A Dimension-Wise Particle Swarm Optimization Algorithm Optimized via Self-Tuning

Justin Schlauwitz*
Petr Musilek*†
*Department of Electrical and Computer Engineering, University of Alberta, Edmonton AB, Canada
†Department of Cybernetics, Faculty of Science, University of Hradec Králové, Czech Republic

***Abstract*—This article proposes an improvement to the traditional Particle Swarm Optimization (PSO) via modifications w.r.t. how particles move and are attracted to optimal positions. The performance is evaluated based on how well the algorithm is able to perform w.r.t. finding the global maxima of the Sinc, Marr-Wavelet, and Drop-Wave functions in multi-dimensional problem spaces. Each algorithm is put through a session of self-tuning with a sufficient number of iterations to ensure convergence so as to demonstrate that the evaluation of each algorithm is done with justified optimal parameters.**

***Index Terms*– Particle swarm optimization.**

I. Introduction

Particle Swarm Optimization (PSO) algorithms have a reputation of being highly capable of optimizing complex multi-dimensional problems. PSO relies on a weighted combination of momentum, local/personal best attraction, and global best attraction to find a position which yields the best score, *i.e.* global maxima/minima. Though PSO tends to be slightly slower than Differential Evolution (DE), it can be expected to have similar or even slightly better convergence rates depending on the problem [1], [2]. PSO algorithms also tend to me much faster and more efficient than Genetic Algorithms (GAs), giving us some leeway to make some small trade-offs between complexity and convergence reliability without compromising its perk of being a fast algorithm [3], [4].

A secondary component of this article is the use of self-tuning to optimize the parameters of a given optimization method [5]. Self-Tuning relies on the fundamental expectation that the optimization algorithm itself can satisfy two conditions, *i.e.* its parameters can be optimized and it has the capacity to pursue locations with better results. In this way the algorithm should be able to demonstrate that, within its capacity as an optimization method, its own parameters can be optimized to the point of convergence using its own methods either internally or externally. Alternatively, if an algorithm cannot converge when self-tuning, the final choice of parameters will likely not be much better or worse than any other combination. In difference to some self-tuning methods, the external approach we intend to use does not require specialized modifications or assigned rules to adjust the parameters [6], [7]; instead, it relies on the concept of bootstrapping, *i.e.* an incremental improvement based on a prior assumption, as a means of self-justification [8]. This approach only serves to get the best performance out of a given algorithm, but does not tell us how well it performs relative to other algorithms. To evaluate how the modifications influence performance, we will also test the traditional method in the same way and compare the results. To verify the degree to which the self-tuning process has served its purpose, we will also perform a sensitivity sweep of each PSO parameter.

II. Traditional PSO

The general structure of PSO remains largely unchanged over the years and is still a preferred method of parameter optimization. In a traditional PSO (PSO^{Trad}) algorithm, each individual possesses velocity and position updates [9]:

$$\vec{v}_{t+1} = m \cdot \vec{v}_t + c_1 \cdot b_1 \cdot (\vec{p}_{global} - \vec{p}_t) + c_2 \cdot b_2 \cdot (\vec{p}_{local} - \vec{p}_t), \quad (1)$$

and

$$\vec{p}_{t+1} = \vec{p}_t + \vec{v}_{t+1}, \quad (2)$$

respectively. Here, m is the momentum constant, c_1 and c_2 are the global and local attraction strengths, and b_1 and b_2 are generated using $rand(0,1)$ which determines the degree of attraction to the respective best value at a given iteration. It should be noted that $rand(0,1)$ is a uniform random number in the range $[0,1)$. The local best position is the individual's recorded best optimal value such that, in the event of maximizing the reward r_t:

$$\vec{p}_{local} = \begin{cases} \vec{p}_t & \text{If } r_t > r_{local}, \\ \vec{p}_{local} & \text{Otherwise,} \end{cases} \quad (3)$$

and

$$r_{local} = \begin{cases} r_t & \text{If } r_t > r_{local}, \\ r_{local} & \text{Otherwise.} \end{cases} \quad (4)$$

This comparison can also be made for the global best by evaluating across each individual i of the current iteration such that:

$$\vec{p}_{global} = \begin{cases} \vec{p}_{local,i} & \text{If } r_{local,i} > r_{global}, \\ \vec{p}_{global} & \text{Otherwise,} \end{cases} \quad (5)$$

and

$$r_{global} = \begin{cases} r_{local,i} & \text{If } r_{local,i} > r_{global}, \\ r_{global} & \text{Otherwise.} \end{cases} \quad (6)$$

The reward r_t is an evaluation of fitness w.r.t. the parameters when they are applied to the problem of interest. If the reward

978-1-7281-6929-3/20/$31.00 ©2020 IEEE

is a function of the parameters, it can also be interpreted as $r_{\vec{p}}$. If a problem can be executed relatively quickly, one may choose to update the reward for the position $\vec{p}$ every time it is encountered. Otherwise, if the problem requires a relatively large amount of time to process each step and the result is constant for a given position, it may be preferred to log the data in a table so that it can be recalled, saving processing time at the expense of memory. If the result is stochastic in nature, it may be preferred to use a mean reward $\bar{r}_{\vec{p}}$ or something similar over a statistically significant number of executions on $\vec{p}$.

