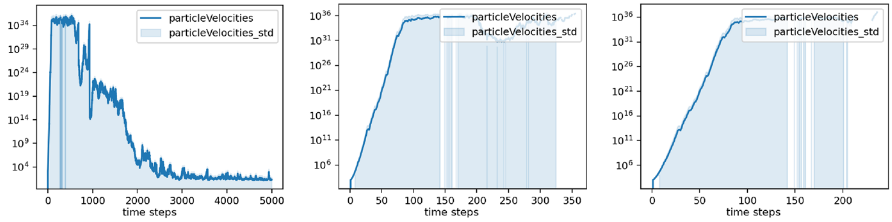
5.6 Stability-guided BS-SAPSO

The experiments in this section repeat the three best performing experiments from Sects. 5.2 to 5.4, with the additional condition that control parameters must adhere to Poli's convergence criterion [Eq. (4)], and that the belief spaces are prevented from assuming ranges which preclude sampling of convergent control parameters. The reason for enforcing Poli's criterion is to prevent the particles from permanently leaving the search space,



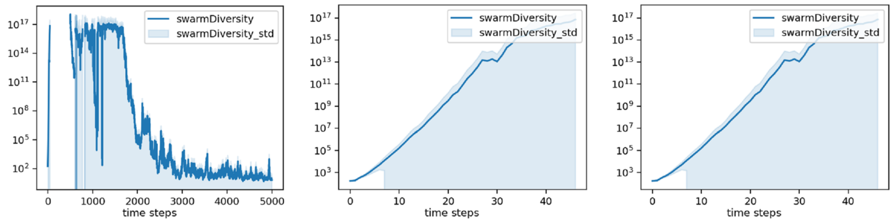
(a) $p = 100$ (b) $p = 200$ (c) $p = 300$

Fig. 24 Infeasible particles for improvement selection for $n_e = 30$



(a) $p = 100$ (b) $p = 200$ (c) $p = 300$

Fig. 25 Particle velocity vectors for improvement selection for $n_e = 30$



(a) $p = 100$ (b) $p = 200$ (c) $p = 300$

Fig. 26 Swarm diversity for improvement selection for $n_e = 30$

thereby resulting in a more thorough search of the feasible search space, which will hopefully lead to better solution quality.

Sampling control parameter values which adhere to Poli's convergence criterion, given by Eq. (4), is straightforward; control parameters which do not adhere to the criterion can simply be discarded and resampled, or else the sampled values for ω and c_1 can place boundaries on allowable values for c_2, for more computational efficiency.

However, the situation can arise where belief space boundaries assume values which do not permit the sampling of any convergent control parameters. This can happen because belief space boundaries are updated only with reference to a number of influential particles, without considering whether the combination of these new belief space boundaries allows for convergent control parameter values.

To prevent the aforementioned, limits need to be placed on the values which belief space boundaries can assume. Limiting the belief space is, however, not straightforward, due to

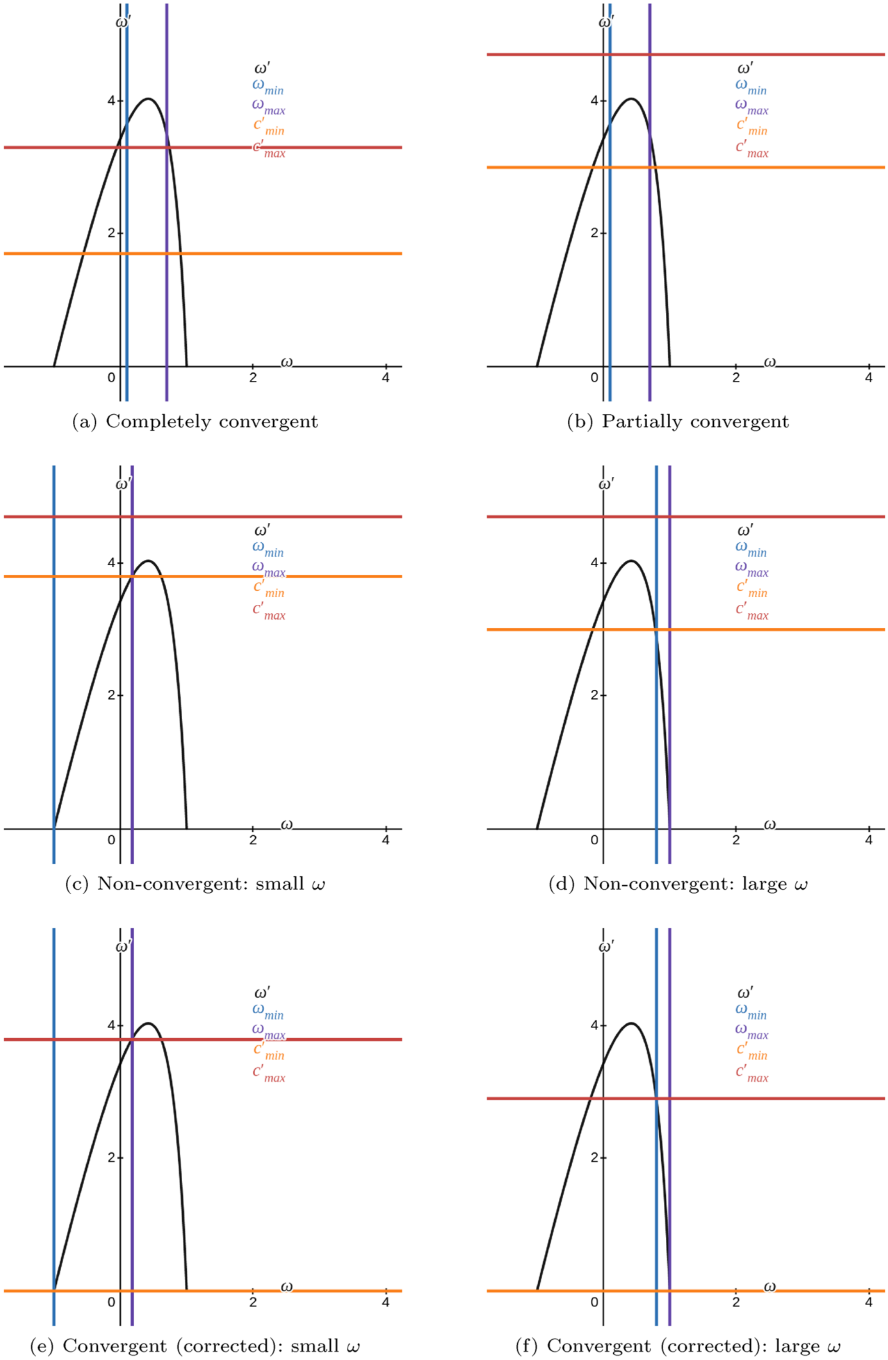


Fig. 27 Potential belief space configurations for Poli's convergence criterion

the parabolic part of Eq. (4). The possible cases which can arise are shown in Fig. 27, where

$$\omega' = \frac{24\left(1 - w^2\right)}{7 - 5w} \quad \text{for } w \in [-1, 1] \quad \text{and} \quad c' = c_1 + c_2$$

with ω the x-axis and ω' the y-axis. Also, c'_{min} (orange) and c'_{max} (red) represent the range c' can assume, based on the belief space boundaries c_1 and c_2. Furthermore, ω_{min} (blue) and ω_{max} (purple) represent the range ω (not ω') can assume—these are therefore the belief space boundaries for ω. The area enclosed by c'_{min}, c'_{max}, ω_{min} and ω_{max} demarcates the control parameter values which can potentially be sampled. To adhere to the convergence criterion, these control parameters must therefore fall underneath the curve ω' (black).

Figure 27a demonstrates the completely convergent case; the enclosed area completely falls under ω' —it is therefore impossible to sample non-convergent control parameters. Figure 27b shows overlap: both convergent and non-convergent configurations are possible, which is acceptable, because non-convergent configurations can simply be discarded. In Fig. 27c, the relatively high c' range, in conjunction with the low ω-range, disallows convergent control parameters. The same is true for Fig. 27d, except that here both the c' range and the ω range are too high.

Belief space updates which move belief space boundaries outside the ranges which allow for convergent parameters can be prevented by requiring that the updated belief space adheres to either

$$c_1 + c_2 < \frac{24\left(1 - w_{max}^2\right)}{7 - 5w_{max}} \quad \text{for } w_{max} \in [-1, 1] \tag{12}$$

to prevent the situation in Fig. 27c, or

$$c_1 + c_2 < \frac{24\left(1 - w_{min}^2\right)}{7 - 5w_{min}} \quad \text{for } w_{min} \in [-1, 1] \tag{13}$$

to prevent the situation in Fig. 27d. Note the inversion: ω_{max} has to be considered for the small ω case, producing Fig. 27e as the corrected version of Fig. 27c with Eq. (12) enforced. Similarly, ω_{min} has to be considered for the large ω case, producing Fig. 27f as the corrected version of Fig. 27d with Eq. (13) enforced. In order to clearly demonstrate the difference between Fig. 27c, d, the ω range has been kept fixed, adjusting only the c' range. In reality, both ranges would be adjusted to maintain the possibility of sampling convergent parameters, as once again both convergent and non-convergent control parameters can be sampled, discarding non-convergent parameters.

As is clear from Fig. 28, the belief space never actually converges. Furthermore, the stability guiding mechanism works perfectly, as can be seen from Fig. 29, which shows a completely stable swarm, leading to a very small number of infeasible particles in Fig. 30. As a result, particle velocities in Fig. 31 and swarm diversity in Fig. 32 take on values which are orders of magnitude smaller than in Sect. 5.5. However, while the stability guiding mechanism reduces the instability of the search, it actually degrades performance, as is clear when comparing Table 6, where the prefix ‘sg’ indicates experiments employing stability guiding, to Table 5. The performance degradation is probably attributable to exploration of the search space being reduced when only stable control parameter values can be sel ected.