III. Dimension-Wise PSO

The dimension-wise PSO (PSO^{Mod}) algorithm makes several changes to the traditional form. The first set of changes are w.r.t. the velocity update:

$$\begin{aligned}\vec{v}_{t+1} = c_0 \cdot \overrightarrow{rand}(-1,1) \cdot |\vec{v}_t| + m \cdot \vec{v}_t \\ + c_1 \cdot \vec{b}_1 \cdot (\vec{p}_{global} - \vec{p}_t) \\ + c_2 \cdot \vec{b}_2 \cdot (\vec{p}_{local} - \vec{p}_t)\end{aligned} \quad (7)$$

where velocity is also subject to the dimension-wise limitations:

$$v_{t+1} = \begin{cases} v_{lim} & \text{If } v_{lim} < v_{t+1}, \\ -v_{lim} & \text{If } -v_{lim} > v_{t+1}, \\ v_{t+1} & \text{Otherwise,} \end{cases} \quad (8)$$

and

$$v_{t+1} = \begin{cases} v_{t+1} & \text{If } p_{max} > p_{t+1} > p_{min}, \\ -v_{t+1} & \text{Otherwise,} \end{cases} \quad (9)$$

given the position update in equation (2). This approach allows particles to bounce off the walls defined by the limits of the range for each parameter without killing the individual's speed. To complement the velocity restrictions, we also apply a hard boundary to the position in the form:

$$p_{t+1} = \begin{cases} p_{t+1} & \text{If } p_{max} > p_{t+1} > p_{min}, \\ p_{min} & \text{If } p_{t+1} \leq p_{min}, \\ p_{max} & \text{If } p_{max} \leq p_{t+1}. \end{cases} \quad (10)$$

Returning to equation (7), c_0 is a new parameter for the percent noise injection. In equation (8), v_{lim} is the restriction of velocity w.r.t. a given dimension; and for equations (9) and (10), p_{max} and p_{min} are the position boundaries bracketing the valid range of exploration along a given dimension. Another notable change is that b_1 and b_2 in equation (7) are randomly activated binary values which determine if the individual should attempt to move toward the respective best position of a given dimension. The activation of b_1 can be described as:

$$b_1 = \begin{cases} 1 & \text{If } rand(0,1) < \tau_1, \\ 0 & \text{Otherwise,} \end{cases} \quad (11)$$

where $rand(0,1)$ is a uniform random number in the range $[0,1)$, and τ_1 denotes the probability of activation. This equation can also applied to b_2 using τ_2. It must be stressed

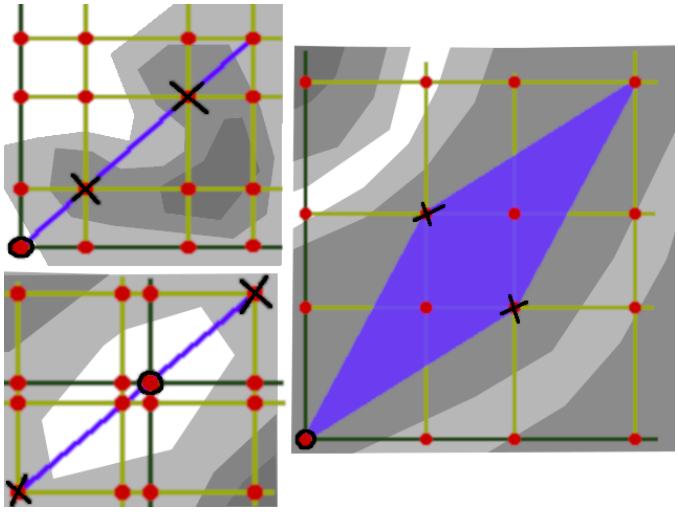


Fig. 1: Exploitation differs between PSO^{Trad} (the purple envelope) and PSO^{Mod} (the red points on the grid).

that b_1 and b_2 are evaluated on a per-dimension basis. Though this reduces the coverage of points between the two known best locations, it increases the exploration of points in the surrounding neighborhood in-line with each dimension (See Figure 1). When momentum is accounted for, though it may take a round about course to the target, it is less likely to become fixated on a single vector. The probability of moving to a given point is based on the products of τ_1, τ_2, $1-\tau_1$, and/or $1-\tau_2$ for each attractor and dimension. For the right-most example in Figure 1, if the lower 'x' marks the global optimum, the point in the top left corner would have a transition probability of:

$$P(tl) = (1-\tau_1^{hor}) \cdot (1-\tau_2^{hor}) \cdot \tau_1^{vir} \cdot \tau_2^{vir}, \quad (12)$$

while the probability of transitioning to the global optimum's mark is:

$$P(glob) = \tau_1^{hor} \cdot (1-\tau_2^{hor}) \cdot \tau_1^{vir} \cdot (1-\tau_2^{vir}). \quad (13)$$

As with PSO^{Trad}, when c_1 and c_2 are reduced, the scale of movement, *i.e.* the grid size, is reduced while the position the particle would move to without the attractors is the anchor point, determined by the noise injection and momentum.