Table 6 Normalized global best solutions of the stability-guided BS-SAPSO

	Evaluation	Testing
sg_stagnate300_elitist30	0.2057	0.1002
sg_stagnate300_improve30	0.2060	0.1001
sg_stagnate100_improve30	0.2087	0.1027
sg_stagnate200_improve30	0.2093	0.1007
sg_stagnate200_elitist30	0.2102	0.1006
sg_stagnate100_elitist30	0.2133	0.1028
sg_fixed300_improve30	0.2180	0.1045
sg_fixed300_elitist30	0.2191	0.1042
sg_fixed200_improve30	0.2232	0.1069
sg_fixed200_elitist30	0.2244	0.1067
sg_fixed100_improve30	0.2267	0.1076
sg_fixed100_elitist30	0.2273	0.1084
sg_always_improve30	0.2654	0.1627
sg_always_elitist30	0.2747	0.1711
sg_always_improve20	0.2836	0.1814
sg_always_elitist20	0.2891	0.1803
sg_always_improve10	0.3035	0.1934
sg_always_elitist10	0.3101	0.1945

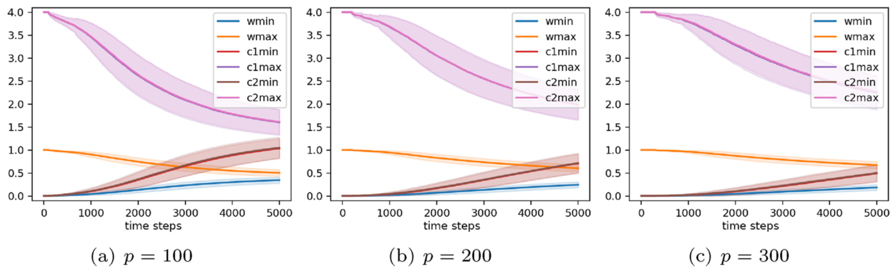


Fig. 28 Belief space for 'Stagnate', improvement selection, stability guided

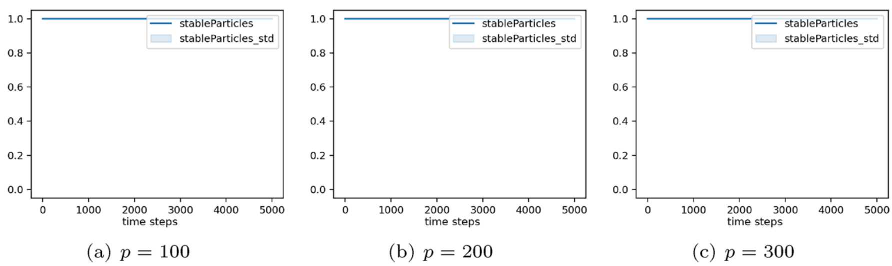
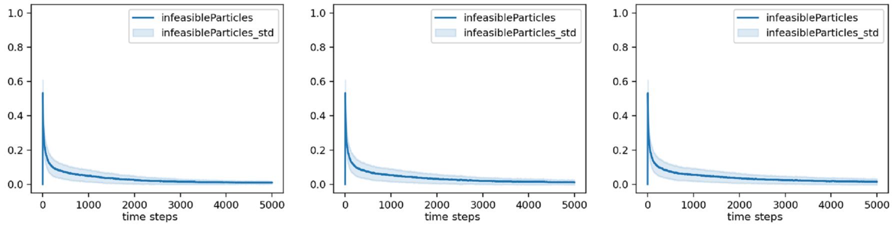
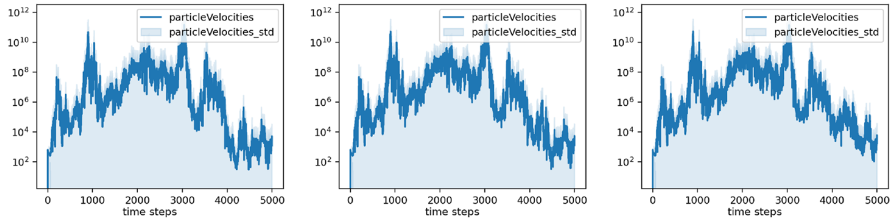


Fig. 29 Stable particles for 'Stagnate', improvement selection, stability guided



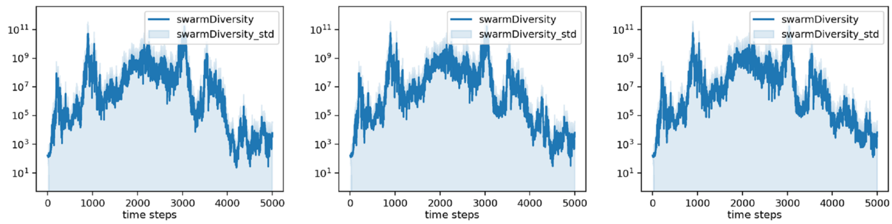
(a) $p = 100$ (b) $p = 200$ (c) $p = 300$

Fig. 30 Infeasible particles for 'Stagnate', improvement selection, stability guided



(a) $p = 100$ (b) $p = 200$ (c) $p = 300$

Fig. 31 Velocity vectors for 'Stagnate', improvement selection, stability guided



(a) $p = 100$ (b) $p = 200$ (c) $p = 300$

Fig. 32 Swarm diversity for 'Stagnate', improvement selection, stability guided

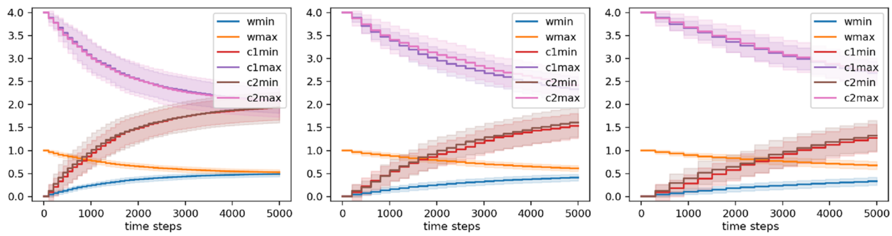
5.7 BS-SAPSO with velocity clamping

The experiments in this section repeat the experiments from Sect. 5.5 while imposing velocity clamping per dimension in an attempt to mitigate velocity explosion, because the stability guidance mechanism from Sect. 5.6, which also sought to reduce velocity explosion, degraded performance of the PSO. The maximum velocity, V_{max}, set equal to the velocity that would allow a particle to traverse the whole search space in one step, and is therefore determined by the domains over which each function is defined. The results are shown in Table 7, where the prefix 'vc' indicates velocity clamping (Figs. 33, 34, 35).

The benefits of velocity clamping are clear, with performance scores that are much better than those of the baseline methods in Table 1. The particle velocities in Fig. 36 and swarm diversity plots in Fig. 37 now clearly do not grow without bounds, and remain in the same order of magnitude as the search space dimensions.

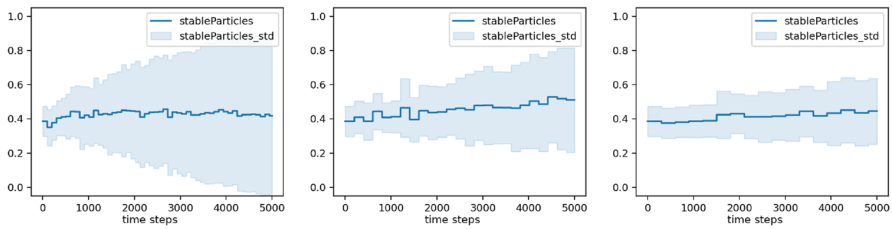
Table 7 Normalized global best solution of velocity clamped BS-SAPSO

	Evaluation	Testing
vc_fixed200_improve30	0.1587	0.0782
vc_fixed100_improve30	0.1631	0.0744
vc_fixed300_improve30	0.1657	0.0812
vc_fixed300_elitist30	0.1693	0.0814
vc_fixed200_elitist30	0.1712	0.0816
vc_fixed100_elitist30	0.1827	0.0819
vc_stagnate100_improve30	0.1921	0.0882
vc_stagnate100_elitist30	0.2027	0.0878
vc_stagnate200_elitist30	0.2039	0.0907
vc_stagnate200_improve30	0.2042	0.0901
vc_stagnate300_improve30	0.2122	0.0936
vc_stagnate300_elitist30	0.2141	0.0933