Percent noise injection is used as a precaution to improve variations in movement and increase the likelihood of 'hopping' out of local minima without causing an unreasonably large change in course direction. In opposition to the effect of percent noise injection, the velocity limit attempts to restrict overshooting caused by an excessive build up of momentum and attraction. The reason for allowing particles to bounce off the upper and lower limits is that doing so maintains a degree of activity, preventing early termination of exploration. Additionally, given that the position is stopped at the respective limit, a measure of performance at said limit will still be obtained.

Regarding the velocity limit, to prevent an excessive expansion in required search parameters, v_{lim} will be simplified to:

$$v_{lim} = v_{lim}^{\%} \cdot (p_{max} - p_{min}), \tag{14}$$

where $v_{lim}^{\%}$ is the percentage of a given dimensions span, allowing all PSO parameters to be in the range $[0, 1]$ and reducing the number of velocity limits that must be defined to one. It is worth noting that setting $c_0 = 0$ and $v_{lim}^{\%} = 1$ gives us PSO^{Mod} in its closest possible form to PSO^{Trad}, where the key differences are the attraction random values.

IV. Methodology

For our tests, it is assumed that the problem will be expensive to re-process and the results will be stochastic in nature. This prevents us from being able to use the returned values of a single sample outright and must rely on a suitable representation of the distribution. The method of choice is a mean of the reward less its standard deviation which we will refer to as the score, *i.e.*:

$$r_{\vec{p}} = r_{mean} - r_{std}. \tag{15}$$

The use of this score is acceptable, subject to the condition that the mean and standard deviation are taken across a statistically significant number of runs on $\vec{p}$. If the score is regarded as statistically significant, it is also reasonable to expect it to hold for future executions of the same parameter set. To save time, we can log the score and use it for occasions where the re-processing of $\vec{p}$ is requested. This particular method of reward calculation is preferred because it places equal importance on mean performance and reliability.

We have defined a set of rules, demonstrated in Figure 2, to be applied on both algorithms. Firstly, occupation of a location is on a first-come-first-served basis, such that the colliding particle's position will be randomly re-initialized and checked for collisions, *i.e.* warped to a new location. This rule increases the potential for exploration in the event that two particles are found to occupy the same region. Secondly, if a particle's velocity is not sufficient to allow it to move to a new position based on the log's numerical resolution (4 decimal points in our case), the particle will be warped to a new location. Thirdly, warping will be repeated until the particle occupies an unlogged/unoccupied location or until the warp counter exceeds 3. If a particle's warp counter exceeds the specified number, it is forced to remain at its new position, where it is expected to use the logged value. In the unlikely event that an individual re-processes a given location, the logged values are updated and an associated counter for the number of visits is incremented. Additionally, warping largely serves the purpose of random exploration, maintaining a given particle's activity after it has converged to a local or global maxima. The use of a try-except was also implemented to deal with parameter combinations that caused the problem code to crash, *e.g.*value overflow and DIV0 errors. In the event that a particle's chosen parameters cause the problem to fail, $r_{\vec{p}}$ is assigned a large negative value, *e.g.*$-\inf$, before moving to the next location.

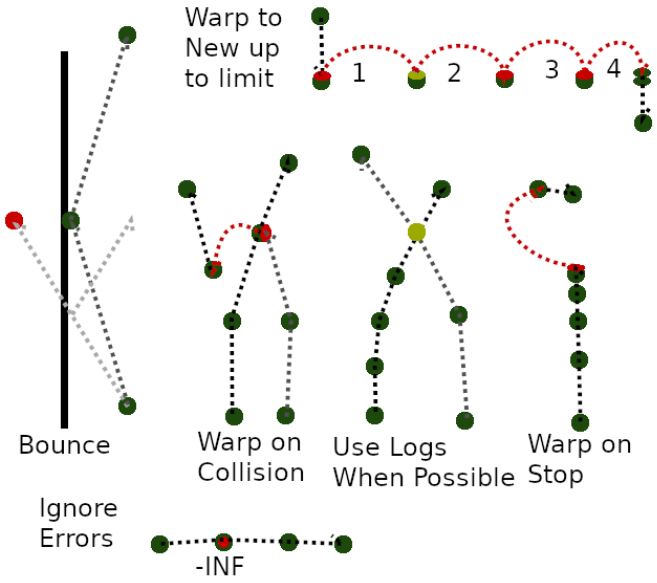


Fig. 2: Rules and restrictions applied to both PSO methods.

The established rules place emphasis on time efficiency and statistical significance for large and potentially stochastic problems, but can also be applied to smaller deterministic problems such as the test functions we will be evaluating on. As is the case with the PSO parameters, our implementation expects that the parameter space to be explored is bounded and continuous. The usage of the same evaluation criteria is also expected, therefore, equation (15) will be used where possible as a measure of fitness.

A. Function Problems

For the first problem, we would like to use the multi-dimensional Sinc function calculated as:

$$f(\vec{p}, \vec{y}, \vec{z}) = mean\left(\frac{sin((\vec{p} - \vec{y}) \cdot \vec{z})}{(\vec{p} - \vec{y}) \cdot \vec{z}}\right), \tag{16}$$

where all dimensions of $\vec{p}$ are limited to the range $[-1, 1]$, $\vec{y}$ is a randomly initialized to a value within the range $[-1, 1]$ and maintained for the duration of the PSO process, and $\vec{z}$ is a constant vector of linearly increasing values (starting at π and ending at 20π). This problem was chosen because it has a clear, graphically identifiable global maximum and a large number of local maxima. In the event that element z is large, we come across regions where there is minimal change in the reward value, *i.e.* a relatively flat problem space with a very small window for the global maximum.

To increase the credibility of our results, the global maximum $\vec{y}$ is randomly initialized at the start of PSO$_2$ to reduce the likelihood of certain speed related parameters being tuned to favor a particular problem configuration.[1] The linear distribution of $\vec{z}$ is used to provide a progressive transition

[1]For clarity, we will refer to the PSO code being optimized while optimizing the given function problem as PSO$_2$ and the PSO code that optimizes PSO$_2$ as PSO$_1$.

in difficulty. As z becomes larger, the attenuation of the Sinc function's features become more severe, increasing the risk of overshooting the global maximum location. Given that an optimization algorithm with a notably larger immunity to parametric range sensitivity is preferred, it is expected that the algorithm which is able to achieve the highest score would be better. As an additional method of evaluation, we will also check if a solution is within the central node of each dimension using:

$$g(p, y, z) = \begin{cases} 1 & \text{If } \frac{sin((p-y)\cdot z)}{(p-y)\cdot z} > 0.2. \\ 0 & \text{Otherwise.} \end{cases} \quad (17)$$

This method of evaluation only claims the point to be within the global maxima iff the score for a given dimension is decidedly better than what could be achieved at any other local maxima.[2] It should be noted that, since PSO is not a gradient based method, we do not need to be concerned about the depth of the local maxima and minima. Instead, attention must be given to the effective width of neighboring regions, *i.e.* the area for which $f(\vec{p}, \vec{y}, \vec{z}) \leq f(\vec{p}_{local}, \vec{y}, \vec{z})$ will not affect the particle's movement while the area for which $f(\vec{p}, \vec{y}, \vec{z}) > f(\vec{p}_{local}, \vec{y}, \vec{z})$ will be guaranteed to have some influence once encountered. This means that, w.r.t. the Sinc function problem, all regions along a given dimension for which $|p_t - y| > |p_{local} - y|$ is true, the particle's local values will remain unchanged while small movements toward y can only show improvements if the particle lands in another local maxima. It must be noted that, should one dimension show sufficient improvement, p_{local} will be changed even if $|p_t - y| > |p_{local} - y|$ of a given dimension holds true.

To be more thorough, we will also test both PSO methods on the Marr Wavelet (MW) and Drop-Wave (DW) functions with 6 individuals, 3 dimensions, and a range of $[-6, 6]$ per dimension [10]. The MW function will use the spread value $\sigma = 0.5$, and for both the MW and DW, the range of values for the offset $\vec{y}$ will be adjusted such that it can appear anywhere within the explorable search space. The evaluation will also be conducted on a test problem that uses a ratio of $3:3:3$ for the MW, DW, and Sinc function (for $\vec{z}$ starting at 10.5π and ending at 20π), *i.e.* a Mixed (MX) function where the output r_t is the average of its sub-functions. This mixed approach is expected to encourage the PSO to find a good compromise for optimal parameters targeting each individual problem set. Though the result of tuning on the mixed problem may not result in the fastest convergence on any individual problem, it should still be able allow the PSO to perform better than if it was tuned on a largely different problem space, *e.g.* tuning on the Sinc function and expecting it to perform well on the others. It should be noted that each function output is scaled such that the global optimum is gives a reward of $+1$.

[2]The value of 0.2 was chosen as it is a sufficiently close approximation of the local maxima found near 2.5π while also excluding the local peak value.

B. Self-Tuning

Self-tuning relies on the PSO's ability to find its own best parameters. Naturally, to make this possible, a problem or set of problems with a sufficient degree of dynamic difficulty and flexibility are required to get reasonable measures of performance while minimizing potential biases towards particular problem configurations. Due to the stochastic initialization process of the offset $\vec{y}$, it is necessary to obtain a performance distribution over a notable number of initialized vectors $\vec{y}$. For our tests, the reward received by PSO_1 will be based on the distribution of 500 samples, *i.e.* 500 separate executions of PSO_2 on different initializations of a given test problem. It should be noted that, as the test functions optimized in PSO_2 are not stochastic, we only need to calculate the value returned by a position once. The key point to be aware of is that the initially passed in PSO_1 parameters, determined through a series of educated guesses or prior executions, are pushed to the first PSO_2 individual on the first iteration to test the performance of said initial values, *i.e.* we seed the first individual. After each iteration, the global best PSO_2 parameter combination is assigned to PSO_1 before moving on. In this way, self-tuning is comparable to bootstrapping as the learning/optimization occurs progressively.