(a) $p = 100$ (b) $p = 200$ (c) $p = 300$

Fig. 33 Belief space boundaries for improvement selection for $n_e = 30$

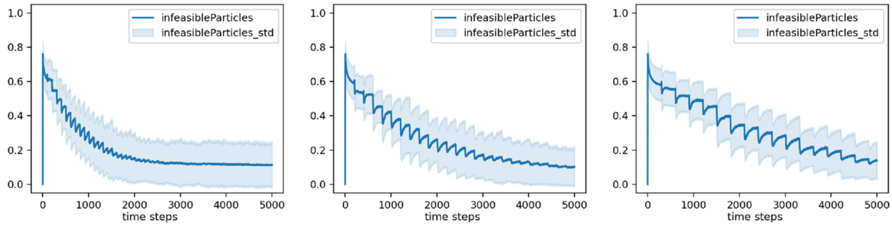


(a) $p = 100$ (b) $p = 200$ (c) $p = 300$

Fig. 34 Stable particles for improvement selection for $n_e = 30$

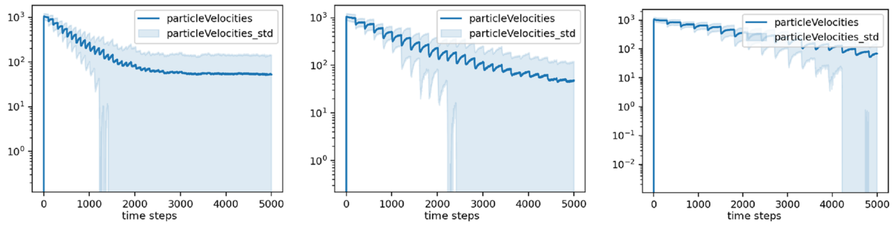
5.8 Overview of results

Generally, the trend observed throughout is that selection methods with higher selective pressure and less randomness, such as elitist and improvement selection, perform better. Furthermore, higher values for n_e and p lead to slower convergence of the belief space, and perform better. Whereas updating the control parameters at fixed intervals results in the belief space boundaries following the statistical mean, updating the belief space only upon stagnation of the global best position demonstrates more nuanced updating of belief



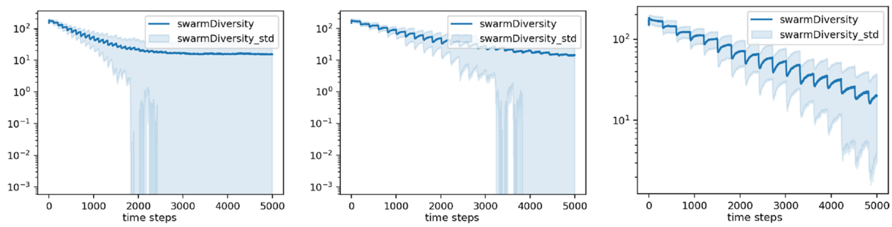
(a) $p = 100$ (b) $p = 200$ (c) $p = 300$

Fig. 35 Infeasible particles for improvement selection for $n_e = 30$



(a) $p = 100$ (b) $p = 200$ (c) $p = 300$

Fig. 36 Particle velocity vectors for improvement selection for $n_e = 30$



(a) $p = 100$ (b) $p = 200$ (c) $p = 300$

Fig. 37 Swarm diversity for improvement selection for $n_e = 30$

space boundaries, because the time step at which the loss stagnates is largely influenced by the landscape of the specific objective function. Doing so does, however, not improve performance, most likely because belief space boundaries are only updated when the loss has already stagnated, at which point it may be too late to select new control parameters which could otherwise promote more exploration. Introduction of a stability criterion for the control parameters, which ensures that the control parameters are always within the belief space boundaries, does not improve performance, but does reduce the number of infeasible particles. Finally, velocity clamping per dimension is shown to be a very effective mechanism for mitigating velocity explosion, and greatly improves performance.

Table 9 in Appendix B shows that the best performing experiments updated the belief space at fixed intervals with $p = \{100, 200, 300\}$, used improvement-based selection with $n_e = 30$, and implemented velocity clamping. The best performing BS-SAPSO variant achieves an improvement of 20% in global best solution quality over PSO-IW on the evaluation function set, and 3% on the test set. When compared to the PSO-TVIW baseline, the improvements in performance are 46 and 55%, respectively.

5.9 Statistical significance

In order to determine whether the performance differences between the BS-SAPSO variants and the baselines are statistically significant, Friedman and Mann–Whitney U tests are performed on the normalized global best solutions, on both the evaluation and test sets.

For the evaluation set, the Friedman test resulted in a test statistic of 2052.02 and a p-value of 0.0, indicating statistical significance. For the test set, the Friedman test resulted in a test statistic of 2237.54 and a p-value of 0.0, indicating statistical significance.

While the Friedman test was performed for all BS-SAPSO variants, the Mann–Whitney U test (and subsequent analysis) is only performed for the 18 best performing variants, together with PSO-RAC. The Mann–Whitney U test used the Bonferroni correction to account for multiple comparisons, with $\alpha = 0.02/num\_comparisons$. These are compared against PSO-IW, because it is the best performing baseline. The results of the Mann–Whitney U tests are shown in Table 8, with box plots in Fig. 38 and 39. Corresponding Mann–Whitney U heatmaps of the p-values are given in Figs. 40 and 41 in Appendix A.

The top-performing algorithms, such as `vc_fixed100_improve30`, are therefore vetted as statistically significant improvements over PSO-IW.

Table 8 Mann–Whitney U test results for BS-SAPSO variants compared to PSO-IW

Name	Evaluation			Testing		
	MWU	p-Value	Sig	MWU	p-Value	Sig
vc_fixed100_elitist30	257.0	0.0044	No	484.0	0.6204	No
vc_fixed100_improve30	91.0	0.0000	Yes	240.0	0.0020	Yes
vc_fixed200_elitist30	161.0	0.0000	Yes	457.0	0.9234	No
vc_fixed200_improve30	59.0	0.0000	Yes	334.0	0.0877	No
vc_fixed300_elitist30	109.0	0.0000	Yes	447.0	0.9705	No
vc_fixed300_improve30	95.0	0.0000	Yes	457.0	0.9234	No
vc_stagnate100_elitist30	506.0	0.4119	No	673.0	0.0010	Yes
vc_stagnate100_improve30	361.0	0.1907	No	673.0	0.0010	Yes
vc_stagnate200_elitist30	515.0	0.3403	No	746.0	0.0000	Yes
vc_stagnate200_improve30	535.0	0.2116	No	722.0	0.0001	Yes
sg_fixed100_improve30	839.0	0.0000	Yes	898.0	0.0000	Yes
fixed10_improve30	582.0	0.0519	No	559.0	0.1087	No
fixed20_elitist30	616.0	0.0144	No	715.0	0.0001	Yes
fixed20_improve30	492.0	0.5395	No	525.0	0.2707	No
fixed50_elitist30	551.0	0.1373	No	753.0	0.0000	Yes
fixed50_improve30	347.0	0.1297	No	561.0	0.1023	No
fixed100_improve30	378.0	0.2905	No	698.0	0.0003	Yes
stagnate20_improve30	643.0	0.0044	No	770.0	0.0000	Yes
stagnate10_improve30	695.0	0.0003	Yes	724.0	0.0001	Yes
PSO-RAC	414.0	0.5997	No	142.0	0.0000	Yes

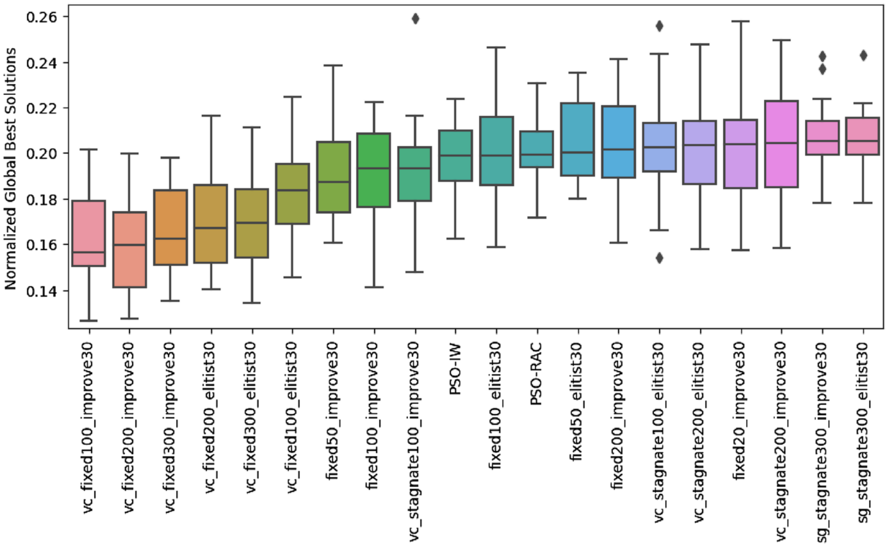


Fig. 38 Box plot of normalized global best solutions on the evaluation set

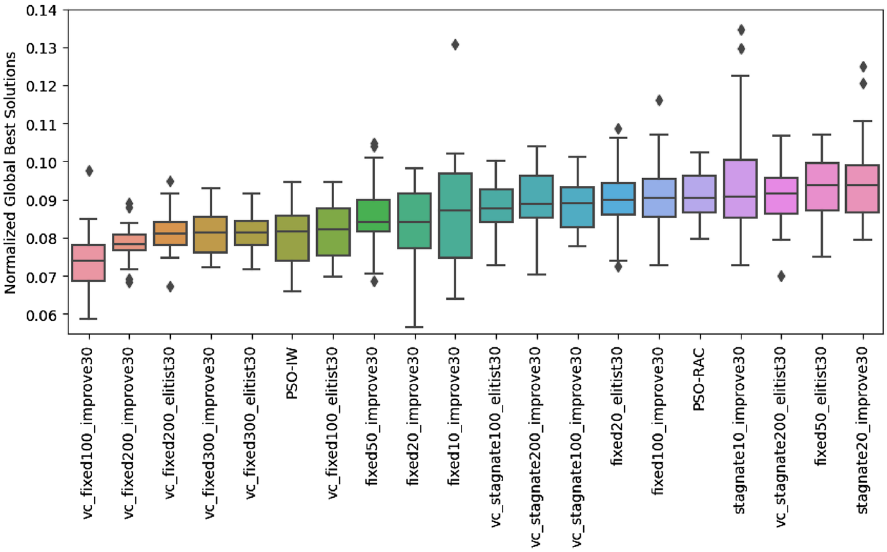


Fig. 39 Box plot of normalized global best solutions on the test set

5.10 Computational complexity

The computational load of the inertia weight PSO is influenced by a number of factors:

- The number of particles in the swarm (n_s): each particle represents a potential solution to the optimization problem, therefore more particles imply the exploration of more potential solutions.