V. Testing

For the initialization, we used the parameter combinations:

- Percent noise: $c_0 = 0.02$ and $c_0 = 0.0$,
- Momentum: $m = 0.7$,
- Global attraction: $c_1 = 0.5$,
- Local attraction: $c_2 = 0.5$,
- Global probability: $\tau_1 = 0.3$ and $\tau_1 = N/A$,
- Local probability: $\tau_2 = 0.2$ and $\tau_2 = N/A$,
- Speed limit: $v_{lim}^{\%} = 0.2$ and $v_{lim}^{\%} = 1.0$,
- Number of individuals: 6, and
- Number of iterations for PSO_2: 50

for the dimension-wise and traditional PSO_2 algorithms respectively. The Sinc problem dimensions were tested with sizes 3, 13, and 49 to demonstrate varying degrees of population sparseness relative to the number of dimensions. It should be noted that increasing the number of dimensions does not change the range of difficulty of finding the global maxima for each dimension of the Sinc function, but increases the burden of dimension-wise exploration for each individual. To improve the potential for convergence, PSO_1 uses 20 individuals and 100 iterations.

VI. Results

After self-tuning, the optimal PSO parameters and their scores were gathered (see Table I), an evaluation of per-dimension performance was conducted on the Sinc function (see Tables II and III, Figure 3, and Figure 4), and parameter sensitivity sweeps were done to evaluate how well the methods converged on each problem. The score improvement from the initial parameters to the final optimized parameters show that there is a notable improvement in performance for both forms

TABLE I: Global best PSO parameters (X_{dim}^{ind}).

	m	c_0	c_1	τ_1	c_2	τ_2	$v_{lim}^{\%}$
Trad_3^6	0.847347	0.0	0.625448	–	0.283466	–	1.0
Mod_3^6	0.646900	0.886046	0.993560	0.297974	0.986679	1.0	0.826609
Trad_{13}^6	0.792969	0.0	0.993248	–	0.807046	–	1.0
Mod_{13}^6	0.350952	0.425527	0.999966	0.214726	0.995807	1.0	0.977800
Trad_{13}^{26}	0.784900	0.0	1.0	–	0.737775	–	1.0
Mod_{13}^{26}	0.205239	0.277784	0.999824	0.175739	0.999022	0.990494	1.0
Trad_{49}^6	0.777349	0.0	0.956472	–	0.473307	–	1.0
Mod_{49}^6	0.158574	0.307773	0.999532	0.239501	0.996855	0.999393	0.944697
Trad_{3DW}^6	0.828378	0.0	0.619558	–	0.134687	–	1.0
Mod_{3DW}^6	0.994308	0.132850	0.929807	0.739896	0.876021	0.525148	0.070311
Trad_{3MW}^6	0.417640	0.0	1.0	–	0.280108	–	1.0
Mod_{3MW}^6	1.0	0.0	0.936990	0.760820	0.526155	0.0	0.899560
Trad_{9MX}^6	0.823595	0.0	0.497509	–	0.493242	–	1.0
Mod_{9MX}^6	0.537582	0.768016	0.999995	0.629405	0.999383	0.872157	0.418376

TABLE II: Scores calculated with equation (15) for different numbers of dimensions with the same population size.

dimensions=	3	13	49
$\text{Trad}_{t=0,i=0}$	0.621713	0.336391	0.204468
$\text{Mod}_{t=0,i=0}$	0.655652	**0.418092**	**0.248847**
$\text{Trad}_{t=100}$	**0.692226**	0.391568	0.232197
$\text{Mod}_{t=100}$	**0.906178**	**0.596545**	**0.366413**

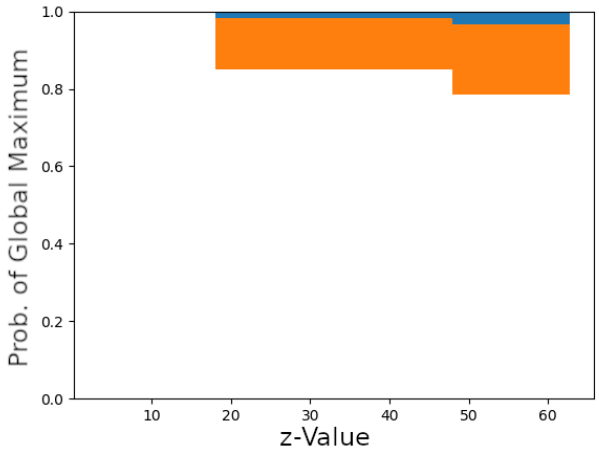
TABLE III: Select dimension specific mean performances.

	$\text{Trad}_{t=100}$			$\text{Mod}_{t=100}$		
$z=$	π	10.5π	20π	π	10.5π	20π
3	**100**%	86.8%	71.4%	**100**%	**98.2**%	**96.6**%
13	**98.0**%	42.8%	32.4%	**99.8**%	70.2%	48.0%
49	89.2%	24.8%	17.6%	**94.0**%	41.2%	31.4%

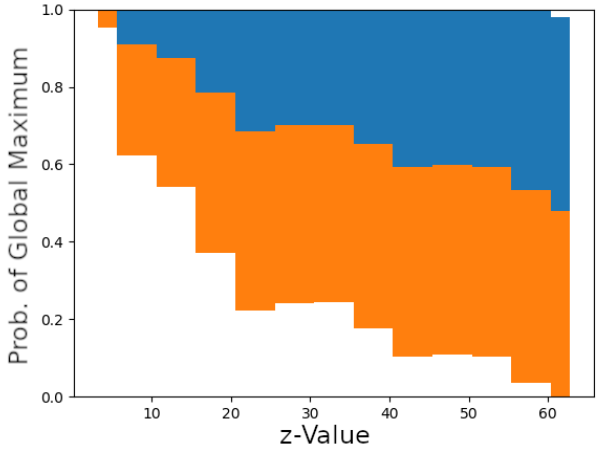
of PSO, however, regardless of the number of dimensions in the Sinc problem, PSO^{Mod} was found to perform better than the equivalently tuned PSO^{Trad} (see Table II). From the sensitivity sweeps, it was found that PSO^{Mod} and PSO^{Trad} are comparable in performance for lower dimensions, but PSO^{Mod}'s rate of performance loss for each added dimension was lower, even to the point where PSO^{Mod}'s tuned sensitivity results became statistically better, *i.e.* PSO^{Mod}'s $(r_{mean} - r_{std})$ was higher than PSO^{Trad}'s $(r_{mean} + r_{std})$.

The Sinc function proved to be, for the most part, the easiest of the three problems to solve; likely because each dimension can be solved independently. The gathered results affirm that, regardless of the optimization method, an increase in the number of searchable dimensions will have an exponential decay in performance. The values in Table III show the performance drop for select dimensions that have different degrees of sensitivity to the input, *i.e.* specific z values. The comparison of probability distributions for progressively more sensitive dimensions also show that the ability to find the global maximum or its region when z is large will be hindered more severely than it is for smaller values when the number of dimensions increases (see Figures 3 and 4). The probability of finding the region of the global maximum is the intersection of the two regions while the length of regions above/below are the plus/minus standard deviation limited to the range of $[0, 1]$. It must be emphasized that these probabilities are w.r.t. the boolean measure of being within the region of the global maximum and not the measure of how close the solution came to the exact value. The standard deviation similarly suggests the deviation in probability readings for falling within the dimension's region of global optimum based on the results of our experiment. Based on the plotted data, we expect that, for any improvement in finding the region of the global maximum in one dimension, it would come at the expense of another. The drop in probability across different z values not being smooth for problems with a larger number of dimensions, having occasional spikes and drops despite being averaged over 500 PSO_2 results, supports this speculation. However, the degree to which each influenced dimension is hindered would likely be subject to the relative difficulty of improved dimension. We also ran a test with 13 dimensions and 26 particles on PSO_2 which resulted in a score of 0.524360 for PSO^{Trad} and 0.872302 for PSO^{Mod}. The select z value's probabilities, equivalent to what is mentioned in Table III, were 99.8%, 57.2%, and 39.4% for PSO^{Trad} and 100%, 94.4%, and 85.4% for PSO^{Mod}. These values suggest that identical dimension-population ratios for different numbers of dimensions on similar problems will yield slightly lower performance values as the number of dimensions and their sensitivities have a greater influence on the PSO's performance than its population size; however, this is more predominant in PSO^{Trad} than in PSO^{Mod}.

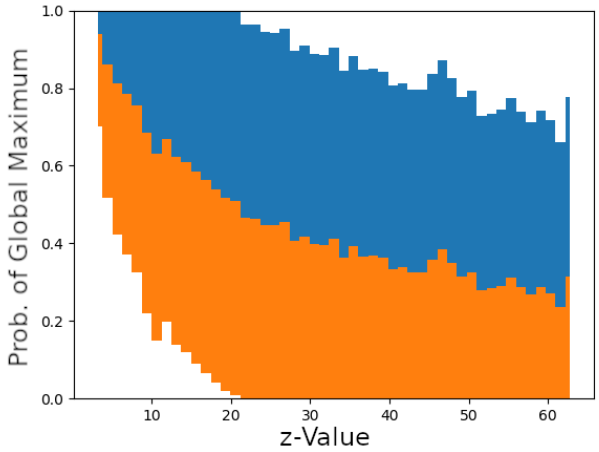
The DW function is a multi-modal problem, similar to the Sinc function, but with a score that depends on the simultaneous optimization of all parameters to achieve the