- The dimensionality of the problem (n_d): higher dimensionality implies a larger search space and therefore more complexity.
- The number of iterations (i_{max}): the PSO algorithm iteratively updates the particles' positions based on the best-known positions, and more iterations require more compute.

As such, the computational complexity of the BS-SAPSO algorithm is expressed as follows:

- Initialization of particles has a complexity of $\mathcal{O}(n_s \cdot n_d)$, because each particle needs an initial position in each dimension.
- The main loop, where the particles' positions and velocities are updated, has a complexity of $\mathcal{O}(i_{max} \cdot n_s \cdot n_d)$. The position and velocity has to be updated in every dimension, for every particle, on every time step.
- Selection of the global best solution has a complexity of $\mathcal{O}(n_s)$, because it requires each particle to be checked.
- Updating of the belief space has a complexity of $\mathcal{O}(n_s)$, because it requires each particle to be checked.
- Updating of the control parameters has a complexity of $\mathcal{O}(1)$, because it is a simple operation.

The total computational complexity of the BS-SAPSO algorithm is $\mathcal{O}(i_{max} \cdot n_s \cdot n_d)$, which is the same as the complexity of the inertia weight PSO algorithm. The BS-SAPSO does not introduce nested loops, and the additional operations are $\mathcal{O}(1)$ or $\mathcal{O}(n_s)$. Therefore, the BS-SAPSO algorithm does not increase the asymptotic computational complexity of the inertia weight PSO algorithm.

6 Conclusions

The study designed a self-adaptive particle swarm optimization (SAPSO) algorithm using a belief space (BS). Considering the performance improvements obtained above the baseline, the proposed belief space-guided self-adaptive particle swarm optimization (BS-SAPSO) algorithm can be considered successful. The best performance is achieved by setting the number of particles which update the BS (n_e) to a high value, as well as using large values for the interval at which the belief space is updated (p), which prevents premature BS convergence by delaying updates. However, while improving performance and adapting the PSO control parameters (CPs), the algorithm does introduce n_e and p as new parameters, as well as the velocity clamping bound v_{max}. That being said, if n_e is always set to equal the swarm size (which seems to be optimal in almost all experiments conducted), and v_{max} as the distance between the points furthest apart in the search space, the case can be made that the CPs have been reduced from three to one. While not exhaustive, the brute force grid search undertaken in Sect. 5 does serve to elucidate which type of belief space behaviour results in relatively better and worse performance, and shows that it is generally the case that more exploration (slower convergence) of the belief space leads to better performance.

It is also important to note that all experiments were conducted for a fixed swarm size, function dimensionality, and maximum time steps. To summarize, the algorithm improves performance, prevents velocity explosion, and reduces the number of CPs from three to one.

7 Future work

Since the belief space tends to converge around the average of the initial belief space boundaries, it might be worth performing an ablation study to see whether the belief space mechanism is truly adapting to each objective function, or whether it is simply converging at the statistical mean. To test this, the convergence of the belief space limits could be hard-coded, after which the performance of the algorithm could be compared to the performance of the algorithm with a self-adaptive belief space mechanism. If the performance is the same, it would suggest that the belief space mechanism is not adapting to the objective function, but that decreasing the belief space boundaries in a way such as in Fig. 33 does nonetheless improve performance.

Evaluation of the BS approach to self-adaptation on other metaheuristics, such as genetic algorithms and differential evolution, could also be worthwhile and shed light on generalisability of the algorithm. Similarly with other problem classes, such as large-scale, dynamic, multi-objective, and real-world optimization problems.

A potentially promising avenue for improving adaptiveness might be the use of reinforcement learning (RL). Since the problem of finding a self-adaptive PSO algorithm essentially amounts to finding a policy which governs the adjustment of control parameter values, based on observing the particles, in order to find the optimal objective function value, this presents an exemplary reinforcement learning problem where an agent finds a policy to govern its actions, based on observations, in order to maximise a reward signal. Such an approach stands to gain from the following advantages:

- an RL agent could explore millions of policies, many more than could be investigated manually;
- if the RL experiment is configured correctly, and if there is a policy to be found, there is a higher likelihood of finding said policy, because the agent can take into account far more correlations and relationships in the runtime behaviour than can be done manually; and
- a policy found in this way should represent as an algorithmic set of actions and will therefore hopefully not introduce additional control parameters

An investigation into RL-based SAPSO therefore seems like a promising avenue for future work.

Appendix A: Heatmaps

The heatmaps in Figs. 40 and 41 show the Mann–Whitney *U* *p*-values for the 20 top performing algorithms, on the evaluation and test sets, respectively.

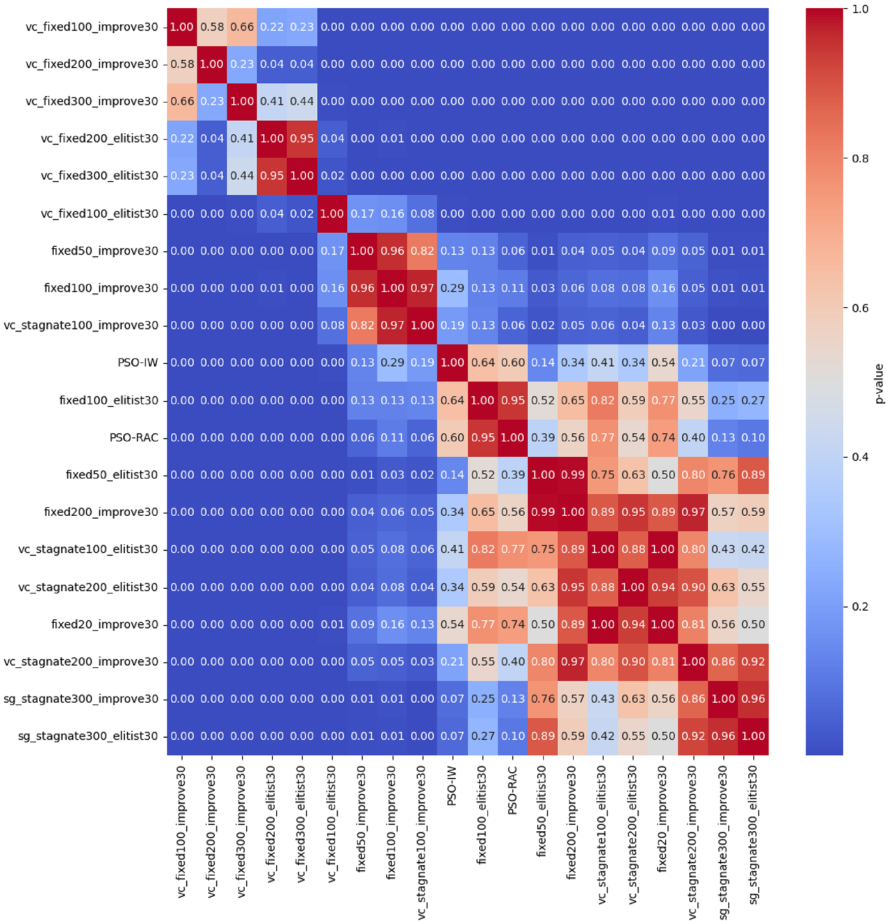


Fig. 40 Heatmap of normalized global best solutions on the evaluation set

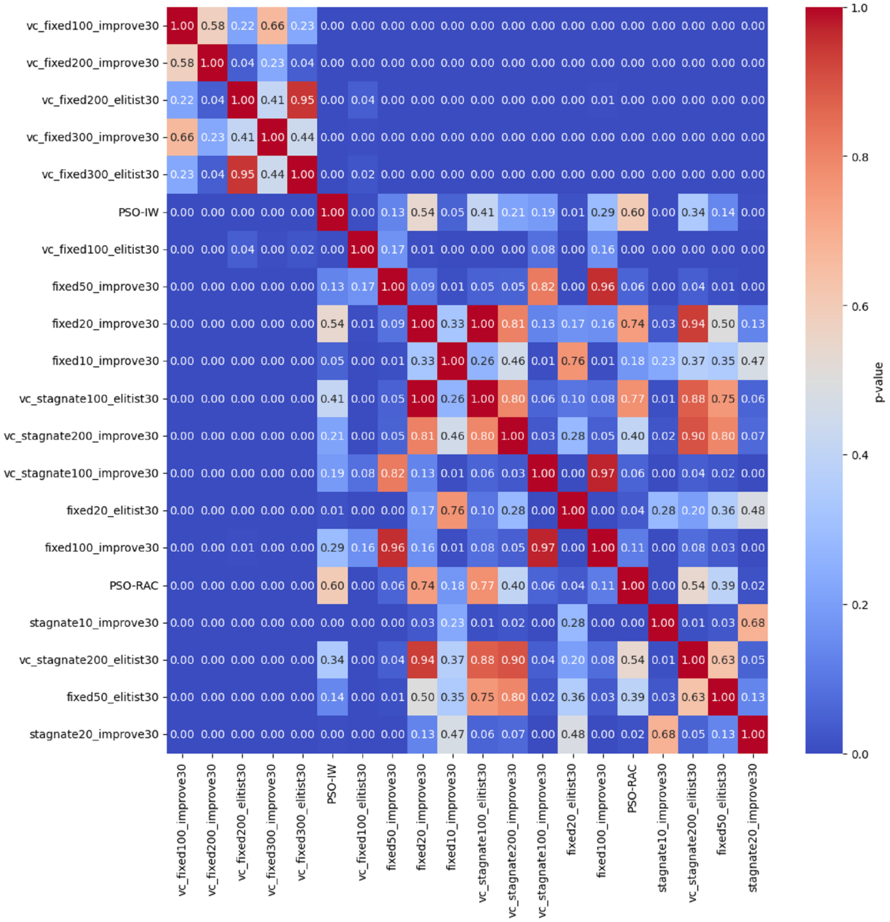


Fig. 41 Heatmap of normalized global best solutions on the test set

Appendix B: Normalized global best solutions

The normalized best solutions for all experiments on both the evaluation and test sets are given in Table 9, sorted from best to worst.

Table 9 Normalized best solutions for all experiments

	Evaluation	Testing
vc_fixed200_improve30	0.1587	0.0782
vc_fixed100_improve30	0.1631	0.0744
vc_fixed300_improve30	0.1657	0.0812
vc_fixed300_elitist30	0.1693	0.0814
vc_fixed200_elitist30	0.1712	0.0816
vc_fixed100_elitist30	0.1827	0.0819
fixed100_improve30	0.1919	0.0900
vc_stagnate100_improve30	0.1921	0.0882
fixed50_improve30	0.1922	0.0852
PSO-IW	0.1976	0.0808
PSO-RAC	0.2004	0.0910
fixed100_elitist30	0.2023	0.0959
vc_stagnate100_elitist30	0.2027	0.0878
vc_stagnate200_elitist30	0.2039	0.0907
vc_stagnate200_improve30	0.2042	0.0901
fixed200_improve30	0.2050	0.0996
sg_stagnate300_elitist30	0.2057	0.1002
sg_stagnate300_improve30	0.2060	0.1001
fixed200_elitist30	0.2064	0.0998
fixed50_elitist30	0.2077	0.0946
sg_stagnate100_improve30	0.2087	0.1027
fixed20_improve30	0.2090	0.0882
sg_stagnate200_improve30	0.2093	0.1007
sg_stagnate200_elitist30	0.2102	0.1006
fixed20_improve20	0.2112	0.0957
vc_stagnate300_improve30	0.2122	0.0936
sg_stagnate100_elitist30	0.2133	0.1028
vc_stagnate300_elitist30	0.2141	0.0933
stagnate20_improve30	0.2157	0.0966
fixed20_elitist30	0.2169	0.0963
sg_fixed300_improve30	0.2180	0.1045
fixed10_improve30	0.2181	0.0939
fixed50_improve20	0.2186	0.0954
stagnate10_elitist30	0.2188	0.0969
sg_fixed300_elitist30	0.2191	0.1042
fixed300_elitist30	0.2195	0.1081
stagnate10_improve30	0.2208	0.0960
fixed300_improve30	0.2213	0.1084
stagnate10_improve20	0.2228	0.1009
fixed20_elitist20	0.2231	0.1024
fixed10_elitist30	0.2232	0.1014
sg_fixed200_improve30	0.2232	0.1069
sg_fixed200_elitist30	0.2244	0.1067
stagnate20_improve20	0.2253	0.1012
stagnate20_elitist30	0.2255	0.1006
sg_fixed100_improve30	0.2267	0.1076

Table 9 (continued)

	Evaluation	Testing
stagnate20_elitist20	0.2270	0.1052
stagnate50_improve20	0.2270	0.1033
sg_fixed100_elitist30	0.2273	0.1084
fixed10_improve20	0.2281	0.1049
stagnate10_elitist20	0.2287	0.1088
stagnate50_improve30	0.2297	0.1003
fixed50_elitist20	0.2305	0.1031
stagnate50_elitist30	0.2317	0.1047
stagnate50_elitist20	0.2320	0.1050
stagnate100_improve30	0.2345	0.1098
fixed10_elitist20	0.2380	0.1150
stagnate100_elitist30	0.2398	0.1093
stagnate50_improve10	0.2455	0.1204
stagnate50_elitist10	0.2480	0.1198
stagnate200_elitist30	0.2483	0.1168
stagnate20_improve10	0.2487	0.1321
stagnate200_improve30	0.2488	0.1173
stagnate10_improve10	0.2498	0.1344
stagnate10_elitist10	0.2501	0.1425
stagnate20_elitist10	0.2512	0.1343
PSO-TVAC	0.2543	0.1395
stagnate300_improve30	0.2555	0.1211
stagnate300_elitist30	0.2560	0.1208
fixed50_improve10	0.2600	0.1258
fixed50_elitist10	0.2604	0.1240
sg_always_improve30	0.2654	0.1627
fixed20_improve10	0.2677	0.1387
fixed20_elitist10	0.2723	0.1427
fixed10_improve10	0.2743	0.1450
sg_always_elitist30	0.2747	0.1711
always_elitist20	0.2764	0.1570
fixed10_elitist10	0.2787	0.1542
always_improve20	0.2801	0.1569
always_improve10	0.2813	0.1665
sg_always_improve20	0.2836	0.1814
always_improve30	0.2838	0.1596
sg_always_elitist20	0.2891	0.1803
always_elitist10	0.2944	0.1770
PSO-TVIW	0.2955	0.1756
always_improveMag30	0.2982	0.1799
always_elitist30	0.2984	0.1798
always_improveMag10	0.2989	0.1792
always_improveMag20	0.2989	0.1858
sg_always_improve10	0.3035	0.1934
sg_always_elitist10	0.3101	0.1945

Table 9 (continued)

	Evaluation	Testing
always_tournament10	0.3127	0.1890
always_tournament20	0.3150	0.1828
always_random	0.3209	0.2122
always_tournament30	0.3226	0.1858
always_roulette	0.3298	0.2088
always_rank	0.3305	0.2081

Appendix C: Function set

The functions used in the experiments are given in Table 10, with test set functions at the end in bold. Equations for all functions follow afterwards. *C* Continuous, *NC* non-continuous, *D* differentiable, *ND* non-differentiable, *S* separable, *NS* non-separable, *MM* multi-modal, *UM* unimodal.

- Ackley 1 for $x_i \in [-32, 32]$:

$$f(\mathbf{x}) = -20e^{-0.2\sqrt{\frac{1}{n}\sum_{j=1}^{n} x_i^2}} - e^{\frac{1}{n}\sum_{j=1}^{n}\cos(2\pi x_i)} + 20 + e \tag{C1}$$

- Alpine 1 for $x_i \in [-10, 10]$:

$$f(\mathbf{x}) = \sum_{i=1}^{n} |x_i \sin(x_i) + 0.1x_i| \tag{C2}$$

- Attractive Sector for $x_i \in [-1, 1]$:

$$f(\mathbf{x}) = T_{\text{osz}}\left(\sum_{i=1}^{D} \left(s_i x_i\right)^2\right)^{0.9} + f_{\text{opt}} \tag{C3}$$

where

$$s_i = \begin{cases} 10^2 & \text{if } z_i \times x_i^{\text{opt}} > 0 \\ 1 & \text{otherwise} \end{cases}$$

- Bohachevsky 1 for $x_i \in [-15, 15]$:

$$f(\mathbf{x}) = \sum_{i=1}^{n-1}(x_i^2 + 2x_{i+1}^2 - 0.3\cos(3\pi x_i) - 0.4\cos(4\pi x_{i+1}) + 0.7) \tag{C4}$$

- Bonyadi Michalewicz for $x_i \in [-5, 5]$:

$$f(\mathbf{x}) = \frac{\prod_{i=1}^{n}(x_i + 1)}{\prod_{i=1}^{n}((x_i + 1)^2 + 1)} \tag{C5}$$

Table 10 Function characteristics (Gavana, 2022)

Function	Equation	Cont.	Diff.	Sep.	Mod.
Ackley 1	(C1)	C	D	NS	MM
Alpine 1	(C2)	C	ND	S	MM
Bohachevsky1	(C4)	C	D	S	MM
Cross leg table	(C8)	?	ND	?	MM
Deflected corrugated spring	(C9)	C	D	??	MM
Egg holder	(C13)	C	ND	NS	MM
Holder Table 1	(C18)	C	ND	NS	MM
Lanczos 3	(C19)	C	D	NS	MM
Levy 3	(C20)	C	D	NS	MM
Levy-Montalvo 2	(C21)	C	D	NS	MM
Michalewicz	(C22)	C	D	S	MM
Norwegian	(C26)	C	D	NS	MM
Pathological	(C27)	C	D	NS	MM
Penalty 1	(C29)	C	D	NS	MM
Penalty 2	(C30)	C	D	NS	MM
Periodic	(C31)	C	D	S	MM
Quadric	(C35)	C	D	NS	UM
Quintic	(C36)	C	ND	S	MM
Rana	(C37)	C	ND	NS	MM
Rastrigin	(C38)	C	D	S	MM
Salomon	(C41)	C	D	NS	MM
Schubert 4	(C43)	C	D	S	MM
Sine envelope	(C45)	C	D	NS	MM
Sinusoidal	(C46)	C	D	NS	MM
Stretched V sine wave	(C48)	C	D	NS	MM
Trid	(C49)	C	D	NS	MM
Trigonometric	(C50)	C	D	NS	MM
Vincent	(C53)	C	D	S	MM
Wavy	(C51)	C	D	S	UM
Xin-She Yang 1	(C54)	NC (?)	D (?)	S	MM
Xin-She Yang 2	(C55)	C	ND	NS	MM
Attractive sector	(C3)	C	D	S	UM
Bonyadi–Michalewicz	(C5)	C	D	NS	MM
Brown	(C6)	C	D	NS	UM
Cosine mixture	(C7)	C	D	S	MM
Discuss	(C10)	C	D	S	UM
Drop wave	(C11)	C	D	??	MM
Egg crate	(C12)	C	D	S	MM
Elliptic	(C14)	C	D	S	UM
Ellipsoid	(C14)	C	D	S	UM
Exponential	(C16)	C	D	??	UM
Giunta	(C17)	C	D	S	MM
Mishra 1	(C23)	C	D	NS	MM
Mishra 7	(C24)	C	ND	NS	MM
Needle eye	(C25)	C	ND	S	MM
Paviani	(C28)	C	D	NS	MM