(a) 3 dimensions

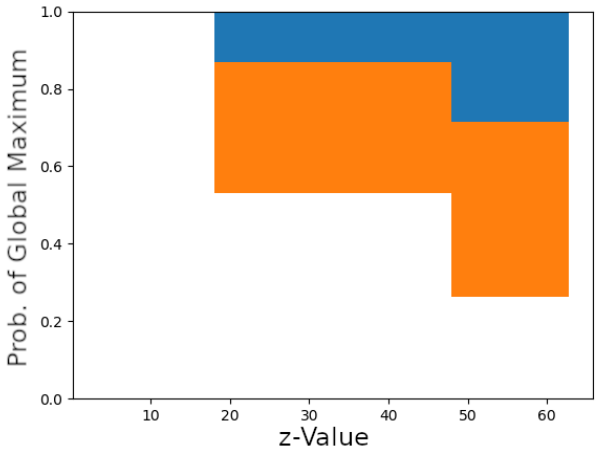


(b) 13 dimensions

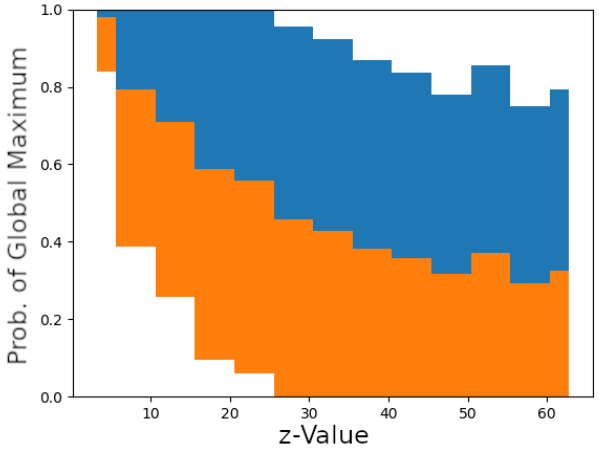


(c) 49 dimensions

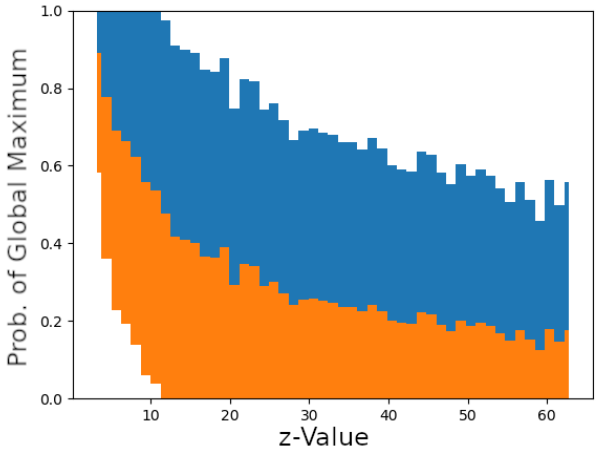
Fig. 3: PSO^{Mod} scaling capacity with 6 particles.



(a) 3 dimensions



(b) 13 dimensions



(c) 49 dimensions

Fig. 4: PSO^{Trad} scaling capacity with 6 particles.

most positive value of $+1$. For PSO^{Trad}, the initial score was 0.826949 while the tuned method gave a score of 0.832991, achieving the global optimum 47.7% of the time over 500 tests. For PSO^{Mod}, the initial score was 0.817345, with a score of 0.921444 and a 55.0% occurrence of achieving the global maximum after tuning.

The MW function is a unimodal problem for which the global optimum is surrounded by a small trench which is further surrounded by a relatively flat monotonic plane. The MW is similar to the DW function as they both require simultaneous optimization of all dimensions to achieve the global maximum of $+1$, but there is no indication that a particle is approaching the global maxima until it is already within the region that would indicate thus, *i.e.* when the reward is greater than zero. For PSO^{Trad}, the resulting score was -0.107540 with a 3.82% likelihood falling into the region of the global optimum and an initial score of -0.154830. For PSO^{Mod}, the resulting score was -0.044672 with an 8.21% likelihood of falling into the region of the global optimum and an initial score of -0.056891. A discovery with the sensitivity sweep of $v_{lim}^{\%}$ on the MW was that the best value was actually at approximately $v_{lim}^{\%} = 0.5$, which would have boosted the score to roughly 0.2. The expected reason for PSO^{Mod} being unable to find this optimum is that the sensitivity curve ended up resembling the MW.[3] This situation only came up with PSO^{Mod}, but even without achieving its optimal value, it was able to out-perform PSO^{Trad}. Unfortunately, this finding also demonstrates that there are some problems to which, even if they are used in the self-tuning process, will still prove difficult to achieve the best PSO parameter combination for its optimization.

The Mixed problem is more difficult overall as the PSO must balance its parameters to best solve all of the aforementioned problems simultaneously. To a degree, it was expected that the optimized parameters will largely favor the easiest sub-problem before attempting to improve performance on the progressively more difficult sub-problems; however, the results suggest that both methods balanced their parameters or even focused more heavily on the more difficult problem spaces. For PSO^{Trad}, the resulting score was 0.274131 with a likelihood falling into the region of the global optimum being 5.31%, 24.94%, and 16.20% for the MW, DW, and Sinc functions respectively; its initial score was 0.271496. For PSO^{Mod}, the resulting score was 0.327322 with a 6.19%, 41.95%, and 27.66% likelihood of falling into the respective region of the global optimum; its initial score was 0.285483. Ironically, PSO^{Trad} performed better on the MW after it was optimized on the mixed problem in comparison to when it was solely optimized on the MW. Both algorithms suffered a drop in performance on the Sinc and DW functions, however, PSO^{Mod}'s drop in performance was notably smaller.

In almost all cases, the algorithms were able to find a set of optimal parameters for themselves where small changes in each parameter would only result in equal or worse performance. The degree of sensitivity w.r.t. parameter selection for PSO^{Mod} and PSO^{Trad}, *i.e.* the drop in performance, was largely similar when compared on the same problem. That said, the sensitivity was largely dependent on the problem's difficulty, *e.g.* the sensitivity on the MW function, a very difficult optimization problem, was relatively flat compared to the others. As both algorithms tended to converge before 50 iterations of 100, and given that the results are averaged over 500 samples per PSO_2 individual, we can be confident that the results we found would be reliable/repeatable.