Table 10 (continued)

Function	Equation	Cont.	Diff.	Sep.	Mod.
Pinter 2	(C32)	C	D	NS	MM
Price 2	(C33)	C	D	NS	MM
Qings	(C34)	C	D	S	MM
Ripple 25	(C39)	C	D	S	MM
Rosenbrock	(C40)	C	D	NS	MM
Schaffer F7	(C42)	C	D	NS	MM
Schwefel 1	(C44)	C	D	S	UM
Step function 3	(C47)	NC	ND	S	MM
Weierstrass	(C52)	C	D	S	MM

- Brown for $x_i \in [-1, 1]$:

$$f(\mathbf{x}) = \sum_{i=1}^{n-1} (x_i^2)^{(x_{i+1}^2+1)} + (x_{i+1}^2)^{(x_i^2+1)} \tag{C6}$$

- Cosine Mixture for $x_i \in [-1, 1]$:

$$f(\mathbf{x}) = 0.1 \sum_{i=1}^{n} \cos(5\pi x_i) + \sum_{i=1}^{n} x_i^2 \tag{C7}$$

- Cross Leg Table for $x_i \in [-10, 10]$:

$$f(\mathbf{x}) = -\frac{1}{\left(\left| e^{|100 - \frac{\sqrt{\sum_{i=1}^{n} x_i^2}}{\pi}|} \prod_{i=1}^{n} \sin(x_i) \right| + 1\right)^{0.1}} \tag{C8}$$

- Deflected Corrugated Spring for $x_i \in [0, 2\alpha]$, $K = 5$, $\alpha = 5$:

$$f(\mathbf{x}) = 0.1 \sum_{i=1}^{n} \left[(x_i - \alpha)^2 - \cos\left(K \sqrt{\sum_{i=1}^{n} (x_i - \alpha)^2} \right) \right] \tag{C9}$$

- Discuss for $x_i \in [-100, 100]$:

$$f(\mathbf{x}) = 10^6 x_1^2 + \sum_{i=2}^{n} x_i^2 \tag{C10}$$

- Drop Wave for $x_i \in [-5.12, 5.12]$:

$$f(\mathbf{x}) = -\frac{1 + \cos(12\sqrt{\sum_{i=1}^{n} x_i^2})}{2 + 0.5 \sum_{i=1}^{n} x_i^2} \tag{C11}$$

- Egg crate for $x_i \in [-5, 5]$:

$$f(\mathbf{x}) = \sum_{i=1}^{n} x_i^2 + 24 \sum_{i=1}^{n} \sin(x_i)^2 \tag{C12}$$

- Egg holder for $x_i \in [-512, 512]$:

$$f(\mathbf{x}) = \sum_{i=1}^{n-1} \left(-(x_{i+1} + 47) \sin(\sqrt{|x_{i+1} + x_i/2 + 47|}) - x_i \sin(\sqrt{|x_i - (x_{i+1} + 47)|}) \right) \tag{C13}$$

- Elliptic for $x_i \in [-100, 100]$:

$$f(\mathbf{x}) = \sum_{i=1}^{n} (10^6)^{\frac{i-1}{n-1}} x_i^2 \tag{C14}$$

- Ellipsoid for $x_i \in [-10, 10]$:

$$f(\mathbf{x}) = \sum_{i=1}^{n} i x_i^2 \tag{C15}$$

- Exponential for $x_i \in [-1, 1]$:

$$f(\mathbf{x}) = -e^{-0.5 \sum_{i=1}^{n} x_i^2} \tag{C16}$$

- Giunta for $x_i \in [-1, 1]$:

$$f(\mathbf{x}) = 0.6 + \sum_{i=1}^{2} \left(\sin(\frac{16}{15} x_i - 1) + \sin(\frac{16}{15} x_i - 1)^2 + \frac{1}{50} \sin(4(\frac{16}{15} x_i - 1)) \right) \tag{C17}$$

- Holder Table 1 for $x_i \in [-10, 10]$:

$$f(\mathbf{x}) = -\left| \left(\prod_{i=1}^{n} \cos(x_i) \right) e^{|1 - \sqrt{\sum_{i=1}^{n} x_i}/\pi|} \right| \tag{C18}$$

- Lanczos 3 for $x_i \in [-20, 20]$:

$$f(\mathbf{x}) = \prod_{i=1}^{n} \text{sinc}\,(x_i)\, \text{sinc}\,(x_i/k) \tag{C19}$$

- Levy 3 for $x_i \in [-10, 10]$:

$$f(\mathbf{x}) = \sin^2(\pi y_1) + \sum_{i=1}^{n-1} (y_i - 1)^2 [1 + 10 \sin^2(\pi y_{i+1})] + (y_n - 1)^2 \tag{C20}$$

where

$$y_i = 1 + \frac{x_i - 1}{4}$$

- Levy–Montalvo 2 for $x_i \in [-5, 5]$:

$$f(\mathbf{x}) = 0.1\sin(3\pi x_1)^2 + 0.1\sum_{i=1}^{n-1}(x_i-1)^2\left[\sin^2\left(3\pi x_{i+1}\right)+1\right] + 0.1\left(x_n-1\right)^2\left[\sin^2\left(2\pi x_n\right)+1\right] \tag{C21}$$

- Michalewicz for $x_i \in [0, \pi]$:

$$f(\mathbf{x}) = -\sum_{i=1}^{n}\sin(x_i)\left(\sin\left(\frac{ix_i^2}{\pi}\right)\right)^{2m} \tag{C22}$$

- Mishra 1 for $x_i \in [0, 1]$:

$$f(\mathbf{x}) = \left(1 + n - \sum_{i=1}^{n-1} x_i\right)^{(n-\sum_{i=1}^{n-1} x_i)} \tag{C23}$$

- Mishra 7 for $x_i \in [-10, 10]$:

$$f(\mathbf{x}) = \sqrt{\left|\sin\sqrt{|\sum_{i=1}^{n} x_i^2|}\right|} + 0.01\sum_{i=1}^{n} x_i \tag{C24}$$

- Needle eye for $x_i \in [-10, 10]$ and $eye = 0.0001$:

$$f(\mathbf{x}) = \begin{cases} 1 & \text{if } |x_i| < eye \ \ \forall i \\ \sum_{i=1}^{n}(100+|x_i|) & \text{if } |x_i| > eye \\ 0 & \text{otherwise} \end{cases} \tag{C25}$$

- Norwegian for $x_i \in [-1.1, 1.1]$:

$$f(\mathbf{x}) = \prod_{j=1}^{n_x}\left(\cos(\pi x_j^3)\left(\frac{99+x_j}{100}\right)\right) \tag{C26}$$

- Pathological for $x_i \in [-100, 100]$:

$$f(\mathbf{x}) = \sum_{i=1}^{n-1}\left(\frac{\sin^2\sqrt{100x_i^2+x_{i+1}^2}-0.5}{0.5+0.001(x_i-x_{i+1})^4}\right) \tag{C27}$$

- Paviani for $x_i \in [2.001, 9.999]$:

$$f(\mathbf{x}) = \sum_{i=1}^{n}\left[\log^2\left(10-x_i\right)+\log^2\left(x_i-2\right)\right] - \left(\prod_{i=1}^{n} x_i^{10}\right)^{0.2} \tag{C28}$$

- Penalty 1 for $x_i \in [-50, 50]$:

$$f(\mathbf{x}) = \frac{\pi}{30}\left[10\sin^2(\pi y_1) + \sum_{i=1}^{n-1}(y_i-1)^2\left[1+10\sin^2(\pi y_{i+1})\right] + (y_n-1)^2\right] + \sum_{i=1}^{n} u(x_i, 10, 100, 4) \tag{C29}$$

where

$$y_i = 1 + \frac{1}{4}(x_i + 1)$$

and

$$u(x_i, a, k, m) = \begin{cases} k(x_i - a)^m & \text{if } x_i > a \\ 0 & \text{if } -a \le x_i \le a \\ k(-x_i - a)^m & \text{if } x_i < -a \end{cases}$$

- Penalty 2 for $x_i \in [-50, 50]$:

$$f(\mathbf{x}) = 0.1\left(\sin^2(3\pi x_1) + \sum_{i=1}^{n-1}(x_i-1)^2\left[1+\sin^2(3\pi x_{i+1})\right] + (x_n-1)^2\left[1+\sin^2(2\pi x_n)\right]\right) + \sum_{i=1}^{n} u(x_i, 5, 100, 4) \tag{C30}$$

where

$$u(x_i, a, k, m) = \begin{cases} k(x_i - a)^m & \text{if } x_i > a \\ 0 & \text{if } -a \le x_i \le a \\ k(-x_i - a)^m & \text{if } x_i < -a \end{cases}$$

- Periodic for $x_i \in [-10, 10]$:

$$f(\mathbf{x}) = 1 + \sum_{i=1}^{n} \sin^2(x_i) - 0.1e^{-\sum_{i=1}^{n} x_i^2} \tag{C31}$$

- Pinter for $x_i \in [-10, 10]$:

$$f(\mathbf{x}) = \sum_{i=1}^{n} i x_i^2 + \sum_{i=1}^{n} 20 i \sin^2 A + \sum_{i=1}^{n} i \log_{10}(1 + iB^2) \tag{C32}$$

- Price 2 for $x_i \in [-10, 10]$:

$$f(\mathbf{x}) = 1 + \sum_{i=1}^{n} \sin^2(x_i) - 0.1e^{-\sum_{i=1}^{n} x_i^2} \tag{C33}$$

- Qings for $x_i \in [-500, 500]$:

$$f(\mathbf{x}) = \sum_{i=1}^{n} (x_i^2 - i)^2 \tag{C34}$$

- Quadric for $x_i \in [-100, 100]$:

$$f(\mathbf{x}) = \sum_{i=1}^{n} \left(\sum_{j=1}^{i} x_j \right)^2 \tag{C35}$$

- Quintic for $x_i \in [-10, 10]$:

$$f(\mathbf{x}) = \sum_{i=1}^{n} \left| x_i^5 - 3x_i^4 + 4x_i^3 + 2x_i^2 - 10x_i - 4 \right| \tag{C36}$$

- Rana for $x_i \in [-500, 500]$:

$$f(\mathbf{x}) = \sum_{i=1}^{n-1} (x_{i+1} + 1) \cos(t_2) \sin(t_1) + x_i \cos(t_1) \sin(t_2) \tag{C37}$$

where

$$t_1 = \sqrt{|x_{i+1} + x_i + 1|} \quad \text{and} \quad t_2 = \sqrt{|x_{i+1} - x_i + 1|}$$

- Rastrigin for $x_i \in [-5.12, 5.12]$

$$f(\mathbf{x}) = 10n + \sum_{i=1}^{n} \left(x_i^2 - 10 \cos(2\pi x_i) \right) \tag{C38}$$

- Ripple 25 for $x_i \in [0, 1]$:

$$f(\mathbf{x}) = \sum_{i=1}^{n} -e^{-2 \log_2 \left(\frac{x_i - 0.1}{0.8}\right)^2 \left(\sin^6(5\pi x_i)\right)} \tag{C39}$$

- Rosenbrock for $x_i \in [-30, 30]$:

$$f(\mathbf{x}) = \sum_{i=1}^{n-1} \left(100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2 \right) \tag{C40}$$

- Salomon for $x_i \in [-100, 100]$:

$$f(\mathbf{x}) = -\cos(2\pi \sum_{i=1}^{n} x_i^2) + 0.1 \sqrt{\sum_{i=1}^{n} x_i^2} + 1 \tag{C41}$$

- Schaffer F7 for $x_i \in [-100, 100]$:

$$f(\mathbf{x}) = \sum_{i=1}^{n-1} \left(0.5 + \frac{\cos^2(\sin(x_i^2 - x_{i+1}^2))^2 - 0.5}{(1 + 0.001(x_i^2 + x_{i+1}^2))^2} \right) \tag{C42}$$

- Schubert 4 for $x_i \in [-10, 10]$:

$$f(\mathbf{x}) = \sum_{i=1}^{n} \sum_{j=1}^{5} j \cos((j + 1)x_i + j) \tag{C43}$$

- Schwefel 1 for $x_i \in [-100, 100]$ and $\alpha = \sqrt{\pi}$:

$$f(\mathbf{x}) = \left(\sum_{i=1}^{n} x_i^2 \right)^{\alpha} \tag{C44}$$

- Sine Envelope for $x_i \in [-100, 100]$:

$$f(\mathbf{x}) = \sum_{i=1}^{n-1} \left(0.5 + \frac{\sin^2 \sqrt{x_i^2 + x_{i+1}^2} - 0.5}{(1 + 0.001(x_i^2 + x_{i+1}^2))^2} \right) \tag{C45}$$

- Sinusoidal for $x_i \in [0, 180]$:

$$f(\mathbf{x}) = -\left[A \prod_{i=1}^{n} \sin(x_i - z) + \prod_{i=1}^{n} \sin(B(x_i - z)) \right] \tag{C46}$$

- Step function 3 for $x_i \in [-5.12, 5.12]$:

$$f(\mathbf{x}) = \sum_{i=1}^{n} \lfloor x_i^2 \rfloor \tag{C47}$$

- Stretched V sine wave for $x_i \in [-10, 10]$:

$$f(\mathbf{x}) = \sum_{i=1}^{n-1} (x_i^2 + x_{i+1}^2)^{0.25} [\sin^2(50(x_i^2 + x_{i+1}^2)^{0.1}) + 0.1] \tag{C48}$$

- Trid for $x_i \in [-20, 20]$:

$$f(\mathbf{x}) = \sum_{i=1}^{n} (x_i - 1)^2 - \sum_{i=2}^{n} x_i x_{i-1} \tag{C49}$$

- Trigonometric for $x_i \in [0, \pi]$:

$$f(\mathbf{x}) = \sum_{i=1}^{n} \left[n - \sum_{j=1}^{n} \cos(x_j) + i(1 - \cos(x_i) - sin(x_i)) \right]^2 \tag{C50}$$

- Wavy for $x_i \in [-\pi, \pi]$:

$$f(\mathbf{x}) = 1 - \frac{1}{n} \sum_{i=1}^{n} \cos(kx_i) e^{\frac{-x_i^2}{2}} \tag{C51}$$

- Weierstrass for $j_{max} = 20, a = 0.5, b = 3$:

$$f(\mathbf{x}) = \sum_{i=1}^{n} \left(\sum_{j=0}^{j_{max}} a^j \cos(2\pi b^j (x_i + 0.5)) - n \sum_{j=1}^{j_{max}} a^j \cos(\pi b^j) \right) \tag{C52}$$

- Vincent for $x_i \in [0.25, 10]$:

$$f(\mathbf{x}) = -\sum_{i=1}^{n} \sin(10\log(x)) \tag{C53}$$

- Xin-She Yang 1 for $\epsilon_i \sim U(0,1)$:

$$f(\mathbf{x}) = \sum_{i=1}^{n} \epsilon_i |x_i|^i \tag{C54}$$

- Xin-She Yang 2 for $x_i \in [-2\pi, 2\pi]$:

$$f(\mathbf{x}) = \left(\sum_{i=1}^{n} |x_i|\right) e^{-\sum_{i=1}^{n}\sin(x_i^2)} \tag{C55}$$

Acknowledgements The authors acknowledge the computing resources provided by the National Integrated Cyber Infrastructure System's centre for high-performance computing.

Author Contributions Conceptualization was done by Andries Engelbrecht. Development of algorithms, methodology, data collection and analysis were performed by Daniel von Eschwege. The first draft of the manuscript was written by Daniel von Eschwege, and review of the manuscript was done by Andries Engelbrecht. All authors approved the final manuscript.

Funding Open access funding provided by Stellenbosch University. The authors did not receive support from any organization for the submitted work.

Data Availability The datasets generated during and/or analysed during the current study are available from the corresponding author on reasonable request.

Declarations

Conflict of interest The authors have no relevant financial or non-financial interests to disclose.

Ethical approval and consent to participate Not applicable.

Consent for publication All authors gave consent that the paper may be submitted and consent has bene provided by the institutions to which the authors are affiliated.

References

Beielstein, T., Parsopoulos, K. E., & Vrahatis, M. N. (2002). *Tuning pso parameters through sensitivity analysis* (Technical Report Interner Bericht des Sonderforschungsbereichs (SFB) 531 Computational Intelligence No. CI-124/02, Universitätsbibliothek Dortmund).

Bergh, F., & Engelbrecht, A. P. (2006). A study of particle swarm optimization particle trajectories. *Information Sciences, 176*(8), 937–971.

Bonyadi, M. R., & Michalewicz, Z. (2016). Impacts of coefficients on movement patterns in the particle swarm optimization algorithm. *IEEE Transactions on Evolutionary Computation, 21*(3), 378–390.

Bratton, D., & Kennedy, J. (2007). Defining a standard for particle swarm optimization. In *Proceedings of the IEEE swarm intelligence symposium* (pp. 120–127). IEEE.

Cenikj, G., Lang, R. D., Engelbrecht, A. P., Doerr, C., Korošec, P., & Eftimov, T. (2022). Selector: selecting a representative benchmark suite for reproducible statistical comparison. In *Proceedings of the genetic and evolutionary computation conference. GECCO '22* (pp. 620–629). Association for Computing Machinery. https://doi.org/10.1145/3512290.3528809

Chahar, V., Katoch, S., & Chauhan, S. (2021). A review on genetic algorithm: Past, present, and future. *Multimedia Tools and Applications, 80*(5), 8091–8126. https://doi.org/10.1007/s11042-020-10139-6

Dong, C., Wang, G., Chen, Z., & Yu, Z. (2008). A method of self-adaptive inertia weight for PSO. In *Proceedings of the 2008 international conference on computer science and software engineering* (Vol. 1, pp. 1195–1198).

Engelbrecht, A. P. (2013). Roaming behavior of unconstrained particles. In *Proceedings of the BRICS congress on computational intelligence and 11th Brazilian congress on computational intelligence* (pp. 104–111). https://doi.org/10.1109/BRICS-CCI-CBIC.2013.28

Engelbrecht, A. (2022). Stability-guided particle swarm optimization. In Dorigo, M., et al. (Eds.), *Swarm intelligence. ANTS 2022*. Lecture Notes in Computer Science (Vol. 13491, pp. 360–369). Cham: Springer. https://doi.org/10.1007/978-3-031-20176-9_33

Engelbrecht, A. P. (2007). *Computational intelligence: An introduction* (2nd ed.). Wiley.