[3]This was also tested by starting with the parameter values specified in Section V and found to achieve similar results.

VII. Conclusions and Future Work

The parameter sweeps for each problem revealed notably different degrees of sensitivity and preference for optimal parameters, affirming that, whenever possible, we should tune on the problem the PSO is tasked to resolve. This is because PSO relies on the exploitation of features within the problem space to accelerate optimization. That said, tuning on the problem of interest cannot guarantee the true optimal combination will be found, as was the case with the MW function [5]. If one is to insist on having the optimal parameters, your best bet on a complex surface is exhaustive search. Tuning on a generalized problem, such as the mixed problem, and applying the resulting parameters to a dedicated problem can be expected to be faster when the problem of interest is large. Doing so will likely still yield better results than an educated guess; however, one must be aware of the fact that the convergence speed will likely be slower than tuning on the dedicated problem. This suggests that we should seek a compromise between efficacy and optimality. A tuning problem that is reasonably similar to the one of interest while being below a specified level of computational demand should be selected for self-tuning.

An alternative approach could be to tune the optimizer on a set of relatively less taxing problems that progressively approach true problem or a set of sub-problems that emulate different aspects of the larger problem [11]. Unfortunately, the problem space's features are rarely known, but if some key features are vaguely known, it may be sufficient. One could also have the PSO tune itself in parallel with its attempt to optimize the target problem, but consideration must be made for extra processing which gives minimal-to-no-improvement in convergence. The use of rule sets to determine the parameter value can also be expected to have similar issues in regards to rule configurations being somewhat problem dependent [6], [7]. A major convenience of the self-tuning approach used in this article is that it can be applied to any optimization algorithm without notable modifications; however, as mentioned earlier, there are a number of other alternatives which may be preferred depending on the circumstances.

Regarding the performance of the tuned PSO^{Mod}, we found that it performed notably better than PSO^{Trad}. Though it had more parameters and required processing the local and global attractions for each dimension separately, it was more resilient to increases in the number of dimensions and more capable w.r.t. finding the global optimum. For future work, a more in

depth analysis of how the chosen rules and their variations can affect performance should be conducted. It may also be worth comparing the dimension-wise PSO with DE since DE tends to have a similar capacity for finding the global optimum as PSO but with less computational demand.

REFERENCES

[1] C. K and N. Ramana, "Performance comparison of ga, de, pso and sa approaches in enhancement of total transfer capability using facts devices," *Journal of Electrical Engineering and Technology*, vol. 7, 07 2012.

[2] A. Deb, J. S. Roy, and B. Gupta, "Performance comparison of differential evolution, particle swarm optimization and genetic algorithm in the design of circularly polarized microstrip antennas," *IEEE Transactions on Antennas and Propagation*, vol. 62, no. 8, pp. 3920–3928, Aug 2014.

[3] A. Khosla, S. Kumar, and K. R. Ghosh, "A comparison of computational efforts between particle swarm optimization and genetic algorithm for identification of fuzzy models," in *NAFIPS 2007 - 2007 Annual Meeting of the North American Fuzzy Information Processing Society*, June 2007, pp. 245–250.

[4] X. Xu, W. Hu, D. Cao, Q. Huang, C. Chen, and Z. Chen, "Optimized sizing of a standalone pv-wind-hydropower station with pumped-storage installation hybrid energy system," *Renewable Energy*, vol. 147, pp. 1418 – 1431, 2020. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0960148119314375

[5] M. Meissner, M. Schmuker, and G. Schneider, "Optimized particle swarm optimization (opso) and its application to artificial neural network training," *BMC Bioinformatics*, vol. 7, no. 1, p. 125, Mar 2006. [Online]. Available: https://doi.org/10.1186/1471-2105-7-125

[6] M. S. Nobile, P. Cazzaniga, D. Besozzi, R. Colombo, G. Mauri, and G. Pasi, "Fuzzy self-tuning pso: A settings-free algorithm for global optimization," *Swarm and Evolutionary Computation*, vol. 39, pp. 70 – 85, 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S2210650216303534

[7] M. S. Nobile, G. Pasi, P. Cazzaniga, D. Besozzi, R. Colombo, and G. Mauri, "Proactive particles in swarm optimization: A self-tuning algorithm based on fuzzy logic," in *2015 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, Aug 2015, pp. 1–8.

[8] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, ser. Computational neuroscience series. Cambridge (Mass.): MIT Press London, 2013.

[9] S. M. H. Mousakazemi, "Computational effort comparison of genetic algorithm and particle swarm optimization algorithms for the proportionalintegralderivative controller tuning of a pressurized water nuclear reactor," *Annals of Nuclear Energy*, vol. 136, p. 107019, 2020. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0306454919305213

[10] M. N. Ab Wahab, S. Nefti-Meziani, and A. Atyabi, "A comprehensive review of swarm optimization algorithms," *PLOS ONE*, vol. 10, no. 5, pp. 1–36, 05 2015. [Online]. Available: https://doi.org/10.1371/journal.pone.0122827

[11] L. Hardesty. (2015, 1) Optimizing optimization algorithms. [Online]. Available: http://news.mit.edu/2015/optimizing-optimization-algorithms-0121