Engelbrecht, A. P. (2010). Heterogeneous particle swarm optimization. In M. Dorigo, M. Birattari, G. A. Di Caro, R. Doursat, A. P. Engelbrecht, D. Floreano, L. M. Gambardella, R. Groß, E. Şahin, H. Sayama, & T. Stützle (Eds.), *Swarm Intelligence* (pp. 191–202). Springer.

Gavana, A.: Global optimization benchmarks and AMPGO (2022). http://infinity77.net/global_optimization/genindex.html

Hansen, N., Finck, S., Ros, R., & Auger, A. (2009). Real-parameter black-box optimization benchmarking 2009: noiseless functions definitions (Research report rr-6829, INRIA, Le Chesnay-Rocquencourt, France).

Harrison, K. R., Engelbrecht, A. P., & Ombuki-Berman, B. M. (2017). An adaptive particle swarm optimization algorithm based on optimal parameter regions. In *2017 IEEE symposium series on computational intelligence (SSCI)* (pp. 1–8). https://doi.org/10.1109/SSCI.2017.8285342

Harrison, K. R., Engelbrecht, A. P., & Ombuki-Berman, B. M. (2016). The sad state of self-adaptive particle swarm optimizers. In *Proceedings of the IEEE congress on evolutionary computation* (pp. 431–439). https://doi.org/10.1109/CEC.2016.7743826

Harrison, K., Engelbrecht, A. P., & Ombuki-Berman, B. (2018). Self-adaptive particle swarm optimization: A review and analysis of convergence. *Swarm Intelligence, 12*, 187–226. https://doi.org/10.1007/s11721-017-0150-9

Harrison, K. R., Engelbrecht, A. P., & Ombuki-Berman, B. M. (2018). Optimal parameter regions and the time-dependence of control parameter values for the particle swarm optimization algorithm. *Swarm and Evolutionary Computation, 41*, 20–35. https://doi.org/10.1016/j.swevo.2018.01.006

Hashemi, A. B., & Meybodi, M. R. (2011). A note on the learning automata based algorithms for adaptive parameter selection in PSO. *Applied Soft Computing, 11*(1), 689–705. https://doi.org/10.1016/j.asoc.2009.12.030

Jalili, S., Hosseinzadeh, Y., & Rabczuk, T. (2019). In: Kulkarni, A. J., Singh, P. K., Satapathy, S. C., Husseinzadeh Kashan, A., & Tai, K. (eds.) *Simultaneous size and shape optimization of dome-shaped structures using improved cultural algorithm* (pp. 93–120). Springer. https://doi.org/10.1007/978-981-13-6569-0_5

Jalili, S., & Hosseinzadeh, Y. (2014). A cultural algorithm for optimal design of truss structures. *Latin American Journal of Solids and Structures*. https://doi.org/10.1590/1679-78251547

Jiang, M., Luo, Y., & Yang, S. (2007). Stochastic convergence analysis and parameter selection of the standard particle swarm optimization algorithm. *Information Processing Letters, 102*(1), 8–16.

Jun, S., & Jian, L. (2009). An improved self-adaptive particle swarm optimization algorithm with simulated annealing. In *Proceedings of the 3rd international symposium on intelligent information technology application* (Vol. 3, pp. 396–399). https://doi.org/10.1109/IITA.2009.476

Kennedy, J., & Eberhart, R. (1995). Particle swarm optimization. In *Proceedings of the ICNN'95—international conference on neural networks* (Vol. 4, pp. 1942–1948). https://doi.org/10.1109/ICNN.1995.488968

Lang, R. D., & Engelbrecht, A. P. (2020). Decision space coverage of random walks. In *2020 IEEE congress on evolutionary computation (CEC)* (pp. 1–8). https://doi.org/10.1109/CEC48606.2020.9185623

Lang, R. D., & Engelbrecht, A. P. (2020). Distributed random walks for fitness landscape analysis. In *Proceedings of the 2020 genetic and evolutionary computation conference. GECCO '20* (pp. 612–619). Association for Computing Machinery. https://doi.org/10.1145/3377930.3389829

Lang, R., & Engelbrecht, A. (2021). An exploratory landscape analysis-based benchmark suite. *Algorithms, 14*, 78. https://doi.org/10.3390/a14030078

Li, X., Fu, H., & Zhang, C. (2008). A self-adaptive particle swarm optimization algorithm. In *Proceedings of the 2008 international conference on computer science and software engineering* (Vol. 5, pp. 186–189). https://doi.org/10.1109/CSSE.2008.142

Liang, J., Qu, B., Suganthan, P., & Chen, Q. (2014). *Problem definitions and evaluation criteria for the CEC 2015 competition on learning-based real-parameter single objective optimization*. Technical Report 201411A. Zhengzhou: Computational Intelligence Laboratory, Zhengzhou University.

Liang, J., Qu, B., Suganthan, P., & Hernández-Díaz, A. (2013a). *Problem definitions and evaluation criteria for the CEC 2013 special session on real-parameter optimization*. Technical Report 201212. Zhengzhou: Computational Intelligence Laboratory, Zhengzhou University.

Liang, J., Qu, B., & Suganthan, P. (2013b). *Problem definitions and evaluation criteria for the CEC 2014 special session and competition on single objective real-parameter numerical optimization*. Technical Report 201311. Zhengzhou: Computational Intelligence Laboratory, Zhengzhou University.

Maheri, A., Jalili, S., Hosseinzadeh, Y., Khani, R., & Miryahyavi, M. (2021). A comprehensive survey on cultural algorithms. *Swarm and Evolutionary Computation, 62*, 100846. https://doi.org/10.1016/j.swevo.2021.100846

Nepomuceno, F. V., & Engelbrecht, A. P. (2013). A self-adaptive heterogeneous PSO for real-parameter optimization. In *2013 IEEE congress on evolutionary computation*, 361–368

Oldewage, E. T., Engelbrecht, A. P., & Cleghorn, C. W. (2017). The merits of velocity clamping particle swarm optimisation in high dimensional spaces. In *Proceedings of the IEEE symposium series on computational intelligence (SSCI)* (pp. 1–8). https://doi.org/10.1109/SSCI.2017.8280887

Olorunda, O., & Engelbrecht, A. P. (2008). Measuring exploration/exploitation in particle swarms using swarm diversity. In *2008 IEEE congress on evolutionary computation (IEEE world congress on computational intelligence)* (pp. 1128–1134). https://doi.org/10.1109/CEC.2008.4630938

Pan, Z.-L., Chen, L., & Zhang, G.-Z. (2010). Cultural algorithm for minimization of binary decision diagram and its application in crosstalk fault detection. *International Journal of Automation and Computing, 7*(1), 70. https://doi.org/10.1007/s11633-010-0070-2

Poli, R., & Broomhead, D. (2007). Exact analysis of the sampling distribution for the canonical particle swarm optimiser and its convergence during stagnation. In *Proceedings of the 9th annual conference on genetic and evolutionary computation* (pp. 134–141). Association for Computing Machinery. https://doi.org/10.1145/1276958.1276977

Poli, R. (2009). Mean and variance of the sampling distribution of particle swarm optimizers during stagnation. *IEEE Transactions on Evolutionary Computation, 13*(4), 712–721. https://doi.org/10.1109/TEVC.2008.2011744

Ratnaweera, A., Halgamuge, S. K., & Watson, H. C. (2004). Self-organizing hierarchical particle swarm optimizer with time-varying acceleration coefficients. *IEEE Transactions on Evolutionary Computation, 8*(3), 240–255. https://doi.org/10.1109/TEVC.2004.826071

Reynolds, R. G. (1994). An introduction to cultural algorithms. In Sebald, A. V., & Fogel, L. J. (eds.) *Evolutionary programming—Proceedings of the 3rd annual conference* (pp. 131–139). World Scientific Press. https://doi.org/10.1142/9789814534116 . http://ai.cs.wayne.edu/ai/availablePapersOnLine/IntroToCA.pdf

Sermpinis, G., Theofilatos, K., Karathanasopoulos, A., Georgopoulos, E. F., & Dunis, C. (2013). Forecasting foreign exchange rates with adaptive neural networks using radial-basis functions and particle swarm optimization. *European Journal of Operational Research, 225*(3), 528–540. https://doi.org/10.1016/j.ejor.2012.10.020

Shi, Y., & Eberhart, R. (1998). A modified particle swarm optimizer. In *Proceedings of the 1998 IEEE international conference on evolutionary computation* (Vol. 6, pp. 69–73). https://doi.org/10.1109/ICEC.1998.699146

Shi, Y., & Eberhart, R. (1998). A modified particle swarm optimizer. In *Proceedings of the IEEE international conference on evolutionary computation* (pp. 69–73). https://doi.org/10.1109/ICEC.1998.699146

Tanweer, M. R., Suresh, S., & Sundararajan, N. (2015). Self regulating particle swarm optimization algorithm. *Information Sciences,294*, 182–202. https://doi.org/10.1016/j.ins.2014.09.053. Innovative Applications of Artificial Neural Networks in Engineering

Wu, G., Mallipeddi, R., & Suganthan, P. (2016). *Problem definitions and evaluation criteria for the CEC 2017 competition and special session on constrained single objective real-parameter optimization*. Technical Report. Singapore: Nanyang Technological University.

Xue, Y., Zhang, Q., & Neri, F. (2021). Self-adaptive particle swarm optimization-based echo state network for time series prediction. *International Journal of Neural Systems, 31*(12), 2150057. https://doi.org/10.1142/S012906572150057X

Zhan, Z.-H., Zhang, J., Li, Y., & Chung, H.S.-H. (2009). Adaptive particle swarm optimization. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics), 39*(6), 1362–1381. https://doi.org/10.1109/TSMCB.2009.2015956